Upravljanje promjenjivom signalizacijom graničnih prijelaza

Grmovšek, Kristijan

Sveučilište
Sjever

Završni rad br. 427/EL/2018.

Upravljanje promjenjivom signalizacijom graničnih prijelaza

Kristijan Grmovšek, 0021/336

Varaždin, listopad 2018. godine

Sveučilište
Sjever

Odjel za elektrotehniku

Završni rad br. 427/EL/2018

Upravljanje promjenjivom signalizacijom graničnih prijelaza

Student

Kristijan Grmovšek, 0021/336

Mentor

mr.sc. Ivan Šumiga, dipl.ing.

Varaždin, listopad 2018. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL: Odjel za elektrotehniku

PRISTUPNIK: Kristijan Grmovšek | MATIČNI BROJ: 0021/336

DATUM: 19.07.2018. | KOLEGIJ: Digitalna elektronika

NASLOV RADA: Upravljanje promjenjivom signalizacijom graničnih prijelaza

NASLOV RADA NA ENGL. JEZIKU: Control of variable signaling of border crossing

MENTOR: mr.sc. Ivan šumiga, dipl.ing. | ZVANJE: Viši predavač

ČLANOVI POVJERENSTVA:

1. mr.sc. Matija Mikac, dipl.ing., viši predavač
2. Miroslav Horvatić, dipl.ing., predavač
3. mr.sc. Ivan Šumiga, dipl.ing., viši predavač
4. Stanko Vincek, struč.spec.ing.el., predavač (zamjenski član)
5.

Zadatak završnog rada

BROJ: 427/EL/2018

OPIS:
Na graničnim prijelazima se ugrađuju znakovi sa promjenjivom signalizacijom koja usmjerava promet za određena vozila na određene prometne trake. Upravljanje signalizacijom se obavlja pomoću programa nadređenog računala serijski povezanog s mikroračunalom koje upravlja znakovima.

U radu je potrebno:
-opisati sustav upravljanja promjenjivom signalizacijom graničnih prijelaza,
-opisati sklopovske i programske komponente konkretnog sustava,
-navesti prednosti korištenja tako koncipiranog sustava.

ZADATAK URUČEN: 10. 9. 2018. | POTPIS MENTORA

REPUBLIKA HRVATSKA
SVEUČILIŠTE SJEVER

Predgovor

Obavljajući stručnu praksu u tvrtci Elektrokem d.o.o. radio sam na zanimljivom projektu koji sam odlučio obraditi i iskoristiti za svoj završni rad. Naziv projekta je Upravljanje promjenjivom signalizacijom graničnih prijelaza. Djelatnici Elektrokema ponudili su pomoć oko upoznavanja sa samim projektom i uređaje iz njihove vlastite proizvodnje kako bi sam projektirao upravljanje signalizacijom za jedan od graničnih prijelaza. Ovaj završni rad detaljno opisuje projektiranje takvog sustava.

Sažetak

Na graničnim prijelazima Republike Hrvatske ugrađeni su znakovi sa promjenjivom signalizacijom koji su napravljeni u LED tehnologiji. Tom signalizacijom korisnici sustava usmjeravaju promet za određena vozila na određene prometne trake. Upravljanje znakovima promjenjive signalizacije obavlja se putem računala koja imaju instalirani upravljački software, tj. grafičko korisničko sučelje.

KLJUČNE RIJEČI: granični prijelaz, programibilni logički modul, mikrokontroler, znak promjenjive signalizacije, grafičko korisničko sučelje, programibilni logički sklopovi.

Abstract

At the border crossings of the Republic of Croatia are embedded signs with variable signaling that are made in LED technology. By signaling this, system users direct traffic to certain vehicles on certain traffic lanes. Variable signaling control is performed via a computer that has installed driver software, ie graphical user interface.

KEY WORDS: border crossing, programmable logic module, microcontroller, variable signaling sign, graphical user interface, programmable logic circuits

Popis korištenih kratica

AC-DC Izmjenična struja - istosmjerna struja

LED eng. *Light emiting diode*
Dioda koja zrači svjetlost

RS485 Vrsta protokola za serijski prijenos podataka na veće udaljenosti

USB eng. *Universal Serial Bus*
Univerzalna serijska sabirnica za komunikaciju računala s vanjskim uređajima

SPI eng. *Serial Peripheral Interface*
Protokol za serijski prijenos podataka na manje udaljenosti

UART eng. *Universal Asynchronous Receiver/Transmitter*
Protokol za serijski prijenos podataka na manje udaljenosti

MOSI eng. *Master Out Slave In*
Prijenos podataka od mastera prema slave-u

MISO eng. *Master In Slave Out*
Prijenos podataka od slave-a prema masteru

SCK eng. *SPI Clock*
Linija za takt pod kojim se izvodi komunikacija

CS eng. *Chip Select*
Pin za odabir slave modula

Sadržaj

1. Uvod

Kako bi se regulirao promet, na nadstrešnicama graničnih prijelaza Republike Hrvatske ugrađuju se znakovi s promjenjivom signalizacijom. Upravljanje znakovima promjenjive signalizacije obavlja se putem centralnog računala koje može upravljati svim znakovima na cijelom graničnom prijelazu. Također, znakovima se može upravljati i s pomoćnim računalima koja se nalaze u kućicama smještenim uz prometne trake kojima se upravlja.

Za potrebe upravljanja promjenjivom signalizacijom graničnog prijelaza, ugrađuje se upravljački ormar koji kontrolira prikaz oznaka promjenjivih svjetlosnih znakova toka prometa te njihov intenzitet. Upravljački ormar može upravljati s 18 znakova. Glavno računalo i pomoćna računala omogućuju daljinsko upravljanje oznakama i intenzitetom znakova s više mjesta u isto vrijeme putem RS485 komunikacije. [1]

Znakovi s promjenjivom signalizacijom na graničnim prijelazima su standardizirani i podijeljeni u tri funkcionalne grupe:[1]

1. promjenjivi znakovi dopuštenja prolaska :
   a) znak s prometnom oznakom G09 (crveni križ) – prometna traka je zatvorena
   b) znak s prometnom oznakom G10 (zelena strelica) – prometna traka je otvorena
2. promjenjivi znakovi vrste vozila:
   a) znak s prometnom oznakom AUTO – za prolazak automobila
   b) znak s prometnom oznakom BUS – za prolazak autobusa
3. promjenjivi znakovi porijekla putnika:
   a) znak s prometnom oznakom SVI PUTNICI/ ALL PASSENGERS – za prolazak svih putnika
   b) znak s prometnom oznakom EU EEA CH – za prolazak iz Europske unije i Švicarske

Uz promjenu prikaza oznaka svi promjenjivi svjetlosni znakovi toka prometa te oznake mogu prikazivati s dva različita intenziteta svjetlosti:

DAN – jači intenzitet svjetla

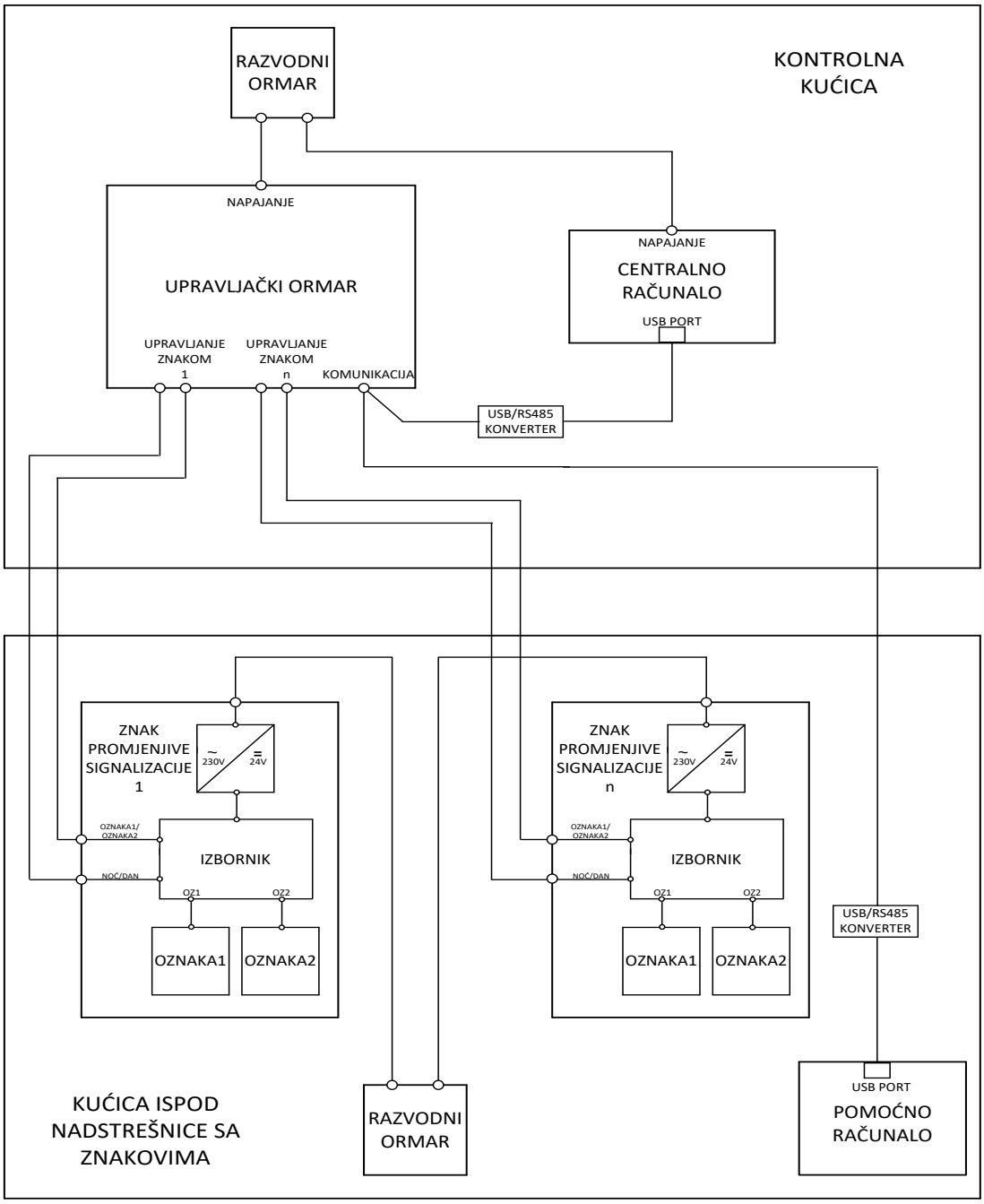
NOĆ – slabiji intenzitet svjetla

Promjena oznake i stupnja intenziteta svjetlosti promjenjivih svjetlosnih znakova vrši se upravljačkim signalima na izlaznim stezaljkama ormara.

U slučaju greške upravljanja promjenjivom signalizacijom putem računala, upravljanje je moguće i prekidačima koji se nalaze na upravljačkom ormaru koji je smješten u kontrolnoj kućici.

2. Opis sustava upravljanja promjenjivom signalizacijom graničnih prijelaza

Na slici 2.1. prikazana je blok shema sustava upravljanja promjenjivom signalizacijom graničnih prijelaza. Shema je općenita radi same preglednosti i sastoji se od osnovnih dijelova sustava. Ovisno o veličini graničnog prijelaza, pojedini elementi sustava višestruko se ponavljaju.



Slika 2.1. Blok shema sustava upravljanja promjenjivom signalizacijom graničnih prijelaza[1]

Svi elementi sustava smješteni su na dva tipa lokacije i to kontrolnu kućicu i kućicu ispod nadstrešnice sa znakovima. Kontrolna kućica je samo jedna, a broj kućica ispod nadstrešnice sa znakovima ovisi o broju prometnih traka na tom graničnom prijelazu.

U kontrolnoj kućici upravljački ormar i centralno računalo dobivaju napajanje iz razvodnog ormara. Upravljački ormar i centralno računalo komuniciraju preko RS485 komunikacije. Kako centralno računalo, koje je zapravo klasično stolno računalo sa minimalnom konfiguracijom i operativnim sustavom Windows 10, ne može obraditi RS485 komunikaciju potrebno je između upravljačkog ormara i centralnog računala ugraditi USB/RS485 konverter.[1]

Upravljački ormar na temelju zahtjeva sa centralnog računala upravlja 12-voltnim signalima sa znakovima koji su smješteni u kućici ispod nadstrešnice sa znakovima. Svaki od znakova sastoji se od AC/DC pretvarača, modula izbornika i LED modula koji definiraju promjenjivi znak. U kućici ispod nadstrešnice sa znakovima nalazi se i pomoćno računalo koje može upravljati samo sa znakovima iznad te kućice.

3. Opis modula sustava upravljanja promjenjivom signalizacijom graničnih prijelaza

3.1. Znak promjenjive signalizacije

Svaki od znakova sastoji se od AC/DC pretvarača, modula izbornika i LED modula koji definiraju promjenjivi znak.

