SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni diplomski studij elektrotehnike

BEŽIČNA RFID AUTORIZACIJA KORISNIKA

Diplomski rad

Goran Horvat

Osijek, 2010

SADRŽAJ

1. UVOD

Od vremena kada su se pojavili zaštićeni sustavi pojavila se i potreba za utvrđivanjem identiteta korisnika. U prošlosti se taj postupak obavljao na razne načine: Legitimacijom korisnika uz pomoć iskaznica, popisivanjem u registre, magnetskim karticama i drugo, dok se u novije vrijeme masovno koristi sustav bezkontaktnog pristupa tj. pristup pomoću RFID (eng. *Radio Frequency IDentification*) tehnologije. RFID tehnologija predstavlja novo područje sustava autorizacije korisnika jer pruža mnoge nove mogućnosti koje prije nisu bile moguće, poput koncepta „pametnih kartica“ i drugih svojstava. Generalno rečeno, RFID kartice zamjenjuju prijašnje magnetske kartice na isti način kao što su USB memorije (eng. USB *Tumb Drive*) zamijenile diskete (eng. FDD - *Floppy Disk Drive*).

U današnjim metodama utvrđivanja identiteta korisnika mnogi sustavi koriste RFID tehnologiju, u topologiji pri kojoj su čitači izravno vezani na računala, koja su povezana na lokalnu mrežu (eng. LAN - *Local Area Network*). Time dolazi do problema autorizacije korisnika na višestrukim mjestima jer takav koncept zahtjeva veći broj računala i postojeću mrežnu infrastrukturu, što bi značajno povećalo cijenu takvog projekta. U ovom radu razmotren je drugačiji pristup opisanom problemu. Pristup se bazira na uporabi bežičnog povezivanje RFID čitača s glavnim čvorom, koji obrađuje bazu podataka i vezan je prema LAN mreži.

Sustav bežične RFID autorizacije korisnika osmišljen je kao centralizirani sustav u kojemu je srce sustava glavni čvor (eng. *Master Node*), koji je bežično vezan s ostalim pristupnim RFID točkama (eng. *Slave Nodes*). Bežična veza glavnog čvora i pristupnih čvorova zamišljena je na principu ZigBee tehnologije baziranoj na IEEE 802.15.4 standardu za osobne bežične mreže (eng. *Low-Rate Wireless Personal Area Networks*).

Princip rada sustava bežične RFID autorizacije je sljedeći: Korisnik pristupa pristupnoj točki u želji dokazivanja identiteta sustavu koji ga to potražuje. Bezkontaktnu karticu prislanja čitaču koji očitava jedinstveni serijski broj dane kartice i traži od korisnika autorizaciju na način da korisnik unosi svoj jedinstveni pristupni kod ili PIN (eng. *Personal Identification Number*). Nakon dobivenih podataka od strane korisnika pristupni čvor šalje podatke glavnom čvoru i očekuje odgovor o stanju autorizacije korisnika. Za to vrijeme, na glavnom se čvoru dobivene informacije od pristupne točke analiziraju kroz bazu podataka. Utvrđuje se da li korisnik postoji u sustavu prema jedinstvenom serijskom broju kartice, i da li je njegov pristupni kod ispravan. Ako je autorizacija uspješna glavni čvor javlja pristupnom čvoru da je korisnik uspješno uveden

u sustav. Na glavnom čvoru, u bazu podataka unosi se korisnik po jedinstvenom serijskom broju kartice, vremenu i datumu pristupa. Sa pogleda autorizacije proces je završen. U slučaju da sistem administrator želi pristupiti podacima o autorizaciji korisnika i samoj bazi korisnika, tada pomoću softverske aplikacije pristupa glavnom čvoru koristeći standardnu Ethernet lokalnu mrežu, na koju je glavni čvor stalno priključen. Time je završen postupak prijave korisnika u sustav i nadzora korisnika od strane administratora. U teoretskom razmatranju prilično jednostavno, u izvedbi prilično kompleksno.

Problem koji se rješava uključuje: Osmišljavanje RFID čitača (uz postojeće komponente) na način da se dobiveni signali softverski demoduliraju i obrađuju u postupku pribavljanja serijskog broja s bezkontaktnih kartica. Čitač je integriran u pristupni čvor i mora imati funkciju pribavljanja jedinstvenih serijskih brojeva pohranjenih u RFID kartice. Također, čitač mora imati sposobnost provjere integriteta dobivenih podataka kroz provjeru redundancije.

Na strani pristupnog čvora mora se osmisliti princip unosa osobnog koda (eng. PIN - *Personal Identification Number*). Predloženi način je uporabom LCD ekrana osjetljivog na dodir (eng. *Touch Screen*), te je stoga potrebno osmisliti programsku podršku spomenutom ekranu, izraditi grafičko korisničko sučelje i omogućiti funkciju osjetljivosti na dodir.

Cijeli je sustav potrebno povezati u bežičnu mrežu za što su zasluženi ZigBee bežični komunikacijski moduli, prisutni na strani glavnog i pristupnog čvora. Potrebno je omogućiti glavnom i pristupnim čvorovima međusobnu komunikaciju, s razmjenom informacija potrebnih za uspješnu autorizaciju korisnika u sustavu.

Nadalje, na strani glavnog čvora treba dizajnirati bazu podataka s bazom korisnika, kojima je pristup u sustav dozvoljen i bazom pristupa za svaku pristupnu točku. Treba odabrati metodu pohranjivanja podataka baze podataka te ostvariti pristup spomenutoj bazi (od strane bežične mreže kao i od strane lokalne mreže).

Treba ostvariti pristup lokalnoj *Ethernet* mreži i osigurati mogućnost komunikacije s klijentskom aplikacijom (bilo gdje na lokalnoj mreži). Potrebno je osmisliti protokol i metodu razmjene informacija između glavnog čvora i klijentske aplikacije.

Na posljetku, potrebno je osmisliti klijentsku aplikaciju koja ima svojstva manipulacije bazom podataka pohranjenom na glavnom čvoru, mogućnost pregleda pristupa korisnika na svim pristupnim točkama i mogućnosti dodavanja novih korisnika u sustav. Klijentsku aplikaciju treba izraditi koristeći razvojni sustav *LabView* Američke tvrtke „National Instruments“.

Glavni čvor i pristupne čvorove izraditi uz postojeće razvojne sustave srpske tvrtke „mikroElektronika“ bazirane na *Atmel* AVR mikro upravljačima (eng. *Micro-Controller*). Koristiti dodatne module za ostvarivanje svih željenih funkcija.

Tok razvoja bežične RFID autorizacije korisnika prolazi kroz nekoliko faza. U ovom radu, drugo i treće poglavlje predstavlja teorijsku razradu svih spomenutih tehnologija, od RFID tehnologije, kroz povijest, arhitekturu, radio frekvencijski prijenos podataka, protokole i sigurnost, pa do ZigBee tehnologije bežičnog prijenosa, kroz tipove uređaja, protokolni stog, formiranje ZigBee mreže i samih ZigBee bežičnih modula zvanih XBee.

Četvrto poglavlje rada bazira se na arhitekturi samog sustava Bežične RFID autorizacije korisnika koji prolazi kroz arhitekture mikro upravljača i osmišljanje arhitektura glavnog čvora i pristupnih čvorova. Ovdje su spomenuti svi dodatni moduli koji se koriste za ostvarivanje funkcija čvorova kao što su: RFID čitač, LCD ekran osjetljiv na dodir, XBee moduli za bežičnu komunikaciju, Ethernet modul za povezivanje s LAN mrežom i mnogi drugi.

U petom poglavlju rada govori se o programskom razvoju glavnog i pristupnog čvora prolazeći kroz cijeli tok osmišljavanja sustava, od početka koji je započeo s RFID čitačem pa sve do povezivanja cijele strukture u sustav bežične RFID autorizacije korisnika. Kada govorimo o programskom razvoju čvorova, onda treba naglasiti da je to postupak osmišljavanja programskog koda koji se izvodi na mikro upravljačima. Mikro upravljači predstavljaju „srce“ glavnog i pristupnog čvora. Na posljetku opisana je izrada korisničke aplikacije koristeći razvojni sustav *LabView*.

2. TEORIJSKE OSNOVE RFID TEHNOLOGIJE

Tehnologija radio frekvencijske identifikacije (RFID – *Radio Frequency IDentification*) postoji već desetljećima. RFID tehnologija se koristi za stotine različitih primjena, od alarmnih uređaja, automatskih naplata cestarina (ENC) pa sve do najkompleksnijih primjena. Što su uređaji kompleksniji to sadrže više funkcija i najčešće su povezani na računala ili na mreže, a frekvencije koje se koriste sežu od 100kHz pa sve do 10GHz [1]. Osnovna arhitektura RFID tehnologije općenito sadrži dvije komponente: čitač i transponder. Pod pojmom čitač podrazumijeva se uređaj koji ima funkciju dobavljanja informacija zapisanih u transponderu, ili često nazivan i *tag* (u nastavku teksta *tag*), dok je *tag* elektronički uređaj koji obavlja funkciju pohranjivanja podataka na postojanu memoriju i razmjenu zapisanih podataka sa čitačem [1]. Zapisani podaci mogu se razlikovati po količini i sadržaju, od najjednostavnijih jedinstvenih identifikatora (eng. *Unique Serial Number*) pa sve do čitavih baza podataka. Postoje mnoge vrste čitača i *tagova* koji su opisani kasnije u radu.

2.1. Povijest RFID tehnologije

Rani početak rada na RFID tehnologiji seže sve do 1948. godine, kada je Harry Stockman objavio članak „Communication by Means of Reflected Power“ [2] gdje je izjavio: “Potrebno proći još mnogo vremena i istraživanja prije nego komunikacija pomoću reflektirane snage bude moguća“ [2]. Proći se još 30 godina dok se njegova vizija ispuni, jer tehnologije kao što su tranzistor, integrirani krug i mikroprocesor još nisu postojale.

1950-te godine bile su era istraživanja RFID tehnologije čemu su prethodili izumi radara i radio komunikacije. Mnoge RFID tehnologije su istraživane, kao npr. *tagovi* velikog dometa za identifikaciju „Prijatelj ili neprijatelj“, koja bi se koristila za vojne avione.

1960-te godine bile su dobra podloga za nagli razvoj RFID tehnologije u 1970-tim godinama. Mnogi radovi su objavljivani na temu daljinskog upravljanja radio valovima i napajanja uređaja pomoću radio valova. Komercijalne primjene RFID tehnologije počele su još 1960-ih godina, ali RFID sustavi tog vremena koji su mogli pohraniti veću količinu informacija, bili su velikih dimenzija i stoga nisu imali komercijalnu prednost.

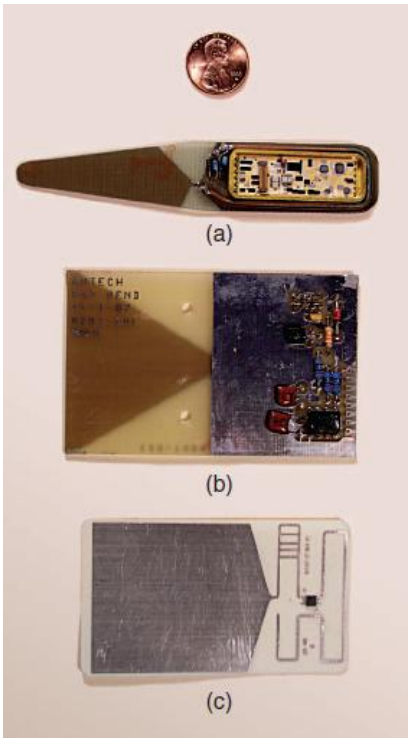
Era 1970-ih godina pružala je povoljno tlo za daljnji razvoj RFID tehnologije. Rani razvoj u tom području vršio je američki *Los Alamos Scientific Laboratory*, čiji je rad prezentiran

pod nazivom „Radio telemetrija kratkog dometa u svrsi elektronske identifikacije pomoću moduliranog povratnog radijskog signala,“ godine 1975. [2] Ovaj rad označio je početak vrlo praktičnim pasivnim *tagovima*, s radnim dometom od desetak metara. U nadolazećim godinama, tehnologija izrade *tagova* kontinuirano je usavršavana imajući za rezultat manje dimenzije i veću funkcionalnost. Ključ ovih poboljšanja ležao je u uporabi CMOS sklopova koji imaju malu potrošnju energije.

Nadolazeće desetljeće, 1980-ih godina, postalo je desetljeće potpune implementacije RFID tehnologije, no interes za samu tehnologiju varirao je u različitim dijelovima svijeta. Najveći interes pokazao se u SAD-u i to za primjenu u transportu i u sustavima pristupa (eng. *Access Cards*). U Europi, primjene su više bile orijentirane na sustave kratkog dometa za praćenje životinja, industrijskih i poslovnih primjena i naplata cestarina. Veliki čimbenik naglog razvoja RFID tehnologije je bio i razvoj osobnog računala koji je omogućio ekonomski isplativo prikupljanje i obradu podataka iz RFID sustava. Prvi komercijalni sustavi 1980-ih godina pojavili su se u Europi i to u Norveškoj 1987. godine kao sustavi el. naplate cestarine [2].

Devedesete godine postale su značajno desetljeće za RFID tehnologiju, nakon što se ugledala prednost uporabe takvih sustava za naplatu cestarina. Prva autocesta na svijetu, koja je imala elektronsku naplatu cestarine otvorena je u Oklahomi SAD 1991. godine, a prednost je bila da su automobili mogli prolaziti naplatu cestarine u punoj brzini, bez zaustavljanja. Novi sustavi razvijeni su od strane američke tvrtke Texas Instruments, koji su služili za pokretanje osobnih automobila (imobilizatori). Daljnjim razvojem CMOS tehnologije RFID sustavi su prešli potpuno u domenu integriranih sklopova.

U današnje vrijeme, RFID *tagovi* postaju toliko malih dimenzija (sastojeći se samo od CMOS integriranog sklopa i antene) da je *tagove* moguće ugraditi u samoljepljive naljepnice. EEPROM memorija ostala je korištena kao izbor permanentne memorije. Tempo razvoja RFID sustava i dalje raste i budućnost izgleda veoma obećavajuća za razvoj ove tehnologije.

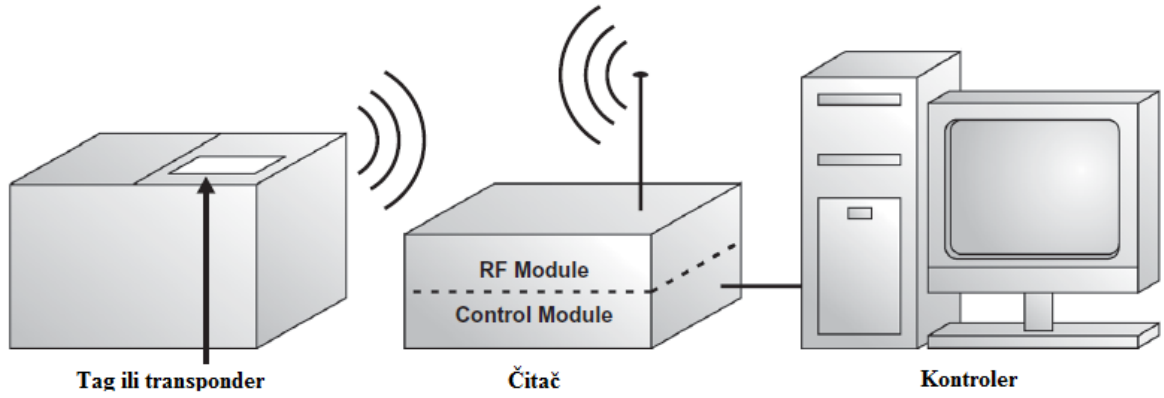


Sl. 2.1. Evolucija RFID *tagova*. (a) 12 bitni read only *tag*, 1976g. (b) 128 bitni *tag*, 1987g. (c) 1024 bitni *tag* sa opcijom čitanja i pisanja, 1999g [2]

2.2. Arhitektura RFID sustava

Kao što je mnogo puta spomenuto, RFID sustav koristi radio komunikaciju da bi jedinstveno identificirao spomenuti objekt ili osobu. Kod RFID sustava postoje tri osnovne komponente [1], prikazane na Slici 2.1:

1. *Tag*, koji se sastoji od poluvodičkog integriranog sklopa, antene, a ponekad i baterije.
2. Čitač, koji se sastoji od antene, radio frekvencijskog (RF) sklopovlja i ostalih upravljačkih elektroničkih modula.
3. Kontroler ili upravljač (eng. *Controller*), čiju ulogu često preuzima osobno računalo ili radna stanica koja odrađuje poslove baze podataka i upravljanja preko softvera.



Sl. 2.1. Arhitektura RFID sustava [1]

Tag i čitač razmjenjuju informacije radio frekvencijskom vezom. Kada objekt koji sadrži RFID *tag* dospije u područje čitača, čitač signalizira tagu da pošalje pohranjene podatke. Tagovi mogu sadržavati mnoge različite informacije o objektima na koje su pričvršćeni (eng. *Tagged*) kao npr. serijske brojeve, vremenske žigove, instrukcije za podešavanje i znatno više. Nakon što je čitač primio podatke s *taga*, ti se podaci šalju kontroleru preko standardne mreže, kao lokalna mreža ili Internet. Kontroler dobivene informacije koristi za razne svrhe, npr. da vodi inventar objekata u bazi podataka, ili može određeni objekt preusmjeriti na drugi dio postrojenja, u nekoj proizvodnji.

RFID sustav se može sastojati od većeg broja čitača postavljanih duž skladišta ili pogona, ali tada su svi čitači vezani na jedan kontroler. Slično spomenutom, jedan čitač može komunicirati s većim brojem *tagova* simultano, a u današnjoj tehnologiji moguće je komunicirati brzinom od 1000 *tagova* u sekundi, s točnosti od 98% [1]. Na posljetku, RFID *tagovi* su postali

toliko razvijeni da se mogu pričvrstiti na gotovo sve, od palete u pogonu neke proizvodnje pa sve do novorođenog djeteta u rodilištu bolnice, sve u svrhu identifikacije željenog predmeta.

2.2.1. RFID *tagovi*

Osnovna funkcija RFID *taga* je pohrana podataka i slanje istih čitaču. Najosnovniji dizajn *taga* sastoji se od integriranog kruga i antene, ugrađenih u proizvoljno kućište. Općenito, integrirani krug sadrži memoriju u kojoj su pohranjene informacije koje se mogu iščitati, a ponekad i upisati. Određene izvedbe *tagova* sadrže u sebi izvor napajanja u obliku baterije i nazivaju se aktivni *tagovi*, za razliku od pasivnih *tagova* koji se napajaju energijom dobivenom na stezaljkama antene.

RFID *tagovi* su aktivni ako u sebi sadrže izvor napajanja, kao što je baterija. Spomenuti *tagovi* koriste izvor napajanja pri slanju podataka radio vezom i zbog toga aktivni *tagovi* mogu komunicirati s čitačima manjih snaga i mogu slati informacije na puno veće udaljenosti (do nekoliko desetaka metara). Nadalje, aktivni *tagovi* najčešće imaju veće memorije (do 128 Kb). Njihov nedostatak je taj što su dimenzijama mnogo veći od pasivnih *tagova* te puno kompleksniji u dizajnu, a samim time i skuplji za proizvodnju. Prednost predstavlja dugi vijek baterije koja može trajati od dvije pa do sedam godina.

Pasivni RFID *tagovi* nemaju internog napajanja. Umjesto toga, oni se energijom opskrbljuju iz radio signala poslanog od strane čitača. Kao rezultat, pasivni *tagovi* su puno manji i puno jeftiniji od aktivnih. Nedostatak ovakvih *tagova* je njihov domet koji rijetko prelazi udaljenosti od 50 cm. Također, za čitanje su potrebni snažniji čitači, i ova vrsta *tagova* sadrži manje memorije nego aktivni, reda veličine nekoliko kilobajta.

Postoje određene vrste pasivnih *tagova* koji sadrže integrirane baterije, ali ne koriste baterije za RF komunikaciju nego samo za napajanje sklopova. Oni se nazivaju djelomično aktivni *tagovi*.

Još jedna razlika između *tagova* je tip memorije. Grubo rečeno postoje dvije vrste *tagova*: *Tagovi* samo sa svojstvom čitanja (eng. *Read Only*) i *tagovi* sa svojstvom čitanja i pisanja (eng. *Read/Write*).

RO (eng. *Read Only*) memorija je baš kao što sam naziv i kaže; memorija iz koje se može samo čitati. *Tagovi* s takvom memorijom su veoma slični BAR kodu na način da se u njih jednom zapisuje serijski broj i taj se broj kasnije ne može mijenjati. Ova vrsta tagova ima jako

ograničenu memoriju jer su namijenjeni da budu statični, kao što su serijski brojevi i da budu integrirani s sustavom BAR koda.

RW (eng. *Read/Write*) *tagovi* su često nazivani i „pametni“ *tagovi*. Oni pružaju korisniku veću fleksibilnost nego RO *tagovi*. Imaju mogućnost pohrane velike količine informacija i imaju adresnu memoriju koja se može jednostavno mijenjati. Podaci na RW *tagu* mogu se brisati i ponovo upisivati neograničeno (u praksi oko 100 000 ciklusa) i zbog toga se takav *tag* ponaša kao putujuća baza podataka, u kojoj se važne dinamičke informacije pohranjuju u sam *tag*, nasuprot standardnom centraliziranom sustavu u čijem je centru kontroler. Mogućnosti primjene ovakvih *tagova* su neograničene i još uz današnju masovnu proizvodnju, cijena jednog *taga* pala je ispod 1$ što ih čini još više interesantnim u budućnosti.

No postoji nekoliko varijacija ova dva tipa memorija. Jedna varijacija je tip memorije nazvan WORM (eng. *Write Once Read Many*). Ova memorija vrlo je slična RO memoriji s iznimkom da se u takvu memoriju podaci mogu upisivat samo jednom. Primjer uporabe ove memorije bi bio u proizvodnji kada se u *tag* upisuje datum proizvodnje. Jednom upisan ne može se više mijenjati.

RFID *tagovi* dolaze u veoma različitim oblicima tj. kućištima. Zbog njihovih malih dimenzija, RFID *tagovi* mogu se ugraditi u vrlo različite oblike: U sustavima elektronske naplate cestarine mogu bili u obliku kartica ili manjih uređaja; u oblicima samoljepljivih naljepnica za kontrolu proizvoda u skladištima; u zatvorskim sustavima RFID *tagovi* su ugrađeni u narukvice zatvorenika; u obliku staklenih ampula koje su predviđene kao kirurški implantati za praćenje životinja i mnogi drugi oblici.

Ukratko rečeno, oblik *taga* jako ovisi o njegovoj primjeni. Neki *tagovi* su izrađeni da podnesu visoke temperature vlagu ili posebne kemikalije, i sukladno tome su zatvoreni u posebne materijale. Ostali su opet napravljeni tako da budu jeftini i potrošni, kao što su „pametne“ etikete. Iako se RFID *tagovi* mogu upakirati u mnoga različita kućišta, trend ide prema malim i tankim samoljepljivim naljepnicama, koje se mogu brzo i jednostavno postaviti, a i maknuti s proizvoda.

2.2.2. RFID čitači

RFID čitač služi kao most između RFID *taga* i kontroler, a osnovne funkcije su mu:

- Čitanje podataka s RFID *taga*;
- Pisanje podataka na RFID *tag*;
- Prijenos podataka od i prema kontroleru;
- Napajanje *taga* (kod pasivnih *tagova*);

RFID čitači su kompleksni sustavu sastavljeni u grubo od tri dijela: Antene, RF modula (koji je zaslužan za komunikaciju s RFID *tagom*) i upravljačkog sklopovlja koje je zaduženo za komunikaciju s kontrolerom.

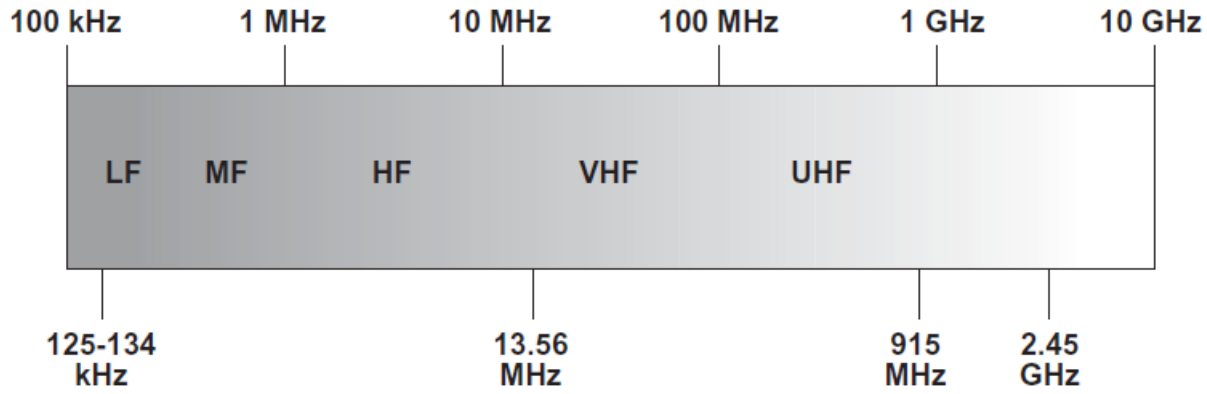
2.2.3. RFID kontroleri

RFID kontroler je „mozak“ cijelog RFID sustava. Oni imaju zadaću umrežavanja velikog broja RFID čitača i centralnu obradu informacija. Kontroler u nekoj mreži je najčešće osobno računalo, radna stanica ili mikro upravljač. On koristi informacije dobivene od čitača da bi:

- Pratio inventar i obavještavao nabavu o stanju robe u skladištu i o potrebi nove nabave
- Pratio kretanje objekata kroz sustav i usmjeravao u određene procese u proizvodnji
- Utvrđivao identitet i autorizirao korisnike, kod sustava bezkontaktnog pristupa (RFID *tagovi* u obliku kartica)
- Teretio određeni bankovni račun kao kod sustava naplate cestarine

2.2.4. Frekvencije rada

Bitno je napomenuti da RFID sustavi funkcioniraju u nekoliko RF područja, a samim time koriste i niske frekvencije i visoke frekvencije za radio komunikaciju:



Sl. 2.2. Radio frekvencijski spektar [1]

Nisko frekventna područja:

- Niske frekvencije: 125 – 134 kHz
- Visoke frekvencije: 13.56 MHz

Visoko frekventna područja:

- Ultravisoke frekvencije: 860 – 960 MHz
- Mikrovalna primjena: iznad 2.5 GHz

Izbor frekvencije uteče na određene karakteristike RFID sustava. Spomenute karakteristike su [1]:

1. Radna udaljenost – na nižim frekvencijama udaljenost čitanja pasivnih *tagova* je svega nekoliko desetaka centimetara, zbog malog dobitka antene. Što su frekvencije veće to se domet povećava, no zbog štetnog djelovanja elektromagnetskog zračenja na višim frekvencijama snaga je ograničena i samim time opao je i domet. Domet iznosi maksimalno 10m za pasivne *tagove*.
2. Pasivni nasuprot aktivnih *tagova* – zbog povijesnih razloga, pasivni *tagovi* su najčešće korišteni na niskim frekvencijskim područjima dok su aktivni *tagovi* korišteni na UHF područjima, no i ova praksa se danas mijenja, pa se danas mogu sresti i pasivni *tagovi* na višim frekvencijama.

3. Interferencija drugih radio uređaja – RFID sustavi su skloniji interferenciji od drugih sustava, pogotovo na niskim frekvencijama, jer na tim frekvencijama odašilju uređaji puno većih snaga nego RFID sustavi pa na prijemnoj anteni može doći do smetnja. Najmanje interferencije imaju RFID *tagovi* u mikrovalnom frekvencijskom području jer su gubitci pri optičkoj vidljivosti (eng. *Line Of Sight Path Loss*) na tim frekvencijama puno veći.
4. Tekućine i metali – Na performanse RFID sustava jako utječu voda i vlažne površine. Signali niske frekvencije lakše prodiru kroz vodu, zbog njihove velike valne duljine, dok su UHF signali prigušeni. Također, metal je reflektor elektromagnetskog zračenja i radio signali ne prolaze kroz nj. Stoga, metal ne samo da ometa radio komunikaciju, nego i sama prisutnost metala u blizini može imati negativne učinke na radio komunikaciju. UHF signali su manje otporni na utjecaj metala za razliku od onih nižih frekvencija.
5. Brzina prijenosa – ako je potrebna veća brzina prijenosa onda je potrebno koristiti više frekvencije u području UHF ili mikrovalnom području
6. Oblik i veličina antene – Zbog velikih valnih duljina, na niskim frekvencijama antene su puno veće od onih za UHF i mikrovalno područje. Najčešće se koriste dva tipa antena: za niske frekvencije koriste se antene s induktivnom spregom (najčešće antene u obliku petlji) dok se za UHF i mikrovalno područje koriste antene s kapacitivnom spregom (najčešće dipoli).
7. Veličina i cijena RFID *taga* – Raniji RFID sustavi koristili su najniže frekvencije i bili su najjeftiniji, no zbog veličine antena postali su skuplji kada se gleda omjer veličine i cijene, pa je HF pojas postao dominantan, čiji su *tagovi* postali najjeftiniji. Napredak u tehnologiji doveo je UHF *tagove* na prag cijeni HF *tagova*.

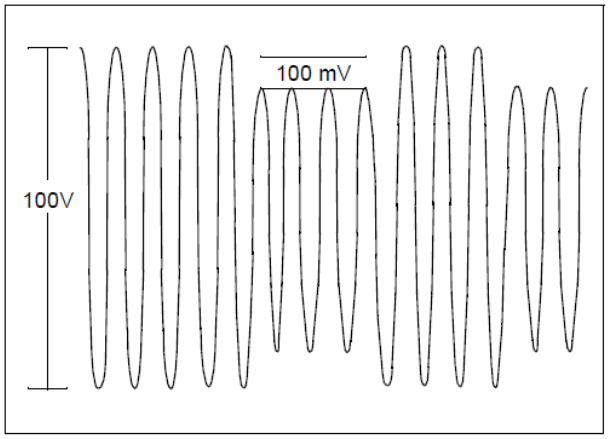
2.3. RF prijenos i modulacija

Modulacija se smatra postupkom promjene parametara prijenosnog signala u svrhu prijenosa informacije. Kod RFID sustava prijenos informacija se vrši na malo drugačiji način nego kod konvencionalnih sustava jer postoji samo jedan odašiljač. *Tag* nema funkciju odašiljača, ali se ipak ostvaruje dvosmjerna komunikacija. RF polje koje stvara čitač ima nekoliko funkcija:

1. Inducirati dovoljno energije u anteni *taga* da bi se ostvarilo napajanje integriranog kruga – pošto pasivni *tagovi* ne sadrže bateriju ili kakav drugi izvor energije, elektromagnetsko polje predstavlja način opskrbe energijom.
2. Osigurati sinkroni izvor signala takta – mnogi RFID *tagovi* dijele prijenosnu frekvenciju određenim faktorom, da bi dobili signal takta koji koristi integrirani krug. Signal takta se koristi u postupku generiranja slijeda podataka koji se šalje čitaču
3. Osigurati prijenosni signal za povratak podataka od strane *taga* – koristi se modulacija povratnog radijskog signala (eng. *Backscatter*), koja je opisana u sljedećem poglavlju.

2.3.1. *Backscatter* modulacija

Pojam *backscatter* modulacije prevodi se kao modulacija povratnog radijskog signala i koristi se kao metoda komunikacije kod pasivnih RFID *tagova* [4]. Komunikacija se vrši na način da *tag* konstantno kratko spaja prijemnu zavojnicu (antenu), preko tranzistora, čime se prigušuje amplituda prijenosnog signala na čitaču. RF veza se ponaša kao transformator (induktivna veza): kako se sekundar (antena na *tagu*) transformatora kratko spaja, na primaru (antena čitača) se osjeti pad napona. Čitač mora detektirati veoma mali pad napona, koji iznosi svega stotinjak milivolta na amplitudi prijenosnog signala od 100 V. Dobiveni valni oblik odgovara amplitudnoj modulaciji signala, točnije rečeno njenoj digitalnoj inačici ASK (eng. *Amplitude Shift Keying*). Izgled signala prikazan je na Slici 2.3.



Sl. 2.3. Izgled moduliranog povratnog radijskog signala [4]

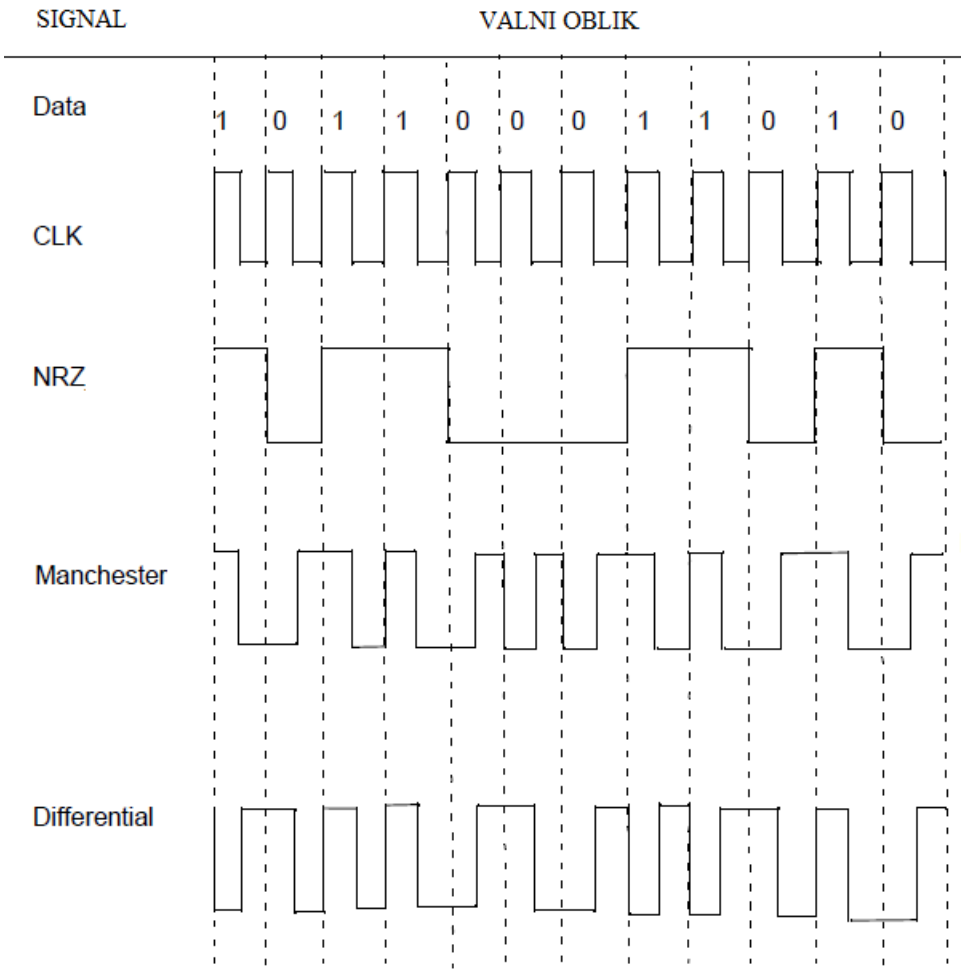
Ova ASK modulacija prijenosnog signala pruža put komunikaciji od *taga* prema čitaču. Podatkovni bitovi se kasnije mogu modulirati na različite načine.

2.3.2. Kodiranje signala

Kodiranje signala se odnosi na proces promjene bitova između vremena kada su podaci dobiveni od strane RFID integriranog sklopa i vremena slanja podataka RF vezom. Različiti algoritmi utječu na cijenu, brzinu i druge čimbenike dizajna sustava. Postoje mnoge vrste kodiranja signala, ali kod RFID sustava se najčešće koriste sljedeće tri [4]:

1. NRZ (eng. *Non Return to Zero*) kodiranje – u ovoj metodi se ne koristi kodiranje signala, nego se samo podatkovni niz prenosi na izlazni tranzistor
2. Diferencijalno kodiranje – podatkovni niz se prijenosi iz polja podataka na način da se prijelaz događa na svaki rub sata takta, a jedinice i nule se razlikuju po prijelazu u sredini trajanja takta sata.
3. *Manchester* kodiranje – varijacija diferencijalnog kodiranja

Izgled različitih formata signala prikazan je na Slici 2.4.



Sl. 2.4. Valni oblici različitih formata signala [4]

2.4. RFID protokoli

Pod pojmom komunikacijskog protokola smatramo skup pravila koji je potreban da se ostvari komunikacija između dva ili više uređaja. U ovom slučaju to su RFID čitač i *tag*. U RFID tehnologiji postoji mnogo vrsta protokola za komunikaciju između *tagova* i čitača, no pošto ovaj rad nije općeniti opis nego konkretna implementacija RFID tehnologije u praksu, ovdje je opisan samo protokol koji je korišten u ovom radu. Protokol koji se koristi ima naziv EM4100 američke tvrtke EM Microelectronics, i jedan je od najzastupljenijih RFID protokola za pasivne RO *tagove*.

2.4.1. EM4100 protokol

EM4100 kompatibilni RFID *tagovi* sadrže 64 byta ROM memorije, što znači da se informacije mogu čitali, ali se ne mogu upisivati [16]. Format podataka prikazan je na Slici 2.5.

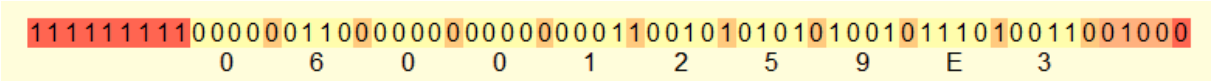
1	1	1	1	1	1	1	1	1	9 bita zaglavlja
8bit ID korisnika ili broj verzije				D00	D01	D02	D03	P0	
				D04	D05	D06	D07	P1	
32 bita podataka				D08	D09	D10	D11	P2	Svaka grupa od 4 bita je popraćena jednim parnim paritetnim bitom
				D12	D13	D14	D15	P3	
				D16	D17	D18	D19	P4	
				D20	D21	D22	D23	P5	
				D24	D25	D26	D27	P6	
				D28	D29	D30	D31	P7	
				D32	D33	D34	D35	P8	
				D36	D37	D38	D39	P9	
4 paritetna bita				PC0	PC1	PC2	PC3	S0	1 stop bit (0)

Sl. 2.5. Izgled podatkovnog polja u EM4100 tagu [16]

Kada se *tag* unese u elektromagnetsko polje čitača, on dobiva napajanje i počinje sa slanjem podataka. Prvih 9 bitova su logičke jedinice. One služe kao marker i označavaju početak niza podataka. Pošto se koristi parni paritet za zaštitu podataka, niti na jednom mjestu u podatkovnom polju ne može se pojaviti sekvenca od 9 logičkih jedinica. Nakon toga slijedi niz od 10 grupa po 4 bita podataka i paritetni bit. Na posljetku nalaze se 4 paritetna bita stupca i

krajnji stop bit. Nakon poslanog niza podataka, *tag* ponavlja slanje cijele sekvence sve dok ima napona napajanja.

Na Slici 2.6 prikazan je primjer niza podataka poslanog od strane *taga*.



Sl. 2.6. Izgled podatkovnog niza poslanog od strane RFID *taga*

Niz podataka dalje se kodira i koristeći modulaciju povratnog radijskog signala spomenuti se šalje čitaču koji demodulira modulirani signal.

2.5. Sigurnost i RFID tehnologija

Veliki rast RFID tehnologije i *tagova* (od nekoliko milijardi) doveli su u pitanje mnoge sigurnosne problem i probleme s privatnosti. Zajednička briga postaje gubitak privatnosti, kada kompanije skeniraju *tagove* radi pribavljanja informacija o kupcima i korištenje tih informacija u razne svrhe.

Kako RFID tehnologija postaje sve sofisticiranija tako se sve više uporabljuje u raznim područjima. Od supermarketa pa sve do označavanja životinja, da bi pratili njihovo kretanje. Ali to nije kraj. Danas se RFID *tagovi* kirurški ugrađuju i u ljude. Američka tvrtka „VeriChip Corporation“ počela je kirurški ugrađivati RFID *tragove* u posjetitelje određenih klubova iz razloga plaćanja računa.Na sve ovo se postavlja dobro pitanje: koliko je ugrožena naša privatnost?

Odgovor na ovo pitane možda nije jednostavan, ali kada se pogleda da je princip izmjene informacija bežična veza, onda se može lako zaključiti da kopirati tuđu karticu, koja sadrži sve osobne podatke, brojeve tekućih računa i ostale osobne podatke, je moguće i to sa „sigurne udaljenosti“. Iako je RFID tehnologija ograničena radnom udaljenosti i dalje je moguće prići osobi i pomoću RFID čitača presnimiti sve potrebne podatke u sekundi. Novije „pametne kartice“ sadrže određene mehanizme zaštite, no kolika je njihova pouzdanost, to je još potrebno utvrditi.

3. TEORIJSKE OSNOVE ZIGBEE TEHNOLOGIJE

ZigBee predstavlja bežični komunikacijski protokol namijenjen osobnim mrežama s malom propusnošću i malom potrošnjom energije. Ciljane primjene ZigBee-a su aplikacije koje zahtijevaju umrežavanje velikog broja uređaja, prijenos male količine podataka, malu potrošnju energije te visoku sigurnost prijenosa. ZigBee se temelji na IEEE normi 802.15.4, ali se često ova dva pojma poistovjećuju [18].

Mrežni protokol ZigBee nastao je kada se pojavila potreba za umrežavanjem velikog broja uređaja, između kojih se prenosi mala količina podataka, a aplikacije zahtijevaju veliku energetsku autonomiju uređaja, te samim time i malu potrošnju. Tipični primjeri aplikacija s ovakvim zahtjevima su bežične senzorske mreže, kontrolne mreže itd, prikupljanje medicinskih podataka i slično. ZigBee uređaji trebaju biti maleni, jeftini i pouzdani.

