



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Лазар Вукасовић

Реализација ZigBee детектора покрета употребом PIR сензора

МАСТЕР РАД

Нови Сад, 2017



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:	
Идентификациони број, ИБР:	
Тип документације, ТД:	Монографска документација
Тип записа, ТЗ:	Текстуални штампани материјал
Врста рада, ВР:	Дипломски – мастер рад
Аутор, АУ:	Лазар Вукасовић
Ментор, МН:	Проф. Др Иштван Пап
Наслов рада, НР:	Реализација ZigBee детектора покрета употребом PIR сензора
Језик публикације, ЈП:	Српски / латиница
Језик извода, ЈИ:	Српски
Земља публикавања, ЗП:	Република Србија
Уже географско подручје, УГП:	Војводина
Година, ГО:	2017
Издавач, ИЗ:	Ауторски репринт
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, ФО: (поглавља/страна/ цитата/табела/слика/графика/прилога)	7/46/0/18/19/0/0
Научна област, НО:	Електротехника и рачунарство
Научна дисциплина, НД:	Рачунарска техника
Предметна одредница/Кључне речи, ПО:	Сензорске мреже, <i>ZigBee</i> , <i>IEEE 802.15.4</i> , <i>PIR</i> сензор
УДК	
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, ВН:	
Извод, ИЗ:	Realizacija detektora pokreta korišćenjem PIR senzora, koji se povezuje u <i>ZigBee</i> mrežu i komunicira sa drugim uređajima na mreži.
Датум прихватања теме, ДП:	
Датум одбране, ДО:	
Чланови комисије, КО:	Председник: Доц. др Небојша Пјевалица
	Члан: Доц. др Иван Мезеи
	Члан, ментор: Проф. др Иштван Пап
	Потпис ментора



KEY WORDS DOCUMENTATION

Accession number, ANO :	
Identification number, INO :	
Document type, DT :	Monographic publication
Type of record, TR :	Textual printed material
Contents code, CC :	Master Thesis
Author, AU :	Lazar Vukasović
Mentor, MN :	Prof. Ph.D. Ištvan Pap
Title, TI :	Realisation of ZigBee motion detection device using PIR sensor
Language of text, LT :	Serbian
Language of abstract, LA :	Serbian
Country of publication, CP :	Republic of Serbia
Locality of publication, LP :	Vojvodina
Publication year, PY :	2017
Publisher, PB :	Author's reprint
Publication place, PP :	Novi Sad, Dositeja Obradovica sq. 6
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)	7/46/0/18/19/0/0
Scientific field, SF :	Electrical Engineering
Scientific discipline, SD :	Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, S/KW :	sensor networks, ZigBee, IEEE 802.15.4, PIR
UC	
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, N :	
Abstract, AB :	Realisation of movement detector device which connects to <i>ZigBee</i> network and communicates with other <i>ZigBee</i> devices in the network.
Accepted by the Scientific Board on, ASB :	
Defended on, DE :	
Defended Board, DB :	President: Asst. Nebojša Pjevalica
	Member: Asst. Ivan Mezei
	Member, Mentor: Prof. Ph.D. Ištvan Pap
	Menthor's sign

Zahvalnost

Zahvaljujem se *OBLO* kompaniji na pruženoj mogućnosti za realizaciju ovog rada, kao i kolegama iz *OBLO node* tima na stručnoj pomoći i savetima prilikom izrade ovog rada. Posebnu zahvalnost dugujem svojim roditeljima Aćimu i Snežani koji su bili veliki oslonac i svojim savetima i moralnom podrškom doprineli mom školovanju.

SADRŽAJ

1. Uvod.....	1
2. Teorijske osnove	3
2.1 ZigBee specifikacija	3
2.2 Osnovni pojmovi	6
2.2.1 PAN ID i prošireni PAN ID.....	6
2.2.2 Korišćenje emitovanja (<i>broadcast</i>).....	6
2.2.3 ZigBee Endpoint	7
2.2.4 ZigBee klaster (<i>ZigBee cluster</i>) komande i atributi	8
2.2.4.1 Occupancy Sensing Klaster	10
2.2.4.2 IAS Zone Klaster.....	11
2.2.4.3 On/Off Klaster	13
2.3 Arhitektura ZigBee protokola.....	15
2.4 Formiranje, pridruživanje i ponovno pridruživanje mreži	16
2.5 CC2530.....	20
2.5.1 Organizacija memorije.....	23
2.6 Fizička arhitektura PIR uređaja.....	23
3. Programska podrška.....	25
3.1 Koncept rešenja	25
3.2 TI OSAL Task system.....	25
3.3 EZ-Mode procedura.....	26
4. Programsko rešenje	29
4.1 Podsystem za interakciju sa korisnikom	29
4.2 Mehanizmi izveštavanja.....	31

4.3	Pomoćne i <i>ZB</i> koordinirajuće funkcije	35
4.4	Iskorišćenost memorije	37
5.	Testiranje i verifikacija	38
5.1	Alati korišćeni pri testiranju	38
5.2	Lista testova.....	40
6.	Zaključak	42
7.	Literatura.....	45

SPISAK SLIKA

<i>Slika 2.1.1 – Logo ZigBee alijanse</i>	<i>3</i>
<i>Slika 2.1.2 – Prikaz mesh mreže</i>	<i>4</i>
<i>Slika 2.1.3 – Različite topologije u ZigBee mreži</i>	<i>5</i>
<i>Slika 2.2.1 – Prikaz emitovanja poruke</i>	<i>6</i>
<i>Slika 2.2.2 – ZigBee endpoints prikaz.....</i>	<i>7</i>
<i>Slika 2.3.1 – ZigBee i IEEE arhitektura.....</i>	<i>16</i>
<i>Slika 2.4.2 – Prikaz frekvencijskog opsega ZigBee kanala u poređenju sa WiFi kanalima</i>	<i>17</i>
<i>Slika 2.4.3 – Proces formiranja ZB mreži.....</i>	<i>18</i>
<i>Slika 2.4.4 – Procedura pridruživanja mreži</i>	<i>19</i>
<i>Slika 2.5.1 – Unutrašnja organizacija modula na CC2530 SoC-u.....</i>	<i>21</i>
<i>Slika 2.5.2 – Prikaz razvojnog okruženja SmartRF EB Rev. 1.3</i>	<i>22</i>
<i>Slika 2.6.1 – Električno kolo za povezivanje PIR senzora</i>	<i>23</i>
<i>Slika 3.3.1 – Blok dijagram EZ-Mode procedure</i>	<i>26</i>
<i>Slika 3.3.2 – Blok dijagram EZ-Mode procedure kod inicijatora</i>	<i>27</i>
<i>Slika 4.2.1 – Tipična sekvenca razmena paketa pri izveštavanju</i>	<i>33</i>
<i>Slika 5.1.1 – Prikaz Ubiqua Protocol Analyzer alata</i>	<i>39</i>
<i>Slika 5.1.2 – Prikaz Tracer alata za verifikaciju</i>	<i>40</i>
<i>Slika 5.2.1 – Prikaz razmene mrežnog ključa na starijim verzijama ZB</i>	<i>43</i>
<i>Slika 5.2.2 – Prikaz razmene mrežnog ključa na ZB 3.0.....</i>	<i>43</i>

SPISAK TABELA

<i>Tabela 2.1.1 – Tipovi ZigBee uređaja.....</i>	<i>4</i>
<i>Tabela 2.2.1 – Prikaz osnovnih klastera sa njihovim identifikatorima.....</i>	<i>9</i>
<i>Tabela 2.2.2 – Atributi Occupancy Sensing klastera.....</i>	<i>10</i>
<i>Tabela 2.2.3 – Grupa atributa sa identifikatorom 0x000 informacije senzora detekcije</i>	<i>10</i>
<i>Tabela 2.2.4 – OccupancySensorType atribut.....</i>	<i>10</i>
<i>Tabela 2.2.5 – Grupa atributa sa identifikatorom 0x001 konfiguracija PIR senzora.....</i>	<i>11</i>
<i>Tabela 2.2.6 – IAS Zone cluster.....</i>	<i>12</i>
<i>Tabela 2.2.7 – ZoneStatus atribut.....</i>	<i>13</i>
<i>Tabela 2.2.8 – Prikaz atributa On/Off Klastera.....</i>	<i>14</i>
<i>Tabela 4.1.1 – Broj pritisaka tastera u korelaciji sa događajima.....</i>	<i>30</i>
<i>Tabela 4.1.2 – Ponašanje LED indikacije.....</i>	<i>31</i>
<i>Tabela 4.2.1 – Parametri funkcije zclSS_IAS_Send_ZoneStatusChangeNotificationCmd..</i>	<i>32</i>
<i>Tabela 4.2.2 – Prikaz zclConfigReportRec_t tipa strukture.....</i>	<i>34</i>
<i>Tabela 4.2.3 – Prikaz zclCfgReportRec_t tipa strukture.....</i>	<i>34</i>
<i>Tabela 4.3.1 – Lista implementiranih funkcija PIR uređaja.....</i>	<i>35</i>
<i>Tabela 4.3.2 – Lista funkcija registrovanih u callback tabeli.....</i>	<i>36</i>
<i>Tabela 4.4.1 – Prikaz iskorišćenosti memorije po sekcijama.....</i>	<i>37</i>
<i>Tabela 5.2.1 – Lista izvršenih testova.....</i>	<i>41</i>

SKRAĆENICE

ZB	- <i>ZigBee network, ZigBee mreža</i>
PIR	- <i>Pyroelectric Infrared Sensor, Piroelektrični infracrveni radijalni senzor</i>
BLE	- <i>Bluetooth Low Energy, Bluetooth tehnologija niske potrošnje</i>
IR	- <i>Infrared , Infracrveno (zračenje, spektar, senzor)</i>
PAN	- <i>Personal Area Network, Lična računarska mreža</i>
TI	- <i>Texas Instruments kompanija</i>
RF	- <i>Radio Frequency</i>
ZC	- <i>ZigBee Coordinator device, ZigBee koordinatorski uređaj</i>
ZR	- <i>ZigBee Router device, ZigBee usmerivački uređaj</i>
ZED	- <i>ZigBee End Device, ZigBee krajnji uređaj</i>
ZCL	- <i>ZigBee Cluster Library, ZigBee klaster biblioteka</i>
PHY	- <i>Physical layer, Najniži OSI sloj (fizički sloj)</i>
MAC	- <i>Medium access control layer, Drugi po redu OSI sloj</i>
NWK	- <i>Network, mreža ili mrežni sloj apstrakcije</i>
APS	- <i>ZigBee Application Support Sublayer, ZigBee aplikativni sloj</i>
HAL	- <i>Hardware Abstraction Layer, Sloj za apstrakciju fizičke arhitekture</i>
GND	- <i>Ground, Oznaka za signal na nultom potencijalu</i>

ZADATAK RADA

Zadatak rada je da se realizuje *ZigBee* detektor pokreta na bazi PIR senzora. Ovaj uređaj bi bio u stanju da detektuje pokret ruke ili tela i da tu informaciju prenese na kontrolnu logiku. Uspešno izveden rad će kao rezultat dati uređaj sa sledećim osobinama:

- Samostalni režim rada: mogućnost kontrole osvetljenja pomoću tastera
- Senzorski režim rada: mogućnost kontrole drugih uređaja iz OBLO ekosistema pomoću tastera
- Komunikacija sa uređajem se vrši pomoću ZigBee tehnologije
- Proširenje aplikacije mikrokontrolera da se uređaj u ZigBee mreži vidi kao senzor po ZigBee standardu
- Mogućnost uređaja da stvori *ZigBee* mrežu kako bi se drugi uređaji mogli povezati na nju
- Mogućnost ulaska uređaja u mrežu kao *Occupancy Device* i kao *IAS Zone Security Device*

1. Uvod

Elektronski sistemi koji se koriste za kontrolu funkcija i instalacija u kućama su sve češći. Ovi sistemi obavješćavaju korisnika o promenama u sistemu i obuhvataju upravljanje grejanjem, ventilacijom, svetlima, roletnama i sličnim uređajima. Najčešće korišćen termin za ovakve sisteme je pametna kuća odnosno *smart home* ili *intelligent home*. Njihova primena zasnovana je na potrebama da se korisniku pruži veći komfor, veća energetska efikasnost, bezbednost i dodatne mogućnosti čak i kada se korisnik ne nalazi u neposrednoj blizini kontrolisanih uređaja.

Kako bi jedan ovakav sistem mogao da se prilagodi čoveku potrebno je detektovati njegovo prisustvo, na primer kako bi se svetla u prostoriji uključila pri ulasku u prostoriju neophodno je prevashodno detektovati kretanje. Otuda se javila potreba za uređajima za detekciju pokreta.

Kao potencijalno rešenje senzora za detekciju pokreta nakon razmatranja svih mogućih opcija izabran je *PIR* senzor. On predstavlja jedan od najčešće korišćenih „detektora prisustva ljudi“ zbog svoje pristupačnosti na tržištu, niske cene, malih dimenzija i relativno lake implementacije. Ono što *PIR* senzor najviše razlikuje od ostalih detektora koji se koriste u identičnim primenama je osobina da je on pasivni element. To znači da ovaj senzor ne emituje nikakve signale, za razliku od ultrasoničnih uređaja, aktivnih (reflektujućih) *IR* snopova i radara. Ova osobina takođe omogućava korišćenje *PIR* modula uz baterijska napajanja.

Osnovni koncept rada ovakvih detektora zasniva se na detekciji *IR* zračenja koje emituje čovek ili neki drugi izvor pokreta u poređenju sa zračenjima u ambijentu. Ovo pokazuje da nije dovoljna samo detekcija *IR* zračenja, već je neophodno obraditi dobijene senzorske informacije, kako bi se one mogle dalje proslediti uređajima u automatskom sistemu ili pametnoj kući.

Komunikacija između uređaja u ovakvom sistemu se svela na trenutno dominantne tehnologije u ovoj oblasti, a to su *ZigBee*, *Zwave*, i *BLE*. Ovi protokoli su veoma efikasni u pogledu potrošnje energije.

Mesh mreže odlikuje nepostojanje središnjeg čvora preko kojeg se izvodi čitava komunikacija. Umesto toga, svaki čvor u mreži služi kao prenosnik na druge čvorove, iz tog razloga one su pouzdanije od ostalih komunikacionih mreža.

Arhitektura sistema za kućnu automatizaciju prikazana je na slici 1, ovi sistemi se najčešće sastoje od pametnih uređaja koje sve objedinjuje centralni kontroler, udaljenih servisa koji omogućavaju uvid u događaje i kontrolu uređaja u sistemu preko interneta, kao i mobilnih i *web* aplikacija za pristup i konfiguraciju sistema.



Slika 1 – Arhitektura sistema kućne automatizacije

U ovom radu predstavljeno je jedno rešenje programske podrške za centralni kontroler koji komunicira preko *ZigBee* protokola i koje je u skladu sa HA 1.2 specifikacijom. Cilj ove specifikacije je da centralni kontroler bude kompatibilan sa svim ostalim uređajima koji su usklađeni sa ovim standardom. To znači da je moguće spojiti više različitih uređaja, različitih proizvođača u jedinstven sistem. Ovo omogućava postojanje uređaja sa identičnom primenom i različitim implementacijama, što daje veći komfor pri odabiru komponenti sistema. Univerzalna podrška daje i dodatnu vrednost sistemu jer proširuje listu podržanih uređaja i olakšava dodavanje novih uređaja sa svim funkcionalnostima. Ovo je jedna od prednosti *ZigBee* u odnosu na neke druge tehnologije, kao na primer *Z-Wave*.

2. Teorijske osnove

U ovom poglavlju objašnjene su osnove karakteristike *ZigBee* protokola, arhitektura sistema i način funkcionisanja kao i osnovni pojmovi koji su neophodni za razumevanje daljeg teksta.

2.1 *ZigBee* specifikacija

ZigBee je bežična tehnologija razvijena kao otvoreni globalni standard koji treba da odgovori na jedinstvene zahteve niske cene i male potrošnje energije bežičnih uređaja *M2M* mreža. *ZigBee* je baziran na IEEE 802.15.4 standardu personalnih računarskih mreža (*PAN – Personal Area Network*), a radi u nelicenciranim opsezima uključujući 2.4 GHz, 900 MHz i 868 MHz.

Ovaj protokol stvorile su i ratifikovale kompanije, članice *ZigBee* alijanse (*ZigBee Alliance*). Preko 300 vodećih kompanija, vodećih u *IT* svetu čini članstvo u ovoj alijansi, čiji je logo prikazan na slici 2.1.1.