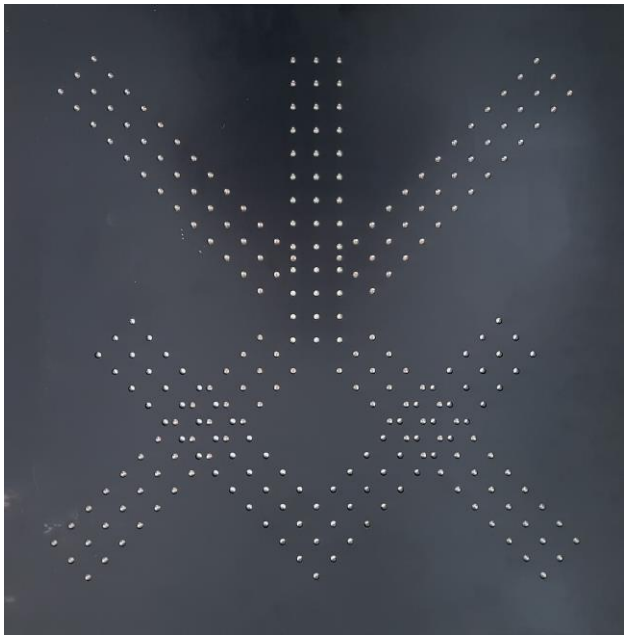
AC/DC pretvarač pretvara izmjenični napon mreže 230V/50Hz u istosmjerni napon iznosa 12V za potrebe napajanja znaka. Može osigurati snagu od 30W.[1]

Modul izbornika ima ugrađene releje koji ovisno o signalima sa upravljačkog ormara prosljeđuju napajanje prema LED modulima i tako se odabire koji će signal biti prikazan na znaku, a također određuje koliki će biti intenzitet svjetlosti svakog od signala prikazanog na znaku (DAN/NOĆ opcija).

Oznaka na znaku sastoji se od LED modula koji su međusobno električki paralelno povezani. Svaki od modula za ispravan rad treba 12 VDC napajanje (za rad preko dana), odnosno 10 VDC (za rad preko noći). Oznaka je predefinirana, a aktivna je ona oznaka koja dobije napajanje iz modula izbornika.

Oznaka G09 (crveni križ) sastoji se od ukupno 156 crvenih signalnih LED dioda koje zrače svjetlo pod kutem 30°, a oznaka G10 (zelena strelica) sastoji se od ukupno 114 zelenih signalnih LED dioda koje zrače svjetlo pod kutem 30°.[1]

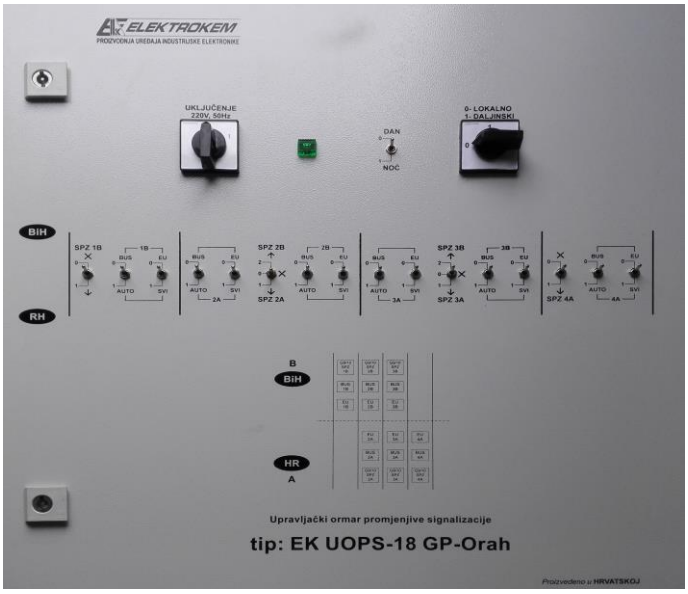
Na slici 3.1. prikazan je znak promjenjive signalizacije za dopuštenje prolaska (G09/G10).



Slika 3.1. Znak promjenjive signalizacije za dopuštenje prolaska (G09/G10)[1]

3.2. Upravljački ormar

Upravljački ormar (slika 3.2. i slika 3.3.) sastoji se od ulaznih sklopnih i zaštitnih elemenata, lokalnih signalizacijskih tinjalica, sklopki koje omogućuju ručno upravljanje znakovima, izlaznog razvoda i svojih najvažnijih modula – programibilnih logičkih sklopova.



Slika 3.2. Izgled upravljačkog ormara izvana

Slika 3.3. Izgled upravljačkog ormara iznutra

3.3. Modul programibilnih logičkih sklopova

Programibilni logički sklopovi čine jedan od modula u sustavu upravljanja graničnim prijelazima. Više ovih modula nalazi se u upravljačkom ormaru, a njihov broj ovisi o broju znakova kojima sustav upravlja. Funkcija ovog modula je uklapanje i isklapanje releja koji kontroliraju prikaz na svjetlosnim promjenjivim znakovima toka prometa. Svaki takav modul sadrži šest releja. Također, preko RS485 sučelja svaki od modula komunicira sa ostalima u upravljačkom ormaru, te s računalom na kojem je instalirana upravljačka aplikacija.[1]

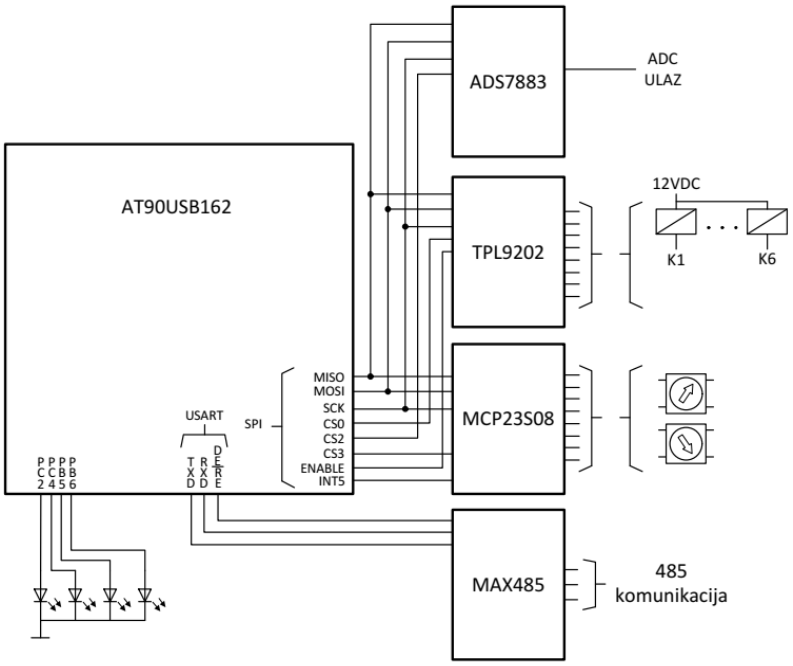
Svaki programibilni modul ima svoju podesivu adresu, koja se podešava pomoću rotirajućih prekidača. Pošto svaki modul ima svoju jedinstvenu adresu, pomoću željene adrese i rednog broja releja u modulu koji se želi uklopiti, moguće je točno definirati koji znak mijenja svoj prikaz. [1]

Također, osim relejnih izlaza, programibilni modul sadrži i jedan ulaz za analogno-digitalnu pretvorbu, što mu omogućuje mjerenje analognih veličina u sustavu, ukoliko je to potrebno.

Programibilni modul sastoji se od više integriranih krugova međusobno povezanih u cjelinu. Glavna upravljačka jedinica ovog modula je mikrokontroler, AT90USB162, proizvođača Microchip. Njegov zadatak je upravljati ostalim elementima u modulu, te komunicirati s ostatkom sustava, tj. drugim takvim modulima i upravljačkom aplikacijom na računalu.[2]

Komunikacija između mikrokontrolera i ostalih integriranih krugova izvedena je SPI protokolom i UART protokolom. Također, na pinove mikrokontrolera spojene su i četiri LED diode, za svjetlosnu signalizaciju stanja sustava.[1]

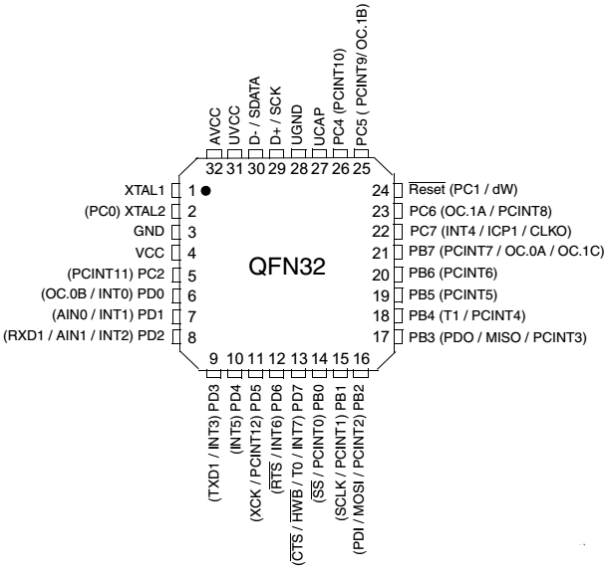
Slika 3.4. prikazuje pojednostavljenu blok shemu modula. U nastavku teksta detaljnije je opisan rad elektroničkog sklopa, te je objašnjen upravljački kod mikrokontrolera.



Slika 3.4. Blok shema programibilnog modula[2]

3.3.1. **Microchip AT90USB162 mikrokontroler**[2]

AT90USB162 mikrokontroler glavna je upravljačka jedinica programibilnog modula. Ovaj mikrokontroler odabran je zbog prikladne cijene, dovoljnog broja pinova (slika 3.5.) za spajanje svih ostalih integriranih krugova, a performanse su mu dovoljne za izvršavanje svih funkcija ovog sustava. Ovaj mikrokontroler je 8-bitni, a maksimalna radna frekvencija mu je 16MHz. Ima 22 programibilna pina, kojima se mogu dodijeliti razne funkcije.



Slika 3.5. Pinovi mikrokontrolera AT90USB162[2]

Program upravljačke logike za ovaj mikrokontroler izrađen je u programskom okruženju Atmel Studio 6.2. Nakon što je program napisan, potrebno ga je kompajlirati, te kopirati u flash memoriju samog čipa, što se izvršava pomoću uređaja Atmel JTAG Ice MK2, koji se preko USB sučelja povezuje s računalom, te se spaja na mikrokontroler.

Na pinove mikrokontrolera spojena su četiri integrirana kruga, od kojih svaki obavlja svoju definiranu funkciju. Tri integrirana kruga spojena su preko SPI protokola, a četvrti je spojen preko UART protokola.

3.3.2. **SPI protokol**

SPI (Serial Peripheral Interface) koristi se za serijski prijenos podataka na manjim udaljenostima, razvijen od strane Motorolinih inženjera 80-ih godina. Široka paleta integriranih krugova za komunikaciju koristi upravo SPI, kao što su SD memorijske kartice, LCD zasloni, analogno-digitalni pretvornici, itd.

Na SPI sabirnicu može se vezati jedan master (nadređeni) uređaj, te više slave (podređeni) uređaja. Za implementaciju protokola potrebne su 4 linije:[7]

1. MOSI – (Master Out Slave In) – Prijenos podataka od mastera prema slave-u
2. MISO – (Master In Slave Out) – Prijenos podataka od slave-a prema masteru
3. SCK – (SPI Clock) – linija za takt pod kojim se izvodi komunikacija. Takt daje master modul
4. CS – (Chip Select) – pritezanjem ovog pina na niski nivo, odabire se slave uređaj kojem se šalju naredbe. Npr. ako je na master modul spojeno više modula, svaki od njih ima posebnu CS liniju prema njemu. Onda, ovisno s kojim modulom master želi komunicirati, njegova CS linija pritegnuta je na niski nivo.

3.3.3. **UART protokol**

UART (Universal Asynchronous Receiver/Transmitter) protokol vrsta je serijske komunikacije između dva uređaja, korištenjem dvije linije, Rx i Tx. Kao što mu i ime govori, prijenos podataka je asinkron. Svako slanje podataka započinje se START bitom, koji je uvijek digitalna nula. Nakon START bita najčešće se šalje 8 bitova podataka. Iza bitova podataka može se slati i paritetni bit, što omogućuje utvrđivanje ispravnosti primljenih podataka. Svako slanje podataka završava se STOP bitom. Za ispravnu komunikaciju, oba uređaja koja komuniciraju

moraju imati podešene iste parametre, brzinu prijenosa, broj bitova podataka, paritetni bit i broj stop bitova. [8]

3.3.4. **RS485 protokol**

RS485 je protokol koji se često koristi za komunikaciju u industrijskim okruženjima. Jedna od prednosti ovog standarda je manja osjetljivost na smetnje. Duljina komunikacijskog voda kod RS485 protokola može biti i do 1200m, a na istu komunikacijsku sabirnicu može se spojiti više uređaja. U ovom sustavu, UART pinovi mikrokontrolera spajaju se na integrirani krug koji poruke konvertira na RS485 standard, što omogućuje komunikaciju između modula, te računala s upravljačkom aplikacijom. [8]

3.3.5. **MCP23S08 – I/O ekspander**

Svrha ovog integriranog kruga „proširenje“ je broja pinova koje mikrokontroler može koristiti. Ovaj integrirani krug sadrži osam linija, koje se mogu ponašati kao digitalni ulazi ili izlazi. Preko SPI sučelja povezan je s mikrokontrolerom. Na blok shemi na Slici 3.4. vidljivo je da spoj mikrokontrolera i MCP23S08 integriranog kruga zahtijeva 5 linija. Pomoću tih 5 linija može se čitati 8 ulaza, ili upravljati s 8 izlaza, što je „ušteda“ od 3 pina. Npr. da je u sustavu potrebno još pinova, pošto se MISO, MOSI te SCK pinovi koriste za sve SPI uređaje, bilo bi potrebno dodati još samo dva pina (CS, INT) za još osam digitalnih ulaza/izlaza.[3]

INT linija koristi se za generiranje tzv. interrupta. Ovisno o definiranom uvjetu, MCP23S08 pomoću te linije „dojavljuje“ mikrokontroleru da se definirani uvjet dogodio, pa onda mikrokontroler može pravovremeno izvršiti dio koda koji je u tom trenutku potreban.

