SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 3

BEŽIČNI OSJETILNI ČVOR NOVE GENERACIJE

Dinko Oletić

Zagreb, lipanj 2010.

Zahvaljujem mentoru prof.dr.sc. Vedranu Bilasu na vodstvu prilikom izrade ovog rada. Također, hvala djelatnicima Siemensa d.d. Tomislavu Ražovu i Dariju Hanžekoviću te djelatnicima Zavoda za elektroničke sustave i obradu informacija Marijanu Kuriju i Tihomiru Marjanoviću na stručnoj pomoći i savjetima.

Hvala obitelji i prijateljima na podršci i strpljenju prilikom izrade ovog rada.

Sadržaj

1 Uvod

Bežične osjetilne mreže već su nekoliko godina zanimljivo interdisciplinarno područje koje objedinjuje istraživanja na područjima napajanja iz ambijenta, senzora i instrumentacije, elektroničkih sklopova za rad u ekstremnim uvjetima, razvoja ugradbenih umreženih računalnih sustava, bežičnih mreža i mrežnih protokola, optimiranja potrošnje, distribuirane obrade informacija itd. Općeniti uvod u problematiku bežičnih osjetilnih mreža dan je u (Holger, i dr., 2005).

Ovaj rad fokusira se na razvoj sklopovlja i programske podrške za osjetilne čvorove senzorske mreže. Najprije je opisan koncept bežičnog osjetilnog čvora, tipični podsustavi i zahtjevi bežičnog čvora te dani primjeri bežičnih osjetilnih čvorova. Potom je opisano sklopovlje novorazvijenog čvora FER Čvorak v2.0b po podsustavima i opisane su specifičnosti tiskane pločice osjetilnog čvora. Nakon toga je opisana programska podrška razvijena za čvor FER Čvorak v2.0b: opisano je korišteno programsko sučelje BitCloud, programska podrška za senzore te mrežne aplikacije pojedinih čvorova. Zatim je dan koncept programske podrške za programiranje čvorova na daljinu. Na kraju su u zaključku dani osvrt na nedovršene dijelove sustava i planovi za daljnji razvoj.

Ovaj rad nastavlja se na razvojno-istraživačke aktivnosti u sklopu projekta Maslinet pa je u nastavku dan kratki opis projekta ne bi li se dao kontekst ovom radu..

1.1 Projekt Maslinet

Bežične mreže osjetila (engl. *wireless sensor networks*) već su nekoliko godina predmet istraživanja u okviru istraživačke grupe AIG (engl. *Advanced Instrumentation Group*) na Zavodu za elektroničke sustave i obradu informacija Fakulteta elektrotehnike i računarstva. U sklopu tih aktivnosti 2007. godine pokrenut je razvojno istraživački projekt Maslinet. Cilj projekta je razviti prototip multimodalne bežične mreže osjetila za primjenu u maslinarstvu. Namjena mreže je praćenje dvaju modaliteta:

1. vizualno praćenje broja maslinovih muha (štetnik koji ugrožava nasade masline)
2. praćenje mikroklimatskih parametara - temperature i mokrine tla, temperature, tlaka i relativne vlažnosti zraka te razine osvjetljenja

Važnost primjene ove mreže u poljoprivredi leži u uštedi koja se može postići prevencijom štete nastale maslinovom muhom. Korisnost ovakve mreže osjetila posebice dolazi do

izražaja u maslinicima na zabačenim lokacijama teško dostupnim poljoprivrednim savjetodavnim službama. Ovakvim sustavom postavljenom na otocima omogućio bi se jeftini centralizirani nadzor nasada iz udaljene lokacije bez potrebe za obilascima pojedinačnih lokacija na terenu. Detalji projekta Maslinet dostupni su na mrežnim stranicama projekta (Maslinet, 2009.).

Kontekst i specifikacije projekta postavljaju određene zahtjeve na bežičnu osjetilnu mrežu razvijenu za potrebe projekta Maslinet. Temeljni zahtjevi su:

- kontinuirani višegodišnji rad bežične mreže osjetila u uvjetima bez stalnog izvora napajanja (bez pristupa javnoj elektroenergetskoj infrastrukturi)
- smještaj mreže u prirodi podrazumijeva rad u ekstremnim uvjetima temperature i vlažnosti
- otpornost na pogreške u radu u smislu sposobnosti detekcije nepredviđenih situacija i oporavka bez potrebe za ljudskom interakcijom
- upravljanje i održavanje mreže s udaljene lokacije putem mogućnosti bežične nadogradnje mreže (engl. *over the air download*)

Aktualna bežična osjetilna mreža razvijena u sklopu projekta Maslinet namijenjena postavljanju na teren zvjezdaste je strukture. Sastoji se od triju osjetilnih čvorova i jednog koordinatorskog čvora. Jedan krajnji čvor mjeri mikroklimatske parametre tla, drugi čvor mjeri mikroklimatske parametre zraka i osvjetljenje, dok treći na sebi ima lovku za maslinove muhe i FPGA kamerom razvijenom specifično za potrebe projekta Maslinet fotografira sadržaj lovke (Jeličić, 2009). Krajnji senzorski čvorovi sa središnjim koordinatorskim čvorom komuniciraju ZigBee protokolom. Krajnji čvorovi većinu vremena provode u spavanju. Bude se periodički u fiksnim vremenskim intervalima i koordinatoru šalju servisni izvještaj – obavijest o naponu napajanja, snazi i kvaliteti radio-signala (RSSI i LQI), temperaturi u kućištu, trenutnoj verziji programske podrške. Mjerne podatke (uključujući sliku s kamere) Krajnji čvorovi mjerne podatke šalju na zahtjev koordinatora ZigBee mreže. Krajnji čvorovi napajaju se iz elektrokemijskih izvora.

Koordinatorski ZigBee čvor je centralni čvor mreže. Koordinator zahtjeve za podacima dobiva ili od krajnjeg korisnika s udaljenog mrežnog poslužitelja preko Interneta na koji je priključen GPRS vezom ili ih generira samostalno u periodičkim vremenskim intervalima. U trenutnoj verziji mreže Maslinet koordinator je centralno mjesto donošenja odluka o

vremenskom rasporedu prikupljanja podataka. Zbog specifikacije ZigBee protokola koordinator mora biti uključen cijelo vrijeme (ZigBee Alliance, 2008.) što bitno utječe na izvedbu napajanja koordinatora. ZigBee koordinator i GPRS poslužitelj su ugrađeni u isto kućište i napajani iz zajedničkog solarnog izvora napajanja s MPPT sklopovljem. Koordinatorski čvor prikupljene podatke prosljeđuje ugradbenom GPRS poslužitelju koji omogućuje dvosmjernu povezivost ZigBee mreže sa udaljenim internetskim poslužiteljem koristeći GPRS protokol i široko dostupnu infrastrukturu GSM baznih stanica. Krajnji korisnik mreže komunicira s bežičnom mrežom osjetila preko interneta pristupajući udaljenom internetskom poslužitelju. Struktura mreže Maslinet opisana u ovom poglavlju prikazana je slikom 1.

Slika 1: Prikaz topologije bežične osjetilne mreže Maslinet

2 Koncept bežičnog osjetilnog čvora

Bežični osjetilni čvorovi su temeljne jedinice bežičnih senzorskih mreža. To su ugradbeni računalni sustavi namijenjeni prikupljanju informacija iz ambijenta, njihovoj obradi i bežičnoj komunikaciji s drugim čvorovima mreže.

2.1 Tipični podsustavi bežičnih osjetilnih čvorova

Pregled nad osnovnim podsustavima bežičnih osjetilnih čvorova već je dan u (Bačan, 2007.). Bežični osjetilni čvor tipično se sastoji od središnje procesorske jedinice (mikrokontrolera), radiokomunikacijskog sklopa, izvora napajanja, vanjske memorije, i priključenih senzora. U tipičnom scenariju većinu vremena provodi u neaktivnom stanju. Povremeno se budi, mjeri, šalje izmjerene podatke i vraća se u stanje manje potrošnje (stanje spavanja).

Mikrokontroler je središnja komponenta bežičnog osjetilnog čvora. Za primjenu u bežičnim senzorskim mrežama koriste se mikrokontroleri različitih procesorskih snaga. Za primjene koje ne zahtijevaju mnogo procesorske snage na tržištu se najčešće nude 8-bitne porodice 8051 i Atmel AVR s reda veličine 4 KB podatkovne memorije te 128 KB programske memorije. Česti su i moćniji 32-bitni mikrokontroleri s više memorije temeljeni na ARM jezgrama. Zajedničko svim korištenim mikrokontrolerima je mala potrošnja u neaktivnom stanju, reda veličine nekoliko stotina nanoampera.

Radiokomunikacijski sklop mora imati mogućnost isključivanja odašiljača. U svrhu smanjenja potrošnje. Radiokomunikacijski sklopovi tipično rade u ISM pojasu (2,4 GHz), a postoje i verzije koje rade na 900 ili 433 MHz. Tipične izlazne snage odašiljača se kreću od 1 mW do 100 mW (najveća dozvoljena snaga odašiljača koji radi u ISM frekvencijskom području). Tipična potrošnja radiokomunikacijskih sklopova je tipično 10 do nekoliko desetaka mA. Korišteni digitalni radio-uređaji sklopovski implementiraju neki od korištenih komunikacijkih protokola i time rasterećuju središnji procesor i olakšavaju programiranje Najpoznatiji proizvođač radio-sklopova za bežične osjetilne mreže je tvrtka Chipcon u vlasništvu Texas Instrumentsa.

Najčešće korišteni komunikacijski protokoli u bežičnim osjetilnim mrežama su protokoli zasnovani na standardu IEEE 802.15.4, uključujući ZigBee protokol. Tijekom posljednjih nekoliko godina oko ovog protokola je razvijen najveći udio komponenata za primjenu u bežičnim osjetilnim mrežama. Problem mreža sastavljenih od čvorova povezanih ZigBee

protokolom predstavlja povezivost bežične mreže osjetila prema okolini. Standard 802.15.4. nije kompatibilan s internet protokolom pa je obično na koordinatorski čvor potrebno povezati neku vrstu *gateway* uređaja za uspostavu veze s udaljenim serverom. Pritom se za uspostavu veze na Internet obično koristi infrastruktura i protokoli mobilne telefonije. Kao rješenje problema IP-povezivosti, na tržištu su se pojavili bežični čvorovi sa sklopovljem koje realizira 6LoWPAN protokol čime se ostvaruje IPv6 povezivost izravno do svakog krajnjeg bežičnog osjetilnog čvora.

Razina integracije između mikrokontrolera i radio-uređaja varira. Moguće je nabaviti odvojene komponente i kombinirati ih po želji što daje veliku fleksibilnost u odabiru mikrokontrolera primjerenog primjeni osjetilnog čvora. Mikrokontroler i radio-sklop najčešće komuniciraju SPI sučeljem. Nedostatak izgradnje osjetilnog čvora od odvojenih komponenti je potreba za projektiranjem analognog radio-izlaza na koji se spaja antena, što može biti nezahvalan posao. Zato se na tržištu pojavio veliki broj gotovih ZigBee modula koji u SiP tehnologiji u zatvorenom kućištu kombiniraju mikrokontroler i radiokomunikacijski uređaj te imaju izveden antenski izlaz ili integriranu keramičku antenu na samom modulu (kod modula nižih snaga). Najpoznatiji su uređaji tvrtki Texas Instruments, Meshnetics (Atmel), Ember, Jennic itd.

Kako tipični čvorovi bežičnih senzorskih mreža, osim integriranog procesorsko-mrežnog modula sadrže još tipično sklopovlje napajanja, potrebne pasivne komponente, vanjske digitalne sklopove poput RTC sklopa, senzore i potrebne konektore, veličina tiskanih pločica im obično iznosi nekoliko desetaka cm^2. Još uvijek nedostižni ideal bežičnog osjetilnog čvora je osjetilni čvor izveden kao samodostatni monolitni SoC čvor površine nekoliko mm^2 koji u sebi integrira sve prije navedene dodatne sklopove i još napajanje. Vrhunac u smjeru minijaturizacije predstavljaju SoC moduli koji u istom čipu integriraju mikrokontroler i radio-sklop, no zahtijevaju veću količinu periferije na tiskanoj pločici (analogni radio-izlaz) kao i SiP moduli čime ne dolazi do značajnog smanjenja dimenzija. Time koncept tzv. *smart-dust* bežičnih osjetilnih čvorova sub-milimetarskih dimenzija ostaje još uvijek neostvaren.

2.2 Napajanje bežičnih osjetilnih čvorova

Čvorovi bežičnih osjetilnih mreža zamišljeni su da rade napajani elektrokemijskim izvorima ili iz obnovljivih izvora iz ambijenta. Od obnovljivih izvora iz ambijenta najčešće se koriste Sunce i vjetar. Gdje je to moguće, mogu se iskoristiti vibracije kao izvor energije, npr. kod bežičnih mjernih čvorova montiranih na osovine rotirajućih strojeva .

Posebno je zanimljivo napajanje iz obnovljivih izvora u kombinaciji s odgovarajućim mjernim sklopovljem (tipično mjerenje napona akumulatora). To omogućava izvedbu energetski svjesnog bežičnog osjetilnog čvora sposobnog donositi odluke (planiranje rasporeda izvršavanja energijski zahtjevnih zadaća, adaptivna izmjena učestalosti buđenja, čitanja senzora itd.) u ovisnosti o količini dostupne energije. Pristupi izvedbi energetski svjesnog upravitelja zadacima dani su u (Brunelli, 2007).