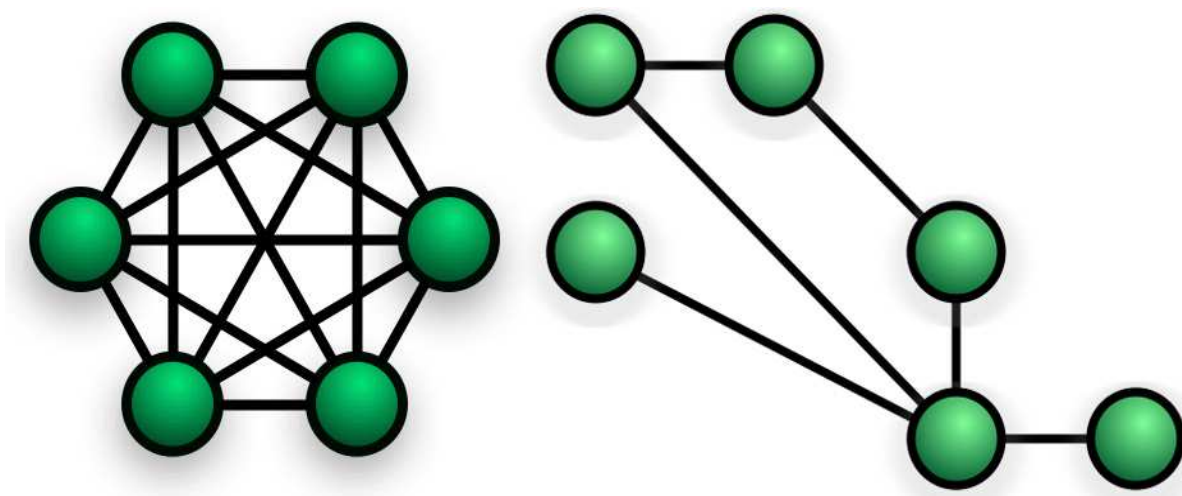
Protokol je dizajniran da na jednostavan način obezbedi bežični prenos podataka koji odlikuje bezbedna i pouzdana arhitektura.



Slika 2.1.1 – Logo *ZigBee* alijanse

Protokol je konstruisan tako da komunicira kroz bučna RF okruženja koja su uobičajna u komercijalnim i industrijskim aplikacijama. *ZigBee* karakteriše podrška za višestruke topologije mreža (kao što su *point-to-point*, *point-to-multipoint* i *mesh* mreže), nizak *duty cycle* (što prolongira životni vek baterije), mala kašnjenja, veliki broj čvorova na mreži (do 65.000 čvorova u teoriji dok je praktično ostvarimo do nekoliko stotina), šifrovanje komunikacije (što obezbeđuje bezbedan prenos podataka).

Ključna komponenta *ZigBee* protokola je mogućnost podržavanja *mesh* topologije (slika 2.1.2). U *mesh* topologiji, čvorovi su povezani sa drugim čvorovima, tako da višestruki putevi povezuju svaki čvor. Veza između čvorova se dinamički ažurira i optimizuje preko sofisticirane



Slika 2.1.2 – Prikaz mesh mreže

ugrađene *mesh* tabele za rutiranje. *Mesh* mreže su decentralizovane, svaki čvor je u stanju da samostalno istražuje mrežu. Takođe, kada čvor napusti mrežu, *mesh* topologija omogućava čvorovima da rekonfigurišu putanje rutiranja na osnovu nove strukture mreže. Karakteristike *mesh* topologije i ad-hoc rutiranje obezbeđuju veću stabilnost prilikom promene uslova ili pojave određenih problema na pojedinim čvorovima.

Postoje tri tipa različitih *ZigBee* uređaja koja su predstavljena tabelom 2.1.1.

1.	ZC – <i>ZigBee</i> Coordinator	<i>ZigBee</i> Koordinator
2.	ZR – <i>ZigBee</i> Router	<i>ZigBee</i> Usmerivač
3.	ZED – <i>ZigBee</i> End Device	<i>ZigBee</i> krajnji uređaj

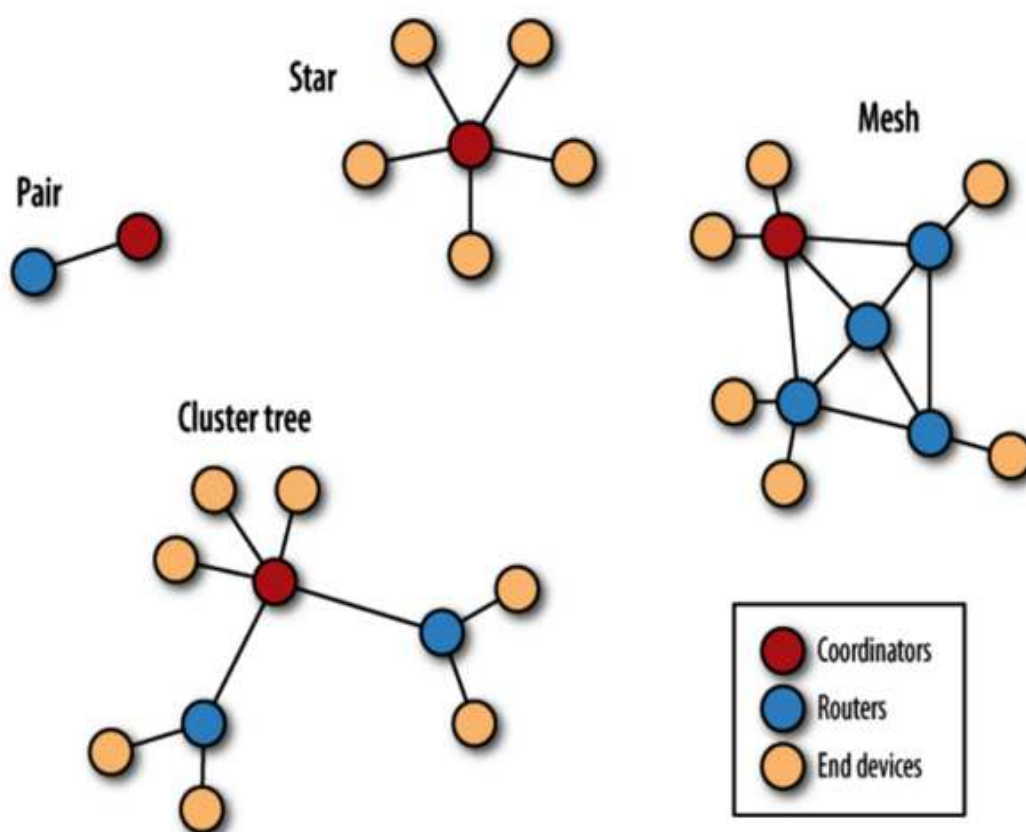
Tabela 2.1.1 – Tipovi *ZigBee* uređaja

ZC predstavlja koren stabla mreže, uređaj koji služi za spajanje sa drugim mrežama, svaka mreža ima jedan koordinator. On formira i održava mrežu. Ima ulogu centra od poverenja (engl. *Trust Center* jer skladišti sigurnosne ključeve

ZR predstavljaju usmerivačke uređaje na koje se ZED mogu povezati pored ZC, ove uređaje karakteriše veća potrošnja stoga oni ne koriste baterijska napajanja. Usmerivač proširuje pokrivenost mreže, formira putanje kako bi isporučio poruku do ciljanog uređaja kao i pronalaženja nove putanje do ciljanog uređaja ako uređaj nije dostupna po staroj. Usmerivač se povezuje ili sa drugim usmerivačem ili sa koordinatorom.

ZED predstavljaju krajnje uređaje, ovi uređaji koriste najčešće baterijsko napajanje i zbog toga veći deo radnog vremena provode u „spavanju“, odnosno u *sleep* režimu. Ovaj režim se napušta kada je potrebno obraditi neki događaj ili poslati poruku. Na ovaj način se vrši ušteda energije, a ujedno omogućuje pravilan rad uređaja.

Različite topologije *ZigBee* mreže sa komunikacionim vezama između uređaja prikazane su slikom 2.1.3.



Slika 2.1.3 – Različite topologije u *ZigBee* mreži

2.2 Osnovni pojmovi

U ovoj sekciji obrađeni su neki od pojmova koji su bitni za razumevanje teksta u nastavku. Zbog neadekvatnih prevoda na srpski jezik za mnoge pojmove uvedene su skraćenice koje su korišćene u daljem tekstu, čiji spisak se nalazi u listi skraćenica na početku.

2.2.1 PAN ID i prošireni PAN ID

Ovaj identifikator se koristi kako bi se unikatno markirala *ZigBee* mreža, tokom kreiranja mreže pomoću posebnog mehanizma se vodi računa da na istom prostoru ne mogu postojati dve mreže sa istim identifikatorom. Ovaj identifikator je definisan šesnaestobitnim brojem u rasponu od 0x0000 do 0x3FFF u heksadecimalnom zapisu. Ovaj raspon se razlikuje od onog definisanog u standardu 15.4 po tome što je broj mogućih *ZB* na istom prostoru manji, iz praktičnih razloga. Pored običnog identifikatora postoji i prošireni koji čini šezdesetičetvorobitni zapis, ova adresa se koristi pri kreiranju odgovora na *beacon* zahteve.

2.2.2 Korišćenje emitovanja (*broadcast*)

Emitovanje se koristi da se pošalje *ZigBee* poruka od jednog uređaja u celokupnu *ZigBee* mrežu u definisanom radijusu. Postoje 3 vrste emitovanja u *ZigBee* mrežama:

- 0xFFFF emitovanje svim uređajima
- 0xFFFFD emitovanje uređajima koji trenutno ne spavaju
- 0xFFFFC emitovanje uređajima koji su ruterski

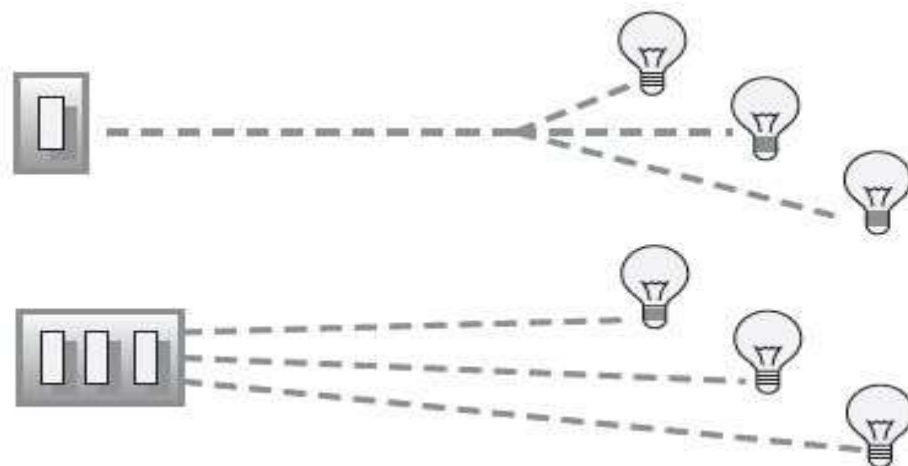
Pomoću radijus parametra moguće je definisati koliko skokova poruka može maksimalno da napravi pre zaustavljanja emitovanja. Slikom 2.2.1 prikazan je primer emitovanja poruke.



Slika 2.2.1 – Prikaz emitovanja poruke

2.2.3 ZigBee Endpoint

Ovaj termin se vezuje za sam *ZigBee* uređaj i predstavlja logički odvojenu celinu koja daje funkcionalnost uređaju. Moguće je na jednom uređaju imati do 240 *endpoint-a* što obezbeđuje veliku raznolikost u funkcijama i opcijama uređaja. Može se reći da *endpoint* funkcionalnost



Slika 2.2.2 – ZigBee endpoints prikaz

omogućava postojanje različitih aplikativnih profila na svakom uređaju, kao i zasebnu kontrolu svakim.

Na slici 2.2.2 prikazana je mogućnost da se na fizički jednom uređaju vežu svi tasteri da komanduju pojedinačnim uređajima i mogućnost da se jedan taster veže da komanduje svim, ovo je moguće upravo zbog postojanja nekoliko *endpoint* tački na uređaju (minimalno tri).

PIR uređaj poseduje četiri *endpoint* tačke ne uključujući nultu, a one su date ispod sa kratkim opisima

- o *Endpoint 0* – Standardni ZB Endpoint u okviru koga su implementirane osnovne ZB funkcije
- o *Endpoint 1* – Implementira IAS_ZONE deo uređaja

Opis:

- *Profile:* HA_PROFILE
- *Device ID:* IAS_ZONE
- *Input Clusters:*
 - o BASIC (Mandatory)
 - o IDENTIFY (Mandatory)
 - o IAS_ZONE (Mandatory)
 - o ALARM (Optional)
- *Otput Clusters:*
 - o IDENTIFY (Optional)

o *Endpoint 2* – Implementira funkcionalnost *Occupancy* klastera

Opis:

- *Profile:* *HA_PROFILE*
- *Device ID:* *OCC*
- *Input Clusters:*
 - *BASIC (Mandatory)*
 - *IDENTIFY (Mandatory)*
- *Otput Clusters:*
 - *IDENTIFY (Optional)*
 - *OCCUPANCY_SENSING (Mandatory)*

o *Endpoint 3* – Koristi se za funkcionalnosti karakteristične za *OBLO* sistem kućne automatizacije.

Opis:

- *Profile:* *OHN_PROFILE*
- *MSP profile id:* *0xC602*
 - *Device ID:* *OBLO_DEVICE_ID*
- *Manufacture Specific Clusters:*
 - *ZCL_CLUSTER_ID_OHN_CONFIG* *0x0000*
 - *ZCL_CLUSTER_ID_OHN_LNS* *0x0003*

o *Endpoint 4* – Pokriva funkcije odloženog gašenja kao i promene stanja iz uključenog u isključeno za uređaje iz *bind device* tabele.

Opis:

- *Profile:* *HA_PROFILE*
- *Device ID:* *ONOFF*
- *Input Clusters:*
 - *BASIC (Mandatory)*
 - *IDENTIFY (Mandatory)*
- *Otput Clusters:*
 - *IDENTIFY (Optional)*
 - *ON_OFF (Mandatory)*

2.2.4 *ZigBee* klaster (*ZigBee cluster*) komande i atributi

Klaster je definisan šesnaestobitnim identifikatorom i određuje tip primene aplikacije. Postoje različiti klasteri za upravljanje različitim uređajima i oni su predefinisani *ZB* standardom,

ukoliko ipak ne postoji određena funkcionalnost za postojeći klaster ili ne postoji klaster koji odgovara uređaju on se može proširiti ili kreirati po specifičnim zahtevima.

Neki od osnovnih klastera prikazani su tabelom 2.2.1, gde su sa desne strane naznačena imena klastera, dok su na levoj identifikatori. Svaki klaster sačinjava skup komandi i atributa. *ZigBee klaster biblioteka* (engl. *ZigBee Cluster Library*, u daljem tekstu *ZCL*) omogućava lako dobavljanje ili podešavanje atributa kroz poznati skup komandi svakog klastera. Komande se označavaju osmobitnim identifikatorom i mogu biti definisane i međuklasterski.

Atributi govore o trenutnim stanjima određenih događaja, ali mogu i da nose informacije o načinima izvršavanja komandi, trenucima u kojima ih je potrebno izvršiti kao i o broju ponavljanja. Atributi se definišu šesnaestobitnim brojem i mogu imati vrednost od 0x0000 do 0xffff u heksadecimalnom zapisu. Mogu se podešavati da prave izveštaje u određenim

Cluster ID	Cluster Name	Description
0x0000	Basic	Attributes for determining basic information about a device, setting user device information such as location, and enabling a device
0x0001	Power	Configuration Attributes for determining more detailed information about a device's power source(s), and for configuring under/over voltage alarms
0x0002	Device Temperature Configuration	Attributes for determining information about a device's internal temperature, and for configuring under/over temperature alarms
0x0003	Identify	Attributes and commands for putting a device into Identification Mode (e.g., flashing a light)
0x0004	Groups	Attributes and commands for group configuration and manipulation
0x0005	Scenes	Attributes and commands for scene configuration and manipulation
0x0006	On/Off	Attributes and commands for switching devices between "On" and "Off" states
0x0007	On/Off Switch Configuration	Attributes and commands for configuring On/Off
0x0008	Level Control	Attributes and commands for controlling devices that can be set to a level between fully "On" and fully "Off"
0x0009	Alarms	Attributes and commands for sending notifications and configuring alarm functionality
0x000a	Time	Attributes and commands that provide a basic interface to a real-time clock
0x000b	RSSI	Location Attributes and commands that provide a means for exchanging location information and channel parameters among devices

Tabela 2.2.1 – Prikaz osnovnih klastera sa njihovim identifikatorima

intervalima, po promeni određenog parametra ili okidanju nekog događaja.

Klasteri koji su korišćeni za izradu *PIR* detektora prisustva su opisani u poglavljima ispod.

2.2.4.1 Occupancy Sensing Klaster

Predstavlja jedan od klastera koji definiše sam uređaj, ovaj klaster je namenjen uređajima koji detektuju prisustvo. Atributi ovog klastera su prilično jednostavni i prikazani su u tabeli 2.2.2.

