Dinko Oletić
0036420869



Fakultet elektrotehnike i računarstva, Sveučilišta u Zagrebu
Zavod za elektroničke sustave i obradbu informacija
Sveučilište u Zagrebu

# ZigBee i IP komunikacija

- ovaj dokument namijenjen je studentima elektrotehnike
- potrebno predznanje programskog jezika C i osnova ZigBee protokola
- opisuje izvedbu bežične senzorske mreže za primjenu u automatizaciji kućanstva

19. lipanj 2009

## Sažetak

Automatizacija u kućanstvu već je dugo prisutan trend i jedan od temeljnih elemenata u "izgradnji" inteligentne kuće. Bežično povezivanje senzora, aktuatora i njihovog središnjeg upravljačkog sustava donosi brojne prednosti. Ovdje je opisana bežična mreža temeljena na ZigBee protokolu. Namjena joj je integracija podsustava inteligentne kuće – alarmnog sustava, sustava za brigu o životinjama i Etherneta. Korišteni su ZigBee moduli Xbee tvrtke Digi montirani na Arduino pločicu korištenjem Xbee Shielda. Prerađen je postojeći xbee-arduino API. Napisani programski kod integriran je u ostale podsustave.

## Sadržaj



Ovaj seminarski rad je izrađen u okviru predmeta „Sustavi za praćenje i vođenje procesa“ na Zavodu za elektroničke sustave i obradbu informacija, Fakulteta elektrotehnike i računarstva, Sveučilišta u Zagrebu.

# 1. Uvod

Automatizacija u kućanstvu (engl. *home automation*) već je dugo prisutan trend i jedan od temeljnih elemenata u "izgradnji" inteligentne kuće. Protuprovalni (alarmni) sustav, sustav za brigu o kućnim ljubimcima, automatizirano upravljanje roletama ili rasvjetom samo su neki od primjera automatizacije u kućanstvu.

Postojanje komunikacijskih protokola poput protokola ZigBee i niska cijena komercijalnih digitalnih radiokomunikacijskih sustava na čipu (engl. *SoC*), omogućili su stvaranje čitave tržišne niše proizvoda namijenjenih izgradnji bežičnih senzorskih mreža (engl. *wireless sensor network*) za primjenu u kućanstvima i industriji. Namjena ovakvih mreža temeljenih na ZigBee protokolu je bežično prikupljanje podataka sa senzora i upravljanje aktuatorima.

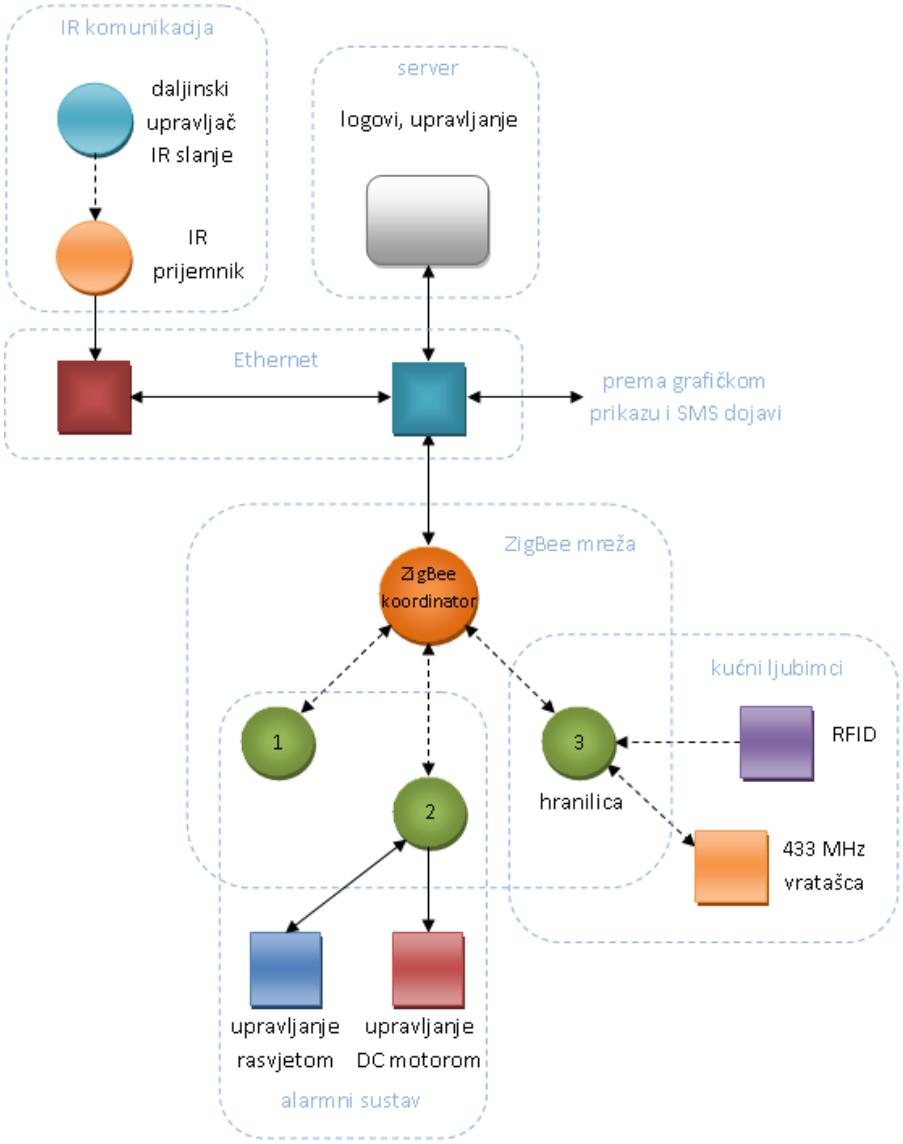
Bežično povezivanja senzora, aktuatora i njihovog središnjeg upravljačkog sustava donosi brojne prednosti: pojednostavljuje montažu, smanjuje broj kablova, smanjuje troškove i vrijeme ugradnje, iziskuje manje planiranja i omogućuje jednostavnu adaptaciju postojećih stambenih objekata (ne zahtjeva objekt u izgradnji).