Solarna napajanja korištena za napajanje osjetilnih čvorova tipično se sastoje od solarnog panela, sklopa za održavanje panela u točki maksimalne snage (engl. MPPT, *maximum power point tracker*) i jedinice za pohranu energije. Za privremenu pohranu koriste se superkondenzatori, dok se za pohranu većih količina energije koriste sekundarni elektrokemijski članci. Kod napajanja krajnjih čvorova male snage koji većinu vremena spavaju je potrebno na temelju iznosa potrošnje, topologije napajanja, odabranog panela i vrste jedinice za pohranu energije potrebno donijeti odluku o svrsishodnosti korištenja MPPT sklopovlja. Topologije napajanja i pristupi projektiranju solarnog napajanja opisani su u (Brunelli, i dr., 2008).

2.3 Programiranje na daljinu

Ideja o primjeni bežičnih mreža na teško dostupnim lokacijama zahtjeva mogućnost bežične nadogradnje programske podrške s udaljene lokacije. Problematika programiranja na daljinu sustavno je obrađena u (Brown, 2006.). Općenito gledano, prilikom prijenosa na daljinu potrebno je svim čvorovima mreže isporučiti promijenjenu verziju programskog koda. U bežičnim mrežama s velikim brojem čvorova i složenom topologijom (npr. *multi-hop* mreža s topologija stabla) potrebno je odgovarajućim protokolima osigurati da obnovljena verzija programskog koda uspješno i učinkovito stigne do svih čvorova.

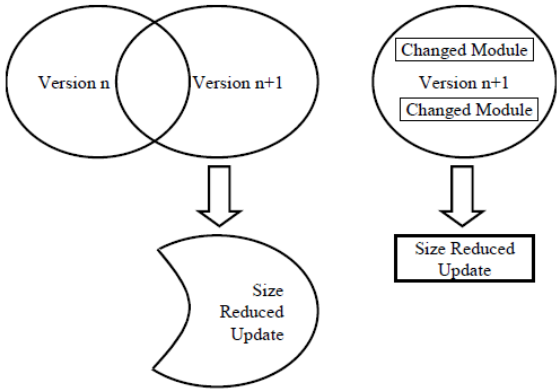
Programiranje na daljinu obuhvaća tri aspekta:

1. Kodiranje na strani izvora programskog koda u svrhu smanjenja mrežnog prometa

2. Prijenos programskog koda kroz mrežu
3. Izvršavanje nadogradnje programske podrške na čvoru mreže

2.3.1 Kodiranje na strani izvora programskog koda

Programiranje na daljinu podrazumijeva prijenos velikih količina programskog koda. Velike količine programskog koda uzrokuju veliki mrežni promet i time troše mnogo energije. Problem se rješava kodiranjem na strani polazišta odakle se injektira novi programski kod u mrežu. Postoje dva temeljna postupka prikazana slikom 2.



Slika 2: Metode smanjenja veličine programskog koda (Brown, 2006.)

Metoda prikazana na slici 2 lijevo je inkrementalna metoda nadogradnje. Uspoređuje trenutnu i novu verziju programskog koda prije prenošenja. Dijelovi koji su zajednički se ne prenose, već se prenosi samo razlika programskih slika (Reijers, i dr., 2003.). Na strani svakog čvora moja postojati skripta za obradu koja smješta primljene dijelove slike na prave lokacije u programskoj memoriji. Uz ovu metodu pogodno je prenositi slike u tekstualnom SREC (S-19), a ne binarnom formatu jer SREC format sadrži metapodatke korisne za obradu na odredišnom čvoru. Takav model u kojem se blok programskog koda prenosi samo ako se CHKSUM vrijednosti trenutno korištenog i novog bloka koda razlikuju opisuje i (Jeong, 2004.).

Metoda sa slike 2 desno je prenošenje i nadograđivanje programskog koda po dijelovima-modulima, a ne u cjelini. Postoje dva slučaja kad je to izvedivo. U prvom slučaju krajnji čvor mora biti ili platforma visoke razine s nekom vrstom operacijskog sustava gdje je programski kod razdijeljen na samostalne aplikacije-procese koji se mogu odvojeno nadograđivati. U tom slučaju najčešće je moguće nadograđivati aplikacije bez zastoja rada čvora, no nije moguće nadograđivati module koji se tiču operacijskog sustava.

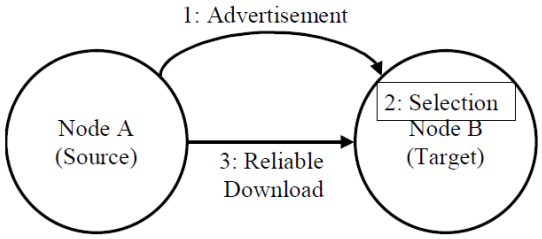
U drugom slučaju, modularna nadogradnja se može primijeniti na platformu niske razine (i niske složenosti), ako moduli koji se prenose imaju nepromjenjive adrese u programskoj memoriji i ako je za svaki modul predviđen dovoljno veliki memorijski prostor da nova verzija modula zbog prevelike veličine ne izađe iz njega (Koshy, i dr., 2005.).

2.3.2 Prijenos programskog koda kroz mrežu

Temeljni zadatak protokola propagacije paketa kroz mrežu je da programski kod uspješno stigne do svih čvorova bežične mreže (pod uvjetom da svi čvorovi moraju primiti isti programski kod).

Programiranje na daljinu podrazumijeva prijenos velikih količina programskog koda. Stoga se programska slika (engl. *image*) dijeli na pakete i prenosi paketno. Za programiranje na daljinu općenito, potrebno je osigurati protokol sigurnog prijenosa jer inače nije moguće. Dvije su varijante najčešće korištene – prijenos uz potvrdu prijenosa te prijenos bez potvrde kod kojeg primatelj nakon prijenosa cijele slike provjerava je li primio sve pakete i šalje zahtjev za retransmisijom nedostajućih.

Programski kod se načelno „difuzijom“ širi kroz mrežu počevši od mjesta injekcije prema čvorovima gdje još nije provedena nadogradnja. Kad neki čvor obavi nadogradnju, može postati izvor novog programskog koda za okolne čvorove. Koncept je prikazan slikom 3.



Slika 3: Temeljni protokol u iterativnom procesu propagacije programskog koda kroz mrežu (Brown, 2006.)

Čvor objavljuje da ima na raspolaganju novi programski kod. Zainteresirani (nenadograđeni) čvorovi se javljaju za nadogradnju. Nakon toga počinje proces sigurnog prijenosa. Moguć je broadcast svim čvorovima u okolini ili pojedinačni prijenos (tzv. *unicast* prijenos), ovisno o odabranom protokolu.

Jedan od važnijih problema protokola propagacije paketa kroz mrežu je energetski aspekt. Naime, slanje programskog koda je u pravilu dugotrajna, energijski zahtjevna mrežna operacija. Zbog toga valja nadogradnju započinjati samo onda kad se procijeni da

u sustavu ima dovoljno energije. Svaki put kad neki čvor mreže mora emitirati programski kod, smanjuje mu se energijska zaliha za iznos određen točnim protokolom prijenosa koji može i ne mora biti predvidiv jer je potrebno ostvariti prijenos bez gubitaka (broj retransmisija nije unaprijed poznat ali se ugrubo može procijeniti na temelju razine snage signala - RSSI), a smanjenjem kvalitete veze među čvorovima broj retransmisija se povećava. Osim smanjenja broja retransmisija, produljenju životnog vijeka pogoduje i što manje upravljačkih poruka u protokolu jer smanjuje ukupni broj transmisija.

Također, važno je imati i sustav upravljanja verzijama koji određuje koje verzije koda su kompatibilne i dopušta započinjanje prijenosa samo kompatibilnih verzija te tako štedi energiju.

Zanimljivo, no ne i primjenjivo rješenje je korištenje posebne servisne mreže koja postoji uz „glavnu“ mrežu čvorova i služi samo za nadogradnju programske podrške. Ovakvo rješenje više se uklapa u scenarije gdje se udaljeno programiranje koristi više radi komocije u slučaju mreže s velikim brojem čvorova (izbjegavanje spajanja programatora/računala na svaki bežični čvor) ili čvorovima na nedostupnim mjestima (npr. proizvodni pogoni), nego u bežičnim senzorskim mrežama na udaljenim lokacijama napajanim iz ambijenta.

2.3.3 Izvršavanje nadogradnje programske podrške na čvoru mreže

Na krajnjem čvoru programski kod se tijekom prijenosa kroz mrežu najprije sprema u odvojeni dio EEPROM/FLASH programske memorije. Nakon završetka prijenosa programskog koda može započeti proces nadogradnje. Za taj proces potreban je odvojeni program u programskoj memoriji pod nazivom *bootloader*. Uloga *bootloader* je odrediti koji programski kod (slika) iz programske memorije će se učitati za izvršavanje. Kod slučaja gdje se kod za nadogradnju sprema u vanjsku memoriju, *bootloader* čita novu sliku iz vanjske memorije i upisuje ju u FLASH memoriju mikrokontrolera. *Bootloader* se obično pokreće usmjeravanjem programskog toka na fiksnu lokaciju gdje je spremljen njegov programski kod (obično postoji i maksimalna duljina koda *bootloadera*). Nakon toga *bootloader* prenosi odabranu sliku (npr. iz vanjske memorije) stranicu po stranicu na mjesto stare slike u mikrokontroleru. Na kraju, *bootloader* resetira mikrokontroler i mikrokontroler će početi izvršavati novi program. *Bootloader* mora biti napisan tako da

osigura dugotrajnost FLASH memorije. Naime, svaka FLASH memorija ima ograničeni broj ciklusa čitanja/pisanja koji je reda veličine 100000 puta.

Posebno je bitno uskladiti trenutak početka pokretanja *bootloadera* da se osigura izbjegne konflikt između različitih verzija programskog koda. Proces može biti iniciran signalom iz mreža ili proizvoljan (npr. čim se dovrši prenošenje slike) ako istodobnost nije važna.

Također, važno je i pitanje koliko verzija programskog koda čuvati. Uobičajena praksa je čuvanje kopije trenutnog koda u vanjskoj memoriji i zapisivanje nove verzije koda u vanjsku memoriju na odvojenu lokaciju (ne prepisivanje trenutne verzije kopije u vanjskoj memoriji). Time se u slučaju pogreške u novom kodu moguće vratiti na staru (trenutnu verziju). Također, Dakle, kapacitet vanjske programske memorije trebao bi biti barem dvostruk u odnosu na najveću moguću duljinu programskog koda. Da se izbjegnu nepotrebna čitanja i pisanja, moguće je praćenjem posebno zadane varijable u EEPROM-u alterirati poziciju (sektor) početka starog/zapisivanja novog programskog koda.

Nakon nadogradnje programskog koda i pokretanja može se dogoditi da nova aplikacija radi lošije od stare verzije. U tom slučaju treba osigurati mehanizme poput npr. reseta nakon preljeva *watch-dog* brojila. Ako se mikrokontroler u razmjerno kratkom periodu s novom verzijom koda više puta resetira uslijed zaglavljenja, to je znak da se treba vratiti na prethodnu (trenutnu) verziju koda.

Na naprednijim čvorovima s operativnim sustavom OS umjesto *bootloader* programa koristi se upravitelj aplikacijama koji služi selektivnim nadogradnjama na razini pojedinačnih aplikacija.

2.4 Pregled postojećih bežičnih osjetilnih čvorova

2.4.1 FER Čvorak

Prva verzija mreže Maslinet bila je zasnovana na čvoru FER Čvorak (Bačan, 2007.). Čvor je bio koncipiran modularno. Sastojao se od jedne sabirničke pločice sa PCI-X4 konektorima. Preko tih konektora priključivali su se moduli – komunikacijski modul s mikrokontrolerom i radiokomunikacijskim sklopom, senzorski modul i *debug*-modul, modul s RTC sklopom.

Korišten je mikrokontroler PIC18F6722 i radio-sklop MRF24J40 tvrtke Microchip. Odabrano je Microchipovo rješenje zbog programskog stoga otvorenog koda. Čvor je

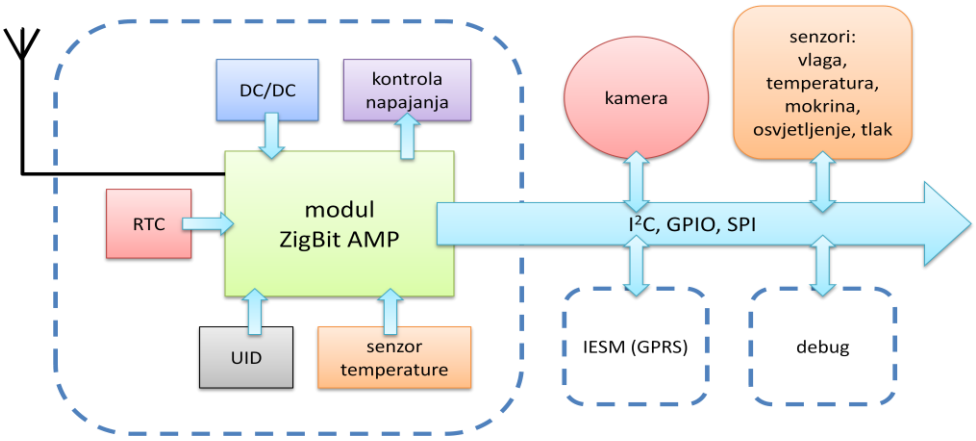
prikazan na slici 3, lijevo montiran u plastično kućište za testiranje na terenu. Nedostaci ovog čvora su mala izlazna snaga (1 mW) i problemi s analognim sklopovljem radio- izlaza.

2.4.2 FER Čvorak v2.0

U sklopu projekta Maslinet dosad su razvijene dvije inačice bežičnih osjetilnih čvorova. Trenutno je u upotrebi druga inačica osjetilnog čvora nazvana FER Čvorak v2.0. Ovaj čvor razvijen je tijekom proljeća 2009. godine za potrebe nastavka rada na projektu Maslinet i postavljanje nove verzije mreže u maslinik tijekom ljeta 2010. godine. Kako je novorazvijeni čvor opisan u okviru ovog rada (FER Čvorak v2.0b) temeljen na postojećem čvoru FER Čvorak v2.0, u nastavku je ukratko ukratko opisana arhitektura postojećeg čvora FER Čvorak v2.0.