U programibilnom modulu, na linije ovog pina spajaju se dva rotacijska prekidača, kojima se odabire adresa svakog programibilnog modula, potrebna za RS485 komunikaciju. Svaki od prekidača može se postaviti na 16 različitih vrijednosti (0-9, te A-F), što omogućuje 256 različitih adresa na RS485 sabirnici. Upravljački kod mikrokontrolera u ovom slučaju koristi MCP23S08 linije kao digitalne ulaze, na kojima čita vrijednosti podešenih prekidača. Slika 3.6. prikazuje vrijednosti na ulazima MCP23S08 integriranog kruga, ovisno o podešenoj adresi.

BINARY CODED HEXADECIMAL				
16 Positions				
Dial No.	1	2	4	8
0				
1	●			
2		●		
3	●	●		
4			●	
5	●		●	
6		●	●	
7	●	●	●	
8				●
9	●			●
A		●		●
B	●	●		●
C			●	●
D	●		●	●
E		●	●	●
F	●	●	●	●

Slika 3.6. Vrijednosti rotacijskih prekidača ovisno o podešenoj adresi[3]

3.3.6. TPL9202 – integrirani krug za upravljanje relejima

Ovaj integrirani krug također je preko SPI sučelja spojen na mikrokontroler. Njegova funkcija je uklapanje/isklapanje šest releja programibilnog modula, koji zatim svojim kontaktima upravljaju prikazima na promjenjivim svjetlosnim prometnim znakovima. [4]

Jedna strana zavojnice releja spojena je na +12V DC, a druga strana je spojena na TPL9202 integrirani krug. Ukoliko treba uklopiti relej, on određenu liniju spaja na 0V DC. Maksimalna struja svakog od izlaza ovog integriranog kruga je 150mA. Također, svaki izlaz ima i zaštitu od prenapona na induktivnom teretu, te zaštitu od prevelike struje na izlazu.

3.3.7. ADS7883 – analogno-digitalni pretvornik

ADS7883 integrirani krug spaja se na mikrokontroler preko SPI sučelja, a njegov zadatak je čitati jednu analognu vrijednost, te ju u digitalnom obliku proslijediti u mikrokontroler. Rezolucija analogno-digitalnog pretvornika u ovom integriranom krugu je 12 bita. [5]

ADS7883 čita analognu vrijednost promjenjivog otpora sonde luksomata.

3.3.8. MAX485

MAX485 integrirani krug spojen je na serijsko sučelje mikrokontrolera, na pinove RX i TX. Njegova zadaća je konverzija zaprimljenih poruka sa strane mikrokontrolera u RS485 standard.

Također, konverzija je dvosmjerna, ukoliko je s RS485 strane zaprimljena poruka, ovaj integrirani krug konvertira ju u oblik prikladan za UART sučelje, te ona kao takva dolazi na RX pin mikrokontrolera. Ovisno o stanju DE/RE pina mikrokontrolera, koji je također spojen na MAX485 integrirani krug, bira se slanje podataka prema 485 sabirnici ili primanje podataka s 485 sabirnice. [6]

4. Opis programa upravljačkog modula

Programiranje programibilnog modula vrši se pomoću programskog paketa Atmel Studio 6.2, u programskom jeziku C. Nakon razvijanja programa i njegovog testiranja, kompajlirani kod kopira se uređajem ATMEL JTAG Ice MK2 u flash memoriju mikrokontrolera. U nastavku teksta opisan je upravljački program modula.

4.1. Definicije varijabli i funkcija

Prije pisanja upravljačke logike modula, potrebno je definirati sve pinove mikrokontrolera koji se koriste za razne funkcije. Npr. potrebno je definirati koji pin će se koristiti za MISO liniju SPI sučelja, na koji pin je spojena dijagnostička LED dioda PLC modula koju je potrebno uključiti i isključiti (npr. LED dioda koja svijetli prilikom paljenja mikrokontrolera), Tx i Rx pinovi za serijsku komunikaciju, te svi ostali potrebni pinovi za ispravan rad modula.

```
1. #include <avr/io.h>
2. #include <avr/eeprom.h>
3. #include <avr/interrupt.h>
4. #include <avr/wdt.h>
5. #include <ctype.h>
6. #include <math.h>
7. #include <stdio.h>
8. #include <string.h>
9. #include <stdlib.h>
10. #include <avr/sleep.h>
```

Programski blok 4.1. Pridruživanje datoteka

#include naredbama programu se pridružuju sve potrebne datoteke s raznim funkcijama koje se koriste u ostatku koda. Npr. definicije matematičkih funkcija nalaze se u math.h datoteci, pa se pridruživanjem ove datoteke one mogu koristiti.

```
1.  #define F_CPU 8000000
2.  #define LED11   PINB6
3.  #define LED12   PINB5
4.  #define LED13   PINC4
5.  #define LED14   PINC2
6.  #define SCLK    PINB1
7.  #define MOSI    PINB2
8.  #define MISO    PINB3
9.  #define CS_0    PINC5
10. #define CS_1    PINC6
11. #define CS_2    PINC7
12. #define CS_3    PINB0
13. #define ENABLE  PIND5
14. #define DE_RE   PIND6
15. /* MCP23S08 register map */
16. #define IODIR       0x00
17. #define IPOL        0x01
18. #define GPINTEN     0x02
19. #define DEFVAL      0x03
20. #define INTCON      0x04
```

```
21. #define IOCON       0x05
22. #define GPPU        0x06
23. #define INTF        0x07
24. #define INTCAP      0x08
25. #define GPIO        0x09
26. #define OLAT        0x0A
```

Programski blok 4.2. Definiranje pinova

U nastavku koda definiraju se pinovi koji se koriste za razne funkcije. Npr. na slici 3.5, vidljivo je da je pin 16 mikrokontrolera definiran kao PB2. Pošto je njegova funkcija u ovom program Master Out Slave In u SPI komunikaciji, on se tako i definira (linija 7 programski blok 4.2.).

Također, definirani su i registri MCP23S08 integriranog kruga, koji se koriste za upravljanje njegovim funkcijama tijekom čitanja/pisanja u njegove digitalne ulaze/izlaze.

```
1.  Unsigned char lookup_table[16] = {
2.      0x00, 0x01, 0x08, 0x09, 0x04, 0x05, 0x0c, 0x0d, 0x02, 0x03, 0x0a, 0x0b, 0x06, 0x07, 0x0e, 0x0f
3.  };
4.  volatile int adc_ref;
5.  volatile int is_lux;
6.  volatile int lux_on_off;
7.  volatile int global_counter;
8.  volatile int ana_in;
9.  volatile int have_line;
10. volatile int device_address;
11. volatile int counter;
12. volatile int gpio;
13. static char RS485_BUFFER[128];
14. void init_usart(unsigned long, uint8_t, uint8_t, uint8_t);
15. void set_device_status(uint8_t);
16. void spi_master_transmit_byte_gpio(uint8_t);
17. void rs485_writeData(char * , int);
18. void parse_cmd(char * );
19. void spi_write_mcp23s08_reg(uint8_t, uint8_t);
20. uint8_t spi_read_mcp23s08_reg(uint8_t);
21. uint8_t read_device_address(void);
22. int read_adc_value(void);
23. int isNumber(char * );
```

Programski blok 4.3. Definiranje varijabli i funkcija

Nakon definicije pinova i pridruživanja svih potrebnih datoteka, potrebno je definirati sve varijable i prototipove funkcija koje se koriste u ostatku koda. Ove funkcije objašnjene su u nastavku teksta.

4.2. Implementacija funkcija za komunikaciju

4.2.1. init_usart

Funkciji init_usart zadaća je postaviti sve registre mikrokontrolera vezane za UART komunikaciju, preko koje se podaci prosljeđuju do MAX485 integriranog kruga. Prilikom

pozivanja funkcije, kao ulazni parametri šalju se željeni podatak za brzinu prijenosa (baudrate), broj podatkovnih bitova, paritet, te broj stop bitova. Registri se podešavaju ovisno o tim parametrima.

Npr. vidljivo je da se ovisno o broju definiranih podatkovnih bitova različito podešavaju bitovi UCSZ10 i UCSZ11 registra UCSR1C, koji u slučaju 5 podatkovnih bitova iznose 0 i 0, a u slučaju šest, 1 i 0.

```
1.        void init_usart(unsigned long baudrate, uint8_t databits, uint8_t parity, uint8_t stopb
   its) {
2.            unsigned long UBRR_VALUE;
3.            UBRR_VALUE = (F_CPU / (baudrate << 4)) - 1;
4.            UBRR1H = (uint8_t)(UBRR_VALUE >> 8);
5.            UBRR1L = (uint8_t)(UBRR_VALUE & 0xFF);
6.            UCSR1A = 0x00;
7.            UCSR1B = 0x00;
8.            UCSR1C = 0x00;
9.            switch (databits) {
10.               case 5:
11.                   UCSR1B |= 0 << UCSZ12;
12.                   UCSR1C |= (0 << UCSZ10) | (0 << UCSZ11);
13.                   break;
14.               case 6:
15.                   UCSR1B |= 0 << UCSZ12;
16.                   UCSR1C |= (1 << UCSZ10) | (0 << UCSZ11);
17.                   break;
18.               case 7:
19.                   UCSR1B |= 0 << UCSZ12;
20.                   UCSR1C |= (0 << UCSZ10) | (1 << UCSZ11);
21.                   break;
22.               case 8:
23.                   UCSR1B |= 0 << UCSZ12;
24.                   UCSR1C |= (1 << UCSZ10) | (1 << UCSZ11);
25.                   break;
26.               case 9:
27.                   UCSR1B |= 1 << UCSZ12;
28.                   UCSR1C |= (1 << UCSZ10) | (1 << UCSZ11);
29.                   break;
30.               default:
31.                   break;
32.           }
33.           UCSR1B |= (1 << RXEN1) | (1 << TXEN1); // TX/RX ENABLE/DISABLE
34.           UCSR1B |= (0 << TXCIE1) | (1 << RXCIE1); // INTERRUPT ENABLE/DISABLE
35.           PORTD |= (1 << ENABLE);
36.       }
```

Programski blok 4.4. Funkcija za komunikaciju

4.2.2. **rs485_writeData**

Ova funkcija preko UART sučelja šalje željene poruke na MAX485 integrirani krug, koji zatim te poruke prosljeđuje ostatku sustava. Parametri koji se predaju funkciji prilikom pozivanja su polje s podacima koje je prethodno definirano (`static char RS485_BUFFER[128]`), te veličina podataka koji se šalju.

Slanje podataka vrši se postavljanjem UDR1 registra na vrijednost koju se želi poslati van. To se obavlja u petlji, koja se izvršava tako dugo dok god ima podataka. Nakon svakog postavljanja UDR1 registra, čeka se dok se on opet ne može postavljati.

Prije slanja, pin DE_RE postavlja se u stanje digitalne '1', što signalizira MAX485 integriranom krugu slanje podataka prema ostatku sustava. Nakon slanja podataka, vrijednost pina vraća se u stanje '0'.

Nakon završetka slanja, vrijednosti polja RS485_BUFFER, memset naredbom postavljaju se u '0'.

```
1.  void rs485_writeData(char * cPtr, int length) {
2.      PORTD |= 1 << DE_RE;
3.      while (length--) {
4.          UDR1 = * (cPtr++);
5.          while (!(UCSR1A & (1 << UDRE1)));
6.      }
7.      UDR1 = '\0';
8.      while (!(UCSR1A & (1 << UDRE1)));
9.      PORTD &= ~(1 << DE_RE);
10.     memset((void * ) RS485_BUFFER, 0x00, sizeof(RS485_BUFFER));
11. }
```

Programski blok 4.5. Funkcija za slanje poruke na MAX485

4.2.3. Interrupt rutina za čitanje zaprimljenih podataka na UART sučelju

Prilikom zaprimanja podataka na Rx pinu mikrokontrolera, pokreće se interrupt rutina za obradu podataka koji su zaprimljeni. Interrupt rutina se koristi zbog toga jer bi inače neprekidno trebalo provjeravati stižu li neki podaci na Rx ulaz. Ovakvom konfiguracijom to nije potrebno, jer to mikrokontroler odrađuje samostalno.