U nastavku teksta opisana je bežična senzorska mreža temeljena na ZigBee protokolu namijenjena integraciji podsustava inteligentne kuće – čvorova alarmnog sustava i čvora sustava za brigu o kućnim ljubimcima. Svrha bežične mreže je povezati navedene udaljene čvorove preko koordinatorskog ZigBee čvora na kućni Ethernet.

# 2. Specifikacije i arhitektura sustava

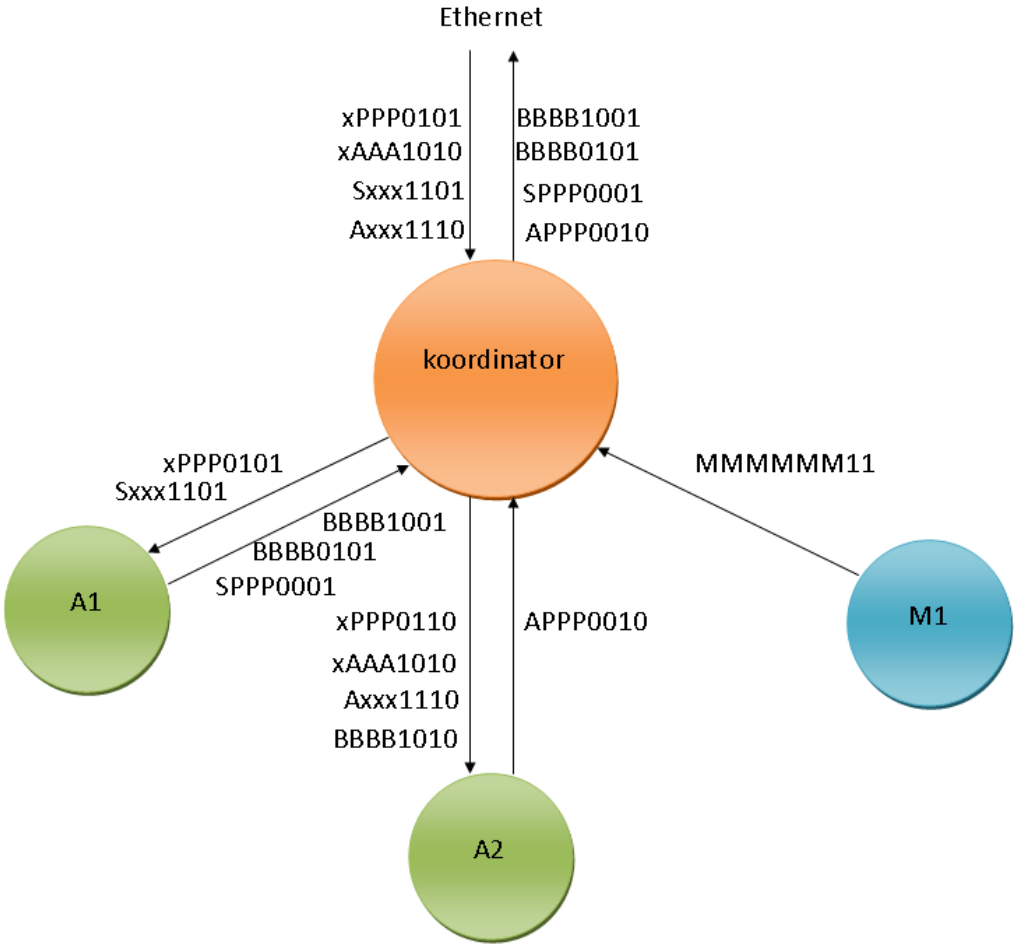
## 2.1. ZigBee mreža

Cilj ovog projekta je izgraditi bežičnu ZigBee mrežu koja se sastoji od jednog koordinatorskog ZigBee čvora i 3 udaljena krajnja ZigBee čvora kako je opisano u projektnom planu. ZigBee mreža u okviru projekta inteligentne kuće prikazana je slikom 1.




**Slika 1: Povezani sustavi u okviru projekta inteligentne kuće**

ZigBee je komunikacijski protokol namijenjen upotrebi u bežičnim senzorskim mrežama. Karakteriziraju ga niska propusnost (engl. *data rate*) od najviše 250 kbit/s, mala potrošnja energije i domet primjeren namjeni. Komunikacija se odvija na 2,4 GHz. Omogućuje jednostavnu izgradnju mreža zvjezdastih i isprepletenih (engl. *mesh*) topologija. Ugrađene sigurnosne mogućnosti čine ZigBee idealnim rješenjem za bežično upravljanje uređajima u WPAN mreži.




**Slika 2: Organizacija ZigBee mreže i formati poruka**

Prema slici 2, ZigBee mrežu čine komponente opisane u nastavku. ZigBee mreža služi samo za transport poruka. Za potpuno razumijevanje značenja pojedinih poruka, nužno je poznavanje protokola po kojem radi alarmni sustav [5] i načina rada sustava za brigu o životinjama [6]. Zadnja 2 bita svake poruke određuju odredište odnosno izvor poruke, već prema kontekstu: 01 → čvor A1, 10 → A2 i 11 → M1.

### 2.1.1. Koordinatorski čvor

Mrežom upravlja jedan koordinatorski ZigBee čvor. Ovaj čvor je središte zvjezdaste topologije ZigBee mreže. Brine o stvaranju bežične mreže, spajanju i autentifikaciji udaljenih čvorova A1, A2 i M1. Sav promet se odvija preko ovog čvora. Ujedno je povezan na Ethernet da bi mogao komunicirati s poslužiteljom i sustavom za slanje IR naredbi.

Koordinator preko Etherneta uvijek prima naredbe širine jednog okteta. Dobiva ih od sustava za upravljanje pomoću IR naredbi. Pomoću naredbe xPPP0101 se mijenja režim rada čvorova alarmnog sustava A1 i A2. Režim rada mijenja se uvijek na oba čvora istovremeno.

xAAA1010 pokreće/zaustavlja rad aktuatora (dizanje/spuštanje roleta) priključenog na čvor A2.

Naredba može biti i zahtjev za dostavom trenutnog stanja senzora/aktuatora alarmnog sustava (naredbe Sxxx1101 i Axxx1110), ukoliko ju poslužitelj (ikad) pošalje. Kojem čvoru je ta naredba namijenjena poznato je na temelju zadnja dva bita naredbe (01 = A1, 10 = A2).