Identifikator grupe atributa	Opis
0x000	Informacije senzora detekcije
0x001	Konfiguracija PIR senzora
0x002	Konfiguracija ultrasoničnog senzora

Tabela 2.2.2 – Atributi Occupancy Sensing klastera

Grupa atributa sa identifikatorom 0x000 informacije senzora detekcije sadrži nekoliko atributa koji su dati tabelom 2.2.3.

Identifikator	Naziv	Tip	Opseg	Pristup	Obavezno
0x0000	<i>Occupancy</i>	8-bit map	0000 000x	Čitanje	Da
0x0001	<i>OccupancySensorType</i>	8-bit enumeration	0x00-0xfe	Čitanje	Da

Tabela 2.2.3 – Grupa atributa sa identifikatorom 0x000 informacije senzora detekcije

Occupancy predstavlja osmobaritnu bit mapu gde je iskorišćen samo poslednji bit, odnosno *LSB* koji daje informaciju o trenutnoj detekciji senzora. Ostali biti su rezervisani i ne koriste se. Ukoliko je ovaj bit jednak jedinici senzor je detektovao, u suprotnom nema detekcije.

OccupancySensorType atribut govori o tipu detektora koji se koristi i moguće vrednosti ovog atributa se nalaze u tabeli 2.2.4.

Vrednost atributa	Opis
0x00	<i>PIR senzor</i>
0x01	Ultrasonični senzor
0x02	PIR i ultrasonični senzor

Tabela 2.2.4 – *OccupancySensorType* atribut

Grupa atributa sa identifikatorom 0x002 predstavlja konfiguraciju *PIR* senzora. U ovoj grupi su svi atributi definisani kao opcioni, stoga ih nije nužno implementirati, takođe svi atributi se mogu čitati i pisati i prikazani u tabeli 2.2.5.

Prvi u nizu je atribut *PIROccupiedToUnoccupiedDelay* ovaj atribut definiše vreme u sekundama koje je potrebno da protekne od trenutka kada se promeni stanje iz stanja detekcije sve do stanje bez detekcije.

Atribut sa nazivom *PIRUnoccupiedToOccupiedDelay* takođe definiše vreme u sekundama, koje treba da protekne od poslednje detekcije pre nego što se omogući ponovno detektovanje. Ovaj atribut je obavezno podesiti ukoliko je implementiran atribut pod nazivom *PIRUnoccupiedToOccupiedThreshold*.

PIRUnoccupiedToOccupiedThreshold predstavlja osmобitni atribut koji broji stvarne detekcije senzora i tek kada se taj broj izjednači sa brojem definisanim u ovom atributu uređaj javlja da je došlo do detekcije. On se koristi kako bi se eliminisale lažne detekcije.

Identifikator	Naziv	Tip	Opseg	Pristup	Početna vrednost
0x0010	PIROccupiedToUnoccupiedDelay	Unsigned 16-bit integer	0x00-0xffff	Čitanje/pisanje	0x00
0x0011	PIRUnoccupiedToOccupiedDelay	Unsigned 16-bit integer	0x00-0xffff	Čitanje/pisanje	0x00
0x0012	PIRUnoccupiedToOccupiedThreshold	Unsigned 16-bit integer	0x01-0xfe	Čitanje/pisanje	0x01

Tabela 2.2.5 – Grupa atributa sa identifikatorom 0x001 konfiguracija PIR senzora

Ovaj klaster ne prima komande, niti poseduje komande koje se generišu kako na svojoj serverskoj tako i na klijentskoj strani. Nakon događaja detekcije uređaj šalje izveštaj o stanju *occupancy* atributa.

2.2.4.2 IAS Zone Klaster

Ovaj klaster definiše spregu funkcionalnosti IAS bezbednosnog zonskog uređaja (engl. *IAS security zone device*). Podržava dve tipa alarma po zoni, izveštaje o stanju baterije i nadgledanje IAS mreže. U tabeli 2.2.6 dati su atributski skupovi IAS Zone klastera.

Identifikator grupe atributa	Opis
0x000	Informacije o zoni (<i>Zone information</i>)
0x001	Podešavanja zone (<i>Zone configuration</i>)

Tabela 2.2.6 – IAS Zone cluster

Zone information skup atributa sadrži attribute *ZoneState*, *ZoneType* i *ZoneStatus*, svi atributi su obavezni za implementaciju, takođe svi su samo čitljivi. *ZoneState* je osmobični atribut koji pokazuje da li je *IAS endpoint* uspešno prošao proceduru *enrollment*, odnosno da li je uređaj u stanju *enrolled*.

ZoneType određuje tip alarma i u zavisnosti od toga određuje se događaji koji će se prozivati u slučaju promene stanja za polje *Alarm1* ili *Alarm2*.

ZoneStatus atribut predstavlja izveštaj koji se šalje u mrežu kada dođe do neke promene u posmatranim vrednostima. U zavisnosti od tipa *IAS* uređaja, odnosno atributa *ZoneType* *ZoneStatus* parametri se mogu drugačije tumačiti, tabelom 2.2.7 je dato jedno moguće tumačenje.

Redni bit atributa	Značenje	Vrednosti
0	Alarm1	1 – Stanje alarma 0 – Bez alarma
1	Alarm2	1 – Stanje alarma 0 – Bez alarma
2	Tamper	1 – Tamper zaštita aktivirana 0 – Tamper zaštita neaktivirana
3	Baterija	1 – Napon baterije nizak 0 – Baterija OK
4	Beleška 1	1 – Slanje izveštaja 0 – Bez slanja izveštaja
5	Beleška 2	1 – Tamper zaštita aktivirana 0 – Tamper zaštita neaktivirana
6	Greška	1 – Tamper zaštita aktivirana 0 – Tamper zaštita neaktivirana

7	AC mreža	1 – Defekt u AC mreži 0 – AC mreža OK
8	Test	1 – Senzor u test režimu 0 – Senzor test režima
9	Defekt baterije	1 – Defektna baterija 0 – Baterija funkcionira normalno

Tabela 2.2.7 – ZoneStatus atribut

Jedan od atributa iz grupe attribute pod nazivom *Zone Settings* je i *IAS_CIE_Address*. Ovaj atribut čuva adresu uređaja koju je generisao server, na koji će komande koje se generišu biti poslate. Od same implementacije zavisi koji atributi će biti u kom trenutku dostupni za pisanje, odnosno za čitanje.

Komande koje se primaju na serverskoj strani *IAS* klastera su *Zone Enroll Response* dok komande koje se generišu su *Zone Status Change Notification* i *Zone Enroll Request*. *Zone Enroll* komande se koriste pri *enrollment* proceduri, dok *Zone Status Change Notification* predstavlja izveštaj koji se šalje uređaju čija adresa je sačuvana u *IAS_CIE_Address*. U implementaciji *PIR* uređaja definisano je da se ovaj izveštaj šalje na svaki broj detekcija koji je jednak *threshold* atributu koji je ranije opisan.

2.2.4.3 On/Off Klaster

Ovaj klaster se koristi za prebacivanje uređaja iz uključenog u isključeno stanje. Pošto je implementacija *PIR* uređaja takva da on može biti samo inicijator (engl. *initiator*), jer može samo da izdaje komande drugim uređajima s obzirom da ne poseduje fizički nijedan aktuator na sebi koji bi drugi mogli komandovati. U nastavku će biti obrađeni samo korišćeni delovi ovog klastera. Neki od atributa koji su korišćeni su *OnOff*, *OnTime* i *OffWaitTime*, njihove osobine date su tabelom 2.2.8.

Identifikator	Naziv	Tip	Opseg	Pristup	Početna vrednost	Obavezno
0x0000	<i>OnOff</i>	Boolean	0x00 – 0x01	Čitanje/ pisanje	0x00	Da
0x4001	<i>OnTime</i>	Boolean	TRUE ili FALSE	Čitanje/ pisanje	TRUE	Ne
0x4002	<i>OffWaitTime</i>	Unsigned 16-bit integer	0x0000 – 0xffff	Čitanje/ pisanje	0x0000	Ne

Tabela 2.2.8 – Prikaz atributa On/Off Klastera

OnOff atribut poseduje dva moguća stanja. Ukoliko je vrednost ovog atributa nula poruka koja se šalje uređaju je da treba da pređe u isključeno stanje, dok ukoliko je vrednost jednaka jedinici poruka predstavlja komandu za uključivanje uređaja ili nekog njegovog aktuatora.

OnTime predstavlja vreme u desetom delu sekunde (1/10 sekunde) koje određuje koliko dugo će se uključeno stanje zadržati. Ukoliko pre isteka ovog vremena pristigne nova komanda za isključeno stanje ona će biti ignorisana. Na isti način funkcioniše i *OffWaitTime*, samo što ovaj atribut određuje period vremena koje mora da istekne od trenutka kada uređaj pređe u isključeno stanje do moguće promene stanja.

Na *PIR* uređaju implementirano je da se nakon svake prave detekcije (detekcije koja ispunjava uslove *threshold* atributa) šalje komanda *On with timed off*. Ova komanda trenutno prebacuje uređaj u uključeno stanje, ali takođe i definiše vreme trajanja tog stanja. Nakon isteka tog vremena uređaj prelazi u isključeno stanje.

2.3 Arhitektura *ZigBee* protokola

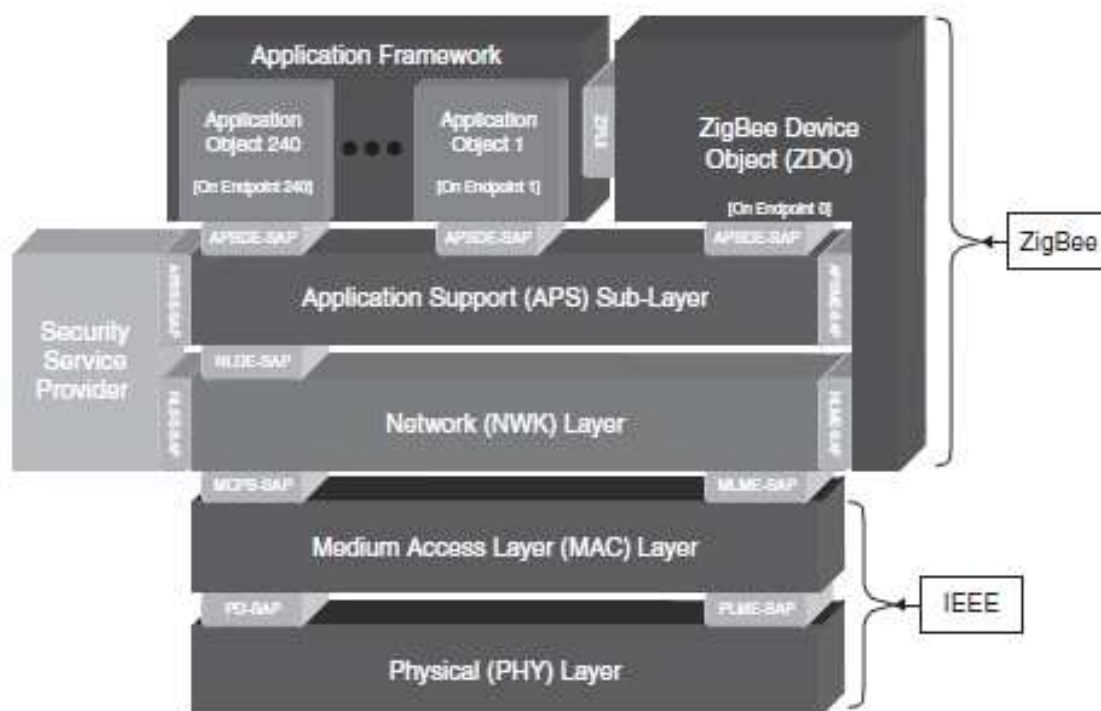
IEEE 802.15.4 standard (ili jednostavnije 15.4 standard) kreiran je i održava se od strane IEEE. Ovaj standard definiše fizički nivo arhitekture (u daljem tekstu *PHY* – *Physical*) kao i *Media Access Control* nivo (u daljem tekstu *MAC*) nad kojim je definisana *ZigBee* specifikacija. Ova specifikacija definiše sledeće mehanizme:

- mehanizme za pronalaženje mreža
- mehanizme za formiranje i priključivanje mrežama
- mehanizme promene kanala
- mehanizme detekcije interefercija i smetnji na kanalima
- metode za prenošenje podataka jednim skokom
- metode emitovanja podataka (*broadcasting*)

Ono što ne definiše 15.4 standard je više skokova u prenosu informacija (*multi-hop*), definisanje adresa kao ni aplikativni nivo kompatibilnosti.

Zadatak *PHY* nivoa je da prevodi pakete iz niza bajtova u *RF* spektar kao i obrnuto, *MAC* nivo omogućuje formiranje mreže, deljenje kanala i prenos podataka preko jednostrukih skokova (*single-hop*) na pouzdane načine.

ZigBee standard definiše sve slojeve iznad ovog koji su prikazani na *Slika 2.3.1 – ZigBee i IEEE arhitektura*, uključujući mrežni sloj (u daljem tekstu *NWK* – *Network layer*), aplikacioni sloj za podršku (u daljem tekstu *APS* – *Application Support*), (u daljem tekstu *ZDO* – *ZigBee Device Object*) i sigurnosne slojeve (*Security Service Provider*). Važno je napomenuti da *ZigBee* ne koristi sve definisano standardima u *PHY* i *MAC* slojevima, već samo deo koji je neophodan kako bi se uštedelo na memoriji.



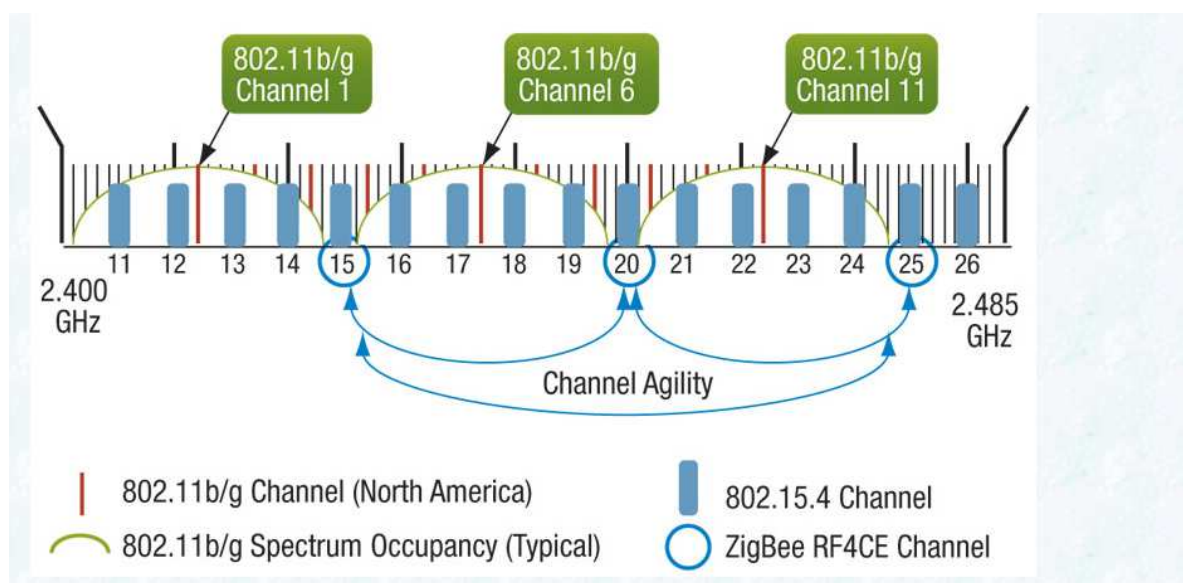
Slika 2.3.1 – ZigBee i IEEE arhitektura

2.4 Formiranje, pridruživanje i ponovno pridruživanje mreži

Kako bi uređaji međusobno mogli komunicirati potrebno je da se nalaze u mreži, uređaji mogu da uđu već postojeću mrežu ili da kreiraju novu. Samo ZC uređaji mogu kreirati mrežu, a ZR i ZED uređaji se mogu pridružiti istoj. Mnogi uređaji poseduju mogućnost da se ponašaju kao ZC, ZR i ZED po potrebi, nekada je ova osobina definisana pre samog kompajliranja (kako bi se uštedelo na memoriji), a nekada se ona definiše u toku rada u zavisnosti od zahteva.

Svaki uređaj dolazi sa definisanom 64-bitnom IEEE adresom (naziva se još i MAC ili duga adresa na engl. *long address*), ova adresa se dodeljuje prilikom proizvodnje. Pri procesu pridruživanja u mrežu uređaju se dodeljuje i unikatna 16-bitna kratka adresa (engl. *short address*) koja se još naziva *NwkAddr* – *Network Address*, ova adresa se koristi za razmenjivanje poruka unutar mreže, prevažodno zbog manje dimenzije kako bi se rasteretio saobraćaj u mreži.