3.1. Klase i tipovi ZigBee uređaja

IEEE norma definira dvije klase uređaja po razini funkcionalnosti: FFD (eng. *Fully Functional Device* – uređaj s potpunom funkcionalnošću) i RFD (eng. *Reduced Functional Device* – uređaj s ograničenom funkcionalnošću). FFD je uglavnom spojen na neki stalni izvor napajanja dok je RFD energetski samostalan (tj. ima neko vlastito autonomno napajanje) te je kao takav ograničen u energetskom smislu i funkcijama u mreži. ZigBee definira uređaje po funkciji u mreži. To su koordinator (mrežni koordinator, eng. *coordinator*, uvijek FFD) koji obavlja inicijalizaciju mreže, uspostavlja vezu među ostalim čvorovima, služi kao izlaz prema nekoj drugoj vrsti mreže (LAN, GSM) itd., usmjerivač (enl. *router*, uvijek FFD) koji služi za povećavanje dometa mreže i krajnji uređaj (eng. *end device*), uglavnom RFD, može biti i FFD) na kojemu su uglavnom smještena osjetila, izvršni članovi (eng. Actuators) ili neke upravljačke jedinice [18].

Tab. 3.1. Klase ZigBee uređaja

ZigBee uređaj	IEEE klasa uređaja	Funkcija
Koordinator	FFD	Uspostavlja mrežu, dodjeljuje mrežne adrese, brine se o sigurnosti i ispravnosti razmjene podataka između čvora.
Usmjerivač	FFD	Neobavezan. Povećava fizički domet mreže, omogućava većem broju čvorova spajanje u mrežu.
Krajnji uređaj	RFD (rijetko FFD)	Osjetila, kontrolni uređaji.

3.2. ZigBee protokolni stog

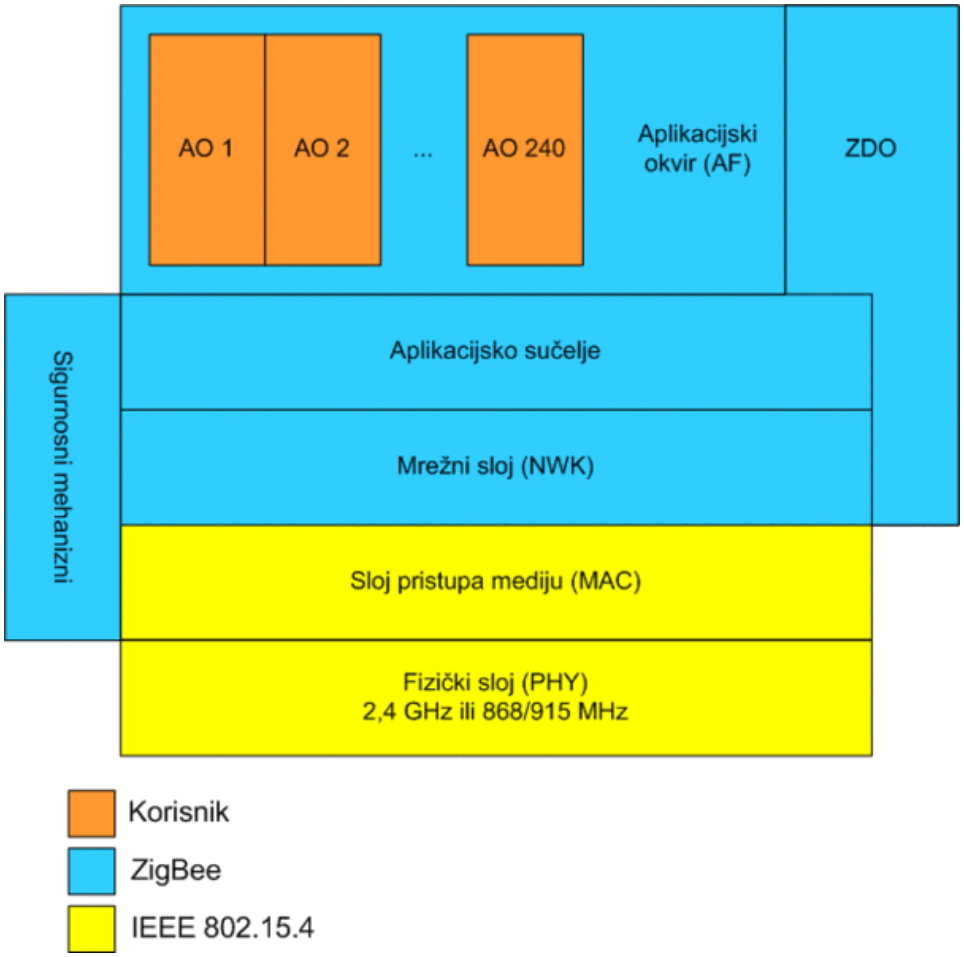
ZigBee protokolni stog temelji se na OSI modelu, ali definira samo one slojeve koji su važni za ostvarivanje funkcionalnosti na željenom području [18].

Uloga **fizičkog sloja** (PHY – eng. P*HYsical layer*) u komunikaciji jest aktiviranje i deaktiviranje primopredajnika, mjerenje razine energije signala (ED – eng. *Energy Detection*), indikacija kvalitete veze (LQI – *eng. Link Quality Indicator*) i provjera oslobođenosti kanala (CCA – eng. *Clear Channel Assesment*), odabir kanala te primanje i slanje podataka. Fizički sloj definira tri pojasa frekvencija: prvi na 868 MHz koji sadrži kanal 0 i podržava prijenos podataka od 20 kbps, drugi na 915 MHz s 10 kanala (1 – 10) i brzinom prijenosa od 40 kbps te treći na 2,4 GHz s 16 kanala (11 – 26) s brzinom prijenosa od 250 kbps i širinom kanala od 5 MHz. Na prva dva pojasa koristi se binarna PSK (eng. *Phase Shift Keying*) modulacija, dok se na trećem pojasu koristi QPSK (eng. *Quadrature Phase Shift Keying*). U frekvencijskom opsegu na 2,4 GHz deklarirani domet u otvorenom prostoru iznosi 100 m, dok u zatvorenom iznosi 30 m pri snazi odašiljanja od 1 mW. Radi poboljšanja odnosa signal/šum te povećanja selektivnosti kanala upotrebljava se tehnika moduliranja raspršenja spektra direktnim postupkom (DSSS – eng. *Direct Sequence Spread Spectrum*).

Sloj za pristup mediju (MAC – eng. *Medium Access Layer*) zadužen je za pristup fizičkom sloju, generiranje i sinkroniziranje na medij, pokretanje koordinatora i generiranje PAN Id-a (*Personal Area Network Identifier*), određivanje i izvršavanje GTS-a (eng. *Guaranteed Time Slot*), korištenje CSMA-CA mehanizma za pristup kanalu, te uspostavljanja veze između MAC entiteta na različitim čvorovima i ostvarivanje pouzdane komunikacije između dva susjedna čvora u mreži. Funkcionalnost uređaja je definirana na ovom sloju. Također, MAC sloj podržava određene sigurnosne mehanizme.

Mrežni sloj (NWK – eng. *NetWorK layer*) je zadužen za pravilno formiranje mrežne topologije, konfiguriranje uređaja, uključivanje i isključivanje čvora s mreže, dostavljanje poruke pravom odredištu, pravilno adresiranje, otkrivanje susjedstva i pravih putova između dva čvora, te prosljeđivanje i primanje podataka od aplikacijskog i MAC sloja. Mrežni sloj podržava usmjeravanje poruka (eng. *Routing*), odnosno formiranje optimalne mrežne topologije. Ove dvije funkcije su glavne značajke ZigBee protokola. Funkcija uređaja je definirana na mrežnom sloju. Mrežni sloj, kao i MAC, podržava određene sigurnosne mehanizme.

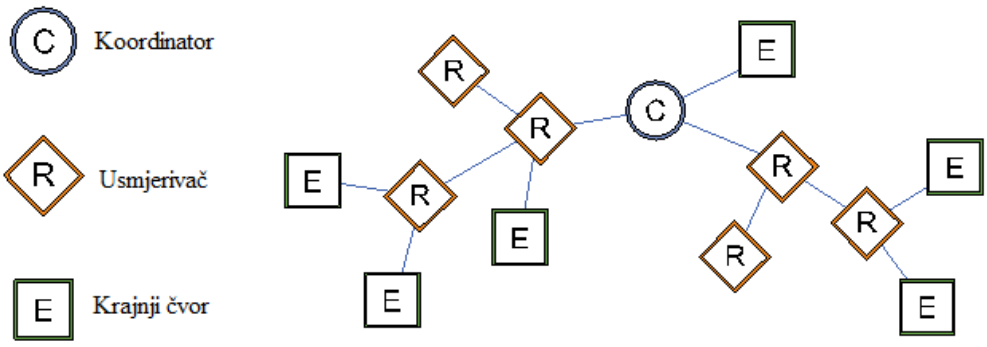
Aplikacijski sloj je zadužen za ispravnu komunikaciju između aplikacija koje koriste ZigBee mrežu kao sredstvo komunikacije među čvorovima.



Sl. 3.1. ZigBee protokolni stog [18]

3.3. Formiranje ZigBee mreže

ZigBee mreže nazivaju se PAN (eng. *Personal Area Network*) mreže. Svaka mreža sadrži 16 bitnu oznaku koja se naziva PAN ID. Kao što je ranije spomenuto, ZigBee protokol definira tri različita uređaja koji postoje u ZigBee mreži: koordinator, usmjerivač i krajnji čvor. Primjer jedne mreže prikazan je na Slici 3.2 [5].



Sl. 3.2. Vrste čvorova u ZigBee mreži [5]

Koordinator – je zadužen za odabir kanala mreže i PAN ID. Koordinator osniva novu PAN mrežu. Kada je osnovao PAN mrežu, koordinator dopušta usmjerivačima i krajnjim čvorovima da se spoje na PAN. Koordinator može slati i primati podatke, može pomagati pri usmjerivanju podataka kroz mrežu. Koordinatori nisu zamišljeni kao uređaji koji se napajaju energijom baterije. Zbog činjenice da koordinator mora združivati i usmjeravati podatke treba biti napajan iz elektroenergetske mreže.

Usmjerivač – prvo se mora pridružiti ZigBee PAN mreži prije nego što može obavljati poslove usmjerivanja podataka. Nakon što se pridruži PAN, usmjerivač može dopustiti ostalim usmjerivačima i krajnjim čvorovima da se pridruže PAN-u. Usmjerivač također može primati i slati podatke i može usmjeravati podatkovne pakete kroz mrežu. Zbog činjenice da usmjerivači mogu dopustiti pridruživanje mreži i usmjeravaju podatke, oni ne bi smjeli biti u opcijama spavanja (eng. *Sleep Mode*), i trebaju se napajati iz elektro energetske mreže.

Krajnji čvor – ima zadaću pridruživanja PAN mreži, ali on za razliku od usmjerivača ne može dopustiti pridruživanje drugih čvorova u mrežu, niti može pomoći u usmjeravanju podataka. Krajnji čvor može samo primati i slati podatke preko RF medija. Oni su predviđeni da se napajaju iz baterija i stoga mogu koristiti opcije spavanja (eng. *Sleep Mode*). Pošto krajnji

čvorovi mogu biti ugašeni određeno vrijeme, usmjerivači ili koordinator za to vrijeme moraju prikupiti sve pakete namijenjene krajnjem čvoru, i čuvati ih dok krajnji čvor ne pristupi mreži. Usmjerivač ili koordinator koji dopusti krajnjem čvoru pridruživanje u mrežu naziva se njegov roditelj (eng. *Parrent*), dok se krajnji čvor smatra njegovim djetetom (eng. *Child*).

3.3.1. Pokretanje PAN mreže

Zbog činjenice da je koordinator odgovoran za pokretanje ZigBee mreže, svaka ZigBee mreža u početku mora imati prisutnog koordinatora. Da bi pokrenuo PAN, koordinator izvodi nekoliko skeniranja radi određivanja RF aktivnosti na različitim kanalima, također i da bi otkrio susjedne PAN mreže.

Kada se koordinator uključi po prvi puta, on izvodi skeniranje energije na nekolicini kanala (frekvencija) da bih odredio količinu šuma na svakom kanalu. Kanali sa prevelikom energijom (prevelikim šumom) odbacuju se s liste potencijalnih kanala za pokretanje PAN mreže.

Kada je skeniranje energije kanala završeno, koordinator skenira preostale „tihe“ kanale ne bi li pronašao postojeće PAN mreže (PAN skeniranje). To radi na način da šalje grupnu (eng. *Broadcast*) poruku, koju kada prime susjedni koordinatori ili usmjerivači, obavještavaju koordinator da je taj kanal zauzet.

Nakon skeniranja energije kanala i PAN skeniranja koordinator obrađuje dobivene podatke i sada sam pokušava osnovati PAN mrežu s PAN ID (koji je jedinstven) i na kanalu koji nije zauzet. Nakon što je PAN mreža osnovana koordinator može dozvoliti pridruživanje ostalih usmjerivača i krajnjih čvorova.

3.3.2. Pridruživanje ZigBee mreži

Usmjerivači i krajnji korisnici moraju otkriti PAN mrežu i pridružiti joj se. Da bi to bilo moguće oni prvo izvode PAN skeniranje (isto kao što koordinator izvodi kada osniva novu mrežu). Iz PAN skeniranja usmjerivač ili krajnji čvor dobiva popis najbližih ZigBee uređaja. Usmjerivač ili krajnji čvor obrađuju popis da bi našli odgovarajuću ZigBee mrežu kojoj se trebaju pridružiti. Usmjerivači i krajnji čvor se mogu postaviti tako da se pridružuju bilo kojoj ZigBee mreži ili mreži s točno određenim PAN ID.

Nakon što uređaj koji se pridružuje pronađe odgovarajuću mrežu (koja dozvoljava pridruživanje) pokušava se spojiti na PAN mrežu šaljući zahtjev za spajanje tom uređaju.

Koordinator i usmjerivači mogu dopustiti novim usmjerivačima i krajnjim čvorovima da se pridruže mreži. Hoće li neki usmjerivač ili koordinator dopustiti pridruživanje novim uređajima u mreži ovisi o dvije stvari:

- O tome da li je pridruživanje dozvoljeno (ako je u opcijama određeno) te
- O broju djece koje taj čvor ima

Ako je sigurnost aktivirana, koordinator koristiti 128 bitni AES enkripcijski ključ. Samo uređaji koji imaju isti enkripcijski ključ mogu komunicirati u PAN-u.

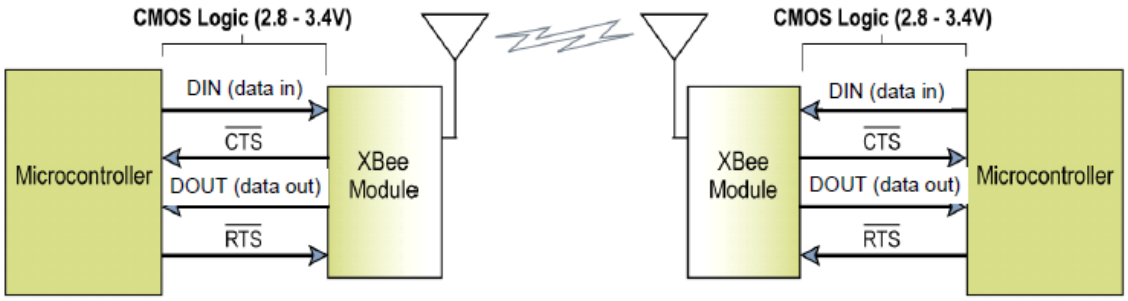
3.4. ZigBee modul – XBee ZNet 2.5

Sl. 3.3. XBee ZNet 2.5 RF modul

XBee ZNet 2.5 RF moduli dizajnirani su da rade unutar ZigBee protokola i podržavaju jedinstvene zahtjeve za bežičnu mrežu male cijene i male potrošnje energije. Svojstva su im da potražuju minimalnu količinu energije i pružaju pouzdani prijenos podataka između udaljenih uređaja. Moduli rade unutar frekvencijskog područja 2.4GHz.

Da bi se moduli mogli konfigurirati i slati podatke potrebno ih je povezati s računalom ili mikro upravljačem. XBee moduli se povezuju s vanjskim uređajima preko asinkronog serijskog porta. Pošto naponske razine ne odgovaraju standardnom RS232 protokolu za izravnu komunikaciju s računalom potreban je naponski pretvornik (npr. MAX232).

Uređaji koji sadrže UART sučelje mogu se izravno povezati s XBee modulom, kao na Slici 3.4.



Sl. 3.4. Povezivanje XBee modula UART sučeljem [5]

Podatkovne linije modula oznakom RTS i CTS mogu se koristiti za kontrolu toka, a ako kontrola toka nije potrebna te linije mogu poslužiti i drugoj svrsi.

3.4.1. Protokol serijskog sučelja

XBee moduli podržavaju dva načina serijskog sučelja i to:

- Transparentno serijsko sučelje i
- API (eng. *Application Programming Interface*) sučelje.

Kada modul radi u transparentnom načinu rada on se ponaša kao zamjena za serijsku liniju. Svi podaci koji su s UART sučelja poslani na XBee modul automatski se šalju na RF medij, a čvor koji prima podatke na isti ih način prosljeđuje na izlaz svoga UART sučelja. Konfiguracija modula postiže se s AT komandama [5].

Poslani podaci se spremaju u privremenu memorija (eng. *Buffer*) a tek nakon nastanka jednog od sljedećih uvjeta paketiziraju se i šalju na RF medij:

1. Niti jedan znak nije primljen preko serijskog porta određeno vrijeme. Vrijeme određuje parametar RO. Ako je RO=0 svaki se znak posebno šalje na RF medij.
2. Primljen je maksimalan broj znakova koji stanu u RF paket (72 bajta)
3. Primljena je komandna sekvenca (GT + CC + GT)

S druge strane, API način rada je alternativan način rada transparentnom načinu rada. API način rada je baziran na okviru koji ima određenu formu i samo okviri koji su ispravni biti će primljeni od strane XBee modula. Postoje dvije vrste okvira i to:

Predajni podatkovni okvir sadrži:

- RF predajni podatkovni okvir
- Komandni okvir (ekvivalent AT komandama kod transparentnog načina)

Prijemni podatkovni okvir sadrži:

- RF prijemni podatkovni okvir
- Komandni odgovor
- Obavijest o određenom događaju kao npr. ponovo pokretanje (eng. *Reset*)

API pruža alternativni način konfiguracije modula i usmjeravanja podataka na aplikacijskom sloju. Aplikacija na strani računala može slati gotove pakete koji u sebi sadrže odredišnu adresu, umjesto da se adresa mijenja preko komandnog načina rada.

Iako API način rada pruža neke prednosti, njegova implementacija u sustavu mikro upravljača nije jednostavna zbog potrebe kreiranja cijelog podatkovnog okvira i svih njegovih dijelova. Stoga je u izvedbi komunikacije u ovom radu korišten transparentni način rada, s konfiguracijom u komandnom načinu rada.

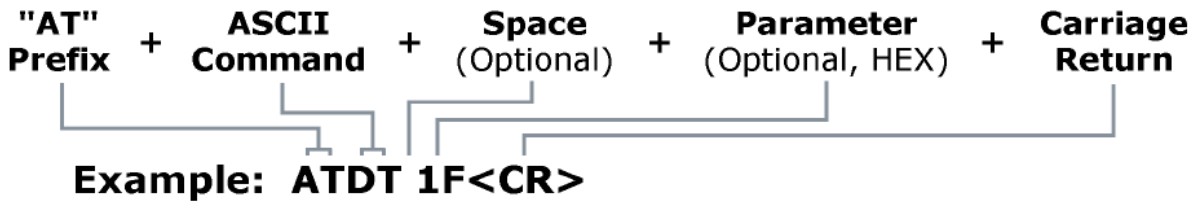
3.4.2. Komandni način rada

Ako je potrebno promijeniti ili samo pročitati neki od parametara zapisan u XBee modulu (kao npr. PAN ID ili broj kanala), modul se mora prebaciti u komandni način rada – stanje u kojemu su nadolazeći serijski simboli tumačeni kao naredbe. Kod transparentnog načina rada koriste se AT komande za konfiguraciju XBee modula.

Princip ulaska u komandni način rada je sljedeći:

- Niti jedan simbol ne smije se slati u trajanju zaštitnog perioda (vrijeme od 1s po tvorničkim postavkama)
- Tokom 1s potrebno je poslati tri uzastopna simbola '+'
- Niti jedan simbol ne smije se slati u trajanju zaštitnog perioda (vrijeme od 1s po tvorničkim postavkama)

Ako je ulazak bio uspješan, modul odgovara s „OK“. Kada je komandni način rada aktiviran XBee modul prihvaća AT komande. Izgled standardne AT komande prikazan je na Slici 3.15.



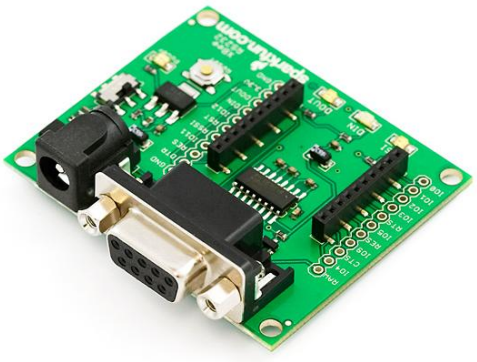
Sl. 3.5. Primjer AT komande [5]

XBee modul sadrži mnoge AT komande koje služe za promjenu postavki modula i mogu se pronaći u [5].

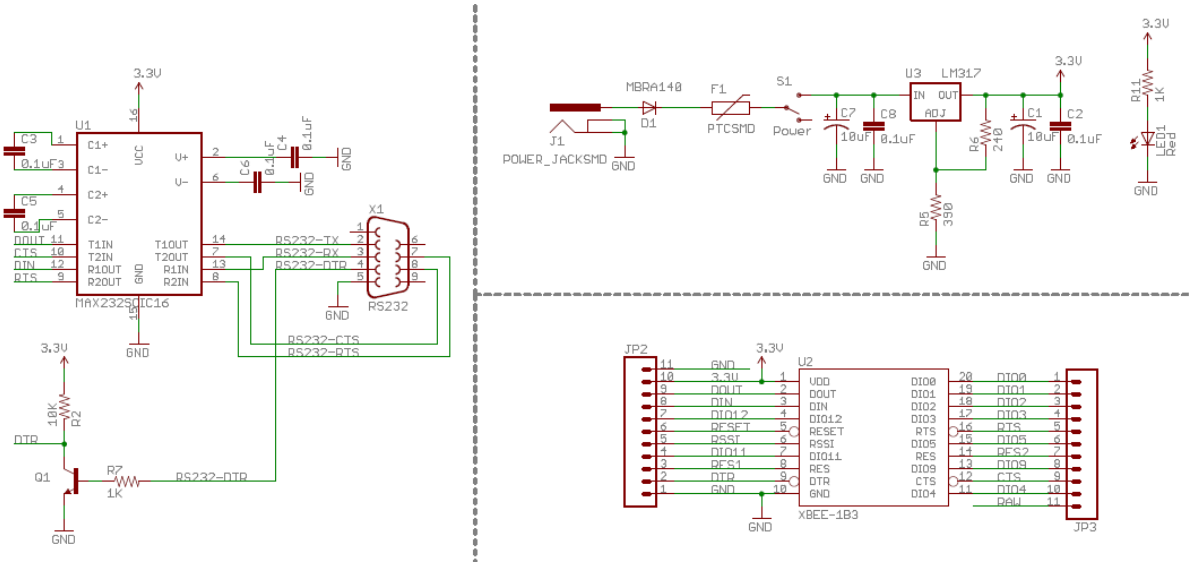
3.4.3. Konfiguracija XBee modula pomoću računala

XBee modul potrebno je konfigurirati da vrši funkciju koordinatora, usmjerivača ili krajnjeg čvora. To se postiže promjenom softvera integriranog u XBee modul (eng. *Firmware*). Promjena *firmwarea* kod XBee modula može se provesti veoma jednostavno, uz pomoć serijskog RS232 sučelja i programskog alata zvanog X-CTU.

Serijsko sučelje koje se koristi naziva se XBee Serial Explorer i sastoji se od translatora naponskih razina, koji prilagođuje razine XBee modula i RS232 linije i izvora napona od 3.3V. Izgled modula prikazan je na Slici 3.6. Shematski prikaz samog sučelja prikazan je na Slici 3.7.



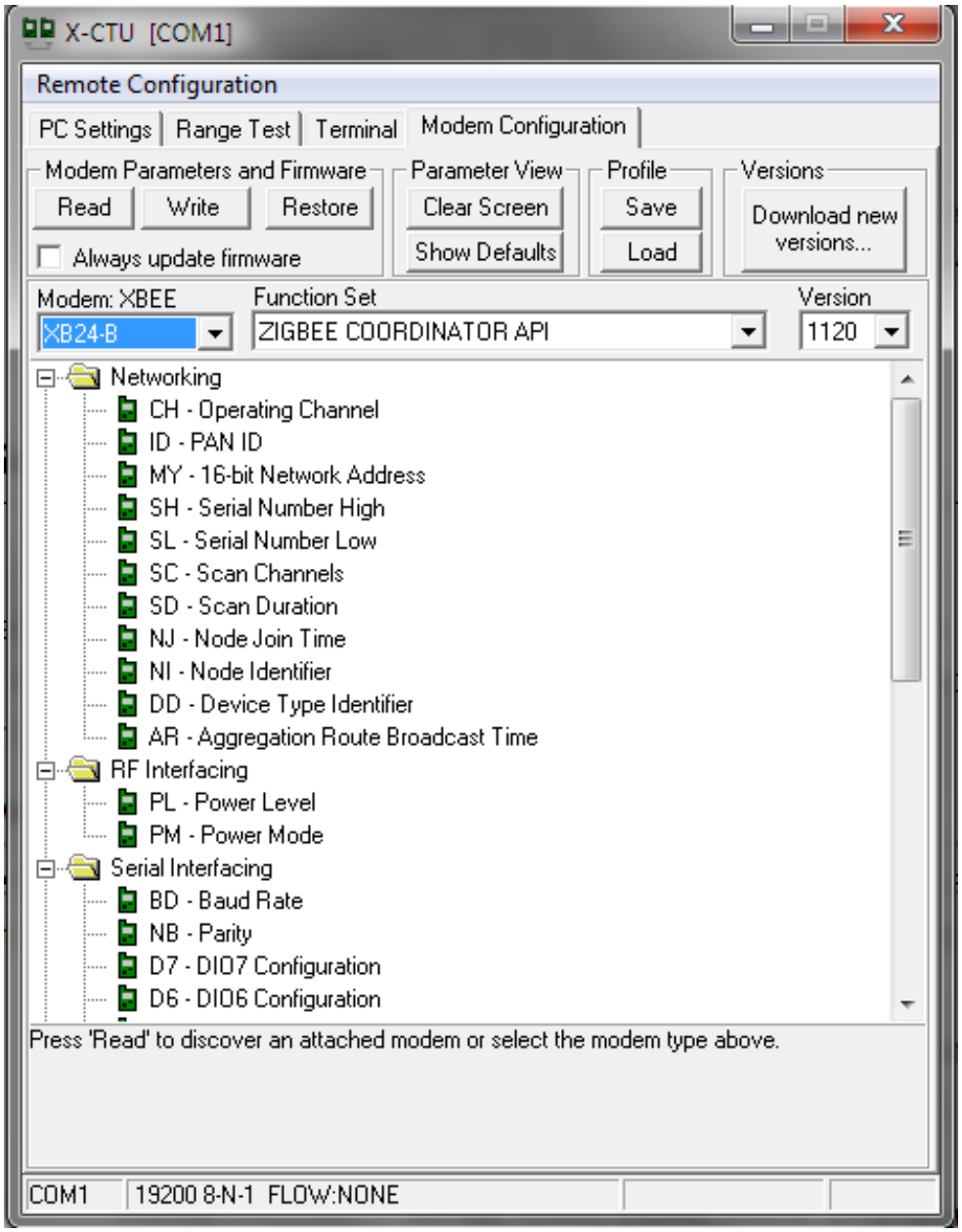
Sl. 3.6. Izgled Serial Explorer sučelja [13]



Sl. 3.7. Shematski prikaz XBee Serial Explorer sučelja [13]

Iz Slike 3.7. vidimo da je sustav translatora naponskih razina riješen pomoću integriranog kruga MAX232, koji u sebi sadrži nabojne pumpe koje opskrbljuju integrirani sklop s bipolarnim izvorom napajanja. Dobiveni naponi koriste se pri formiranju RS232 valnog oblika koji po je po svojoj naravi bipolaran (radi smanjivanja utjecaja smetnja). Izvor napajanja izveden je pomoću integriranog kruga LM317 koji ima svojstvo podešavanja vrijednosti izlaznog napona. Na spomenuto sučelje spaja se XBee modul i uz prisutan izvor napajanja programiranje modula je omogućeno.

Spomenuti programski alat ima mogućnosti konfiguracije XBee modula, promjene *firmwarea* i serijski terminal za testiranje i ručno mijenjanje parametara pomoću komandnog načina rada. Izgled programskog alata prikazan je na Slici 3.8. U konfiguracijskom prozoru prikazani su mnogi parametri koji se mogu mijenjati, kao na primjer PAN ID, OC (eng. *Operating Channel*) i mnoge druge parametre. Kada se čvor konfigurira kao koordinator, tada odabire proizvoljni PAN ID (ime mreže) a XBee modul ostalo odrađuje sam, kako je opisano u poglavlju 3.3.2. Kako bi se XBee modul pripremio za određeni način rada, bilo to kao koordinator, usmjerivač ili krajnji čvor, potrebno je odabrati određeni set funkcija i odabrati komandu *Write* u *Firmware* okviru. Time se u postojeći XBee modul upisuje željeni način rada. Bitno je napomenuti da su funkcije usmjeravanja i koordiniranja potpuno hardverski neovisne, tako da isti modul može obavljati bilo koju funkciju, ovisno o *firmwareu* koji je upisan u nj.

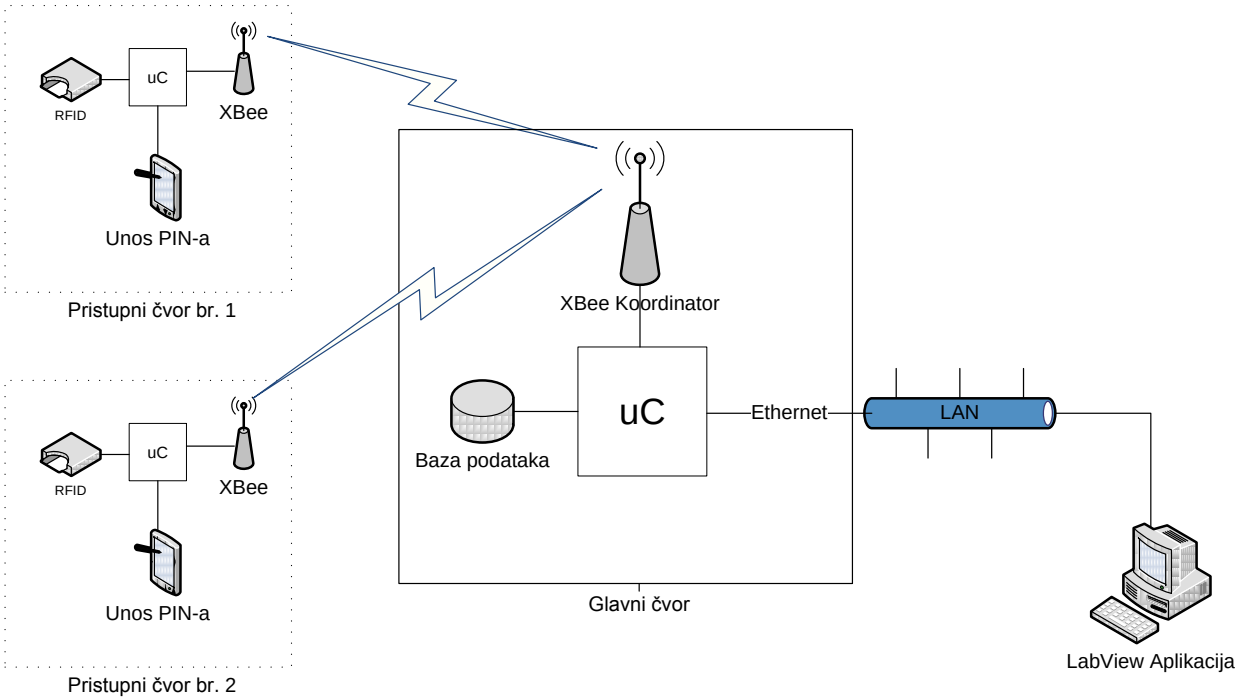


Sl. 3.8. Izgled programskog alata X-CTU

4. IZRADA SUSTAVA BEŽIČNE RFID AUTORIZACIJE KORISNIKA

U praktičnom dijelu rada načinjen je sustav bežične RFID autorizacije korisnika, koristeći XBee module za ostvarivanje bežične komunikacije između glavnog čvora, i pristupnih čvorova.

Princip rada samog sustava je sljedeći: Sustav bežične RFID autorizacije korisnika ima za zadatak autorizaciju korisnika koristeći bezkontaktne RFID kartice, popraćen unosom osobnog koda (PIN). Da bi bilo moguće provoditi evidenciju korisnika u bazi podataka, na glavnom čvoru odrađuje se posao vođenja same baze. Glavni čvor je „srce“ cijelog sustava i na njega je vezan XBee modul (u funkciji koordinatora) koji ima zadaću kreiranja PAN mreže. Također, glavni čvor ima Ethernet vezu prema LAN mreži, i preko spomenutog sučelja moguć je pristup *LabView* aplikaciji vođenja baze podataka. Izgled sustava prikazan je na Slici 4.1.



Sl. 4.1. Sustav bežične RFID autorizacije korisnika

Kao što se vidi iz Slike 4.1, u centru i pristupnih čvorova i glavnog čvora nalaze se mikro upravljači (oznaka uC) različitih karakteristika. Glavni zahtjevi na mikro upravljač glavnog čvora bili bi: dovoljna snaga za obradu podataka baze podataka, UART sučelje za komunikaciju s XBee koordinatorom, mogućnost spajanja Ethernet sučelja i komunikacija preko LAN mreže s aplikacijskim klijentom. Glavni svojstva mikro upravljača pristupnog čvora jesu: sposobnost demodulacije i dekodiranja dolaznih RFID podataka (uz pomoć dodatnog modula), mogućnost spajanja tipkovnice za unos osobnog koda (u ovom radu LCD ekran osjetljiv na dodir) i UART

sučelja za komunikaciju s XBee modulom. Nakon prikazanih zahtjeva na mikro upravljače u nastavku je prikazana arhitektura spomenutih mikro upravljača, koji sadrže potrebne funkcije za uspješno kreiranje čvorova.

4.1. Arhitektura AVR mikro upravljača

Kao što je opisano, zahtjevi na mikro upravljače za glavni i pristupni čvor su raznoliki pa su stoga i mikro upravljači koji se koriste za implementaciju čvorova različiti. Zbog posjedovanja razvojnih sustava srpske tvrtke „mikroElektronika“, i raznih mikro upravljača kompatibilnih s spomenutim razvojnim sustavima, odabrani su sljedeći mikro upravljači:

- Glavni čvor – mikro upravljač: Atmel AVR ATMega128
- Pristupni čvor – mikro upravljač: Atmel AVR ATMega32

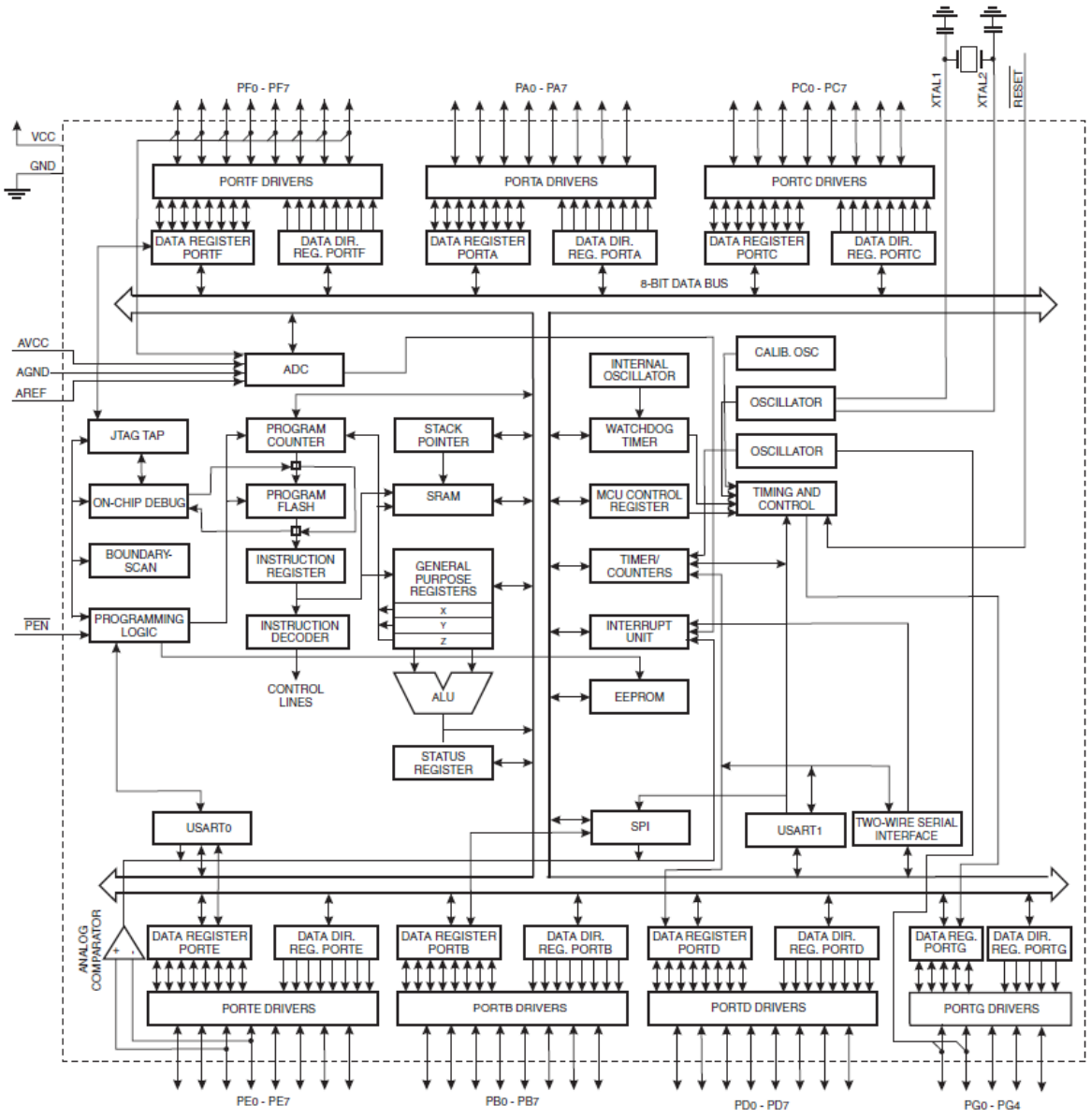
4.1.1. Atmel AVR ATMega128

ATmega128 je 8 bitni CMOS mikro upravljač, male snage, baziran na poboljšanoj AVR RISC arhitekturi. Izvršavajući veoma snažne instrukcije brzinom od jedne instrukcije u jednom taktu, ATmega128 dobiva propusnost od 1 MIPS (eng. *Million Instructions Per Second*) po 1 MHz, čime se može odabirati izbor između brzine i uštede energije. Samim time, ovaj mikro upravljač se postavio idealan za glavni čvor. Arhitektura samog mikro upravljača prikazana je na Slici 4.2.

AVR jezgra kombinira bogati set instrukcija s 32 registra opće primjene. Sva 32 registra su izravno vezani s ALU, čime se ostvaruje istovremeni pristup dvama registrima, u jednoj instrukciji, koja se izvodi u jednom periodu takta. Dobivena arhitektura je više efikasna u pogledu koda dok ostvaruje propusnost ček i deset puta veću nego kod standardnih CISC mikro upravljača.

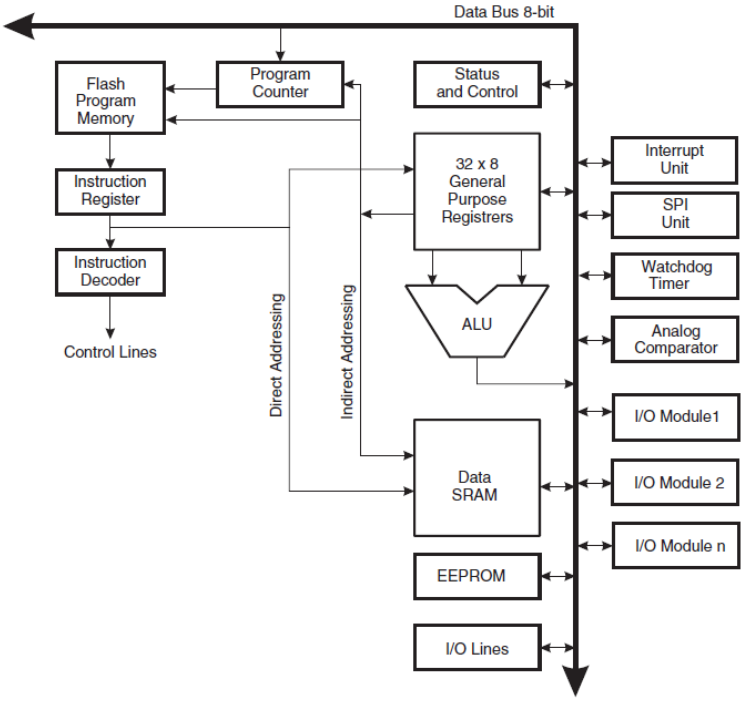
ATmega128 pruža sljedeće pogodnosti: 128kB programabilne *flash* memorije s svojstvima čitanja i pisanja; 4kB EEPROM memorije, 4kB SRAM memorije, 53 linije opće namjene (I/O), 4 fleksibilna Timera/Brojača s mogućnosti PWM izlaza, 2 USART modula, Two-Wire serijsko sučelje, 8 kanalni 10bitni ADC pretvarač, SPI serijsku sabirnicu, JTAG sučelje i mnoga druga svojstva, prema [7].

