Hrvoje Mihaldinec
0036421385

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

Čuvanje kuće

- Bežični alarmni sustavi
- Potrebno predznanje C programskog jezika
- Arduino razvojni sustav

15. lipanj

Sažetak

U ovom radu nastojala se ostvariti praktična primjena izvedbe alarmnog sustava s limitiranim opsegom odlučivanja te dakako pri tome osigurati njegovu univerzalnost te jednostavnost primjene. Koristeći bežični prijenos podataka, lako dostupni razvojni sustav i svojevrsno inteligentno odlučivanje nastojalo se zamijeniti današnju utemeljenu ideju alarmnog sustava.

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

Prva asocijacija naslova teme projektnog zadatka podrazumijeva opću sigurnost kuće, specifično protuprovalnu. Prateći ideju u tom smjeru, cilj je bio ostvariti što više univerzalniju izvedbu modula ali pritom ne ugušiti sposobnost prilagodbe različitim zadaćama. Jednostavnost priključivanja različitih senzora i aktuatora je također bila jedna od niti vodilja. Komunikacija bi se obavljala korištenjem ZigBee komunikacijskih modula te razvijenog protokola, kasnije opisanog u tekstu. Broj takvih bežičnih čvorova bi ovisio o veličini kuće, stana ili ureda te o broju kritičnih ulaza. Primarna ideja je postaviti po jedan čvor u svaku sobu koji bi detektirao prisutnost i aktivnost unutar jedne prostorije te posljedično, zaviseći o situaciji, reagirao. Postoje dvije posebnosti u izvedbi ovog projekta. Prva je činjenica da su moduli bežični s nezavisnim izvorima napajanja. Zajedno s prvom crtom obrane, senzorima, druga, bežičnost, stvara poveću otpornost na bilo kakve neželjene promjene treće strane. Dodatno, sustav uzima u obzir načine rada kuće te posljedično prilagođava svoje reakcije njima. Drugim riječima primjenjuje malu količinu inteligencije prilikom odlučivanja o alarmiranju ukućana.

2. Vrste alarma

Unutar projekta razvila se potreba za različitim vrstama alarmiranja zbog činjenice da alarmni sustav nije bio dizajniran kao običan prekidni sklop. Cilj sustava nije bio, nakon uključivanja, samo detektirati nasilni ulazak u kuću te posljedično tome uključiti alarm. Postoje tri vrste alarma koji se koriste pri različitim načinima rada kuće. Načini će biti objašnjeni u sljedećem poglavlju dok su alarmi:

Tihi alarm – prenosi informaciju centralnom sustavu da je došlo do događaja na jednom od čvorova, odnosno da se jedan od senzora aktivirao. Specifično je za ovu vrstu alarma da se ne uključuje kućni alarm. Razlog tome je dvojaki. Ukoliko dođe do nasilnog ulaska zadnje je što su sustav može učiniti jest da alarmira provalnika. Bolje je da bude uvjeren kako je uspio onesposobiti alarmni sustav te tiho alarmirati ukućane izvan kuće ili pak policiju. Drugo, ova vrsta alarma u nekim načinima rada kuće može poslužiti kao statusni alarm. Putem ovog alarma centralni sustav može primjerice bilježiti prisutnost unutar jedne prostorije te ne prosljeđivati alarm dalje.

Kućni alarm – kao što i samo ime govori primjenom ovog alarma uključuje se kućni alarm. Cilj je, ukoliko je kuća puna, obavijestiti pravovremeno ukućane o nasilnom ulasku ili nekom drugom nastalom problemu u kući. Pri tome se zahtjeva od ZigBee koordinatorskog čvora da prenese informaciju o alarmu svim drugim čvorovima unutar alarmnog sustava.

Verifikacija – ovo je zadnja vrsta alarma koja, ukoliko je primjerice ukućan zaboravio ključ kuće ili neki drugi način direktne verifikacije da se indirektno verificira te na taj način opravda svoj boravak u kući. Načelno prilikom ulaska ne verificirane osobe u kuću čvor bi uz konzultaciju s centralnim sustavom, prema zadanom rasporedu boravka osoba u kući, nastojao odrediti o kome se radi. Nakon prolaska određenog vremena sustav bi trebao potvrditi verifikaciju ili alarmirati ukućane koristeći tihi ili kućni alarm. Ukoliko je prisutan direktno verificiran ukućan on također može verificirati nepoznatu osobu.