ZigBee koordinatori formiraju mrežu, sam proces se svodi na definisanje unikatnog identifikatora mreže koji se naziva *PAN ID*, kao i biranja jednog od 16 mogućih kanala (11 – 26) koji su definisani 15.4 standardom. Kod biranja kanala najčešće se biraju kanali 15, 20, 25, 26 jer se oni nalaze na frekvencijskom opsegu između kanala koje zauzima WiFi mreža, što se može uočiti sa slike 2.4.2 ovime se smanjuje mogućnost interferencije.



Slika 2.4.2 – Prikaz frekvencijskog opsega ZigBee kanala u poređenju sa WiFi kanalima

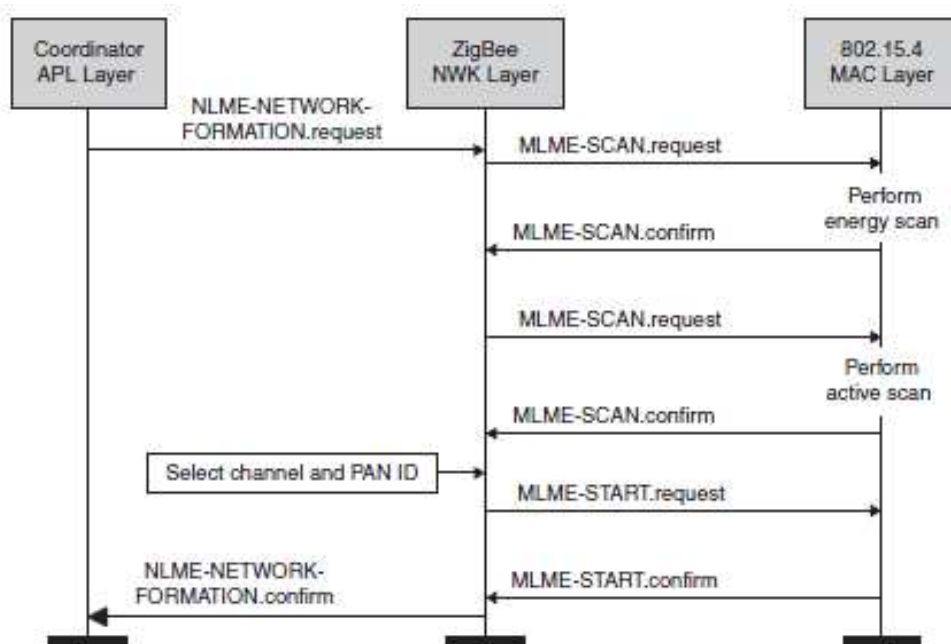
Ovako novoformirana mreža i nije od nekog značaja jer se u njoj nalazi samo jedan uređaj, a to je ZC. U procesu formiranja mreže, jedan paket se šalje na svaki kanal, ovaj paket predstavlja *MAC active scan* ili *beacon* zahtev, ukoliko na kanalu ne postoji nijedna druga mreža ovo će biti jedini paket na tom kanalu.

Zadaci ZC su:

- Formiranje mreže
- Biranje kanala po 15.4 standardu na kojem će mreža operisati
- Definisanje proširenog i kratkog identifikatora *PAN ID* mreže
- Biranje profila steka
- Deluje kao *Trust Center* za sigurnosne aplikacije i mreže
- Arbitruje vezivanje *ZED* u mrežu
- Ponaša se kao usmerivački uređaj za *mesh* rutiranje

Prilikom formiranja mreže posebna pažnja se obraća na biranje kanala koji je najmanje opterećen, kako bi se smanjila mogućnost pojave interferencije ili šuma. Ovo se obavlja tako što se proziva energetska osmatranje (engl. *energy scanning*), što predstavlja računanje prosečnog broja paketa koji se šalje na određenom kanalu u vremenu.

Ovo nije jedna od najpouzdanijih metoda za određivanje zauzetosti kanala jer se posmatra samo jedan kratak trenutak u vremenu. Nakon ove metode proziva se aktivno osmatranje (engl. *active scanning*), ono šalje prazne *beacon* zahteve na koje mreže na tom kanalu odgovaraju ukoliko postoje. Na ovaj način se izbegava da se kreiraju dve mreže sa istim *PAN ID* identifikatorom. Ceo proces formiranja mreže prikazan je slikom 2.4.3.



Slika 2.4.3 – Proces formiranja ZB mreži

Proces pridruživanja mreži rezervisan je za *ZR* i *ZED* uređaje, zadaci *ZR* uređaja su:

- Pronalaženje i pridruživanje mreži
- Prosleđuju podatke putem emitovanja kada se to zahteva
- Učestvuju aktivno u rutiranju, otkrivanju i održavanju putanja
- Omogućuju drugim uređajima da se pridruže mreži ukoliko je to predviđeno
- Čuvanje podataka uređaja koji su u *sleep* režimu, sa kojima ostvaruju *parent* odnos

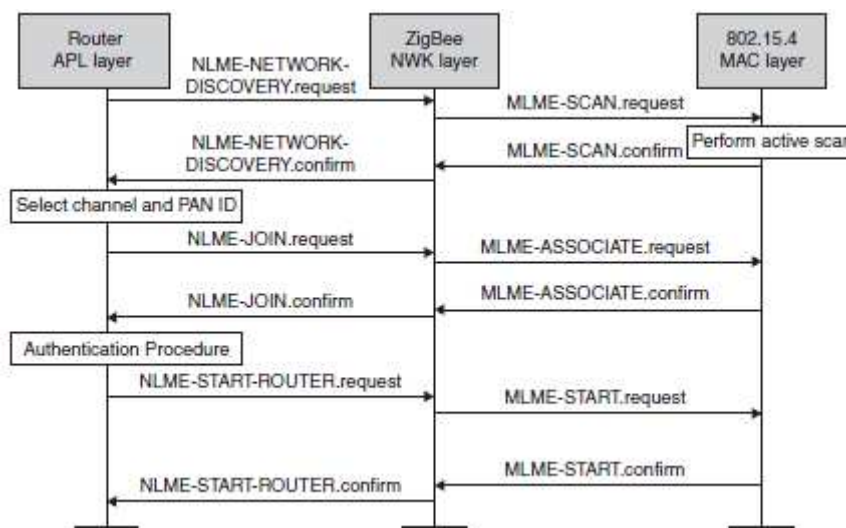
Dok *ZED* uređaji imaju zadatke da:

- Pronalaženje i pridruživanje mreži
- Proveravanje da li uređaji sa kojima ostvaruju *parent-child* odnos imaju pristiglih poruka, koje su pristigle u trenutku kada je uređaj bio u *sleep* režimu
- Pronalaženje novog uređaja koji će predstavljati *parent* ukoliko je prethodni izgubljen ili nedostupan

- Postavljanje u *sleep* režim kad god je to moguće kako bi se očuvala baterija

Proces pridruživanja mreži zasniva se na pronalaženju mreža odnosno uređaja koji su u dometu, a potom biranju jednog od dostupnih, ukoliko je prethodno omogućeno pridruživanje mreži od strane koordinatora. Nakon pridruživanja u mrežu novi uređaj dobija adresu kako bi se mogao adresirati i komunicirati sa ostalim uređajima u mreži.

Kao i kod procesa formiranja mreže tako i u procesu pridruživanja mreži prvo se proziva *active scan* procedura, ova procedura prosleđuje poruku svim kanalima tj. *Beacon request*. Na ovu poruku odgovaraju uređaji i te poruke se sakupljaju i procesiraju na uređaju koji se pokušava priključiti mreži. Pored *PAN ID* identifikatora pomoću kojih se razlikuju mreže potrebno je i razlikovati uređaje što se vrši pomoću *NwkAddr*. Uređaj koji se pridružuje mreži se zapravo povezuje sa nekim od uređaja (sa *ZR* ili *ZC*) stoga je bitno izabrati uređaj po *NwkAddr* i samu mrežu. U profilima kućne automatizacije neophodno je i da se odobri pridruživanje u mrežu, emitovanjem *permit join request*. Ukoliko su svi zahtevi ispunjeni sem ovog uređaj se neće moći priključiti mreži. Nakon što su izabrani uređaj i mreža na koju je neophodno povezati se, uređaj dobija privremenu adresu u mreži. Ova privremena adresa će mu biti stvarno dodeljena ukoliko se uspešno poveže na mrežu, u ovoj fazi uređaj ne može da komunicira sa ostalim uređajima u mreži sem onog preko kojeg se pokušava povezati u mrežu. Procedura koja sledi naziva se *authentication procedure*. Celokupna procedura pridruživanja u mrežu prikazana je slikom 2.4.4



Slika 2.4.4 – Procedura pridruživanja mreži

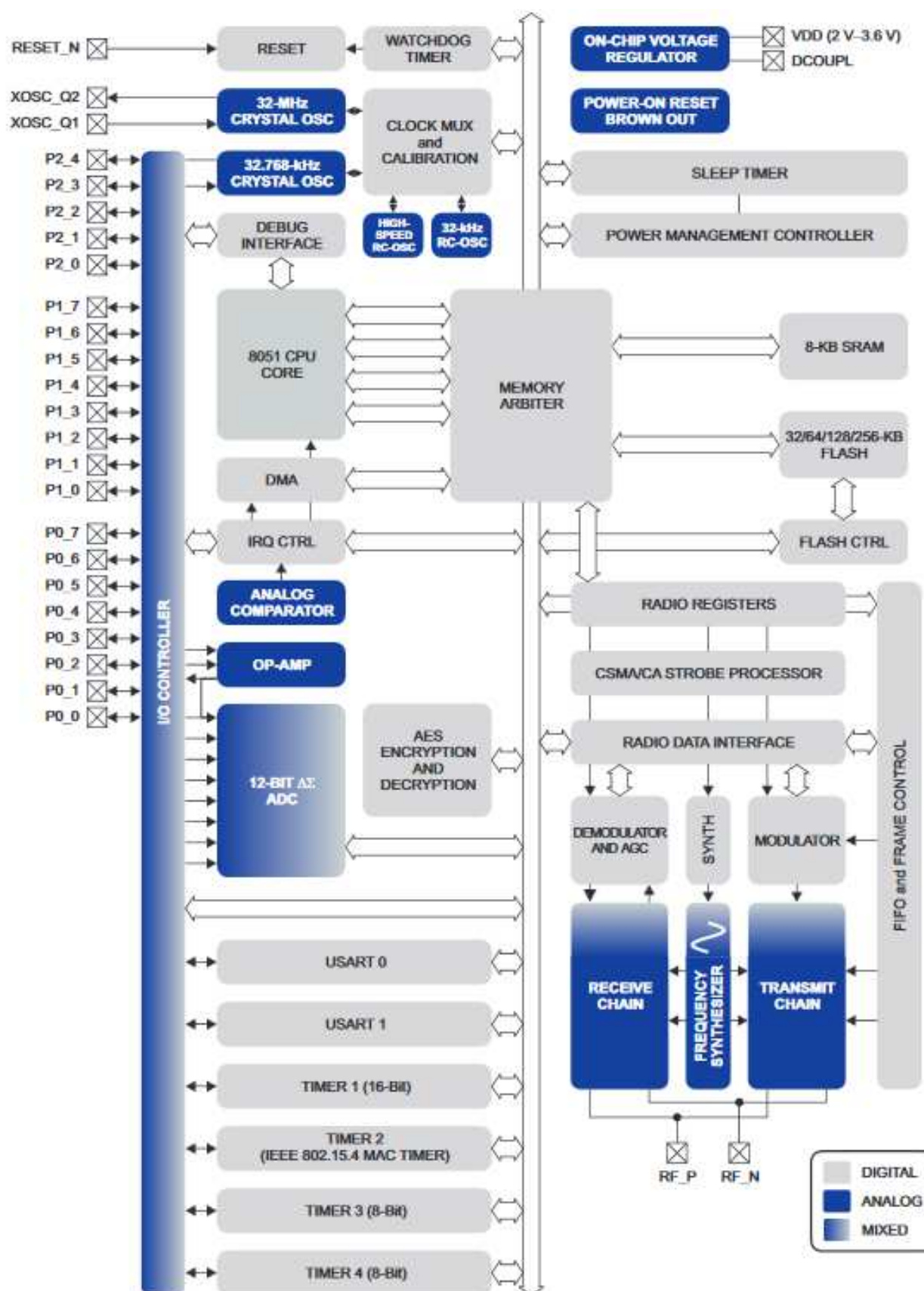
Ponovno pridruživanje mreži je procedura koja se koristi kada je uređaj bio uspešno povezan u mreži, međutim zbog rekonfiguracije mreže on je izgubio *parent* uređaj iz dometa, ili je taj uređaj iz bilo kog drugog razloga nedostupan. *ZED* uređaji uvek komuniciraju sa svojim *parent* uređajem, a ukoliko je on nedostupan potrebno je pronaći novi uređaj koji će preuzeti tu ulogu. Ova procedura je slična prethodno opisanoj samo što prilikom ponovnog pridruživanja nije neophodno da je prevashodno emitovan *permit join request*. Nakon što se emituje *beacon request* analiziraju se pristigli odgovori i bira se mogući *parent* uređaj, koji je podrazumeva se na mreži sa istim *PAN ID* identifikator. Nakon uspešnog povezivanja emituje se *device-announce* tj. obaveštenje da je došlo do pomeranja čvora. Ovaj poslednji korak je veoma važan kako bi koordinatorski i ruterski uređaji bili obavešteni o promeni izgleda stabla mreže što direktno može da utiče na postojeće linije rutiranja kao i na nove moguće koje će tek biti kreirane. Procedura ponovnog pridruživanja u mrežu ilustrovana je na slici 2.4.5..

2.5 CC2530

Fizička arhitektura PIR uređaja zasnovana je na mikrokontroleru *Texas Instruments* kompanije pod oznakom *CC2530*. Na slici 2.5.1 prikazana je unutrašnja organizacija modula u SoC-u. Ovaj mikrokontroler se koristi u aplikacijama gde se implementira *ZigBee*, *IEEE 802.15.4* standard i u *RF4CE* aplikacijama. Zasnovan je na 8051 arhitekturi mikročipova. Pored osnovnih periferijskih blokova poput tajmera, *PWM modula*, *AES* podrške, generator slučajnih brojeva, *DMA*, *WDT*, *UART* i *SPI* modula sadrži i 12-bitni *ADC* konvertor sa 8 kanala što je neophodno za očitavanje informacija sa PIR senzora.