Sklop je proizveden koristeći Atmel-ovu memoriju visoke gustoće. Sklop ISP (eng. *In-System Programming)* programiranja integriran u mikro upravljač omogućuje upisivanje koda u mikro upravljač preko standardne SPI (eng. *Serial Peripheral Interface*) sabirnice, pomoću konvencionalnih programatora ili pomoću programa podizanja (eng. *Boot*) koji je integriran u mikro upravljač.



Sl. 4.2. Blok dijagram Atmel AVR ATmega128 [7]

Ako se pogleda općenita arhitektura AVR mikro upravljača, onda se vidi da je glavni dio arhitekture AVR jezgra, kojoj je glavna funkcija izvođenje programa upisanog u *Flash* memoriju. AVR jezgra ili CPU mora biti u mogućnosti pristupiti memoriji, izvoditi računske operacije, upravljati s periferijom i obrađivati prekide (eng. *Interrupt*).



Sl. 4.3. Blok dijagram AVR arhitekture [7]

Kako bi se maksimalizirale performanse i paralelizam operacija, AVR koristi *Harvard* arhitekturu - s odvojenim memorijama i sabirnicama za program i podatke. Instrukcije se nalaze u programskoj memoriji i izvode se s jednostrukim „cjevovodima“ (eng. *Pipelining*). Dok se jedna instrukcija izvodi, sljedeća instrukcija se povlači iz programske memorije. Ovaj koncept dozvoljava izvođenje instrukcija u svakom taktu sata. Programska memorija je *Flash* memorija s mogućnošću programiranja u radu (ISP).

Registri brzog pristupa sadrže 32 x 8 bita registra opće namjene, s pristupom koji se vrši u jednom taktu sata, što omogućuje ALU rad u jednom ciklusu takta sata. Šest od 32 registra mogu se koristiti kao 16 bitni indirektni adresni pokazivači za adresiranje podatkovnog prostora, što omogućuje efikasno računanje adresa.

ALU podržava aritmetičke i logičke operacije između registara ili između konstante i registra. Također, operacije na samo jednom registru se mogu izvoditi u ALU.

Tok programa je uspostavljen pomoću uvjetnih i bezuvjetnih skokova, i instrukcija poziva potprograma. Većina AVR instrukcija ima format 16bitne riječi. Programska memorija je podijeljena u dva dijela: *Boot* dio i Aplikativni dio.

Tokom prekida programa (eng. *Interrupt*) ili prilikom pozivanja potprograma, povratna adresa programskog brojača PC sprema se u Stog. Stog je lociran u općem SRAM-u i njegova

veličina je ograničena jedino veličinom SRAM-a. Svi korisnički programi moraju inicijalizirati SP (eng. *Stack Pointer*) u rutini ponovnog pokretanja (eng. *Reset*).

Kada se odmakne od ALU jedinice prema periferiji, nailazi se na mnoge module koji koriste u daljnjem razvoju sustava. Posebno treba naglasiti module kao što su UART, SPI sabirnica i I2C sučelje. U nastavku teksta obrađeni su spomenuti modula.

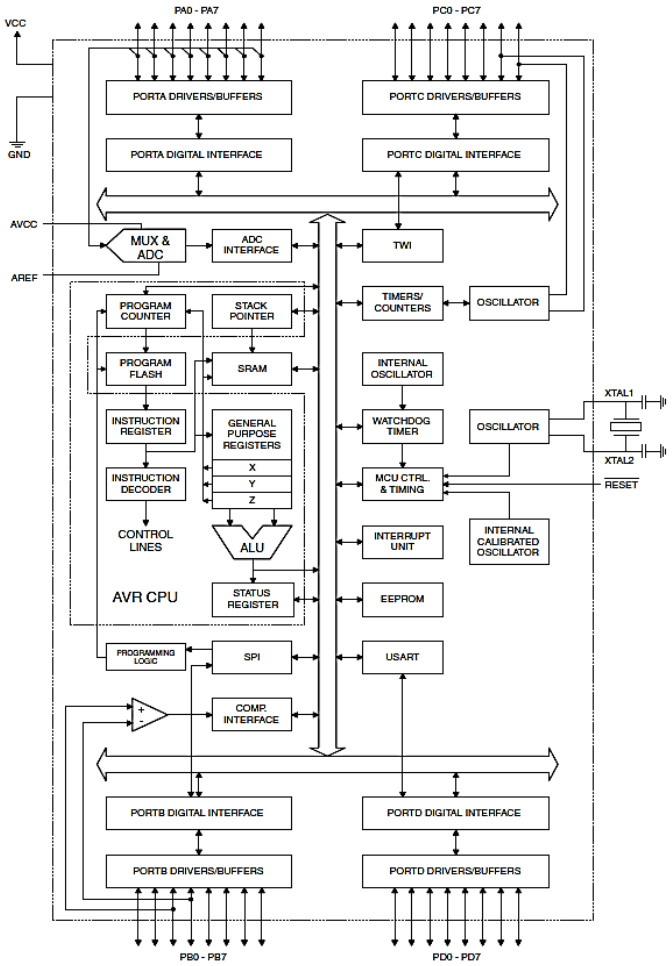
4.1.2. Atmel AVR ATMega32

Manji brat *Atmel* AVR ATmega128 je spomenuti ATmega32. Sa samo dvije instrukcije manje od ATmega128, 32kB programske memorije, 1kB EERPOM memorije, 2kB SRAM memorije, 32 ulazno izlazne linije, 10bitnim ADC pretvornikom, 2 vanjska prekidna ulaza, UART sučeljem i ostalim karakteristikama predstavlja izvanredan izbor za pristupni čvor. Kao što je spomenuto, glavni zahtjevi na pristupni čvor jesu: kompatibilnost s UART sučeljem, mogućnost demoduliranja RFID signala i mogućnost grafičkog prikaza unosa PIN-a na LCD ekranu osjetljivom na dodir. Sve ove karakteristike sadrži spomenuti mikro upravljač, prema [6].

Iz blok dijagrama na Slici 4.4. vidi se da je ATmega32 po funkcijama puno šturiji od njegovog većeg brata ATmega128, a najveća se razlika vidi u ulazno izlaznim linijama, kojih je skoro dvostruko manje. Također, ovaj mikro upravljan ne sadrži određene module koji postoje u ATmega128, ali ti moduli na posljetku nisu ni potrebni. Arhitektura ALU i način izvođenja programa identičan je kao i kod ATmega128, pa se taj dio neće ponovo spominjati.

ATmega32 dolazi u različitim kućištima, od standardnog PDIP 40 pa sve do SMD verzije u kućištu TQFP44, što je prednost kod izrade manjih pristupnih čvorova. Također, potrošnja energije je jako mala, što je značajno ako pristupni čvorovi budu napajani iz baterijskih izvora. Potrošnja od 1.1mA u aktivnom načinu rada i samo 0.35mA u mirovanju (eng. *Idle*) govori o štedljivosti ovog mikro upravljača. Zbog potrebe obrade slike za LCD povećana je brzina rada, a samim time i potrošnja.

Bitan dio koji do sada nije spominjan je oscilator mikro upravljača. Naime, oscilator određuje frekvenciju takta a samim time i brzinu izvođenja instrukcija. Kod AVR mikro upravljača on je integriran u sam integrirani sklop a frekvencija se određuje spajanjem kristalnog rezonatora na izlazne pinove mikro upravljača.



Sl. 4.4. Blok dijagram Atmela ATmega32 [6]

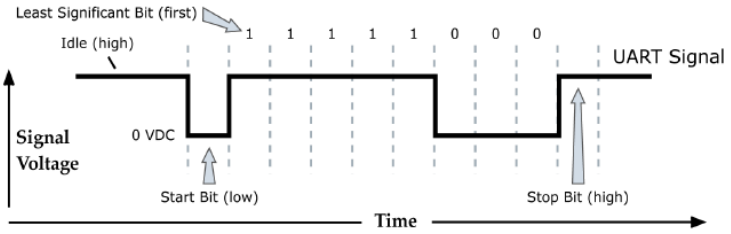
4.1.3. Periferni moduli AVR mikro upravljača

Periferija svake arhitekture odnosi se na vezu s okolinom, preko određenih sučelja. Kod AVR mikro upravljača veza s okolinom provodi se preko nekoliko različitih sučelja:

- Serijsko USART ili UART sučelje;
- Sučelje preko SPI sabirnice;
- I2C sučelje preko TWI modula;
- *Interrupt* ulazi;
- ADC pretvornik;

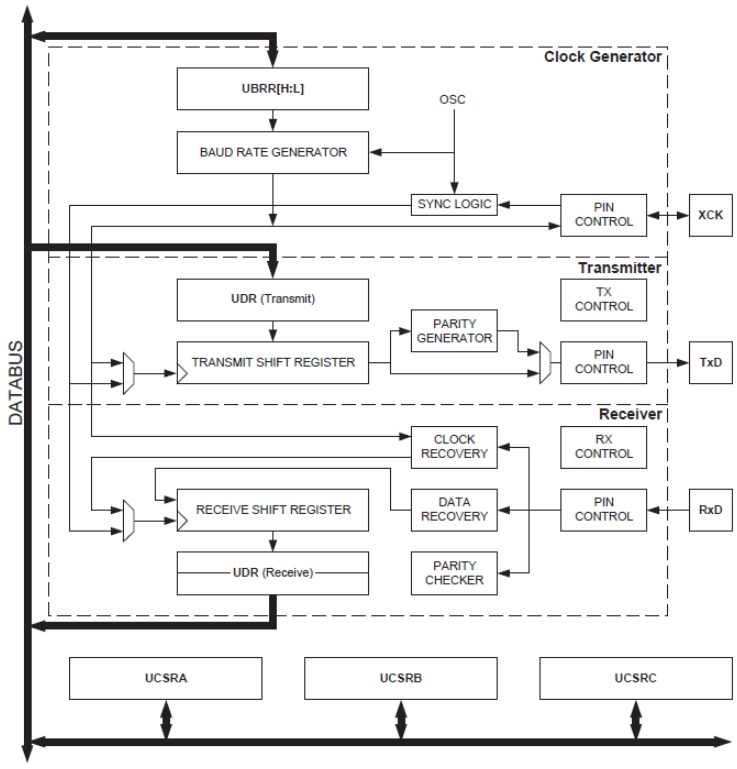
Serijsko USART (UART) sučelje, skraćenica je za Univerzalni Sinkroni i Asinkroni serijski Prijemnik i Predajnik (eng. *Universal Synchronous and Asynchronous serial Receiver and Transmitter*) i predstavlja jako fleksibilni komunikacijski uređaj. Pošto korišteni XBee

modul sadrži UART sučelje kao i AVR mikro upravljač, komunikacija između spomenutih je moguća. UART signal ima oblik prikazan na Slici 4.5.



Sl. 4.5. Izgled UART signala [5]

Izgled USART bloka u AVR mikro upravljaču prikazan je na Slici 4.6. Općenito, vidimo da se USART modul sastoji od tri dijela: Generatora takta, odašiljača i prijemnika. Upravljanje s opcijama USART-a vrši se manipulacijom hardverskih registara UCSRA, UCSRB i UCSRC koji tada konfiguriraju sklopovlje da vrši zadanu funkciju. Logika generiranja takta sastoji se od sinkronizacijske logike koja služi za primanje takta sata, kod sinkronog načina rada, i generatora takta (eng. *Baud Rate Generator*).



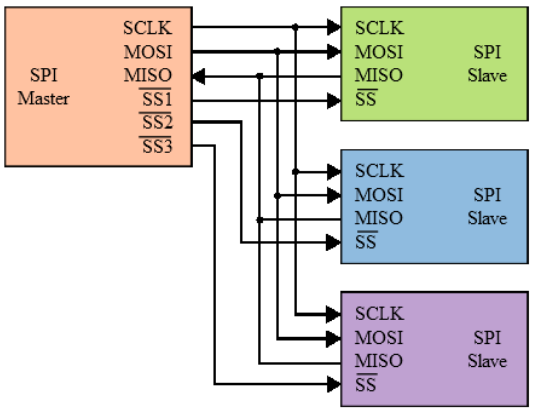
Sl. 4.6. Blok dijagram USART modula [6]

Vrlo je važno napomenuti jednu stvar koja se tiče generatora takta za UART. Naime, UART komunikacija je asinkrona, što znači da se ne prijenosi signal takta, i on se mora izvlačiti

iz podatkovnog signala. Pošto mikro upravljač ima svoj signal takta (dobiven lokalnim oscilatorom) taj se signal mora podijeliti određeni broj puta da bi se dobio standardna brzina ili standardni *baud rate*. Jedna od standardnih brzina je 9600 baud. Pošto se frekvencija lokalnog oscilatora mikro upravljača mora dijeliti, a može se dijeliti samo potencijom broja 2, tada se uviđa određeno ograničenje. Naime, standardne vrijednosti frekvencija signala takta (kao npr. 4 MHz) ne daju adekvatan brzinu (eng. *Baud rate*) nego se javlja odstupanje (npr. za takt od 5 MHz baud rate će biti 19531 baud umjesto 19200 baud). Zbog toga se problema koriste kristali kvarca s posebnim rezonantnim frekvencijama: 4.9152 MHz, 11.0592MHz, 14.7456MHz. U slučaju sustava bežične RFID autorizacije korisnika koristi se frekvencija od 11.059200MHz.

Odašiljač (eng. *Transmitter*) se sastoji od jednog hardverskog registra, serijskog posmičnog registra, generatora pariteta i upravljačke logike za primanje različitih formata. Prijemnik (eng. *Reciever*) je najkompleksniji dio USART modula zbog dijelova koji čine restauraciju podataka i takta sata. Dijelovi koji restauriraju signal koriste se pri asinkronom načinu rada, kada vanjski signal takta nije dostupan. Prijemnik podržava iste formate kao i odašiljač i može otkrivati pogreške u okvirima i paritetu.

SPI sučelje, ili Serijsko periferno sučelje (eng. *Serial Peripheral Interface*) je sinkroni serijski podatkovni link, standardiziran od strane američke tvrtke Motorola, za rad u *full duplex* načinu rada. Uređaji mogu komunicirati u *Master*/*Slave* načinima rada, gdje glavni *Master* uređaj započinje podatkovni okvir. Višestruki *Slave* uređaji dopušteni su, a odabir pojedinog *Slave* uređaja vrši se pomoću *Chip Select* linije. Blok dijagram SPI sabirnice prikazan je na Slici 4.7.

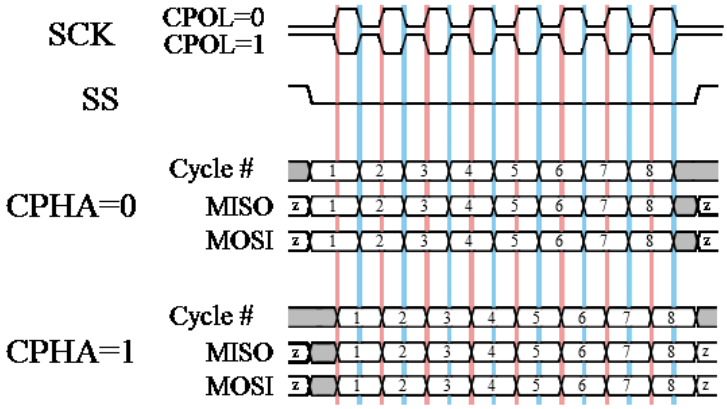


Sl. 4.7. Blok prikaz SPI sabirnice s tri Slave uređaja [6]

SPI sabirnica koristi 4 linije za komunikaciju:

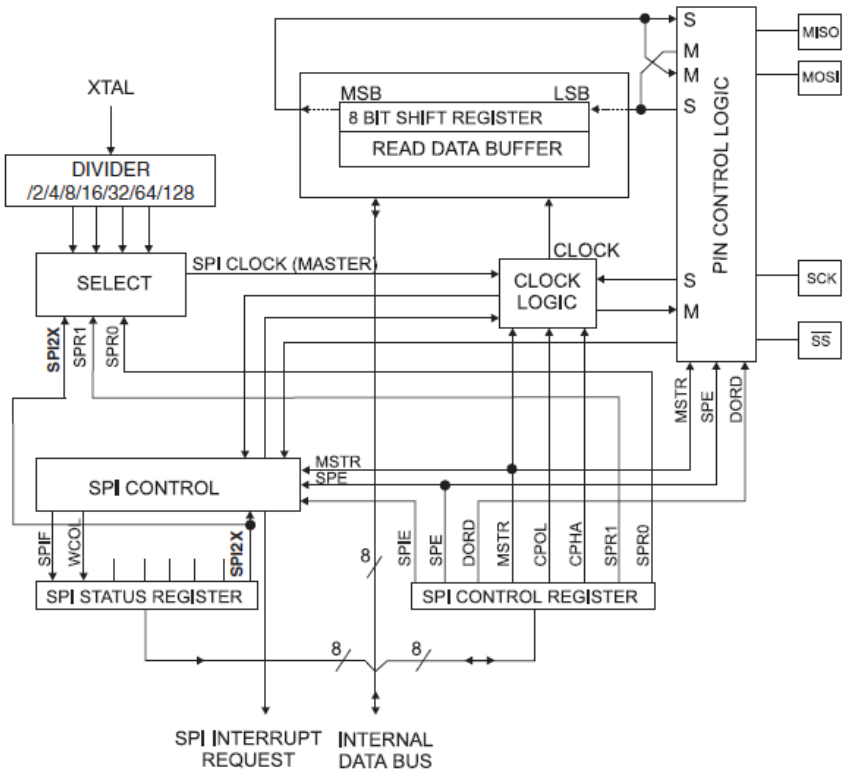
- SCLK (*eng. Serial Clock*) - Izlaz iz *Master* uređaja, takt sata
- MOSI/SIMO (*eng. Master Output, Slave Input*) - Izlaz iz *Master* uređaja
- MISO/SOMI (*eng. Master Input, Slave Output*) - Izlaz iz *Slave* uređaja
- SS (*eng. Slave Select*) - Odabir *Slave* uređaja

Frekvencija rada ovise o uporabi i mogu biti od 1-70 MHz. Uz samu frekvenciju takta, *Master* uređaj treba biti podešen na odgovarajući polaritet sata takta i fazu, u odnosu na podatkovni niz. Ti parametri se nazivaju CPOL i CPHA i u ovisnosti o njima ovisi i valni oblik signala. Svi valni oblici prikazani su na Slici 4.8.



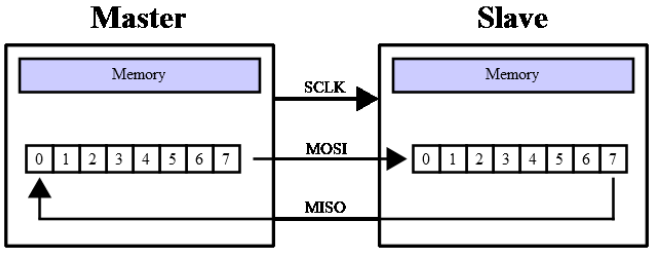
Sl. 4.8. Valni oblici SPI sabirnice [6]

Kako bi se mogla ostvariti komunikaciju s uređajima baziranima na SPI sučelju, potreban je modul koji obrađuje SPI podatke. Odabrani mikro upravljač Atmel AVR ATmega128 ima integrirani SPI modul, i time je komunikacija sa SPI sabirnicom moguća. Izgled SPI modula prikazan je na Slici 4.9.



Sl. 4.9. Blok dijagram SPI modula [6]

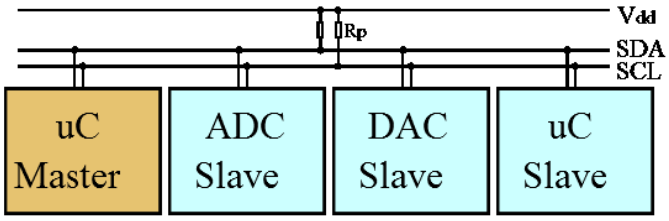
Komunikacija između *Master* i *Slave* uređaja vrši se na principu posmičnih registara, gdje se sadržaj iz registra u *Master* uređaju pomiče u registar u *Slave* uređaju, a sadržaj *Slave* uređaja se pomiče u registar u *Master* uređaju. Princip izmjene informacija prikazan je na Slici 4.10. SPI sabirnica ima mnogo primjena, a u ovom radu koristi se za vezu između SD memorijske kartice i *Ethernet* sučelja, što je opisano u nastavku rada.



Sl. 4.10. Prijenos podataka između *Master* i *Slave* uređaja kod SPI sabirnice[6]

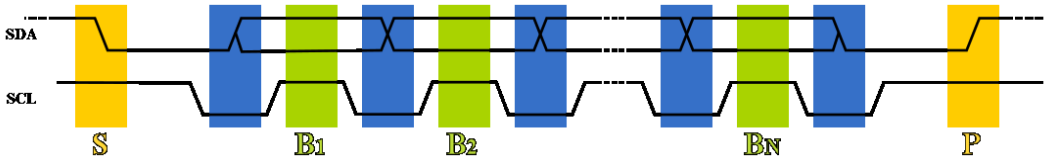
I^{2}C sučelje je vrsta sabirnice standardizirana od strane američke tvrtke „Phillips“, a koristi se za spajanje perifernih uređaja male brzine na matičnu ploču, mobilni telefon ili neki drugi ugrađeni sustav. Za komunikaciju koriste se dvije linije, od koje je jedna taktna linija (SCL) a druga podatkovna linija (SDA), obje dvosmjerne s *pull up* otpornicima. Slično kao kod

SPI sabirnice i ovdje je jedan uređaj *Master* a ostali su *Slave*, i slušaju upute od *Mastera*. Izgled standardne I^{2}C sabirnice prikazan je na Slici 4.11.



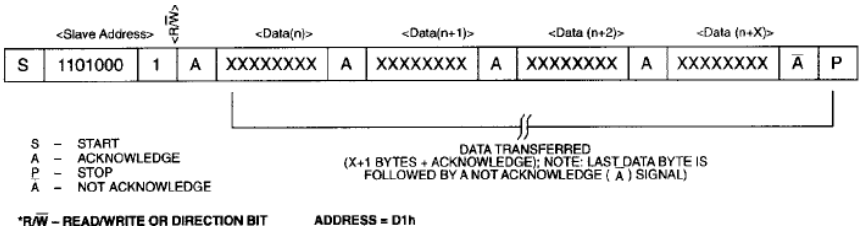
Sl. 4.11. Blok prikaz I^{2}C sabirnice [8]

Prijenos podataka preko I^{2}C sučelja vrši se na sljedeći način. Početak prijenosa podataka označen je sa START bitom, koji je označen s prijelazom linije SDA sa 1 na 0, dok je linija SCL na logičkoj jedinici. Nakon toga na SDA liniji se postavlja bit koji se prijenosi za vrijeme dok je SCL u stanju logičke nule. Kada se SCL promjeni iz 0 u 1, na SDA liniji se vrši uzorkovanje. Kada je prijenos završen šalje se STOP bit na način da SDA linija pređe iz logičke 0 u 1 dok je SCL u stanju logičke jedinice. Prikaz prijenosa podataka prikazan je na Slici 4.12.



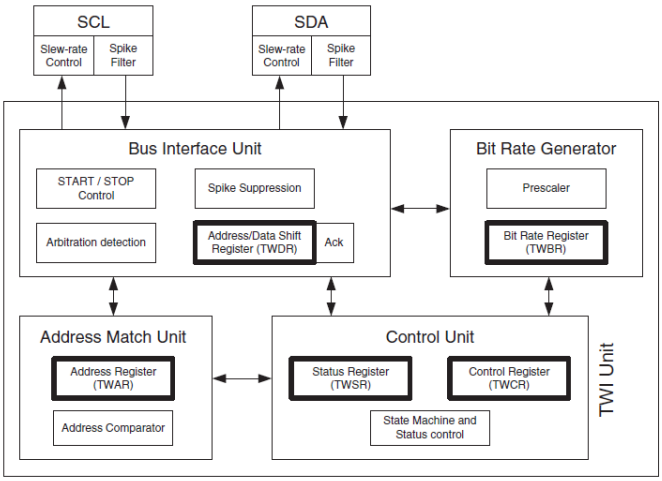
Sl. 4.12. Prijenos podataka kod I^{2}C sabirnice [8]

Adresiranje kod I^{2}C sabirnice vrši se na način da se prvo pošalje adresa *Slave* čvora, čemu slijedi bit koji označuje da li se podaci šalju ili primaju, a ovisno o njemu *Master* prima ili šalje podatke. Nakon toga se šalje memorijska adresa podataka. Primjer čitanja podataka iz I^{2}C *slave* čvora prikazan je na Slici 4.13.



Sl. 4.13. Izgled I^{2}C komunikacije pri čitanju iz *Slave-a* [8]

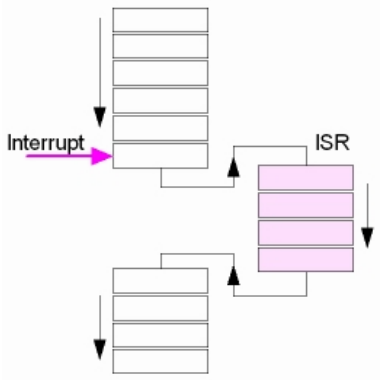
Atmelov mikro upravljač AVR ATmega128 nema izravnu potporu I2C sabirnici, ali ima potporu TWI (eng. *Twin Wire Interface)* modula, koji je kompatibilan s I2C sučeljem i može vršiti sve funkcije kao i standardna I^{2}C sabirnica. Blok dijagram TWI modula prikazan je na Slici 4.14.



Sl. 4.14. TWI modul u AVR mikro upravljaču [7]

TWI modul može funkcionirati u *Master* i *Slave* načinu rada. Kada je u *Master* načinu rada tada on proziva *Slave* uređaje spojene na sabirnicu, i potražuje ili šalje podatke. Uporaba TWI modula za I^2C komunikaciju koristi se pri komunikaciji s hardverskim modulom RTC (eng. *Real Time Clock*) koji pruža točno vrijeme i datum.

***Interrupt* ulazi,** ili ulazi prekidnih rutina su posebni hardverski ulazi koji na određeni uvjet pozivaju prekidnu rutinu. Oni su izravno vezani na prekidni modul AVR arhitekture i pri zadanoj promjeni logičkog stanja na ulazu pozivaju prekidnu rutinu spremljenu u programskoj memoriji mikro upravljača. Prikaz rada prekidne rutina prikazan je na Slici 4.15.

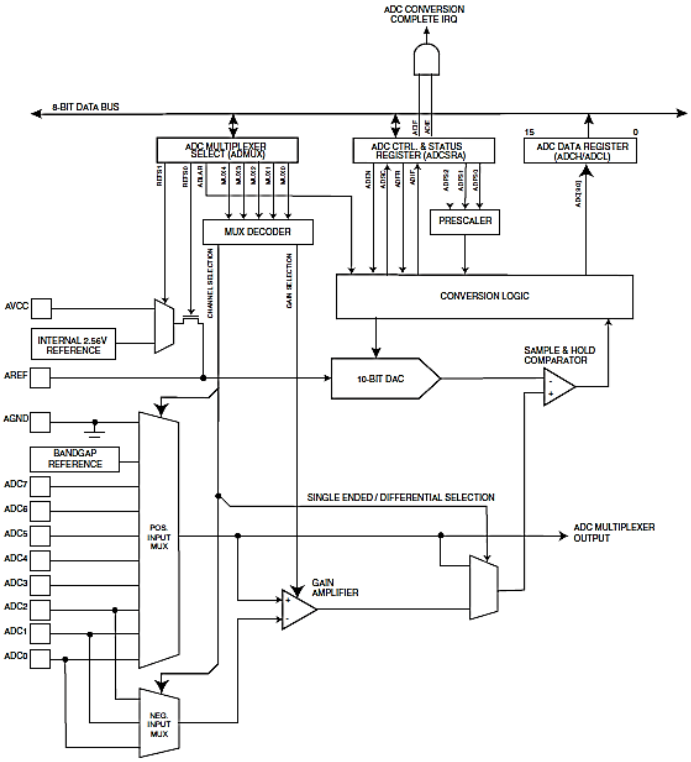


Sl. 4.15. Princip rada prekidne rutine

Interrupt ulazi se mogu koristit za razne primjene. Primjer: ako se detekcija podesi na način da se prekidna rutina poziva na promjenu signala iz logičke nule u jedinicu, tada takva prekidna rutina ima svojstva brojanja perioda nekog signala takta, i sukladno time obrađivati podatke. Ovaj princip se koristi kod dekodiranja RFID signala, što je opisano kasnije u radu.

Posljednji periferni dio AVR mikro upravljača koji je obrađen je **ADC pretvornik.** ADC (eng. *Analog to Digital Converter*) pretvornik je 10 bitni pretvornik, sa sukcesivnim

približavanjem (eng. *Successive approximation ADC*) . ADC je vezan na 8 kanalni multiplekser, koji omogućuje spajanje osam različitih ulaza na mikro upravljač. ADC pretvornik ima mogućnost spajanja diferencijalnih ulaza, koji se u ovom radu nisu upotrebljavali. ADC sadrži sklop uzorkovanja i zadržavanja uzorka (*eng. Sample and Hold*), koji omogućuje konstantni uzorak za vrijeme ADC pretvorbe. Blok dijagram ADC pretvornika prikazan je na Slici 4.16.



Sl. 4.16. Blok dijagram ADC pretvornika [6]

ADC pretvornik pretvara analogni ulazni napon u njegovu 10 bitnu digitalnu reprezentaciju, kroz postupak sukcesivnog približavanja. Minimalna vrijednost predstavlja GND dok maksimalna vrijednost predstavlja napon na pinu AREF minus 1 LSB. Po izboru, može se odabrati unutarnji naponski referentni izvor od 2.56V ili proizvoljni eksterni referentni izvor. Brzina pretvorbe ovisi o željenoj maksimalnoj razlučivosti, a frekvencije pretvorbe se kreću od 50kHz do 200kHz. Vrijeme pretvorbe kreće se reda veličine nekoliko desetaka milisekundi.

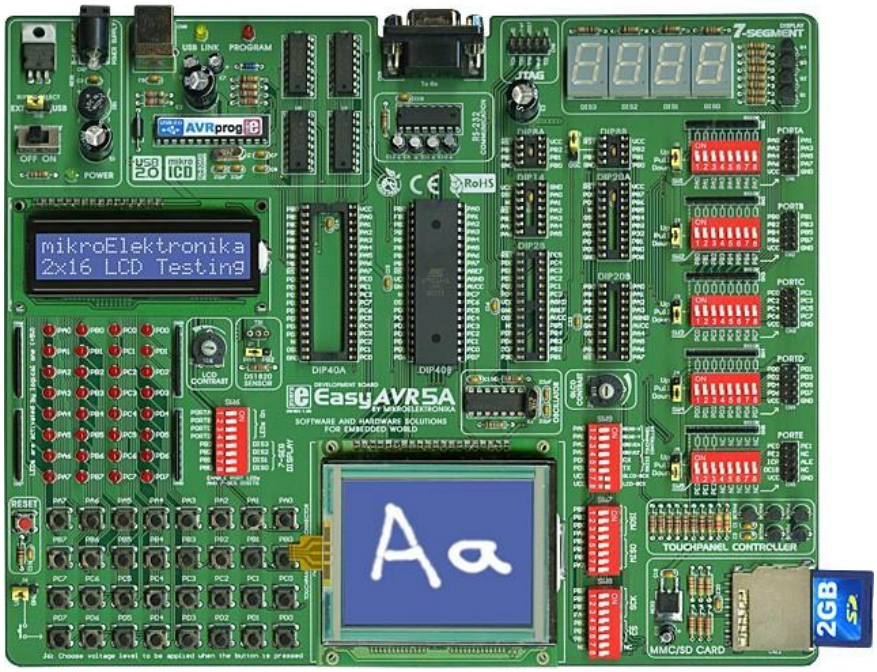
U sustavu bežične RFID autorizacije korisnika, ADC pretvorba se koristi za digitaliziranje zaslona osjetljivog na dodir, opisanog nadalje u radu.

Nakon obrađene arhitekture AVR mikro upravljača naglasak je stavljen na arhitekturu glavnog i pristupnog čvora, počevši od pristupnog čvora i RFID čitača, koji je osnova spomenutog čvora.

4.2. Dizajniranje arhitekture pristupnog čvora

Kao što je prikazano u prethodnom poglavlju, „mozak“ pristupnog čvora predstavlja mikro upravljač ATmega32, čija je radna frekvencija 11.0592 MHZ, što daje mikro upravljaču snagu obrade od malo više od 11 MIPS-a. Ako usporedimo prosječno današnje osobno računalo koje ima prosječno 10000 MIPS-a, onda vidimo pravu snagu korištenog mikro upravljača. Usporedbu na stranu, dobivena snaga obrade sasvim je zadovoljavajuća za planiranu primjenu.

Pošto se za razvoj ovog sustava koristio razvojni sustav srpske tvrtke „mikroElektronika“ pod nazivom EasyAVR5A, njegov izgled prikazan je na Slici 4.17.



Sl. 4.17. Razvojni sustav EasyAVR5A [9]

Kao što se vidi iz priložene slike, spomenuti razvojni sustav ima dva LCD ekrana, od kojih je jedan grafički osjetljivi na dodir dok je drugi *dot matrix* tipa s 2 x 16 simbola, koji se ne koristi u ovom radu. Razvojni sustav također ima mogućnost spajanja vanjskih modula na ulazno izlazne linije.

Kada spominjemo vanjske module onda u priču dolazi i RFID čitač, koji je detaljno opisan u nastavku teksta.

4.2.1. RFID čitač

Kao što je spomenuto, jedan od dodatnih modula spomenutom razvojnom sustavu čini i RFID čitač. Izgled modula RFID čitača prikazan je na Slici 4.18.

Sl. 4.18. Modul RFID čitača [14]

Čitač se zasniva na EM4095 integriranom krugu. Kada je to rečeno treba razjasniti funkciju spomenutog integriranog kruga. Naime, spomenuti integrirani krug ima sljedeće funkcije [14]:

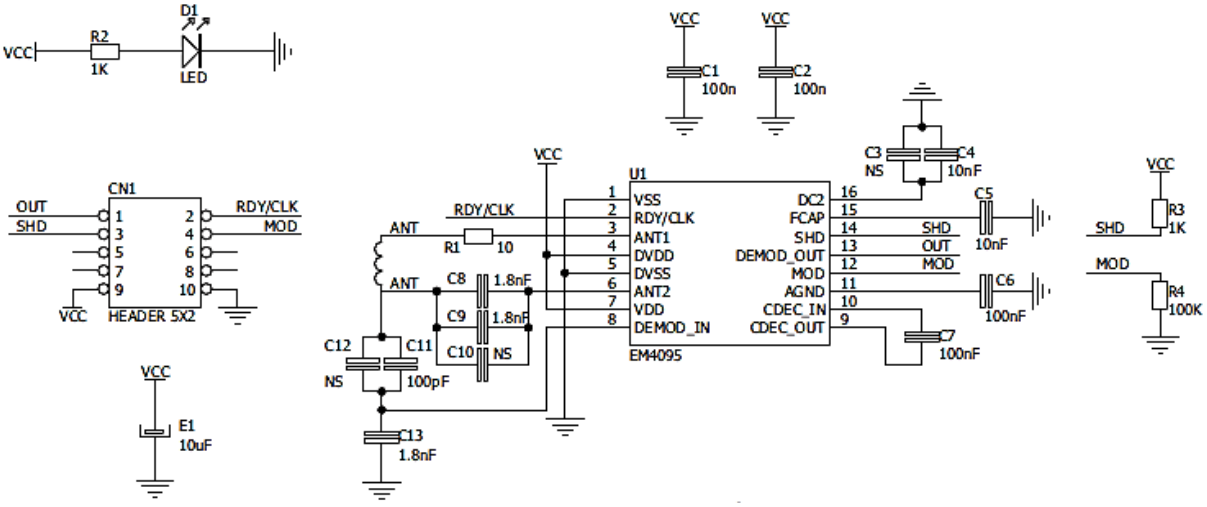
- Opskrba antene s frekvencijom nosioca 125 kHz;
- ASK modulacija polja za RW tagove;
- ASK demodulacija signala s antene, induciranog od strane *taga;*

Iz funkcija integriranog kruga EM4095 koji je „srce“ RFID čitača, možemo zaključiti da je jedina stvarna funkcija ovog modula, detekcija anvelope moduliranog signala (demodulacija), proizvedenog od strane *taga*, što ga čini samo *front end* uređajem.

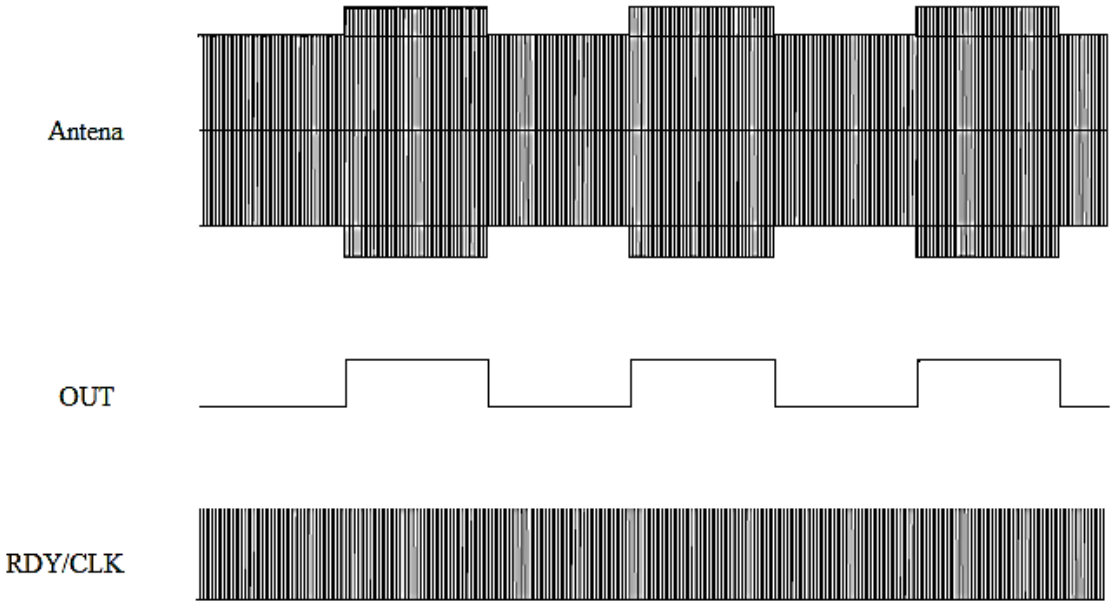
Uz pomoć samo nekoliko dodatnih komponenata EM4095 generira elektromagnetsko polje od 125 kHz potrebno za čitanje i pisanje RFID *tagova*. EM4905 je s mikro upravljačem povezan pomoću 4 signalne linije. One su RDY/CLK, MOD, SHD i DMOD_OUT. SHD linija definira *sleep* način rada. Kada se na njemu postavi signal logičke jedinice, EM4095 radi u *sleep* načinu rada i modul se smatra isključenim. Linija MOD određuje način rada (čitanje ili pisanje). Za čitanje ova linija postavlja se na nisku razinu signala. RDY/CLK linija služi za sinkronizaciju, na način da generira signal takta od 125kHz koji mikro upravljač koristi za daljnju demodulaciju. DEMOD_OUT pin sadrži demoduliran izlazni signal.

Dobiveni signal na mikro upravljaču se sastoji od dva dijela: prvi dio je signal takta sata 125kHz, dok je drugi podatkovni signal (čija je osnovna frekvencija jednaka signalu takta dijeljenog s faktorom 16, 32 ili 64). Sva daljnja demodulacije vrši se softverski i opisana je u dijelu programskog razvoja softvera za mikro upravljače.

Shematski prikaz RFID čitača prikazan je na Slici 4.19., a princip rada RFID čitača na Slici 4.20. Konektor CN1 služi za povezivanje RFID modula na EasyAVR5A razvojni sustav na način da se linije OUT i RDY/CLK spajaju na *interrupt* ulaze, radi obrade RFID signala. Potrebna sabirnica za spomenutu funkciju je PORTD.