Kao temeljna komponenta oko koje je izgrađen bežični osjetilni čvor FER Čvorak v2.0 odabran je bežični komunikacijski modul ZigBit Amp proizvođača Atmel, predviđen za rad na 2,4 GHz u IEEE 802.15.4/ZigBee mrežama. Radi se o modulu izrađenom kao SiP (engl. S*ystem in Package*) čije su glavne komponente mikrokontroler ATmega1281 i komunikacijski sklop komunikacijski sklop AT86RF230. Modul je detaljno opisan u poglavlju 3.1.4. Glavni razlozi za izbor ovog modula u odnosu na konkurentska rješenja bile su velika izlazna snaga odašiljača, mala potrošnja i veliki broj ulazno-izlaznih linija. Potrošnja u neaktivnom stanju iznosi do 6 µA. U stanju bežičnog slanja podataka potrošnja je deklarirana na 50 mA, a u primanju na 23 mA.

FER2 Čvorak v2.0 je projektiran kao bežični čvor sastavljen od jedne univerzalne tiskane pločice koja može preuzeti ulogu bilo kojeg čvora lokalne ZigBee mreže. Blok shema čvora prikazana je slikom 4. ZigBit modul s okolinom komunicira koristeći nekoliko sučelja. Jednožičnim sučeljem (engl. *1-wire interface*) komunicira s UID sklopom u kojem mu je zapisana MAC adresa. I^2C sučelje koristi za komunikaciju sa satom (engl. *real time clock*) i dijagnostičkim senzorom temperature (komponenta LM73) ugrađenim na pločicu te vanjskim senzorom osvjetljenja (ISL29013). Senzor vlažnosti i temperature zraka (SHT75) te senzor tlaka zraka (MSP) spojeni su preko GPIO linija, a senzor mokrine tla spojen je na UART linije.

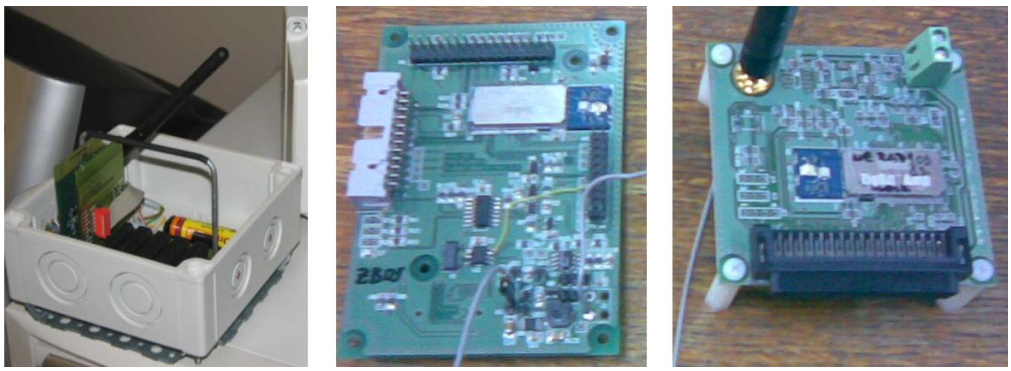


Slika 4: Blok shema osjetilnog čvora FER Čvorak v2.0

Na pločici su izvedena 4 konektora preko kojih se spaja periferija i koji ovisno o spojenoj periferiji definiraju ulogu pločice/čvora u mreži:

- Senzorski konektor – namijenjen spajanju vanjskih senzora. Konektor se koristi na čvorovima za mjerenje mikrometeoroloških podataka.
- Konektor za spajanje kamere na krajnjem čvoru s kamerom.
- *IESM* konektor za spajanje s IESM pločicom koja služi za komunikacijom s *Wavecom* GPRS mrežnim poslužiteljem/*gatewayem*. Ovaj konektor koristi se na koordinatorskom čvoru.
- *Debug* konektor za spajanje pomoćne pločice za testiranje i programiranje. Na ovom konektoru izvedena su JTAG i SPI sučelja za programiranje te tipke i LED diode za testiranje i uhodavanje bežičnog senzorskog čvora.

Slikom 5 u sredini prikazana je tiskana pločica čvora FER Čvorak v2.0. Dimenzije tiskane pločice čvora FER Čvorak v2.0 su 70 x 100 mm. Čvor nema mogućnost udaljene nadogradnje jer nema dovoljno programske memorije za spremanje programskog koda.

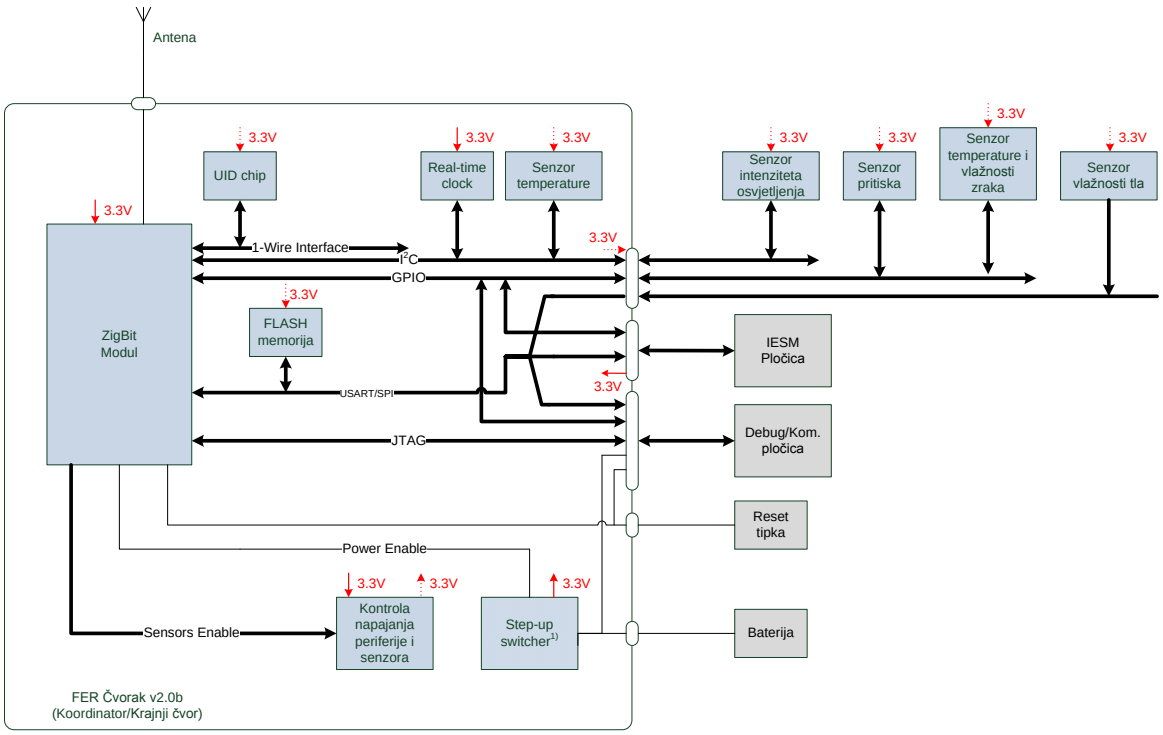


Slika 5: Usporedba triju generacija čvora FER Čvorak

3 Sklopovlje

3.1 Arhitektura čvora FER Čvorak v2.0b

FER2 Čvorak v2.0b je projektiran kao nadogradnja na dizajn postojećeg čvora FER Čvorak v2.0. Čvor je također projektiran kao univerzalna tiskana pločica koja može preuzeti ulogu bilo kojeg čvora ZigBee mreže. Blok shema čvora prikazana je slikom 6.



Slika 6: Blok shema osjetilnog čvora FER Čvorak v2.0b

Čvor dijeli većinu komponenata s postojećim čvorom FER Čvorak v2.0. Najvažnija nova mogućnost je dodatak vanjske FLASH memorije s kojoj se pristupa preko SPI sučelja. Ova memorija služi za spremanje programskog koda i daje čvorovima mogućnost udaljene nadogradnje programske podrške. Također, promijenjen je sustav napajanja zbog problema sa sklopkama AP2280 (Diodes Incorporated, 2009.) korištenim u postojećem sustavu upravljanja DC/DC pretvornikom. Također, ispravljeni su svi dosad uočeni propusti u električkim shemama. Ideja je da nova verzija čvora bude manjih dimenzija i pogodna za ugradnju u robusno kućište za postavljanje na teren. Dogovoreno je da se u svrhu smanjenja dimenzija svi priključci spajaju preko univerzalnog konektora.

Paralelno s osjetilnim čvorom razvija se i nova verzija komunikacijske pločice s Wavecom mrežnim poslužiteljem/*gatewayem*. Čvor FER Čvorak v2.0b opisan u nastavku projektiran tako da bude potpuno kompatibilan s navedenim uređajem. Također, za potrebe

testiranja i uhodavanja čvora FER Čvorak v2.0b i nove Wavecom pločice razvijena je nova verzija *debug*-pločice.

Slijedi detaljni opis sklopovlja čvora FER čvorak v2.0b po podsustavima.

3.1.1 Sustav napajanja

3.1.1.1 Zahtjevi za napajanjem čvora FER Čvorak v2.0b

Sklopovi bežičnog osjetilnog čvora FER Čvorak v2.0b birani su za nominalni napon napajanja VCC = 3,3 V. U primjeni na terenu čvor može biti napajan bilo iz primarnih elektrokemijskih izvora ili iz punjivih NiCd odn. Li-ion akumulatora. Korišteni sekundarni elektrokemijski izvori mogu raditi samostalno ili biti dio solarnog izvora napajanja, ovisno o ciljanom životnom vijeku.

Karakteristika baterijskih i akumulatorskih izvora napajanja je da im se pražnjenjem napon smanjuje nelinearno. Tipični napon punog Li-ion akumulatora je $VBAT_{max}$ = 3,7 V, dok napon potpuno praznog akumulatora iznosi oko 2,5 V. Pritom nagib krivulje pražnjenja ovisi o struji pražnjenja. Dakle, ovisno o trenutno preostalom kapacitetu akumulatora, ulazni napon može biti i viši i niži od potrebnog nominalnog VCC = 3,3 V. Uslijed smanjenja napona akumulatora ispod dozvoljenog minimalnog napona napajanja kritičnih sklopova (posebno su bitni mikrokontroler i radio-komunikacijski sklop) postoji opasnost od prestanka rada osjetilnog čvora.

Također, snaga izlaznog pojačala radiokomunikacijskog sklopa ovisi o naponu napajanja izlaznog pojačala radio-sklopa. Zbog toga je potrebno osigurati da neovisno o trenutno preostalom kapacitetu akumulatora, napon napajanja sklopova bežičnog osjetilnog čvora bude konstantan i dovoljnog iznosa. To se postiže upotrebom zapornog izvora napajanja s prekidanjem struje (engl. *switching mode power supply*), tj. DC/DC pretvornika (engl. *step-up/boost DC/DC converter*). Upotreba zapornog pretvornika omogućuje napajanje osjetilnog čvora naponom manjim od VCC i time optimalno iskorištenje kapaciteta akumulatora.

U svrhu smanjenja potrošnje komponente DC/DC pretvornika, potrebno je moći isključiti pretvornik kad je napon akumulatora dovoljno visok za sigurno napajanje bežičnog osjetilnog čvora. Zato odabrana komponenta DC/DC pretvornika mora imati mogućnost isključivanja. Tipični izlazni regulirani napon pretvornika s fiksnim izlaznim naponom je VOUT = 3,3 V. Napon punog Li-Ion akumulatora $VBAT_{max}$ = 3,6 V, a minimalni napon na

kojem rade svi sklopovi na pločici osjetilnog čvora VCC_{min} = 2,7 V. Iz toga slijedi da je dovoljno imati pretvornik uključen tek za napone niže od 2,7 V. Posebno je nepotrebno imati uključen pretvornik za napone iz raspona [VOUT, $VBAT_{max}$] jer je karakteristika zapornih pretvornika da ne reguliraju izlazni napon ukoliko je ulazni napon VBAT viši od deklariranog izlaznog napona pretvornika VOUT.

DC/DC pretvornik se gasi automatski na temelju izlaznog signala komparatora koji mjeri ulazni napon VBAT ili ručno pomoću signala iz mikrokontrolera.

Kad je pretvornik ugašen, osjetilni se čvor napaja izravno iz akumulatora. Zbog toga je potrebno osmisliti način na koji će se napon akumulatora VBAT propustiti od ulaza do izlaza sustava napajanja a da ne bude reguliran. Moguća su dva rješenja:

1. Upotreba diskretnih sklopki kojima se put signala VBAT preusmjerava na alternativni vod, a DC/DC pretvornik galvanski odvaja od ulaza i izlaza sustava napajanja. Navedeni pristup korišten je kod bežičnog osjetilnog čvora FER Čvorak v2.0. Pristup se pokazao lošim zbog povećanog broja komponenata i problema sa sklopkama AP2280.
2. Odabrati DC/DC pretvornik koji kad je ugašen propušta napon s ulaza na izlaz. Zbog jednostavnosti ovaj je pristup korišten u dizajnu čvora FER Čvorak v2.0b.