Veoma bitan modul je *RF* modul, koji je nezavisni deo ovog sistema, što omogućava da se paralelno sa radom kontrolera obavlja komunikacija i upravlja bežičnim prenosom podataka. Radiofrekvencijski modul čine sledeći podmoduli:

- Automat konačnih stanja (*FSM*)
- Modulator – kao i demodulator transformiše podatke u oblik pogodan za manipulaciju
- Demodulator
- Filtriranje ramova i prepoznavanje izvora
- Radio *RAM* – skladištenje podataka u FIFO registre, gde se jedan registar koristi za primljene podatke, a drugi za podatke koje se šalju

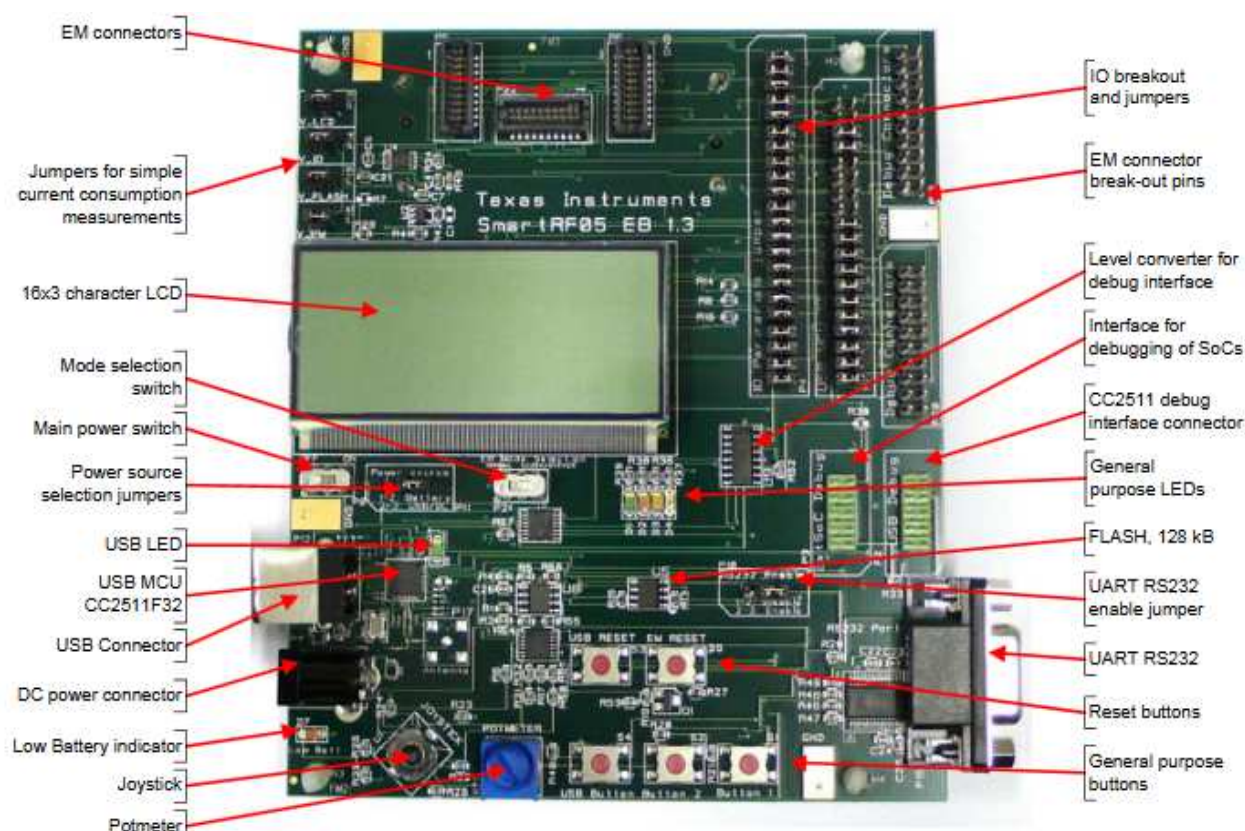


Slika 2.5.1 – Unutrašnja organizacija modula na CC2530 SoC-u

Velika prednost je mogućnost kontrole potrošnje energije tj. postavljanja u režime niske potrošnje (*engl. Power modes*) što ga čini pogodnim za aplikacije gde se koristi baterijsko napajanje, odnosno za primenu kod ZED uređaja. Za razliku od aktivnog režima gde svi delovi mikrokontrolera rade, u ostalim režimima uštede energije gase se pojedine periferije. Srazmerna broju ugašenih periferija je i potrošnja energije. Režimi rada mikrokontrolera prema specifikaciji su sledeći:

- aktivni režim ($\sim 24\text{--}29\text{ mA}$)
- PM1 ($\sim 0,2\text{ mA}$)
- PM2 ($\sim 1\text{ uA}$)
- PM3 ($\sim 0,4\text{ uA}$)

S obzirom da je bilo neophodno testirati projektovani sistem pre izrade štampane ploče, korišćeno je razvojno okruženje sa oznakom *SmartRF05EB Rev. 1.3* čiji izgled je dat slikom 2.5.2. Iako je ovo okruženje prva verzija razvojne ploče koja je puštena u prodaju od strane TI kompanije, ona je odlično poslužila za verifikaciju svih neophodnih segmenata PIR uređaja. Ova ploča dolazi u kombinaciji sa modulom *CC2530EM*. Mikrokontroler na ovom modulu je identičan onom koji je korišćen na štampanoj ploči krajnjeg uređaja.



Slika 2.5.2 – Prikaz razvojnog okruženja SmartRF EB Rev. 1.3

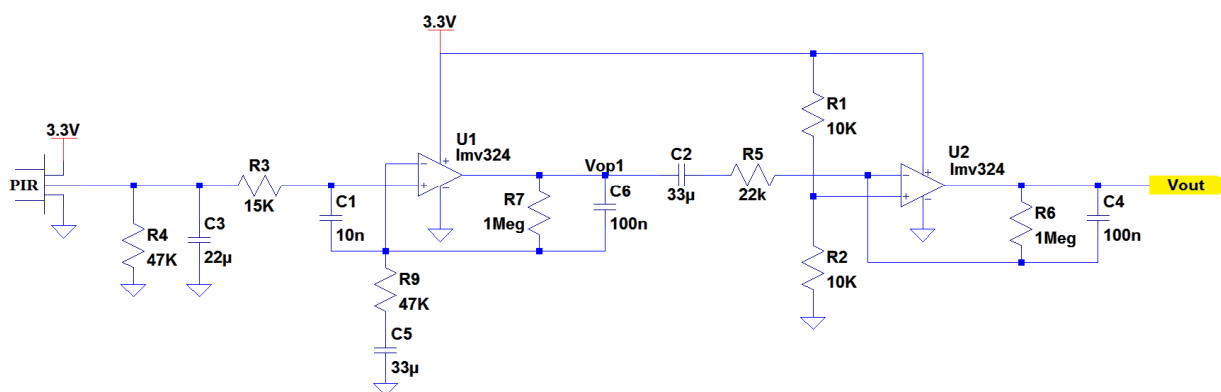
2.5.1 Organizacija memorije

Kod 8051 arhitekture razdvojene su programska memorija i memorija za podatke, a razlikuju se četiri sekcije memorije koje su opisane ispod.

- **CODE** – predstavlja memoriju kod koje je moguće samo iščitavanje (engl. *read-only*). Veličina ove memorijske sekcije je 256 KB.
- **DATA** – je memorija u koju je dozvoljeno pisanje, a moguće je i iščitavanje (engl. *read-or-write*). Pristup ovoj memoriji moguć je direktnim ili indirektnim pristupom (ova podsekcija memorije označena je sa **IDATA**) uz pomoć instrukcije u trajanju od jednog ciklusa. Veličina ove memorijske sekcije je 256 bajta, gde je pristup prvih 128 bajta moguć samo indirektnim putem.
- **XDATA** – takođe predstavlja *read-or-write* memoriju, koja u proseku zahteva između četiri do pet ciklusa instrukcija za pristup. Veličina ove sekcije je 64 KB, a pristup njoj je sporiji od pristupa **DATA** sekciji. Razlog je to što **CODE** i **XDATA** sekcije dele jednu magistralu.
- **SFR** – pristup ovoj sekciji memorije je moguć direktno pomoću instrukcija koje traju jedan ciklus. Veličina ove sekcije koja ujedno predstavlja registarski prostor je 128 bajta. Pristup je moguć u vidu pisanja i čitanja, a registri čije adrese su deljive sa osam imaju mogućnost adresiranja na nivou individualnih bita.

2.6 Fizička arhitektura PIR uređaja

Iako je glavni zadatak ovog rada projektovanje programske podrške, ona ne bi bila moguća bez prapratne fizičke arhitekture. Kao zaseban rad izučavana je optimalna arhitektura koja bi zadovoljila potrebe *ZigBee* komunikacije kao i obrade signala sa PIR senzora. Jedno rešenje za obradu signala sa PIR senzora do koga se došlo predstavljeno je i na slici 2.6.1.



Slika 2.6.1 – Električno kolo za povezivanje PIR senzora

Na datoj električnoj shemi može se videti sam PIR senzor koji pored napajanja poseduje jednu liniju kojom se šalje signal o trenutnom stanju detekcije. Ovaj signal je prilično niske vrednosti stoga ga je potrebno pojačati. U tu svrhu se koriste dva pojačavačka stepena. Na posletku tako pojačan signal dolazi na pin mikrokontrolera koji je konfigurisan kao ulaz na *ADC* konvertor.

PIR senzor detektuje promene IR zračenja, tj. promene u temperaturi. Veličina IR izvora ima značaja uticaj na osetljivost senzora, veći izvor zrači većom energijom i senzor ga lakše detektuje. Takođe senzor je osetljiviji ako je razlika temperatura između izvora i ambijenta veća. Jedna od važnih karakteristika ove komponente je i opseg napona napajanja, jer on ima uticaja na izlazni signal. Izlazni signal je stabilan na polovini napona napajanja kada nema detekcije. Kada PIR senzor detektuje promenu u IR zračenju, izlazni signal zaosciluje oko te stabilne vrednosti.

Oko samog senzora nalazi se Fresnelovo sočivo koje služi da usmeri zrake IR zračenja na PIR senzor, čime se poboljšava osetljivost, domet detekcije, kao i detekcioni (ugaoni) opseg sistema koji grade ove dve komponente.

3. Programska podrška

3.1 Koncept rešenja

Texas Instruments kompanija izdaje primer aplikacija kao i *ZigBee* stek, tako je za izradu referentnog programskog rešenja nadograđena već postojeća aplikacija. Kao polazna tačka referentnog koda korišćen je i kod scenskog kontrolera *OBLO Living* kompanije.

3.2 TI OSAL Task system

OSAL Task system predstavlja *framework* za jednostavno implementiranje aplikativnih funkcionalnosti u *TI ZB* steku. Ceo sistem se zasniva na podeli svih zadataka u određene zadatke (engl. *Tasks*). Na ovaj način se svim zadacima mogu dodeliti prioriteti, a mogu se definisati i stepeni prioriteta. Ovakav sistem pogoduje i načinu implementacije koji je izabran, a to je korišćenjem mašine cikličnih stanja. Nakon inicijalizacije određenog taska i dodavanjem istog u cikličnu petlju koja rukuje svim taskovima, definišu se zadaci taska. Uvođenjem ovakvog sistema inženjer koji piše aplikativni deo programske podrške je rasterećen od kompleksnosti celog sistema

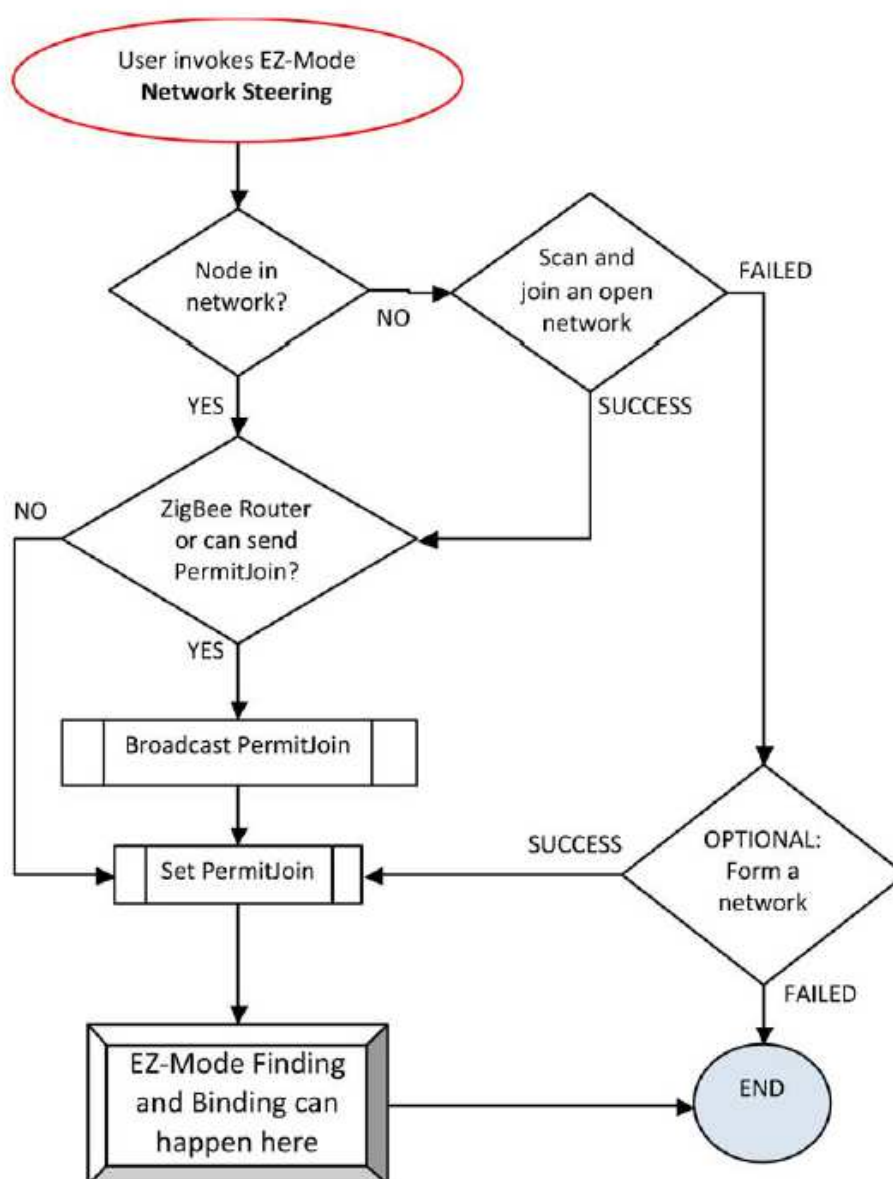
U funkciji *osalInitTasks(void)* koja je tipa *void* mogu se videti svi taskovi kao i njihovi identifikatori. Task koji je dodat je *PIR_Init*. Upravo ovaj task vodi brigu o *ZigBee* delu *PIR* senzora, odnosno uz okidanje određenih događaja poziva procedure izveštavanja (engl. *Reporting*), proceduru *IAS Notification*, ili slanje komandi za uključivanje ili isključivanje.

PIR_Init funkcija inicijalizuje programsku memoriju koja će biti korišćena za čuvanje promenljivih, registruje attribute svih klastera na svim krajnjim tačkama (engl. *Endpoints*), vrši registraciju komandi, podešavanje snage *RF* prenosa i vrši inicijalizaciju svih promenljivih na početne vrednosti.

Nakon ove inicijalizacije, funkcija *PIR_task* metode je da rukuje *ZB* porukama i koordiniše prelazima stanja u mašini konačnih stanja, kao i da upravlja događajima koji se okidaju na pritiske tastera.

3.3 *EZ-Mode* procedura

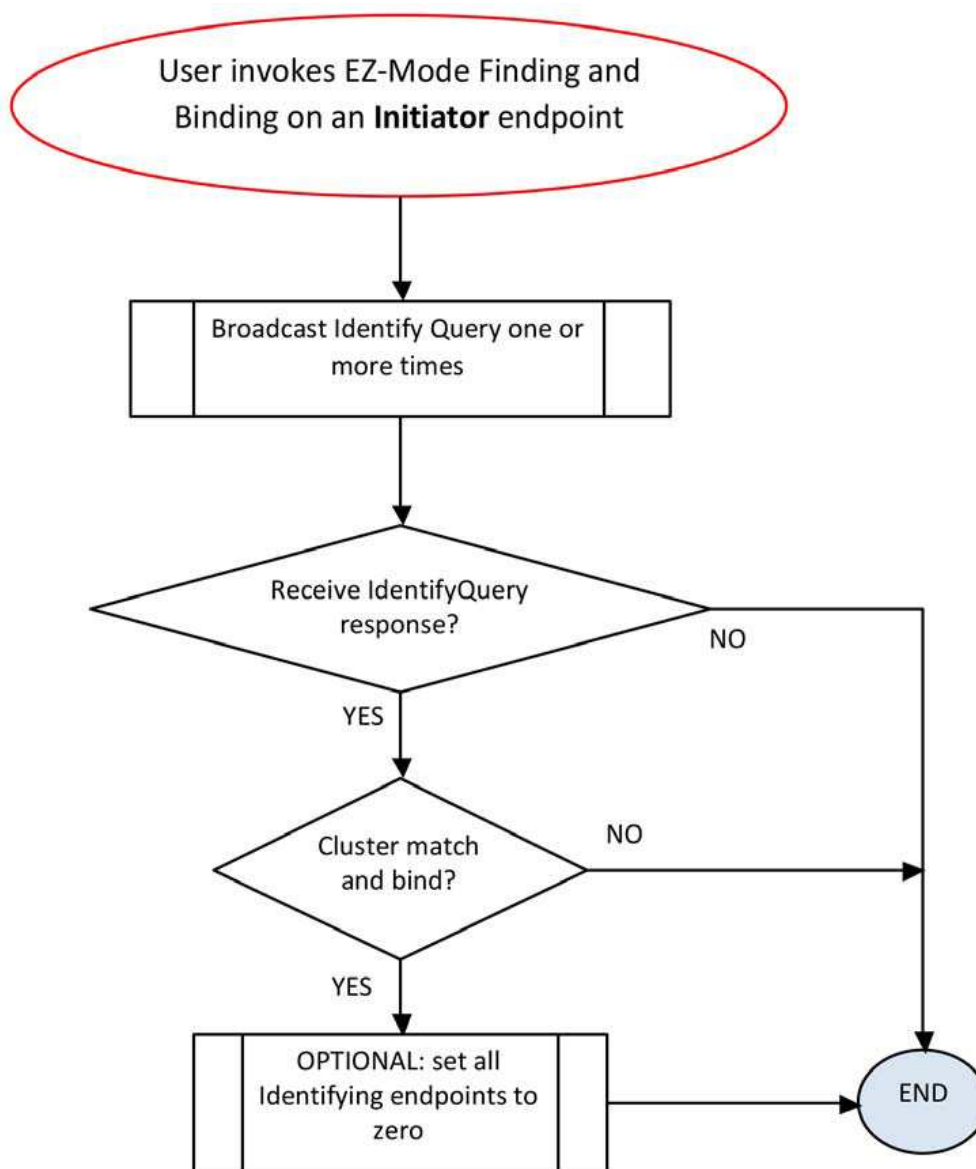
EZ-Mode predstavlja puštanje u rad uređaja uz interakciju korisnika ili nekog drugog događaja. Pri pozivu ove procedure uređaj prolazi kroz razna stanja mašine konačnih stanja u zavisnosti od trenutne konfiguracije. Tako na primer ukoliko uređaj nije na mreži pokretanjem *EZ-Mode* procedure uređaj će pokušati da se pridruži mreži, ukoliko ovo ne bude uspešno formiraće mrežu ukoliko je to moguće. Ukoliko je uređaj ruterski ili koordinatorski pokretanjem



Slika 3.3.1 – Blok dijagram *EZ-Mode* procedure

ove procedure podesiće se *PermitJoin* vrednost, odnosno omogućiće se povezivanje na mrežu u trajanju predefinisano vremenskog intervala. Na ovaj način se izvršavaju prelasci stanja u mašini konačnih stanja što je i prikazano blok dijagramom na slici 3.3.1.