3. Načini rada alarmnog sustava kuće

Postoje šest načina rada alarmnog sustava kuće koji se aktiviraju zaviseći o tome da li je kuća prazna ili puna te o kojem se vremenskom razdoblju radi. Sustav poznaje rad ukoliko je prazna kuća i to:

- Dulje vrijeme – koristi se primjerice u slučaju godišnjeg odmora kada je kuća prepuštena sama sebi. Način rada podrazumijeva da ukoliko dođe do aktiviranja bilo kojeg senzora aktiviranje tihog alarma.
- Kraće vrijeme – koristi se ukoliko ne postoji raspored boravka ukućana. Dodatno, postoji mogućnost indirektne verifikacije u nekom kraćem razdoblju. Isto kao i prethodnom slučaju aktivira se tihi alarm u slučaju provale ili ne dospijeća potvrde o neverificiranom ulasku u kuću

Ukoliko je kuća puna postoje tri načina rada. Dovoljno je napomenuti da se u slučaju pune kuće primarno uvijek aktivira kućni alarm. Načini rada u ovom slučaju su:

- Dan – podrazumijeva da se radi o radnom ili neradnom danu unutar tjedna. Način rada kuće omogućava indirektnu verifikaciju putem provjeravanja rasporeda boravka ukućana.
- Noć – također se podrazumijeva da se radi o radnom ili neradnom danu unutar tjedna. Razlika između dva moda je mogućnost indirektne verifikacije putem provjere jednog od ukućana.
- Zabava – primarno dizajniran za događaje gdje ne postoji raspored boravka ukućana ili kada bi indirektna verifikacija ulaska i izlaska iz prostorije bila zamorna za korisnika. U ovom načinu rada sustav ne vodi se računa o nasilnom ulasku u kuću. Ipak, nadzor na mogućim spojenim ne protuprovalnim sustavima se ne isključuje

Ukoliko korisnik alarmnog sustava ima neki drugi plan o korištenju kuće ostavljena je mogućnost aktiviranja proizvoljnog načina pri kojem sustav samo obavještava ukućane i pritom ih pita koja bi trebala biti reakcija na nastali problem.

4. Protokol i komunikacija

Radi potreba projektnog zadatka razvijen je protokol u suradnji s kolegama povezanih sustava kako bi što jednostavnije i brže mogli komunicirati i posljedično tome prenositi informacije o stanjima čvorova. Postoje dva smjera komunikacije te će se tako protokoli i razmatrati.

4.1. Komunikacija od čvora prema koordinatoru

Zbog nastojanja minimiziranja protokola komunikacije između krajnjeg čvora te koordinatorskog čvora koristi se, u većini vremena, jednobajtna instrukcija čija je struktura sljedeća:

Tabela 1. Format naredbe prema koordinatoru

7	6	5	4	3	2	1	0
BBBB				RR		II	

Tabela 1. Format naredbeopisuje izgled jednobajtne naredbe koja se može podijeliti u tri dijela. Prvi, *BBBB* koji zauzima gornjih četiri bitova odnosi se na broj prisutnih ljudi unutar jedne prostorije. Pri tome činimo pretpostavku da u jednoj prostoriji može biti maksimalno 15 ljudi. Drugi dio instrukcije, *RR* se odnosi na tip alarma. Prema Tabela 3. Instrukcije i alarmi. možemo vidjeti koji je kod pridružen kojoj vrsti alarma. Treći dio, *II* nam govori o kojem se izvorišnom čvoru radi. Ukoliko se pošalje rezervirana vrijednost *RR* krajnji čvor će promijeniti format podatka u:

Tabela 2. Posebni format naredbe

7	6	5	4	3	2	1	0
A/S	PPP			00		II	

Ova vrsta naredbe primarno služi koordinatorskom čvoru ukoliko je potrebno saznati u kojem su stanju senzori ili aktuatori. To jest, na poslani zahtjev krajnji čvor odgovara tako da mu prvi bit označava da li se radi o stanju senzora (0) ili aktuatora (1). Sljedeća tri bita, *PPP* govore o kojem se senzoru ili aktuatoru radi. Moguće je, na jedan čvor, priključiti do osam senzora s time da senzor pod rednim brojem sedam ima rezerviranu funkciju direktne verifikacije. Iza posebnog formata naredbe slijede dva bajta koji sadrže bilo koju vrijednost koja se povezuje s navedenim senzorom.