Koordinator može od krajnjih čvorova ZigBee mreže primati 1 ili 3 okteta.

Od čvora M1 koordinator uvijek prima 1 oktet – MMMMMM11. Taj podatak predstavlja alarm sustava za brigu o kućnim ljubimcima i šalje se preko Etherneta na poslužitelj.

Od čvorova A1 i A2 koordinator može primati podatak o stanju senzora/aktuatora. Širina ovog podatka je 3 okteta. Prvi oktet (SPPP0001 ili APPP0010) pritom služi za identificiranje čvora i senzora/aktuatora koji šalje svoje stanje, dok preostala 2 okteta prenose digitalnu vrijednost očitanu na senzoru/aktuatoru. Ovaj trobajtni podatak šalje se preko Etherneta na poslužitelj.

Jednooktetni podatak koji koordinator može primiti od čvora A1 je "kućni" ili "tihi alarm". U slučaju "tihog alarma" primljeni podatak se prosljeđuje samo na poslužitelj (BBBB0101), dok se kod "kućnog alarma" podatak šalje i na poslužitelj i do aktuatorskog čvora A2 (BBBB1001, tj. BBBB1010 nakon prolaska kroz koordinator).

### 2.1.2. A1 - senzorski čvor alarmnog sustava

Čvor broj 1 prema slici 1, odnosno A1 prema slici 2 je čvor alarmnog sustava. Služi otkrivanju prisutnosti osoba u hodniku. Ovo je isključivo senzorski čvor. Na njemu je spojen kapacitivni senzor. Prvi oktet svih poruka koje prima i šalje završava s 01 i to služi za identifikaciju čvora.

### 2.1.3. A2 - aktuatorski čvor alarmnog sustava

Čvor broj 2, odnosno A2 je aktuatorski čvor alarmnog sustava. Služi lokalnom oglašavanju (tzv. "kućni alarm") i ima spojen aktuator za automatsko zatvaranje prozora/roleta pomoću DC motora. Zadnja 2 bita svake poruke identificiraju ovaj čvor kao 10.

### 2.1.4. M1 - čvor sustava za brigu o kućnim ljubimcima

Ovo je senzorski čvor koji je dio sustava za brigu o kućnim ljubimcima. Donosi odluku o prisutnosti kućnog ljubimca i upravlja hranilicom. Ovaj čvor može samo slati koordinatoru (ne prima podatke od koordinatora) i ne ovisi o protokolu alarmnog sustava. Periodički šalje alarm kad nešto nije u redu s kućnim ljubimcem. Identifikator ovog čvora čine zadnja dva bita svake poruke koja su uvijek 11.

# 3. Korišteno sklopovlje

## 3.1. Digi Xbee Series 2

ZigBee mreža fizički je realizirana korištenjem ZigBee modula Xbee. Korištena su 4 modula novije generacije - Series 2 (ZB Pro/ZNet).

Moduli rade na 2,4 GHz, omogućavaju najviše brzine prijenosa od 250 kbit/s (sirovi podaci). Najveća potrošnja definirana je kod primanja i iznosi 38 mA. U stanju mirovanja (engl. *sleep*), potrošnja pada do ispod 1 uA. Izlazna snaga odašiljača je mala i iznosi najviše 2 mW (u tzv. *boost* modu). Antena je keramička. Izvedena je na čipu. Domet bi trebao biti do 120 m na otvorenom, odnosno 40 m u zatvorenom prostoru.

## 3.2. Arduino Xbee Shield

Xbee modul se na Arduino montira koristeći pločicu za proširenja Xbee Shield (slika 3).


**Slika 3: XBee Shield s utaknutim XBee m**

Mikrokontroler s Arduino pločice i Xbee Shield komuniciraju serijskim sučeljem preko linija RX i TX. Stoga je, prilikom korištenja Xbee Shielda, gledano sa strane PC računala, moguće jedino čitanje (tj., kad se šalju podaci od Arduinovog mikrokontrolera prema XBee Shieldu s namjerom da

budu bežično poslani, šalju se i na PC računalo). [3] S druge strane, **nije** moguće čitanje prilikom primanja podataka (smjer podataka XBee → Arduino). Ovo otežava pronalaženje i uklanjanje pogrešaka.

Prilikom programiranja Arduino pločica s montiranim XBee Shieldovima, potrebno je oba kratkospojnika na XBee Shieldovima (slika 3) premjestiti iz standardnog radnog položaja XBEE u položaj USB.

Napajanje se na XBee Shield dovodi preko 3x2 konektora na središnjem dijelu s donje strane XBee Shielda. Pinovi koji ulaze u ovaj konektor na Arduino pločici označeni su s ICSP. Tipka RESET na Xbee Shield pločici (vidi sliku 3) resetira mikrokontroler na glavnoj Arduino pločici, a ne XBee modul (što bi bilo logičnije).

Bitno je napomenuti da XBee Shield **ne** koristi nikakve dodatne pinove osim RX i TX, što omogućuje spajanje brojnih ulazno-izlaznih jedinica.

Pri korištenju XBee Shielda u kombinaciji s Ethernet Shieldom javlja se problem utoliko što treba XBee Shieldu dovesti napajanje, a Ethernet Shield fizički smeta. Rješenje je da se žicama povežu istovrsni pinovi 2 i 6 na Arduino pločici i XBee Shieldu. Pritom je dobro konzultirati [8]. Po potrebi prospojiti i reset (pin 5). Također, valja napomenuti da Ethernet Shield koristi većinu ulazno-izlaznih pinova pa uz istovremeno priključene Ethernet i XBee Shieldove nije moguće spojiti nikakvo dodatno periferno sklopovlje [1].

Nedostatak XBee Shield pločice je što nema izvedenu tipku za reset Xbee modula. Stoga pri reprogramiranju modula, kad to program X-CTU opisan u nastavku zahtjeva, žicom treba na nekoliko sekundi kratkospojiti pin 5 Xbee modula i masu (GND).