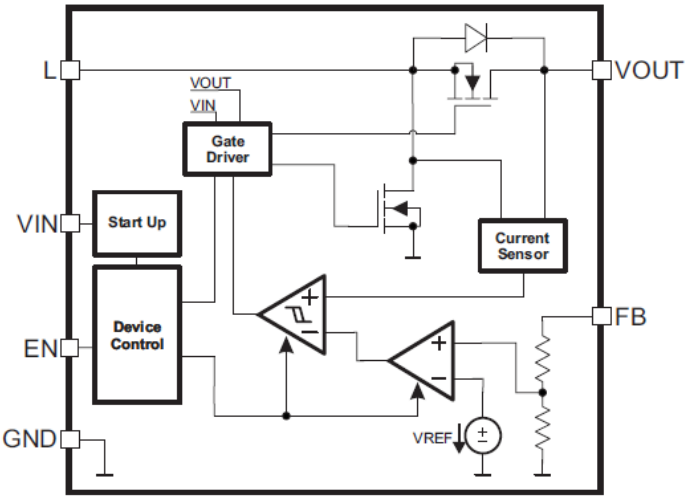
Osim preko DC/ DC pretvornika potrebno je osigurati mogućnost napajanja krajnjeg čvora i iz sljedećih izvora:

1. napajanje iz vanjskog laboratorijskog izvora napona preko pomoćne *debug* pločice naponom VBAT (koristi se DC/DC pretvornik na pločici osjetilnog čvora) i VCC (izravno, bez korištenja DC/DC pretvornika) – koristi se prilikom testiranja i uhodavanja sustava u laboratoriju
2. Napajanje sa VCC = 3,3 V od ugradbenog mrežnog poslužitelja/gatewaya Wavecom – koristi se kod koordinatorskog čvora na terenu. Pritom se ne koristi DC/DC pretvornik osjetilnog čvora, već se dovodi stabilizirani napon s Wavecom-pločice

3.1.1.2 DC/DC pretvornik TPS61221

Korištena je komponenta TPS61221. Najniži ulazni napon mu je 0,7 V, a vlastita potrošnja mu iznosi 7,5 µA kad radi i 0,5 µA kad je isključen. Maksimalna struja koju pretvornik može dati na izlazu uz ulazni napon od VBAT = 2,7 V (prag rada svih sklopova) je 175 mA

što je dovoljno za napajanje cijelog osjetilnog čvora. (Texas Instruments, 2009.) Unutarnja građa komponente prikazana je slikom 7. Na njoj se vidi mehanizam koji omogućuje propuštanje ulaznog napona na izlaz u stanju kad je pretvornik isključen – blok označen kao *Gate Driver* upravlja MOSFET sklopkom spojenom serijski između ulaza sklopa L i izlaza VOUT. Kad je pretvornik ugašen (ulaz EN = 0), serijska sklopka vodi i prosljeđuje ulazni napon na izlaz.

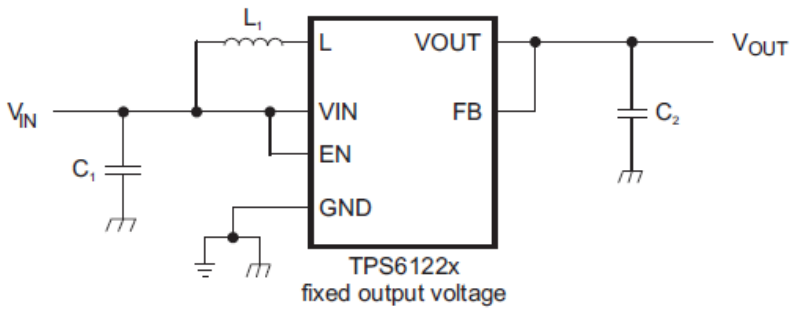


Slika 7: Blok shema građe DC/DC pretvornika TPS61221 (Texas Instruments, 2009.)

Pretvorniku je na ulazu još potrebno dodati paralelno spojen kondenzator C_1, serijski spojenu zavojnicu L_1 i izlazni kondenzator C_2. Iznosi komponenata predloženi u (Texas Instruments, 2009.) su dani u tablici 1. Na slici 8 prikazana je shema spajanja pretvornika u krug zajedno s pasivnim komponentama.

Tablica 1: Iznosi pasivnih komponenti DC/DC pretvornika

Komponenta	**Vrijednosti**
C1	10 µF, 6,3 V, X5R keramički kondenzator
C2	10 µF, 6,3 V, X5R keramički kondenzator
L1	4,7 µH



Slika 8: Shema spajanja pretvornika TPS61221 u krug (Texas Instruments, 2009.)

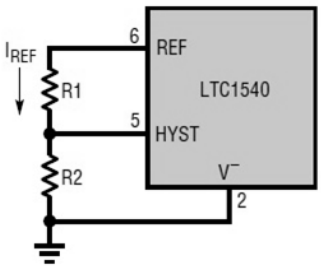
3.1.1.3 Komparator LTC1540

Za upravljanje radom DC/DC pretvornika korišten je komparator LTC1540. Potrošnja sklopa je 0,71 µA (Linear Technology, 1997.). Dizajn komparatora i odabrane vrijednosti pasivnih komponenata koje određuju prag komparacije i histerezu preuzeti su iz dizajna postojećeg čvora FER Čvorak v2.0. Iznosi pasivnih komponenata su dani u tablici 2:

Tablica 2: Iznosi pasivnih komponenata komparatora LTC1540

komponenta	iznos
R14	4,7 MΩ
R16	3,9 MΩ
R18	66,5 kΩ
R19	5,6 MΩ

U nastavku je izveden proračun prema (Texas Instruments, 2009.) kojim su za vrijednosti komponenata iz tablice 2 provjereni iznosi praga komparacije i histereze. Histereza se dobije formulom (3) uvrštavanjem formule (2) u (1) dobivenih sa slike 9 (uz zamjenu R18 = R1, R19 = R2):



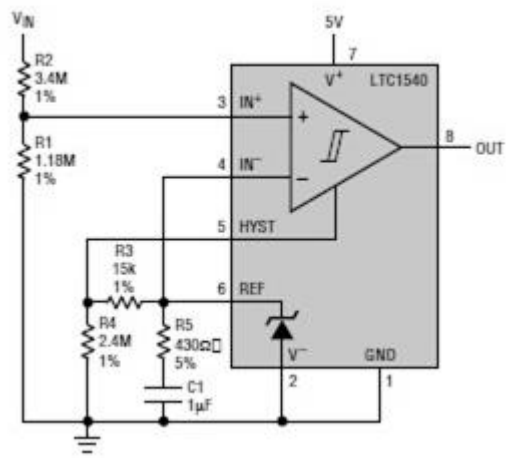
Slika 9: Proračun histereze (Linear Technology, 1997.)

$$R19 = (1{,}182 - \frac{V_{HB}}{2})/I_{REF} \quad (1)$$

$$I_{REF} = V_{HB}/(2 \cdot R18) \quad (2)$$

$$V_{HB} = 1{,}182\ V\ /(\frac{1}{2} + \frac{R19}{2 \cdot R18}) = 27{,}277\ \text{mV} \quad (3)$$

Nadalje, potrebno je odrediti napon praga komparacije. On se može izvesti iz formula na slici 10 (uz zamjenu R14 = R2, R16 = R1):



Slika 10: Proračun praga komparacije (Linear Technology, 1997.)

Iz toga iz formule (4) dobivene sa slike 12 slijedi napon praga V_{IN} prema formuli (5):

$$R14 = R16 \cdot \left(V_{IN}/(V_{REF} + \frac{V_{HB}}{2}) - 1\right) \quad (4)$$

$$V_{IN} = 1{,}103\ V\ \cdot V_{HB} + 2{,}606\ V = 2{,}636\ V \quad (5)$$

V_{HB} je iznos histereze na ulazu komparatora. Iznos histereze gledano na VBAT (odakle se dovodi signal) je V_{HBIN}. Veza između VHB i VHBIN može se odrediti iz formula (3) i (4) pa je konačan iznos histereze dan formulom (5).

$$k = \frac{V_{REF}}{V_{IN}} \quad (3)$$

$$V_{HB} = \frac{V_{HBIN}}{k} \quad (4)$$

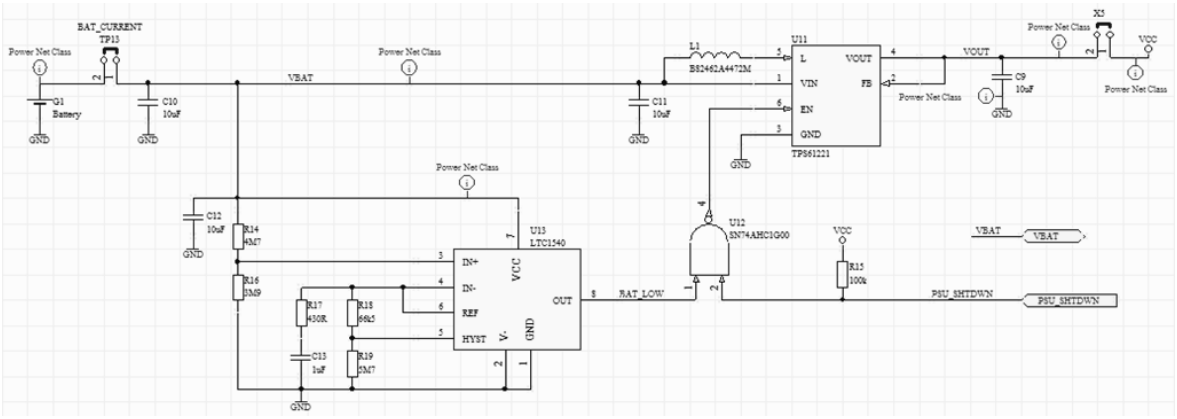
$$V_{HBIN} = V_{HB} \cdot \frac{V_{IN}}{1{,}182\ V} = 61{,}88\ mV \quad (5)$$

Ovaj iznos napona histereze je dovoljan da se u okolini željenog iznosa napona praga uključivanja DC/DC pretvornika eliminira osciliranje izlaza komparatora.

Ovisno o scenariju, prag od 2,636 V može i ne mora biti premalo. Ako se smatra da je DC/DC pretvornik upaljen uvijek kad je čvor u budnom stanju (gašenje DC/DC pretvornika u stanju spavanja ako je baterija dovoljno puna), cijelo budno stanje, onda nema problema. No, ako bi se željelo pogoniti čvor i u budnom stanju bez DC/DC pretvornika, onda neće raditi senzor temperature U3 ugrađen na pločicu (senzor LM73 radi od 2,7 V). Tome se može doskočiti da se uzme R18 = 5,6 MΩ pa iz proračuna onda ispada konzervativniji prag komparacije VIN = 2,913 V**.** Nedostatak ovoga je to što se će se gledano na karakteristici pražnjenja baterije prije uključiti DC/DC pretvornik pa će se do kraja životnog ciklusa baterije imati ukupno više perioda spavanja s uključenim DC/DC pretvornikom. Kad se tome pridoda niski radni omjer čvora (periodi spavanja traju mnogo duže od aktivnih stanja), više se isplati početna inačica R18 = 4,7 MΩ ali uz obavezno uključen DC/DC pretvornik barem tokom dijagnostičkog mjerenja temperature pomoću senzora LM73 u budnom stanju.

3.1.1.4 Izvedba podsustava za napajanje

Cjelovita shema podsustava napajanja dana je slikom 11.



Slika 11: Električka shema podsustava za napajanje

Glavna komponenta podsustava je U11 (komponenta TPS61221). Zajedno sa zavojnicom L1 i kondenzatorom C11 na ulazu te kondenzatorom C9 na izlazu realizira DC/DC pretvornik zaporne topologije (engl. *step-up/boost converter*). Na ulaz pretvornika dovodi se napon akumulatora VBAT filtriran kondenzatorom C10.

Kratkospojnik TP13 kad je odspojen omogućuje serijsko priključenje ampermetra ili spajanje vanjskog voda na koji se može priključiti strujna sonda za potrebe mjerenja potrošnje struje.

Na izlazu postoji kratkospojnik X5. Njegova uloga je da u otvorenom položaju galvanski odvoji izlaz DC/DC pretvornika u slučaju da se koordinatorski čvor napaja naponom VCC iz Wavecom-GPRS pločice (zaštita izlaza DC/DC pretvornika) ili se dovodi napajanje preko debug pločice kod testiranja. Funkcije kratkospojnika TP13 i X5 sažete su tablicom 3:

Tablica 3: Funkcije kratkospojnika TP13 i X5

položaj kratkospojnika	**TP13**	**X5**
spojeno	normalni rad	vlastito napajanje čvora preko konektora G1 iz akumulatora (bez obzira radi li DC/DC pretvornik)
odspojeno	mjerenje struje akumulatora	Napajanje čvora iz vanjskog izvora preko Wavecoma ili debug pločice

Radom DC/DC pretvornika upravljaju komparator U13 (LTC1540) i izlazni signal mikrokontrolera PSU_SHTDWN. Kad je napon akumulatora VBAT niži od podešenog napona praga komparacije V_{komp}, DC/DC pretvornik je uključen neovisno o stanju na liniji PSU_SHTDWN. Za slučaj da je napon VBAT viši od napona praga komparacije V_{komp} (akumulator je dovoljno pun), onda je moguće programski postavljanjem signala PSU_SHTDWN u nisko stanje isključiti DC/DC pretvornik i time uštedjeti energiju.

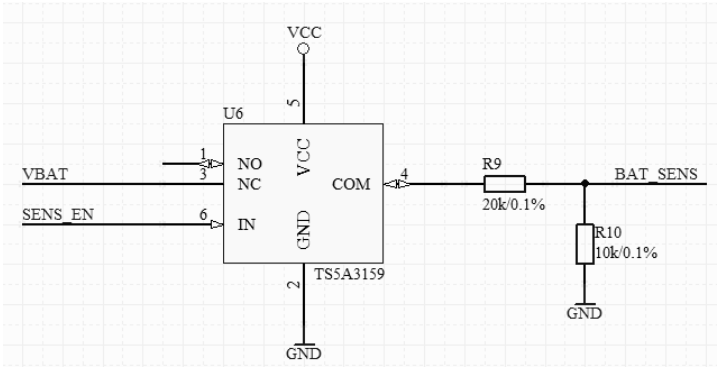
U stanju spavanja mikrokontroler ne kontrolira liniju PSU_SHTDWN pa otpornik R15 pritegnut na napon napajanja VCC osigurava gore opisanu logiku. Logika upravljanja izvorom napajanja sažeta je tablicom 4:

Tablica 4: Logika upravljanja izvorom napajanja

izlaz LTC1540 OUT	**signal PSU_SHTDWN**	**ulaz TPS61221**
0 (VBAT < Vkomp)	0	1 (DC/DC uključen)
0 (VBAT < Vkomp)	1	1 (DC/DC uključen)
1 (VBAT > Vkomp)	0	1 (DC/DC uključen)
1 (VBAT > Vkomp)	1	0 (DC/DC isključen)

Uz navedene otpore R14, R16, R18 i R19, prag komparacije je 2,636 V što je dovoljno za napajanje svih ključnih komponenata osim dijagnostičkog senzora temperature bez da DC/DC pretvornik radi. Korišteni logički NI-sklop U12 troši 10 µA (Texas Instruments, 2008.).