Tabela 3. Instrukcije i alarmi

RR	Objašnjenje	Postupak koordinatorskog čvora
00	Rezervirana vrijednost (status čvora)	Šalje se na zahtjev koordinatorskog čvora
01	Tihi alarm	Prenosi alarm centralnom sustavu. Ne prosljeđuje alarmnom čvoru
10	Kućni alarm	Prenosi alarm centralnom i alarmnom čvoru
11	Verifikacija	Koordinatorski čvor, ukoliko je sve u redu, šalje informaciju potvrde. Ukoliko ne pošalje smatra se da verifikacija nije prošla te se aktivira alarm.

4.2. Komunikacija od koordinatora prema čvoru

Također zbog nastojanja minimiziranja protokola komunikacije radi rasterećenja *ZigBee* modula, koriste se jednobajtne instrukcije. Izgled podatka koji se šalje na krajnji čvor prikazan je u Tabela 4. Format naredbe prema čvoru

Tabela 4. Format naredbe prema čvoru

7	6	5	4	3	2	1	0
PPPP				ZZ		II	

Može se primijetiti da je izgled približno jednak kao i u slučaju komunikacije prema koordinatoru osim što značenje prva četiri bita ovise o *ZZ* odnosno zapovjedi koja se šalje prema krajnjem čvoru. U ovom slučaju *II* postaju oznake odredišnog čvora. Kao što se vidi iz priloženog, postoje četiri zapovjedi koje se mogu poslati prema čvoru. Zapovjedi i značenje prvih četiri bita prikazane su u Tabela 5. Naredbe i značenja

Tabela 5. Naredbe i značenja

RR	Objašnjenje	Značenje PPPP
00	Potvrdni status prilikom verifikacije ili za isključivanje kućnog alarma	XXXX
01	Promjena režima rada alarmnog sustava	Načini rada kuće su prikazani u Tabela 6. Načini rada kuće i pripadni kodovi
10	Promjena stanja aktuatora	Oblik bitova je XAAA, pri čemu AAA označava koji je aktuator potrebno uključiti.
11	Zahtjev za slanjem statusa pojedinog senzora ili aktuatora na pojedinom čvoru	Format je identičan početku posebnog formata naredbi opisanog u prošlom potpoglavlju.

Tabela 6. Načini rada kuće i pripadni kodovi

XPPP	Objašnjenje
X001	Prazna kuća – dulje vrijeme
X010	Prazna kuća – kraće vrijeme
X011	Puna kuća – radni dan – dan
X100	Puna kuća – radni dan – noć
X101	Puna kuća – zabava
X110	Proizvoljno

5. Funkcije čvora

U ovom poglavlju ukratko će se opisati principi prema kojima su programirani čvorovi te pripadne funkcije.

5.1. void odluka_alarm(void)

Dio koda koji nadzire odluku o vrsti alarmiranja prema načinu rada kuće jest sljedeći:

```
void odluka_alarm(void){
  if (dohvat_senzora(senzor7)){
    broj_prisutnih++;
    return;
    }
  else{
    switch (mod_kuce){
      case prazna_kuca_dulje:
        alarm(tihi_alarm);
        break;
      case prazna_kuca_krace:
        alarm(verifikacija);
        delay(2000);
        if ((primaj() & ZZ) == 0)
          return;
        else
          alarm(tihi_alarm);
        break;
      case puna_kuca_radni_dan:
        alarm(verifikacija);
        delay(4000);
        if ((primaj() & ZZ) == 0)
          return;
        else
          alarm(kucni_alarm);
        break;
      case puna_kuca_radni_noc:
        alarm(kucni_alarm);
        alarm(verifikacija);
        break;
      case puna_kuca_veselica:
        alarm(tihi_alarm);
        break;
      case proizvoljno:
        alarm(verifikacija);
        break;
      }
  }
}
```