# 4. Programska podrška

## 4.1. Načini programiranja Xbee modula

Da bi više modula moglo zajedno raditi u istoj mreži, potrebno je na ispravan način konfigurirati XBee module. Moduli se konfiguriraju preko USB/UART sučelja izdavanjem AT naredbi. Najjednostavniji način (i jedini koji stvarno radi) nije korištenje terminala, već Digijevog programa X-CTU.

Navedeni program služi mijenjanju bitnih postavki konfiguracije, poput određivanja uloga (koordinator, router, krajnji čvor), PANID-a, filtra kanala, najveće izlazne snage, podešavanje perioda mirovanja modula, sigurnosnih postavki mreže, brzine serijskog sučelja itd.

Posebno je bitno značenje registara DH, DL, SH i SL. SH i SL određuju jedinstveni 64-bitni identifikator koji omogućuje diferenciranje čvorova u mreži. DH i DL su isto 64-bitni i određuju odredište poruke kod svakog slanja. Kombinacija DH = 0x0 i DL = 0xFFFF ima posebno značenje i označava broadcast – slanje svim čvorovima u mreži. DH = 0x0 i DL = 0x0 je također rezervirana kombinacija i znači slanje poruke koordinatoru mreže.

Konfiguracijom se određuje i mod rada čvorova – automatizirani AT mod gdje se čvorove programira slanjem AT naredbi ili mnogo prilagodljiviji API mod koji omogućuje programiranje čvorova preko API-ja u programskom jeziku C ili nekom sličnom.

### 4.1.1. AT način rada

Ovo je jednostavniji način rada. Radi savršeno za jednostavne tzv. *point-to-point* veze. U slučaju uspostave zvjezdaste mreže s više od 2 čvora gdje koordinator mora slati svima (engl. *broadcast*), mreža se usporava do neupotrebljivosti (empirijski izmjerena propusnost - 1 podatak u 2 sekunde). Tome su vjerojatno krive neoptimizirane Arduinove funkcije za rad sa serijskim sučeljem. Ovaj problem bio je motivacija da se komunikacija između čvorova ostvari u API načinu rada (vidi sljedeće poglavlje).

U AT modu ne postoji poseban API za rad s XBee modulom. Kad je spojen Xbee Shield, jednostavnim slanjem podataka na serijsko sučelje korištenjem standardnih funkcija poput Serial.print(), proizvoljan broj okteta šalje se i bežično preko ZigBeeja i preko standardnog FTDI sklopa na računalo. Problem nastaje kod primanja sa serijskog sučelja jer ugrađena funkcija Serial.Read() može čitati samo jedan (prvi) oktet.

### 4.1.2. API način rada

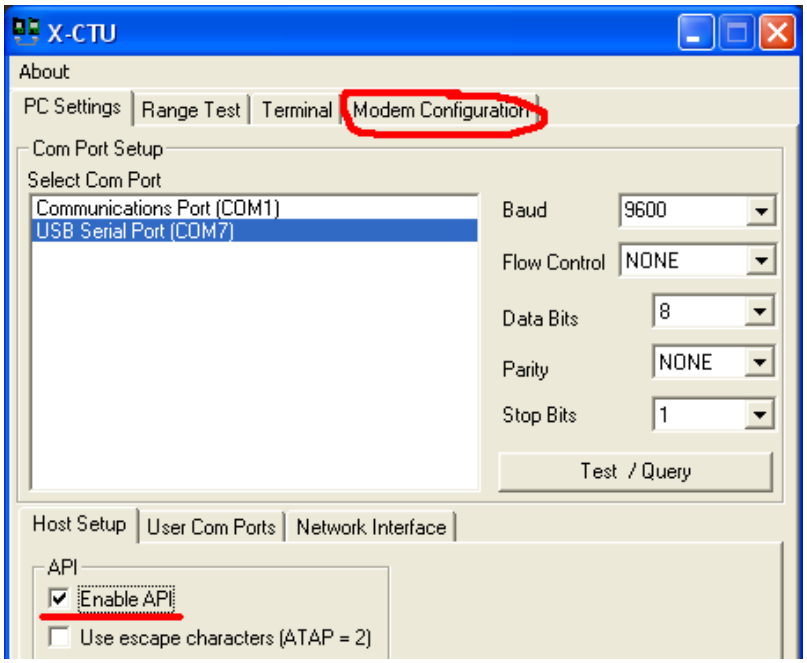
U API načinu rada, programer ima na raspolaganju čitav niz funkcija za rad sa ZigBee modulom (slanje, primanje, konfiguriranje tokom rada itd.). To znači veću prilagodljivost. Zbog problema s AT načinom rada iznesenih u prošlom poglavlju, ovaj projekt je izveden korištenjem API-ja xbee-arduino dostupnog na Google Codeu. [4]

U sljedećem je poglavlju opisano korak po korak kako prije samog programiranja pravilno podesiti XBee modul za rad u API načinu rada pomoću programa X-CTU.