Na početku izvršavanja ove rutine, vrijednost UDR1 registra, koji sadrži zaprimljene podatke učitava se u varijablu c. Vrijednost jednog bajta polja RS485_BUFFER zatim se postavlja u učitanu vrijednost. Nakon što se tako popuni svih 128 bajtova polja, poziva se funkcija parse_cmd, koja obrađuje zaprimljeni podatak.

```
ISR(USART1_RX_vect) {
    static char c;
    static uint8_t byte_counter = 0;
    c = UDR1;
    if (c != '\n') {
        if (byte_counter < sizeof(RS485_BUFFER) - 1) {
            RS485_BUFFER[byte_counter++] = c;
        }
    } else {
        RS485_BUFFER[byte_counter++] = '\n';
        RS485_BUFFER[byte_counter] = '\0';
        byte_counter = 0;
        parse_cmd(RS485_BUFFER);
        memset((void * ) RS485_BUFFER, 0x00, sizeof(RS485_BUFFER));
    }
}
```

Programski blok 4.6. Rutina za čitanje zaprimljenih podataka

4.2.4. **init_spi**

Kao i funkcija init_usart, funkcija init_spi postavlja sve vrijednosti registara vezanih za SPI komunikaciju u mikrokontroleru. Prvo su definirani svi pinovi potrebni za komunikaciju (MOSI, SCLK, CS,…), te im se zatim vrijednosti postavljaju na željene (npr. svi CS pinovi postavljeni su u stanje logičke ‘1’). Nakon tog koraka, postavljaju se bitovi SPE (SPI Enable) te MSTR (Master način rada) u registru SPCR.

```
void init_spi(void) {
    DDRB |= (1 << MOSI) | (1 << SCLK) | (1 << CS_3);
    DDRC |= (1 << CS_2) | (1 << CS_1) | (1 << CS_0);
    PORTB |= (1 << CS_3);
    PORTC |= (1 << CS_0) | (1 << CS_1) | (1 << CS_2);
    SPCR = (0 << SPIE) | (1 << SPE) | (0 << DORD) | (1 << MSTR) | (0 << CPOL) | (0 << CPHA
) | (0 << SPR1) | (0 << SPR0);
    SPSR = (1 << SPI2X);
}
```

Programski blok 4.7. Vrijednosti vezane za SPI komunikaciju

4.2.5. **spi_master_transmit_byte_gpio**

Ova funkcija SPI komunikacijom upravlja TPL9202 integriranim krugom, koji uklapa i isklapa releje na izlazu modula. Linija CS0 prvo se postavlja u stanje ‘0’, a ona je spojena na TPL9202 integrirani krug, što mu signalizira da on prima podatke. Nakon toga, registar SPDR puni se podacima koji se šalju. Nakon toga, u petlji se čeka dok god prijenos podataka nije završen (bit SPIF u SPSR registru). Nakon prijenosa poruke, CS0 postavlja se natrag u stanje ‘1’.

```
1.   void spi_master_transmit_byte_gpio(uint8_t cData) {
2.       if (!(cData & 0x01)) PORTD &= ~(1 << ENABLE);
3.       else PORTD |= (1 << ENABLE);
4.       PORTC &= ~(1 << CS_0);
5.       SPDR = cData;
6.       loop_until_bit_is_set(SPSR, SPIF);
7.       PORTC |= (1 << CS_0);
8.   }
```

Programski blok 4.8. Upravljanje TPL9202 integriranim krugom

4.2.6. spi_write_mcp23s08_reg i spi_read_mcp23s08_reg

Ova funkcija zapisuje podatke u MCP23S08 integrirani krug. Njezin rad sličan je prethodno opisanoj funkciji. Određena CS linija postavlja se u stanje '0', registar SPDR puni se podacima, te se čeka dok prijenos podataka nije završen. Ukupno se šalju tri podatka. Prvi podatak je kontrolni bajt. Njegova vrijednost iznosi 0|1|0|0|0|A1|A0|R/W. Pošto su A1 i A0 jednaki nuli (to su pinovi integriranog kruga, pritegnuti na 0V, što označava vrijednost 0), jedina razlika je R/W bit, koji označava čita li se ili piše u čip. Pošto ova funkcija zapisuje podatke, on je 0. Onda je vrijednost kontrolnog bajta 0100 0000, što je u heksadecimalnom zapisu 0x40.

Drugi podatak koji se šalje je adresa registra u koji se želi upisivati, a treći je vrijednost koja se upisuje.

```
1.   void spi_write_mcp23s08_reg(uint8_t reg, uint8_t cData) {
2.       PORTB &= ~(1 << CS_3);
3.       SPDR = 0x40;
4.       loop_until_bit_is_set(SPSR, SPIF);
5.       SPDR = reg;
6.       loop_until_bit_is_set(SPSR, SPIF);
7.       SPDR = cData;
8.       loop_until_bit_is_set(SPSR, SPIF);
9.       PORTB |= (1 << CS_3);
10.  }
```

Programski blok 4.9. Zapisivanje podataka u MCP23S08

Funkcija za čitanje je veoma slična onoj za pisanje. U ovom slučaju, kontrolni bajt je 0x41, pošto se mijenja vrijednost zadnjeg bita.

```
1.          uint8_t spi_read_mcp23s08_reg(uint8_t reg) {
2.              uint8_t retval;
3.              PORTB &= ~(1 << CS_3);
4.              SPDR = 0x41;
5.              loop_until_bit_is_set(SPSR, SPIF);
6.              SPDR = reg;
7.              loop_until_bit_is_set(SPSR, SPIF);
8.              SPDR = 0xFF;
9.              loop_until_bit_is_set(SPSR, SPIF);
10.             retval = SPDR;
11.             PORTB |= (1 << CS_3);
12.             return retval;
13.         }
```

Programski blok 4.10. Čitanje podataka iz MCP23S08

4.2.7. **read_adc_value**

Ova funkcija preko SPI sučelja komunicira s ADS7883 integriranim krugom, koji čita analognu vrijednost te ju šalje u mikrokontroler u digitalnom obliku. Način implementacije je istovjetan prošlim isječcima koda vezanim uz SPI komunikaciju.

```
1.  int read_adc_value() {
2.      int val;
3.      val = 0x0000;
4.      PORTC &= ~(1 << CS_2);
5.      SPDR = 0xFF;
6.      loop_until_bit_is_set(SPSR, SPIF);
7.      val = SPDR << 8;
8.      SPDR = 0xFF;
9.      loop_until_bit_is_set(SPSR, SPIF);
10.     val |= SPDR;
11.     PORTC |= (1 << CS_2);
12.     return (int)((val >> 2));
13. }
```

Programski blok 4.11. Komunikacija s ADS7883

4.3. Ostale funkcije

4.3.1. **isNumber**

Funkcija isNumber provjerava je li zadani podatak broj. Ukoliko zadani podatak nije slovo abecede, ni ‘\0’, ‘\r’, ‘\n’, proglašava se brojem.

```
int isNumber(char * ptr) {
    int i;
    for (i = 0;
        ((ptr[i] != '\0') && (ptr[i] != '\r') && (ptr[i] != '\n')); i++) {
        if (isalpha(ptr[i])) return 0;
    }
    return 1;
}
```

Programski blok 4.12. Provjeravanje zadanog podatka

4.3.2. **read_device_adress**

Funkcija read_device_adress čita stanje dva rotirajuća prekidača kojima se podešava adresa pojedinog programibilnog modula.

```
uint8_t read_device_address() {
    static int i = 0;
    static unsigned char LB = 0;
    static unsigned char HB = 0;
    uint8_t input;
    input = spi_read_mcp23s08_reg(GPIO);
    LB = input & 0x0F;
    HB = (input >> 4) & 0x0F;
    for (i = 0; i < sizeof(lookup_table); i++) {
        if (lookup_table[i] == LB) {
            LB = i;
            break;
        }
    }
    for (i = 0; i < sizeof(lookup_table); i++) {
        if (lookup_table[i] == HB) {
            HB = i;
            break;
        }
    }
    input = (LB << 4) | HB;
    return input;
}
```

Programski blok 4.13. Čitanje adrese programibilnog modula

4.3.3. set_device_status

Funkcija set_device_status koristi se za uključenje/isključenje signalne LED diode modula, čime se signalizira status programibilnog modula. Npr. PLC na koji se spaja luksomat može imati upaljenu narančastu LED diodu, što olakšava lakšu dijagnostiku sustava.

```
1.   void set_device_status(uint8_t val) {
2.       if (val) PORTB |= (1 << LED11);
3.       else PORTB &= ~(1 << LED11);
4.   }
```

Programski blok 4.14. Signalizacija statusa programibilnog modula

4.3.4. start_timer

Funkcija start_timer podešava vrijednosti brojača koji periodički generira interrupte. Prilikom svakog generiranog interrupta, u interrupt rutini se izvršava glavni upravljački dio programa. Ovisno o podešenoj vrijednosti brojača na kojoj se generira interrupt (OCR0A registar), te o frekvenciji takta, interrupt se može generirati u različitim vremenskim razmacima.

```
1.   void start_timer() {
2.       OCR0A = 0xFA;
3.       TCNT0 = 0x00;
4.       TCCR0A = (0 << COM0B0) | (0 << COM0B1) | (0 << WGM00) | (1 << WGM01) | (0 << WGM02);
5.       TCCR0B = (1 << CS00) | (1 << CS01) | (0 << CS02);
6.       TIFR0 = (1 << OCF0A);
7.       TIMSK0 = (1 << OCIE0A);
8.   }
```

Programski blok 4.15. Start timer

4.3.5. main

U ovom dijelu programa inicijaliziraju se stanja svih pinova, pozivaju se funkcije koje inicijaliziraju komunikacijske kanale, čita se adresa uređaja, te ostale rutine potrebne za početnu inicijalizaciju mikrokontrolera.

Također, ovdje se i inicijalizira tzv. watchdog timer. Ukoliko, zbog neke neočekivane greške u logici programa, ili iz nekog drugog razloga dođe do situacije da se program ne izvršava nego je negdje “zapeo”, watchdog timer resetira mikrokontroler. Za ispravan rad, watchdog timer se treba resetirati u određenom vremenskom periodu. Ukoliko se to ne dogodi, smatra se da se program ne izvršava, te watchdog timer resetira mikrokontroler.

```
int main(void) {
    cli();
    DDRB = 0x00;
    DDRB |= (1 << LED11) | (1 << LED12);
    DDRC = 0x00;
    DDRC |= (0 << LED13) | (0 << LED14);
    DDRD = 0x00;
    DDRD |= (1 << ENABLE) | (1 << DE_RE);
    PORTB = 0x00;
    PORTC = 0x00;
    PORTD = 0x00;
    wdt_enable(WDTO_2S); //enable wdt interrupt
    gpio = 0;
    is_lux = 0;
    init_spi();
    init_usart(19200, 8, 0, 1);
    spi_master_transmit_byte_gpio(0x00);
    spi_write_mcp23s08_reg(IODIR, 0xFF);
    spi_write_mcp23s08_reg(IPOL, 0x00);
    spi_write_mcp23s08_reg(GPPU, 0x00);
    device_address = read_device_address();
    adc_ref = eeprom_read_word(0x00);
    set_device_status(1);
    ana_in = read_adc_value();
    start_timer();
    sei();
    for (;;) {
        wdt_reset();
    }
}
```

Programski blok 4.16. Početna inicijalizacija mikrokontrolera

4.3.6. **parse_cmd**

Zbog nepreglednosti, parse_cmd naredba izostavljena je iz prikaza i priložena je u prilogu 11.1. Funkcija parse_cmd naredbe obrada je podataka zaprimljenih preko RS485 linije. Ovisno o zaprimljenoj naredbi, mikrokontroler uklapa releje, vraća status releja, zapisuje podatke u EEPROM, čita podatke iz EEPROM-a, ili čita analogni ulaz.

Naredbe koje su implementirane su:

$GPIO_SET:ADRESA, NAREDBA – ova naredba uklapa releje definirane NAREDBA bajtom programibilnog modula s adresom ADRESA. Ukoliko je sve uspješno izvršeno, modul vraća poruku "$GPIO_SET:OK”, a ukoliko nije, "$GPIO_SET:ERROR”.

$GPIO_GET:ADRESA – ispisuje se stanje uklopljenih/isklopljenih releja

$EEPROM_DATA:ADRESA, VRIJEDNOST – vrijednost VRIJEDNOST upisuje se u EEPROM modula s adresom ADRESA

$GET_EEPROM:ADRESA – ispisuje se vrijednost EEPROM-a modula s adresom ADRESA

$GET_ANA_IN:ADRESA – ispisuje se vrijednost pročitana na analognom ulazu analogno-digitalnog konvertera programibilnog modula s adresom ADRESA.

4.3.7. **Interrupt rutina – timer**

Zadaća ove funkcije je periodički provjeravati stanje sustava, te ovisno o tome reagirati i dojaviti stanje upravljačkoj aplikaciji na računalu. Ova interrupt rutina vezana je s brojačem koji je prethodno inicijaliziran, te se periodički izvršava.

Ovisno o adresi modula, provjerava se stanje na analognom ulazu. Na analogni ulaz pojedinih programibilnih modula u upravljačkom ormaru vezana je grebenasta sklopka, kojoj je s jedne strane spojeno 5V DC. Ukoliko se sklopka zatvori, 5V DC prosljeđeno je na ulaz ADC pretvornika, te on može detektirati stanje sklopke.

Na analogni ulaz se također spaja i fotoosjetljivi otpornik, koji, ovisno o razini sunčevih zračenja, povećava ili smanjuje svoj otpor. Čitanjem tog analognog ulaza može se odrediti je li vani dan ili noć, te sukladno tome, uključiti dnevni ili noćni režim rada znakova.

Stanja sklopki koja se dojavljuju su:

$OPERATION MODE: AUTO ili MANUAL – pošto upravljački ormari imaju mogućnost rada bez programibilnog modula, ovom sklopkom se signalizira njihovo isključenje iz upravljačkog kruga.