Može se primijetiti nekoliko stvari u kodu. Prvo, postoji glavni *if* uvjet koji se pita, bez obzira na način rada kuće, da li je došlo do direktne verifikacije. Postupak direktne verifikacije je rezerviran na mjestu osjetila pod rednim brojem sedam. Ukoliko je došlo do direktne verifikacije bilježi se samo povećanje broja prisutnosti unutar kuće. U drugom dijelu *if* naredbe nalazi se *switch-case* uvjetno grananje koje s obzirom na trenutni način rada kuće odlučuje koji je alarm potrebno poslati. Pri tome se koristi funkcija `alarm()`. Dodatno u slučajevima kao što su `prazna_kuca_krace` i `puna_kuca_radni_dan` ostavljena je mogućnost zaustavljanja odluke na neko određeno vrijeme kako bi se izvršila verifikacije nepoznate prisutnosti.

5.2. void alarm(byte ulaz)

Zaviseći o tipu alarma sljedeći kod šalje podatak koordinatorskom čvoru:

```
void alarm(byte ulaz){
  byte pom = 0;
  pom = (broj_prisutnih & 0xFF) << 4;
  posalji_podatke(pom | ulaz | cvor, 0);
}
```

Kod je vrlo jednostavan. Prema zadanom ulazu koji određuje tip alarma, broju ljudi unutar prostorije te imenu čvora određuje izlazni podatak koji se šalje *ZigBee* protokolom. Pri tome se vrijednost `broj_prisutnih` pomiče u gornjih četiri bita te se spaja s ILI funkcijom s ostalim podatcima.

5.3. void rezim_rada(byte ulaz)

Kod pomoću kojega mijenjamo režim rada kuće je sljedeći:

```
void rezim_rada(byte ulaz){
  mod_kuce = 0;
  if ((ulaz & PPPP) != 0)
     mod_kuce = ulaz & PPPP;
  else
     mod_kuce = prazna_kuca_dulje;
}
```

Koristeći prije definiranu masku PPPP možemo I funkcijom izolirati prva četiri bita te sukladno s njima postaviti način rada kuće. Ukoliko je došlo do pogreške u prijenosu ili moguće greške unutar sustava, čvor automatski postavlja način rada na najstroži mogući. Važno je navesti da varijabla `mod_kuce` jest globalna varijabla koja sadrži način rada kuće u bajtnoj reprezentaciji.

5.4. void aktuator(byte ulaz)

Prema zadanom ulazu koji je primljen putem *ZigBee* protokola mijenja se stanje aktuatora slijedećim kodom:

```
void aktuator (byte ulaz){
  switch (ulaz & PPPP){
    case aktuator0:
      // Gasi sve
      break;
    case aktuator1:
      // Set naredbi 1
      break;
    case aktuator2:
      // Set naredbi 2
      break;
    case aktuator3:
      // Set naredbi 3
      break;
    case aktuator4:
      // Set naredbi 4
      break;
    case aktuator5:
      // Set naredbi 5
      break;
    case aktuator6:
      // Set naredbi 6
      break;
    case aktuator7:
      // Set naredbi 7
      break;
  }
}
```

Jednostavna funkcija koja koristeći *switch-case* uvjetno grananje i masku PPPP odlučuje prema ulaznom podatku o kojem se aktuatoru radi te posljedično vrši naredbe. Ovakav način odlučivanja osigurava potrebnu modularnost kako za trenutne zahtjeve projekta tako i za buduća proširenja ili nadogradnje sustava. Važno je za navesti da ukoliko se primi podatak 0000 automatski se gase svi aktuatori spojeni na čvor.

5.5. void status(byte ulaz)

Prema zadanoj zapovjedi o statusu pojedinog čvora koja se šalje od koordinatorskog čvora poziva se sljedeći kod:

```
void status(byte ulaz){
  if ((AS & ulaz) == 0)
    posalji_podatke((PPPP & ulaz) | cvor | R0 , dohvat_senzora(ulaz));
  else if ((AS & ulaz) != 0)
    posalji_podatke((PPPP & ulaz) | cvor | R0, dohvat_aktuator(ulaz));
}
```

S maskom AS i I logičkom funkcijom izolira se prvi bit te pomoću njega određuje da li je došao zahtjev za stanjem senzora ili aktuatora. Posljedično tome šalju se zatraženi tipovi podataka prema već prije definiranom protokolu.