Čvor može mjeriti napon vlastitog akumulatora VBAT. VBAT se dovodi na sklopku U6 (komponenta TS5A3159). Sklopka U6 vodi kad je signal SENS_EN u niskoj razini. Izmjereni napon se prosljeđuje na otporničko dijelilo R9-R10 odakle se signal vodi na ulaz analogno-digitalnog pretvornika mikrokontrolera BAT_SENS kako je prikazano slikom 12:



Slika 12: Sklopka TS5A3159 za omogućavanje mjerenje napona akumulatora

3.1.2 Upravljanje perifernim sklopovima

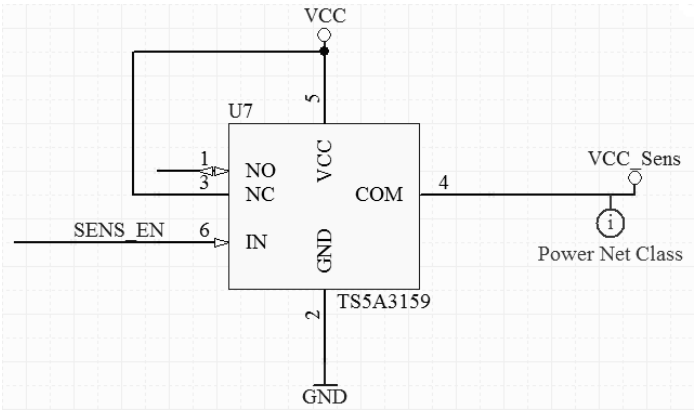
U svrhu smanjenja potrošnje, potrebno je omogućiti isključivanje perifernih sklopova, senzora i kamere, koji se izvana spajaju na pločicu osjetilnog čvora. U tu svrhu potrebne su sklopke za selektivno isključivanje periferije.

Za upravljanje periferijom koriste se analogne sklopke TS5A3159. Troše struju napajanja od 0,1 µA, a najveće struje curenja na digitalnim ulaznim pinovima su im reda veličine do 1 µA. Otpor ovih sklopki u vođenju iznosi 1 Ω. Na pločici postoje 3 ovakve sklopke. Upravljane su signalima sa ZigBit modula. Njihove uloge su popisane u tablici 5.

Tablica 5: Analogne sklopke za gašenje periferije

oznaka sklopke	Namjena	upravljački signal s modula ZigBit Amp
U7	Isključivanje napona napajanja senzora.	SENS_EN
U8	Isključivanje napona napajanja kamere.	CAM_EN
U6	Dovođenje napona napajanja na ulaz A/D pretvornika	SENS_EN

Sklopke rade na način da ovisno o upravljačkom signalu na ulazu IN spajaju ili odspajaju napajanje senzora VCC_Sens (odnosno napajanje kamere VCC_Cam) na napon napajanja VCC kao što je prikazano shemom sklopke U7 na slici 13. Sklopke vode kad je signal na ulazu IN u niskom stanju (Texas Instruments, 2004.) (SENS_EN i CAM_EN dakle rade u inverznoj logici).



Slika 13: Sklopka TS5A3159 za upravljanje napajanjem senzora

Kad čvor ode u stanje spavanja, mikrokontroler postavlja linije SENS_EN i CAM_EN u stanje visoke impedancije. Kako u stanju spavanja periferija mora biti isključena, koriste se otpornici R5 i R6 koji pritežu linije SENS_EN i CAM_EN na napon napajanja. Time su sklopke U7 i U8 isključene.

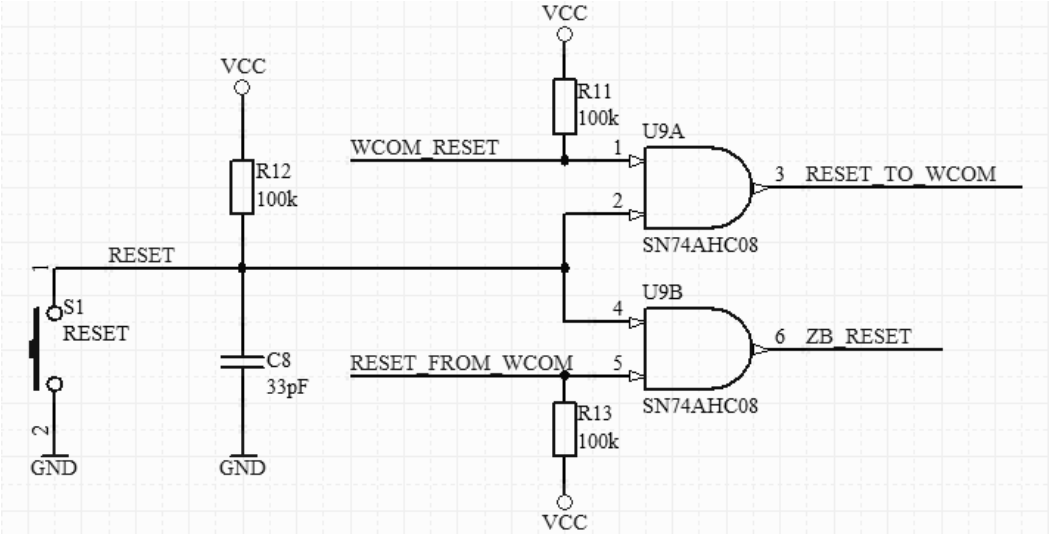
Kao što je već spomenuto u poglavlju 3.1.1.4., sklopka U6 služi dovođenju napona akumulatora na analogno-digitalni pretvornik modula mikrokontrolera (spaja ili odspaja napon VBAT od ulaza u ADC). Upravljana je kao i senzorska sklopka signalom SENS_EN.

Kamerom se upravlja pomoću sklopke U8. Izlaz sklopke U8 je napon napajanja kamere VCC_Cam. Zbog učestalih problema s propadima napona napajanja kamere VCC_Cam u praksi, na izlaz sklopa U8 dodan je blokadni kondenzator C22. Napon VCC_Cam služi kao ulaz sklopu U10 (STM810) (ST Microelectronics, 2010.). Radi se o sklopu za reset kamere. Dakle, signalom CAM_EN se kamera ujedno i resetira.

3.1.3 Sklopovlje za reset

Bežični osjetilni čvor FER Čvorak v2.0b posjeduje iste mehanizme resetiranja kao i čvor FER Čvora v2.0. Reset igra osobito važnu ulogu na koordinatorskom ZigBee čvoru koji je u mreži Maslinet serijskim UART sučeljem spojen s GPRS *gatewayem*/ugradbenim

poslužiteljem Wavecomom. Zamišljeno je da svaki od dvaju sklopova može u slučaju pogreške u radu programski resetirati drugi sklop i da postoji zajednički vanjski (sklopovski) reset koji resetira oba sklopa. Na slici 14 prikazana je logika za reset.



Slika 14: Shema logičkog sklopa za res et

Svi signali vezani uz reset sklopovlje (RESET, WCOM_RESET, RESET_FROM_WCOM, RESET_TO_WCOM i ZB_RESET) su aktivni u niskoj razini. Prema slici 16 sklopovlje za reset obavlja sljedeće zadaće:

1. Tipka S1 aktivira RESET signal koji istovremeno resetira ZigBit modul i Wavecom. Na terenu je umjesto tipke na stezaljke S1 moguće spojiti *reed-relej* montiran s unutarnje strane kućišta da se izbjegne potreba za izvođenjem tipke na vanjsku stranu kućišta (zaštita od vlage). U tom slučaju se uređaj resetira magnetom s vanjske strane kućišta.
2. WCOM_RESET je izlazni signal ZigBita. Postavljanjem u nisku razinu spušta signal RESET_TO_WCOM u nisku razinu čime resetira Wavecom.
3. RESET_FROM_WCOM je izlazni signal Wavecoma. Spuštanjem u nisku razinu, signal ZB_RESET se spušta u nisku razinu i resetira ZigBit.

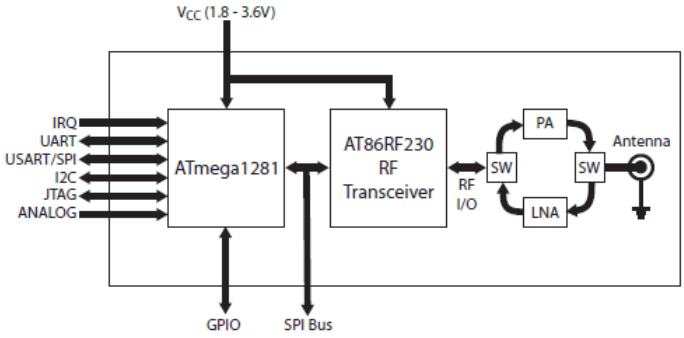
Otpornici R11, R12 i R13 drže izlazne signale logičkih sklopova U9A i U9B u logičkoj jedinici (neaktivno stanje). Kondenzator C8 filtrira istitravanje kontakata tipke S1.

Sklop U9 troši u radu 20 µA (Texas Instruments, 2003.).

3.1.4 Obrada podataka, radiokomunikacija

3.1.4.1 Modul ZigBit Amp

Za obradu podataka i radio-komunikaciju korišten je Atmelov SiP modul ZigBit Amp sastavljen od mikrokontrolera ATmega1281 i radio-komunikacijskog sklopa AT86RF230. Blok shema modula vidi se na slici 15.



Slika 15: Blok shema modula ZigBit Amp (Atmel, 2009.)

ATmega1281 ima 128 kilobajta ugrađene programske FLASH memorije i 8 kilobajta RAM memorije. Komunikacijski sklop AT86RF230 radi na frekvenciji 2,4 GHz i predviđen je za rad s IEEE 802.15.4/ZigBee bežičnim mrežama. Karakteriziraju ga velika osjetljivost ulaznog pojačala prijemnika (-104 dBm) i velika snaga izlaznog pojačala predajnika (+20 dBm, što daje maksimalno dozvoljenih 100 mW izlazne snage u ISM području) (Atmel, 2009.).

SiP modul dolazi na kompaktnoj PCB pločici malih dimenzije (38.0 x 13.5 x 2.0 mm) namijenjenoj za površinsku montažu. U Amp verziji, modul dolazi s balansiranim antenskim izlazom. Antena se koaksijalnim kabelom spaja na U.FL konektor na modulu.

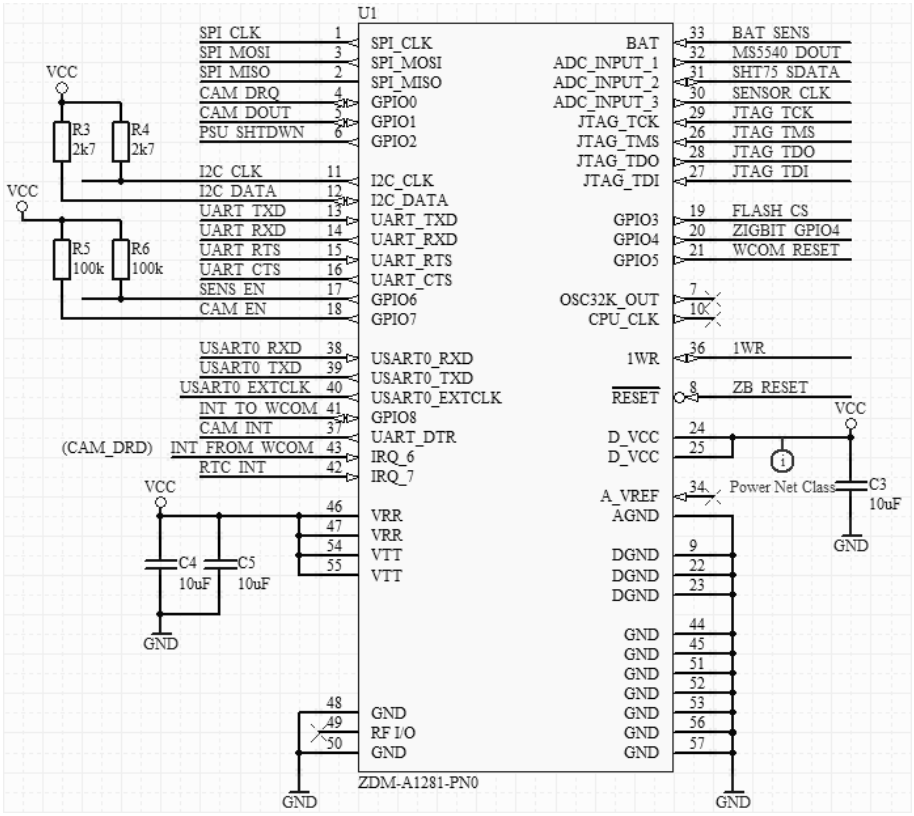
Od ulazno-izlaznih sučelja, modul nudi 9 GPIO linija, 2 ulazne prekidne linije, 4 ADC linije, jednu zasebnu analognu liniju za praćenje napona napajanja, UART, USART, I^2C, SPI i *1-Wire* sučelje. Od toga se čak 30 linija može prenamijeniti u GPIO digitalne linije opće namjene (Atmel, 2009.). U tablici 6 je opisana funkcionalnost pojedinih pinova. Smjerovi signala (ulazni/izlazni) promatraju se iz perspektive ZigBit modula, osim ako nije drugačije navedeno. Slika 16 prikazuje električku shemu modula ZigBit sa izvodima prema tablici 6.