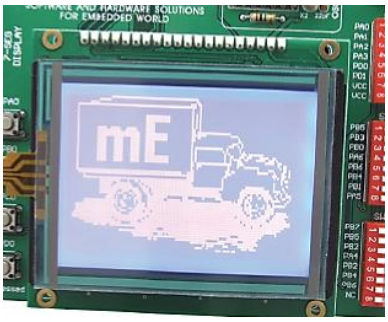


Sl. 4.19. Shematski prikaz RFID čitača [14]



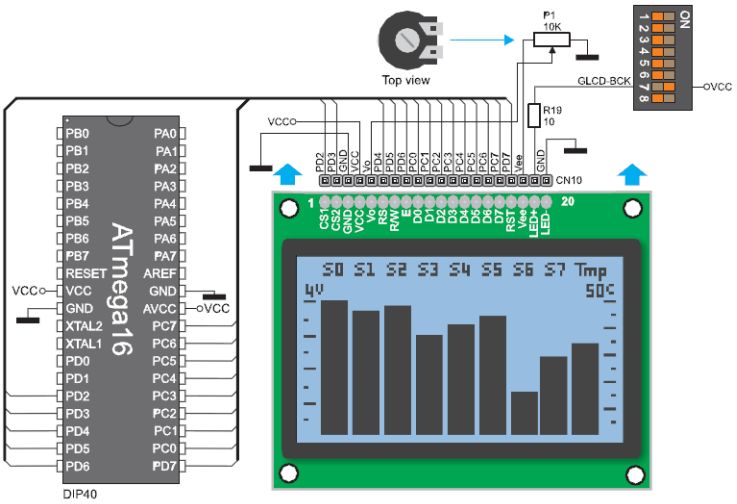
Sl. 4.20. Valni oblici signala na anteni i izlazu RFID čitača [14]

4.2.2. Grafički LCD ekran osjetljiv na dodir



Sl. 4.21. Grafički LCD ekran [9]

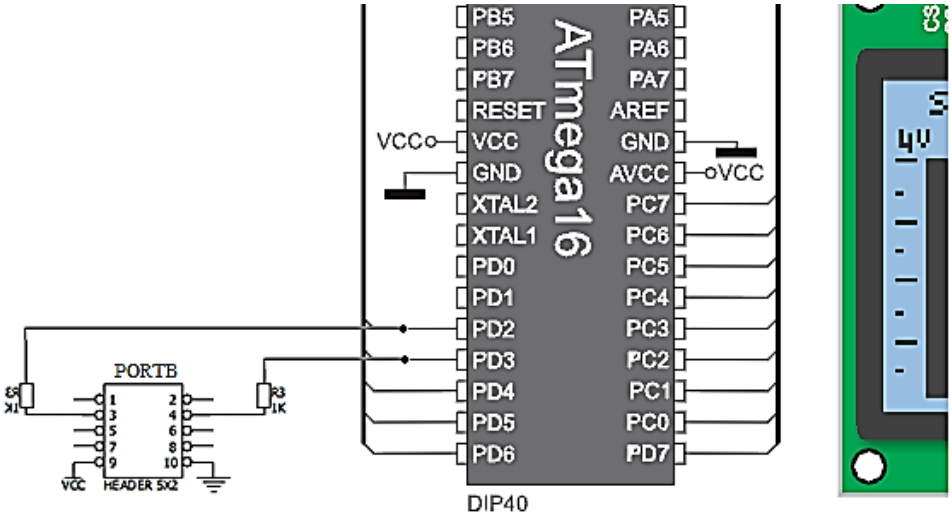
Grafički LCD ekran korišten je kao numerička tipkovnica, za unos osobnog identifikacijskog broja. LCD je grafičkog tipa s 128 x 64 točaka [9], a priključen je na mikro upravljač preko posebnog konektora i omogućuje prikazivanje grafičkog sadržaja. Kontrast ekrana se regulira pomoću posebnog trimer potenciometra integriranog u razvojni sustav. Shema spajanja grafičkog LCD ekrana prikazana je na Slici 4.22.



Sl. 4.22. Shema spajanja grafičkog LCD ekrana. [9]

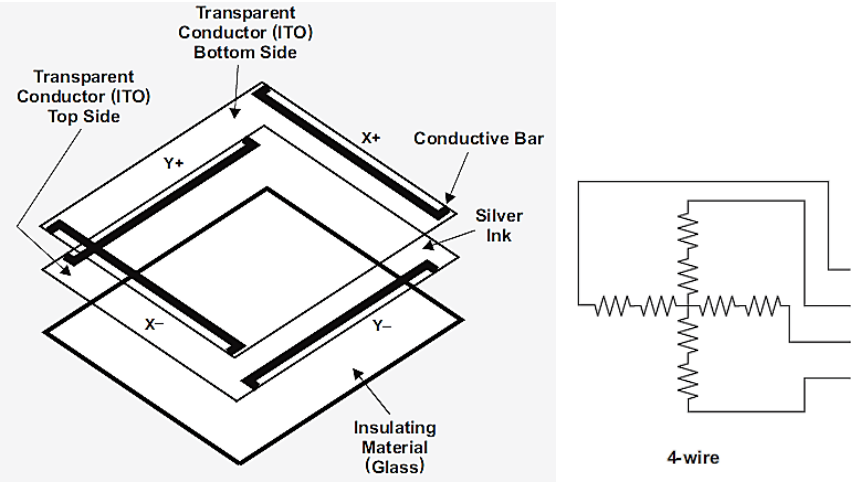
Prema Slici 4.22. vidimo da je grafički LCD spojen na sabirnicu PORTC mikro upravljača, kao podatkovna sabirnica, i na PORTD kao upravljačka sabirnica. Ovdje nastaje problem, jer se spajanjem LCD-a na sabirnicu PORTD onemogućilo korištenje *interrupt* ulaza koji su od velike važnosti za demodulaciju RFID signala. Problem je nastao u samom dizajnu razvojnog sustava koji se ne može ispraviti. Kada bi se sustav pristupnog čvora radio samo za tu svrhu, u obliku SMT ili THT tehnologije tiskanih pločica, taj bih se problem mogao vrlo jednostavno izbjeći na način da se linije PD2 i PD3 odspoje s LCD-a i prespoje na proizvoljne slobodne izlazne linije (kao npr. PA2 i PA3). Time bi se oslobodili potrebni pinovi PD2 i PD3.

Pošto promjena dizajna razvojnog sustava ne dolazi u obzir, primijenjeno je rješenje u kojemu su pinovi PD2 i PD3 korišteni kao bi direkcioni, na način da kada su potrebni za uporabu kao *interrupt* ulazi (za svrhu RFID čitača) oni su u stanju ulaza, a kada su potrebni za funkciju LCD-a prepuštaju se programu koji kontrolira rad LCD-a. Shema spajanja prikazana je na Slici 4.23 gdje su otpornici korišteni da bi spriječili interferenciju dvaju uređaja. Nakon preinake, PORTB je korišten za spajanje RFID čitača.



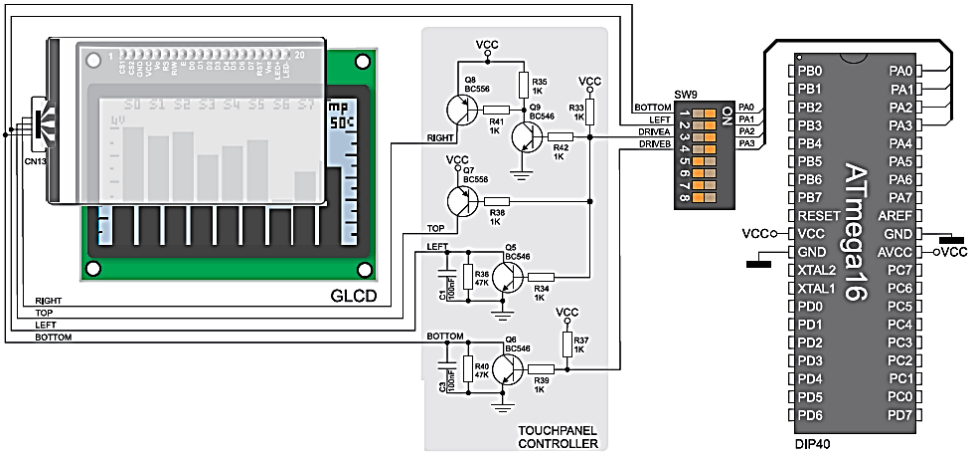
Sl. 4.23. Spajanje RFID čitača zajedno s grafičkim LCD-om [9]

Nakon spajanja grafičkog LCD ekrana, potrebno je spojiti i *touch screen* dodatak, koji pretvara LCD u uređaj osjetljiv na dodir. Princip rada je sljedeći: *touch screen* folija je kompaktni „sendvič“ s tri sloja. Donji sloj je staklo, koji je osnova. Na osnovu dolaze dva tanka sloja napravljena od sloja ITO (eng. *Indium Tin Oxide*) koji se ponaša kao otpornik. Slojevi su odvojeni i ne dodiruju se, prema [11] (Slika 4.24 a)).



Sl. 4.24. a) Izgled *touch screen* folije; b) ekvivalentna shema [11]

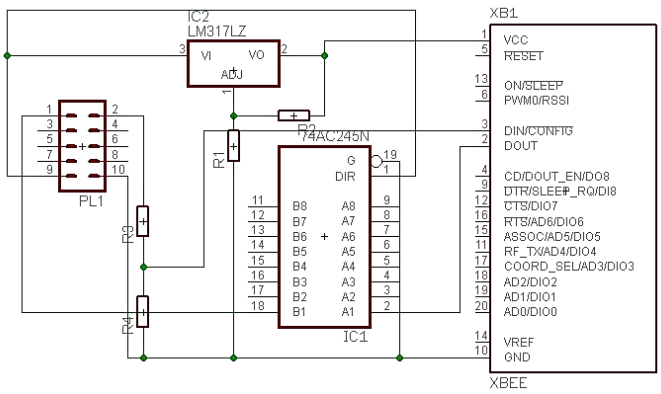
Kada dođe do pritiska na slojeve, dolazi do spoja između dva ITO sloja i dobiva se ekvivalentna shema prikazana na Slici 4.24 b). Mjerenjem otpora sva 4 otpornika određuju se koordinate mjesta gdje je izvršen dodir. Za mjerenje otpora se korisit ADC pretvornik, jer je pad napona na nekom otporniku uz konstantnu struju proporcionalan otporu. Shema spajanja *Touch screen* dodatka s mikro upravljačem prikazana je na Slici 4.25.



Sl. 4.25. Shema spajanja *touch screen* dodatka [9]

4.2.3. XBee modul u službi bežične komunikacije

Svaki pristupni čvor opremljen je s XBee modulom u službi komunikacije s glavnim čvorom. XBee moduli su u funkciji usmjerivača i krajnjeg čvor (radi mogućosti usmjeravanja paketa od drugih čvorova prema glavnom čvoru). Pošto je komunikacija kod XBee modula bazirana na UART sučelju, iz arhitekture AVR mikro upravljača vidi se da su kompatibilni s mikro upravljačem. Problem nastaje u sljedećem: AVR mikro upravljači (kao i RFID čitač a i grafički LCD ekran) rade na naponu napajanja od +5V, dok XBee moduli rade na naponskoj razini od 3.3V. Stoga je bilo potrebno izraditi naponski translator razina, koji prilagođava naponske razine. Najjednostavniji naponski translator čini otporno dijelilo, koje s napona od 5V (koristeći dva otpornika) smanjuje napon na 3.3V. Veći problem nastaje pri podizanju naponskih razina na 3.3V na 5V. Tada se koriste integrirani krugovi za tu funkciju. Jedan od tih integriranih krugova je 74HC245. Us translaciju razina načinjen je i izvor napajanja koji osigurava napon od 3.3V za napajanje XBee modula. Napajanje je izvedeno pomoću regulatora LM317 i odgovarajućeg para otpornika.



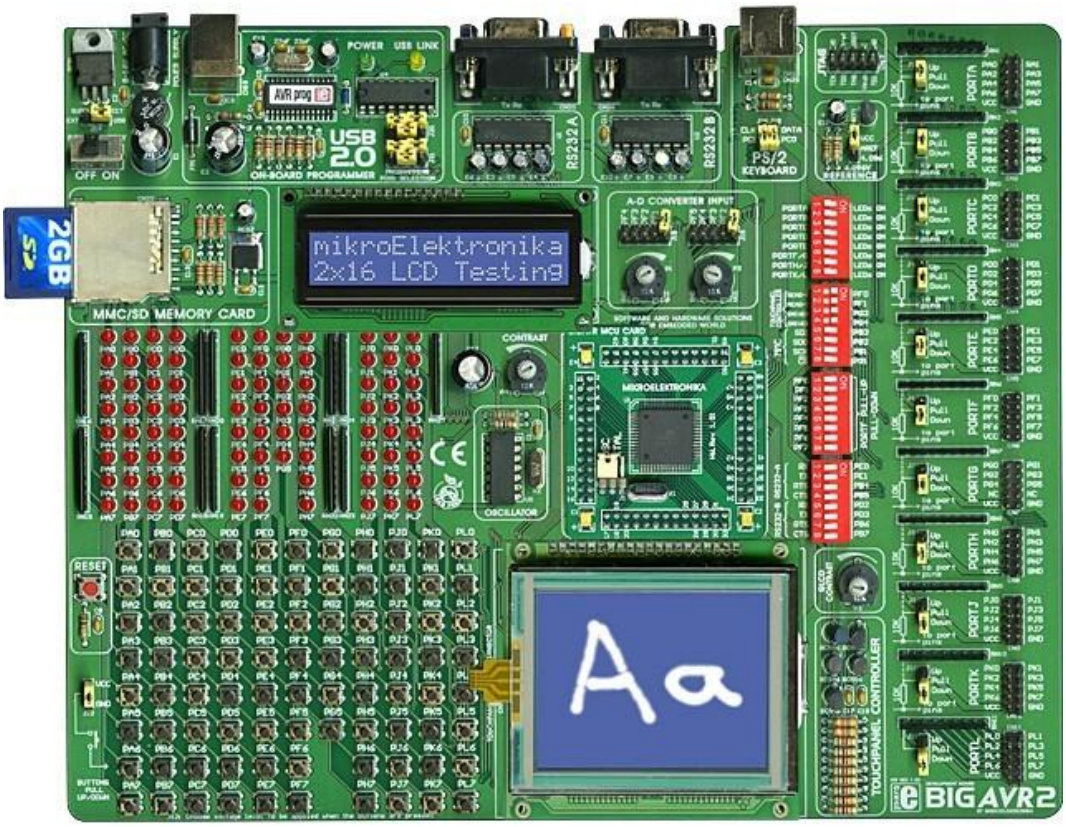
Sl. 4.26. Shema spajanja XBee modula s naponskim translatorom

Nakon arhitekture pristupnog čvora opisana je arhitektura glavnog čvora, koji osim mikro upravljača sadrži Ethernet sučelje, XBee modul u funkciji koordinatora, 16x2 LCD ekran za ispih kratkih informacija i stvarno vremenski sat (eng. *Real Time Clock*).

4.3. Dizajniranje arhitekture glavnog čvora

Glavni čvor je srce cijelog sustava bežične RFID autorizacije korisnika. Na njemu se nalazi glavni XBee koordinator, koji osniva PAN mrežu i spaja u mrežu sve ostale usmjerivače i krajnje čvorove. Također, na glavnom čvoru vodi se baza podataka u kojoj se nalazi evidencija svih korisnika u sustavu, s pripadajućim serijskim brojevima kartica i PIN brojevima. Isto tako u bazi podataka nalaze se datumi i vremena pristupa korisnika u sustav, za svaki pristupni čvor. Razmišljajući o mediju za pohranu baze podataka, zanimljiva se ideja postavila za uporabu SD/MMC memorijskih kartica, zbog činjenice da razvojni sustavi tvrtke „mikroElektronika“ sadrže integriran čitač spomenutih kartica. Stoga je metoda pohranjivanja baze podataka obarana kao SD memorijska kartica. Isto tako da bi bilo moguće upisivati datum i vrijeme pristupa korisnika, potreban je modul zvan RTC (eng. *Real Time Clock*), koji u sebi vodi evidenciju o točnom vremenu i datumu, a pristupa se preko I^2C sabirnice. I na posljetku dodano je *Ethenet* sučelje. Razvoj samog *Ethernet* uređaja i kreiranje programskih rutina za obradu prometa bio bi previše kompliciran u okvirima ovog rada, pa je za *Ethernet* komunikaciju iskorišten *Ethernet* modul koji je opisan u daljnjem tekstu.

Kao „srce“ glavnog čvora izabran je mikro upravljač AVR ATmega128, a za razvojni sustav koristi se sustav proizvođača „mikroElektronika“ pod nazivom BigAVR2. Frekvencija rada spomenutog mikro upravljača iznosi 11.059200 MHz (radi ostvarivanja kompatibilnost s UART modulom). Izgled razvojnog sustava prikazan je na Slici 4.27.



Sl. 4.27. Razvojni sustav BigAVR2 [10]

Prvi od opisanih perifernih uređaja je XBee modul. Zbog kompatibilnosti razvojnih sustava XBee modul i pripadajući naponski translatori identični su na glavnom i na pristupnom čvoru, a kao što je spomenuto jedina razlika između koordinatora i krajnjeg čvora je u *firmwareu* ili softveru upisanom u sam XBee modul. Hardverske razlike nema stoga se XBee modul nije ponovo obrađivao.

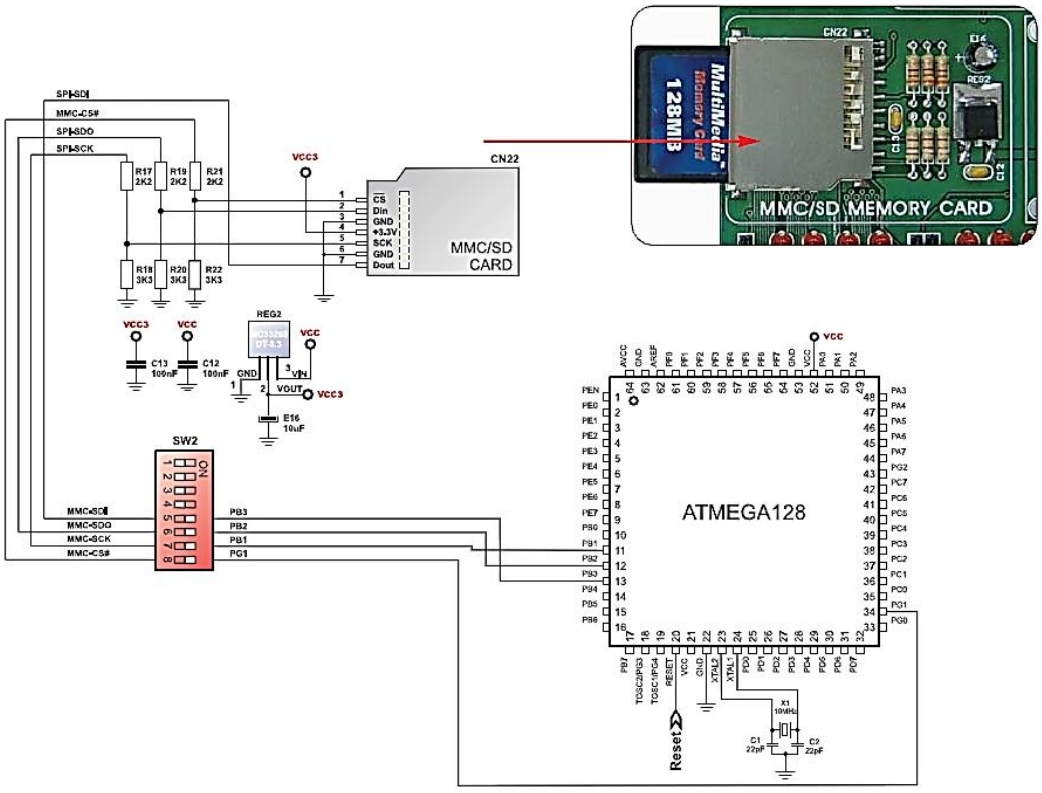
Sljedeći modul koji je obrađivan je MMC/SD adapter, koji ima svrhu sučelja između memorijske kartice i mikro upravljača.

4.3.1. MMC/SD modul za memorijske kartice

MMC/SD (eng. *MultiMediaCard*, *Secure Digital*) memorijske kartice se koriste kao medij za pohranu podataka u mobilnim uređajima iz kojih se mogu veoma lako izvaditi i staviti. Podaci se mogu veoma jednostavno pročitati i to ne samo na prijenosnim uređajima nego i na osobnim računalima uz pomoć čitača kartica. Moderna prijenosna računala najčešće imaju integriran čitač kartica kojim se može veoma jednostavno pristupiti sadržaju kartice. Zbog činjenice da se kod sustava bežične RFID autorizacije korisnika MMC kartica koristi kao medij

za pohranu baze podataka, očigledno je veoma lako spomenutu bazu pročitati na bilo kojem računalu, što olakšava posao obrade evidencije ili promjene korisnika u bazi. Samim time smanjuju se procesorski zahtjevi na sam mikro upravljač.

Komunikacija između mikro upravljača i MMC kartice odvija se preko SPI sabirnice. Zbog činjenice da MMC kartica ima napon napajanja od 3,3V potreban je naponski translator, kao što je slučaj s XBee modulom. Prednost MMC kartice je ta što je spomenuti naponski translator integriran u razvojni sustav BigAVR2. Shema spajanja MMC kartice na razvojnom sustavu prikazana je na Slici 4.28.

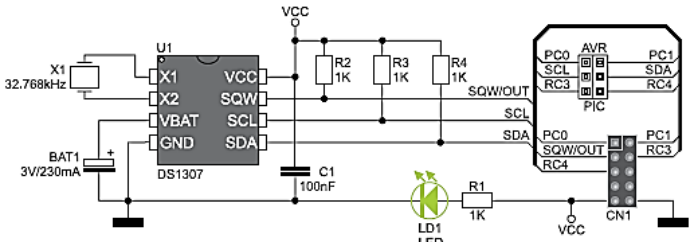


Sl. 4.28. Shema spajanja MMS/SD kartice u razvojnom sustav BigAVR2 [10]

4.3.2. RTC modul

Sl. 4.29. Izgled RTC modula [15]

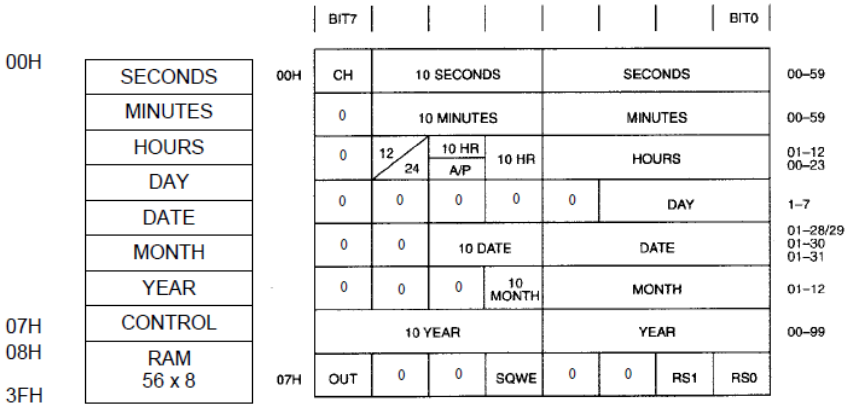
RTC (eng. *Real Time Clock*) ili Stvarno vremenski sat omogućuje mikro upravljaču da prati realno vrijeme i datum, uključujući korekciju za prijestupne godine i mjesece s manje od 31 dan. Zbog činjenice da sadrži baterijski izvor napajanja, stvarno vremenski sat će očuvati točno vrijeme i u slučaju kada mikro upravljač ostane bez napajanja. RTC komunicira s mikro upravljačem koristeći I^2C sučelje a spaja se na PORTD gdje postoje linije TWI modula (PD0 i PD1, prema [10]). Shema RTC modula prikazana je na Slici 4.30.



Sl. 4.30. Shematski prikaz RTC modula [15]

Srce RTC modula čini integrirani krug DS1307 male snage s BCD kodiranim satom i kalendarom. Specifikacije spomenutog integriranog kruga uključuju sposobnost održavanja točnog vremena i datuma, kompenzacija prijestupnih godina sve do 2100g, 56 bajta RAM memorije, I^2C sučelje, potrošnja energije iz baterije manje od 500nA idr. Sat i kalendar pružaju podatke kao što su: sekunde, minute, sati, dan u tjednu, dan, mjesec i godina. Sat ima opciju za rad u 24 satnom načinu rada ili u 12 satnom načinu rada, prema [8].

DS1307 radi u *Slave* načinu rada na serijskoj I^2C sabirnici. Pristup se postiže izdavanjem START naredbe, kao što je opisano u poglavlju 4.1.3. Adresna mapa RTC i RAM registara prikazana je na Slici 4.31. RTC registri se nalaze na memorijskim lokacijama 00H do 07H dok su RAM registri na memorijskim lokacijama 08H do 3FH.



Sl. 4.31. Memorijske lokacije i kodiranje bitova u DS1307 memoriji [8]

Iz Slike 4.31. vidi se da su podaci u memoriji BCD kodirani tako da prva polovica bajta označava prvu znamenku (jedinice) dok druga polovica bajta označava drugu znamenku (desetice). Kod pribavljanja podataka iz RTC modula prvo se u modul zapisuje početna adresa memorije 00H, a nakon toga se počinje čitanjem podataka bajt po bajt.

Na posljetku, najzahtjevniji modul je Ethernet modul koji je opisan u idućem poglavlju.

4.3.3. *Ethernet* modul

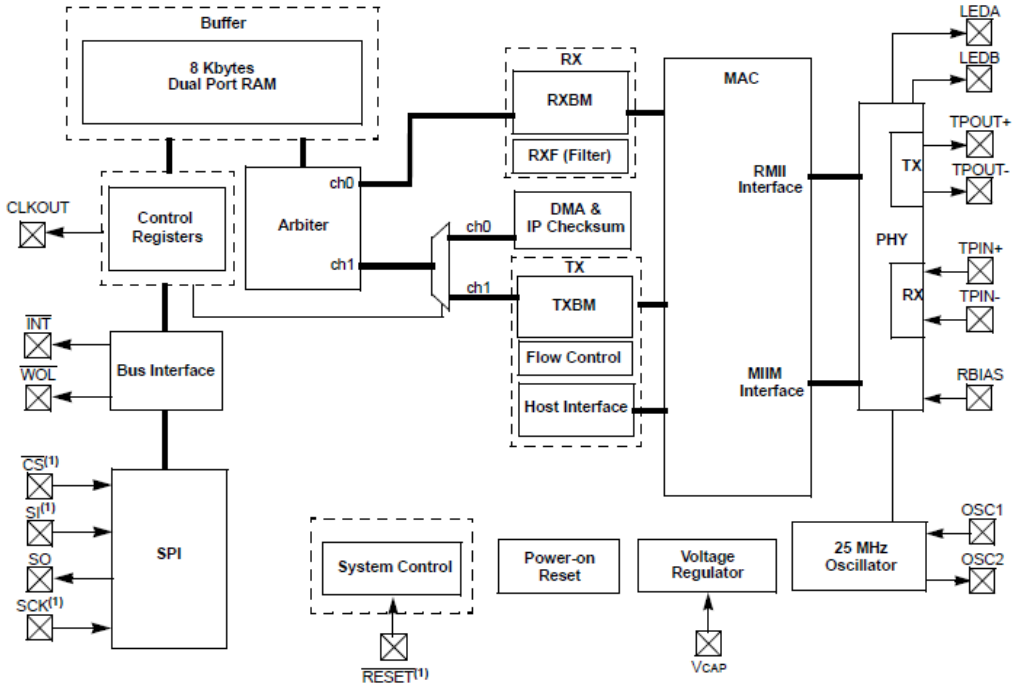
Serijski *Ethernet* modul koristi se pri spajanju mikro upravljača s lokalnom *Ethernet* mrežom. On sadrži *Ethernet* kontroler ENC28J60 koji izmjenjuje podatke između *Ethernet* mreže i mikro upravljača pomoću SPI sučelja. Brzine prijenosa kreću se do 10 Mbit/s. Modul služi kao *Ethernet* sučelje za sve mikro upravljače koji imaju pogodnosti SPI sabirnice.

ENC28J60 podržava sve specifikacije dane IEEE 802.3 normom. Također sadrži nekoliko načina filtriranja prometa (u svrhu ograničenja dolaznog prometa) te integrirani DMA (eng. *Direct Memmory Access*) modul za brzi prijenos podataka i hardversko računanje IP (eng. *Internet Protocol*) *checksuma*, prema [12]. Pojednostavljeni blok prikaz integriranog kruga prikazan je na Slici 4.32.

Integrirani krug sadrži sljedećih sedam osnovnih blokova:

1. SPI sučelje
2. Upravljački registri
3. RAM memorija, s dvostrukim ulazom, za pohranjivanje primljenih i poslanih paketa

4. Arbitar, koji upravlja s pristupom RAM memoriji, kada je zatražen pristup od strane DMA blokova
5. Sučelje sabirnice, koje tumači podatke i komande primljene od strane SPI sabirnice
6. MAC (eng. *Medium Access Layer*) modul, koji utjelovljuje IEEE 802.3 MAC logiku
7. PHY (eng. *PHYsical layer*) modul, koji kodira i dekodira analogne podatke prisutne na parici



Sl. 4.32. Blok dijagram integriranog kruga ENC28J60 [12]

Dosad prikazano je općeniti prikaz *Ethernet* modula, i zbog kompleksnosti samog integriranog kruga i mnoštva instrukcija i naredbi detaljan opis nije proveden. Sve dodatne informacije mogu se pronaći u [12]. Adresiranje i rad s *Ethernet* modulom proveden je pomoću postojećih bibliotečnih datoteka (eng. *Library*) koje su napravljene da bih se olakšao rad s kompleksnim sklopovima i protokolima.

Nakon provedene analize arhitekture cijelog RFID sustava, izvršen je programski razvoj glavnog i pristupnog čvora. Pošto je cijeli razvoj počeo s RFID čitačem, daljnji opis u radu prati tu nit. Na prvom mjestu izvršen je programski razvoj RFID čitača, nastavljeno je s razvojem XBee mreže i razvojem baze podataka a završeno s razvojem *Ethernet* sučelja i osmišljavanja protokola komunikacije s LabView aplikacijom.

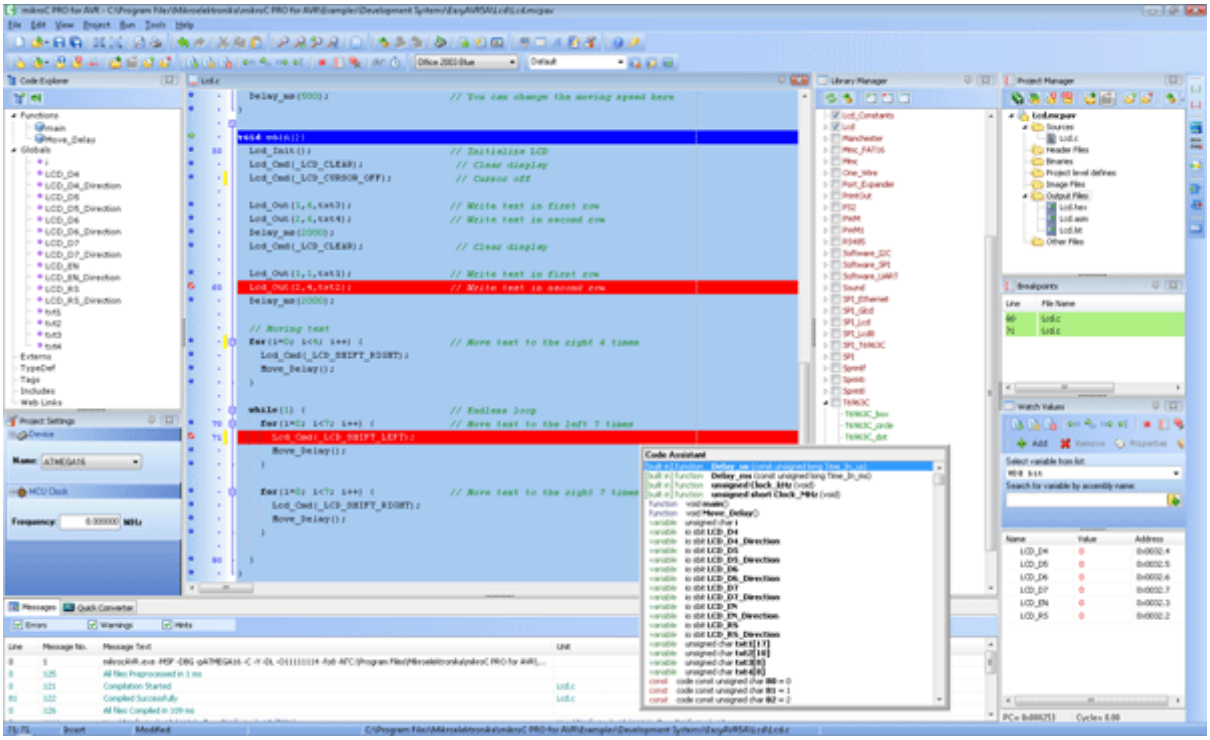
5. PROGRAMSKI RAZVOJ GLAVNOG I PRISTUPNOG ČVOROVA

Kao što je rečeno u prethodnom poglavlju, srce cijelog sustava čine mikro upravljači na strani glavnog pristupnog čvora. No sam mikro upravljač bez programa nema potpuno nikakvu svrhu, pa je stoga bilo potrebo osmisliti programsku potporu koja prazni mikro upravljač popunjava s funkcionalnim programom. Budući da je sustav bežične RFID autorizacije razvijan počevši od RFID čitača i pristupnog čvora pa sve do glavnog čvora i vođenja baze podataka, tako i ovaj rad slijedi spomenutu nit.

Kako bi programiranje mikro upravljača bilo olakšano, i da se izbjegne programiranje u čistom strojnom jeziku (asembleru) koristi se programski alat koji je kompatibilan s razvojnim sustavima tvrtke „mikroElektronika“ i korištenim mikro upravljačima.

5.1. MikroC PRO za AVR mikro upravljače

MikroC PRO je moćan alat s mnogim dodatcima i opcijama, a namijenjen je razvoju AVR mikro upravljačkih sustava. Osmišljen je kako bi programerima olakšao programiranje i razvijanje aplikacija za ugrađene sustave, bez žrtvovanja performansi ili upravljanja.



Sl. 5.1. Izgled razvojnog alata MikroC za AVR [19]

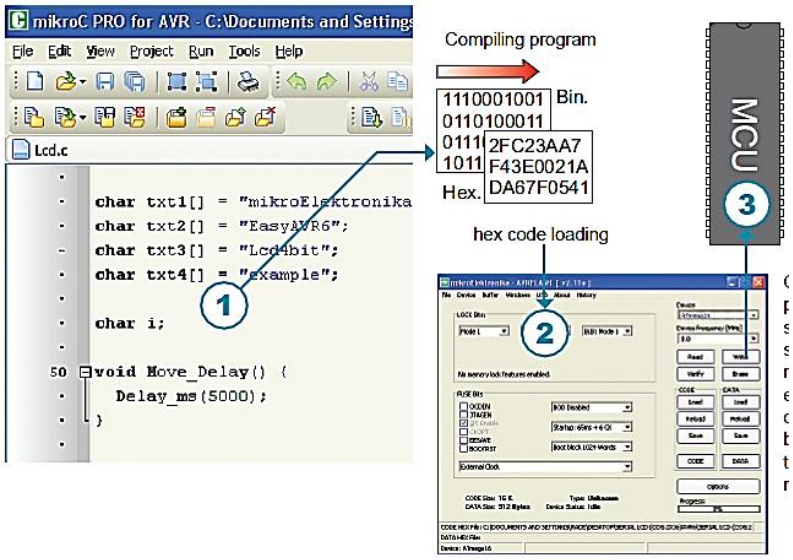
Osnovne značajke MikroC razvojnog alata su [19]:

- Pisanje vlastitog izvornog koda u C programskom jeziku pomoću ugrađenog *Code Editora*
- Korištenje postojećih bibliotečnih datoteka (eng. *Library*) prilagođenih za AVR mikro upravljače čime se drastično smanjuje vrijeme programiranja
- Nadgledanje strukture programa, korištenih varijabli i funkcija
- Generiranje asemblerskog koda i standardnih HEX datoteka koje su kompatibilne sa svim programatorima
- Ispitivanje toka programa uz ugrađeni simulator
- Dobivanje detaljnih grafova i izvještaja: ROM i RAM memorije, statistike koda idr
- Uz sve navedeno, dostupni su mnogi primjeri koda

Iz osnovnih značajki razvojnog alata MikroC vidimo da je osnova za pisanje aplikacija programski jezik C, što je veoma pozitivno jer je C kao programski jezik jako zastupljen i većina programera ima iskustva s njim. Zbog spomenutog, pratiti tok koda programa od njegovog početka pa do kraja uvelike je olakšano. U narednom tekstu poziva se na linije koda koji je dostupan u Prilogu 1.

Nakon napisanog koda potrebno je dotični i upisati u sam mikro upravljač. Zbog kompatibilnosti razvojnog alata i razvojnog sustava (oba proizvod tvrtke „mikroElektronika“), programiranje mikro upravljača je veoma jednostavno. Razvojni sustav povezan je s osobnim računalom preko USB sučelja, i pomoću opcije *mE Programmer* prevedeni se sadržaj iz C jezika u asembler upisuje se izravno u mikro upravljač. Nakon programiranja mikro upravljač sam vrši programiranu funkciju bez potrebe za osobnim računalom.

Princip programiranja prikazan je na Slici 5.2.



Sl. 5.2. Način programiranja mikro upravljača [19]

5.2. Razvoj pristupnog čvora

Pristupni čvor sastoji se od mikro upravljača koji je srce čvora, RFID čitača, XBee bežičnog modula i LCD ekrana osjetljivog na dodir za unos PIN-a i prikaza grafičkog sučelja korisniku (Slika 4.1). U idućem poglavlju opisan je razvoj softvera za uspješan rad RFID čitača, LCD ekrana i za uspješnu komunikaciju s glavnim čvorom koristeći XBee modul.

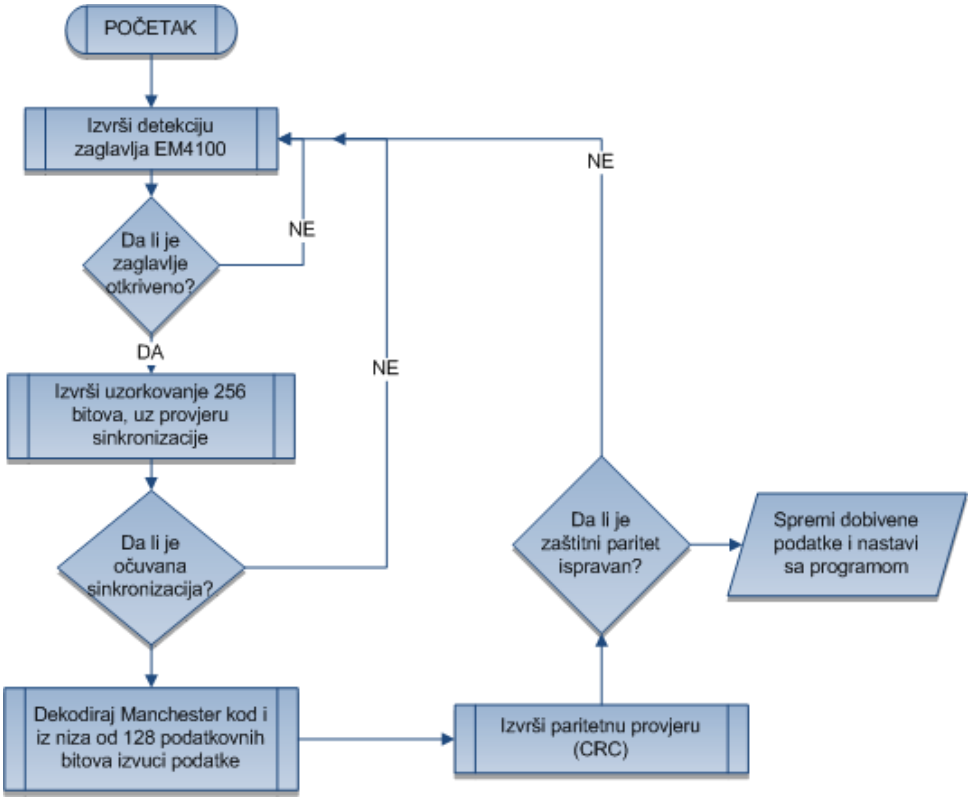
5.2.1. Zasnivanje RFID čitača

Razvojni sustav pristupnog čvora spojen je s RFID čitačem preko sabirnice PORTB, koja je preko otpornika vezana na *interrupt* ulaze (PD2 i PD3). Kako bi uporaba RFID čitača bila moguća potrebno je spomenute linije koristit kao ulaze. Konfiguracija spomenutih linija se u toku programa mijenja ovisno o potrebi korištenja RFID čitača ili LCD ekrana.

Općenito rečeno, dio programa koji obrađuje informacije dobivene od RFID čitača mora imati sljedeće funkcije:

- Uzorkovanje linije podatkovnog signala (OUT), u ovisnosti o liniji takta (CLK), dobivene od strane RFID čitača.
- Dekodiranje Manchester koda koje se koristi u EM4100 protokolu za RFID *tagove*.
- Detektiranje početka slijeda podataka (zaglavlje od devet logičkih jedinica).
- Provjera pariteta RFID podataka.
- Prosljeđivanje dobivenog serijskog broja daljnjim potprogramima.

Dijagram toka RFID čitača prikazan je na Slici 5.3.



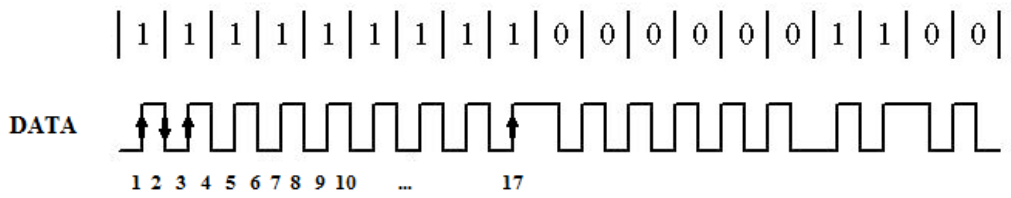
Sl. 5.3 Dijagram toka programa RFID čitača

Kako bi se navedeni zahtjevi mogli ispuniti prvo je potrebno detektirati postojanje *taga* (u radu izveden kao kreditna kartica) u elektromagnetskom polju čitača. U idealnim okolnostima kada se u EM polju čitača ne nalazi kartica podatkovna linija OUT nalazi se u stanju logičke nule. Tada se pri pojavi rastućeg brida na spomenutoj liniji moglo zaključiti da je kartica u polju čitača. No u praksi je slučaj kompliciraniji zbog velike količine elektromagnetskih smetnji u okolišu, koje su produkt velikog broja električkih uređaja. Tada dolazi do pojave da se na izlazu čitača pojavljuju se smetnje u obliku električnih impulsa nepravilnog trajanja i nepredvidivog pojavljivanja. Razlog tomu je činjenica da RFID čitač ima za zadaću demodulaciju modulacije povratnog radijskog signala čiji je indeks modulacije veoma nizak (reda veličine 1%) pa svaka mala promjena amplitude daje impulse na izlazu.