$NETWORK: ON ili OFF – ukoliko se ormar isključi, ova poruka se šalje upravljačkoj aplikaciji na računalu, tako da korisnik može vidjeti da je ormar ugašen.

```
1.          ISR(TIMER0_COMPA_vect) {
2.              static int status = 0;
3.              static int net_status = 0;
4.              static int sw_on = 1;
5.              if ((counter > 100) && (counter < 200)) {
6.                  if (device_address == 0x01) {
7.                      if (sw_on) {
8.                          ana_in = read_adc_value();
9.                      }
10.                     if (ana_in >= 2500) {
11.                         if (!status) {
12.                             status = 1;
13.                             memset((void * ) RS485_BUFFER, 0x00, sizeof(RS485_BUFFER));
14.                             sprintf((void * ) RS485_BUFFER, "%s%s\r\n", "$OPERATION_MODE:", "AU
    TO");
15.                             rs485_writeData(RS485_BUFFER, strlen(RS485_BUFFER));
16.                         }
17.                     } else if (ana_in <= 2400) {
18.                         if (status || sw_on) {
19.                             memset((void * ) RS485_BUFFER, 0x00, sizeof(RS485_BUFFER));
20.                             sprintf((void * ) RS485_BUFFER, "%s%s\r\n", "$OPERATION_MODE:", "MA
    NUAL");
21.                             rs485_writeData(RS485_BUFFER, strlen(RS485_BUFFER));
22.                             status = 0;
23.                             sw_on = 0;
24.                         }
25.                     }
26.                 }
27.             }
28.             if (counter++ >= 500) {
29.                 ana_in = read_adc_value();
30.                 if (device_address == 0x0f) {
```

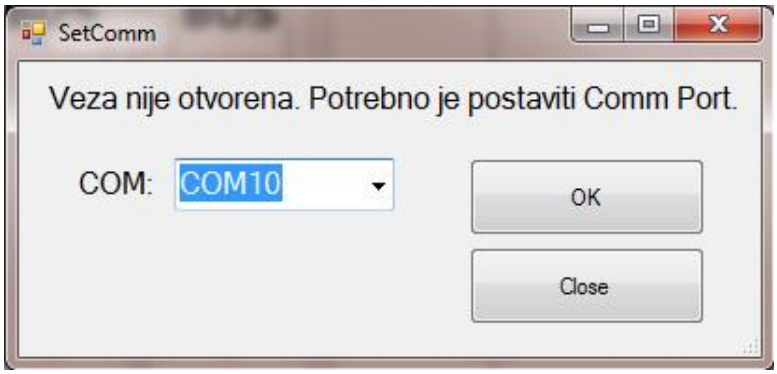
```
                    is_lux = 1;
                    PORTB |= (1 << LED12);
                    if (ana_in >= (adc_ref + 50)) {
                        gpio &= ~(1 << 5);
                        spi_master_transmit_byte_gpio((uint8_t) gpio);
                        lux_on_off = 0;
                    } else if (ana_in <= adc_ref) {
                        gpio |= (1 << 5);
                        spi_master_transmit_byte_gpio((uint8_t) gpio);
                        lux_on_off = 1;
                    }
                } else if (device_address == 0x02) {
                    if (ana_in >= 2500) {
                        if (!net_status) {
                            memset((void * ) RS485_BUFFER, 0x00, sizeof(RS485_BUFFER));
                            sprintf((void * ) RS485_BUFFER, "%s%s\r\n", "$NETWORK:", "ON");
                            rs485_writeData(RS485_BUFFER, strlen(RS485_BUFFER));
                            net_status = 1;
                        }
                    } else if (ana_in <= 2400) {
                        if (net_status) {
                            memset((void * ) RS485_BUFFER, 0x00, sizeof(RS485_BUFFER));
                            sprintf((void * ) RS485_BUFFER, "%s%s\r\n", "$NETWORK:", "OFF");
                            rs485_writeData(RS485_BUFFER, strlen(RS485_BUFFER));
                            net_status = 0;
                        }
                    }
                } else if (device_address != 0x0F) {
                    PORTB &= ~(1 << LED12);
                    is_lux = 0;
                }
                device_address = read_device_address();
                counter = 0;
            }
        }
```

Programski blok 4.17. Interrupt rutina

5. Opis programa nadređenog računala

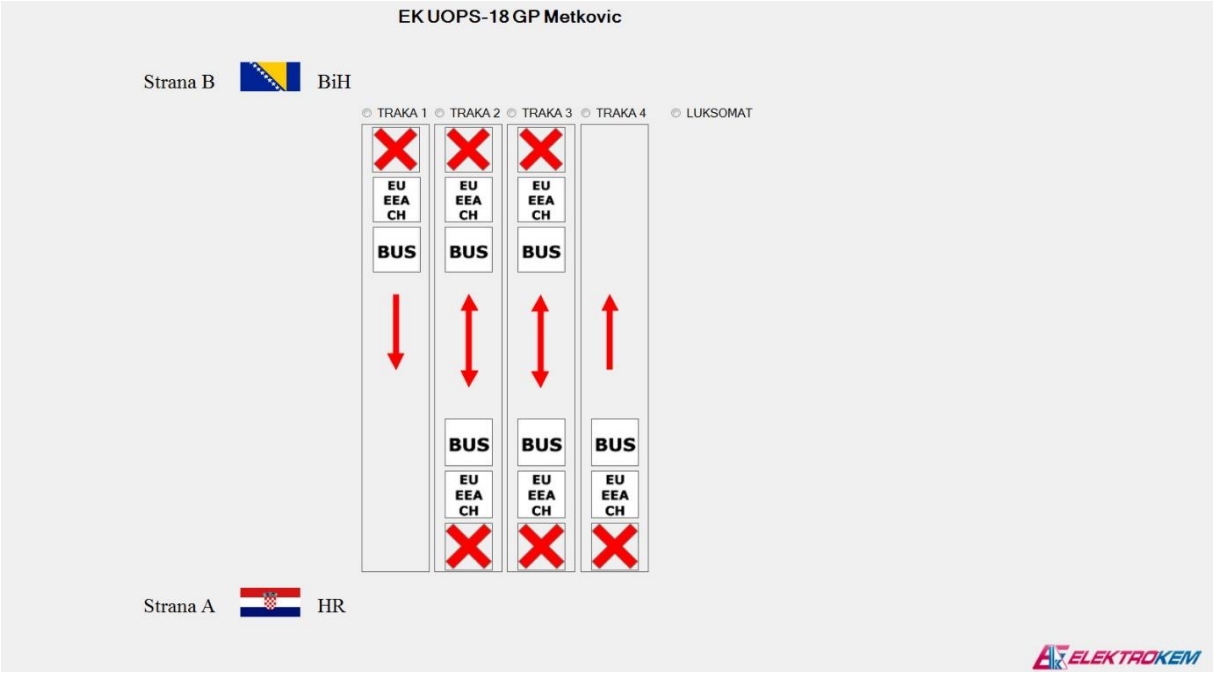
Upravljačka aplikacija upravlja s 18 promjenjivih svjetlosnih znakova toka prometa, spojenih na upravljački ormar. Upravljanje se vrši na tri tipa znakova. Svakom od znaka može se mijenjati prikaz, a mijenja mu se i intenzitet prikaza znakova, ovisno o tome nalazi li se u dnevnom režimu rada ili noćnom. U noćnom režimu rada intenzitet se smanjuje.

Na upravljački ormar može biti priključeno više računala. Sva računala spojena su na istu RS485 sabirnicu. Komunikacija između upravljačkog ormara i računala ostvarena je preko RS485 protokola. Za vezu između RS485 sabirnice i računala korišten je uređaj USB-RS485 model TCSMCNAM3M002P, koji se priključuje na USB ulaz računala. Računalo priključeni uređaj detektira kao Virtual COM Port uređaj. Prilikom pokretanja aplikacije otvara se prozor u kojem je potrebno odabrati COM port koji računalu služi za komunikaciju, što prikazuje slika 5.1.



Slika 5.1. Izgled prozora za postavljanje COM porta

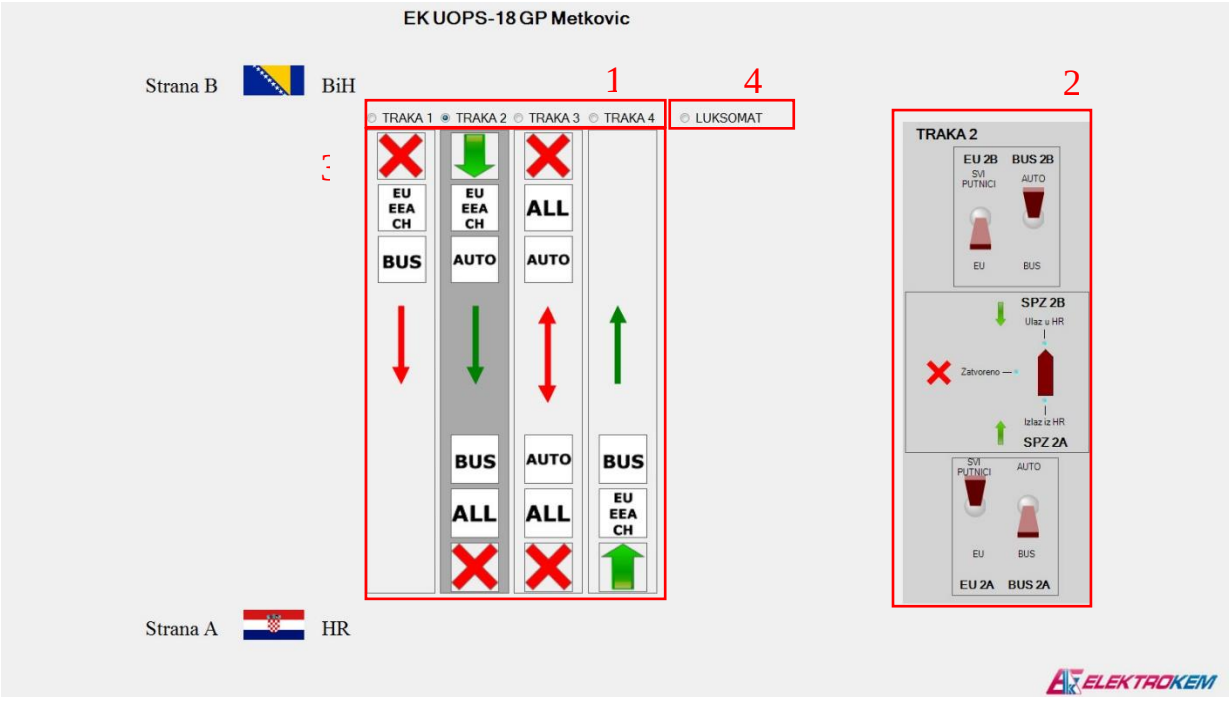
Izgled aplikacije nakon što je uspostavljena komunikacija uređaja USB-RS485 s upravljačkim ormarom prikazan je na slici 5.2. Aplikacija prilikom pokretanja provjerava stanje svih oznaka promjenjivih svjetlosnih znakova toka prometa te osvježava prikaz prema njihovom stanju. Nakon povezivanja računala s upravljačkim ormarom i odabira ispravnog COM port-a prilikom pokretanja aplikacije, omogućeno je daljinsko upravljanje oznakama promjenjivih svjetlosnih znakova toka prometa.



Slika 5.2. Izgled aplikacije nakon pokretanja

Grafičko sučelje aplikacije je na slici 5.3. gdje su brojevima označeni pojedini dijelovi (blokovi) aplikacije:

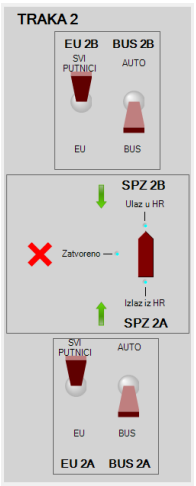
1. Izbornik za odabir trake kojom se želi upravljati
2. Panel za kontrolu oznaka promjenjivih svjetlosnih znakova toka prometa odabrane trake
3. Prikaz trenutno odabrane trake i trenutnog stanja oznaka svih promjenjivih svjetlosnih znakova toka prometa
4. Gumb za odabir kontrole stupnja intenziteta svjetlosti promjenjivih svjetlosnih znakova toka prometa.



Slika 5.3. Grafičko sučelje aplikacije

Unutar izbornika (blok 1.) klikom na naziv trake odabire se traka kojom se želi upravljati, tj. mijenjati stanje oznake promjenjivog svjetlosnog znaka toka prometa. Odabirom trake kojom se želi upravljati otvara se odgovarajući kontrolni panel (blok 2.) i zatamnjuje se traka koja je trenutno aktivna. Kontrolni panel sadrži sklopke kojima je moguće upravljati promjenjivim svjetlosnim znakom toka prometa.

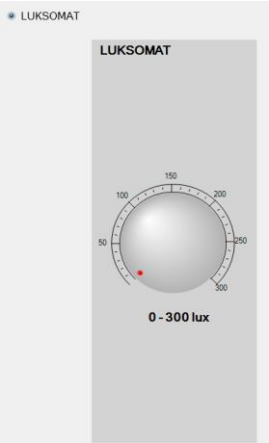
Kontrolni panel (blok 2.) detaljnije je prikazan na slici 5.4. Svaki promjenjivi svjetlosni znak toka prometa ima odgovarajuću sklopku za upravljanje (promjenu oznake). Stanje sklopke mijenja se klikom miša na nju, ili, ukoliko se radi o pomoćnim računalima koja imaju ekran osjetljiv na dodir, pritiskom prsta na sklopku na zaslonu.



Slika 5.4. Kontrolni panel za upravljanje promjenjivim svjetlosnim znakovima toka prometa

Trenutno stanje oznaka znakova prikazano je u bloku 3. Promjenom stanja sklopki na kontrolnom panelu mijenja se stanje na promjenjivom svjetlosnom znaku toka prometa i prikaz trenutnog stanja unutar bloka 3.

Klikom na LUKSOMAT unutar bloka 4. otvara se kontrolni panel za postavljanje praga promjene stupnja intenziteta svjetlosti promjenjivih svjetlosnih znakova toka prometa (slika 5.5.). Prag je moguće postaviti u rasponu od 0 - 300 lux-a. Odabirom praga promjene stupnja intenziteta svjetlosti može se odabrati vrijeme dana u kojem se intenzitet svjetla znakova mijenja iz dnevnog u noćni. Npr. ukoliko se prag postavi na 100 lux-a, u razdoblju u kojem je osvjetljenje okoline manje od 100 lux-a, znakovi će biti u noćnom načinu rada (slabiji intenzitet svjetla), a kad osvjetljenje okoline pređe granicu od 100 lux-a, znakovi rade u dnevnom režimu (jači intenzitet svjetla).