5.6. word dohvat_senzora(byte ulaz)

Ukoliko je potrebno saznati stanje pojedinog senzora poziva se izvršavanje sljedećeg koda:

```
word dohvat_senzora(byte ulaz){
  switch (ulaz & PPPP){
    case senzor0:
      //Stanje senzora 0
      return 0;
      break;
    case senzor1:
      // Stanje senzora 1
      return 0;
      break;
    case senzor2:
      // Stanje senzora 2
      return 0;
      break;
    case senzor3:
      // Stanje senzora 3
      return 0;
      break;
    case senzor4:
      // Stanja senzora 4
      return 0;
      break;
    case senzor5:
      // Stanja senzora 5
      return 0;
      break;
    case senzor6:
      // Stanja senzora 6
      return 0;
      break;
    case senzor7:
      //Direktna verifikacija
      return 0;
    default:
      return 0;
  }
}
```

Slijedeći iste principe koji su se primijenili unutar funkcije za promjenu stanja aktuatora. *Switch-case* uvjetnim grananjem odlučujemo, prema ulaznom podatku te maskom PPPP, o kojem se senzoru radi te posljedično

vrše funkcije za dohvat podataka. Razlika između dvije navedene funkcije jest u činjenici da dohvat senzora vraća dvobajtni podatak vrijednosti senzora.

5.7. byte dohvat_aktuatora(byte ulaz)

Analogno prošloj funkciji za senzore izvedena je funkcija za dohvat stanja pojedinog aktuatora. Preporuča se da se prilikom poziva promjene stanja aktuatora unutar globalnih varijabli zapišu i njihova stanja koja se putem ove funkcije mogu poslati.

```
byte dohvat_aktuator(byte ulaz){
  switch (ulaz & PPPP){
    case aktuator0:
      //Svi aktuatori ugašeni
      break;
    case aktuator1:
      // Stanja aktuatora 1
      break;
    case aktuator2:
      // Stanja aktuatora 2
      break;
    case aktuator3:
      // Stanja aktuatora 3
      break;
    case aktuator4:
      // Stanja aktuatora 4
      break;
    case aktuator5:
      // Stanja aktuatora 5
      break;
    case aktuator6:
      // Stanja aktuatora 6
      break;
    case aktuator7:
      // Stanja aktuatora 7
      break;
  }
}
```

Može se primijeti da je jedina razlika, od prošle funkcije, povratna vrijednost koja je sada jednobajtna.

5.8. void zapovjed(byte ulaz)

Sukladno s prijašnjim funkcijama razvijen je i kod koji s obzirom na ulazni podatak i maskom ZZ odlučuje o kojoj se zapovjedi radi te djeluje sukladno.

```
void zapovjed(byte ulaz){
  byte instrukcija = 0;
  instrukcija = ZZ & ulaz;
  switch (instrukcija){
    case promjena_rezima:
      rezim_rada(ulaz);
      break;
    case promjena_aktuatora:
      aktuator(ulaz);
      break;
    case status_cvora:
      status(ulaz);
      break;
  }
}
```

Radi se ponovno o *switch-case* uvjetnom grananju koje zaviseći o zapovjedi poziva potrebne funkcije.

5.9. Napomene

Funkcije za slanje i primanje podataka neće biti obrađene unutar ovog rada pošto je to bio predmet drugog projekta. Dovoljno je reći da u svakom trenutku maksimalan broj podataka koji se šalje jest tri bajta dok se prima samo jedan. Dodatno važno je napomenuti da bi se sve navedene funkcije trebale pozivati sljedećim slijedom:

```
if (dohvat_senzora(senzori[i]) == 0){
   podatak = primaj();
   if ((podatak & II) == cvor ){
      zapovjed(podatak);
      podatak = 0;
   }
}
else {
   odluka_alarm();
}
```

Ukoliko se ništa nije dogodilo, odnosno ukoliko niti jedan senzor nije u aktivnom stanju vrše se druge lokalne funkcije održavanja čvora kao što je primanje podataka. Ukoliko postoji podatak i ako je namijenjen čvoru provjerava se o kojoj se zapovjedi radi. Ako se s druge strane dogodilo nešto poziva se funkcija `odluka_alarm` koja sukladno o namještenom načinu rada kuće odlučuje o potrebnim akcijama.