Rješenje spomenutog problema detekcije RFID *taga* nalazi se u Poglavlju 2.4.1., zbog činjenice da EM4100 ima format zaglavlja definiran tako da zaglavlje čini 9 bita koji su jednaki logičkim jedinicama. Zbog strukture protokola u polju podataka se nikako ne može ponoviti slijed od 9 logičkih jedinica, što znači da se spomenuto zaglavlje koristi kao metoda detekcije prisutnosti kartice. Pošto je u ovom slučaju način kodiranja signala izveden *Manchester* kodom, tada je poznato da se traži devet uzastopnih perioda pravokutnog signala, prema Sl. 2.5. Kao što

je opisano u poglavlju 4.2.1 RFID *tag* koristi frekvenciju prijenosnog signala kao signal takta za obradu podataka, a izlazni niz podataka je frekvencije prijenosnog signala dijeljene s određenim faktorom. U ovom slučaju frekvencija podataka je $f_p/64$, što znači da jedan podatkovni bit traje 64 perioda signala takta.

Nakon što je poznato trajanje podatkovnog bita, daljnja detekcija devet uzastopnih jedinica veoma je olakšana: mikro upravljač počinje detekciju kartice na način da prati pojavu rastućeg ruba signala na podatkovnoj liniji. Kada detektira prvi rastući brid mikro upravljač zbraja periode signala takta sve do idućeg brida (koji je padajući). Ako je broj perioda takta sata jednak 32 (-/+ 4 periode, tolerancija) nastavlja od trenutnog ruba prema idućem rubu, sve dok ne detektira 17 uzastopnih promjena razina, što označuje 9 uzastopnih logičkih jedinica, Sl. 5.3.



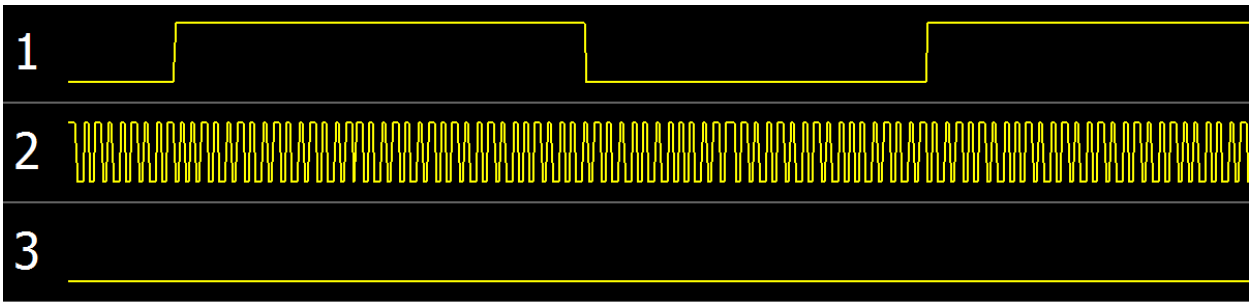
Sl. 5.3. Detekcija zaglavlja

Spomenuta metoda detekcije osmišljena je koristeći specifikacije EM4100 protokola i samog *Manchester* kodiranja, ali kako bi se spomenuta metoda potvrdila kao ispravna potrebno je snimiti valne oblike signala dobivene od strane modula RFID čitača. Kako bi se snimili valni oblici digitalnog signala potrebno je koristiti Logički analizator signala. Logički analizator je mjerni instrument sličan digitalnom osciloskopu. Osnovna razlika je u njegovoj prilagođenosti promatranju signala u digitalnom sustavu. Digitalni signali koji su mjereni jesu signal takta CLK i podatkovni signal DTA. Dobiveni signali prikazani su na Slici 5.3.

Sl. 4.4. Valni oblici digitalnih signala s RFID čitača

Iz Slike 5.4. uočava se da se pod oznakom 1 nalazi podatkovni signal dok se pod oznakom 2 nalazi signal takta koji je puno veće frekvencije od podatkovnog signala. Ako se

pogleda podatkovni signal onda se primjećuje da se na uzorku podataka u dva slučaja javlja zaglavlje od 9 logičkih jedinica (na početku uzorka i pred kraj) a podaci između predstavljaju informaciju zapisanu u *tagu.* Kada se signal dekodira iz *Manchester* koda dobije se format podataka specificiran EM4100 protokolom. Kako bi se dokazalo da je signal takta jednak frekvenciji koja je 64 puta veća od signala podataka, signal je uvećan i prikazan na Slici 5.5.



Sl. 5.5. Valni oblici signala s RFID čitača, uvećano

Iz Slike 5.4 vidi se da je trajanje jednog podatkovnog bita uistinu 64 takta prijenosne frekvencije i na temelju spomenutog može se zaključiti da je metoda detekcije 17 uzastopnih rubova ispravan način detektiranja postojanja kartice.

Nakon što je mikro upravljač detektirao 17 uzastopnih rubova s trajanjem od 32 takta sata (64 takta traje jedan bit, a pošto je kodirano Manchester kodom, svakih pola bita postoji promjena, pa je razmak između dvije promjene naponske razine 32 takta) počinje uzorkovanje podataka.

Spomenuta detekcija zaglavlja nalazi se u dvije prekidne rutine na linijama 300. i 309. Prva prekidna rutina (linija 309.) zadužena je za brojanje taktova signala sata frekvencije 125kHz, koji se dobiva od strane RFID čitača i ona se pokreće pri detekciji rastućeg ruba na liniji CLK (PD2). Prekidna rutina na liniji 309. ima za zadaću ispitivanje trajanja impulsa i odlučivanje da li je detektirani impuls dio zaglavlja ili nije. Ispitivanje se vrši na liniji 317. Spomenuta prekidna rutina je aktivna samo za vrijeme detekcije zaglavlja i kada je zaglavlje detektirano više se ne koristi. Ova rutina alternira detekciju brida na način da kada detektira rastući brid automatski prebacuje detekciju na padajući brid i obratno. Time se dobiva mogućnost detekcije uzastopnih padajućih i rastućih bridova, prema Sl. 5.3. U toku rada detekcije zaglavlja glavni program je zaustavljen i ne izvodi se sve dok se ne postavi zastavica koja označava detekciju zaglavlja, linija 623.

Nakon što je detektirano zaglavlje program se priprema za uzorkovanje signala. Uzorkovanje se vrši svakih 32 takta sata ali na način da se uzorci uzimaju na ¼ i na ¾ trajanja

bita (zbog činjenice da su na spomenutim vremenima podaci sigurno postojani i ne postoji opasnost od uzorkovanja na rubu signala). Uzorkovanje podataka izvodi se na linijama 634 do 647. Dodatna sigurnosna funkcija koja je dodana je provjera sinkronizacije koja funkcionira na sljedeći način: Prema Manchester kodu, u svakom podatkovnom bitu postoji promjena logičkog stanja i ovisno o smjeru promjene dekodira se da li je podatkovni bit logička nula ili logička jedinica. Zbog činjenice da u svakom podatkovnom bitu postoji rastući ili padajući rub, postoje i dva komplementarna logička stanja. Provjera sinkronizacije analizira podatkovne bitove i traži postojanje dva komplementarna logička stanja koji označuju da je podatkovni niz ispravan. Ako se u kojem slučaju dogodi da se izgubi sinkronizacija (zbog smetnje ili drugog razloga) program se vraća na početak i započinje detekciju kartice i zaglavlja.

Nakon uzorkovanja podataka, i provjere sinkronizacije, u memoriji mikro upravljača pohranjeni su podaci s kartice. Daljnjim dekodiranjem Manchester koda dobiva se čisti podatkovni niz od 128 podatkovnih bitova koji u sebi sadržavaju 64 bita podataka. Daljnje izdvajanje podatkovnih bitova vrši se opet na način detekcije zaglavlja, koje sadrži 9 logičkih jedinica, ali sada na softverskoj razini (s izvornim podacima) za razliku od detekcije zaglavlja koje funkcionira na nižoj razini. Ova dio programa nalazi se između linije 666. i 695. Nakon ovog dijela programa u memoriji je spremljen podatkovni niz sa Slike 2.9.

Posljednji korak je provjera križnog pariteta s paritetnim algoritmom. Provjera pariteta odvija se u potprogramu *CRC_Check* koji se nalazi na linijama 348 do 405 i za rezultat vraća 1 ili 0, ovisno o ispravnosti ili neispravnosti pariteta. Ako je paritet neispravan program se vraća na početak detekcije zaglavlja.

Nakon što je dokazan ispravan paritet sirovi podaci se obrađuju u ASCII formu i pripremaju za slanje glavnom čvoru, linije 823. do 839. Podaci koji su dobiveni predstavljaju serijski broj *taga* tj. RFID kartice i spremljeni su u heksadekadskom obliku od 10 znamenki (40 bita).

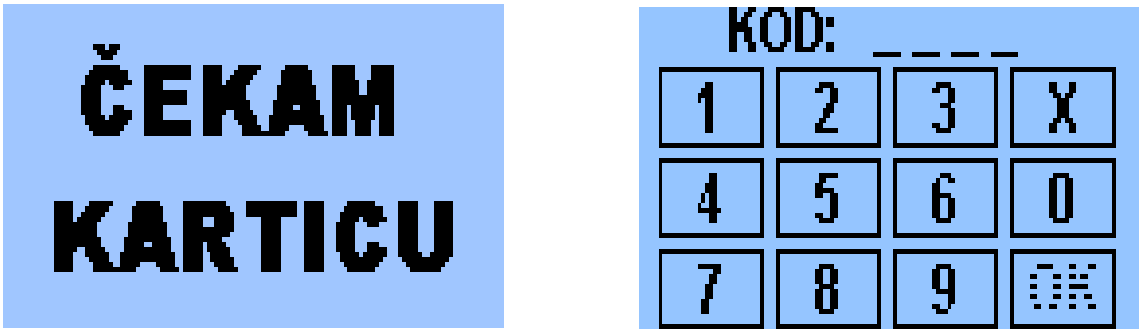
Nakon obrađenog RFID prijemnika razrađen je grafički sustav LCD ekrana osjetljivog na dodir, u funkciji sučelja za unos osobnog PIN broja.

5.2.2. LCD ekran osjetljiv na dodir u službi unosa koda

Kao što je opisano u poglavlju 4.2.2. grafički LCD ekran koristi se za prikazivanje grafičkog sadržaja u službi interakcije s korisnikom. Njegovo dodatno svojstvo je osjetljivost na dodir (eng. *Touch Screen*). Uz spomenuta svojstva ovaj LCD može veoma jednostavno poslužiti kao numerička tipkovnica za unos podataka. LCD modul vezan je s mikro upravljačem pomoću dvije sabirnice, od kojih je jedna podatkovna sabirnica a druga je upravljačka. Za ostvarivanje veze s LCD modulom korištene su bibliotečne datoteke (eng. *Library*) za upravljanje i prikazivanje simbola na LCD ekranu.

Kako bi se LCD modul mogao koristiti potrebno ga je inicijalizirati. Inicijalizacija grafičkog LCD ekrana izvedena je na linijama 516 do 518. Nakon inicijalizacije LCD je spreman za prikazivanje podataka. Zbog činjenice da je adresiranje LDC ekrana standardno moguće je postojeće *bitmap* slike pretvoriti u oblik koji se može prikazati na LCD-u. Dio koja koji sadrži slike nalazi se na linijama 36 do 178. Pošto je rezolucija ekrana 64x128, veličina svake slike koja se pohranjena iznosi 1kB, što se ne čini velikim zauzećem memorije ali mora se uzeti u obzir da mikro upravljač ima samo 32kB programske memorije, pa tu činjenicu treba imati na umu pri integriranju slika u programski kod.

Program sadrži dvije glavne slike gdje je prva slika tekst: „Čekam karticu“ dok je druga slika numerička tipkovnica za unos koda. Izgled LCD ekrana za spomenute dvije slike prikazan je na Slici 5.6. Prikaz slika izvodi se pomoću naredbe *GLCD_Image()*, kao u liniji 707.



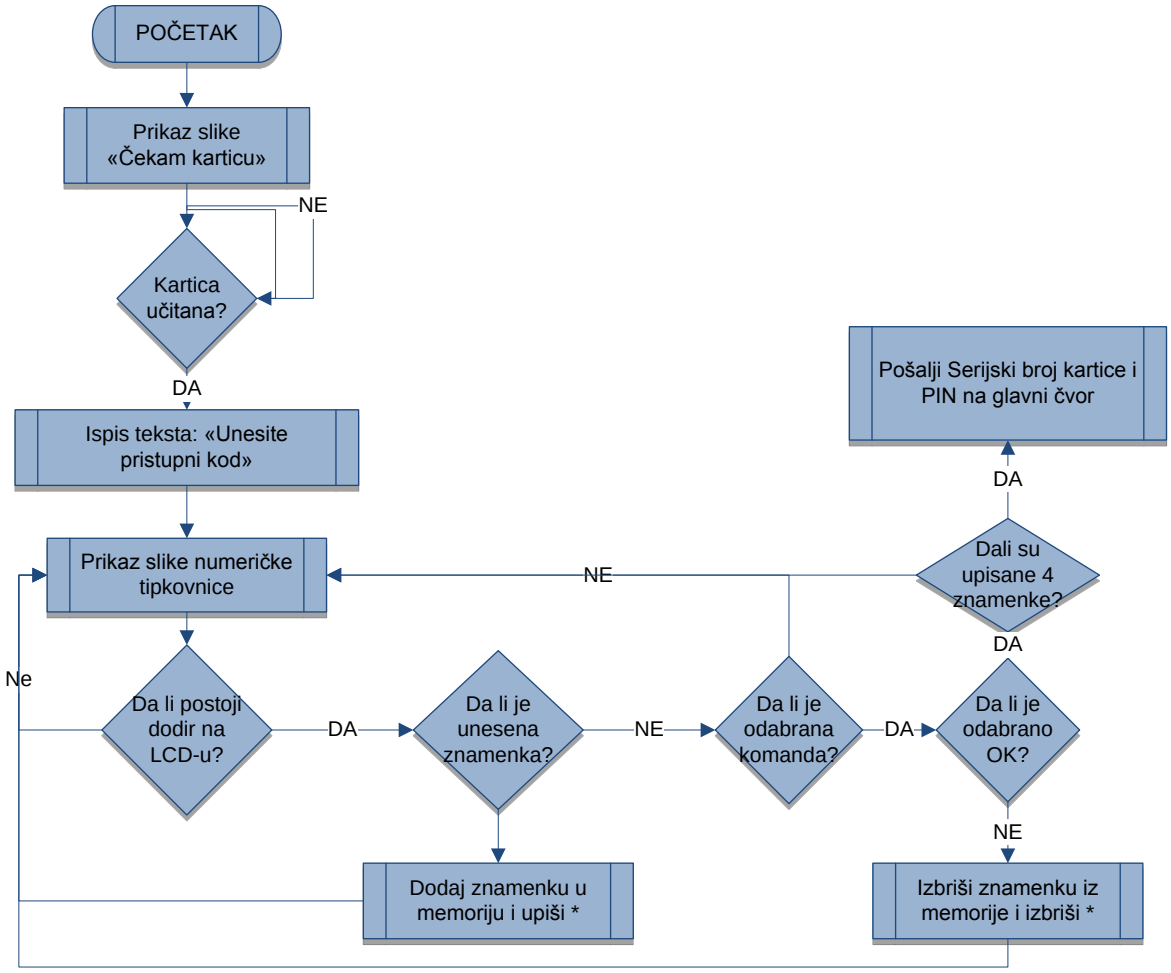
Sl. 5.6. Izgled LCD-a za slike „Čekam karticu“ i numeričku tipkovnicu

Sljedeći dio odnosi se na implementaciju *touch screen* funkcije u službi numeričke tipkovnice. Princip rada *touch screen* dodatka opisan je u poglavlju 4.2.2. gdje je prikazan način određivanja položaja uz pomoć otpora otpornika. Određivanje *X* i *Y* koordinata vrši se vrlo jednostavno. Za određivanje *X* koordinate na izlaz *Drive_A* postavlja se logička jedinica a na izlaz *Drive_B* logička nula. Nakon toga očitava se vrijednost analognog signala na ulazu *Bottom*

i ta vrijednost predstavlja položaj u *X* osi. Slično se određuje i položaj *Y* osi što se može vidjeti iz linija 234 - 254. No dobiveni rezultat ne predstavlja položaj u koordinatama *pixela* na LCD ekranu, pa se u toku rada taj položaj preračunava. Također, bitna stvar za napomenuti je i kalibracija. Naime, zbog tvorničkog procesa i vanjskih uvjeta vrijednost koordinata dobivena ovim putem nije dovoljno točna. Stoga se koristi princip kalibriranja *touch screen* dodatka koji je odrađen pomoću funkcije *Calibrate()* koja se nalazi na liniji 256. Nakon kalibracije podaci se upisuju u EEPROM memoriju i prilikom ponovnog pokretanja pristupnog čvora ti se podaci iščitavaju iz spomenute permanentne memorije, što se može vidjeti iz linija 524 – 539. Ako se prilikom pokretanja postavi logička jedinica na ulaznu liniju PA7 pokreće se SETUP sustav prilikom kojega se vrši kalibracija LCD-a.

Osim mogućnosti prikaza slika LCD ima i mogućnost ispisa teksta u standardnom formatu. Ekran je podijeljen u 8 tekstualnih linija i sukladno tome ispisuju se željene poruke.

Nakon što je ostvarena funkcija osjetljivosti ekrana na dodir nastavlja se program s dijelom unosa koda pomoću numeričke tipkovnice. Sve započinje na liniji 707 gdje se prikazuje slika Unosa (Slika 5.6.). Nakon prikaza slike program ulazi u beskonačnu petlju gdje opetovano provjerava stanje ekrana osjetljivog na dodir, i da li je došlo do dodira. Ako je do dodira došlo računaju se koordinate dodira koje se dalje obrađuju. Ako je unos bio na dijelu tipkovnice s brojevima, unosi se znamenka, a ako položaj odgovara tipkama *OK* ili *X* sukladno tome se izvršava funkcija. Tipka *X* ima za funkciju brisanja zadnje unesene znamenke dok tipka *OK* ima za funkciju potvrdu unosa. Tipka *OK* postaje funkcionalna tek kada su unesene sve 4 znamenke koda, nakon čega poprima puni oblik. Spomenute funkcije odvijaju se pri linijama 709 – 795. Također, dodana je grafička potpora korisniku koji unosi PIN na način da se pri unosu svake znamenke na vrhu ekrana kod teksta *KOD* pojavljuju simboli „*“, koji označuju broj unesenih znamenki. Kada se linija popuni s 4 zvjezdice, tipka *OK* dobije puni oblik i korisnik može potvrditi svoj pristupni kod pritiskom na *OK* komandu. Tok programa prikazan je na dijagramu toka na Slici 5.7.



Sl. 5.7. Dijagram toka od uključivanja pristupnog čvora do slanja podataka

5.2.3. XBee sustav za prijenos podataka

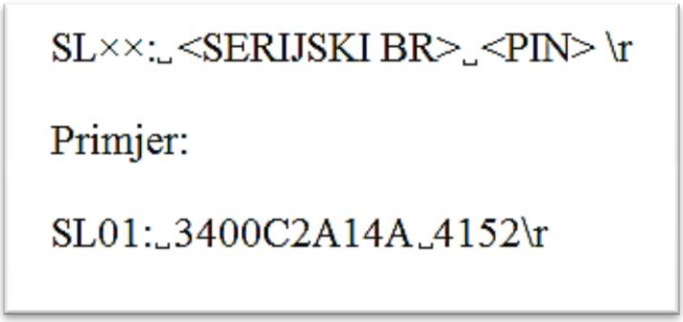
XBee moduli s pripadajućim translatorom naponskih razina imaju višestruku primjenu. Jedan XBee modul može služiti kao koordinator, usmjerivač ili kao krajnji čvor. U slučaju pristupnog čvora XBee modul konfiguriran je kao usmjerivač i krajnji čvor. Tim postavkama omogućen je prijenos podataka uz istovremenu mogućnost usmjeravanja paketa kroz mrežu. Pri prvom pokretanju XBee modul je potrebno inicijalizirati s parametrima kao što su PAN ID (ime PAN mreže) i ime čvora, uz uvjet da je *firmware* modula namijenjen za uporabu kao usmjerivač/krajnji čvor. Ako to nije slučaj, potrebno je XBee modul povezati s računalom pomoću RS232 sučelje i programirati željeni *firmware*, kako je opisano u poglavlju 3.4.3. Inicijalizacija XBee modula radi se samo pri prvom pokretanju i može je obaviti mikro upravljač. Slično kao što je bila riječ kod kalibracije ekrana osjetljivog na dodir, na isti način se inicijalizira i XBee modul i to tako da se pri podizanju postavlja logička jedinica na liniju PA7. Pri takvom podizanju javlja se poruka koja traži unos rednog broja pristupnog čvora. Nakon

odabira rednog broja pristupnog čvora, mikro upravljač automatski postavlja postavke i komunikacija postaje moguća. Inicijalizacija XBee modula nalazi se na programskim linijama 435 do 495.

Komunikacija s XBee modulom obavlja se preko USART modula i to korištenjem bibliotečnih datoteka za potporu USART modulu. Naredbe koje se koriste jesu *UART1_Write()* , koja šalje podatke na UART, i *UART1_Read()*, koja prima podatke poslane mikro upravljaču. Konfiguriranje parametara vrši se ulaskom u komandni način rada, koji je prije objašnjen, izvodi se pri linijama 480 do 490.

Nakon što je modul konfiguriran on je automatski spreman za uporabu i primanje podataka. Kod krajnjih čvorova postoji pravilo da bilo koji serijski podatak koji XBee modul primi, automatizmom se preusmjerava na koordinator, u ovom slučaju glavni čvor. Stoga u prijenosu podataka od pristupnog čvora prema glavnom čvoru XBee sustav glumi standardnu serijsku liniju i nikakva podešavanja nisu potrebna u toku rada. Ovom činjenicom veoma je olakšan rad na strani pristupnog čvora.

Na strani pristupnog čvora komunikacija se odvija na sljedeći način: Pristupni čvor sažima informacije kao što su: serijski broj kartice, pristupni kod i broj pristupnog čvora u jednu tekstualnu poruku i šalje na UART sučelje. Spomenuto sučelje preko XBee modula prosljeđuje poruku na glavni čvor. Nakon toga pristupni čvor čeka potvrdu od strane glavnog čvora: da li je autorizacija uspješna, da li korisnik postoji u bazi i da li je ispravan pristupni kod. Zahtjev koji se šalje glavnom čvoru ima znakovni ASCII oblik i sljedećeg je formata: Prvi znakovi u nizu koji se šalje jesu znakovi „SL“, što označava da se radi o poruci koja je poslana od strane pristupnog čvora. Iduća dva simbola označavaju redni broj pristupnog čvora nakon kojih slijedi simbol „:“ i podaci koji se šalju. Prvi od podataka je serijski broj RFID kartice koji je dobiven dekodiranjem sirovih RFID podataka, opisanog u poglavlju 5.2.1. Nakon toga slijedi pristupni kod ili PIN a sve je zaključeno s znakom nove linije 0dH. Format zahtjeva prikazan je na Slici 5.8. Dio programa koji je odgovoran slanju zahtjeva nalazi se na linijama 839 – 848.



Sl. 5.8. Format zahtjeva poslane od pristupnog čvora prema glavnom čvoru

Nakon poslanog zahtjeva glavnom čvoru, on odgovara jednom od potvrda. Potvrda od strane glavnog čvora sadrži svega nekoliko simbola:

- Ako je autorizacija uspješna – pristupni čvor prima poruku sadržaja „OK <*Ime korisnika*>“ pri čemu se na LCD ekranu ispisati poruka „Autorizacija uspješna“ i na dnu ekrana ime i prezime korisnik, koji se prijavio u sustav
- Ako je neispravan pristupni kod – pristupni čvor prima poruku sadržaja „PIN“, što znači da je pristupni kod neispravan i autorizacija nije uspješna.
- Ako korisnika nema u bazi podataka – pristupni čvor čeka potvrdu od glavnog čvora, ali glavni čvor neće poslati nikakvu poruku, i nakon određenog vremena (cca 2 sekunde) pristupni čvor javlja poruku „Autorizacija neuspješna“.

Spomenute poruke su popraćene zvučnim signalom. Nakon uspješne ili neuspješne autorizacije program pristupnog čvora vraća se u početno stanje, čekanja RFID kartice. Dio programa koji je zadužen za dekodiranje potvrde od glavnog čvora i izvještavanju korisnika o stanju autorizacije nalazi se na programskim linijama 861 – 905.

Ovim se zaključuje programski razvoj pristupnog čvora. Nakon opisivanja dobivenog programa koji se nalazi u Privitku 1, pristupni čvor spreman je za samostalan rad i komunikaciju s glavnim čvorom koristeći XBee module. Razvoj se nastavlja s glavnim čvorom koji je u određenim aspektima kompliciraniji a u nekima i jednostavniji od pristupnog čvora.

5.3. Razvoj glavnog čvora

Osnovne funkcije glavnog čvora su:

- Uspostavljanje XBee PAN mreže
- Ostvarivanje komunikacije s pristupnim čvorovima preko XBee mreže
- Vođenje baze podataka na SD memorijskoj kartici
- Uspostavljanje *Ethernet* sučelja prema LAN mreži i potpora softverskoj LabView aplikaciji

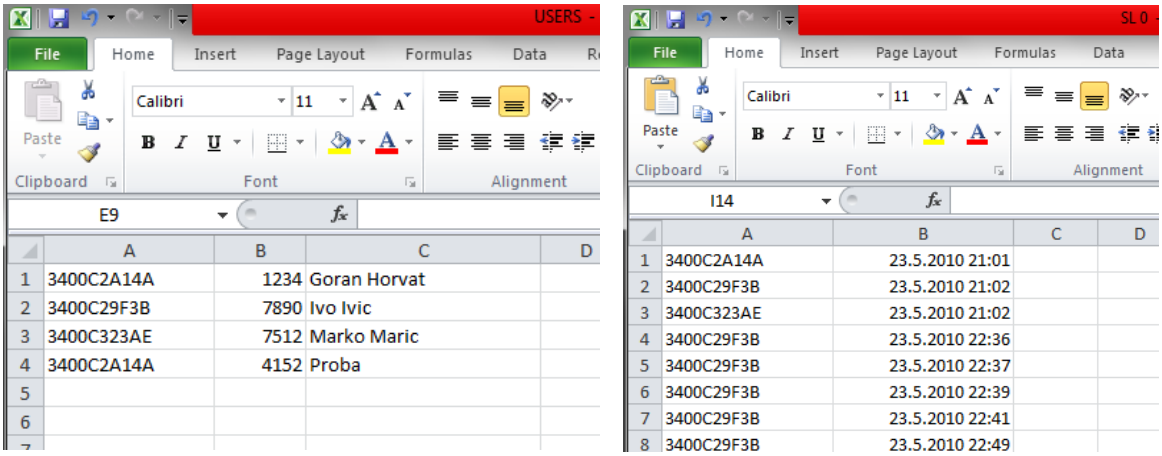
Pošto je u razvoju pristupnog čvora završeno s XBee komunikacijom, gdje su spomenute potvrde od strane glavnog čvora, ovdje se nastavlja razvojem XBee aplikacije, ali prije toga potrebno je objasniti vođene baze podataka na SD memorijskoj kartici. Razvoj sustava prolazi kroz programski kod iz Priloga 2.

5.3.1. Baza podataka na SD memorijskoj kartici

Za metodu skladištenja podataka i vođenja baze podataka odabrana je memorijska SD kartica, zbog jednostavnog SPI sučelja i postojećih bibliotečnih datoteka koje omogućuju izravnu manipulaciju FAT16 (eng. *File Allocation Table*) sustavom.

Način vođenja baze podataka je sljedeći: Baza podataka korisnika s pripadajućim imenima i serijskim brojevima kartica te pristupnim kodovima pohranjena je u datoteku naziva *USERS.txt*. Ova datoteka sadržava sve korisnike u sustavu i kada pristupni čvor zatraži autorizaciju, glavni čvor pretražuje datoteku *USERS.txt* po serijskim brojevima kartica, i uspoređuje pristupni kod za danog korisnika. Ako je korisnik u bazi i pristupni kod je ispravan glavni čvor odgovara pristupnom čvoru porukom *„OK“* i iz datoteke *USERS.txt* ime i prezime korisnika. Time je završen prvi dio autorizacije.

Svakom pristupnom čvoru pridijeljena je posebna datoteka naziva *SL__.txt*, gdje razmak označava broj pristupnog čvora. Kada je prvi dio autorizacije završen glavni čvor dodaje u pripadajuću datoteku *SL__.txt* liniju koji se sastoji od: serijskog broja kartice, datuma i vremena pristupa (koje je dobio od stvarno vremenskog RTC modula). Time je završen postupak autorizacije i korisnik je evidentiran da je pristupio sustavu. Spomenute datoteke su potpuno kompatibilne s programskim alatom Microsoft Excel i primjer baze korisnika i baze pristupa čvora br. 2 prikazane su na Slici 5.9 a) i b).



	A	B	C	D
1	3400C2A14A	1234	Goran Horvat	
2	3400C29F3B	7890	Ivo Ivic	
3	3400C323AE	7512	Marko Maric	
4	3400C2A14A	4152	Proba	
5				
6				

	A	B	C	D
1	3400C2A14A	23.5.2010 21:01		
2	3400C29F3B	23.5.2010 21:02		
3	3400C323AE	23.5.2010 21:02		
4	3400C29F3B	23.5.2010 22:36		
5	3400C29F3B	23.5.2010 22:37		
6	3400C29F3B	23.5.2010 22:39		
7	3400C29F3B	23.5.2010 22:41		
8	3400C29F3B	23.5.2010 22:49		

Sl. 5.9. a) Baza korisnika b) Baza pristupa čvoru br. 0

Manipulacija bazom podataka s mikro upravljačem jako je ograničena. Naime, mikro upravljač nema dovoljno memorije da obradi cijelu bazu podataka, stoga naprednije funkcije poput manipulacije s korisnicima i prijavama nisu moguće. Zbog ograničenosti mikro upravljača, manipulacija bazom podataka prebačena je na osobno računalo i LabView aplikaciju koja je s glavnim čvorom vezana preko Ethernet sučelja. Vođenje same baze može se odraditi i preko Microsoft Excel programa jednostavnim prebacivanjem SD memorijske kartice iz glavnog čvora u računalo.

Kada se govori o programskoj potpori SD kartici na mikro upravljaču, onda je bitno napomenuti da SD kartica koristi SPI sučelje za komunikaciju, stoga je prije svega potrebno inicijalizirati SPI sučelje. Tu se pojavljuje dodatni problem. Zbog činjenice da *Ethernet* modul isto koristi SPI sučelje potrebno je koristiti kontrolu pristupa koristeći CE (eng. C*hip Enable*) sabirnice, kao što se vidi na linijama 403. i 404 te 410 i 411.

Manipulacija bazom podataka koja je pohranjena na SD memorijsko kartici obavlja se sa sljedećim funkcijama: *MMC_Fat_Read()* - čitanje podataka; *MMC_Fat_Write()* – upisivanje podataka; *MMC_Fat_Assign()* – odabir datoteke, i ostalim pomoćnim funkcijama. Čitanje podataka iz datoteke obavlja se pri linijama 267 – 270, gdje se čitaju podaci iz baze korisnika iz razloga provjere postojanja korisnika u sustavu. To je funkcije potprograma koji se nalazi na linijama 253 do 292. Pri pokretanju spomenutog potprograma na pristupnom čvoru dobiveni su podaci o serijskom broju kartice i pristupnom kodu. Dobiveni podaci se provjeravaju preko dva algoritma: prvi je provjera postojanja korisnika u bazi i ispranosti PIN-a (koji se nalazi na

linijama 272 – 278) dok je drugi algoritam zadužen za dohvaćanje imena i prezimena korisnika koji se šalju pristupnom čvoru u *„OK“* potvrdi.

Drugi dio manipulacije bazom podataka obuhvaća dodavanje korisnika i vremena pristupa u bazu pristupa. Spomenuti potprogram nalazi se na linijama 294 – 335. Prvi dio potprograma je formatiranje unosa, gdje se na postojeći serijski broj kartice dodaje vrijeme i datum pristupa, linije 299 – 326. Ostali dio potprograma zadužen je za upisivanje podataka u bazu pristupa na način da se na postojeću datoteku dodaje novi unos, dok prijašnji podaci ostaju nepromijenjeni (eng. *Append Mode*). Ovim se načinom rada rješava problem male memorije mikro upravljača jer nije potrebno učitavanje cijele datoteke radi dodavanja unosa.

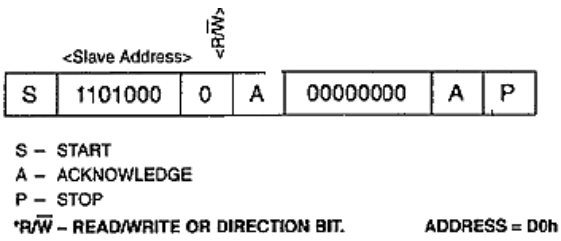
Posljednji dio manipulacije bazom podataka odnosi se na razmjenu podataka između mikro upravljača i *Ethernet* sučelja i nalazi se na linijama 420 – 540. Zadaća ovog dijela programa je paketiziranje podataka iz baze podataka radi prijenosa preko paketne *Ethernet* mreže. Paketiziranje se radi na način da jedan paket sadržava 30 linija baze podataka, što u konačnici daje veličinu od prosječno 900 bajta (što je manje od MTU Etherneta koji iznosi 1500 bajta). Također, funkcija ovog dijela obuhvaća i upisivanje podataka u bazu podataka, točnije unos novih korisnika (linije 493 – 517) i brisanje postojećih korisnika iz baze podataka (linije 521 – 540). Detalji o ovom dijelu programa nalaze se u poglavlju razvoja *Ethernet* sučelja. Ovime je zaključena programska potpora bazi podataka.

5.3.2. Implementacija RTC modula

RTC modul ima za zadaću pružanje točnog datuma i vremena, što je vrlo značajno pri upisivanju korisnika u bazu pristupa. Bez stvarnog vremena jednini parametar s kojim se može raspolagati je radni broj prijave u sustav, što nije iskoristivo za većinu primjena.

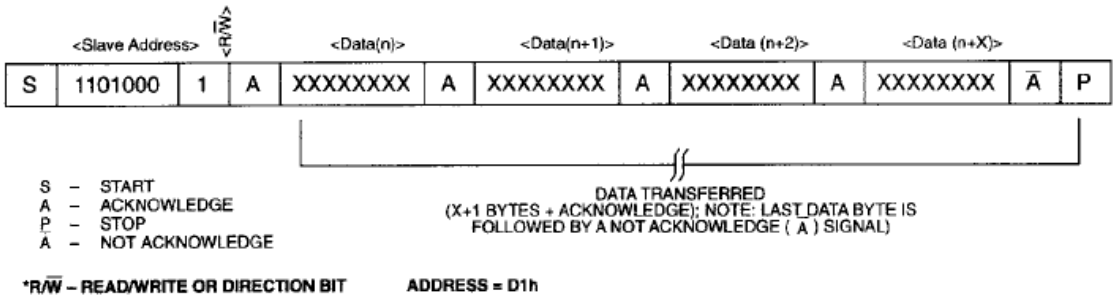
RTC modul koristi I^2C sučelje za komunikaciju s mikro upravljačem. Kako bi komunikacija s RTC modulom bila moguća potrebne su bibliotečne datoteke koje pružaju potporu I^2C sabirnici. U *MikroC* razvojnom alatu te se biblioteke nazivaju TWI biblioteke. Pomoću funkcija kao što su *TWI_Start*, *TWI_Stop*, *TWI_Read* i *TWI_Write* omogućena je komunikacija s RTC modulom i DS1307 integriranim sklopom. Čitanje podataka vrši se na sljedeći način: Kako bi se iščitali podaci iz memorije DS1307 integriranog kruga prvo je potrebno postaviti memorijski pokazivač na početnu adresu 0h. To jer izvedeno na način da se

pomoću funkcije *TWI_Write* upisuje adresa RTC modula i nakon toga početna memorijska adresa. Izgled komunikacije prikazan je na Slici 5.10, a može se vidjeti iz linija 171 – 173.



Sl. 5.10. Postavljanje memorijskog pokazivača na vrijednost 0 [8]

Nakon što se pokazivač nalazi na početku memorije, čitanje podataka može započeti. Princip čitanja podataka prikazan je na Slici 5.11. i može se vidjeti iz linija 174 – 183.



Sl. 5.11. Princip čitanja podataka s DS1307 [8]

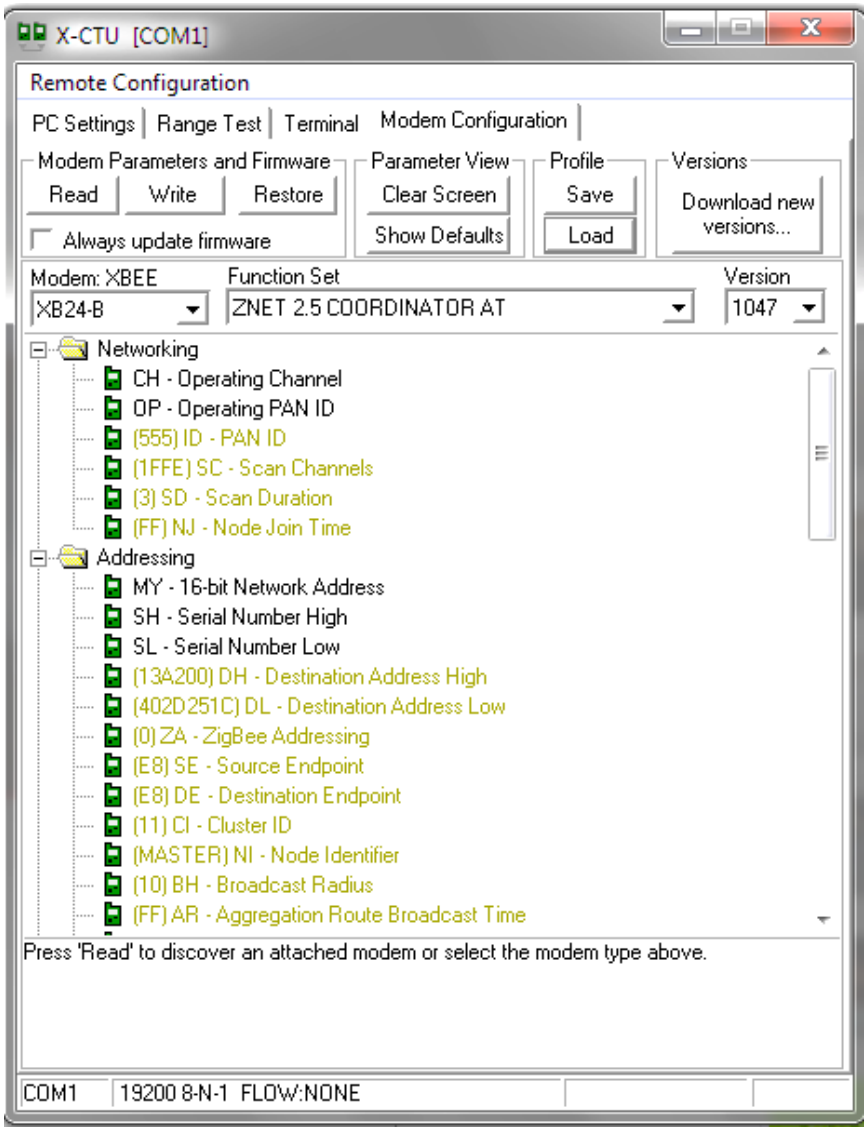
Nakon što mikro upravljač pošalje adresu RTC modula i bit koji označava zahtjev za čitanjem, nadolazeći podaci se iščitavaju bajt po bajt i spremaju u varijable. Zbog samog formata ispisa koji je prikazan na Sl 4.31. potrebno je ponovno formatirati podatke da odgovaraju tekstualnom ASCII prikazu. To je ostvareno u potprogramima *Display_Time* i *Transform_Time* na linijama 187 – 235. Nakon poziva potprograma *Display_Time* na lokalnom 16 x 2 LCD ekranu ispisuje se datum i vrijeme, što označava vrijeme zadnjeg pristupa korisnika.

Nakon razvoja baze podataka i stvarno vremenskog sata odrađen je središnji dio, a to je XBee komunikacija.

5.3.3. XBee modul u služni koordinatora glavnog čvora

U ovom poglavlju slijedi razrada XBee sustava koji vrši funkciju koordinatora u bežičnoj mreži. Kao što je prikazano u Poglavlju 3.3. funkcija XBee koordinatora je osnivanje nove PAN mreže i dopuštanje ostalim čvorovima pristup u mrežu. Zbog funkcije koordinatora XBee modul mora u sebi nositi *firmware* koji omogućuje spomenuti rad. Pošto promjena *firmwarea* nije

moguća od strane samog mikro upravljača potrebno je uz pomoć računala (programski alat X-CTU) postaviti XBee modul. Pri postavkama modula unosi se ime PAN mreže tj. PAN ID i ime čvora (MASTER) tj. NI (eng. *Network Identifier*). Ostale postavke ostaju pri tvorničkim postavkama i po volji ih je moguće postavljat. Izgled postavljenog čvora prikazan je na Slici 5.12.