Slika 5.5. Kontrolni panel za luksomat

Upravljanje pomoću aplikacije omogućeno je sve dok je postavljeno stanje '1 – DALJINSKI' na sklopci LOKALNO/DALJINSKI na upravljačkom ormaru. U trenutku prebacivanja sklopke LOKALNO/DALJINSKI u stanje '0 – LOKALNO' aplikacija javlja poruku “Lokalno upravljanje!“ (slika 5.6.) i onemogućuje se upravljanje promjenjivim svjetlosnim znakovima toka prometa preko računala. Stanje oznaka promjenjivih svjetlosnih znakova toka prometa ne gubi se prebacivanjem sklopke LOKALNO/DALJINSKI u stanje '0 – LOKALNO'. Ponovnim prebacivanjem sklopke LOKALNO/DALJINSKI u stanje '1 – DALJINSKI', stanje oznaka na znakovima biti će isto kao što je bilo prije prebacivanja sklopke LOKALNO/DALJINSKI u stanje '0 – LOKALNO' neovisno u kojem su stanju bile oznake u lokalnom načinu upravljanja.

Slika 5.6. Poruka “Lokalno upravljanje“

Aplikacija će automatski prepoznati da je došlo do isključenja upravljačkog ormara. Isključenje ormara signalizira se korisniku porukom “ORMAR JE ISKLJUČEN!“ (slika 5.7.) Isključenje upravljačkog ormara može biti izazvano na dva načina: nestankom napajanja ili isključivanjem preko sklopke UKLJUČENJE. Ponovnim uključenjem upravljačkog ormara poruka sa slike 5.3. nestaje. U slučaju nestanka napajanja sva prethodno postavljena stanja oznaka promjenjivih svjetlosnih znakova toka prometa u aplikaciji se resetiraju (oznake se postavljaju u inicijalno stanje).

Prisilni izlaz iz aplikacije moguće je napraviti preko tipkovnice kombinacijom tipki ALT+F4.

Slika 5.7. Poruka “ORMAR JE ISKLJUČEN!“

5.1. Opis algoritma aplikacije

Aplikacija je napisana u programskom jeziku C#, u programskom okruženju Visual Studio 2008. Nakon testiranja, kompajlirani kod kopiran je na upravljačko računalo, te se automatski pokreće prilikom uključenja računala.

Zbog preglednosti, potpuni kod nije prikazan, prikazani su samo bitni elementi algoritma.

Općenito, funkcija ove aplikacije je prikaz trenutnog stanja svih promjenjivih svjetlosnih znakova toka prometa, kontrola istih, slanjem naredbi programibilnim modulima, te podešavanje praga promjene dnevnog i noćnog režima rada. Npr. ukoliko je na trećem releju programibilnog modula s adresom 2 potrebno promijeniti prikaz znaka, aplikacija preko RS485 sabirnice šalje poruku „$GPIO_SET:2, 4“. U ovoj naredbi, 2 je adresa programibilnog modula, a brojka četiri,

kad se zapiše u binarnom obliku, iznosi 0000 0100, što znači da je treći relejni izlaz u stanju '1', tj. uklopljen.

5.1.1. Struktura BinValue – spremanje stanja releja programibilnog modula

Stanja svih znakova spremljena su u strukture, čiji je kod prikazan u nastavku teksta. Struktura BinValue ima šest bitova, u koje se sprema stanje šest releja u svakom programibilnom modulu. Također, u strukturi su definirane i funkcije setBitValue i getRegisterValue, koje u nju upisuju, odnosno čitaju trenutne vrijednosti iz nje. Stanje ovih šest bitova preko funkcije send(ADRESA) šalje se do programibilnog modula s adresom ADRESA, preko RS485 sučelja.

```
1.         struct BinValue {
2.             public bool bit_0;
3.             public bool bit_1;
4.             public bool bit_2;
5.             public bool bit_3;
6.             public bool bit_4;
7.             public bool bit_5;
8.             public void setBitValue(int value) {
9.                 bit_0 = Convert.ToBoolean(value & 0x01);
10.                bit_1 = Convert.ToBoolean((value >> 1) & 0x01);
11.                bit_2 = Convert.ToBoolean((value >> 2) & 0x01);
12.                bit_3 = Convert.ToBoolean((value >> 3) & 0x01);
13.                bit_4 = Convert.ToBoolean((value >> 4) & 0x01);
14.                bit_5 = Convert.ToBoolean((value >> 5) & 0x01);
15.            }
16.            public int getRegisterValue() {
17.                int reg = 0;
18.                reg = (Convert.ToInt32(bit_5) << 5) | (Convert.ToInt32(bit_4) << 4) | (Convert.
    ToInt32(bit_3) << 3) | (Convert.ToInt32(bit_2) << 2) | (Convert.ToInt32(bit_1) << 1) | (Conver
    t.ToInt32(bit_0));
19.                return reg;
20.            }
21.        }
```

Programski blok 5.1. Spremanje stanja releja programibilnog modula

Npr. aplikacija iz ovog primjera ima 18 znakova kojima se upravlja, pa su u programu definirane tri strukture BinValue, što omogućuje spremanje podataka o 18 releja.

```
1.    BinValue slot1_bin = new BinValue();
2.    BinValue slot2_bin = new BinValue();
3.    BinValue slot3_bin = new BinValue();
```

Programski blok 5.2. Primjer za tri programibilna modula(18 releja)

5.1.2. Funkcija gui_update – osvježavanje trenutnog prikaza stanja programibilnog modula

Nakon pokretanja aplikacije, ona osvježava prikaz stanja svih prometnih znakova. Kod prikazan u nastavku prema sva tri programibilna modula upravljačkog ormara šalje naredbu "$GPIO_GET:ADRESA", sprema stanje BinValue struktura, te poziva funkciju gui_update(slot), koja osvježava prikaz.

```
1.  for (int i = 1; i <= 3; i++) {
2.      string cmd = "$GPIO_GET:" + i.ToString() + "\r";
3.      serialPort1.WriteLine(cmd);
4.      string retval = string.Empty;
5.      retval = serialPort1.ReadLine().TrimEnd();
6.      string[] tokens = retval.Split(new char[] {
7.          ':', ','
8.      });
9.      switch (Convert.ToInt32(tokens[1])) {
10.         case 1:
11.             slot1_bin.setBitValue(Convert.ToInt32(tokens[2]));
12.             GUI_Update(1);
13.             break;
14.         case 2:
15.             slot2_bin.setBitValue(Convert.ToInt32(tokens[2]));
16.             GUI_Update(2);
17.             break;
18.         case 3:
19.             slot3_bin.setBitValue(Convert.ToInt32(tokens[2]));
20.             GUI_Update(3);
21.             break;
22.     }
```

Programski blok 5.3.:Osvježavanje prikaza stanja svih prometnih znakova

Kod funkcije gui_update prikazan je u nastavku. Ova funkcija uspoređuje učitane bitove structure BinValue s predefiniranim bitovima za usporedbu, te, ukoliko se oni razlikuju, osvježava stanje kontrolnih prekidača u aplikaciji. Npr. u dolje prikazanom kodu, sklopka switchLever_strelicaX_4B pridružena je slot1_bin.bit_0 podatku. Ukoliko se on promijenio, čita se njegova vrijednost (true ili false), te se na tu vrijednost podešava i vrijednost prekidača.

```
1.                  if (compare_bin.bit_0 != slot1_bin.bit_0) {
2.      switchLever_strelicaX_4B.Invoke((MethodInvoker) delegate {
3.          if (slot1_bin.bit_0 == true) {
4.              switchLever_strelicaX_4B.Value = true;
5.          } else if (slot1_bin.bit_0 == false) {
6.              switchLever_strelicaX_4B.Value = false;
7.          }
8.      });
9.  }
```

Programski blok 5.4. Primjer za programski blok 5.3.

Za potpuni prikaz stanja grafičkog sučelja, potrebno je poslati još nekoliko upita programibilnim modulima. Naredbama $GET_EEPROM:ADRESA, te $GET_ANA_IN:ADRESA čita se vrijednost o postavkama luksomata koja je zapisana u

EEPROM-u mikrokontrolera, te stanje sklopke RUČNO/AUTOMATSKI. Ovisno o njezinom stanju, prikazuje se ili ne prikazuje poruka “LOKALNO UPRAVLJANJE”.

```
string cmd = "$GET_EEPROM:" + LUX_slot + "\r";
serialPort1.WriteLine(cmd);
string retval = serialPort1.ReadLine().TrimEnd();
string[] tokens = retval.Split(new char[] {
    ':', ','
});
knob1.Value = luksomat_read(Convert.ToDouble(tokens[2]));
string cmd1 = "$GET_ANA_IN:1\r";
serialPort1.WriteLine(cmd1);
string retval1 = serialPort1.ReadLine().TrimEnd();
string[] tokens1 = retval1.Split(new char[] {
    ':', ','
});
if (Convert.ToInt32(tokens1[2]) < 2500) {
    label84.Visible = true;
    Application.DoEvents();
} else {
    label84.Visible = false;
    Application.DoEvents();
}
```

Programski blok 5.5. Čitanje ostalih stanja

5.1.3. valueChanged event – promjena stanja kontrolnih prekidača

Prilikom bilo koje promjene vrijednosti prekidača za kontrolu prikaza u grafičkom sučelju, poziva se valueChanged event, koji obavlja svoju funkciju. Promjena vrijednosti prekidača može se dogoditi ukoliko korisnik sam promijeni vrijednost, ili kao posljedica funkcije gui_update, koja je objašnjena u prethodnom poglavlju. Nakon što je vrijednost prekidača promijenjena, novo stanje šalje se programibilnom modulu funkcijom send(ADRESA). Također, osvježava se grafički prikaz stanja u sučelju aplikacije.

```
1.  private void switchLever_strelicaX_4B_ValueChanged(object sender, Iocomp.Classes.ValueBooleanEven
    tArgse) {
2.      bool UPDATEcomponent = false;
3.      if (e.ValueNew == true) {
4.          if (!IS_UPDATE) {
5.              slot1_bin.bit_0 = true;
6.              if (send(1)) {
7.                  UPDATEcomponent = true;
8.              }
9.          }
10.         if (UPDATEcomponent || IS_UPDATE) {
11.             pictureBox_strelicaX_4B.Image = imageList1.Images[1];
12.             ledArrow_traka4.Value = true;
13.         }
14.     } else if (e.ValueNew == false) {
15.         if (!IS_UPDATE) {
16.             slot1_bin.bit_0 = false;
17.             if (send(1)) {
18.                 UPDATEcomponent = true;
19.             }
20.         }
21.         if (UPDATEcomponent || IS_UPDATE) {
22.             pictureBox_strelicaX_4B.Image = imageList1.Images[0];
23.             ledArrow_traka4.Value = false;
24.         }
25.     }
26. }
```

Programski blok 5.6. Promjena stanja kontrolnih prekidača

5.1.4. Send funkcija – slanje naredbi programibilnom modulu

Send funkcija šalje naredbu programibilnom modulu, u obliku stanja releja koje on treba uklopiti/isklopiti. Svi bitovi pojedine binValue structure spajaju se u jedan bajt, te se on naredbom $GPIO_SET:ADRESA, VRIJEDNOST šalje programibilnom modulu.

```
1.  bool send(int slot) {
2.          bool ret = false;
3.          int val = 0;
4.          try {
5.              if (slot == 1) {
6.                  val = (Convert.ToInt32(slot1_bin.bit_5) << 5) | (Convert.ToInt32(slot1_bin.bi
    t_4) << 4) | (Convert.ToInt32(slot1_bin.bit_3) << 3) | (Convert.ToInt32(slot1_bin.bit_2) << 2
    ) | (Convert.ToInt32(slot1_bin.bit_1) << 1) | Convert.ToInt32(slot1_bin.bit_0);
7.              } else if (slot == 2) {
8.                  val = (Convert.ToInt32(slot2_bin.bit_5) << 5) | (Convert.ToInt32(slot2_bin.bi
    t_4) << 4) | (Convert.ToInt32(slot2_bin.bit_3) << 3) | (Convert.ToInt32(slot2_bin.bit_2) << 2
    ) | (Convert.ToInt32(slot2_bin.bit_1) << 1) | Convert.ToInt32(slot2_bin.bit_0);
9.              } else if (slot == 3) {
10.                 val = (Convert.ToInt32(slot3_bin.bit_5) << 5) | (Convert.ToInt32(slot3_bin.bi
    t_4) << 4) | (Convert.ToInt32(slot3_bin.bit_3) << 3) | (Convert.ToInt32(slot3_bin.bit_2) << 2
    ) | (Convert.ToInt32(slot3_bin.bit_1) << 1) | Convert.ToInt32(slot3_bin.bit_0);
11.             }
12.             string cmd = "$GPIO_SET:" + slot + "," + val.ToString() + "\r";
13.             serialPort1.DataReceived -= serialPort1_DataReceived;
14.             serialPort1.WriteLine(cmd);
15.             string retval = serialPort1.ReadLine().TrimEnd();
16.             if (retval.Equals("$GPIO_SET:OK")) ret = true;
17.             serialPort1.DiscardInBuffer();
18.             serialPort1.DiscardOutBuffer();
19.             return ret;
```

Programski blok 5.7. Slanje naredbi programibilnom modulu

5.1.5. serialPort_DataReceived event – obrada pristiglih podataka

SerialPort_DataReceived event izvršava se prilikom pristizanja podataka s RS485 sabirnice u računalo. Zadaća mu je obraditi pristigle podatke, te zaključiti o kojoj naredbi/podatku se radi i zatim odraditi zadanu funkciju. Pristigli podaci se razdvajaju na dijelove, te se zatim isčitava naredba (koja može biti jedna od onih koje programibilni modul šalje na RS485 sabirnicu), i vrijednosti. Npr. ukoliko je pristigla naredba $OPERATION_MODE, postupak je sljedeći:

```
1.    command = serialPort1.ReadLine().TrimEnd();
2.    string[] tokens = command.Split(new char[] {
3.        ':', ','
4.    });
5.    if (tokens[0].Equals("$OPERATION_MODE")) {
6.        if (tokens[1].Equals("MANUAL")) {
7.            label84.Invoke((MethodInvoker) delegate {
8.                label84.Visible = true;
9.            });
10.       } else if (tokens[1].Equals("AUTO")) {
11.           label84.Invoke((MethodInvoker) delegate {
12.               label84.Visible = false;
13.           });
14.       }
15.   }
```

Programski blok 5.8. Obrada pristiglih podataka

Naredbom serialPort1.ReadLine učitava se naredba u varijablu command. Ona se zatim rastavlja na dijelove (tokens), naredbom command.Split(). Razdvajanje se vrši definiranjem uvjeta, tj. definiranjem znakova prije i poslije kojih će se naredba razdvojiti. U ovom slučaju, naredba se razdvaja na znakovima ‘:’ i ‘,’. Tokens[0] zbog toga ima vrijednost “$OPERATION_MODE”, a tokens[1] vrijednost “AUTO” ili “MANUAL”. Ovisno o tom tekstu prikazuje se ili sakriva label84, što je ustvari tekst sa porukom “LOKALNO UPRAVLJANJE”, koji se prikazuje na zaslonu aplikacije.