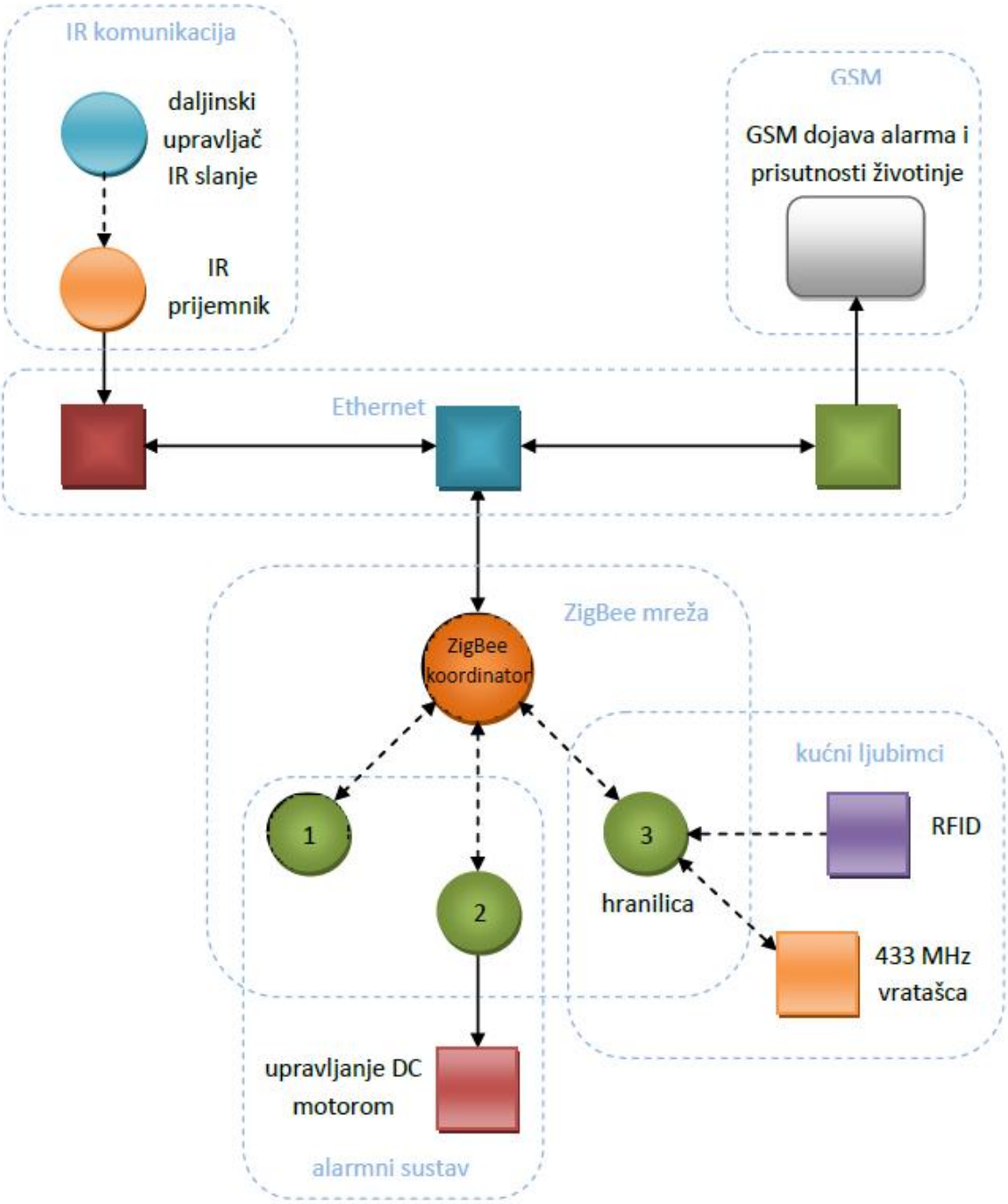
Dodatno prilikom ispitivanja svih uvjeta unutar prije navedenih funkcija koristile su se labele definirane na početku programskog koda. Unutar labela definiran je protokol komunikacije u bajtnom prikazu radi manjeg zauzeća radne memorije. Korištene labele su:

```
// Modovi rada kuce
#define prazna_kuca_dulje    B00010000
#define prazna_kuca_krace    B00100000
#define puna_kuca_radni_dan  B00110000
#define puna_kuca_radni_noc  B01000000
#define puna_kuca_veselica   B01010000
#define proizvoljno          B01100000

// Stanja senzora SSS
#define senzor0 B00000000 // Nijedan senzor nije pobuden
#define senzor1 B00010000
#define senzor2 B00100000
#define senzor3 B00110000
#define senzor4 B01000000
#define senzor5 B01010000
#define senzor6 B01100000
#define senzor7 B01110000

// Stanja aktuatora AAA
#define aktuator0 B10000000 // Svi aktuatori su ugaseni
#define aktuator1 B10010000
#define aktuator2 B10100000
#define aktuator3 B10110000
#define aktuator4 B11000000
#define aktuator5 B11010000
#define aktuator6 B11100000
#define aktuator7 B11110000

// Vrste alarma RR
#define R0            B00000000
#define tihi_alarm    B00000100
#define kucni_alarm   B00001000
#define verifikacija  B00001100

//Instrukcije od koordinatorskog cvora ZZ
#define promjena_rezima    B00000100
#define promjena_aktuatora B00001000
#define status_cvora       B00001100
#define sve_ok             B00000000

//Cvorovi II
#define cvor         B00000001 // Trenutni cvor je A1
#define A1           B00000001
#define A2           B00000010
#define koordinator  B00000011

// Maske
#define AS   B10000000
#define PPPP B11110000
#define ZZ   B00001100
#define II   B00000011
```