Sl. 5.12. Postavke XBee koordinatora na glavnom čvoru

Komunikacija između glavnog i pristupnog čvora odvija se na sljedeći način: Glavni čvor se nalazi u stanju čekanja. On konstantno provjerava postoje li nadolazeće poruke na UART sučelju i to u beskonačnoj petlji. Provjera nadolazećih poruka obavlja se na liniji 555. Pošto UART modul ne posjeduje memoriju za ulazne podatke (pohranjuje samo 2 bajta) program mora neprestano provjeravati stanje UART modula (postoji li novi dolazni bajt). Ako postoji, program ga učitava i sprema u vektor, što se vidi iz linije 620. Kada UART modul primi nadolazeći bajt

program provjerava odgovara li nadolazeći bajt zadnjem bajtu iz poslane poruke od strane pristupnog čvora (Znak nove linije, Slika 5.8.). Ako je taj bajt jednak vrijednosti 0Dh tada je prijem poruke završen i slijedi obrada.

Prvi dio obrade obuhvaća provjeru da li je primljena poruka zaista poslana od strane pristupnog čvora, na način da se provjerava zaglavlje poruke koje glasi „SL“ (Slika 5.8.). Ako je potvrđeno da je poruka od pristupnog čvora kreće pretraživanje baze korisnika. Spomenuto pretraživanje obavlja potprogram *M_Open_File_Read* koji je opisan u prošlom poglavlju. Program se poziva na liniji 568 i daljnji tok se račva u ovisnosti o rezultatu potprograma. Ako spomenuti potprogram vrati vrijednost koja označava uspješnu autorizaciju, priprema se poruka za slanje sadržaja „OK <*Ime i Prezime*>\r“. Ako je pristupni kod neispravan priprema se poruka za slanje sadržaja „PIN“. U slučaju da korisnik nije pronađen u bazi podataka, poruka se ne šalje.

Slijedi slanje pripremljenih poruka pristupnom čvoru koji je poslao zahtjev. Slanje podataka od strane glavnog čvora pristupnom čvoru nije jednostavno kao slanje od pristupnog čvora prema glavnom čvoru. Prema tvorničkim postavkama, svi podaci koji se šalju s pristupnog čvor XBee mreže automatski se preusmjeravaju na koordinator (Glavni čvor), tako da se na USART sučelju glavnog čvora izravno dobiju podaci od pristupnog čvora. Kod slanja podataka od strane glavnog čvora prema točno određenom pristupnom čvoru situacija je kompliciranija. Naime, kao što topologija ZigBee mreže govori, koordinator je samo jedan u mreži dok je usmjerivača i krajnjih čvorova više i tu nestaje problem: Koordinator nema informaciju kojem pristupnom čvoru treba poslati potrebne informacije. Stoga je potrebno prije slanja podataka narediti koordinatoru kojem krajnjem čvoru treba dostaviti željene podatke. U transparentnom načinu rada koji se koristi u ovom sustavu to znači da je potreban ulazak u komandni način rada. Ulazak u komandni način rada opisan je u Poglavlju 3.4.2. Nakon ulaska u komandni način rada potrebno je odabrati kojem krajnjem čvoru/usmjerivaču se šalju podaci. To je omogućeno koristeći AT komande, točnije AT komandu ATDN. Spomenuta komanda ima značenje odredišni čvor (eng. *Destination Node*). Pošto XBee mreža koristi identifikatore za imena čvorova, nije potrebno unositi numeričku adresu čvora nego samo naziv čvora tj. NI (eng. *Network Identifier*). Pošto svi pristupni čvorovi imaju isti format NI polja, slanje na točno određeni čvor je olakšano. Format NI polja kod pristupnog čvorova je : „SL××“ gdje simboli ×× predstavljaju redni broj čvora (npr. „SL 2“). Stoga, kada mikro upravljač želi pristupiti čvoru s rednim brojem 2, on šalje sljedeću sekvencu XBee modulu:

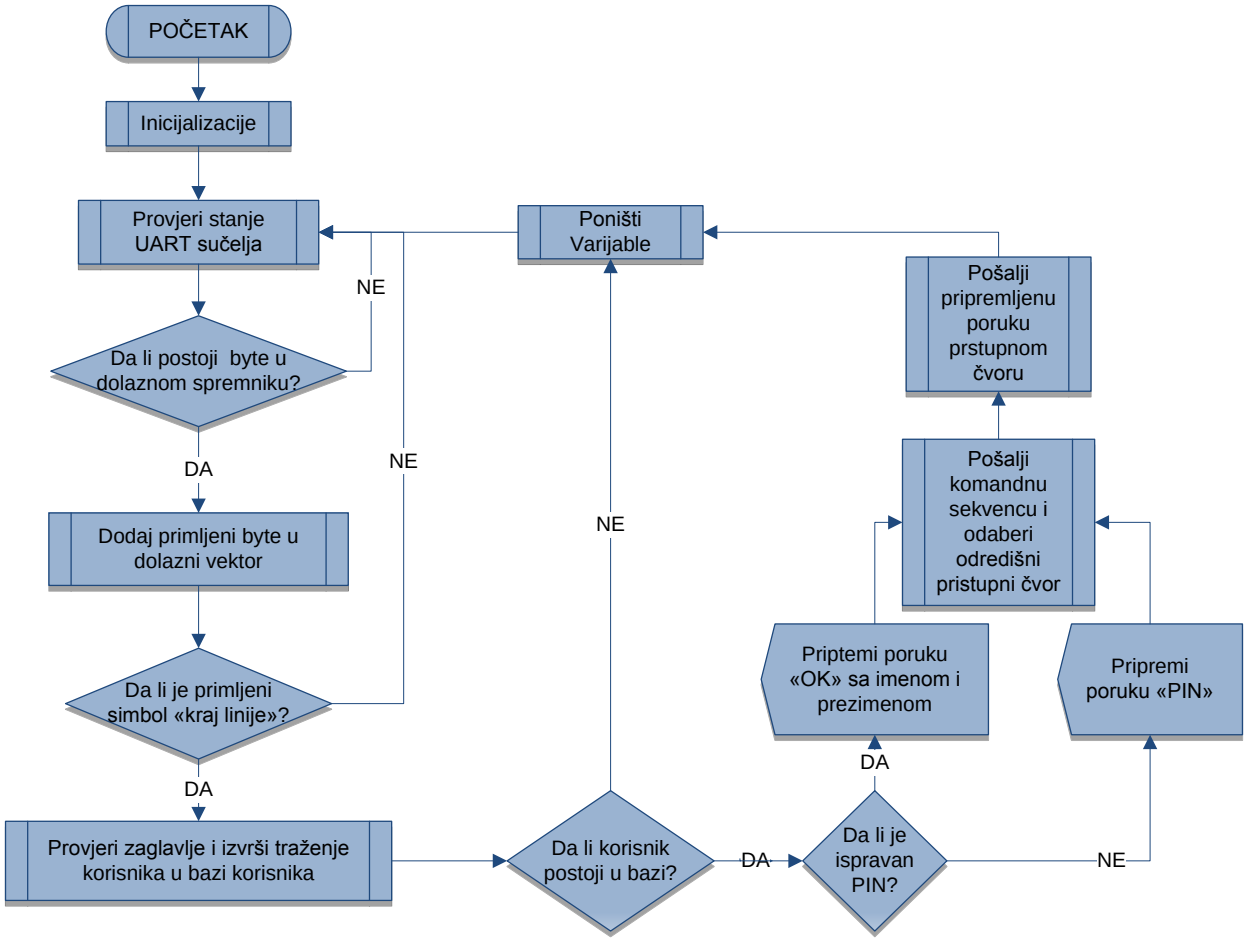
+++OK
ATDN SL 2\r
OK

Gdje su crvenim fontom označeni odgovori od strane XBee modula glavnom čvoru, a crnim poruke poslane od strane glavnog čvora XBee modulu. Nakon poslane sekvence, XBee modul sve nadolazeće podatke šalje čvoru SL 2, gdje su podaci izravno dostupni na UART sučelju mikro upravljača (pristupnog čvora broj 2). Prikazana sekvenca šalje se na linijama 574 – 585.

Nakon što je odabran željeni pristupni čvor, pripremljena poruka se šalje, što se vidi iz linija 588 – 594. Ako je autorizacija uspješna poziva se potprogram dobavljanja točnog vremena *Clock()* (linija 597) i nakon toga vrši se unos vremena i serijskog broja u bazu pristupa dotičnog pristupnog čvora. Unos u bazu pristupa obavlja potprogram *M_Open_File_Append()* koji je opisan u Poglavlju 5.3.2., a poziva se na liniji 602.

Završetkom procesa autorizacije program se vraća u početno stanje, kontinuirano provjeravajući stanje UART modula i stanje dolazećih bajtova. Ovim je zaključen dio autorizacije korisnika i trenutni razvijeni program može samostalno razmjenjivati informacije s pristupnim čvorovima. Na taj način vodi se evidencija pristupa korisnika pristupnim čvorovima. Dijagram toka opisanog programa prikazan je na Slici 5.13.

Preostali dio programa ne utječe na rad autorizacije korisnika i ima svrhu pružanja korisničke podrške kroz *LabView* aplikaciju. Veza između glavnog čvora i *LabView* aplikacije je *Ethernet* sučelje i standardna lokalna mreža. U nastavku opisano je programsko rješenje *Ethernet* sučelja i razmijene informacija s *LabView* aplikacijom.



Sl. 5.13. Dijagram toka XBee komunikacije na glavnom čvoru

5.3.4. Programska potpora *Ethernet* modulu i *LabView* aplikaciji

Ethernet modul opisan je u Poglavlju 4.3.3 korišten je za uspostavu komunikacije između glavnog čvora i LabView korisničke aplikacije. Spomenuti modul koristi SPI sabirnicu za komunikaciju s mikro upravljačem i prethodno izvedene bibliotečne datoteke za pojednostavljenu uporabu samog modula. Pri inicijalizaciji glavnog čvora, uz inicijalizaciju UART sklopova, SPI sabirnice i MMC kartice inicijalizira se i *Ethernet* modul. U prijašnjim opisima inicijalizacije nisu spominjane jer se one sve baziraju na jednoj funkciji tipa *UART_Init()*, koja automatski inicijalizira sve potrebno sklopovlje. Ali kod *Ethernet* modula situacija je kompliciranija, jer je pri inicijalizaciji potrebno definirati i MAC i IP adresu i to sve uskladiti s odgovarajućim DHCP (eng. *Dynamic Host Configuration Protocol*) serverom na mreži. Spomenute adrese u svom početnom obliku dane su na linijama 47 i 48. Inicijalizacija *Ethernet* modula izvodi se na programskim linijama 371 do 389 i to na sljedeći način: Prvo se općenito inicijalizira *Ethernet* modul s početnim adresama. Nakon toga, poziva se DHCP inicijalizacija koja potražuje DHCP server na mreži i preuzima dinamičnu IP adresu. Nakon što

je IP adresa preuzeta ona se prikazuje na lokalnom 16x2 LCD ekranu, kako bi korisnik imao spomenutu informaciju. Dobivena IP adresa unosi se u *LabView* aplikaciju, kako bi aplikacija imala pristup glavnom čvoru. U slučaju da ne želimo koristit DHCP server, tada je potrebno pomoću *jumpera* dovesti logičku jedinicu na ulaznu liniju PB0, i uporaba DHCP adresiranja je onemogućena. Nakon inicijalizacije *Ethernet* modul je spreman za rad.

Princip rada samog modula je sljedeći: bibliotečne datoteke *Ethernet* modula prilagođene su dolasku TCP ili UDP paketa sa mreže, preusmjeravajući tok programa u posebnu prekidnu rutinu, ovisno o vrsti primljenog paketa. U prekidnoj rutini vrši se potrebna obrada podataka i nakon obrade program se vraća u prvobitni tok. Kako bi detekcija dolazećih paketa bila moguća potrebno je kontinuirano provjeravanje *Ethernet* modula, da li je novi paket stigao, isto kao i kod ispitivanja UART-a da li postoji novi nadolazeći bajt. Stoga se funkcija provjeravanja paketa na *Ethernet* modulu smješta uz funkciju provjeravanja UART-a i nalazi se na liniji 547, unutar glavne beskonačne petlje. U glavnoj beskonačnoj petlji su također i dijelovi koda koji se bave pristupom bazi podataka preko *Ethernet* veze, linije 420 – 539, koje su spomenute u Poglavlju 5.3.1. Oni u ovisnosti o postavljanim zastavicama čitaju podatke iz baze podataka, upisuju podatke u bazu podataka ili brišu bazu podataka.

Komunikacija od strane aplikacije osmišljena je na sljedeći način: Aplikacija šalje određeni zahtjev glavnom čvoru koristeći Ethernet sučelje (LAN mrežu) i UDP protokol komunikacije. Razlog odabranog protokola je taj što je UDP bezkonekcijski orijentiran, pa nije potrebno uspostavljanje konekcije i podaci se mogu razmijeniti u bilo kojem trenutku. Oblici zahtjeva koje aplikacija šalje jesu:

Zahtjev za bazom korisnika – aplikacija potražuje bazu korisnika koja se nalazi u datoteci *USERS.txt* na SD memorijskoj kartici. Zahtjev od strane aplikacije se formulira na način da aplikacija šalje UDP paket koji ima sadržaj *„SLxx“,* što predstavlja komandu glavnom čvoru da isporuči bazu korisnika aplikaciji. Nakon što je glavni čvor primio UDP paket pokreće se prekidna rutina koja se nalazi na linijama 94 – 148, gdje se UDP zahtjev obrađuje i šalje odgovarajući odgovor. Ako je sadržaj zahtjeva *„SLxx“* tada glavni čvor odgovara porukom *"OK> SLxx\r\n"* koja označava da je primio zahtjev i spreman je proslijediti bazu korisnika. Tada se postavljaju određene zastavice koje označavaju da se u glavnom programu pripreme potrebni podaci za slanje. To je dio programa na linijama 420 – 487 koji ima funkciju paketiziranja baze podataka u pakete manje od MTU *Etherneta*, te pohranjivanje paketa u memoriju za slanje. Nakon što je aplikacija dobila potvrdu od glavnog čvora da je njezin zahtjev

zaprimljen, ona šalje drugu komandu poruke „NL“ što govori gavnom čvoru da isporuči prije paketizirane podatke aplikaciji, što se može vidjeti na linijama 138 – 140. Aplikacija konstantno šalje *„NL“* komande sve dok ne primi paket koji na kraju sadržaja sadrži poruku *„EOF“,* što označava da je kraj datoteke. Iza poruke *„EOF“* nalaze se 4 znamenke koje označavaju broj poslanih batova od strane glavnog čvora (eng. *Checksum*), kao sigurnosna opcija zbog prirode UDP protokola. Ovaj dio programa može se promatrati na linijama 462 – 480.

Zahtjev za bazom pristupa – ovaj zahtjev izvodi se na identični način kao i zahtjev za bazom podataka, uz neke iznimke. U ovom slučaju zahtjev od strane aplikacije izgleda: „SL__“, gdje se umjesto razmaka upisuje redni broj pristupnog čvora za koji se traži baza pristupa (npr. „SL 2“). Postupak isporuke podataka preko UDP protokola identičan je kao i kod zahtjeva za bazom korisnika, pa nije opisan.

Zahtjev za posljednjom prijavom – aplikacija potražuje mikro upravljač podatak na kojem se čvoru izvršila zadnja prijavu u sustav. Izgled zahtjeva koji je poslan od strane aplikacije ima oblik: „SL**“ i nalazi se u poslanom UDP paketu. Kada mikro upravljač primi UDP paket poziva se prekidna rutina koja ispituje sadržaj paketa. Ako je sadržaj paketa „SL**“ mikro upravljač šalje paket u kojemu su podaci o zadnjoj prijavi u sustav. Format paketa prikazan je na Slici 5.6. a obrada se odvija na linijama 121 – 136. Ne temelju ove informacije moguće je dodati novog korisnika u bazu korisnika, čemu služi sljedeći zahtjev.

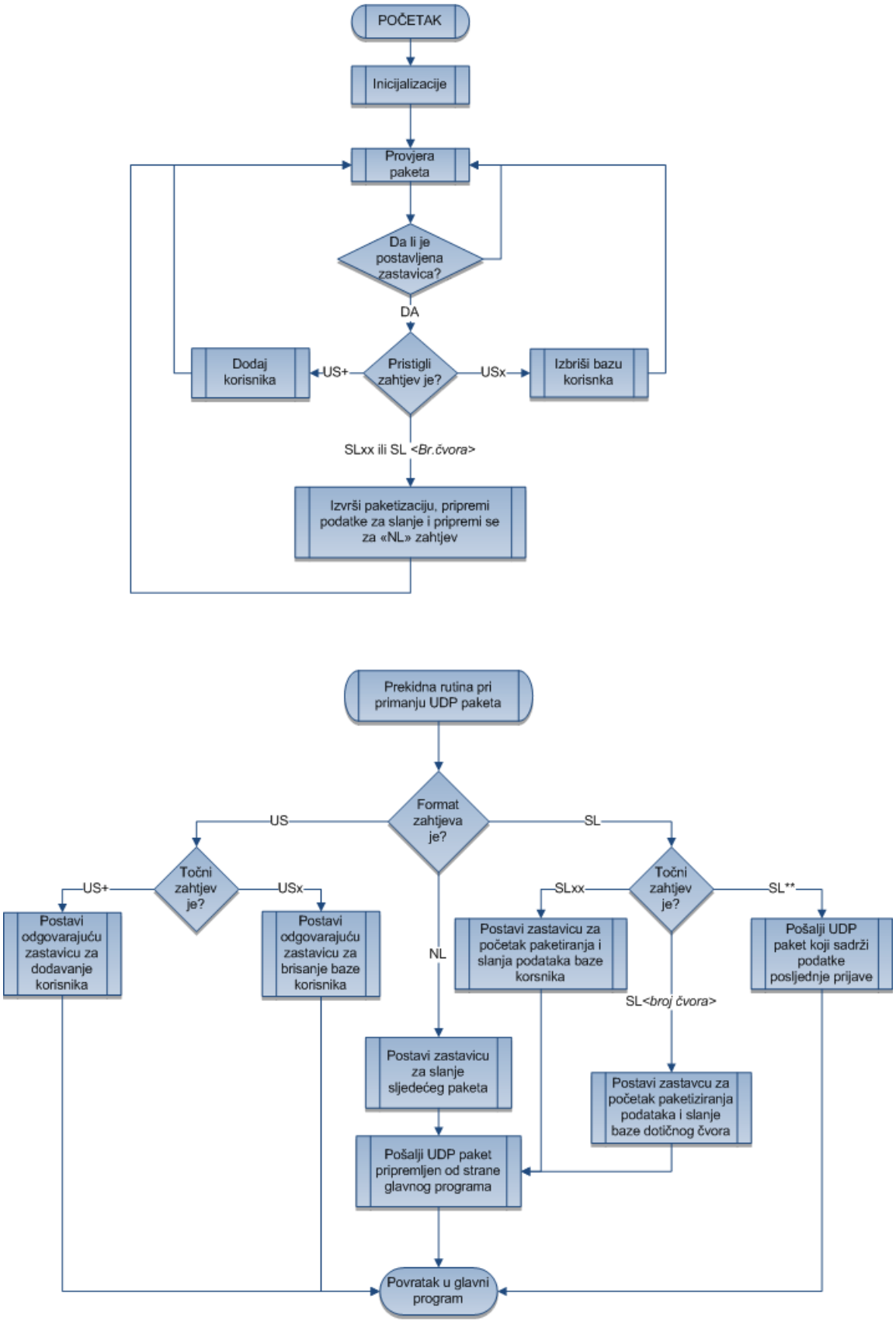
Zahtjev za unosom novog korisnika – kada je aplikacija primila podatke o novom korisniku preko Zahtjeva za posljednjom prijavom, ona šalje paket sadržaja:

„US+ <Serijski br. Kartice> <PIN> <Ime i prezime>\r“

Mikro upravljač prima zahtjev obrađuje ga i postavlja zastavicu dodavanja novog korisnika, linije 109 – 117. Tada se u glavnom programu izvodi dodavanje samog korisnika, što se vidi iz linija 493 do 517. Otvara se datoteka *USERS.txt* i pomoću *append* načina rada dodaje se novi unos korisnika sa serijskim brojem kartice PIN-om te imenom i prezimenom.

Zahtjev za brisanjem baze korisnika – zbog ograničene memorije mikro upravljača nije moguća manipulacija s korisnicima, nego se dopušta samo brisanje cijele baze korisnika. To se postiže slanjem komande „USx“ od strane aplikacije glavnom čvoru. Brisanje baze odvija se u glavnom programu na linijama 527 – 539.

Dijagram toka *Ethernet* programske podrške prikazan je na Slici 5.14.



Sl. 5.14. Dijagram toka *Ethernet* dijela programa

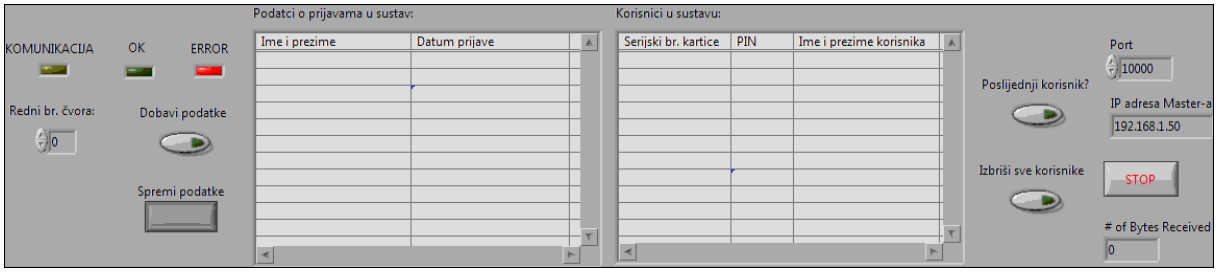
Ovim je upotpunjen razvoj glavnog čvora i pristupnih čvorova. Programi koji se nalaze u Prilozima 1. i 2. prevedeni su u strojni jezik i upisani u mikro upravljače, čime je Sustav bežične RFID autorizacije korisnika postao potpuno funkcionalan! Sljedeće poglavlje opisuje razvoj korisničke aplikacije za pristup bazi podataka u programskom alatu *LabView*.

5.4. Razvoj *LabView* korisničke aplikacije

LabView (eng. *Laboratory Virtual Instrumentation Engineering Workbench*) je platforma i razvojno okruženje za vizualni programski jezik američke tvrtke „National Instruments". Grafički jezik naziva se *„G"* i on predstavlja vrstu programskog jezika koja je bazirana na programskom toku. Izvođenje programa je određeno strukturom grafičkih blokova na koje se spajaju različiti funkcijski čvorovi, metodom vodiča. Ti vodiči prenose podatke i svaki čvor može izvršiti svoju funkciju tek kada svi podaci postanu dostupni. Zbog ove činjenice moguće je postojanje simultanih radnji što programski jezik *„G"* podržava, jer ima svojstvo paralelnog izvođenja.

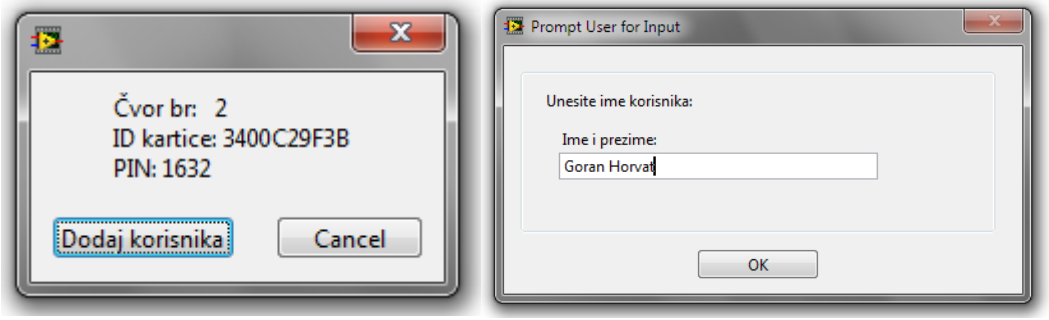
Programiranje se sastoji u stvaranju programskog koga u *„G"* programskom jeziku i stvaranja grafičkog sučelja za interakciju s korisnikom. *LabView* platforma je jako popularna kod inženjera jer pruža jednostavno grafičko sučelje za programiranje kompleksnih radnji, a ujedno prezentira korisniku jednostavno grafičko sučelje (eng. *Graphic User Interface*) za laganu uporabu.

U sustavu Bežične RFID autorizacije korisnika osmišljena je aplikacija koja koristeći *Ethernet* vezu (točnije UDP protokol) komunicira s bazom podataka pohranjenom na glavnom čvoru, uz prije definirani protokol komunikacije. Aplikacija ima mogućnost prikaza korisnika u sustavu, prikaza pristupa određenom čvoru, dodavanja korisnika u bazu i brisanje baze korisnika. Početak osmišljavanja aplikacija bazirao se na izradi korisničkog sučelja, imajući u obzir funkcionalnost. Izgled korisničkog sučelja prikazan je na Slici 5.15.



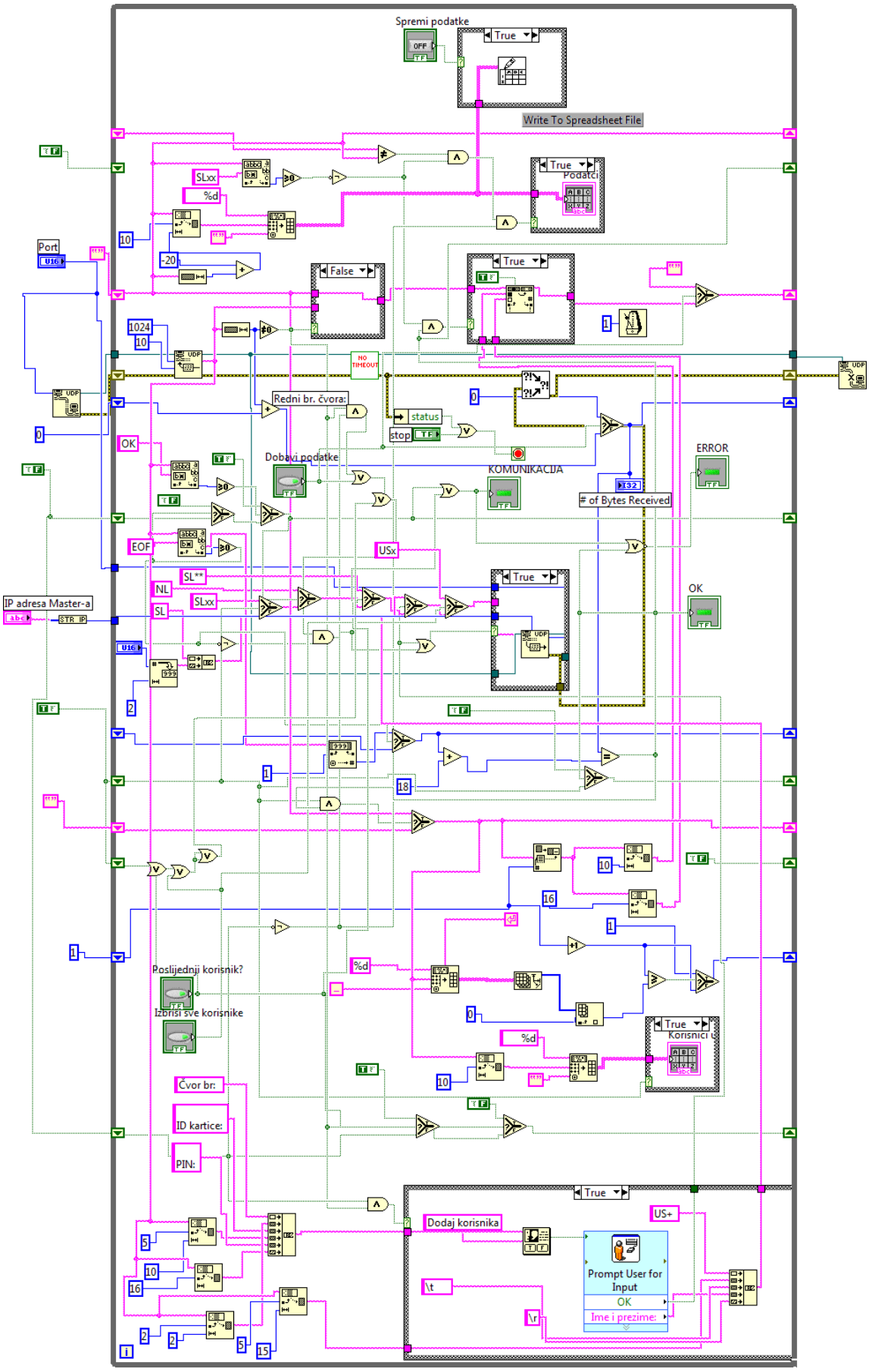
Sl. 5.15. Izgled korisničkog sučelja *LabView* aplikacije

Na Slici 5.13. prikazan je glavni prozor koji se sastoji od dva manja prozora (tabličnog prikaza), u kojima su prikazani korisnici u sustavu i popis prijava za odabrani pristupni čvor. Također prikazani su statusni indikatori koji govore o toku komunikacije: jesu li podaci koji su primljeni ispravni (pomoću *Checksum* svojstva koje mikro upravljač pruža) i postoji li greška u komunikaciji. Ispod spomenutih indikatora postoji odabir rednog broja pristupnog čvora koji nakon što odaberemo aktiviramo opciju *Dobavi podatke*. U prostor ekrana *Podatci o prijavama u sustav* pojavit će se imena korisnika i datum prijave, za odabrani čvor. *Spremi podatke* opcija ispisuje podatke o prijavama u datoteku prilagođenu *Microsoft Excel* programskom alatu. Opcija *Izbriši sve korisnike* radi upravo to, dok opcija *Posljednji korisnik* otvara prozor na kojemu su ispisani podaci posljednje prijave u sustav, kao što su: Serijski broj kartice (ID) i PIN, i tu se nudi mogućnost spremanja korisnika u bazu. Izgled prozora prikazan je na Slici 5.16. Ako se odabere opcija *Dodaj korisnika*, otvara se novi prozor gdje aplikacija traži unos imena i prezimena. Potvrdom na *OK* korisnik je dodan u sustav. U desni dio ekrana upisuje se IP adresa glavnog čvora i dodijeljeni *port*, kako bi komunikacija s glavnim čvorom bila moguća.



Sl. 5.16. Izgled prozora Posljednji korisnik i prozora za unos Imena i Prezimena

Sa strane funkcionalnosti aplikacije treba napomenuti sljedeće: Pri pokretanju aplikacije šalje se zahtjev za bazom korisnika, „*SLxx*“. Aplikacija određuje slanje zahtjeva i slanje „*NL*“ komandi sve dok se na primi cijela baza. Primitkom kraja paketa oznake „*EOF*“ i *checksum* broja prijenos je završen. Nakon što je primljena baza korisnika daljnje zahtjeve aplikacija odrađuje ovisno o korisnikovim komandama. Detaljni opis same komunikacije s glavnim čvorom otežan je zbog grafičke prirode samog programskog jezika „*G*“ ali komunikacija koja obuhvaća i stranu aplikacije opisana je u Poglavlju 5.4.3. Izgled aplikacije u „*G kodu*“ prikazan je na Slici 5.17. a princip rada moguće je pogledati u samom *LabView* okruženju.

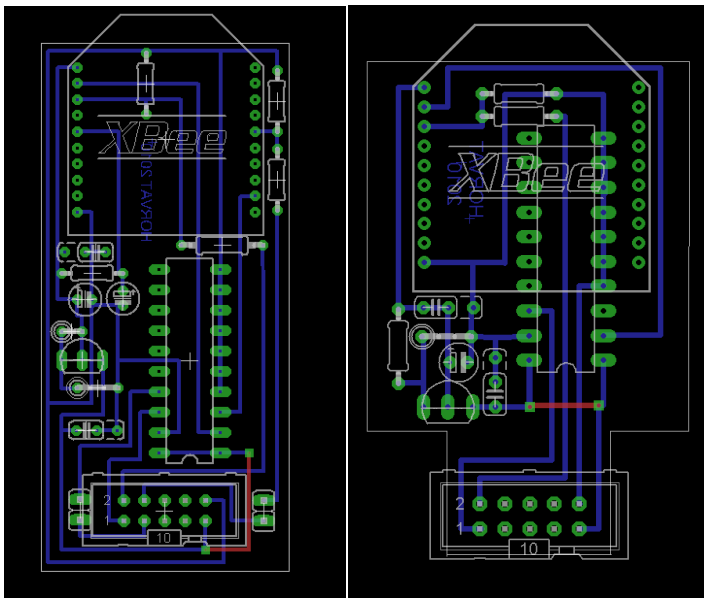


Sl. 5.17. Izgled „G“ koda za aplikaciju Bežične RFID autorizacije korisnika

5.5. Praktična izrada i testiranje sustava

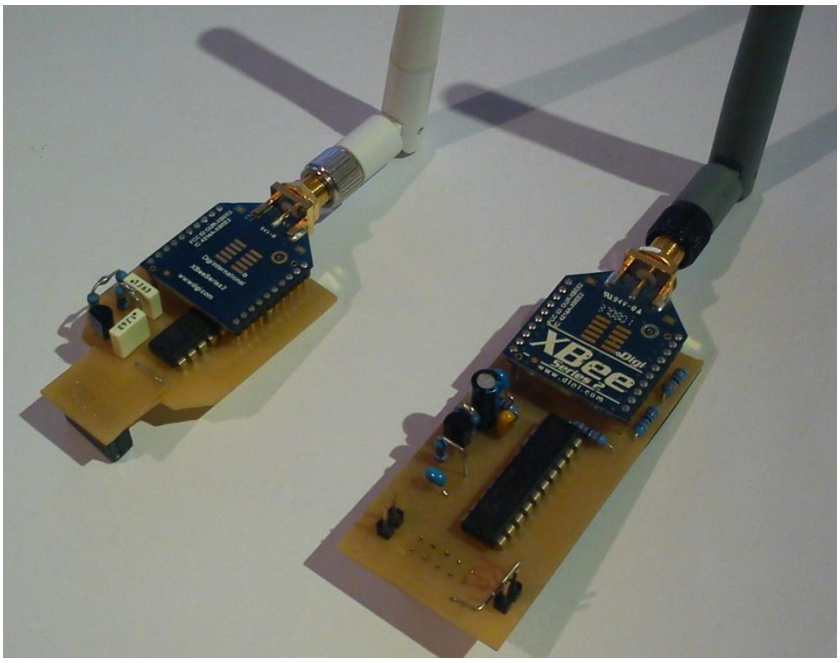
U poglavljima četiri i pet ovog rada opisani su svi hardverski i softverski zahtjevi koji su potrebni kako bi sustav Bežične RFID autorizacije korisnika bio funkcionalan. Nakon što su svi zahtjevi ispunjeni moguće je i praktično ostvariti spomenuti sustav.

Prvi korak u ostvarivanju sustava je fizičko spajanje razvojnih sustava i svih dodatnih modula. Ovdje problem nastaje u činjenici da neki od opisanih modula nisu praktično izvedeni nego su samo teorijski razloženi. Jedan od tih modula je i naponski translator razine koji služi kao most između razvojnog sustava i XBee modula (jedan za Glavni čvor, jedan za pristupni čvor). Zbog činjenice da je shema spajanja osmišljena u Poglavlju 4.2.2., izrada i dizajn tiskane pločice nije predstavljao veliki problem. Izgled tiskane pločice za oba čvora prikazan je na Slici 5.18.



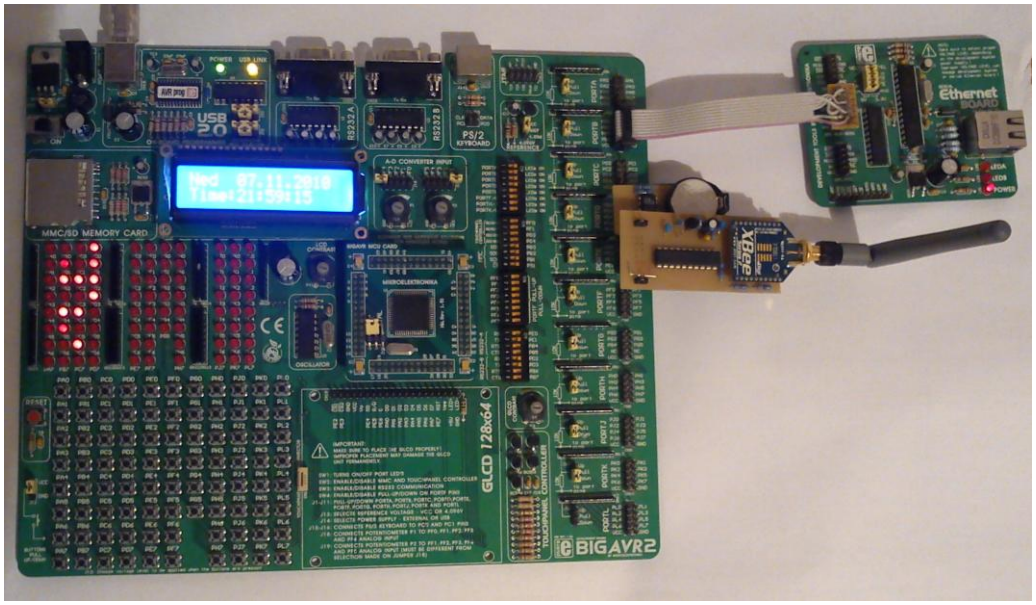
Sl. 5.18. Izgled tiskane pločice naponskog translatora

Dobivene tiskane pločice izrađene su koristeći *photo* postupak prijenosa vodova na bakrenu tiskanu pločicu. Nakon jetkanja i bušenja pločice, odgovarajući elementi su zalemljeni i naponski translator postao je funkcionalan. Izgled pločica prikazan je na Slici 5.19.



Sl. 5.19. Izrađeni moduli naponskih translatora s XBee modulima

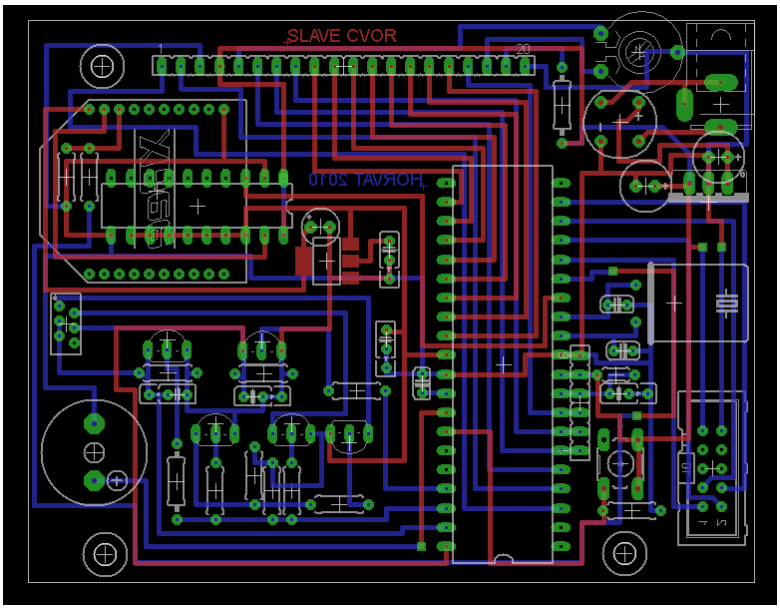
Dobiveni moduli koriste se s razvojnim sustavima srpske tvrtke „mikroElektronika“, čiji se razvojni sustavi koriste u ovom radu. Nakon izrađenih modula naponskih translatora moguće je fizički spojiti Sustav bežične RFID autorizacije korisnika. Izgled glavnog čvora prikazan je na Slici 5.20. Na isti način spojen je i pristupni čvor.



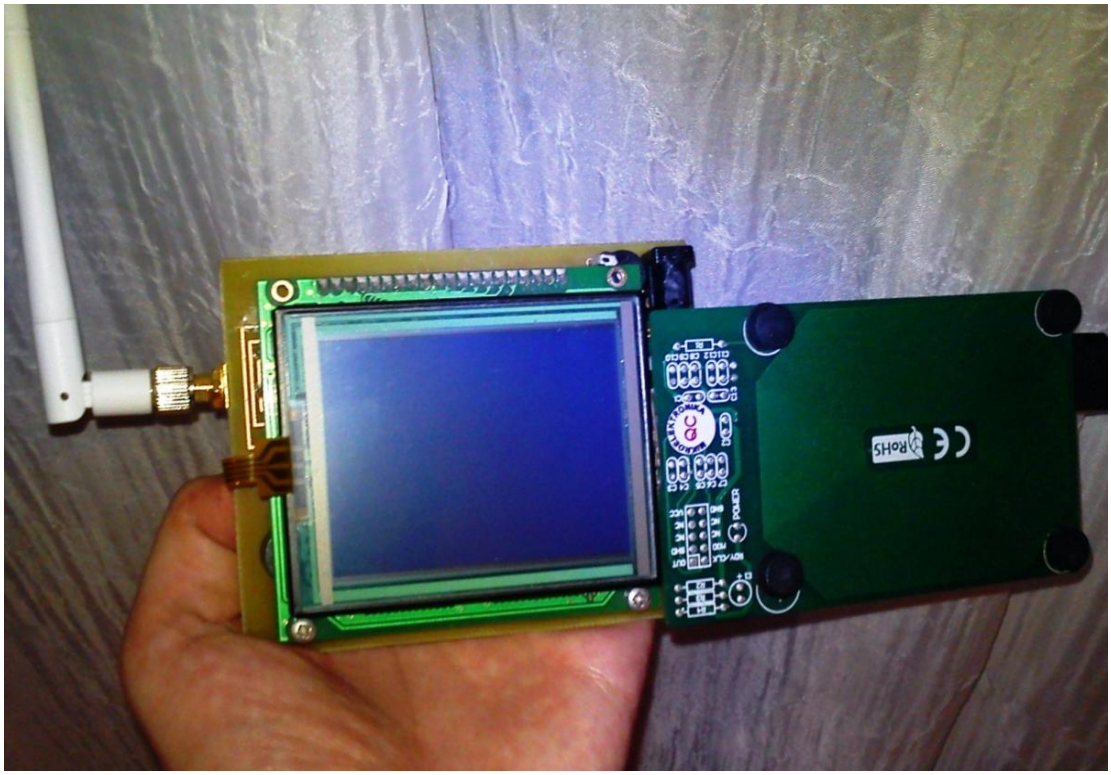
Sl. 5.20. Izgled glavnog čvora

Kada se pogleda da je cilj ovog rada osigurati bežičnu vezu pristupnog čvora s glavnim čvorom, stavljajući naglasak na malu potrošnju energije pristupnog čvora i praktičnost istog, onda ovakva izvedba (iako i u okvirima ovog rada) postaje nepraktična. Stoga, objedinjujući sve

vanjske module koji su korišteni u izvedbi pristupnog čvora i spajajući spomenute u jednu cjelinu, pristupni čvor je poprimio oblik kompaktnog i praktičnog uređaja. Izgled tiskane pločice pristupnog čvora i sam izgled prikazani su na Slikama 21. i 22.