Pri proceduri bajndovanja pomoću *EZ-Mode* pri čemu je posmatrani uređaj inicijator što je uvek slučaj sa *PIR* uređajem (jer on ne poseduje fizičke mogućnosti da realizuje bilo kakve komande jer ne poseduje aktuator, već može samo da izdaje komande drugim uređajima) mogu se uočiti koraci opisani blok dijagramom 3.3.2.



Slika 3.3.2 – Blok dijagram *EZ-Mode* procedure kod inicijatora

Na početku procedure emituje se *Identify query* jednom ili više puta u zavisnosti od konfiguracije mreže, a potom se čeka odgovor. Ukoliko odgovor pristigne znači da postoji uređaj koji je moguće uvezati (bajndovati), međutim to još uvek ne određuje da će uvezivanje biti moguće. Nakon ovog koraka sledi najvažniji korak u ovoj proceduri, a to je analiziranje podržanih klastera. Naime da bi se dva uređaja mogla uvezati potrebno je da oba podržavaju isti

klaster, sa ograničenjem da na jednom uređaju taj klaster mora biti na serverskoj strani, dok na drugom mora biti na klijentskoj strani.

Ukoliko se ispuni i ovaj uslov i nakon analize *cluster match* odgovora utvrdi da postoje mogućnosti za uvezivanje, uređaji će se uvezati i inicijator će dodati metu uređaj u svoju bajnding tabelu. Nakon toga se sve promenljive koje nisu potrebne deinicijalizuju.

4. Programsko rešenje

Analizom koja je sprovedena u prethodnim poglavljima došlo se do aplikacije u kojoj su implementirane aplikativne funkcionalnosti uređaja. Aplikativne funkcionalnosti se sastoje iz:

- Podсистема za interakciju sa korisnikom
- Mehanizma izveštavanja
- Pomoćne i ZB koordinirajuće funkcije

4.1 Podсистem za interakciju sa korisnikom

Podсистem za interakciju sa korisnikom realizovan je fizički pomoću jednog tastera i dve LED diode koje služe za signalizaciju korisniku. Sprega tastera i dioda sa aplikativnim slojem realizovana je preko HAL nivoa. Prilikom pritiska na taster poziva se funkcija pod nazivom *PIR_ProccessModeKeyPress*. Ova funkcija upoređuje dužinu pritiska kao i broj pritisaka na taster, a potom u zavisnosti od ova dva parametra okida određeni događaj ili poziva određenu proceduru. Pritisak na taster može biti dug što je definisano u kodu kao *HAL_KEY_LONG_PRESS* ili kratak *HAL_KEY_SHORT_PRESS*. Ukoliko je pritisak kratak, moguće je da bude niz kratkih pritisaka pa je za njihovu analizu neophodno pozvati proceduru u kojoj se vrši ova analiza. Ta procedura se poziva okidanjem događaja koji je definisan u kodu kao *PIR_MODESW_CONSECUTIVE_PRESS_EVT*. Nakon što se okine ovaj događaj poziva se funkcija pod nazivom *PIR_ProccessModeSWConsecutivePress*. Ova funkcija pomoću mehanizma koji uključuje tajmer broji pritiske na taster, definisano je vreme koje sme da protekne između dva pritiska i na taj način se računa broj pritiska i detektuje završetak niza pritisaka. Nakon poslednjeg pritiska kada tajmer pretekne, poziva se događaj koji koordinira sa datim brojem pritisaka.

Vrednosti koje su u korelaciji sa brojem pritisaka tastera i dužinom pritiska date su tabelom 4.1.1.

Događaj koji se okida	Broj pritisaka
JOINING_NWK_CNT	1
PIR_MODESW_CNT_ONOFF_EZMODE_INITIATOR_CNT	2
PIR_MODESW_CNT_OCC_EZMODE	3
PIR_MODESW_TEST_INVOKE_OCC_DETECTION	4
PIR_MODESW_CNT_FACTORY_RESET	10
Funkcija koja se poziva	Dužina pritiska (s)
<i>StartCoordinator()</i>	5+

Tabela 4.1.1 – Broj pritisaka tastera u korelaciji sa događajima

Prilikom okidanja *JOINING_NWK_CNT* događaja poziva se procedura pridruživanja na mrežu ukoliko uređaj nije na mreži. Ukoliko je uređaj koordinator i ukoliko nije omogućeno pridruživanje u mrežu pritiskom na taster jednom kratko omogućuje se ulazak u mrežu (emitovaće se *Permit Joining*). Prilikom ulaska u mrežu koristi se *IAS_ENDPOINT* sa ulaznim i izlaznim klasterima koji su definisani za ovaj *endpoint*.

Okidanjem *PIR_MODESW_CNT_SWITCH_EZMODE_INITIATOR_CNT* pokrenuće se EZ-Mode procedura bajndovanja ukoliko je uređaj na mreži, međutim ukoliko uređaj nije na mreži on će pritiskom na ovo dugme pokušati da uđe u mrežu. Bitno je napomenuti da se sve ove funkcije obavljaju korišćenjem *ONOFF_ENDPOINT*, uz ulazne i izlazne klastere koji su definisani.

Na dugi pritisak tastera, što se znači držanje tastera pritisnutog duže od pet sekundi poziva se funkcija *StartCoordinator*. Ova funkcija služi za postavljanje uređaja u *standalone* režim, odnosno uređaj postaje koordinator. Ukoliko je uređaj prethodno bio povezan na mrežu potrebno ga je restartovati kako bi se deinicijalizovale sve korišćene promenljive i obrisale eventualne bajnding tabele.

Okidanjem *PIR_MODESW_CNT_OCC_EZMODE* pokreće se *EZ-Mode* procedura pomoću koje uređaj ukoliko nije na mreži pokušava da uđe korišćenjem *OCC_ENDPOINT* koji je definisan *Occupancy Sensing* klasterom.

Posebnu pažnju treba obratiti na događaj za ispitivanje i verifikaciju pod nazivom *PIR_MODESW_TEST_INVOKE_OCC_DETECTION*. Ovaj događaj se okida na četiri kratka pritiska tastera i simulira detekciju *PIR* senzora, na taj način se uređaj može verifikovati i bez stvarnog senzora.

Kod interakcije korisnika veoma je bitno da korisnik dobija povratnu informaciju o uspešnosti nekih radnji, zbog toga su dodate dve svetleće diode jedna u crvenoj, a druga u zelenoj boji. Tabelom 4.1.2 data su ponašanja *LED* indikacije u određenim stanjima i pri određenim pritiscima na taster.

Pritisak tastera	Aktivnost	LED indikacija
	Uređaj nije na mreži	Crveno
1x kratak	Pridruživanje <i>ZigBee</i> mreži	Crveno treperenje
2x kratak	Uparivanje kao inicijator	Zeleno treperenje
4x kratak	Simulacija detekcije senzora	1x zeleni treptaj
10x kratak	Vraćanje na fabrička podešavanja	Crveno
Dugačak pritisak (5+ sekundi)	Kreiranje sopstvene <i>ZigBee</i> mreže	Naizmenično crveno i zeleno

Tabela 4.1.2 – Ponašanje *LED* indikacije

Kako bi ovakav način obaveštavanja bio moguć svetlećim diodama implementirane su funkcije *HalLedSet* i *HalLedBlink*. Ove funkcije omogućavaju postavljanje određene svetleće diode u adekvatno stanje prosleđivanjem maske pinova korespondirajućih dioda (*HAL_LED_GREEN*, *HAL_LED_RED* ili *HAL_LED_ALL*) i jednog od 6 mogućih režima (*HAL_LED_MODE_OFF*, *HAL_LED_MODE_FLASH*, *HAL_INFINITE_FAST_FLASH*, *HAL_LED_MODE_BLINK*, *HAL_LED_MODE_TOGGLE* i *HAL_LED_MODE_ON*).

4.2 Mehanizmi izveštavanja

Postoje dva mehanizma za izveštavanje o detekciji pokreta, o tome koji će se mehanizam koristiti odlučuje način ulaska u mrežu, odnosno koji *endpoint* je korišten. Ukoliko se u mrežu ušlo pomoću *IASZone* klastera koristiće se *IASZone* notifikacija, koja izveštava korišćenjem *ZoneStatus* atributa koji je prethodno opisan (tabela 2.2.7).

Ukoliko je inicijalizacija *IASZone* klastera uspešna procedura obaveštavanja se može pokrenuti pozivom funkcije *zclSS_IAS_Send_ZoneStatusChangeNotificationCmd* (slika 5.1.1). Parametri funkcije *zclSS_IAS_Send_ZoneStatusChangeNotificationCmd* dati su tabelom 4.2.1.

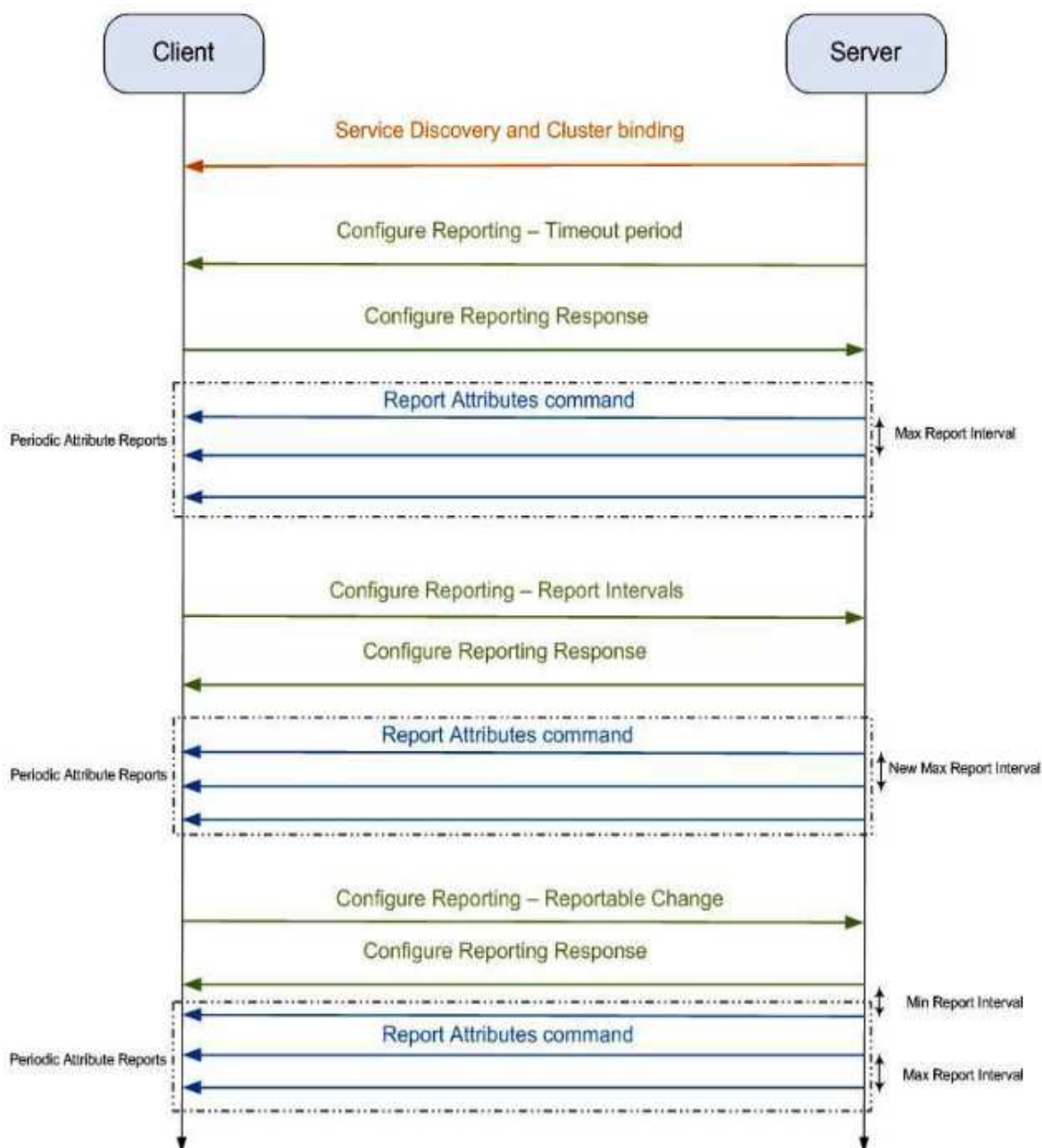
Tip	Parametar	Vrednost parametra
<i>uint8</i>	<i>srcEP</i>	IAS_ENDPOINT
<i>afAddrType_t*</i>	<i>dstAddr</i>	&CIE_addr
<i>uint16</i>	<i>zoneStatus</i>	zclPIR_ZoneStatus
<i>uint8</i>	<i>extendedStatus</i>	0x00
<i>uint8</i>	<i>zoneID</i>	zclPIR_ZoneId
<i>uint16</i>	<i>delay</i>	0
<i>uint8</i>	<i>disableDefaultRsp</i>	1
<i>uint8</i>	<i>seqNum</i>	ZCL_TRANS_SEQ_NUM()

Tabela 4.2.1 – Parametri funkcije *zclSS_IAS_Send_ZoneStatusChangeNotificationCmd*

Drugi način izveštavanja je korišćenjem komandi za izveštavanje atributa tzv. *Report Command*. Ukoliko postoje atributi koje je moguće izveštavati u mrežu i ukoliko je način izveštavanja uspešno konfigurisan, onda je moguće proizvesti izveštaj (engl. *report*). U realizaciji ovog uređaja definisano je da se posmatra promena atributa *Occupancy* (tabela 2.2.3) i ukoliko je došlo do promene stanja kreira se novi izveštaj. Ukoliko uređaj kojem su namenjene ove poruke i koji ih parsira ne podržava klaster pod kojim su smešteni atributi koji se izveštavaju onda će te poruke biti ignorisane, a da bi se sprečilo slanje nepotrebnih poruka u mrežu u referentnom rešenju implementirano je da se *reporting* komande ne izvršavaju osim u slučaju kada se u mrežu ušlo korišćenjem *Occupancy* klastera. Tipična sekvenca razmene paketa između servera i klijenta pri izveštavanju prikazana je slikom 4.2.1.

Pri registraciji atributa za izveštavanje kreiran je poseban događaj (engl. *Event*) pod nazivom *ATTRIBUTE_REPORT_EVT*. Ovaj događaj se proziva na svakih 1000 *ms*, a on proziva funkciju *zclReport_CheckConfig*. Ova funkcija je zadužena za proveru svih reportabilnih atributa i prozivanju njihovih arbitražnih funkcija kako bi se ustanovilo da li ima potrebe za novih izveštajem.

Za oba ova mehanizma implementirane su vremenske zaštite koje su podesive i definišu dužinu trajanja signala detekcije na mreži, kao i period nakon detekcije koji mora da protekne kako bi uređaj ponovo bio u stanju da detektuje novi pomeraj. U svrhe ispitivanja dužina trajanja signala detekcije je dve sekunde, nakon čega se postavlja u stanje u kojem ne može doći do detekcije koje traje tri sekunde.



Slika 4.2.1 – Tipična sekvenca razmena paketa pri izveštavanju

U posebnoj strukturi sa oznakom *zclConfigReportRec_t* podešavaju se parametri atributa o kojem se izveštava, opis strukture dat je tabelom 4.2.2.