Tablica 6: Popis korištenih pinova modula ZigBit Amp

priključnica	naziv signala	opis
1	SPI_CLK	U normalnom radu se ne koristi. Rezervirano za internu komunikaciju između mikrokontrolera i RF sklopa. Prilikom programiranja modula preko SPI konektora na debug pločici koristi se kao ulazni signal takta.
2	SPI_MISO	Ne koristi se. Rezervirano za internu komunikaciju između mikrokontrolera i RF sklopa. Nije spojen nigdje.
3	SPI_MOSI	Ne koristi se. Rezervirano za internu komunikaciju između mikrokontrolera i RF sklopa. Nije spojen nigdje.
4	CAM_DRQ	Upravljački signal kamere
5	CAM_DOUT	Izlazni signal kamere.
6	PSU_SHTDWN	Isključivanje DC/DC pretvornika. Aktivan u niskom stanju.
7	OSC32K_OUT	Izvor takta 32 kHz. U praksi daje zašumljen signal pa se ne koristi. Nije spojen nigdje.
8	ZB_RESET	Reset ZigBit modula. Aktivan u niskoj razini.
9, 22, 23	DGND	Masa digitalnih sklopova modula ZigBit..
10	CPU_CLK	Izlaz generatora takta. Daje signal frekvencije 4 MHz.
11	I2C_CLK	Signal takta I^2C sabirnice. Izlazni signal. Pritegnut je na napon napajanja otpornikom R4 iznosa 2,7 kΩ po I^2C specifikaciji.
12	I2C_DATA	Podatkovna linija I^2C sabirnice. Pritegnut je na napon napajanja otpornikom R3 iznosa 2,7 kΩ po I^2C specifikaciji.
13	UART_TXD	Podatkovna linija za primanje preko UART serijskog sučelja. Ulazna linija. Vodi se na *debug* pločicu. Koristi se za komunikaciju s osobnim računalom.
14	UART_RXD	Podatkovna linija za slanje preko UART serijskog sučelja. Izlazna linija. Linija za primanje preko UART serijskog sučelja. Ulazna linija. Vodi se na *debug* pločicu. Koristi se za komunikaciju s osobnim računalom.
15	UART_RTS	*Request to send* ulazna linija za serijsko sučelje UART. Linija za primanje preko UART serijskog sučelja. Ulazna linija. Vodi se na *debug* pločicu. Koristi se za komunikaciju s osobnim računalom.
16	UART_CTS	*Clear to send* izlazna linija za serijsko sučelje UART. Linija za

		primanje preko UART serijskog sučelja. Ulazna linija. Vodi se na *debug* pločicu. Koristi se za komunikaciju s osobnim računalom.
17	SENS_EN	Izlazni signal za uključivanje napajanja senzora VCC_Sens i omogućavanje mjerenja napona akumulatora (uključuje signal BAT_SENS). Signal je aktivan u niskoj razini.
18	CAM_EN	Izlazni signal za uključivanje napajanja kamere VCC_Cam. Signal je aktivan u niskoj razini.
19	FLASH_CS	Uključuje vanjsku FLASH memoriju. Koristi se u sekvencama pristupa FLASH memoriji. Signal je aktivan u niskoj razini.
20	ZIGBIT_GPIO4	Neiskorišteni digitalni pin opće namjene. Izveden je na konektor X1.
21	WCOM_RESET	Izlazna priključnica za resetiranje Wavecom modula. Aktivan u niskoj razini.
24, 25	D_VCC	Napajanje digitalnog dijela sklopova ZigBit modula (VCC)
26	JTAG_TMS	*JTAG Test Mode Select*. Služi za programiranje preko JTAG sučelja na *debug* pločici.
27	JTAG_TDI	*JTAG Test Mode Input.* Služi za programiranje preko JTAG sučelja na *debug* pločici.
28	JTAG_TDO	*JTAG Test Data Output.* Služi za programiranje preko JTAG sučelja na *debug* pločici.
29	JTAG_TCK	*JTAG Test Clock.* Služi za programiranje preko JTAG sučelja na *debug* pločici.
30	SENSOR_CLK	Digitalni signal opće namjene namijenjen komunikaciji sa senzorima. U projektu Maslinet korišten kao izlazni signal takta za senzor temperature i vlažnosti zraka SHT75.
31	SHT75_DATA	Digitalni signal opće namjene namijenjen komunikaciji sa senzorima. U projektu Maslinet korišten kao ulazni signal za primanje podataka sa senzora temperature i vlažnosti zraka SHT75.
32	MS5540_DOUT	Digitalni signal opće namjene namijenjen komunikaciji sa senzorima. U projektu Maslinet korišten kao ulazna linija ZigBita preko koje senzor tlaka zraka MS5540 šalje podatke ZigBitu.

33	BAT_SENS	Ulaz analogno digitalnog pretvornika. Koristi se za mjerenje napona akumulatora.
34	A_VREF	Izvor referentnog napona analogno digitalnog pretvornika. Budući da se koristi unutarnja *band-gap* referenca, priključnica može ostati nespojena.
35	AGND	Masa analognog dijela ZigBitovih sklopova.
36	1WR	Jednožično sučelje za komunikacijom s UID sklopom (vanjska memorija sa zapisanom MAC adresom) U4.
37	UART_DTR	*Data terminal ready* upravljačka linija za serijsko sučelje UART. Ulazna linija. Aktivna je u niskoj razini. Vodi se na *debug* pločicu. Koristi se za komunikaciju s osobnim računalom.
38	USART0_RXD	Ulazna podatkovna linija (gledano sa ZigBita) za USART0 sučelje. Pinovi USART0 sučelja se koristi za komunikaciju sa senzorima (u Maslinetu senzor mokrine tla ECH_2O) i SPI FLASH memorijom.
39	USART0_TXD	Izlazna podatkovna linija (gledano sa ZigBita) za USART0 sučelje. Pinovi USART0 sučelja se koristi za komunikaciju sa senzorima (u Maslinetu senzor mokrine tla ECH_2O)) i SPI FLASH memorijom.
40	USART0_EXTCLK	Izlazna linija signala takta (gledano sa ZigBita) za USART0 sučelje. Koristi se samo za komunikaciju s vanjskom SPI FLASH memorijom.
41	INT_TO_WCOM	Izlazna linija. Izaziva prekid na Wavecomu.
42	RTC_INT	Ulazna linija. Prekid izazvan RTC sklopom.
43	INT_FROM_WCOM	Ulazna linija. Prekid izazvan Wavecomom.
44, 45, 51, 52, 53, 56, 57	GND	Masa digitalnog dijela sklopovlja ZigBita.
46, 47	VRR	Napajanje radio-prijemnika.
48, 50	GND	Analogna masa RF sklopovlja.
49	RF_IO	Diferencijalni RF ulaz/izlaz. Ne koristi se.
54, 55	VTT	Napajanje radio-predajnika.



Slika 16: Shema modula ZigBit Amp

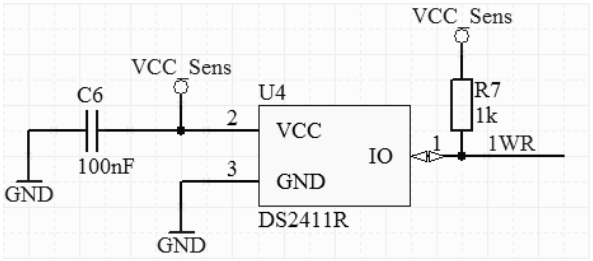
Modul ZigBit ima tri odvojena napajanja:

1. D_VCC je napajanje digitalnih sklopova (mikrokontroler ATmega 1281 i digitalni dio radio-prijemnika AT86RF230)
2. VRR je analogno napajanje prijemnika (engl. *receiver*).
3. VTT je analogno napajanje predajnika (engl. *transmitter*).

Sva tri napajanja napajana su iz istog izvora (napon VCC iz DC/DC pretvornika). Uz svaki par priključnica napajanja nalazi se blokadni kondenzator kapaciteta 10 µF (C3, C4, C5).

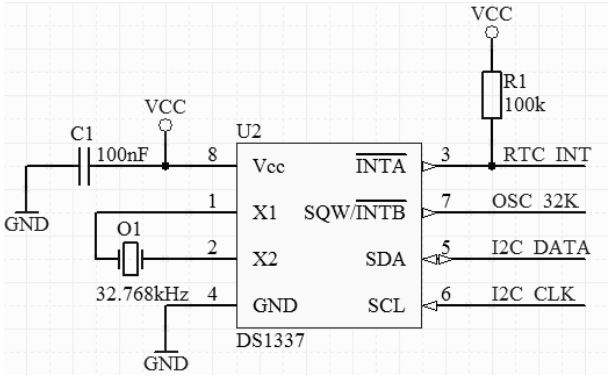
3.1.4.2 Pomoćni sklopovi

Sklop U4 (DS2411R) služi za pohranu MAC adrese bežičnog čvora. To je 64-bitna ROM memorija. Sadrži pohranjen 48-bitni jedinstveni serijski broj koji se koristi kao MAC adresa, 8-bitni CRC kod i 8-bitni kod porodice sklopa (01h). Sklop komunicira s mikrokontrolerom preko linije IO. Struja napajanja iznosi 100 µA , a struja u stanju čekanja (engl. *standby*) 1 µA (Dallas Semiconductor). Shema mu je na slici 17.



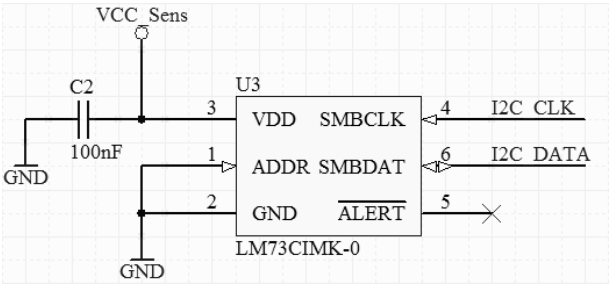
Slika 17: Shema UID sklopa DS2411R

U2 (DS1337) je RTC sklop za mjerenje protoka stvarnog vremena. Sklop se konfigurira preko I^2C sabirnice. Omogućava čitanje vremena i postavljanje dviju vrsta alarma: periodički alarm i pojedinačni alarm u predefinirano vrijeme. Na alarm sklop generira prekid na liniji RTC_INT (ulazna linija modula ZigBit). Na izlazu SQW/INTB DS1337 može dati pravokutni signal frekvencije 32,768 kHz što se koristi kao izvor signala vremenskog vođenja kod senzor tlaka MS5540. U aktivnom radu DS1337 troši 150 µA, a u stanju čekanja (engl. *standby*) 1,5 µA uz deklarirane struje curenja na ulazima i izlazima od najviše 1 µA (Dallas Semiconductor, 2009.). Sklop je prikazan slikom 18.



Slika 18: Shema RTC sklopa DS1337

Osjetilo temperature LM73 je digitalni senzor spojen na I^2C sabirnicu koji služi mjerenju temperature na pločici. Kako će čvorovi u prirodi biti hermetički zatvoreni u vodonepropusna kućišta i izloženi širokom rasponu temperatura, izmjerena temperatura u kućištu je veoma koristan dijagnostički podatak koji se mjeri i šalje koordinatoru svaki put kad se krajnji čvor probudi. Ovaj sklop se tretira kao senzor i gasi u spavanju pomoću signala SENS_EN. U radu komponenta troši najviše 495 µA, a u spavanju 8 µA (National Semiconductor, 2009.). Sklop je prikazan slikom 19.



Slika 19: Shema dijagnostičkog senzora temperature pločice LM73CIMK-0

3.1.5 Vanjska memorija

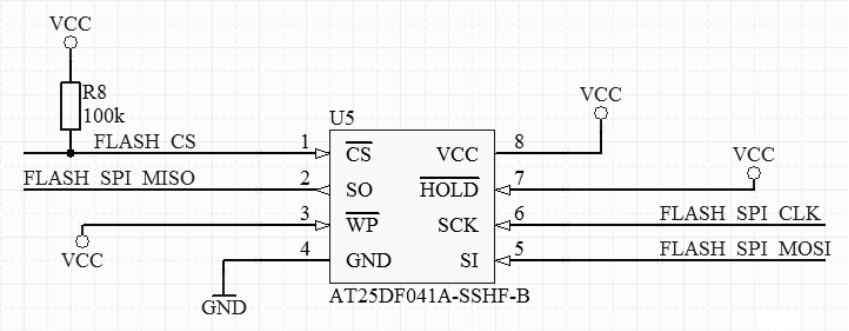
Vanjska memorija služi pohrani programskog koda koji se koristi za udaljenu nadogradnju programske podrške. U ZigBit modulu koristi se mikrokontroler ATmega1281 sa 128 KB ugrađene FLASH memorije. Dakle, prema poglavlju 2.3.3. potrebno je izabrati memoriju barem dvostrukog kapaciteta za spremanje barem dviju programskih slika.

U ponudi su memorije sa serijskim i paralelnim sučeljem. Odabrano je serijsko sučelje zbog malog broja slobodnih priključnica na ZigBit modulu. Nude se različita serijska sučelja (I2C, SPI, 3-žično). 3-žično teoretski omogućava istovremeno čitanje i pisanje jer ima odvojene linije za upis i ispis podataka (provjeriti na konkretnim komponentama), no kod nas ne dolazi u obzir jer zahtjeva dodatne GPIO pinove i komplicira izradu programske podrške (ručno postavljanje pinova bit po bit). Zbog toga je najprihvatljivije rješenje memorija sa SPI sučeljem. Također, SPI sučelje je dostupno na modulu ZigBit.

Odabrana je Atmelova serijska FLASH memorija zbog jednostavnog SPI sučelje, male potrošnje, malih dimenzija, dovoljnog kapaciteta, malog broja izvoda, dobre dokumentacija i dostupnih gotovih programskih biblioteka.