### 4.1.3. Upute za postavljanje API načina korištenjem programa X-CTU

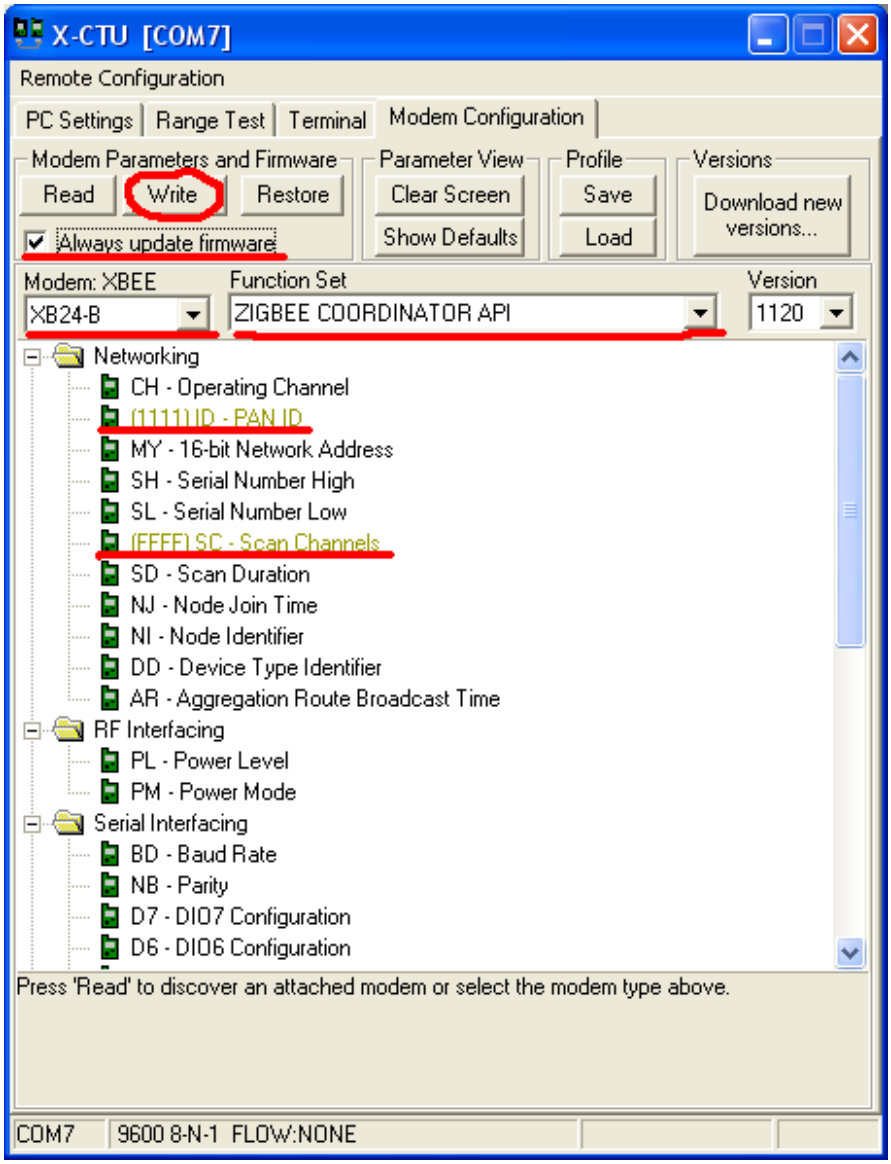
**Koordinatorski čvor:**

- Postaviti mikrokontroler na Arduino pločici u reset kratkim spajanjem pinova RESET i GND.
- Montirati XBee Shield na Arduino pločicu.
- Postaviti kratkospojnike na XBee Shieldu u položaj USB.
- Povezati pločicu na USB priključak PC računala.
- Pokrenuti X-CTU.
- Odabrati COM priključak na kojem je pronađen XBee modul, uključiti svojstvo Enable API i odabrati Modem Configuration.




**Slika 4: X-CTU**

- Odabrati sve značajke kao na slici 5 i pritisnuti Write.

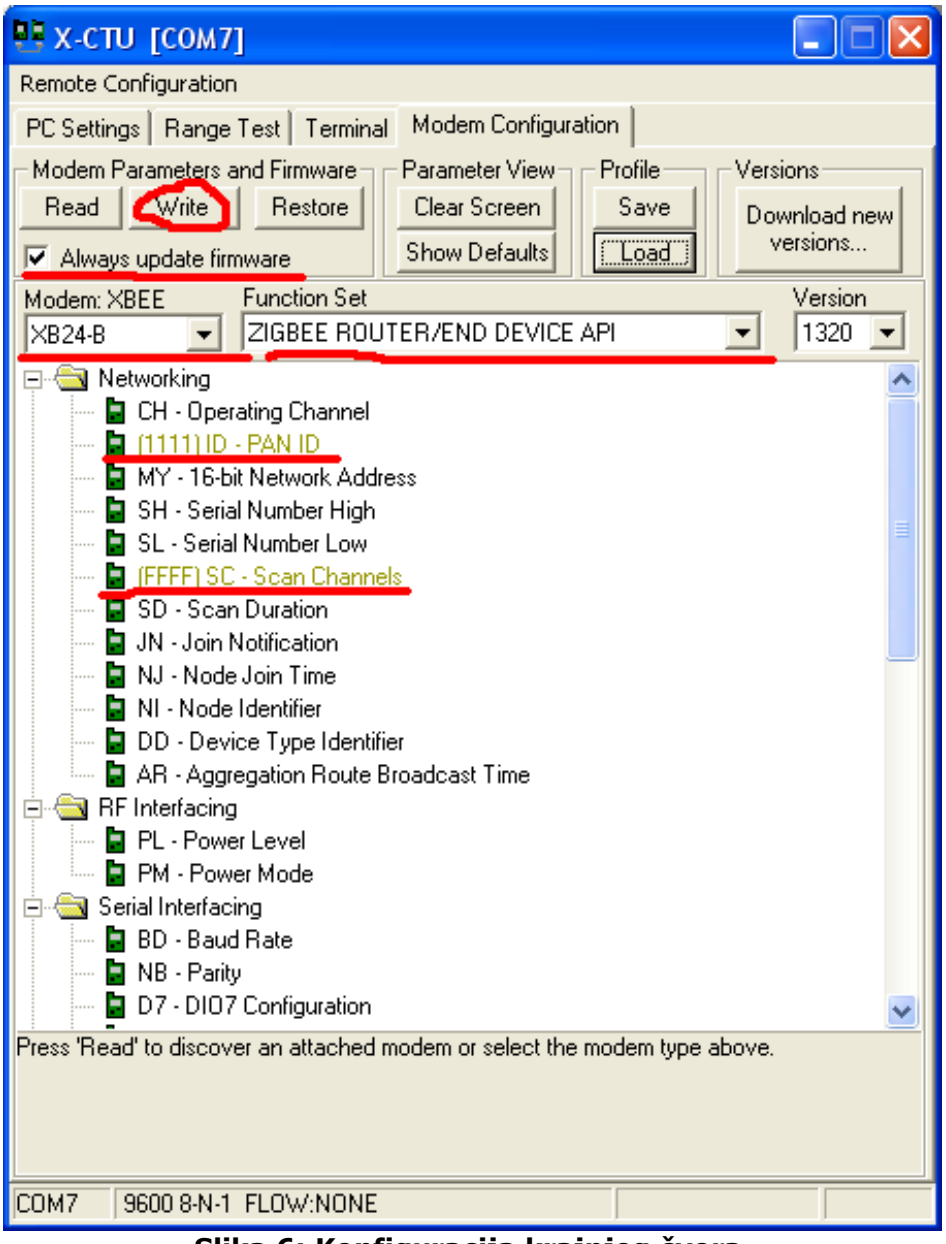



**Slika 5: Konfiguracija koordinatora.**

- Po potrebi resetirati XBee modul ručnim kratkim spajanjem pina 5 **XBee modula** i mase.
- Izvaditi mikrokontroler iz reseta i postaviti kratkospojnike XBee Shielda u položaj XBEE.