Tip	Parametar	Dodeljena vrednost
<i>uint16</i>	<i>clusterId</i>	<i>ZCL_CLUSTER_ID_MS_OCCUPANCY_S ENSING</i>
<i>uint16</i>	<i>timeup</i>	0x00
<i>uint8 *</i>	<i>lastReportValue</i>	<i>PIR_OccupancyLastReportValue</i>

<i>bool*</i>	<i>(* arbitrator)(struct zclConfigReportRec const *const cfg, zclAttrRec_t const *const attrRec);</i>	<i>zclReport_digital_arbit</i>
<i>zclResendParams_t</i>	<i>resend</i>	{3,0,0,0}
<i>zclCfgReportRec_t</i>	<i>cfgReportRec</i>	Tabela 4.2.3

Tabela 4.2.2 – Prikaz *zclConfigReportRec_t* tipa strukture

Posebnu pažnju treba obratiti na arbitražnu funkciju, za koju je odabrana funkcija *zclReport_digital_arbit* koja se nalazi u *zcl_reporting*. Ona je izabrana jer je posmatrani atribut jedan bit u osmobitnoj mapi, i može da ima binarne vrednosti. Zadatak ove funkcije je da posmatra da li je došlo do promene promenljivu koja čuva trenutnu vrednost u odnosu na promenljivu koja sadrži vrednost pre promene (*PIR_OccupancyLastReportValue*). Ukoliko je došlo do promene ova funkcija vraća pozitivnu vrednost (*TRUE*) u suprotnom negativnu vrednost (*FALSE*).

U tabeli 4.2.3 prikazane su vrednosti struktura tipa *zclCfgReportRec_t* sa nazivom *cfgReportRec*.

Tip	Parametar	Dodeljena vrednost
<i>uint8</i>	<i>direction</i>	0x00
<i>uint16</i>	<i>attrID</i>	<i>ATTRID_MS_OCCUPANCY_SENSING_CONFIG_OCCUPANCY</i>
<i>uint8</i>	<i>dataType</i>	<i>ZCL_DATATYPE_BITMAP8</i>
<i>uint16</i>	<i>minReportInt</i>	0x0000
<i>uint16</i>	<i>maxReportInt</i>	0xFFFE
<i>uint16</i>	<i>timeoutPeriod</i>	0
<i>uint8*</i>	<i>reportableChange</i>	NULL

Tabela 4.2.3 – Prikaz *zclCfgReportRec_t* tipa strukture

Adekvatnim podešavanjem *minReportInt* i *maxReportInt* omogućeno je da se izveštava nakon isteka *maxReportInt* čak i ukoliko ne dođe do promene atributa. On je podešen na 0xFFFE vrednost (u decimalnoj vrednosti to je ekvivalentno broju 65534) kako bi se izbegao

poseban slučaj pri vrednosti od 0xFFFF koji sa *minReportInt* postavljenoj na nuli onemogućava slanje izveštaja, a pritom obezbedilo maksimalno vreme.

4.3 Pomoćne i ZB koordinirajuće funkcije

Lista implementiranih funkcija PIR uređaja data je tabelom 4.3.1, ove funkcije se koriste za kreiranje, održavanje i terminaciju ZB mreža kao i za pomoćne procese (led indikacija, funkcije detekcije pritisaka tastera, kontrolu uređaja, inicijalizaciju).

Tip	Naziv funkcije	Parametri
<i>void</i>	<i>PIR_Init</i>	<i>byte</i> task_id
<i>uint16</i>	<i>PIR_event_loop</i>	<i>byte</i> task_id, <i>uint16</i> events
<i>void</i>	<i>PIR_ResetConfigParameters</i>	<i>void</i>
<i>void</i>	<i>PIR_SetLEDIndication</i>	<i>uint8</i> LEDnum, <i>LedAction_t</i> action
<i>void</i>	<i>PIR_ProcessLEDIndication</i>	<i>void</i>
<i>void</i>	<i>PIR_ProcessZDOMsgs</i>	<i>zdoIncomingMsg_t</i> *inMsg
<i>void</i>	<i>PIR_ProcessZDOStateChange</i>	<i>zdoIncomingMsg_t</i> *inMsg
<i>void</i>	<i>PIR_ProcessIncomingMsg</i>	<i>zclIncomingMsg_t</i> *msg
<i>void*</i>	<i>PIR_Process_nwk_leave</i>	<i>void</i>
<i>uint8</i>	<i>PIR_ValidateAttributeChange</i>	<i>uint8</i> endpoint, <i>zclAttrRec_t</i> *pAttr, <i>zclWriteRec_t</i> *pAttrInfo
<i>void</i>	<i>PIR_ProccessModeKeyPress</i>	<i>uint8</i> pressLenth
<i>void</i>	<i>PIR_ProccessModeSWConsecutivePress</i>	<i>void</i>
<i>void</i>	<i>PIR_StartCoordinator</i>	<i>void</i>
<i>void</i>	<i>PIR_ProcessIdentifyTimeChange</i>	<i>void</i>
<i>void</i>	<i>PIR_SendEnrollReq</i>	<i>afAddrType_t</i> *pAddr
<i>ZStatus_t</i>	<i>zclPIR_EnrollResponseCommand</i>	<i>zclZoneEnrollRsp_t</i> *pRsp
<i>afStatus_t</i>	<i>PIR_IAS_ZoneMatchDescReq</i>	<i>uint16</i> addr
<i>void</i>	<i>PIR_EnrollProcess</i>	<i>dwsEnrollEvent_t</i> event, <i>void</i> *pData

Tabela 4.3.1 – Lista implementiranih funkcija PIR uređaja

Funkcija koja je zadužena za inicijalizaciju uređaja je *PIR_Init*. Ona iščitava parametre upisane u *flash* memoriju, priprema uređaj za ulazak u mrežu na pritisak tastera, pokreće sistem taskova. Ciklično se poziva *PIR_event_loop* koja sadrži listu taskova i listu u kojoj su upisani prioriteta taskova, na osnovu trenutnog taska i prioriteta odlučuje se koji će se sledeći task prozvati za izvršavanje. Resetovanje konfiguracionih parametara obavlja se pozivom *PIR_ResetConfigParameters*. Kako bi se lako obezbedila mogućnost podešavanja *LED* indikacije implementirana je funkcija pod nazivom *PIR_SetLEDIndication*. Njoj se prosleđuje akcija koju je potrebno da indikacija izvrši kao i broj svetleće diode kojom se upravlja ukoliko postoji više. Da bi ovo bilo moguće ciklično se poziva funkcija *PIR_ProcessLEDIndication* koja proverava da li se javila potreba za promenom stanja svetlećih dioda. Procesuiranje pristilih poruka na ZDO mašini cikličnih stanja obavlja se pomoću *PIR_ProcessZDOMsgs*, dok se promena stanja verifikuje u funkciji *PIR_ProcessZDOStateChange*. Prilikom izlaska iz mreže potrebno je obrisati sve sačuvane podatke mreže, kao što su bajnding tabele, tabele sa adresama i slične parametre, za to je zadužena funkcija *PIR_Process_nwk_leave*. Promena atributa se validira pozivom funkcije *PIR_ValidateAttributeChange*. Pokretanje režima u kome uređaj postaje koordinatorski uređaj, dešava se na poziv *PIR_StartCoordinator* funkcije, nakon koje se uređaj restartuje. Stanje identifikacije je vremenski ograničeno i na svaku promenu ovog parametra poziva se funkcija *PIR_ProcessIdentifyTimeChange* koja u zavisnosti od preostalog vremena utiče na dalji tok izvršavanja programa. Za slanje *enroll* zahteva koji je opisan u prethodnim poglavljima poziva se *PIR_SendEnrollReq*, dok se za procesiranje istog koristi *PIR_EnrollProcess*.

Lista funkcija koje su registrovane u *callback* tabeli date su tabelom 4.3.2

Tip	Naziv funkcije	Parametri
<i>void</i>	<i>zclPIR_IAS_IdentifyCB</i>	<i>zclIdentify_t *pCmd</i>
<i>void</i>	<i>zclPIR_ONOFF_IdentifyCB</i>	<i>zclIdentify_t *pCmd</i>
<i>void</i>	<i>zclPIR_IdentifyQueryRspCB</i>	<i>zclIdentifyQueryRsp_t *pRsp</i>
<i>void</i>	<i>PIR_BasicResetCB</i>	<i>void</i>
<i>void</i>	<i>PIR_OCCIdentifyCB</i>	<i>zclIdentify_t *pCmd</i>
<i>void</i>	<i>PIR_OHNConfigCB</i>	<i>zclOhnConfig_t *pCmd</i>
<i>void</i>	<i>PIR_OHNLnsCB</i>	<i>zclOhnLns_t *pCmd</i>

Tabela 4.3.2 – Lista funkcija registrovanih u *callback* tabeli

4.4 Iskorišćenost memorije

U poglavlju 5.2.1. opisani su tipovi memorije kod 8051 arhitekture, a zauzeće u prikazanom programskom rešenju data je tabelom 4.4.1.

Memorija	Alocirano	Jedinica
<i>CODE</i>	229408	bajt
<i>DATA</i>	32	bajt
<i>XDATA</i>	7767	bajt
<i>IDATA</i>	192	bajt
<i>BIT</i>	8	bit
<i>CONST</i>	395	bajt

Tabela 4.4.1 – Prikaz iskorišćenosti memorije po sekcijama

Potrebno je naglasiti da je kao lokacija za smeštanje *string* i konstanti izabrana opcija u kompajleru koja koristi *ROM* koji je mapiran kao *DATA* sekcija. Pored ove opcije moguće je bilo izabrati *RAM* ili *CODE* sekciju memorije. Razliku između ovih memorijskih sekcija predstavlja pristup, odnosno tip komandi koji će se koristiti pri iščitavanju ili upisivanju u memoriju. Pri iščitavanju promenljivih (koje su jednake veličine) iz različitih sekcija memorije možemo uočiti da je potreban različiti broj instrukcija, što utiče na dužinu trajanja ovih operacija.

5. Testiranje i verifikacija

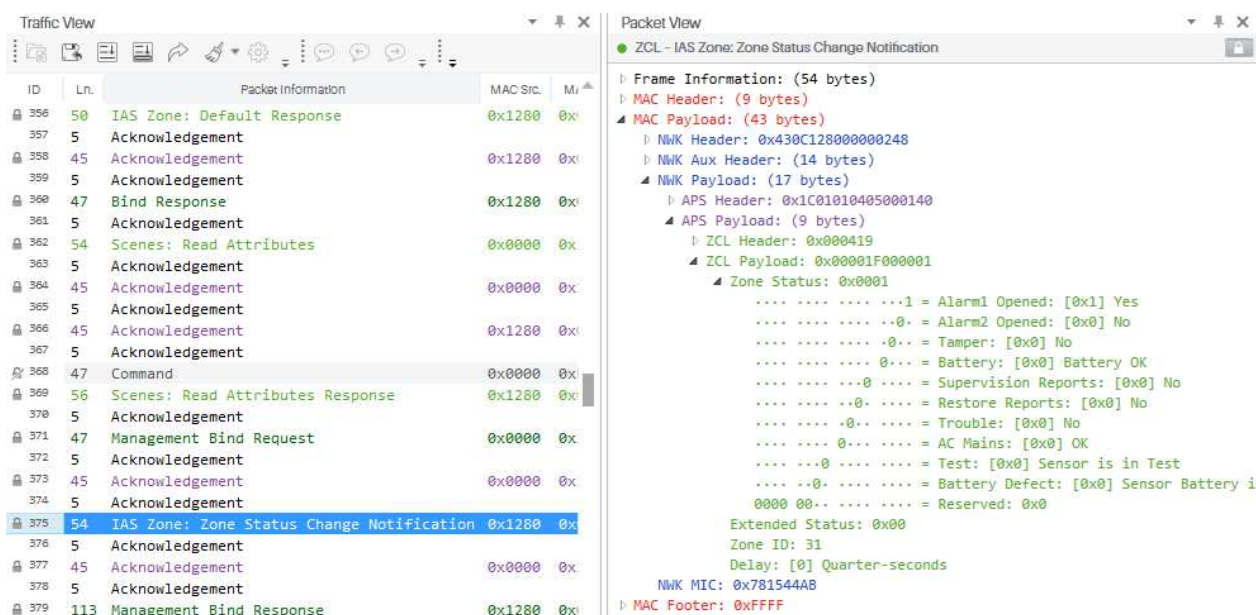
Pri testiranju i verifikaciji neophodno je verifikovati usklađenost sa HA 1.2 specifikacijom, a potom i aplikativno rešenje. HA 1.2 specifikacija definiše skup testova koje uređaj treba da prođe kako bi se potvrdilo da je u skladu sa specifikacijom, koji su definisani u zvaničnom dokumentu od strane ZB alijanse. Međutim kako referentno rešenje koristi već implementiran ZB stek na modulu CC2530 i kako je taj modul već prošao testove vezane za sam ZB protokol, ti testovi nisu morali ponovo da se ispituju. Preostalo je samo testirati aplikativni nivo koji je implementiran. Testovi se sastoje od ispitivanja funkcionalnosti klastera, režima u kojima se može naći uređaj i obaveštenja korisniku o raznim stanjima pomoću svetlosnih dioda na uređaju.

5.1 Alati korišćeni pri testiranju

Jedan od najkorisnijih alata pri verifikaciju ispravne funkcionalnosti uređaja je *Ubiqua Protocol Analyzer*. Ovaj programski alat u kombinaciji sa *USB ZB* uređajem za osluškivanje (koji je baziran na *TI CC2531* mikročipu) omogućuje sagledavanje svih podataka koji se prenose na odabranom kanalu. Uz korišćenje ključa ZigBee komunikacije koji je definisan za celu alijansu moguće je dekodovanje podataka u ZB format, što je mnogo pojednostavilo proces verifikacije. Grafički prikaz ovog alata sa primerom analize poruka dat je slikom 5.1.1.

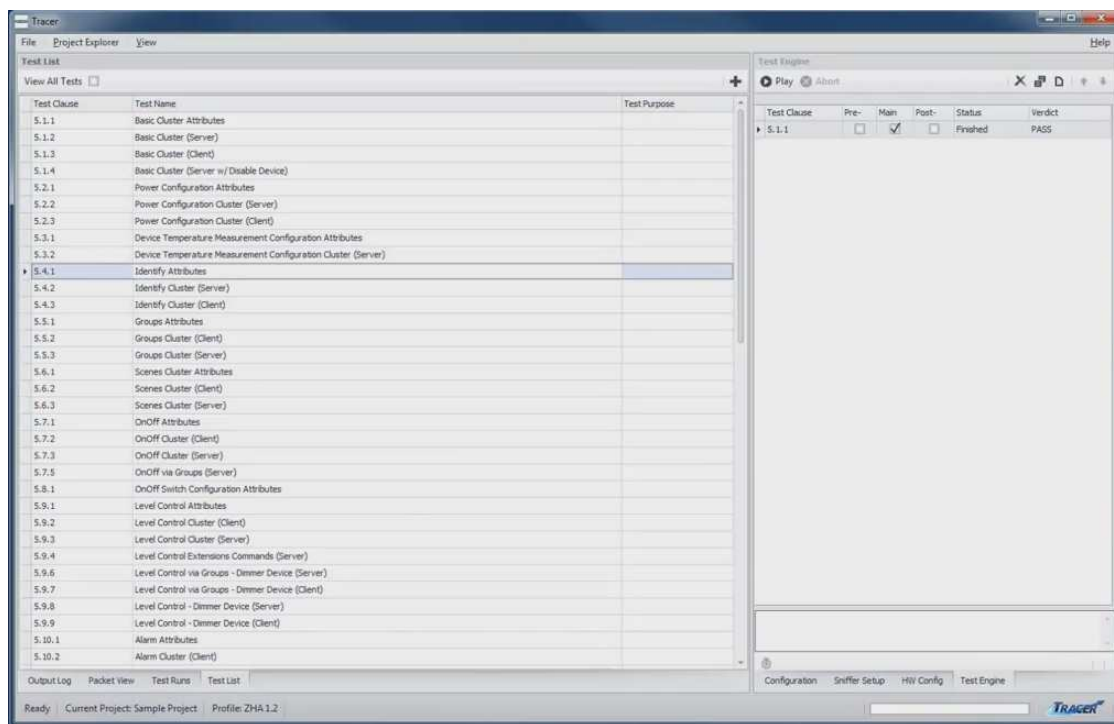
Uz pomoć ovog alata moguće je videti sve pakete podataka na posmatranom kanalu, a zbog velikog broja mreža i uređaja na kanalima potrebno je filtrirati sadržaj kako bi se iz analize izbacile nepotrebne informacije. *Ubiqua* alat omogućava filtriranje po *PAN ID* identifikatoru, što znači da je moguća analiza svake mreže pojedinačno. Zbog mogućnosti dešifrovanja ZB protokola sva polja odgovora i upita su označena, osim u slučajevima gde su korišćeni specijalni upiti koji nisu definisani standardom.