Sl. 5.20. Izgled tiskane pločice pristupnog čvora



Sl. 6.21. Izgled izrađenog uređaja pristupnog čvora

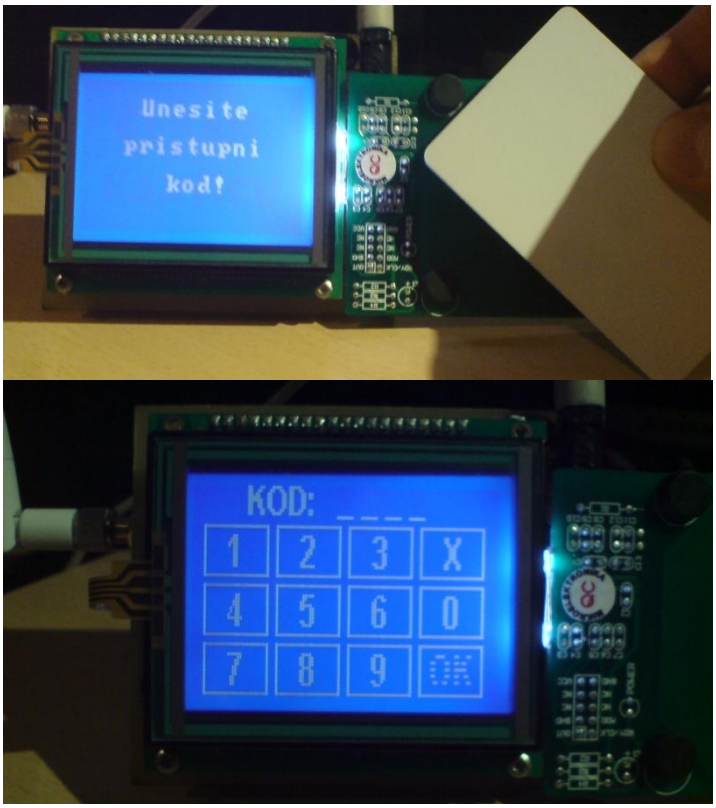
5.5.1. Ispitivanje sustava

Nakon što je sustav osposobljen potrebno je sustav i ispitati. Ispitivanje je obavljeno koristeći malu bazu korisnika koja je unesena prije testiranja.

Testiranje je započeto sa bezkontaktnom karticom prije upisanom u bazu pod imenom i prezimenom „Goran Horvat“. Izgled pristupnog čvora pije prislanjanja kartice prikazan je na Slici 5.22.

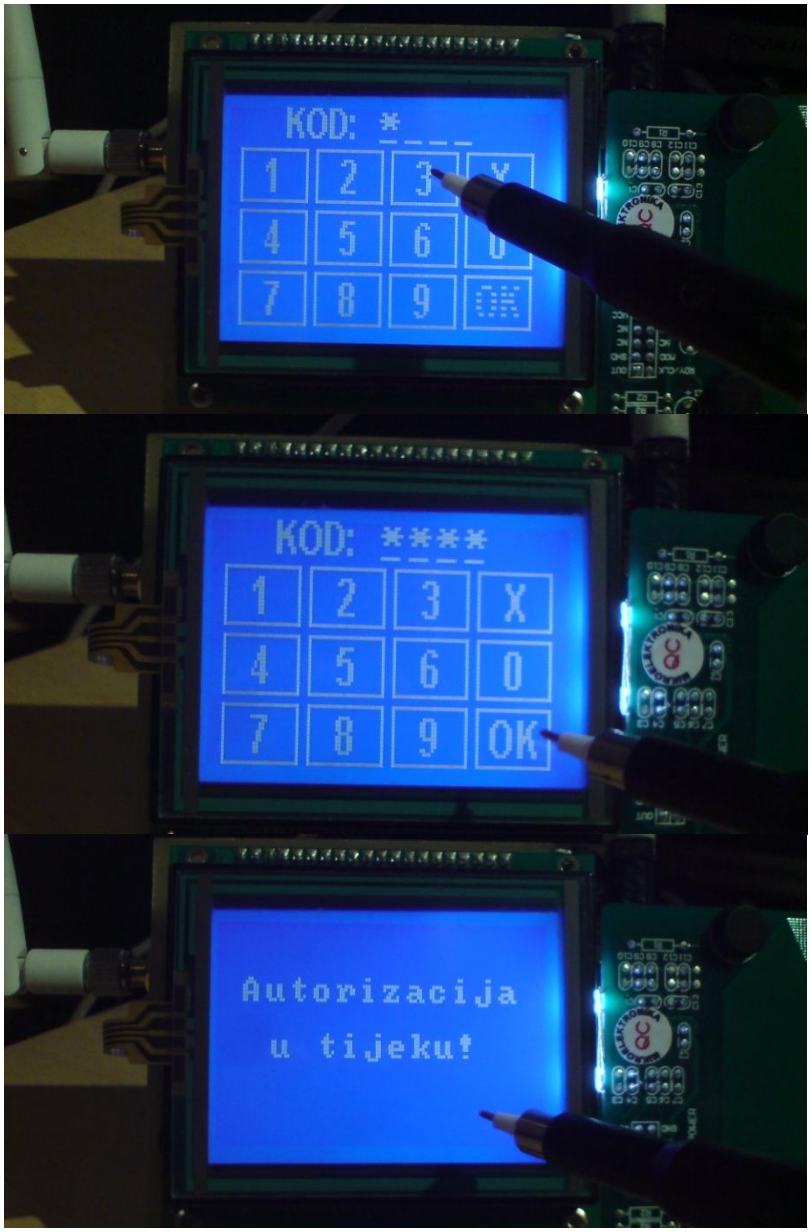
Sl. 5.22. Početno stanje pristupnog čvora

Nakon što je čitač očitao bezkontaktnu karticu (uz zvučni signal) korisnik je zatražen za unos osobnog identifikacijskog koda.



Sl. 5.23. Unos osobnog identifikacijskog koda

Unos koda se vrši koristeći zaslon osjetljiv na dodir. Kada je cijeli kod unesen omogućena je opcija *„OK“*, kojom korisnik potvrđuje spoje podatke. Nakon toga podaci se šalju glavnom čvoru i autorizacija je u tijeku.

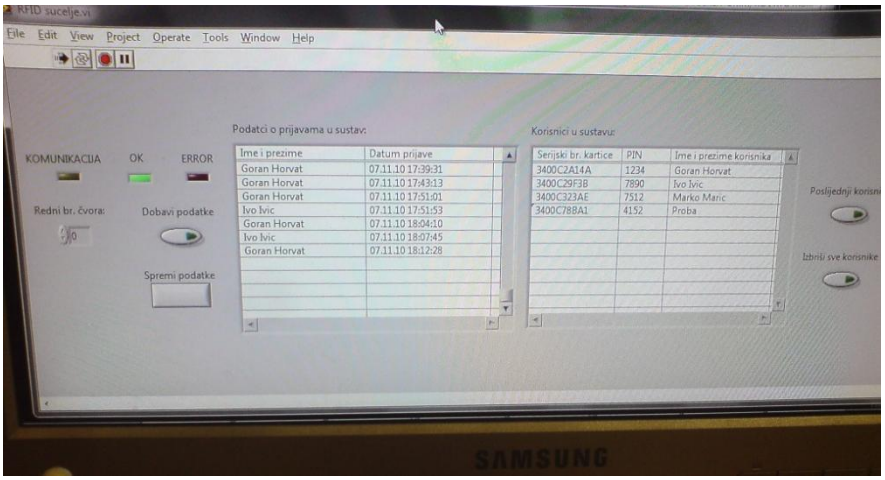


Sl. 5.24. Unos PIN-a i postupak autorizacija

Postupak autorizacije može imati nekoliko krajnjih rezultata: Autorizacija može biti uspješna, osobni identifikacijski kod može biti pogrešan i autorizacija može biti neuspješna. Ako je autorizacija uspješna izgled pristupnog čvora prikazan je na Slici 5.25. U protivnom, sukladna poruka biti će ispisana.

Sl. 5.25. Izgled čvora pri uspješnoj autorizaciji

Uspješnu autorizaciju slijedi i upis korisnika u bazu pristupa određenog pristupnog čvora. Kako bi se došlo u uvid evidencije pristupa potrebno je glavni čvor povezati s lokalnom mrežom i pristupiti aplikaciji za manipulaciju bazom podataka. Izgled aplikacije, kada je spojena s glavnim čvorom prikazana je na Slici 5.26.



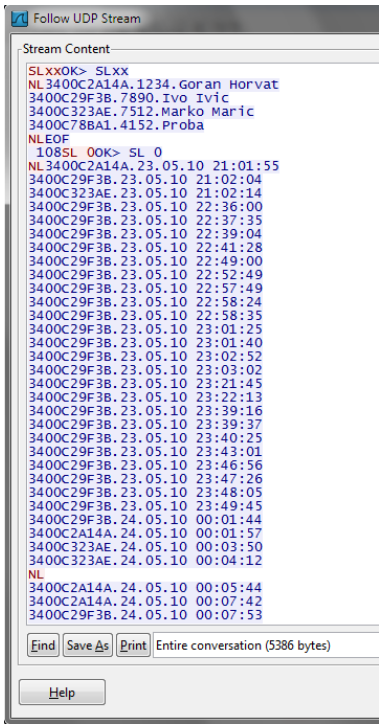
Sl. 5.26. Izgled aplikacije na osobnom računalu

Iz prijava u sustav vidi se da je zadnji unos korisnik „Goran Horvat“ čija se bezkontaktna kartica i koristila pri ispitivanju prijave u sustav. Ako se korisnik prijavi u sustav s karticom koja nije u sustavu, ili s neispravnim PINom, taj korisnik se neće uvesti u bazu pristupa.

Ako pogledamo bežičnu komunikaciju između pristupnog i glavnog čvora onda se opaža nekoliko stvari: Bežična komunikacija će biti moguća samo ako je uspostavljana ZigBee PAN mreža. Ako se glavni čvor uključi nakon pristupnih čvorova ZigBee mreža neće biti uspostavljena na vrijeme i komunikacija neće biti moguća. Ispravan postupak je uključivanje glavnog čvora prvo, a nakon određenog vremena i ostale pristupne čvorove. Nakon što su svi

čvorovi uključeni potrebno je određeno vrijeme da se ZigBee mreža formira (ovisno o broju čvorova, nekoliko sekundi do nekoliko desetaka sekundi). Ako se u vremenu kada PAN mreža nije formirana pristupi komunikaciji, komunikacija će biti neuspješna.

Ispitivanje *Ethernet* sučelja izvršilo se koristeći analizator mrežnog programa koji je detektirao UDP pakete poslane od strane glavnog čvora i *LabView* aplikacije. Primjer komunikacije kada aplikacija potražuje bazu korisnika i bazu pristupa za pristupni čvor broj 0 prikazana je na Slici 5.27.



Sl. 5.27. Tok podataka UDP paketa

Iz priloženog se vidi da je komunikacija ostvarena na način na koji je i zamišljena, koristeći definirane komande i komunikaciju tipa pitanje odgovor. Problemi pri komunikaciji su se događali kod postavki mrežnog usmjerivača da filtrira promet (eng. *Firewall*) što je ispravljeno koristeći nove postavke mrežnog usmjerivača.

Ovim je zaključen i praktični dio ovog rada. U međuvremenu, sustav je dodatno ispitivan i nisu se pokazali ikakvi drugi nedostatci ili problemi koji nisu navedeni u ovom diplomskom radu.

6. ZAKLJUČAK

Prijava i autorizacija korisnika u današnjim sustavima pronalazi primjenu u mnogim područjima, od evidencije zaposlenih pa sve do pristupa bankovnim računima. Koristeći tehniku bezkontaktne autorizacije tj. RFID tehnologije s unosom osobnog koda, povećava se sigurnost uz zadržavanje komocije korisnika. Tema ovog rada bila je osmisliti sustav autorizacije korisnika koristeći pristupne čvorove koji su bežično vezani s glavnim čvorom, na kojemu se vodi baza podataka, koristeći ZigBee protokol i *XBee* komunikacijske module.

Počevši s teorijskim osnovama RFID tehnologije i ZigBee protokola osmišljen je sustav Bežične RFID autorizacije korisnika. Uz korištenje *Atmelovih* mikro upravljača iz serije AVR, razvojnih sustava srpske tvrtke „mikroElektronika“ i dodatnih modula bilo je moguće ostvariti zadani cilj i ostvariti sustav bežične RFID autorizacije korisnika.

Počevši s razradom RFID čitača uspješno je dekodiran sirovi slijed podataka koji je rezultirao iz perifernog RFID modula. Svrha spomenutog modula bila je demodulacija moduliranog povratnog radijskog signala (eng. *Backscatter Modulation*), točnije rečeno demodulacija ASK diskretne modulacije. Dekodiranja binarnih podataka (*Manchester* kodiranje*)*, paritetna provjera redundancije i ostali poslovi odrađeni su na softverskoj razini u mikro upravljaču. Nastavak je slijedio programiranjem grafičkog sučelja LCD ekrana osjetljivog na dodir, stvaranjem grafičkog korisničkog sučelja i implementiranjem zaslona osjetljivog na dodir (eng. *Touch Screen*). Programiranje funkcije LCD ekrana urađeno je uz pomoć bibliotečnih datoteka, koje su mnogo pojednostavile uporabu spomenutog LCD ekrana. Nakon funkcionalnog dijela pristupnog čvora, nastavak je slijedio ostvarivanjem bežične ZigBee komunikacije između glavnog čvora i pristupnog čvora. Za ulogu koordinatora odabran je glavni čvor, dok su pristupni čvorovi vršili funkciju krajnjih čvorova. Nakon uspostavljene PAN ZigBee mreže na strani glavnog čvora kreirana je baza podataka čije je skladištenje odabrano na SD memorijskoj kartici. Razvijen je sustav provjeravanja korisnika u bazi podataka i izvještavanja o uspješnosti autorizacije. Dodan je *Ethernet* modul koji je povezan s bazom podataka na glavnom čvoru i s korisničkom aplikacijom (koja je osmišljena koristeći razvojni sustav *Labview*).

Nakon povezivanja cijelog sustava izvršeno je testiranje i ispravljeni su manji nedostatci. Pokazalo se da je komunikacija između glavnog čvora i pristupnog čvora bila uspješna u velikoj većini slučajeva. U slučajevima u kojima nije bila uspješna problem se pripisuje neuspješnoj

koordinaciji spomenutih XBee modula. Naime, pri pokretanju sustava prvo je potrebno pokrenuti koordinator, tj. Glavni čvor i pričekati određeni vremenski period da se uspostavi ZigBee PAN mreža. Nakon toga pokreću se pristupni čvorovi kojima je također potrebno određeno vrijeme za sinkronizaciju s mrežom. Nakon što su se spomenuti moduli sinkronizirali na ZigBee PAN mrežu komunikacija nije ničim spriječena.

Sa pogleda vođenja baze podataka, pokazalo se da je sustav unosa korisnika pouzdan i niti u jednom slučaju nije se dogodila pogreška pri unosu, na način da je korisnik unesen u bazu pristupa a da autorizacija nije bila uspješna. Također, pokazalo se da je pouzdanost RTC stvarno vremenskog modula visoka, jer u tjednima rada nije bilo većeg odstupanja od pravog vremena. Jedini nedostatak koji nije riješen sa strane mikro upravljača je manipuliranje s bazom korisnika, točnije, pojedinačno brisanje korisnika, zbog malog kapaciteta radne memorije mikro upravljača. Ovaj problem može se riješiti obradom baze pri korisničkoj aplikaciji na osobnom računalu.

Komunikacija između korisničke aplikacije i glavnog čvora u većini slučajeva bila je uspješna. Problemi su se pojavljivali zbog *firewall*-a usmjerivača u LAN mreži i zbog filtriranja prometa po *portovima*. Kada su spomenute postavke podešene na usmjerivaču problema nije bilo. Pokazalo se da je *Checksum* provjera veoma pouzdana i na gubitak paketa javlja pogrešku, te stoga ne prikazuje oštećene podatke. Gubitak paketa pri ispitivanju bio je zanemariv. Dodatni problem se znao događati pri uporabi DHCP metode za određivanje IP adrese, ali samo u malom postotku slučajeva. Zbog toga je dodan hardverski prekidač kojim je moguće deaktivirati DHCP, kada *Ethernet* modul poprima početnu adresu (192.168.1.50).

Spomenutim je zaključeno ispitivanje sustava Bežične RFID autorizacije korisnika. Globalno se može reći da je sustav zadovoljio tražene uvjete i zahtjeve. Problemi koji su se događali mogu poslužiti za daljnje istraživanje i usavršavanje ovog sustava. U pogledu daljnjeg poboljšanja, moguće je dodavanje više mogućnosti manipulacija bazom korisnika i bazom pristupa, jer softverski nije riješen slučaj uređivanja ili brisanja baza pristup. Iako SD memorijska kartica predstavlja određenu prednost za vođenje baze podataka, veća prednost bi bila korištenje postojećeg SQL (eng. *Structured Queried Language*) servera koji je dostupan na LAN mreži preko *Ethernet* sučelja. Time bi se jako olakšala manipulacija korisnicima i bazama podataka, jer je SQL standardizirani način vođenja baze podataka i nije potrebna posebna aplikacija za pristup.

Sumirajući rezultate ovog rada može se reći da je u ovom radu veoma uspješno realiziran sustav autorizacije korisnika koristeći RFID tehnologiju i bežični prijenos informacija. Poseban

naglasak je dan na Bežičnom sustavu čime se eliminiraju problemi mrežne infrastrukture, pogotovo u primjenama gdje je udaljenost relativno velika.

Mnogo je rada potrebno ne bi li se ovaj sustav komercijalizirao no ovo je dobar početak. Uz nove tehnologije i nova saznanja ovaj sustav će zasigurno potpomoći razvoju nekog sličnog sustava, ako ne po dobivenim rezultatima onda barem po ideji!

LITERATURA

[1] V. Daniel, Albert Puglia, Mike Puglia, RFID A Guide To Radio Frequency Identification, Wiley, 2007.

[2] Jeremy Landt, The history of RFID, IEEE POTENTIALS, 2005.

[3] Christoph Jechlitschek, A Survey Paper on Radio Frequency IDentification (RFID) Trends, 2006.

[4] Pete Sorrells, Passive RFID Basics, Microchip Technology Inc., 1998.

[5] Digi International Inc., XBee™ ZNet 2.5/XBee-PRO™ ZNet 2.5 OEM RF Modules, 2008.

[6] Atmel Datasheet: ATmega32, 2009.

[7] Atmel Datasheet: ATmega128, 2009.

[8] Dallas Semiconductor, DS1307 64 x 8 Serial Real-Time Clock datasheet

[9] mikroElektronika, EasyAVR5A User Manual, 2008.

[10] mikroElektronika, BigAVR2 User Manual, 2008.

[11] Rick Downs, Using resistive touch screens for human/machine interface Texas Instruments Incorporated, 2005.

[12] Microchip Technology Inc., ENC28J60 Datasheet, 2004.

[13] SparkFun Electronics , XBee Serial Explorer Datasheet, 2009.

[14] mikroElektronika, RFID module datasheet, 2009.

[15] mikroElektronika, RTC module datasheet, 2009.

[16] EM Microelectronics, EM4100 Read Only Contactless Identification Device, 2002.

[17] Fread Eady, Implementing 802.11 with Microcontrollers, Elsevier, 2005.

[18] K. Sohraby, D. Minoli, T. Zanati, Wireless sensor networks, Wiley, 2007.

[19] mikroElektronika, Introduction to MikroC PRO for AVR, 2009.

SAŽETAK

U ovom radu prikazan je razvoj jednog sustava autorizacije korisnika, pomoću RFID tehnologije i bežične komunikacije. U teorijskom dijelu opisana je RFID tehnologija za autorizaciju i ZigBee tehnologija bežične komunikacije. Nadalje, osmišljen je sustav bežične RFID autorizacije koristeći glavni čvor i pristupne čvorove. Vođenje baze podataka ostvareno je na glavnom čvora, koji komunicira s pristupnim čvorovima na kojima se nalaze RFID čitači i numerički unos PIN-a. Glavni čvor komunicira i s *LabView* aplikacijom pomoću *Ethernet* sučelja. Prikazan je programski razvoj oba čvora sa svim potrebnim perifernim modulima kao što su: RFID čitač, LCD ekran osjetljiv na dodir, Ethernet sučelje i XBee bežični moduli. Na posljetku su opisani rezultati testiranja, prednosti i nedostatci sustava.

Ključne riječi: RFID, tag, ZigBee, XBee, Autorizacija, Atmel, AVR, mikroElektronika, LabView

ABSTRACT

Wireless RFID user authorization

This paper shows a development of a user authorization system using RFID technology and wireless communication. In a theoretic part of this paper, RFID technology and ZigBee technology of wireless communication are described. Next on, an entire system of wireless user authorization is designed using Master and Slave nodes. Idea is to manage a data base on the Master node, which is connected to the Slave nodes through wireless link. Slave nodes consist of RFID reader, LCD touch screen input keyboard for PIN input. Master node communicates with *LabView* application though Ethernet interface. In the paper, a program development is shown for both nodes, with all the peripheral modules such as: RFID reader, LCD touch screen, *Ethernet* interface and XBee wireless modules. In the end, test results are described as long as the advantages and disadvantages of this system.

Keywords: RFID, tag, ZigBee, XBee, Authorization, Atmel, AVR, mikroElektronika, LabView

POPIS KRATICA

Kratica	Značenje
ADC	Analog-to-Digital Converter
AES	Advance Encryption Standard
ALU	Arithmetic and Logical Unit
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
AT	ATtention command
BCD	Binary Coded Decimal
CE	Chip Enable
CISC	Complex Instruction Set Computer
CMOS	Complementary Metal-Oxide Semiconductor
CPHA	Clock PHAse
CPOL	Clock POLarization
CPU	Central Processor Unit
CRC	Cyclic Redundancy Check
CSMA-CA	Collision Sense Multiple Access with Collision Avoidance
CTS	Clear To Send
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
DSSS	Direct-Sequence Spread Spectrum
ED	Energy Detection
EEPROM	Electronically-Erasable Programmable Read-Only Memory
ENC	Elektronska Naplata Cestarine
EOF	End Of File
FAT	File Allocation Table
FDD	Floppy Disk Drive
FFD	Fully Functional Device
FSK	Frequency Shift Keying
GSM	Global System for Mobile communications
GTS	Guaranteed Time Slot
GUI	Graphical User Interface
HF	High Frequency
I^2C	Inter-Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
ID	IDentifier
IP	Internet Protocol
ISP	In-System Programming
ISR	Interrupt Service Routine
ITO	Indium Tin Oxide
JTAG	Joint Test Action Group

LF	Low Frequency
LAN	Local Area Network
LCD	Liquid Crystal Display
LOS	Line Of Sight
LQI	Link Quality Indicator
LSB	Least Significant Bit
MAC	Media Access Control
MF	Medium Frequency
MIPS	Million Instructions Per Second
MISO/SOMI	Master Input Slave Output /Slave Output Master Input
MMC/SD	Multi Media Card / Secure Digital
MOSI/SIMO	Master Output Slave Input/Slave Input Master Output
MSB	Most Significant Bit
MTU	Maximum Transmission Unit
NI	Network Identifier
NRZ	Non-Return-to-Zero
NWK	NetWorK layer
PAN	Personal Area Network
PC	Personal Computer
PDIP	Plastic Dual-In-line Package
PHY	PHYsical layer
PIN	Personal Identification Number
PSK	Phase-Shift Keying
PWM	Pulse Width Modulation
QPSK	Quadrature Phase Shift Keying
RAM	Random Access Memory
RF	Radio Frequency
RFD	Reduced Functional Device
RFID	Radio Frequency Identification
RISC	Reduced Instruction Set Computer
RO	Read Only
ROM	Read Only Memory
RTC	Real-Time Clock
RTS	Ready To Send
RW	Read / Write
SCK	Signal ClocK
SDA	Signal DAta
SD	Secure Digital
SMD	Surface Mount Devices
SMT	Surface Mount Technology
SPII	Serial Peripheral Interface
SQL	Structured Query Language
SRAM	Static Random Access Memory

TCP	Transmission Control Protocol-
THT	Through Hole Technology
TQFP	Thin Quad Flat Pack
TWI	Two Wire Interface
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
UHF	Ultra High Frequency
USART	Universal Synchronous and Asynchronous Receiver Transmitter
VHF	Very High Frequency
WORM	Write Once Read Many

ŽIVOTOPIS

Goran Horvat rođen je 23.02.1987. godine u Požegi, od majke Snježane i oca Borisa. Osnovnoškolsko obrazovanje završio je u Požegi. Godine 2001. upisuje Tehničku školu u Požegi, smjer Elektrotehničar. Na početku srednje škole uključuje se u rad Mladih tehničara i ACSL (eng. *American Computer Science League*) grupe i naredne četiri godine sudjeluje na mnogim natjecanjima. Posebno se može istaći osvojeno prvo mjesto na ACSL međunarodnom natjecanju četiri godine za redom i sudjelovanje na završnom natjecanju u Chicagu, Sjedinjene Američke Države. Godine 2004. sudjeluje na Prvom međunarodnom sajmu inovacija, novih proizvoda i tehnologija ARCA s radom: „Alarmni uređaj sa EEPROM ključem i mogućnošću programiranja“. Srednju školu maturira sa odličnim uspjehom 2005. godine. Te se godine upisuje na Preddiplomski studij Elektrotehnike na Elektrotehničkom fakultetu u Osijeku (smjer komunikacije) plasirajući se na 7. mjestu razredbenog postupka. U toku preddiplomskog studija osvaja državnu stipendiju za nadarene koju zadržava do kraja trajanja studija. Preddiplomski studij završava 2008. godine sa završnim radom na temu: „Ultrazvučno mjerenje i regulacija razine s ATMega8535“. Godine 2008. upisuje Diplomski studij Elektrotehnike (smjer komunikacije i informatika) na Elektrotehničkom fakultetu u Osijeku. Ponovo osvaja državnu stipendiju za deficitarna zanimanja koju zadržava do kraja studija