**Krajnji čvor:**

- Ponoviti sve prethodno napisano, jedino u programu X-CTU odabrati značajke kao na slici 6.




**Slika 6: Konfiguracija krajnjeg čvora.**

## 4.2. xbee-arduino API

Na Googleovom servisu Google Code 29.3.2009. objavljena je biblioteka *xbee-arduino*. Omogućuje programiranje Xbee modula sa Xbee Shieldovima za Arduino u API načinu rada [4], što trenutno nije podržano na službenoj stranici Arduina. Biblioteka se trenutno nalazi u verziji 0.1.2 te podržava Series 1 (802.15.4) i Series 2 (ZB Pro/ZNet) module. Sav programski kod napisan u sklopu ovog projekta nastao je korištenjem te biblioteke.

Biblioteka se instalira na način da se nakon otpakiravanja arhive xbee-arduino-0.1.2.zip, mapa Xbee prekopira u mapu \hardware\libraries koja se nalazi u korijenskoj mapi u kojoj je instalirano razvojno okruženje za Arduino.

U primjerima koji dolaze s bibliotekom dani su odsječci koda za slanje i primanje. Budući da u ovom projektu ZigBee mreža služi za povezivanje podsustava više studenata, bilo je potrebno enkapsulirati dani kod u jednostavne funkcije za slanje i primanje preko ZigBeeja tako da svatko može na jednostavan način upotrijebiti kod.

### 4.2.1. Inicijalizacija

Za inicijalizaciju API-ja, potrebno je uključiti biblioteku Xbee.h i stvoriti globalne objekte prema sljedećem odsječku koda:

```
#include <XBee.h>
#define ZIGBEE_MAXBYTES 10    // najveci broj okteta u jednom paketu
// create the XBee object
XBee xbee = XBee();

XBeeResponse response = XBeeResponse();
// create reusable response objects for responses we expect to handle
ZBRxResponse rx = ZBRxResponse();
ModemStatusResponse msr = ModemStatusResponse();

uint8_t payload[ZIGBEE_MAXBYTES] = { 0 };

// SH + SL Address of receiving XBee
XBeeAddress64 addr64 = XBeeAddress64(0x0, 0xFFFF);    // odrediste - svi
cvorovi (broadcast)
ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));
ZBTxStatusResponse txStatus = ZBTxStatusResponse();
```

Najbitnije je uočiti da je u gornjem kodu prilikom stvaranja objekta XBeeAddress64 potrebno odrediti sadržaje registara DH i DL Xbee modula, koji definiraju odredište svakog slanja. Očito se u gornjem kodu čvor postavlja da šalje svim čvorovima (*broadcast*).

Također, u funkciji setup() potrebno je pokrenuti ZigBee komunikaciju:

```
xbee.begin(9600);
```

Nakon toga mogu se koristiti funkcije za slanje i primanje podataka.

### 4.2.2. Slanje podataka

Funkcija za slanje nazvana je ZigBeeSend(). Kao argumente prima pokazivač na polje koje sadrži podatke za slanje i veličinu tog polja. U slučaju uspješnog slanja, funkcija vraća 0. U nastavku slijedi njena definicija.

```
char ZigBeeSend(uint8_t *outData, char outDataSize)
{
    char returnVal = 0; char i;
    // punjenje izlaznog polja
    if(outDataSize <= ZIGBEE_MAXBYTES)
    {
        for (i = 0; i < outDataSize; i++)
        {
            payload[i] = *(outData + i);
        }
    }
    else   // u slucaju da korisnik salje vise od maks. dopustenog broja
okteta
    {
        for (i = 0; i < outDataSize; i++)
        {
            payload[i] = *(outData + i);
        }
        returnVal = 1;
    }
    xbee.send(zbTx);
    // flash TX indicator
    flashLed(statusLed, 1, 100);
    // after sending a tx request, we expect a status response
    // wait up to half second for the status response
    if (xbee.readPacket(500))
    {
        // got a response!
        // should be a znet tx status
      if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE)
        {
          xbee.getResponse().getZBTxStatusResponse(txStatus);

          // get the delivery status, the fifth byte
            if (txStatus.getDeliveryStatus() == SUCCESS)
            {
              // success.  time to celebrate
              flashLed(statusLed, 5, 50);
            }
            else
            {
              // the remote XBee did not receive our packet. is it powered on?
              flashLed(errorLed, 3, 500);
              returnVal = 2;
```

```
            }
        }
    }
    else
    {
      // local XBee did not provide a timely TX Status Response -- should not
happen
      flashLed(errorLed, 2, 50);
      returnVal = 3;
    }
    return returnVal;
}
```