Kako bi se verifikovala sva stanja *PIR* uređaja putem *debugger*-a na samom razvojnom okruženju pratila su se stanja mašine konačnih stanja, kao i procedure koje se u svakom stanju prozivaju. Istovremeno analizirane su poruke koje su osluškivane uz *Ubiqua* alat, na ovaj način jasno se mogao ispratiti tok programa, što je olakšalo otklanjanje grešaka i pronalaženje nedefinisanih stanja.



Slika 5.1.1 – Prikaz Ubiqua Protocol Analyzer alata

Za ispitivanje centralnog kontrolera korišćen je alat priznat od strane *ZigBee* alijanse pod nazivom *Tracer*. U okviru ovog alata dolazi *USB* modul i aplikacija. Kroz spregu aplikacije moguće je izabrati tip uređaja koji se testira i specificirati klastere koji su implementirani na datom uređaju (slika 5.1.2). Nakon odabira podržane funkcionalnosti alat formira skup testova. Nakon pokretanja testova *Tracer* šalje poruke uređaju i beleži da li su njegovi odgovori, vrednosti koje se prenose u tim odgovorima, kao i vremena u skladu sa propisanim standardom. Jasno indikacijom u aplikaciji mogu se uočiti neuspešni testovi i razlozi njihovog neuspeha. Ukoliko je potrebno moguće je podesiti i broj ponavljanja za određene testove. U poglavlju 5.2 data je lista testova koji su izvršeni. Neki od ovih testova nisu bili generisani kroz *Tracer* pa su oni izvršeni ručno pomoću skripti ili korisničke interakcije.



Slika 5.1.2 – Prikaz Tracer alata za verifikaciju

5.2 Lista testova

S obzirom da *PIR* uređaj može da se ponaša kao ruterski uređaj i koordinator, a da pored toga poseduje nekoliko krajnjih tačaka pomoću kojih se može odraditi bajnding procedura neophodno je bilo verifikovati ponašanje u svim stanjima kao i mogućim prelascima iz jednog stanja u drugo. U tabeli 5.2.1 data je lista testova

Važno je napomenuti da je kod nekih testova bilo potrebno testirati uređaj sa različitim krajnjih tački. Tako je na primer pri ulasku u mrežu bilo potrebno verifikovati ulazak u mrežu sa *OCC_ENDPOINT*, pri čemu bi se inicijalizovale sve *occupancy* klaster komande i atributi, kao i ulazak pomoću *IAS_ENDPOINT* kod kojeg bi se inicijalizovali svi parametri vezani za sigurnosni zonski klaster. Takođe kod procedure bajndovanja na uređaj koji se koristi kao meta, korišćen je *ONOFF_ENDPOINT* koji inicijalizuje sve parametre neophodne za korišćenje *ONOFF* klastera.

Urađeni testovi potvrđuju niz kompleksnih funkcionalnosti koje pokazuju da je uređaj kompatibilan sa drugim uređajem koji podržavaju proverenu funkcionalnost. Testovi proveravaju celokupnu funkcionalnost klastera kao što su mogućnost obrađivanje i slanje obaveznih i opcionih komandi sa korektno postavljenim parametrima, čitanje i pisanje opcionih i obaveznih atributa kao i slanje izveštaja o promeni stanja.

Naziv testa	Uspešnost
Uređaj se testira kao koordinator	Prošao
Sigurnosna mogućnost (uređaj se testira kao centar od poverenja)	Prošao
Otkrivanje komandi (server)	Prošao
Otkrivanje atributa (server)	Prošao
Test topologije – ruterski uređaj	Prošao
Test topologije – koordinator	Prošao
Test bajndovanja	Prošao
Veličina bajnd tabele	Prošao
Osnovni klaster atributi	Prošao
Osnovni klaster (server)	Prošao
Identifikacija klaster atributi	Prošao
Identifikacija klaster (server)	Prošao
Identifikacija klaster (klijent)	Prošao
Uključi isključi klaster (klijent)	Prošao
Okupacioni klaster (klijent)	Prošao
Sigurnosne zone klaster (klijent)	Prošao
Otvaranje mreže	Prošao
Skeniranje i formiranje mreže	Prošao
Pronalaženje i bajndovanje: uređaj koji se testira kao meta	Prošao

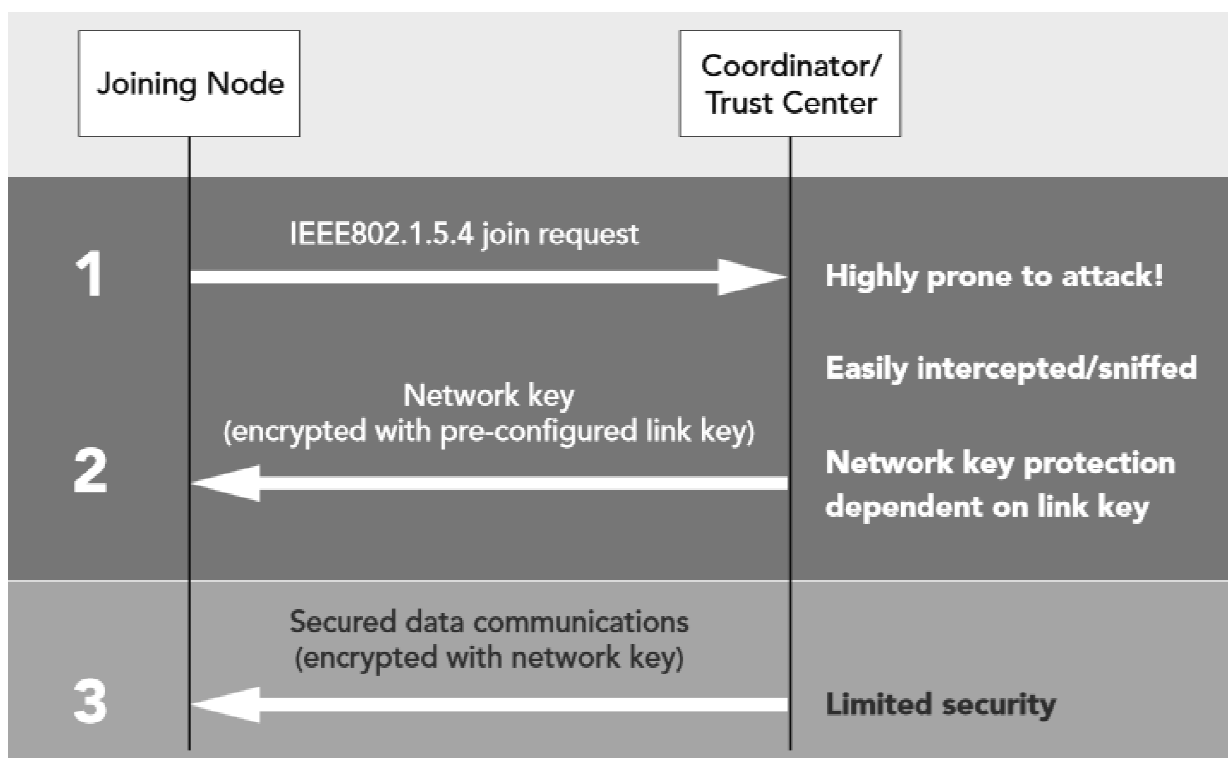
Tabela 5.2.1 – Lista izvršenih testova

6. Zaključak

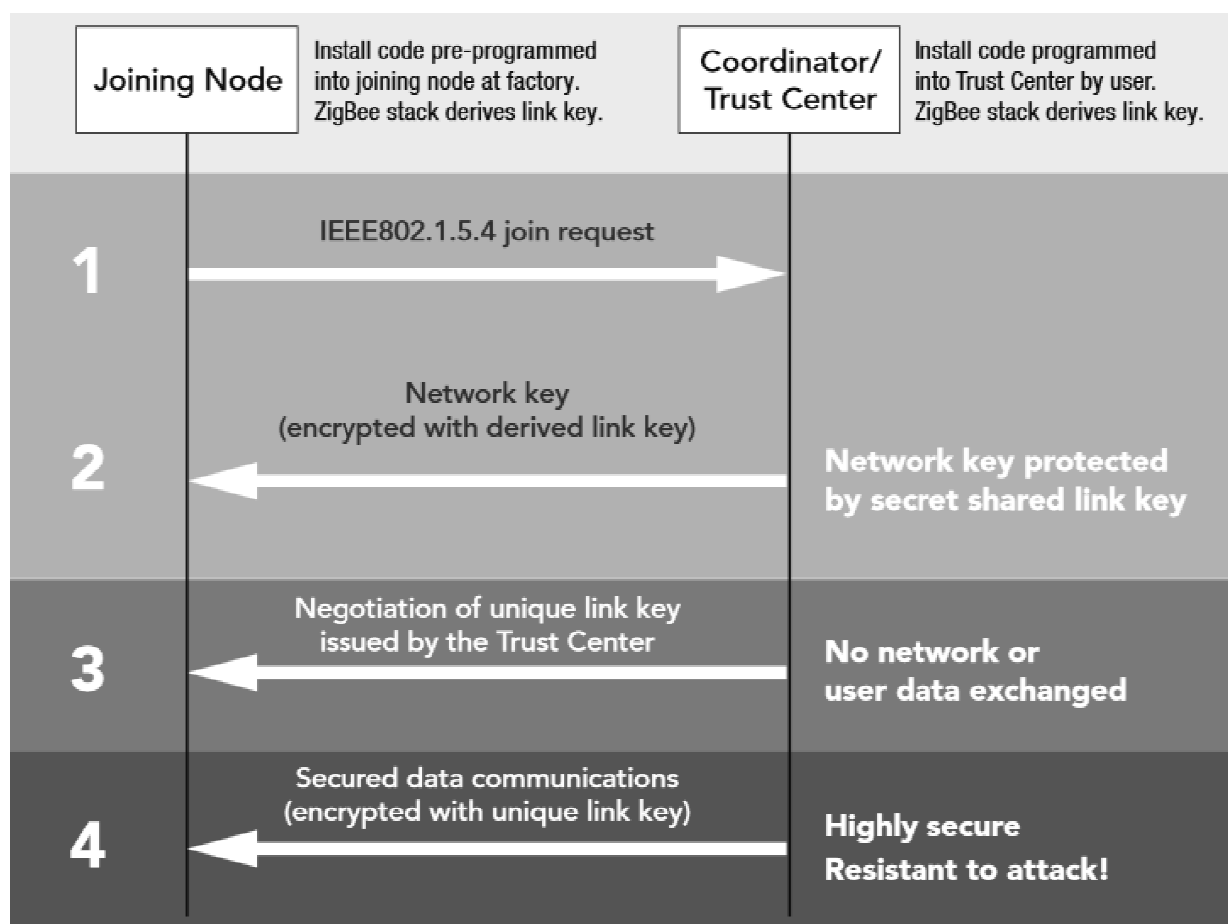
Rezultati testiranja pokazuju da realizovano rešenje ispunjava sve prethodno definisane zadatke, kao i da se referentno rešenje može koristiti jer pored zahteva definisanim u HA 1.2 specifikaciji ispunjeni su i korisnički zahtevi.

Dalja poboljšanja i usavršavanja ovog rešenja mogu biti u pogledu dodavanja novih funkcionalnosti kao što su biranje *LED* osvetljenja u određenim režimima rada, omogućavanje postavke pomenutih *delay* i *threshold* parametara kroz tastere uređaja i podržavanje istih kroz internet portal, kao i prelazak na *ZigBee* 3.0 specifikaciju koja ukida profile i uvodi drugačiju organizaciju funkcionalnosti. Ujedno prelazak na kontroler *TI CC2538* obezbedio bi više memorije, što bi omogućilo da se jedan uređaj može bajndovati sa mnogo više uređaja u odnosu na *TI CC2530*, a prednosti bi bile vidljive i u pogledu potrošnje i kvaliteta signala.

Sigurnost *IoT* uređaja kao i *ZigBee* mreže koja čini najzastupljeniji protokol u ovoj grani industrije je često razlog zabrinutosti i tema mnogih debata. Baš iz ovog razloga *ZigBee* alijansa je uložila veliki trud da se isprave svi sigurnosti propusti koji su napravljeni u verzijama pre. Veliki sigurnosni napreci su napravljeni kako bi se konekcija enkriptovala i zaštitili osetljivi podaci. Na prethodnim verzijama (slika 5.2.1) mrežni ključ se razmenjivao između koordinatora i krajnjeg uređaja preko nezaštićene veze, pa je zbog toga bio vidljiv softverskim alatima koji u trenutku razmene snimaju mrežu. Na novoj verziji *ZigBee* 3.0 ovaj ključ se generiše pomoću *install* koda (slika 5.2.2) koji se nasnimi u sam uređaj u toku fabričke izrade i *Matyas-Mayer-Oseas (MMO) hash* funkcije. Za vreme komisije instalacioni kod se ukuca u centar od poverenja i *ZigBee* stek kreira identičan kod za povezivanje od instalacionog koda.



Slika 5.2.1 – Prikaz razmene mrežnog ključa na starijim verzijama ZB



Slika 5.2.2 – Prikaz razmene mrežnog ključa na ZB 3.0

Najveća mana ovakvih sistema za detekciju pokreta predstavlja lažna detekcija, odnosno detektovanje pokreta koji će postaviti sistem u stanje alarma kada to nije predviđeno. Jedna od najčešćih pogrešnih detekcija je detekcija kućnih ljubimaca ili pomeranja listova biljaka usled strujanja vazduha koje se nalaze u dometu senzora. Trenutno postoje sistemi koji poseduju posebne algoritme kojima se odstranjuje mogućnost detektovanja usled kretanja kućnih ljubimaca pa bi mogući nastavak razvoja uređaja mogao ići ka razvoju i implementaciji takvih algoritama.

7. Literatura

- [1] Vanja Eličić, Nada Eličić: *Arhitektura protokola i topologija senzorskih ZigBee mreža konferencija Infoteh, Jahorina Mart 2010*
- [2] Drew Gislason: *ZigBee Wireless Networking*, Elsevier Inc. 2008
- [3] Texas Instruments, <http://www.ti.com/> (pristupano 1.8.2017)
- [4] ZigBee, <http://www.zigbee.org/> (pristupano 1.8.2017)
- [5] *Z-Stack Developers Guide*, Texas Instruments, Inc.San Diego, California USA Drew Gislason: *ZigBee Wireless Networking*, Elsevier Inc. 2008
- [6] *Z-Stack Home Developer's Guide*, Texas Instruments, Inc.San Diego, California USA
- [7] *Z-Stack Home Sample Application User's Guide*, Texas Instruments, Inc.San Diego, California USA
- [8] *ZigBee wireless standard*, <https://www.automatika.rs/baza-znanja/obrada-signala/zigbee-wireless-standard.html> (pristupano 1.8.2017)
- [9] *Mesh mreže*, <http://www.cis.hr/dokumenti/mesh-mreze.html> (pristupano 1.8.2017)
- [10] *Oblo Living*, <http://www.obloliving.com/technology/> (pristupano 1.8.2017)
- [11] *ZigBee Reporting*, http://www.atmel.com/images/Atmel-42334-ZigBee-Attribute-Reporting_ApplicationNote_AT08550.pdf (pristupano 1.8.2017)
- [12] *Z-Stack Sample Applications*, <http://weber.itn.liu.se/~qinye29/tne090/Z-Stack%20Sample%20Applications.pdf> (pristupano 1.8.2017)
- [13] *ZigBee ZCL Reporting Configuration*, <https://www.slideshare.net/simenli/implementation-of-the-zigbee-zcl-reporting-configuration-freatures> (pristupano 1.8.2017)

-
- [14] *Occupancy sensor*, http://processors.wiki.ti.com/index.php/Occupancy_Sensor
 - [15] *RF Layout Reference Design for 2.4 GHz (CC2530) Schematic*,
<https://www.ti.com/seclit/df/tidr231/tidr231.pdf> (pristupano 1.8.2017)
 - [16] *CC2530 User's Guide*, <http://www.ti.com/lit/ug/swru191f/swru191f.pdf> (pristupano 1.8.2017)
 - [17] *CC2530 datasheet* <http://www.ti.com/lit/ds/symlink/cc2530.pdf> (pristupano 1.8.2017)
 - [18] *SmartRF05EB User's Guide (Rev. A)*, <http://www.ti.com/lit/ug/swru210a/swru210a.pdf> (pristupano 1.8.2017)
 - [19] *Realizacija senzora vrata u sistemu pametnih kuća zasnovanoj na ZigBee protokolu*,
Lazar Tešanović
 - [20] *Jedno rešenje sistema za testiranje modula za detekciju pokreta zasnovanog na PIR senzoru*, Miloš Janković
 - [21] *Rešenje ZigBee programske podrške na centralnom kontroleru u skladu sa HA 1.2 specifikacijom*, Vladimir Moravčević
 - [22] *Poboljšanje rutiranja u ZigBee mesh mrežama*, Aleksandar Majdak, Milorad Vukadinović, Lazar Tešanović, Miloš Subotić, Konferencija ETRAN