Goran Horvat

```
 //#include <built_in.h> ;

// Glcd module connections

sbit DATA at PINB0_bit;

sbit GLCD_CS1 at PORTD2_bit;
sbit GLCD_CS2 at PORTD3_bit;
sbit GLCD_RS  at PORTD4_bit;
sbit GLCD_RW  at PORTD5_bit;
sbit GLCD_EN  at PORTD6_bit;
sbit GLCD_RST at PORTD7_bit;

sbit GLCD_CS1_Direction at DDD2_bit;
sbit GLCD_CS2_Direction at DDD3_bit;
sbit GLCD_RS_Direction  at DDD4_bit;
sbit GLCD_RW_Direction  at DDD5_bit;
sbit GLCD_EN_Direction  at DDD6_bit;
sbit GLCD_RST_Direction at DDD7_bit;

char GLCD_DataPort at PORTC;
char GLCD_DataPort_Direction at DDRC;

sbit DRIVE_A at PORTA2_bit;
sbit DRIVE_B at PORTA3_bit;
sbit DRIVE_A_Direction at DDA2_bit;
sbit DRIVE_B_Direction at DDA3_bit;
const char READ_X_CHANNEL = 0;     // READ X line is connected to analog channel 0
const char READ_Y_CHANNEL = 1;     // READ Y line is connected to analog channel 1
// End Touch Panel module connections


const unsigned int ADC_THRESHOLD = 900;                   // Threshold value for press detecting


unsigned char const CEKAM[1024] = {
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,192,224,241,243,247,118,247,243,241,224,224,
  128,  0,  0,  0,240,240,240,240,240,112,112,112,112,112,112,112,
    0,  0,  0,240,240,240,240,240,  0,128,192,224,240,240,240,112,
   48, 16,  0,  0,  0,  0,  0,224,240,240,240,240,240,240,224,  0,
    0,  0,  0,  0,240,240,240,240,240,240,128,  0,  0,128,240,240,
  240,240,240,240,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0, 63,255,255,255,255,192,128,192,241,241,241,224,
   96,  0,  0,  0,255,255,255,255,255,142,142,142,142,142,142,128,
    0,  0,  0,255,255,255,255,255, 63, 31, 31,127,255,255,248,224,
  192,  0,  0,  0,192,248,255,255,255,255,239,239,255,255,255,255,
  248,192,  0,  0,255,255,255,255, 15,127,255,252,252,255,127, 15,
  255,255,255,255,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  1,  3,  3,  3,  3,  3,  3,  3,  1,  0,
    0,  0,  0,  0,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,
```

```
  64      0,  0,  0,  3,  3,  3,  3,  3,  0,  0,  0,  0,  0,  3,  3,  3,
  65      3,  3,  2,  2,  3,  3,  3,  3,  1,  0,  0,  0,  0,  1,  3,  3,
  66      3,  3,  2,  0,  3,  3,  3,  3,  0,  0,  3,  3,  3,  3,  0,  0,
  67      3,  3,  3,  3,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  68      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  69      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,128,128,128,
  70    128,128,  0,  0,  0,  0,128,128,128,128,128,128,  0,  0,  0,  0,
  71      0,  0,128,128,128,128,128,128,  0,  0,  0,  0,  0,  0,128,128,
  72    128,128,128,128,128,128,128,128,128,  0,  0,  0,  0,  0,128,128,
  73    128,128,128,128,128,128,128,128,128,128,128,  0,  0,128,128,128,
  74    128,128,  0,  0,  0,  0,  0,  0,128,128,128,128,128,128,128,  0,
  75      0,  0,  0,  0,  0,128,128,128,128,128,  0,  0,  0,  0,128,128,
  76    128,128,128,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  77      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,255,255,255,
  78    255,255,248,252,254,255,255,255,199,  3,  1,  0,  0,  0,  0,192,
  79    248,255,255,255,127,127,255,255,255,248,192,  0,  0,  0,255,255,
  80    255,255,255,115,243,243,255,255,191, 63, 30,  0,  0,  0,  3,  3,
  81      3,  3,255,255,255,255,255,  3,  3,  3,  3,  0,  0,255,255,255,
  82    255,255,  0,  0,  0,248,254,255,255,255,  7,  3,  7,143,143,143,
  83      7,  4,  0,  0,  0,255,255,255,255,255,  0,  0,  0,  0,255,255,
  84    255,255,255,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  85      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 31, 31, 31,
  86     31, 31,  1,  0,  0,  3,  7, 31, 31, 31, 30, 24, 16, 16, 30, 31,
  87     31, 31, 15,  7,  7,  7,  7, 15, 31, 31, 31, 30, 16,  0, 31, 31,
  88     31, 31, 31,  0,  1,  7, 15, 31, 31, 31, 28, 16,  0,  0,  0,  0,
  89      0,  0, 31, 31, 31, 31, 31,  0,  0,  0,  0,  0,  0, 31, 31, 31,
  90     31, 31,  0,  0,  0,  1,  7, 15, 31, 31, 30, 28, 30, 31, 31, 15,
  91      7,  3,  0,  0,  0,  3,  7, 15, 31, 31, 30, 28, 28, 30, 31, 31,
  92     15,  7,  3,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  93      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  94      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  95      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  96      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  97      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  98      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  99      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 100      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0
 101  };
 102
 103
 104
 105  unsigned char const UNOS[1024] = {
 106     0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 107     0,  0,  0,  0,  0,  0,  0,255,255, 24, 28,126,243,129,  0,124,
 108   254,131,  1,  1,131,254,124,  0,255,255,  1,  1,131,254,124,  0,
 109   140,140,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 110     0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 111     0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 112     0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 113     0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 114     0,  0,  0,  0,  0,240, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16,
 115    16,144,144, 16, 16, 16, 16, 17, 17, 16, 16, 16, 17, 17, 16,240,
 116     0,  1,  1,241, 17, 16, 16, 16, 17, 17, 17, 17, 17, 16,144,144,
 117   145,145, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 18,242,  2,  2,
 118     2,242, 18, 16, 16, 16, 18, 18, 18, 18, 18, 18,146,144,144,144,
 119    18, 18, 18, 18, 18, 18, 18, 16, 16, 16, 18,242,  2,  2,  2,242,
 120    18, 16, 16, 16, 16, 16, 16, 16, 16,144,144, 16, 16, 16,144,144,
 121    16, 16, 16, 16, 16, 16, 16, 16, 16,240,  0,  0,  0,  0,  0,  0,
 122     0,  0,  0,  0,  0,255,  0,  0,  0,  0,  0,  0,  0,  0,  0,  6,
 123     3,255,255,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,255,
 124     0,  0,  0,255,  0,  0,  0,  0,  0,  0,  0,  0,  0,129,193,112,
 125    56, 31,  7,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,255,  0,  0,
 126     0,255,  0,  0,  0,  0,  0,  0,  0,  0,  0,129,129,  8,  8,255,
 127   247,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,255,  0,  0,  0,255,
```

```
    0,  0,  0,  0,  0,  0,  0,  0,  0,129,195,126, 60,126,195,129,
    0,  0,  0,  0,  0,  0,  0,  0,  0,255,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,207, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72,
   72, 73, 73, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72,207,
    0,  0,  0,207, 72, 72, 72, 72, 72, 72, 72, 72, 72, 73, 73, 73,
   73, 73, 73, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72,207,  0,  0,
    0,207, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 73, 73, 73, 73,
   72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72,207,  0,  0,  0,207,
   72, 72, 72, 72, 72, 72, 72, 72, 72, 73, 73, 72, 72, 72, 73, 73,
   72, 72, 72, 72, 72, 72, 72, 72, 72,207,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,255,  0,  0,  0,  0,  0,  0,  0,  0,  0,128,
   96, 24,254,254,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,255,
    0,  0,  0,255,  0,  0,  0,  0,  0,  0,  0,  0,  0, 56, 62, 22,
   18,242,226,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,255,  0,  0,
    0,255,  0,  0,  0,  0,  0,  0,  0,  0,  0,248,252, 34, 34,230,
  196,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,255,  0,  0,  0,255,
    0,  0,  0,  0,  0,  0,  0,  0,  0,252,254,  2,  2,254,252,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,255,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0, 63, 32, 32, 32, 32, 32, 32, 32, 32, 32, 33,
   33, 33, 39, 39, 33, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 63,
    0,  0,  0, 63, 32, 32, 32, 32, 32, 32, 32, 32, 32, 34, 38, 36,
   36, 39, 35, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 63,  0,  0,
    0, 63, 32, 32, 32, 32, 32, 32, 32, 32, 32, 35, 39, 36, 36, 39,
   35, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 63,  0,  0,  0, 63,
   32, 32, 32, 32, 32, 32, 32, 32, 32, 35, 39, 36, 36, 39, 35, 32,
   32, 32, 32, 32, 32, 32, 32, 32, 32, 63,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,255,  1,  1,  1,  1,  1,  1,  1,  1,  1,  9,
    9,137,233,121, 25,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,255,
    0,  0,  0,255,  1,  1,  1,  1,  1,  1,  1,  1,  1,113,249,137,
  137,249,113,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,255,  0,  0,
    0,255,  1,  1,  1,  1,  1,  1,  1,  1,  1,241,249,  9,  9,249,
  241,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,255,  0,  0,  0,255,
    1,  1,  1,  1,161,161,  9,  9,  9,  9,161,161,  1,  1,169,169,
  129,161,161,  9,  9,  1,  1,  1,  1,255,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,255,128,128,128,128,128,128,128,128,128,128,
  156,159,131,128,128,128,128,128,128,128,128,128,128,128,128,255,
    0,  0,  0,255,128,128,128,128,128,128,128,128,128,143,159,144,
  144,159,143,128,128,128,128,128,128,128,128,128,128,255,  0,  0,
    0,255,128,128,128,128,128,128,128,128,128,136,153,145,145,143,
  135,128,128,128,128,128,128,128,128,128,128,255,  0,  0,  0,255,
  128,128,128,128,130,138,136,128,128,136,138,130,128,128,138,138,
  128,128,130,138,136,128,128,128,128,255,  0,  0,  0,  0,  0,  0
};


unsigned char const OK_bmp[54] = {
 255,  1,  1,  1,  1,225,241, 25,  9,  9, 25,241,225,  1,  1,249,
 249,129,225,241, 25,  9,  1,  1,  1,  1,255,
 255,128,128,128,128,135,143,152,144,144,152,143,135,128,128,159,
 159,129,128,135,159,152,128,128,128,128,255};

 // ------------------------------------------------------
// GLCD Picture name: u tijeku.bmp
// GLCD Model: KS0108 128x64
// ------------------------------------------------------


unsigned short sync_flag,     // in the sync routine if this flag is set
               one_seq,       // counts the number of 'logic one' in series
               data_in,       // gets data bit depending on data_in_1st and data_in_2nd
               cnt,           // interrupt counter
               cnt1, cnt2,    // auxiliary counters
               broj,x1, y1,x_cord128, y_cord64;
unsigned char KOD[4],broj_unesene_znamenke;
```

```
192
193   unsigned short data_index;    // marks position in data arrey
194   unsigned int i;
195   unsigned char ADRESA;
196   char podatci[256],ID_CODE[]="            ", PIN[]="    ";
197   const char *usp[]="uspjesna!";
198   const char *nusp[]="neuspjesna!";
199
200   char data_valid[64];
201   char bad_synch;               // variable for detecting bad synchronization
202   char txt[4], *output;
203   char ch;
204   unsigned int x_coord, y_coord, x_coord_old, y_coord_old, x_coord_diff, y_coord_diff;
205   long         x_coord128, y_coord64;
206   int          cal_x_min, cal_y_min, cal_x_max, cal_y_max;
207   unsigned char uart_rd;
208
209   typedef char * string;
210
211   string a="A Long string";
212
213
214   char PressDetect() {
215     unsigned adc_rd;
216     char result;
217
218     // PRESS detecting
219     DRIVE_A = 0;         // DRIVEA = 0 (LEFT drive off, RIGHT drive off, TOP drive on)
220     DRIVE_B = 0;         // DRIVEB = 0 (BOTTOM drive off)
221     Delay_ms(5);
222      DDRC=0xff;
223
224     adc_rd = ADC_Read(READ_Y_CHANNEL);      // READ-Y
225     result = (adc_rd > ADC_THRESHOLD);      // if logical one is detectet
226
227     //debouncing, repeat detecting after 2ms
228     Delay_ms(2);
229     adc_rd = ADC_Read(READ_Y_CHANNEL);      // READ-Y
230     result = result & (adc_rd > ADC_THRESHOLD);
231     return result;
232   }
233
234   unsigned int GetX() {
235     unsigned int result;
236     //reading X
237     DRIVE_A = 1;          // DRIVEA = 1 (LEFT drive on, RIGHT drive on, TOP drive off)
238     DRIVE_B = 0;          // DRIVEB = 0 (BOTTOM drive off)
239
240     Delay_ms(5);
241     result = ADC_Read(READ_X_CHANNEL);      // READ-X (BOTTOM)
242     return result;
243   }
244
245   unsigned int GetY() {
246     unsigned int result;
247     //reading Y
248     DRIVE_A = 0;          // DRIVEA = 0 (LEFT drive off, RIGHT drive off, TOP drive on)
249     DRIVE_B = 1;          // DRIVEB = 1 (BOTTOM drive on)
250
251     Delay_ms(5);
252     result = ADC_Read(READ_Y_CHANNEL);      // READ-X (LEFT)
253     return result;
254   }
255
```

```
256   void Calibrate() {
257
258     Glcd_Dot(0,63,1);
259     Glcd_Write_Text("LEFT",12,3,1);
260
261     while (!PressDetect());
262
263     // get calibration constants (reading and compensating TouchPanel nonlinearity)
264     cal_x_min = GetX() - 10;
265     cal_y_min = GetY() - 10;
266     Delay_ms(200);
267
268     Glcd_Fill(0);
269     Glcd_Dot(127,0,1);
270     Glcd_Write_Text("RIGHT",12,4,1);
271     while (!PressDetect()) ;
272
273     // get calibration constants (reading and compensating TouchPanel nonlinearity)
274     cal_x_max = GetX() + 5;
275     cal_y_max = GetY() + 5;
276     Delay_ms(200);
277
278     EEPROM_Write(0x10, (cal_x_min & 0xFF));
279     EEPROM_Write(0x11, ((cal_x_min >> 8) & 0xFF));
280     EEPROM_Write(0x12, (cal_y_min & 0xFF));
281     EEPROM_Write(0x13, ((cal_y_min >> 8) & 0xFF));
282     EEPROM_Write(0x14, (cal_x_max & 0xFF));
283     EEPROM_Write(0x15, ((cal_x_max >> 8) & 0xFF));
284     EEPROM_Write(0x16, (cal_y_max & 0xFF));
285     EEPROM_Write(0x17, ((cal_y_max >> 8) & 0xFF));
286
287     }
288
289   void Initialize() {
290     DRIVE_A_Direction = 1;       // Set DRIVE_A pin as output
291       DRIVE_B_Direction = 1;       // Set DRIVE_B pin as output
292                 // Initialize GLCD
293
294                  }
295
296     // This is external INT1 interrupt (for sync and sample)
297     //   - this interrupt is enabled once we get falling edge on RB0 (in the INT0
      interrupt routine)
298     //     it is enabled until we get 128 data bits
299
300   void INT1_Interrupt() org IVT_ADDR_INT1 {         // clock interrupt, na rastući brid
      se aktivira
301         // if (INTCON3.INT1IF = 1) and (INTCON3.INT1IE = 1) then
302         cnt++;                                          // count interrupts on INT1 pin
      (RB1)
303        GICR.INT1=1;
304     }
305
306     // This is external INT0 interrupt (for sync start)
307     //   - once we get falling edge on RB0 we are disabling INT0 interrupt
308
309   void INT0_Interrupt() org IVT_ADDR_INT0 {  //data interrupt
310         GICR.INT0=0;
311         GICR.INT1=0;
312
313
314        if (  MCUCR== 0b00001110) MCUCR= 0b00001111;     //alternira okidanje na padajući
      ili rastuci brid
315          else MCUCR= 0b00001110;
```

```
316
317         if (one_seq !=0 && cnt>28 && cnt<37)        // granica trajanja pola bita
318        {     if (one_seq == 17){
319                             sync_flag=1;
320                             GICR.INT0=0;
321                             cnt=0;
322                             one_seq=0;
323                             GICR.INT1=1;
324                             GICR.INT0=0;
325                             MCUCR= 0b00001110;
326
327                             goto kraj;
328
329                             }}
330        else if (one_seq!=0){
331                   MCUCR= 0b00001110;
332                   one_seq=0;
333                     cnt=0;
334
335                   GICR.INT0=1;
336
337                   goto kraj;
338                   }
339
340       GICR.INT1=1;
341        one_seq++;
342        cnt=0;
343
344        GICR.INT0=1;
345        kraj:  ;
346    }
347
348    char CRC_Check(char *bit_array) {
349
350   char row_count, row_bit, column_count;
351   char row_sum, column_sum;
352   char row_check[5];
353   char column_check[11];
354
355      // row parity check:
356      row_count = 9;                      // count rows
357      while (row_count < 59) {
358        column_count = 0;                 // count columns
359        while (column_count < 5) {
360          row_check[column_count] = bit_array[row_count+column_count];
361          column_count++;
362        }
363        row_bit = 0;                      // count row bits
364        row_sum = 0;
365        while (row_bit < 4) {
366          row_sum = row_sum + row_check[row_bit];
367          row_bit++;
368        }
369
370        if (row_sum.F0 != row_check[4].F0) {
371          return 0;
372        }
373        row_count = row_count + 5;
374      }
375      // end row parity check
376
377      // column parity check
378      column_count = 9;                   // count columns
379      while (column_count < 13) {
```

```
380       row_bit = 0;                    // count column bits
381       row_count = 0;                  // count rows
382       while (row_bit < 11) {
383         column_check[row_bit] = bit_array[column_count+row_count];
384         row_bit++;
385         row_count = row_count + 5;
386       }
387
388       row_bit = 0;                    // count column bits
389       column_sum = 0;
390       while (row_bit < 10) {
391         column_sum = column_sum + column_check[row_bit];
392         row_bit++;
393       }
394
395       if (column_sum.F0 != column_check[10].F0) {
396         return 0;
397       }
398       column_count++;
399     }
400     // end column parity check
401     if (bit_array[63] == 1) {
402       return 0;
403     }
404     return  1;
405   }
406
407
408
409     void Sound( int freq, unsigned int dur){
410      unsigned int p,q;
411
412      for (p=0;p<dur;p++){
413      PORTA |= 128;
414      for (q=0;q<freq;q++) asm nop;
415      PORTA &= ~128;
416      for (q=0;q<freq;q++) asm nop; }
417
418
419     }
420
421
422
423    char * Mid( char * Source,int Start,int Length) {
424           char *result;
425           char *temp;
426           cnt=0;
427           *result=' ';
428      for(cnt1=Start;cnt1<=Length;cnt1++) result[cnt++]= Source[cnt1];
429       result[cnt++]='\0';
430           return result;
431   }
432
433
434
435   void Konfiguracija(){
436   unsigned char br_slavea=0;
437   char tx[3];
438   while ((PINA & 64) == 64) asm nop;
439     Glcd_Write_Text("  Unesite redni broj  ", 0, 0, 1);
440     Glcd_Write_Text("     slave cvora:     ", 0, 1, 1);
441
442     Glcd_Write_Text("   -               +  ", 0, 3, 1);
443
```

```
444
445     Glcd_Write_Text("          OK          ", 0, 6, 1);
446
447                 while(1){
448                 ByteToStr(br_slavea, tx);
449                   Glcd_Write_Text(tx , 55, 3, 1);
450                  if (PressDetect()){
451                                  x_coord = GetX() - cal_x_min;
452                  y_coord = GetY() - cal_y_min;
453
454
455                  // When lifting pen from the touchpanel surface GetX and GetY readings
456                  // (after correct PressDetect reading) may be incorrect
457
458                  x_coord128 =((long)x_coord * 128) / (cal_x_max - cal_x_min);
459                  y_coord64 = (64 -(y_coord *64) / (cal_y_max - cal_y_min));
460
461
462
463                  if (x_coord128 >=10 && x_coord128 <= 50 && y_coord64 >=16 && y_coord64
    <=45 && br_slavea>0) br_slavea--;
464                  if (x_coord128 >=80 && x_coord128 <= 120 && y_coord64 >=16 && y_coord64
    <=45 && br_slavea<32) br_slavea++;
465
466                  if (x_coord128 >=45 && x_coord128 <= 75 && y_coord64 >=48 && y_coord64
    <=60 ) break;
467
468
469                    Delay_ms(10);
470                    while (PressDetect()) ;
471                  }
472
473
474
475                  }
476
477   EEPROM_Write(0x20,br_slavea);
478           Delay_ms(1000);
479           while (UART1_Data_Ready())uart_rd = UART1_Read();
480           UART1_Write_Text("+++");
481                             while (1)if (UART1_Data_Ready()) if (UART1_Read() ==0x0d)
    break;
482
483           UART1_Write_Text("ATNI SL");
484           UART1_Write(tx[1]);
485           UART1_Write(tx[2]);
486           UART1_Write(0x0D);
487                           while (1)if (UART1_Data_Ready()) if (UART1_Read() ==0x0d) break;
488           UART1_Write_Text("ATID 555");
489           UART1_Write(0x0D);
490                           while (1)if (UART1_Data_Ready()) if (UART1_Read() ==0x0d) break;
491
492       Glcd_Fill(0x00);
493       Glcd_Write_Text("   UGASITE CVOR  ", 0, 4, 1);
494        while(1) asm nop;
495   }
496
497   // main program
498   void main() {
499
500     unsigned short temp;
501     DDRD  = 0b11110011;
502     PORTD  = 0x00;
503     TCCR1A = 0x00;  //Normal operation  TIMER1 za mjerenje širine pulsa injektora
```

```
504     TCCR1B = 0x00;  //Prescaler=64 inaće ali je clock source disabled, ne broji
505        DDRB=0b00001100;
506       PORTB= 0b00001100;
507     PORTD.B2=1;
508     PORTD.B3=1;
509
510           // Initialise USART communication
511      Initialize();
512                      // Initialize LCD
513       DDRA.B6=0;          //tipka za konfiguraciju
514      UART1_Init(38400);
515
516         Glcd_Init();
517        Glcd_Fill(0x00);
518         Initialize();
519
520
521      Delay_ms(200);
522
523
524             temp = EEPROM_Read(0x11);
525           if (temp == 0xff) Calibrate( ); else {
526                     cal_x_min= EEPROM_Read(0x10);
527                     cal_x_min |= EEPROM_Read(0x11)<<8;
528                     cal_y_min= EEPROM_Read(0x12);
529                     cal_y_min |= EEPROM_Read(0x13)<<8;
530                     cal_x_max= EEPROM_Read(0x14);
531                     cal_x_max |= EEPROM_Read(0x15)<<8;
532                     cal_y_max= EEPROM_Read(0x16);
533                     cal_y_max |= EEPROM_Read(0x17)<<8;
534
535
536           }
537
538           temp = EEPROM_Read(0x20);
539               if (temp==0xff || ((PINA & 64) == 64)) Konfiguracija();
540
541
542
543
544     Delay_ms(1000);
545          while (UART1_Data_Ready())uart_rd = UART1_Read();
546          UART1_Write_Text("+++");
547          while (1)if (UART1_Data_Ready()) if (UART1_Read() ==0x0d) break;
548         UART1_Write_Text("ATND");
549        UART1_Write(0x0D);
550        while (1)if (UART1_Data_Ready()) if (UART1_Read() ==0x0d) break;
551
552        UART1_Write_Text("ATDN MASTER");
553        UART1_Write(0x0D);
554
555        while (1)if (UART1_Data_Ready()) if (UART1_Read() ==0x0d) break;
556
557
558         Delay_ms(1000);
559            //UART1_Init(38400);
560
561
562
563
564
565
566
567
```

```
568                   ADRESA= EEPROM_Read(0x20);
569     sync_flag = 0;                // sync_flag is set when falling edge on RB0 is detected
570     one_seq = 0;                  // counts the number of 'logic one' in series
571     data_in = 0;                  // gets data bit
572     data_index = 0;               // marks position in data arrey
573     cnt = 0;                      // interrupt counter
574     cnt1 = 0;                     // auxiliary counter
575     cnt2 = 0;                     // auxiliary counter
576
577     // setup interrupts
578
579
580
581
582
583     while (1) {
584     ponovo:
585        Glcd_Fill(0x00);
586     Glcd_Image(CEKAM);
587     Delay_ms(200);
588
589       MCUCR= 0b00001110;      // Interrupt on falling edge on RB0
590                           // Interrupt on rising edge on RB1
591   //  Sound_Init(&PORTA,7);
592
593
594    GICR.INT0=0;           // turn OFF interrupt on INT0
595    GICR.INT1=0;             // turn OFF interrupt on INT1
596
597      SREG_I_bit = 1;           // enable GIE
598
599       bad_synch = 0;             // set bad synchronization variable to zero
600       cnt = 0;                   // reseting interrupt counter
601       sync_flag = 0;             // reseting sync flag
602       DDRB=0b00001100;
603       DDRA.B7=1;
604
605
606       DDRD  = 0b11110011;
607      // PORTD=0b10110000;
608
609       Delay_ms(200);
610
611
612
613
614       GICR.INT1=0;
615                                 // disable external interrupt on RB1 (for sync and sample)
616
617       PORTB.B2=0;
618     PORTB.B3=0;
619
620       GICR.INT0=1;           // enable external interrupt on RB0 (start sync procedure)
621
622
623       while (sync_flag == 0) {   // waiting for syns_flag ili čeka da se pojavi 9 jedinica
624        asm nop
625       }
626       while (cnt != 48) {        // waiting 16 clocks on RB1 (positioning for sampling)
627        asm nop
628       }
629
630       cnt = 0;
631       podatci[0] = (DATA) & 1 ;
```

```
632       PORTD.B2=0;
633     PORTD.B3=0;
634       for (data_index = 1; data_index !=0; data_index++) {   // getting 128 bits of data
      from RB0
635         while (cnt != 32) {                                    // getting bit from RB0
      every 32 clocks on RB1
636           asm nop
637         }
638         cnt = 0;
639        podatci[data_index] =  (DATA) & 1;                                          //
      reseting interrupt counter
640                               // geting bit
641         if(data_index & 1)
642         if (!(podatci[data_index] ^ podatci[data_index-1]))
643            {
644               bad_synch = 1;
645              break;                              //bad synchronisation
646            }
647       }
648       GICR.INT0=0;                                      // disable external interrupt on
      RB1 (for sync and sample)
649     PORTB.B2=1;
650     PORTB.B3=0;
651     DDRD  = 0b11111111;
652
653
654
655
656       if (bad_synch) {
657
658          ByteToStr(data_index, txt);
659         // Glcd_Write_Text("¤", 0, 7, 1);
660               Delay_ms(100);
661             goto ponovo;
662        }
663
664
665
666        for(cnt1 = 0; cnt1 <= 127; cnt1++) {    // we are counting 'logic one' in the data
      array
667         if (podatci[cnt1 << 1] == 1) {
668           one_seq++;
669           }
670         else {
671           one_seq = 0;
672           }
673
674         if (one_seq == 9) {                 // if we get 9 'logic one' we break from the
      loop
675             break;
676       }
677       }
678
679                       //   (the position of the last  'logic one' is in the cnt1)
680      // if we got 9 'logic one' before cnt1 position 73
681           if ((one_seq == 9) && (cnt1 < 73)) {                                    //
      we write that data into data_valid array
682          data_valid[0] = 1;                 //   (it has to be before cnt1 position 73
      in order
683          data_valid[1] = 1;                 //    to have all 64 bits available in data
      array)
684          data_valid[2] = 1;
685          data_valid[3] = 1;
686          data_valid[4] = 1;
```

```
687          data_valid[5] = 1;
688          data_valid[6] = 1;
689          data_valid[7] = 1;
690          data_valid[8] = 1;
691          for(cnt2 = 9; cnt2 <= 63; cnt2++) {      // copying the rest of data from the
     data array into data_valid array
692             cnt1++;
693             data_valid[cnt2] = podatci[cnt1 << 1];
694           }
695           }                 // if data in data_valid array pass the CRC check
696
697           if (CRC_Check(data_valid) == 1) {
698
699                 Sound(100, 1000);
700                           Glcd_Fill(0x00);
701                                   Glcd_Set_Font(Character8x7, 8, 7, 32);
702                          Glcd_Write_Text("Unesite", 35, 1, 1);
703                          Glcd_Write_Text("pristupni", 25, 3, 1);
704                          Glcd_Write_Text("kod!", 50, 5, 1);
705                    Delay_ms(1000);
706              Glcd_Fill(0x00);
707              Glcd_Image(UNOS);
708                          broj_unesene_znamenke=0;
709               while(1){
710
711                   if (PressDetect()){
712
713                   x_coord = GetX() - cal_x_min;
714                   y_coord = GetY() - cal_y_min;
715
716
717                   // When lifting pen from the touchpanel surface GetX and GetY readings
718                   // (after correct PressDetect reading) may be incorrect
719
720                   x_coord128 =((long)x_coord * 128) / (cal_x_max - cal_x_min);
721                   y_coord64 = (64 -(y_coord *64) / (cal_y_max - cal_y_min));
722
723
724
725                   broj=0;
726                   if ((x_coord128>= 5 && x_coord128<=91 && y_coord64>= 12)|| (x_coord128>=
     95 && y_coord64<=45 && y_coord64>=30)  ){  //lijeva tri stupca,tipkovnica
727                         x1=x_coord128-5;
728                         y1=y_coord64-12;
729                         broj=x1/30+1;
730                         broj = broj+ (unsigned char)(y1/18)*3;
731                         if (x_coord128>=95) broj=0;
732
733                        if (broj_unesene_znamenke<4){
734                         KOD[broj_unesene_znamenke]=broj;
735                         broj_unesene_znamenke++;
736
737                             Glcd_Set_Font(Character8x7, 8, 7, 32);
738                            Glcd_Write_Text("*", 50+(broj_unesene_znamenke*10),0, 1);
739
740                            Sound(100, 300);
741                         //treba dodati stavljanje zvjezdice kada se unese znamenka
742                            }
743                   }
744
745                   if (x_coord128>= 95 && x_coord128<=121 ){    //desni stupac
746
747
748                       if(y_coord64>= 12 && y_coord64<=27) {     //tipka za brisanje X
```

```
749
750                         Sound(100, 300);
751
752                               if( broj_unesene_znamenke ==0) break;
753                                  if(broj_unesene_znamenke==4){
754
755                                                    Glcd_Set_X(95);
756                                                 Glcd_Set_Page(6);
757                                                   for (i=863;i<1024;i++){
758                                                       Glcd_Write_Data(UNOS[i]);
759                                                       if (i==896){ Glcd_Set_Page(7);
760                                                                    i=990;
761                                                                  Glcd_Set_X(95); }
762
763                                                   }
764                                    }
765
766                                     Glcd_Write_Text(" ", 50+(broj_unesene_znamenke*10), 0, 1);
767                                    broj_unesene_znamenke--;
768                                     }
769
770                         else if ( y_coord64>45 ){    // OK tipka
771                             if (broj_unesene_znamenke==4) {
772                             Sound(100, 500);
773                             break;          }
774
775                                  }
776
777                     }
778
779                      if(broj_unesene_znamenke==4){
780                                                   //enable OK button
781                                                 Glcd_Set_X(95);
782                                                 Glcd_Set_Page(6);
783                                                   for (i=0;i<54;i++){
784                                                       Glcd_Write_Data(OK_bmp[i]);
785                                                       if (i==26){ Glcd_Set_Page(7);
786                                                                  Glcd_Set_X(95); }
787                                                       }
788                                                   }
789
790                            Delay_ms(100);
791
792                             while (PressDetect()) ;
793
794                      }
795                     }
796
797                     if ( broj_unesene_znamenke==4){//slanjee podataka na zigbee
798
799                                // Write text in second row
800
801                              //   ByteToStr(ADRESA, txt);
802
803                           Delay_ms(100);
804                     for (i=0;i<=3;i++){
805                     ByteToStr( KOD[i], txt);
806                       PIN[i]=txt[2];
807                       }
808                       Glcd_Fill(0x00);
809
810
811                       Delay_ms(100);
812
```

```
813                       cnt1=0;
814                         // Glcd_Image(AUTH);
815
816
817                         Glcd_Write_Text("Autorizacija", 10, 2, 1);
818                          Glcd_Write_Text("u tijeku!", 21, 4, 1);
819                        //treba ispisti nalcd AUTORIZACIIIIIA U TIJEKU
820                  for (i = 13; i <= 60; i=i+5){                 // This part of the code
821
822
823                       one_seq=data_valid[i-4]*8+ data_valid[i-3]*4 + data_valid[i-2]*2 +
    data_valid[i-1]  ;
824
825                       ByteToStr(one_seq, txt);
826
827                                       switch (one_seq) {
828                    case 10:ID_CODE[cnt1]=65 ; break;
829                    case 11:ID_CODE[cnt1]=66 ; break;
830                    case 12:ID_CODE[cnt1]=67 ; break;
831                    case 13:ID_CODE[cnt1]=68 ; break;
832                    case 14:ID_CODE[cnt1]=69 ; break;
833                    case 15:ID_CODE[cnt1]=70 ; break;
834                    default: ID_CODE[cnt1]=txt[2] ;
835
836                                                 }
837                cnt1++;                                  //  dislays the number of the specific
    RfID CARD
838                }                                              //   specific to a single RfID
    CARD
839                 ID_CODE[cnt1]='\0';
840                   ByteToStr(ADRESA, txt);
841                 UART1_Write_Text("SL");
842                              UART1_Write(txt[1]);
843                              UART1_Write(txt[2]);
844                 UART1_Write_Text(":");
845                 UART1_Write_Text(ID_CODE);            // Line Feed (view ASCII chart)
846                 UART1_Write(0x09);
847                 UART1_Write_Text(PIN);
848                 UART1_Write(0x0D);
849
850
851
852          while (UART1_Data_Ready())uart_rd = UART1_Read();
853
854                    cnt1=0;
855
856                    //for (i = 0; i <= 12; i++) output[i]=0x20;
857                    *output=' ';
858
859
860                             i=0 ;
861                     while (i<25000) {                     // Endless loop
862                     Delay_us(100);
863                     i++;
864                      if (UART1_Data_Ready()) {       // If data is received,
865                        uart_rd = UART1_Read();       //   read the received data,
866                        if (uart_rd==0x0d) break;
867                        cnt1++;
868                        output[cnt1]=uart_rd;
869                                //   and send data via UART
870                      }
871                     }
872                     cnt1++;
873                     output[cnt1]='\0';
```

```
874
875                     if  (  strstr(output, "OK")!=0){
876
877
878                      Glcd_Set_Font(Character8x7, 8, 7, 32);
879
880                         Glcd_Write_Text("uspjesna!", 20, 4, 1);
881
882
883                     output[1]=0x20;
884                     output[2]=0x20;
885                     Glcd_Write_Text(Mid(output,4,18), 0, 7, 1);
886                     Glcd_Write_Text("Korisnik:", 0, 6, 1);
887                             Sound(100, 200);
888                             Sound(200, 100);
889
890                     }
891                     else if ( strstr(output, "PIN")!=0) {
892
893                             Glcd_Set_Font(Character8x7, 8, 7, 32);
894                             Glcd_Write_Text(" Neispravan ", 10, 2, 1);
895                             Glcd_Write_Text("pristupni kod!", 5, 4, 1);
896                            Delay_ms(100);
897                            Sound(100, 3000);
898
899
900                     }else{
901
902                           Glcd_Set_Font(Character8x7, 8, 7, 32);
903                           Glcd_Write_Text("neuspjesna!", 15, 4, 1);
904                            Delay_ms(100);
905                            Sound(100, 3000);
906                            }
907
908                 TCNT1H=0x0F;
909                 TCNT1L=0x00;
910                 Delay_ms(2000);
911                 //while (!PressDetect());
912                  }
913
914
915
916                  }
917                  Delay_ms(500);
918       }
919      }
```

```
1
2   #define SPI_Ethernet_HALFDUPLEX     0
3   #define SPI_Ethernet_FULLDUPLEX     1
4
5   // mE ehternet NIC pinout
6   sfr sbit SPI_Ethernet_Rst at PORTB4_bit;
7   sfr sbit SPI_Ethernet_CS  at PORTB5_bit;
8   sfr sbit SPI_Ethernet_Rst_Direction at DDB4_bit;
9   sfr sbit SPI_Ethernet_CS_Direction  at DDB5_bit;
10
11  unsigned char httpMethod[] = "GET /";
12
13
14  sbit LCD_RS at PORTC2_bit;
15  sbit LCD_EN at PORTC3_bit;
16  sbit LCD_D4 at PORTC4_bit;
17  sbit LCD_D5 at PORTC5_bit;
18  sbit LCD_D6 at PORTC6_bit;
19  sbit LCD_D7 at PORTC7_bit;
20
21  sbit LCD_RS_Direction at DDC2_bit;
22  sbit LCD_EN_Direction at DDC3_bit;
23  sbit LCD_D4_Direction at DDC4_bit;
24  sbit LCD_D5_Direction at DDC5_bit;
25  sbit LCD_D6_Direction at DDC6_bit;
26  sbit LCD_D7_Direction at DDC7_bit;
27
28
29
30  #define NULL    0
31
32  #define Spi_Ethernet_FULLDUPLEX 1
33
34  #define TRANSMIT_START  0x19AE
35  #define REPLY_START     (TRANSMIT_START + 1)            // reply buffer starts after
    per packet control byte
36
37
38  #define ERDPTL  0x00
39  #define ERDPTH  0x01
40  #define ECON1   0x1f
41  #define EDMACSL 0x16
42  #define EDMACSH 0x17
43
44  /*********************************
45  * RAM variables
46  */
47  unsigned char   myMacAddr[6] = {0x00, 0x14, 0xA5, 0x76, 0x19, 0x3f} ;   // my MAC address
48  unsigned char   myIpAddr[4] = {192, 168, 1, 50} ;                       // my IP address
49  unsigned char   getRequest[35] ;                                         // HTTP request
    buffer
50  char   dyna[1024] ;                                                  // buffer for dynamic
    response
51  unsigned long   httpCounter = 0 ;                                        // counter of
    HTTP requests
52  unsigned char   *lcdch;
53  char *korisnici;
54
55  unsigned char sec, min1, hr, week_day, day, mn, year, temp;
56  char *txt, tnum[10];
57
58  bit adres_f, LAN_korisnici, write_f, brisi_f;
59  sbit Mmc_Chip_Select at PORTG1_bit;
60  sbit Mmc_Chip_Select_Direction at DDG1_bit;
```

```
61  // eof MMC module connections
62
63  const LINE_LEN = 43;
64  char err_txt[20]       = "FAT16 not found";
65  char file_contents[LINE_LEN] = "XX MMC/SD FAT16 library by Anton Rieckert\n";
66  char           filename[14] = "USERS.TXT";          // File names
67  char           slave_filename[14] = "SL x.TXT";          // File names  LAN_filename
68  char           LAN_filename[14] = "SL x.TXT";          // File names
69  char           reply[14] = "OK> SL x\r\n";          // File names
70  char result[30];
71  unsigned short loop, loop2;
72  unsigned long  i, size,stao_na;
73  char           output[25]="                        ", adresiranje[10]="ATDN SL  ",
    tekst_imena[20]= "                  ";
74  unsigned char cnt, uart_rd, par_cnt;
75
76   unsigned int cnt1, cnt2;
77
78   unsigned int    len ;                          // my reply length
79               unsigned char a;
80                                    char character;
81    unsigned char IpAddr[4]  = {192, 168,   1,  2 };  // remote IP address
82
83  void M_Open_File_Append(char tekst);
84
85
86  #define putConstString  SPI_Ethernet_putConstString
87  #define putString  SPI_Ethernet_putString
88
89  unsigned int    SPI_Ethernet_UserTCP(unsigned char *remoteHost, unsigned int remotePort,
     unsigned int localPort, unsigned int reqLength)
90          {
91           return(0)  ;
92          }
93
94  unsigned int    SPI_Ethernet_UserUDP(unsigned char *remoteHost, unsigned int remotePort,
     unsigned int destPort, unsigned int reqLength)
95          {
96
97            //Lcd_Cmd(_LCD_CLEAR);
98
99            if (destPort != 10000) return(0);
100         for(i = 0 ; i < 35 ; i++)
101                {
102                a= SPI_Ethernet_getByte() ;
103                getRequest[i] =a;
104              // Lcd_Chr_CP(a);
105                }
106         getRequest[i] = 0 ;
107         len=0;
108
109         if(  getRequest[0] == 'U' && getRequest[1]=='S' ){
110
111           if(  getRequest[2] == '+') write_f=1;
112           if(  getRequest[2] == 'x') brisi_f=1;
113
114           return(0);
115
116
117         }
118
119
120
121          if(  getRequest[0] == 'S' && getRequest[1]=='L' ){       // only GET method is
```

```
      supported here
122
123           if(  getRequest[2] == '*' && getRequest[3]=='*' )  {
124                len =  putString(output);
125                return(len);}
126                 stao_na=0;
127
128                 adres_f=1;
129                 reply[6]=getRequest[2];
130                reply[7]=getRequest[3];
131                LAN_filename[3] = getRequest[3];
132                LAN_filename[2] = getRequest[2];
133              if(  getRequest[2] == 'x' && getRequest[3]=='x' )  LAN_korisnici=1;
134                  len =  putString(reply);
135
136                 }
137
138            if( getRequest[0] == 'N' && getRequest[1]=='L'){
139             adres_f=1;       // only GET method is supported here
140             len =  putString(dyna);}
141
142
143
144
145
146            return(len) ;           // back to the library with the length of the UDP reply
147           // return(0)  ;
148           }
149
150
151
152   char *Mid( char *Source,int Start,int Length) {
153    char *result;
154          cnt2=0;
155      *result=' ';
156      for(cnt1=Start;cnt1<=Length;cnt1++) result[cnt2++] = Source[cnt1];
157     result[cnt2++]='\0';
158    return result;
159   }
160
161    void Zero_Fill(char *value) {    // fill text repesentation
162     if (value[1] == 0) {           //      with leading zero
163       value[1] = value[0];
164       value[0] = 48;
165       value[2] = 0;
166     }
167   }//~
168
169   //--------------------- Reads time and date information from RTC (DS1307)
170   void Read_Time(char *sec, char *min, char *hr, char *week_day, char *day, char *mn, char
       *year) {
171     TWI_Start();
172     TWI_Write(0xD0);
173     TWI_Write(0);
174     TWI_Start();
175     TWI_Write(0xD1);
176     *sec =TWI_Read(1);
177     *min =TWI_Read(1);
178     *hr =TWI_Read(1);
179     *week_day =TWI_Read(1);
180     *day =TWI_Read(1);
181     *mn =TWI_Read(1);
182     *year =TWI_Read(0);
183     TWI_Stop();
```

```
184   }//~
185
186   //-------------------- Formats date and time
187   void Transform_Time(char  *sec, char *min, char *hr, char *week_day, char *day, char *mn
      , char *year) {
188     *sec  =  ((*sec & 0x70) >> 4)*10 + (*sec & 0x0F);
189     *min  =  ((*min & 0xF0) >> 4)*10 + (*min & 0x0F);
190     *hr   =  ((*hr & 0x30) >> 4)*10 + (*hr & 0x0F);
191     *week_day =(*week_day & 0x07);
192     *day  =  ((*day & 0xF0) >> 4)*10 + (*day & 0x0F);
193     *mn   =  ((*mn & 0x10) >> 4)*10 + (*mn & 0x0F);
194     *year =  ((*year & 0xF0)>>4)*10+(*year & 0x0F);
195   }//~
196
197   //-------------------- Output values to LCD
198   void Display_Time(char sec, char min, char hr, char week_day, char day, char mn, char
      year) {
199      switch(week_day){
200        case 1: txt="Ned"; break;
201        case 2: txt="Pon"; break;
202        case 3: txt="Uto"; break;
203        case 4: txt="Sri"; break;
204        case 5: txt="Cet"; break;
205        case 6: txt="Pet"; break;
206        case 7: txt="Sub"; break;
207      }
208        Lcd_Cmd(_LCD_CLEAR);
209         Lcd_Out(1,1,txt);
210
211        Lcd_Chr(1,8,'.');
212     Lcd_Chr(1,11,'.');
213     txt = "Time:";
214     Lcd_Out(2,1,txt);
215     Lcd_Chr(2,8,':');
216     Lcd_Chr(2,11,':');
217     txt = "20";
218     Lcd_Out(1,12,txt);
219
220
221      Lcd_Chr(1, 6, (day / 10)   + 48);    // Print tens digit of day variable
222      Lcd_Chr(1, 7, (day % 10)   + 48);    // Print oness digit of day variable
223      Lcd_Chr(1, 9, (mn / 10) + 48);
224      Lcd_Chr(1,10, (mn % 10) + 48);
225      Lcd_Chr(1,14,  (year/10)  + 48);          // Print year vaiable + 8 (start from year
      2008)
226      Lcd_Chr(1,15,  (year%10)  + 48);
227
228      Lcd_Chr(2, 6, (hr / 10)   + 48);
229      Lcd_Chr(2, 7, (hr % 10)   + 48);
230      Lcd_Chr(2, 9, (min / 10) + 48);
231      Lcd_Chr(2,10, (min % 10) + 48);
232      Lcd_Chr(2,12, (sec / 10) + 48);
233      Lcd_Chr(2,13, (sec % 10) + 48);
234
235   }//~
236
237   //------------------ Performs project-wide init
238   void Init_Main() {
239
240      Lcd_Cmd(_LCD_CLEAR);
241     TWI_Init(100538);                        // initialize I2C
242
243     Lcd_Cmd(_LCD_CURSOR_OFF);
244   }//~
```

```
245  /*
246  * main entry
247  */
248
249
250
251
252
253  int M_Open_File_Read(char find_str[]) {
254    char character;
255    unsigned char cnt3;
256
257    int flag;
258
259         flag=0;
260         cnt2=0;
261     for(cnt1=5;cnt1<=19;cnt1++) result[cnt2++] = find_str[cnt1];
262      result[cnt2++]='\0';
263    //filename[6] = '0';
264
265
266    cnt3=0;
267    Mmc_Fat_Assign(&filename, 0);
268    Mmc_Fat_Reset(&size);            // To read file, procedure returns size of file
269    for (i = 1; i <= size; i++) {
270      Mmc_Fat_Read(&character);
271
272      if (character == result[cnt3])  {
273
274            cnt3++;
275            if (cnt3==15) break;
276            if (cnt3==10) flag=1;
277           }else cnt3=0;
278    }
279    cnt2=0;
280    if (cnt3==15){
281              Mmc_Fat_Read(&character);
282              for (cnt1=i;cnt1<i+20;cnt1++){
283                     Mmc_Fat_Read(&character);
284                     if (character==0x0d) break;
285                     tekst_imena[cnt2++]=character; }
286   }
287            tekst_imena[cnt2++]='\0';
288
289  if (flag==1 && cnt3!=15) return 10;
290    return cnt3;
291
292  }
293
294  void M_Open_File_Append(char tekst[]) {
295
296   char vrijeme[15];
297   unsigned int cnt1,cnt2;
298         cnt2=0;
299    slave_filename[2]=output[2];
300    slave_filename[3]=output[3];
301     for(cnt1=5;cnt1<=15;cnt1++) result[cnt2++] = tekst[cnt1];
302
303
304
305     result[cnt2++]=(day / 10)   + 48;    // Print tens digit of day variable
306    result[cnt2++]= (day % 10)   + 48;    // Print oness digit of day variable
307     result[cnt2++]='.';
308    result[cnt2++]=(mn / 10) + 48;
```

```
309    result[cnt2++]= (mn % 10) + 48;
310    result[cnt2++]='.';
311    result[cnt2++]=  (year/10)  + 48;          // Print year vaiable + 8 (start from year
       2008)
312    result[cnt2++]=(year%10)  + 48;
313      result[cnt2++]=' ';
314
315    result[cnt2++]= (hr / 10)   + 48;
316    result[cnt2++]= (hr % 10)   + 48;
317      result[cnt2++]=':';
318    result[cnt2++]=(min1 / 10) + 48;
319    result[cnt2++]= (min1 % 10) + 48;
320       result[cnt2++]=':';
321    result[cnt2++]= (sec / 10) + 48;
322    result[cnt2++]=(sec % 10) + 48;
323
324      result[cnt2++]='\r';
325      result[cnt2++]='\n';
326      result[cnt2++]='\0';
327      Mmc_Fat_Assign(&slave_filename, 0xA0);
328
329     // Mmc_Fat_Set_File_Date(2005,6,21,10,35,0);
330      Mmc_Fat_Append();                                    // Prepare file for append
331
332     // *tekst++='\n';
333
334      Mmc_Fat_Write(result, strlen(result));   // Write data to assigned file
335  }
336
337  void Clock(){
338                Read_Time(&sec,&min1,&hr,&week_day,&day,&mn,&year);      // read time from
       RTC(DS1307)
339                 Transform_Time(&sec,&min1,&hr,&week_day,&day,&mn,&year); // format date
       and time
340                 Display_Time(sec, min1, hr, week_day, day, mn, year);    // prepare and
       display on LCD
341                 }
342
343
344
345
346
347
348
349
350
351  void    main()
352          {
353                    // set_time();
354     Lcd_Init();                // Initialize LCD
355     Lcd_Cmd(_LCD_CLEAR);       // Clear LCD display
356     Lcd_Cmd(_LCD_CURSOR_OFF);  // Turn cursor off
357
358           // Set Ports Per User-Specific Board Configuration...
359          /*
360            * starts ENC28J60 with :
361            * reset bit on RC0
362            * CS bit on RC1
363            * my MAC & IP address
364            * full duplex
365            */
366      // Cursor off
367
368
```

```
369             Lcd_Out(1,1,"Aquiring address");
370           SPI1_Init();
371           SPI_Ethernet_Init(myMacAddr, myIpAddr, SPI_Ethernet_FULLDUPLEX) ;
372           //SPI_Ethernet_doDHCP();
373           if (PINA.B0==0) SPI_Ethernet_initDHCP(5);
374
375           adres_f=0;
376           write_f=0;
377           brisi_f=0;
378           LAN_korisnici=0;
379           memcpy(myIpAddr, SPI_Ethernet_getIpAddress(), 4);
380           ByteToStr(myIpAddr[0], dyna) ;                // first IP address byte
381           dyna[3] = '.' ;
382           ByteToStr(myIpAddr[1], dyna + 4) ;            // second
383           dyna[7] = '.' ;
384           ByteToStr(myIpAddr[2], dyna + 8) ;            // third
385           dyna[11] = '.' ;
386           ByteToStr(myIpAddr[3], dyna + 12) ;           // fourth
387
388                                       // add separator
389           Lcd_Out(1,1,dyna);
390
391
392
393
394            UART1_Init(9600);
395            UART2_Init(9600);
396            Delay_ms(1000);
397
398
399
400
401
402                  cnt=0;
403            SPI_Ethernet_CS=1;
404            PORTG.B1=0;
405           if (Mmc_Fat_Init() == 0){
406            Lcd_Out(1,1,"SD initialized  ");
407            Delay_ms(500);
408           // M_Create_New_File();
409           }
410              PORTG.B1=1;
411            SPI_Ethernet_CS=0;
412            Init_Main();
413
414
415            Clock();
416
417           while (UART1_Data_Ready())uart_rd = UART1_Read();
418           while(1)
419                   {
420                                    if (adres_f){
421                                  // Lcd_Out(2,2,"IF");
422                                len=0;
423                                adres_f=0;
424
425                                    SPI_Ethernet_CS=1;
426                                     par_cnt=0;
427                                    //Delay_ms(200);
428                                    PORTG.B1=0;
429                                    size=0;
430                                    if (LAN_korisnici)a = Mmc_Fat_Assign(&filename, 0);
      else a=Mmc_Fat_Assign(&LAN_filename, 0);
431
```

```
                                    if (a==1){
                                          Mmc_Fat_Reset(&size);            // To read
      file, procedure returns size of file


                                          for (i = 1; i <= size; i++) {

                                            Mmc_Fat_Read(&character);
                                           //UART2_Write(character);

                                           if (i>stao_na){  dyna[len]=character ;
                                                            len++;

                                                            if (character==0x0d){
                                                            par_cnt++;
                                                             if (par_cnt==30) break; }


                                                             }

                                          }
                                    }

                                    PORTG.B1=1;
                                    SPI_Ethernet_CS=0;

                                  stao_na=i;
                                  if (len<10){
                                  stao_na=0;
                                  adres_f=0;
                                  LAN_korisnici=0;
                                                       dyna[len]='E';
                                                        len++;
                                                       dyna[len]='O';
                                                        len++;
                                                       dyna[len]='F';
                                                        len++;
                                                       dyna[len]=0x0D;
                                                        len++;

                                                        LongWordToStr(size,tnum);

                                                       dyna[len]=tnum[6];
                                                        len++;
                                                       dyna[len]=tnum[7];
                                                        len++;
                                                       dyna[len]=tnum[8];
                                                        len++;
                                                       dyna[len]=tnum[9];
                                                        len++;
                                  }
                                   dyna[len]=0;



                                }



                                if(write_f){
                                write_f=0;
```

```
495
496                                  SPI_Ethernet_CS=1;
497
498                                 //Delay_ms(200);
499                                 PORTG.B1=0;
500                                      Mmc_Fat_Assign(&filename, 0xa0);
501                                      Mmc_Fat_Append();
502                                       cnt2=0;
503                                       for(cnt1=4;cnt1<=40;cnt1++){
504
505                                        dyna[cnt2++] = getRequest[cnt1];
506                                        if   (getRequest[cnt1]== 0x0d) break;}
507                                        dyna[cnt2++]=0x0a;
508                                        dyna[cnt2++]=0x00;
509                                       Mmc_Fat_Write(dyna, strlen(dyna));
510
511
512
513
514
515                                                                    PORTG.B1=1;
516                                 SPI_Ethernet_CS=0;
517                             }
518
519
520
521                         if(brisi_f){
522                             brisi_f=0;
523
524                                   SPI_Ethernet_CS=1;
525
526                                 //Delay_ms(200);
527                                 PORTG.B1=0;
528                                      Mmc_Fat_Assign(&filename, 0xa0);
529                                      Mmc_Fat_Rewrite();
530                                       cnt2=0;
531                                       Mmc_Fat_Write("", 0);
532
533
534
535
536
537                                                                    PORTG.B1=1;
538                                 SPI_Ethernet_CS=0;
539                             }
540
541
542
543
544
545
546
547                     SPI_Ethernet_doPacket() ;   // process incoming Ethernet packets
548
549
550
551
552                     if (UART2_Data_Ready()) {      // If data is received,
553                          uart_rd = UART2_Read();
554                           UART1_Write(uart_rd);}
555                        if (UART1_Data_Ready()) {      // If data is received,
556                          uart_rd = UART1_Read();      //   read the received data,
557                          UART2_Write(uart_rd);
558                          if (uart_rd==0x0d){
```

```
559                         output[cnt++]='\0';
560                         cnt=0;
561
562                         Lcd_Out(1,1,output);
563                         if ( output[0]==0x53 && output[1]== 0x4c){      //ako su prva dva
    znaka SL
564                         SREG_I_bit=0;
565                        SPI_Ethernet_CS=1;
566
567                               PORTG.B1=0;
568                               temp=  M_Open_File_Read(output);
569                              if (temp == 15 || temp == 10){
570
571                             Delay_ms(300);
572                             while (UART1_Data_Ready())uart_rd = UART1_Read();
573
574                             UART1_Write_Text("+++");
575                              while (1)if (UART1_Data_Ready()) if (UART1_Read() ==0x0d)
    break;
576
577
578                              UART1_Write_Text("ATDN SL");
579                             UART1_Write(output[2]);
580                             UART1_Write(output[3]);
581                             UART1_Write(0x0D);
582                             while (1)if (UART1_Data_Ready()) {
583                                    uart_rd = UART1_Read();      //   read the received data,
584                                    UART2_Write(uart_rd);
585                                   if (uart_rd ==0x0d) break; }
586
587                              //Delay_ms(100);
588                              if (temp==15){
589                              UART1_Write_Text("OK ");
590                              UART1_Write_Text(tekst_imena);
591                              }else{
592                               UART1_Write_Text("PIN");
593                               }
594                              UART1_Write(0x0D);
595
596                              while (UART1_Data_Ready())uart_rd = UART1_Read();
597                              Clock();
598                               if (temp==15){
599
600
601
602                              M_Open_File_Append(output);
603
604                              }
605                              Delay_ms(500);
606
607                                     }
608
609                                      PORTG.B1=1;
610                                     // SPI_Ethernet_CS=0;
611
612
613                                      SREG_I_bit=1;
614
615                                    //  SPI_Ethernet_sendUDP(IpAddr, 11000, 11000, "NJi" ,
    3);
616                              }
617
618                         }
619                          else{
```

```
620                         output[cnt]=uart_rd;
621                         cnt++; }
622                                 //   and send data via UART
623                         }
624
625
626                         }
627             }
```