### 4.2.3. Primanje podataka

Funkcija za slanje nazvana je ZigBeeReceive(). Kao argumente prima pokazivač na polje u koje će se spremiti ulazni podatci i veličinu tog polja. U slučaju uspješnog primanja svježeg podatka, funkcija vraća 1. Tipično se poziva u if strukturi. Provjeravajući je li funkcija ZigBeeReceive vratila 1, zapravo se provjerava je li preko ZigBeeja stigao novi podatak. Naime, funkcija ZigBeeReceive se u glavnoj petlji programa poziva iterativno, no samo kad preko ZigBeeja stigne okvir (engl. *frame*) s novim podacima, ona će pročitati dio okvira koji sadrži korisničke podatke i vratiti 1, što je signal ostatku programa da se s primljenim podacima može/mora nešto učiniti. Funkcija bi se uz malo dodatnog istraživanja API-ja mogla doraditi na način da u slučaju pojave novog paketa umjesto vrijednosti 1 vraća broj korisničkih okteta u primljenom paketu. U tu svrhu trebalo bi proučiti razred ZBRxResponse. Slijedi definicija funkcije ZigBeeReceive().

```
char ZigBeeReceive(uint8_t *inData, char inDataSize)
{
    char returnVal = 0; char i
    xbee.readPacket()
    if (xbee.getResponse().isAvailable())
    {
        // got something
        if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
        {
            // got a zb rx packet
            // now fill our zb rx class
            xbee.getResponse().getZBRxResponse(rx);
            if (rx.getOption() == ZB_PACKET_ACKNOWLEDGED)
            {
                // the sender got an ACK
                flashLed(statusLed, 1, 10);
            }
            else
            {
                // we got it (obviously) but sender didn't get an ACK
                flashLed(errorLed, 2, 20);
                returnVal = 3;
            }
            // get the data
            if (inDataSize <= ZIGBEE_MAXBYTES)
            {
```

```
                for (i = 0; i < inDataSize; i++)
                {
                    *(inData + i) = rx.getData(i);
                }
                returnVal = 1;
            }
            else   // ako je korisnik zadao preveliko polje za spremanje
ulaznih podataka
            {
                for (i = 0; i < ZIGBEE_MAXBYTES; i++)
                {
                    *(inData + i) = rx.getData(i);
                }
                returnVal = 2;
            }
        }
        else if (xbee.getResponse().getApiId() == MODEM_STATUS_RESPONSE)
        {
            xbee.getResponse().getModemStatusResponse(msr);
            // the local XBee sends this response on certain events, like
association/dissociation
            if (msr.getStatus() == ASSOCIATED)
            {
              // yay this is great.  flash led
              flashLed(statusLed, 10, 10);
            }
            else if (msr.getStatus() == DISASSOCIATED)
            {
                // this is awful.. flash led to show our discontent
                flashLed(errorLed, 10, 10);
                returnVal = 4;
            }
            else
            {
                // another status
                flashLed(statusLed, 5, 10);
            }
        }
        else
        {
           // not something we were expecting
            flashLed(errorLed, 1, 25);
           returnVal = 5;
        }
    }
    return returnVal;
}
```

## 4.3. Integracija koda – koordinatorski čvor

Funkcije za slanje i primanje opisane u prethodnim odlomcima proslijeđene su ostalim članovima projektnog tima koji su se bavili realizacijom čvorova A1 i A2 alarmnog sustava te čvora M1 sustava za brigu o životinjama. Tamo su navedene funkcije korištene za komunikaciju s koordinatorskim čvorom. Osobno sam se bavio programiranjem koordinatorskog čvora. U nastavku slijedi opis njegovog programskog koda.

U programski kod koordinatorskog čvora dodane su Ethernet funkcije Salji_TCP() i Primi_TCP() dokumentirane u [7]. Također, dodan je sav kod potreban za inicijalizaciju Etherneta. Detalji se mogu vidjeti u kompletnom programskom kodu. Ukratko, prototip funkcije Salji_TCP() glasi:

```
void Salji_TCP(byte send_cvor, char *send_podatak)
```

Funkcija na odredišni Ethernet čvor specificiran varijablom send_cvor šalje polje čija se adresa prosljeđuje preko pokazivača send_podatak.

Funkcija Primi_TCP() prima kao argument pokazivač na polje u koje će se spremiti polje ulaznih (primljenih) podataka. U slučaju uspješnog prijema (ukoliko je stigao novi podatak), funkcija vraća 1. Prototip funkcije je:

```
int Primi_TCP(char *receive_data)
```

Glavna funkcija loop() ostvaruje vezu između Etherneta i krajnjih čvorova A1, A2 i M1 te vezu između čvorova A1 i A2, kako je definirano protokolom rada alarmnog sustava [5] i protokolom rada sustava za brigu o životinjama [6]. Formati podataka koji se razmjenjuju opisani su u potpoglavljima 2.1.1 do 2.1.4. i slikom 2.

Za razmjenu podataka između Ethernet i ZigBee dijela koriste se sljedeća polja:

```
uint8_t myOutZigBeeData = 0;
uint8_t myInZigBeeData[3] = {0, 0, 0};

char myInEthernetData = 0;
char myOutEthernetData[3] = {0,0,0};
char myOutEthernetByte = 0;
```

Funkcionalnost koordinatora, kao što je već bilo rečeno, opisana je u poglavlju 2.1.1. Slijedi programski kod iz funkcije loop() koji ju realizira:

```
if(Primi_TCP(&myInEthernetData) == 1)
    {
        //Salji_TCP(3, &myInEthernetData);    // samo provjera jel prima
        myOutZigBeeData = myInEthernetData;

        // posalji konfiguraciju alarmnog sustava na cvor A1
        // promjena stanja aktuatora na A2
        // zahtjev za stanjem senzora/aktuatora cvorovima A1 ili A2
```

```
        // potvrda provjere identiteta (OK)
        ZigBeeSend(&myOutZigBeeData, sizeof(myOutZigBeeData));

        // posalji konfiguraciju alarmnog sustava jos na cvor A2
        if ((myOutZigBeeData & MASK_ZZ) == 4)
        {
            myOutZigBeeData = myOutZigBeeData ^ MASK_II;  // II: 01 -> 10
            ZigBeeSend(&myOutZigBeeData, sizeof(myOutZigBeeData));
        }
    }

    // je li stigla nova poruka od ZigBee krajnjih cvorova
    if (ZigBeeReceive(myInZigBeeData, sizeof(myInZigBeeData)) == 1)
    {
        // je li podatak trobajtni - stanje senzora/aktuatora s cvorova A1
ili A2?
        if ( ( ((myInZigBeeData[0] & MASK_II) == 1) || ((myInZigBeeData[0] &
MASK_II) == 2) ) && ((myInZigBeeData[0] & MASK_RR) == 0) )
        {
            myOutEthernetData[0] = myInZigBeeData[0];
            myOutEthernetData[1] = myInZigBeeData[1];
            myOutEthernetData[2] = myInZigBeeData[2];

            // salji stanje senzora/aktuatora na poslužitelj
            Salji_TCP(1, myOutEthernetData);
            flashLed(dataLed, myOutEthernetData[0] & MASK_II, 1000);
        }
        else
        {
            myOutEthernetByte = myInZigBeeData[0];

            // salji na poslužitelj tihi alarm, kucni alarm, zahtjev za
verifikacijom ili alarm kucnog ljubimca
            Salji_TCP(1, &myOutEthernetByte);
            flashLed(dataLed, myOutEthernetByte & MASK_II, 100);

            // tihi alarm - proslijedi cvoru A2
            if ( ((myInZigBeeData[0] & MASK_II) == 1) && ((myInZigBeeData[0]
& MASK_RR) == 8) )
            {
                myOutZigBeeData = myInZigBeeData[0];
             myOutZigBeeData = myOutZigBeeData ^ MASK_II;  // II: 01 -> 10

                ZigBeeSend(&myOutZigBeeData, sizeof(myOutZigBeeData));
            }
        }
    }
```