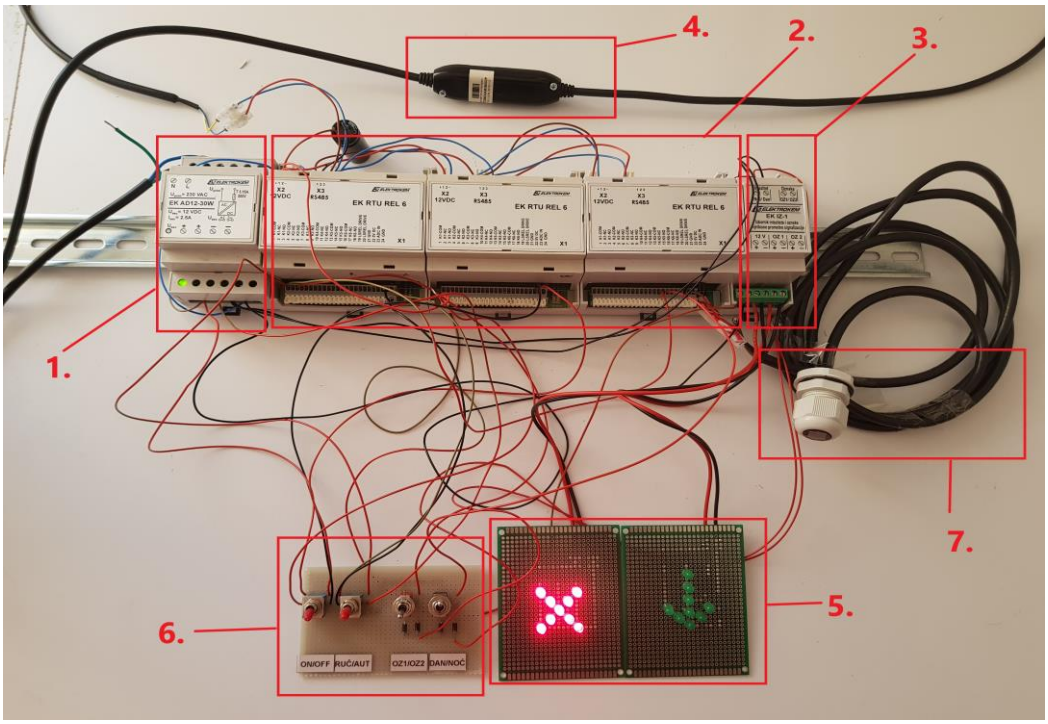
6. Model sustava upravljanja promjenjivom signalizacijom graničnih prijelaza

Za praktični dio ovog Završnog rada projektiran je pojednostavljeni model sustava upravljanja promjenjivom signalizacijom graničnih prijelaza.

Željeni cilj je upravljati znakom sa dvije oznake preko korisničke aplikacije i preko prekidača na modulu napravljenom za tu svrhu.

Električki su povezani svi moduli koji su neophodni. Kao što se vidi sa slike 6.1. to su:

1. AC/DC pretvarač
2. Programibilni logički sklopovi (3 komada)
3. Modul izbornika
4. USB/RS485 konverter
5. Oznake G09 (crveni križ) i G10 (zelena strelica) izvedene sa LED diodama
6. Modul sa prekidačima za odabir:
   a) ON/OFF – uključenje/isključenje
   b) RUČ/AUT – ručno/automatski
   c) OZ1/OZ2 – odabir oznake1 ili oznake 2 (križ ili strelica)
   d) DAN/NOĆ
7. Fotoosjetljivi otpornik koji odrađuje funkciju luksomata



Slika 6.1. Pojednostavljeni model sustava upravljanja promjenjivom signalizacijom

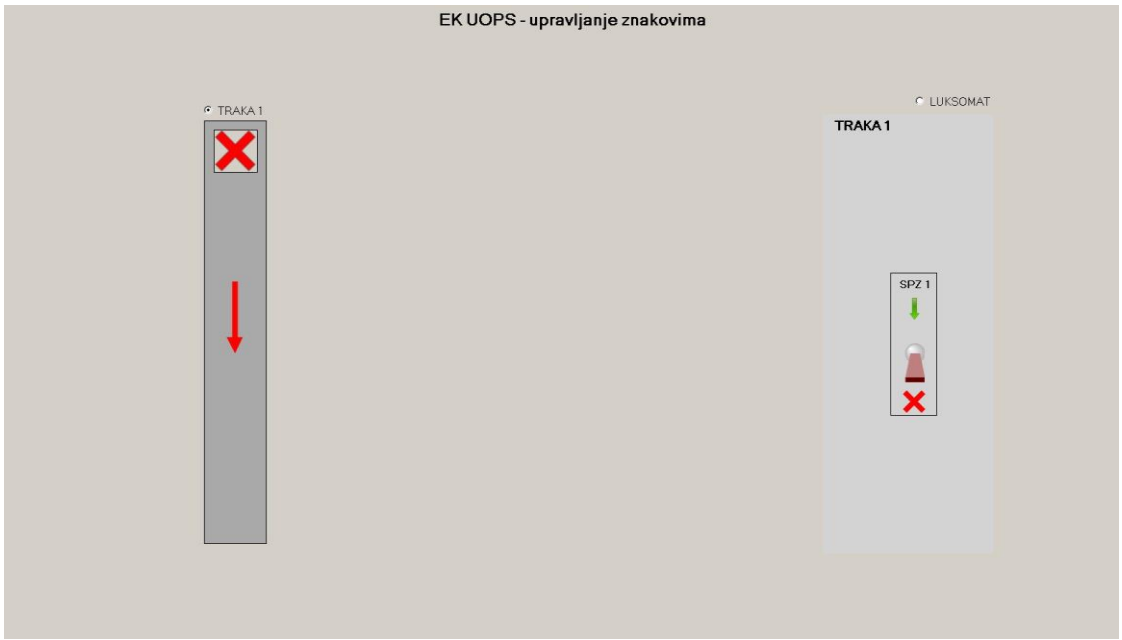
7. Programiranje znaka s dvije oznake

Nakon povezivanja pojednostavljenog modela sustava upravljanja promjenjivom signalizacijom graničnih prijelaza, izvršeno je potrebno programiranje. Programiranje programibilnog modula izvršeno je pomoću programskog paketa Atmel Studio 6.2, u programskom jeziku C.

Postojeći izvorni (source) kod upravljanja promjenjivom signalizacijom jednog od sustava koji je ugrađen na graničnom prijelazu prenamijenjen je za praktični dio ovog Završnog rada. Kako bi se to realiziralo urađeno je slijedeće:

- definirani su svi pinovi mikrokontrolera koji se koriste
- pridružene su sve potrebne datoteke s raznim funkcijama koje su opisane u ovom radu
- implementirane su funkcije za komunikaciju
- u main dijelu programa inicijalizirana su stanja svih pinova, pozvane su funkcije koje inicijaliziraju komunikacijske kanale, čitaju adrese uređaja, te su iskorištene ostale rutine potrebne za početnu inicijalizaciju mikrokontrolera.

Kako bi se dobio željeni izgled upravljačke aplikacije kojom se upravlja ovim modelom, bile su potrebne prilagodbe. Za to su korištene postojeće funkcije i dijelove kodova koje su detaljnije opisane na početku ovog Završnog rada, a programirane su u programskom jeziku C# (programsko okruženje Visual Studio 2008). Nakon testiranja i kompajliranja na računalu se dobio prikaz koji se vidi na slici 7.1.



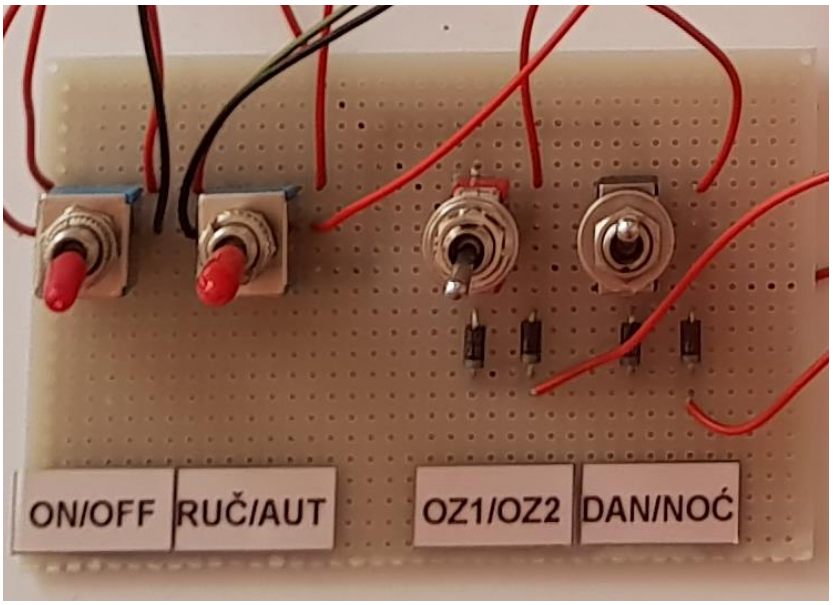
Slika 7.1. Izgled aplikacije praktičnog rada nakon pokretanja

8. Test pojednostavljenog modela sustava upravljanja promjenjivom signalizacijom

Nakon izrade pojednostavljenog modela sustava upravljanja promjenjivom signalizacijom graničnih prijelaza i nakon izrade korisničke aplikacije dobivena je potpuna i ispravna funkcija ovako pojednostavljenog modela.

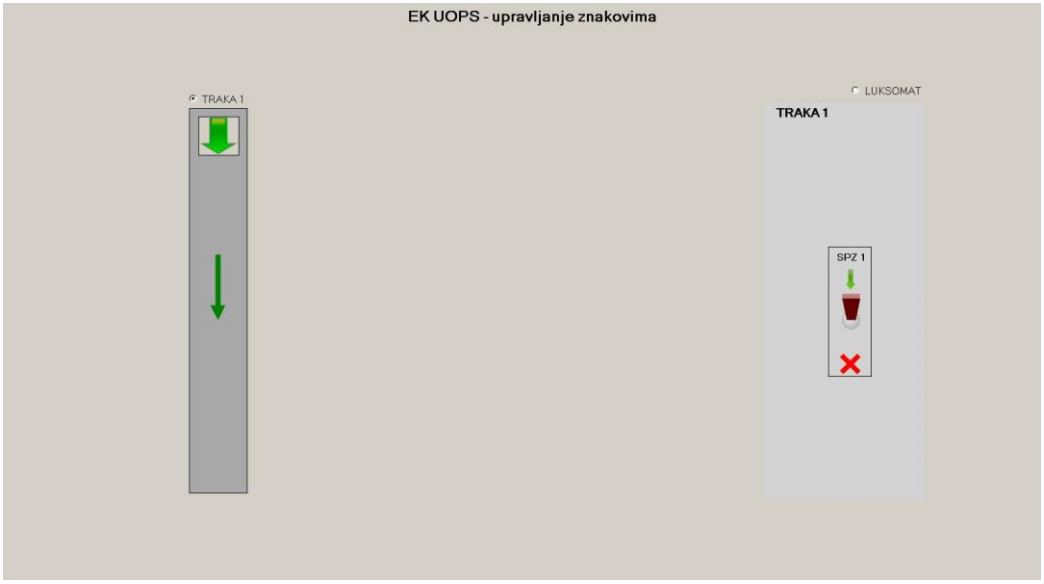
Ako se na modulu sa prekidačima (slika 8.1.) prekidač RUČ/AUT stavi u položaj RUČ, tada prekidač OZ1/OZ2 u jednom položaju uključuje jednu oznaku, a u drugom položaju prekidač OZ1/OZ2 uključuje drugu oznaku. Prekidač DAN/NOĆ u jednom položaju osigurava jači intenzitet oznake koja je uključena, a u drugom položaju slabiji intenzitet.

U ovom modu rada na upravljačkoj aplikaciji je prikaz koji se vidi na slici 8.2.