Za ovu primjenu odabrana je komponenta AT25DF041. Kapacitet joj je 4 Mbit (512 kilobajta). Obzirom na kapacitet unutarnje programske memorije modula ZigBit, to omogućuje istovremenu pohranu 4 različite inačice programskog koda, što je dovoljno za realizaciju DOTA (engl. *download over the air*) scenarija. Napaja se na 2,3 - 3,6 V što je čini idealnom u slučaju programiranja čvora uz ispražnjeni akumulator (čak pod uvjetom da DC/DC pretvornik ne radi). Memorija podržava SPI mod 0 i mod 3 sučelja. Moguće su sklopovska i programska zaštita sektora memorije. Postoji 7 sektora veličine 64 kilobajta, dva 8-kilobajtna, te po jedan 16 i 32-kilobajtni sektor. Pisanje bajt po bajt ili po stranicama (engl. *page*). Stranica je veličine 256 bajtova. Potrošnja memorije je 5 mA u aktivnom stanju i 15 µA u stanju spavanja. Dolazi u jednostavnom SOIC kućištu s 8 priključnica

(Atmel, 2008.). Shema memorije dana je slikom 20. Pin WP služi sklopovskoj zaštiti sektora i ne koristi se pa je spojen na napon napajanja. Također, mogućnost privremenog zaustavljanja rada memorije se ne koristi pa je izvod HOLD također pritegnut na napon napajanja.

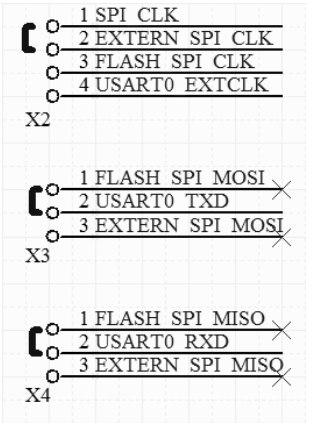


Slika 20: Shema vanjske FLASH memorije AT25DF041A

Komunikacijski izvodi FLASH_SPI_MISO, FLASH_SPI_MOSI i FLASH_SPI_CLK vode se na kratkospojnike X4, X3 i X2. Prikazani su slikom 21 i služe određivanju uloge SPI linija. Moguće kombinacije i njihova značenja dane su tablicom 7.

Tablica 7: Korištenje SPI linija i položaji kratkospojnika X2, X3 i X4

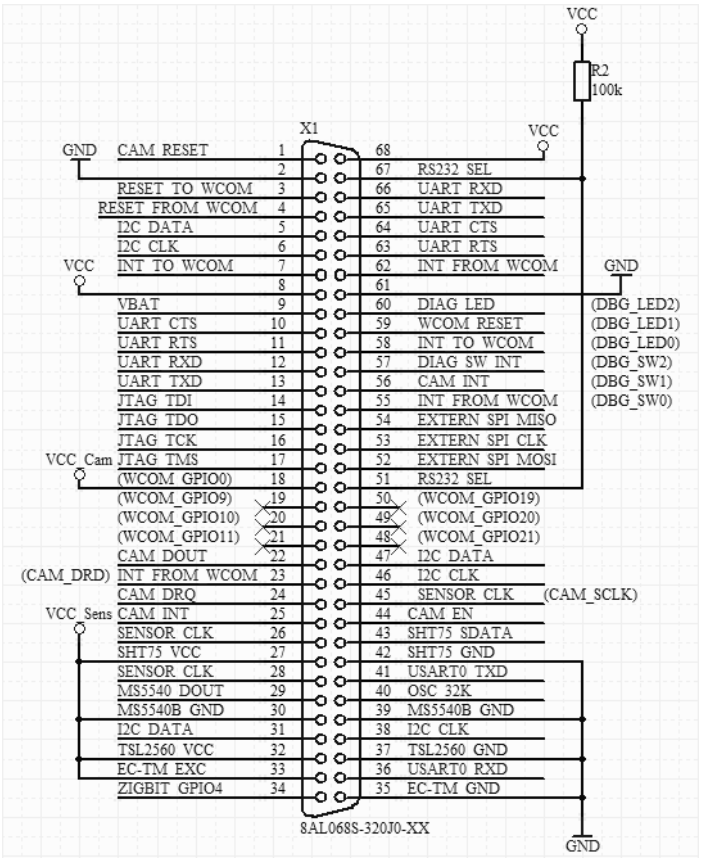
funkcija	**Položaj X2**	**Položaj X3**	**Položaj X4**
Normalni rad Flash memorije	3-4	1-2	1-2
Programiranje FLASH memorije SPI sučeljem izvana preko *debug* pločice	2-3	1-3	1-3
Programiranje ZigBit modula koristeći SPI sučelje	1-2	2-3	2-3



Slika 21: Kratkospojnici za upravljanje FLASH memorijom

3.1.6 Spajanje perifernih jedinica

Svi sklopovi koji se na nalaze na tiskanoj pločici spajaju se izvana preko jedinstvenog konektora X1. Na konektor X1 izvedeni su svi digitalni i analogni signali ZigBit modula koji se mogu iskoristiti za povezivanje s vanjskim uređajima po izboru. Unatoč tome, signali su izvedeni tako da se ostvari kompatibilnost s postojećim vanjskim jedinicama: postojećim senzorima, kamerom, Wavecom pločicom i pomoćnom *debug* pločicom. Konektor je shematski prikazan na slici 22. Fizički je izveden kao 68-pinski ženski SCSI konektor. Sve periferne jedinice spajaju se preko 68-žičnog ribon kabela na koji se priključuju muški SCSI konektori. Svi periferni sklopovi dakle posjeduju ženske SCSI konektore istovjetne X1.



Slika 22: Konektor za spajanje periferije

Signali s konektora X1 mogu se grupirati na sljedeći način:

1. Nova izvedba Wavecoma, ugradbenog poslužitelja/GPRS gatewaya koristi isti konektor kao što je X1 na pločici osjetilnog čvora. Raspored pinova za komunikaciju Wavecoma i osjetilnog čvora se na obje pločice podudaraju. Signali za komunikaciju s Wavecom prikazani su u tablici 8. Kako je značenje svih signala već opisano u poglavlju 3.1.4.1, u tablicama su dani samo komentari gdje je to potrebno.

Tablica 8: Priključnice konektora X1 za komunikaciju s Wavecom modulom

pin	naziv	Komentar
68	VCC	Vodovi napajanja - Wavecom može napajati osjetilni čvor naponom od 3,3 V
2	GND	
3	RESET_TO_WCOM	Resetira Wavecom
4	RESET_FROM_WAVECOM	Resetira ZigBee
5	I2C_DATA	I2C linije prema Wavecomu
6	I2C_CLK	
7	INT_TO_WCOM	Prekidni signal za prema Wavecom modulu
62	INT_FROM_WCOM	Prekidni signal od Wavecoma.
63	UART_RTS	UART linije prema Wavecomu. Budući da ZigBit modul Iste linije se koriste za komunikaciju s računalom preko debug pločice pa se koristi signal RS232_SEL za odabir uređaja koji koristi UART port
64	UART_CTS	
65	UART_TXD	
66	UART_RXD	
67	RS232_SEL	Kad je linija u logičkoj jedinici, Wavecom koristi ZigBitove UART linije. Linija je pritegnuta na napon napajanja pa je to normalno stanje. Kad je priključena debug pločica, **signalom RS232_SEL je moguće upravljati pomoću kratkospojnika X7 na debug pločici.** Kratkospojnik u položaju „spojeno“postavlja RS232_SEL u logičku nulu pa debug pločica koristi UART linije ZigBita. Kratkosponik X7 u položaju „odspojeno“ postavlja RS232_SEL u logičku jedinicu (Wavecom koristi UART).

2. Razvijena je nova verzija debug pločice s istovjetnim konektorom kao što je X1. Raspored pinova na obje pločice se podudaraju da se pločice mogu spojiti. Pinovi za komunikaciju s debug pločicom prikazani su u tablici 9:

Tablica 9: Priključnice konektora X1 za komunikaciju s debug pločicom

Pin	naziv	Komentar
8	VCC	Vodovi napajanja. ZigBit može biti napajan preko debug pločice. Pritom se pomoću kratkospojnika X3
9	VBAT	

61	GND	na debug pločici bira hoće li se napon dovoditi na mrežu VBAT (ulaz DC/DC pretvornika) – X3 u položaju 1-2 ili na VCC (izlaz DC/DC pretvornika) – X3 u položaju 2-3. **Ukoliko se dovodi na VCC, na ZigBit pločici je potrebno skinuti kratkospojnik X5 da se spriječi oštećenje DC/DC pretvornika U11.**
10	UART_CTS	UART linije za komunikaciju s osobnim računalom. Debug pločica se na računalo spaja preko konektora X1. Vidi napomene za pin 67 – RS232_SEL.
11	UART_RTS	
12	UART_RXD	
13	UART_TXD	
14	JTAG_TDI	Linije za programiranje preko JTAG sučelja koje se na debug pločici nalazi na konektoru X4.
15	JTAG_TDO	
16	JTAG_TCK	
17	JTAG_TMS	
51	RS232_SEL	Vidi prethodne napomene za RS232_SEL
52	EXTERN_SPI_MOSI	SPI sučelje. Spojeni na SPI konektor X5 na debug pločici. Služe za programiranje ZigBit modula ili Flash memorije izvana, preko debug pločice.
53	EXTERN_SPI_CLK	
54	EXTERN_SPI_MISO	
55	DBG_SW0	Ulazni pinovi tipki za na debug pločici.
56	DBG_SW1	
57	DBG_SW2	
58	DBG_LED0	Izlazni pinovi za upravljanje LED diodama na debug pločici.
59	DBG_LED1	
60	DBG_LED2	

3. Još ne postoji razvijeno novo sklopovsko sučelje za spajanje kamere s novim osjetilnim čvorom FER Čvorak v2.0b preko konektora X1. Zbog toga su signali preuzeti iz postojećeg dizajna. Priključnice za komunikaciju s kamerom su popisane u tablici 10.

Tablica 10: Priključnice konektora X1 za komunikaciju s kamerom

pin	**Naziv**	**komentar**
1	CAM_RESET	Resetira kameru. Dovodi se sa reset sklopa U10.
18	VCC_Cam	Napajanje kamere.

22	CAM_DOUT	Upravljački i podatkovni signali kamere.
23	CAM_DRD	
24	CAM_DRQ	
25	CAM_INT	
45	CAM_SCLK	
44	CAM_EN	Uključuje kameru. Aktivan u niskoj razini.
46	I2C_DATA	Linije I^2C sabirnice se također koriste za upravljanje kamerom.
47	I2C_CLK	

4. Senzorske priključnice organizirane su na način da se postojeći senzori korišteni u projektu Maslinet mogu neovisno spajati. Svi senzori mogu biti istovremeno spojeni na isti čvor. Svaki senzor ima vlastite pinove napajanja koje se dovodi iz zajedničke mreže VCC_Sens. Senzorske priključnice su dane u tablici 11.

Tablica 11: Priključnice konektora X1 za komunikaciju sa senzorima

pin	naziv	komentar
26	SENSOR_CLK	Upravljački, podatkovni i pinovi napajanja senzora temperature i vlažnosti zraka SHT75.
27	VCC_SENS	
42	GND	
43	SHT75_DATA	
28	SENSOR_CLK	Upravljački, podatkovni i pinovi napajanja senzora tlaka zraka MS5540B.
29	MS5540_DOUT	
30	VCC_Sens	
39	GND	
40	OSC_32K	
41	USART0_TXD	
31	I2C_DATA	Upravljački, podatkovni i pinovi napajanja senzora osvjetljenja ISL29013.
32	VCC_Sens	
37	GND	
38	I2C_CLK	
33	VCC_Sens	Upravljački, podatkovni i pinovi napajanja senzora mokrine tla ECH_2O-TM.
35	GND	
36	USART0_RXD	

GPIO4 je nesvrstani digitalni signal sa ZigBita koji se također dovodi na priključnicu 34 konektora X1. Priključnice 19, 20 i 21 te 48, 49 i 50 na konektoru X1 ostale su neiskorišteni. Navedene priključnice se koriste na istovjetnim konektorima na Wavecom i *debug* pločicama za testiranje Wavecom pločice (kad su međusobno spojene jedino Wavecom i *debug* pločice).

Zbog rasporeda priključnica, ovaj dizajn omogućuje istovremeno spajanje Wavecom i debug pločica na isti osjetilni čvor (pritom je potrebno rascijepiti jedan kraj 68-žičnog paralelnog kabela da se odvoje signali za Wavecom i *debug* pločicu na zasebne muške SCSI konektore).