Potrebno je i napomenuti da je prilikom programiranja bio korišten Arduino razvojni sustav te bi se korisnik ovog dokumenta trebao prije konzultirati o programskom jeziku te razvojnom sustavu.

6. Povezani podsustavi

Na sljedećoj slici nalaze logička shema povezanih sustava te položaj ovog projekta kao samoga.



Slika 1. Povezani podsustavi

Ovaj projekt je izrađen u suradnji sa:

- ZigBee koordinator i komunikacija sa ZigBee čvorovima 1, 2 i 3 – Dinko Oletić
- upravljanje DC motorom (zatvaranje prozora u spavaćoj sobi kao dio alarmnog sustava) – Nikša Maslović
- hranilica za mačku s dojavom (ne)prisutnosti životinje (ZigBee čvor 3) – Edo Pekarić

7. Zaključak

Daljnja unaprjeđenja su neizostavna ako bi se ovaj sustav doista htio plasirati na tržište. Na sreću nedostaci većina nedostataka se može riješiti na jednostavan način. Primjerice ukoliko se želi povećati broj čvorova potrebno je samo proširiti raspon podataka koji se šalje iz ili u pojedini čvor. Dodatno kao treći krug sigurnosti trebala bi se ostvariti sigurnost komuniciranja između pojedinih čvorova koristeći enkripciju ili slične metode zaštite. U nekim područjima trebalo bi se ipak više potruditi kako bi se razvila nesmetana komunikacija između više spojenih modula, kao što su ZigBee i Ethernet. Iako je ovaj rad razvijen na jednostavnim principima dokazuje da se uz korištenje pristupačnih razvojnih sustava, kao što su Arduino te uz malo mašte i razmišljanja tokom programiranja, mogu izvesti, u relativno kratkom roku, nove ideje i što je važnije primjene samih.

8. Literatura

[1] http://www.arduino.cc/playground/Main/CapSense, Capacitive Sensing Library, Paul Badger

[2] The alarm, sensor & securitycircuit cookbook, Thomas Petruzzellis, McGraw-Hill 1994.

9. Pojmovnik

Pojam	Kratko objašnjenje	Više informacija potražite na
ZigBee	ZigBee je bežični komunikacijski protokol namjenjem osobnim mrežama s malom propusnošću i malom potrošnjom energije.	http://hr.wikipedia.org/wiki/ZigBee
Maske	Postupak izoliranja pojedinih bitova unutar slijeda bitova	http://en.wikipedia.org/wiki/Mask_(computing)
Arduino	Open-source razvojni sustav	http://en.wikipedia.org/wiki/Arduino