# 5. Zaključak

U odnosu na specifikacije iz projektnog plana, napravljeno je nekoliko izmjena. Tako sustav za brigu o kućnim ljubimcima komunicira s koordinatorskim čvorom samo jednosmjerno (krajnji čvor → koordinator). U alarmnom sustavu nisu izvedene sve mogućnosti pa neke vrste poruka nikad neće biti poslane/primljene – npr. provjera identiteta ukućana. Također, koordinatorski čvor ne komunicira izravno s računalnim čvorom za SMS dojavu, već sve podatke da je došlo do alarma šalje u središnji poslužitelj, odakle se može doći do informacije o alarmu.

Sustav je ispitan po dijelovima i u cjelini. Utvrđeno je da komunikacija između čvorova ZigBee mreže radi. Koordinator omogućuje spajanje triju krajnjih čvorova u mrežu (A1, A2 i M1). Razmjena poruka prema dogovorenom protokolu između čvora A1 alarmnog sustava, koordinatora i čvora A2 radi u oba smjera. Također, komunikacija između čvora M1 za brigu o kućnim ljubimcima i ZigBee koordinatora funkcionira. Prilikom slanja/primanja poruka nije uočeno nikakvo kašnjenje niti gubljenje paketa.

Iako prilikom ispitivanja pojedinačnih dijelova koda nije bilo problema, kod integracije komponenata u cjeloviti sustav inteligentne kuće uočen je problem u komunikaciji između ZigBee koordinatora i Etherneta (potpuna nemogućnost komunikacije između ova dva sustava kad se na istoj Arduino pločici izvodi i programski kod ZigBee koordinatora i programski kod Ethernet čvora). Mogući izvor ovog problema je u bibliotekama korištenim za ZigBee. Naime, ove biblioteke [4] ne nalaze se još na popisu standardnih Arduino biblioteka i trenutno su u ranoj razvojnoj fazi (verzija 0.1.2). Također, drugi članovi razvojnog tima koji su radili samo s bibliotekama za Ethernet (bez ZigBeeja), žalili su se na sporost i gubljenje paketa.

Obzirom na nesrazmjer vremena dostupnog za rad na ovom projektu i složenosti cjelokupnog sustava inteligentne kuće, ponestalo je vremena za temeljito ispitivanje zajedničkog simultanog rada svih komponenata. Rad na ovom projektu trebalo bi nastaviti ponajprije u smjeru detaljnog ispitivanja i dorađivanja – primjerice, funkciju za primanje moglo bi se preraditi da vraća broj primljenih okteta. U tu svrhu potrebno je proučiti razred ZBRxResponse i ostale vezane razrede unutar biblioteke xbee-arduino. Tek nakon dorade trebalo bi krenuti u smjeru dodavanja mogućnosti koje trenutno nedostaju (npr. neke funkcije alarmnog sustava).

# 6. Literatura


[1] Arduino Ethernet Shield Schematic. URL: http://arduino.cc/en/uploads/Main/arduino-ethernet-shield-schematic.pdf (2009-05-27)

[2] Arduino – Language Reference. URL: http://arduino.cc/en/Reference/HomePage (2009-05-22)

[3] Arduino – Xbee Shield. URL: http://arduino.cc/en/Main/ArduinoXbeeShield (2009-05-22)

[4] google Code – xbee-arduino. URL: http://code.google.com/p/xbee-arduino/ (2009-06-01)

[5] Mihaldinec, Hrvoje. Čuvanje kuće. Zagreb: FER, 2009.

[6] Pekarić, Edo. Briga o kućnim ljubimcima. Zagreb: FER, 2009.

[7] Škarica, Anita. Ethernet i IP komunikacija. Zagreb: FER, 2009.

[8] Xbee Shield Schematic. URL: http://www.arduino.cc/en/uploads/Main/XbeeShieldSchematic.pdf (2009-05-27)

[9] XBee & XBee-PRO ZB ZigBee PRO RF Modules. URL: http://www.digi.com/products/wireless/zigbee-mesh/xbee-zb-module.jsp#specs (2009-06-01)

# 7. Pojmovnik

| Pojam | Kratko objašnjenje | Više informacija potražite na |
|---|---|---|
| ZigBee | Komunikacijski protokol korišten u bežičnim senzorskim mrežama. Temelji se na nadograđenoj specifikaciji IEEE 802.15.4., a protokol održava skupina tvrtki – ZigBee Alliance | http://www.zigbee.org/ |
| WSN | engl. Wireless Sensor Network – bežična senzorska mreža | http://en.wikipedia.org/wiki/WSN |
| WPAN | Bežična mreža kratkog dometa. Primjeri - IrDA, Bluetooth, UWB, Z-Wave and ZigBee. | http://en.wikipedia.org/wiki/Wireless_personal_area_network#Wireless |
| Ethernet | Tehnologija za žično povezivanje računala u lokalnu mrežu temeljena na standardu IEEE 802.3 | http://en.wikipedia.org/wiki/Ethernet |
| Point-to-point topology | Najjednostavnije povezivanje dvaju mrežnih čvorova izravnom vezom | http://en.wikipedia.org/wiki/Network_topology#Point-to-point |
| Star topology | Zvjezdasta topologija mreže. Svi čvorovi vezani su na središnji čvor kroz koji prolazi sav promet. | http://en.wikipedia.org/wiki/Star_topology |
| Mesh network | Isprepletena struktura mrežnih čvorova. Veze su gušće, a mreža otpornija na smetnje. | http://en.wikipedia.org/wiki/Mesh_network |