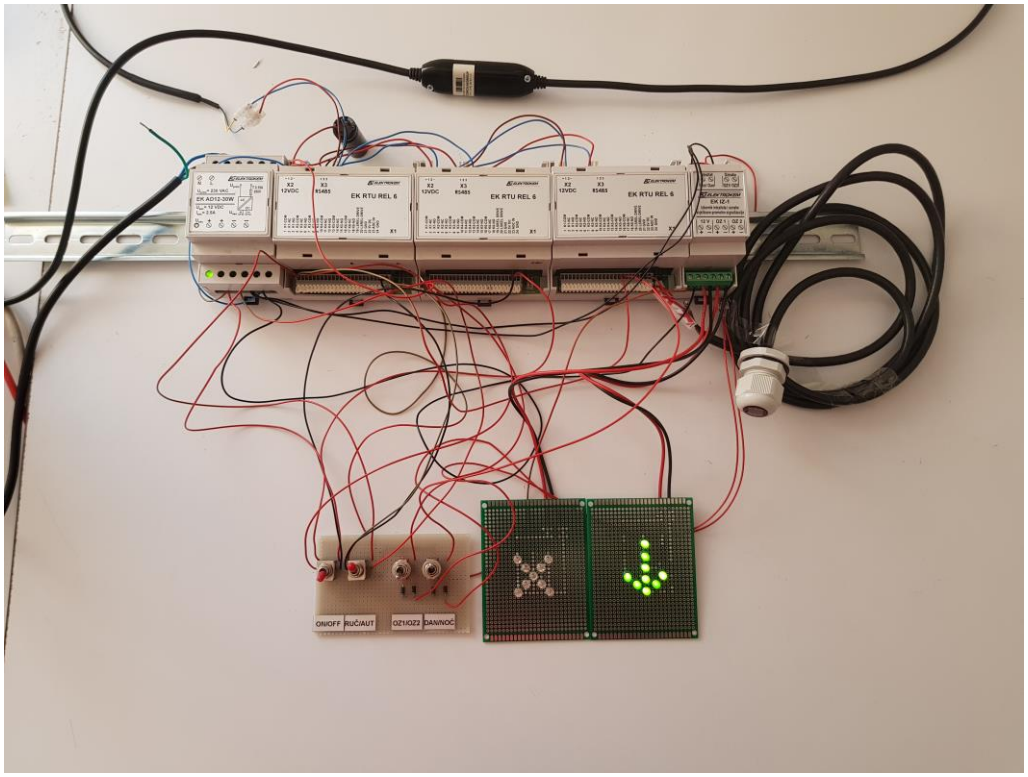
Slika 8.1. Modul sa prekidačima

Ako se na modulu sa prekidačima (slika 8.1.) prekidač RUČ/AUT stavi u položaj AUT, tada se odabirom oznake i intenzitetom oznake upravlja samo iz upravljačke aplikacije na računalu. Tako primjerice nakon početnog prikaza (slika 5.6.), odabirom TRAKA 1 i klikom na prikaz sklopke na aplikaciji dobije se prikaz koji se vidi na slici 8.2.



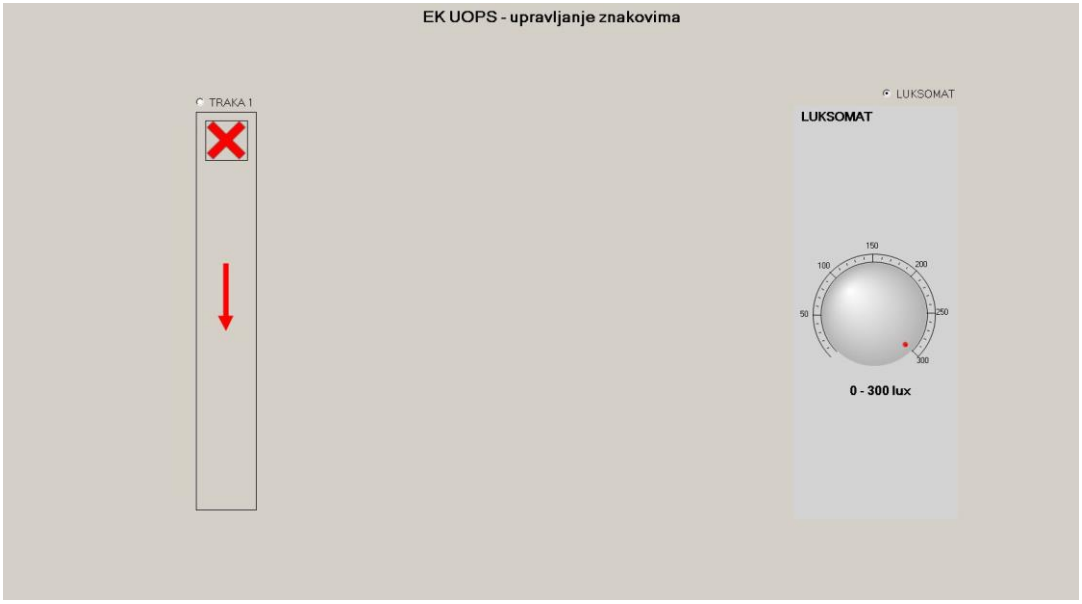
Slika 8.2. Prikaz na upravljačkoj aplikaciji nakon odabira druge oznake

Na pojednostavljenom modelu (slika 8.3.) vidi se kako je sada uključena druga oznaka (prethodno je bio upaljen crveni križić).



Slika 8.3. Uključenje druge oznake nakon odabira na upravljačkoj aplikaciji

Odabirom LUKSOMAT i podešenjem na upravljačkoj aplikaciji (slika 8.4.), te nakon prekrivanja fotoosjetljivog otpornika smanji se intenzitet svjetla trenutno upaljene oznake.



Slika 8.4. Prikaz na upravljačkoj aplikaciji nakon odabira LUKSOMAT

Praktičnim radom ovog Završnog rada prikazano je kako korištenjem elektroničkih modula ostvariti upravljanje sa dislociranim izvršnim elementima. Za to je bilo potrebno implementirati odgovarajuća hardverska i softverska rješenja koja su objašnjena u ovom Završnom radu.

9. Zaključak

Upravljanje promjenjivom signalizacijom graničnih prijelaza sa centralnog mjesta i/ili drugih lokacija omogućuje upravljanje odabirom pojedine oznake na prometnom znaku, te intenzitetom svjetla kojim će ta oznaka biti prikazana.

Upravljanje se vrši pomoću upravljačkog ormara koji sadrži EK RTU-6 relejne module čija je funkcija prikaz željenih znakova te mijenjanje njihovog intenziteta ovisno o tome je li aktivan dnevni ili noćni režim rada. Ove promjene mogu se ručno odabirati pomoću sklopki na ormaru, ili daljinski preko računala sa instaliranom upravljačkom aplikacijom. Upravljačka aplikacija prikazuje trenutno stanje svih znakova na graničnom prijelazu, te se odabirom pojedine trake i znaka može mijenjati njegovo stanje.

Problem koji se pojavio prilikom razvoja ovakvog sustava je nestanak napajanja na graničnom prijelazu, prilikom kojeg se postavke svih znakova resetiraju. Početni prikaz znakova koji omogućuju prolaz kroz pojedine trake je znak „X“, tako da se ne može dogoditi istovremeni ulazak vozila s obje strane graničnog prijelaza, ali nakon svakog nestanka napajanja, potrebno je opet podešavati stanje svih znakova. Ovaj problem može se riješiti ili spremanjem stanja znakova u upravljačko računalo nakon svake promjene, ili ugradnjom besprekidnog napajanja, koje onemogućava isključenje ormara i računala prilikom nestanka napajanja.

Uvođenjem upravljanja promjenjivom signalizacijom graničnih prijelaza znatno je olakšan i ubrzan promet na graničnim prijelazima, što više dolazi do izražaja kod velikih graničnih prijelaza koji imaju mnogo prometnih traka. Također, nije potrebno mnogo korisnika koji imaju ovlasti upravljanja. Korisnik na jednom mjestu može odrediti koje prometne trake i sa koje strane državne granice će biti prolazne i za koju vrstu vozila, a da pritom ima prikaz kompletnog graničnog prijelaza, odnosno da sa jednog mjesta ima uvid koje trake su prolazne, a koje zatvorene.

Kristijan Grmovšek

_____

(vlastoručni potpis)

U Varaždinu, 09.10.2018. godine

10. Literatura

[1] Elektrokem d.o.o.: Upravljački ormar promjenjive signalizacije GP, tip: EK UOPS-18 GP-METKOVIĆ, Tehnička dokumentacija, Zagreb, 2016.

[2] https://www.microchip.com/wwwproducts/en/AT90USB162 (07.08.2018)

[3] https://www.microchip.com/wwwproducts/en/MCP23S08 (08.08.2018)

[4] http://www.ti.com/product/TPL9202# (05.08.2018.)

[5] http://www.ti.com/product/ADS7883 (03.08.2018.)

[6] https://www.maximintegrated.com/en/products/interface/transceivers/MAX485.html (02.08.2012)

[7] https://en.wikipedia.org/wiki/Serial_Peripheral_Interface (05.08.2018)

[8] http://www.micromatic.hr/download/Serijska%20komunikacija%20mikrokontrolera.pdf (06.08.2018)

Popis slika

Popis programskih blokova

11. Prilozi

11.1. parse_cmd

```
1.          // // parse naredbe //
2.          private void serialPort1_DataReceived(object sender, System.IO.Ports.SerialDataReceived
    EventArgs e) {
3.              string command = string.Empty;
4.              try {
5.                  do {
6.                      command = serialPort1.ReadLine().TrimEnd(); // čitanje poruke
7.                      Debug.WriteLine(command);
8.                      string[] tokens = command.Split(new char[] {
9.                          ':', ','
10.                     });
11.                     if (tokens[0].Equals("$GPIO_SET")) // provjera naredbe
12.                     {
13.                         if (tokens[1].Equals("1") || tokens[1].Equals("2") || tokens[1].Equals(
    "3") || tokens[1].Equals("4") || tokens[1].Equals("15")) // provjera ispravnosti adrese PLC-a
14.                         {
15.                             if (Convert.ToInt32(tokens[2]) <= 127 && Convert.ToInt32(tokens[2])
     >= 0) // provjera ispravnosti stanja
16.                             {
17.                                 qpc.is_qureyRun = true; // flag za update sučelja
18.                                 qpc.slot = Convert.ToInt32(tokens[1]); // adresa PLC-a kojem se
    mijenja stanje
19.                                 qpc.value = Convert.ToInt32(tokens[2]); // stanje odabranog
    PLC-a
20.                             }
21.                         } else if (tokens[1].Equals("OK"))
22.                         {
23.                             if (qpc.is_qureyRun)
24.                             {
25.                                 switch (qpc.slot) // update GUI
26.                                 {
27.                                     case 1:
28.                                         compare_bin = slot1_bin;
29.                                         slot1_bin.setBitValue(qpc.value);
30.                                         GUI_Update(1);
31.                                         break;
32.                                     case 2:
33.                                         compare_bin = slot2_bin;
34.                                         slot2_bin.setBitValue(qpc.value);
35.                                         GUI_Update(2);
36.                                         break;
37.                                     case 15:
38.                                         compare_bin = slot15_bin;
39.                                         slot15_bin.setBitValue(qpc.value);
40.                                         GUI_Update(15);
41.                                         break;
42.                                     default:
43.                                         break;
44.                                 };
45.                                 qpc.is_qureyRun = false;
46.                             }
47.                         }
48.                     } // lokalno ili automatski
49.                     if (tokens[0].Equals("$OPERATION_MODE")) {
50.                         if (tokens[1].Equals("MANUAL")) {
51.                             label_lokalno_upravljanje.Invoke((MethodInvoker) delegate {
52.                                 label_lokalno_upravljanje.Visible = true;
53.                             });
54.                         } else if (tokens[1].Equals("AUTO")) {
55.                             label_lokalno_upravljanje.Invoke((MethodInvoker) delegate {
```

```
                                label_lokalno_upravljanje.Visible = false;
                            });
                        }
                    }
                    if (tokens[0].Equals("$NETWORK")) {
                        if (tokens[1].Equals("OFF")) {
                            label_lokalno_upravljanje.Invoke((MethodInvoker) delegate {
                                label_lokalno_upravljanje.SendToBack();
                            });
                            label_iskljucen_ormar.Invoke((MethodInvoker) delegate {
                                label_iskljucen_ormar.Visible = true;
                            });
                        } else if (tokens[1].Equals("ON")) {
                            label_lokalno_upravljanje.Invoke((MethodInvoker) delegate {
                                label_lokalno_upravljanje.BringToFront();
                            });
                            compare_bin = slot1_bin;
                            slot1_bin.setBitValue(0);
                            GUI_Update(1);
                            compare_bin = slot2_bin;
                            slot2_bin.setBitValue(0);
                            GUI_Update(2);
                            compare_bin = slot15_bin;
                            slot15_bin.setBitValue(0);
                            GUI_Update(15);
                            label_iskljucen_ormar.Invoke((MethodInvoker) delegate {
                                label_iskljucen_ormar.Visible = false;
                            });
                        }
                    }
                } while (command != string.Empty);
            } catch (Exception) {;
            }
        }
```

Programski blok 11.1. parse_cmd

Sveučilište Sjever

SVEUČILIŠTE SJEVER

IZJAVA O AUTORSTVU I SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, Kristijan Grmovšek pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor završnog rada pod naslovom Upravljanje promjenjivom signalizacijom graničnih prijelaza te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student:

Kristijan Grmovšek

(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, Kristijan Grmovšek neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog rada pod naslovom Upravljanje promjenjivom signalizacijom graničnih prijelaza čiji sam autor.

Student:

Kristijan Grmovšek

(vlastoručni potpis)