3.2 Izvedba tiskane pločice

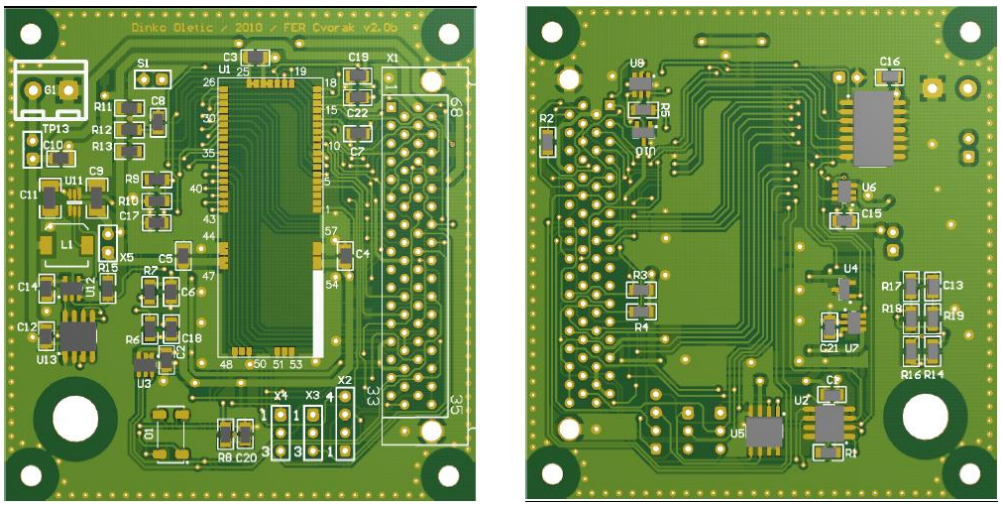
Zamišljeno je da tiskana pločica bude čim manjih dimenzija. Dimenzije referentnog postojećeg čvora FER Čvorak v2.0 iznose 70 x 100 mm. Odlučeno je da novi čvor bude također izveden u obliku jedne pločice. Prije izrade razmatrana su rješenja u kojima je čvor izveden modularno, kao stog („sendvič“) međusobno povezanih pločica (naslaganih jedna na drugu). Pritom bi pojedine slojeve činili recimo ZigBit modul, sloj za komunikaciju s periferijom i sloj napajanja. Svejedno, odlučeno da se ostane pri dizajnu s jednom pločicom. Razlog je što sendvič dizajn zahtjeva da se dio površine tiskane pločice na svakom sloju odvoji za vodove i konektore koji služe komunikaciji između slojeva. Problem je što modul ZigBit sadrži veliki broj ulazno-izlaznih pinova koji se vode na vanjski konektor pa bi mnogo površine tiskane pločice odlazilo na njihovo provođenje do konektora za spajanje između slojeva. Također, veliki su problem robusnost i pouzdanost (tj. trajnost) malih konektora s velikim brojem pinova za spajanje pločica u uređaju namijenjenom ugradnji na teren.

Iz navedenih razloga odlučeno je da se ostane pri dizajnu s jednom pločicom. U svrhu smanjenja dimenzija, komponente su postavljene s obje strane pločice. Sve komponente su grupirane po funkcionalnosti. Konačna veličina tiskane pločice je ispala 66 x 69 mm.

Na pločici su u kutovima postavljene rupe za vijke za montažu u kućište promjera 3,0 mm. U donjem, lijevom kutu se nalazi rupa promjera 6,6 mm koja služi za montažu antene na tiskanu pločicu kad se pločica koristi bez kućišta. Kad je u kućištu, antena se montira na kućište.

Sva ulazno-izlazne priključnice vode se na jedan konektor (X1). Odabran je SCSI 68-pinski ženski konektor za montažu na pločicu zbog velikog broja priključnica, malih dimenzija (horizontalni razmak između pinova 1,27 mm, vertikalni razmak između redova pinova 2,54 mm), dostupnosti, niske cijene i robusnosti. Kao što je opisano u poglavlju 3.1.6, zamišljeno je da sve pločice s kojima se spaja osjetilni čvor također imaju ovakav konektor i da se međusobno spajaju odgovarajućim 68-žičnim paralelnim kabelom na čijim su krajevima muški SCSI konektori.

Sukladno uputama iz dokumentacije (Atmel, 2009.), u svrhu zaštite od smetnji obje strane pločice su zalivene u masu, postavljene su prospojne rupe (engl. *via*) po rubu oko cijele pločice i izbjegavano je provođenje signala na gornjem sloju pločice ispod PCB-a ZigBit modula da se spriječi preslušavanje signala. Skice tiskane pločice dane su slikom 23 u nastavku.



Slika 23: Gornja (lijevo) i donja (desno) strana dizajna tiskane pločice

4 Programska podrška

4.1 Programski stog BitCloud

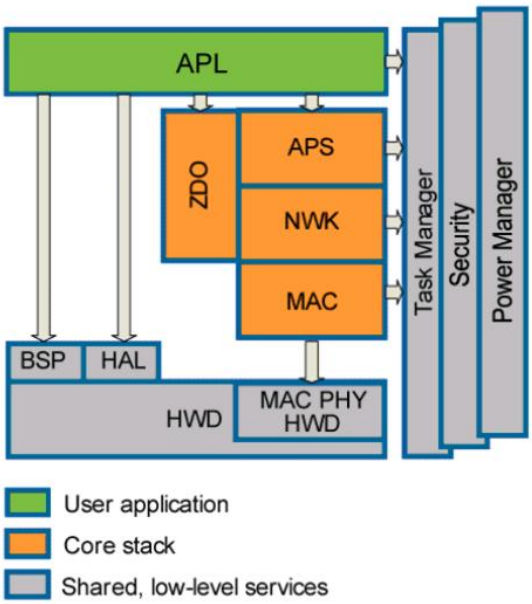
BitCloud je programska realizacija ZigBee protokola od tvrtke Atmel. Proizvođač kao temeljne značajke programskog stoga navodi (Atmel, 2009.):

- Usklađenost sa specifikacijama ZigBee i ZigBee PRO
- Programsko sučelje (API) u programskom jeziku C
- Podrška za *mesh routing* – ZigBee mreže mrežaste topologije, čime je osigurana pouzdanost prijenosa podataka
- Mogućnost izgradnje mreža s više od 100 čvorova
- Niska potrošnja energije za dugi vijek trajanja baterijski napajanih mreža
- Sigurnosni API
- Laka nadogradivost novim verzijama programskog stoga BitCloud

U nastavku slijedi pregled komponenti od kojih je sastavljen.

4.1.1 Arhitektura stoga BitCloud

Programski stog BitCloud prati strukturu ZigBee , no sadrži i dodatne slojeve kao što je prikazano slikom 24:



Slika 24: Slojevi programskog stoga BitCloud (Atmel, 2009.)

4.1.1.1 APL - aplikacijski sloj

Aplikacijski sloj je namijenjen programskom kodu korisničke aplikacije. Od programera se zahtjeva strukturiranje vlastitog koda prema unaprijed zadanoj paradigmi programskog stoga BitCloud. Ta paradigma podrazumijeva događajima upravljan programski model (engl. *event driven programming*), strukturiranje korisničke aplikacije u obliku stroja s konačnim brojem stanja (engl. *statemachine*) korištenjem aplikacijskog upravitelja zadacima i izbjegavanje blokiranja ostatka programskog koda vodeći računa o duljini vremena izvršavanja korisničkog koda i korištenjem asinkronih poziva (korištenje *callback* funkcija). Programski model stoga BitCloud je detaljnije opisan u poglavlju 4.1.2. S aplikacijskog sloja korisnik ima pristup samo ZigBee slojevima APS i ZDO.

4.1.1.2 Upravitelj zadacima

Upravitelj zadacima (engl. *task manager)* je dio stoga BitCloud koji upravlja vremenskim rasporedom izvršavanja pojedinih dijelova programskog koda. Naime, programski kod se izvršava na jednojezgrenom mikrokontroleru s ograničenom količinom memorije. U smislu izvršavanja programa, svaki sloj ZigBee/BitCloud stoga i programski kod korisničke aplikacije se može smatrati zasebnim programskim procesom. Budući da nije moguće osigurati paralelizam u izvršavanju programskog koda, a potrebno je osigurati pravovremeno izvršavanje koda koji pripada različitim slojevima stoga, koristi se upravitelj zadacima. On određuje prioritete pojedinih ZigBee slojeva. Na osnovi toga tijekom izvršavanja raspodjeljuje procesorsko vrijeme između procesa na način da se u jednom trenutku izvršava samo jedan proces (engl. *single-process*). Upravitelj zadacima bit će opisan detaljnije u poglavlju 4.1.2.3.

4.1.1.3 Programsko upravljanje potrošnjom

Sustav za upravljanje potrošnjom (engl. *power manager*) služi za isključivanje pojedinih dijelova programskog stoga i spremanje potrebnih varijabli prilikom odlaska u stanje smanjene potrošnje (engl. *sleep state*) da bi se nakon povratka u aktivno stanje mogao nesmetano i bez gubitka podataka nastaviti programski tok.

4.1.1.4 Podržano sklopovlje

Stog BitCloud namijenjen je za Atmelove ZigBee module. Za svaku vrstu modula predviđena je posebna inačica stoga, premda bi različiti uređaji trebali imati mogućnost međusobne komunikacije (pod uvjetom da rade u istom frekvencijskom području). Podržani su sljedeći moduli:

- 2,4 GHz AVRRAVEN i RZUSBSTICK
- 2,4 GHz ZigBit i ZigBit Amp
- 900 MHz ZigBit 900

Za potrebe ovog rada korištena je platforma ZigBit Amp. Radi se o ZigBee modulu na 2,4 GHz-a koji na jednoj tiskanoj pločici objedinjuje mikrokontroler Atmel ATmega1281 i komunikacijski sklop AT86RF230. O pristupu periferiji mikrokontrolera i vanjskim uređajima brinu sučelja HAL i BSP.

HAL (engl. *hardware abstraction layer*) je sloj BitClouda preko kojeg se pristupa periferiji mikrokontrolera (sklopovima unutar ZigBit modula) – EEPROM memoriji, aplikacijskim brojilima, *Watchdog* brojilima, sklopovlju za postavljanje u stanje smanjene potrošnje, USART-u, SPI sučelju, sučelju za rad s prekidima itd. U okviru BitClouda programer navedenom sklopovlju pristupa s aplikacijskog sloja preko sučelja visoke razine apstrakcije, bez potrebe za poznavanjem detalja sklopovske realizacije uređaja kojima pristupa.

Funkcije HAL-a su izvedene slijedeći standardnu, događajima upravljanu programsku paradigmu stoga BitCloud. Tipično na aplikacijskom sloju postoji funkcija za inicijalizaciju sklopovlja u kojoj se registrira i korisnička *callback* funkcija. Sklopovlju se pristupa postavljanjem asinkronog zahtjeva kojem se prosljeđuje pokazivač na *callback* funkciju. Zahtjev se prosljeđuje s aplikacijskog sloja HAL-u koji izravno komunicira sa sklopovljem postavljajući odgovarajuće registre mikrokontrolera. Kad sklopovlje obavi traženu operaciju (npr. čitanje ili pisanje na neko sučelje), poziva se *callback* funkcija koja obavještava korisnika da je operacija završila. Na taj način je omogućen neometani nastavak izvršavanja glavnog programa nakon poziva procesa čije je trajanje neodređeno sve do njegovog završetka.

BSP je kratica za *board support package*. Radi se o sloju iste razine i funkcionalnosti kao što je HAL. Ovaj sloj služi kao veza sa sklopovljem izvan ZigBit modula (senzori, GPIO linije,

tipke, LED diode itd.). BSP radi na način analogan HAL sloju. BSP koji dolazi s originalnom verzijom BitClouda sadrži programsko sučelje za pristup sklopovlju na razvojnoj pločici MeshBean.

Programski kod slojeva BSP i HAL dostupan je korisniku. To omogućava pisanje upravljačkih funkcija/sučelja (engl. *driver*) za sklopove koje korisnik poželi izvana spojiti na ZigBit modul. Ukoliko se kod pisanja programske podrške za vanjske jedinice (engl. *device drivers*) slijede programska načela i struktura postojećih BSP funkcija, moguće je osigurati jednostavnu nadogradivost postojećeg koda novom verzijom BitClouda bez izmjena na vlastitim BSP funkcijama.

4.1.2 Programski model stoga BitCloud

4.1.2.1 Događajima upravljano programiranje

Aplikacijsko programsko sučelje (API) programskog stoga BitCloud izvedeno je koristeći događajima upravljan programski model (engl. *event driven programming*). To znači da se programski kod nakon početka izvođenja (ulazna točka) ne izvodi slijedno liniju po liniju uz povremena grananja.

Kod događajima upravljanog modela, i dalje postoji ulazna točka od koje program započinje s izvršavanjem. Nakon početka, program ulazi u beskonačnu petlju i u njoj ostaje do završetka programa. Iterativno prolazeći kroz beskonačnu petlju program čeka da se dogodi neki „događaj“ (engl. *event*). Kod ugradbenih računalnih sustava tipični događaj je zahtjev za prekid kojeg postavlja neki periferni sklop ili vanjski uređaj kao rezultat dovršetka neke operacije (npr. dovršenje čitanja sa serijske sabirnice) ili kao rezultat vanjskog podražaja (promjena stanja na ulaznoj GPIO liniji). Svakom događaju pridružena je funkcija u kojoj se događaj obrađuje (engl. *event handler*). Ova funkcija poziva se automatski kad se u glavnoj petlji detektira pojava događaja. Nakon obrade događaja, glavna petlja nastavlja s izvođenjem.

Događaj se ne mora dogoditi „sam od sebe“, već može biti posljedica korisničkog zahtjeva u programskom kodu. To je čest slučaj kod ugradbenih sustava. Naime, kod ugradbenih sustava potrebno je osigurati neblokirajući pristup sklopovlju. „Neblokirajući“ se odnosi na činjenicu da nije dozvoljeno da neki proces svojim dugotrajnim izvršavanjem onemogući izvršavanje nekog drugog procesa (npr. uzastopno provjeravanje GPIO ulaza u

beskonačnoj petlji spriječit će mikrokontroler da radi išta drugo). Ovakav zahtjev kod kojeg se kontrola glavnom programu vraća tek nakon što je zahtjev opslužen, naziva se sinkronim zahtjevom. Sinkroni zahtjevi se mogu koristiti kad je vrijeme opsluživanja zahtjeva kratko i unaprijed determinirano.

S druge strane, postoje asinkroni zahtjevi. Kod asinkronog zahtjeva, ideja je inicirati neku radnju postavljanjem zahtjeva i nastaviti izvršavanje programa bez čekanja da se obrada zahtjeva dovrši. Kad zahtjev bude obrađen, u glavnom programu se automatski poziva callback funkcija čija je uloga obavještavanje da je zahtjev obrađen. Ova se funkcija može iskoristiti za obradu događaja (engl. *event handler*). Asinkroni zahtjevi su pogodni kod radnji čije je vremensko trajanje dugačko i vremenski nedeterminirano.

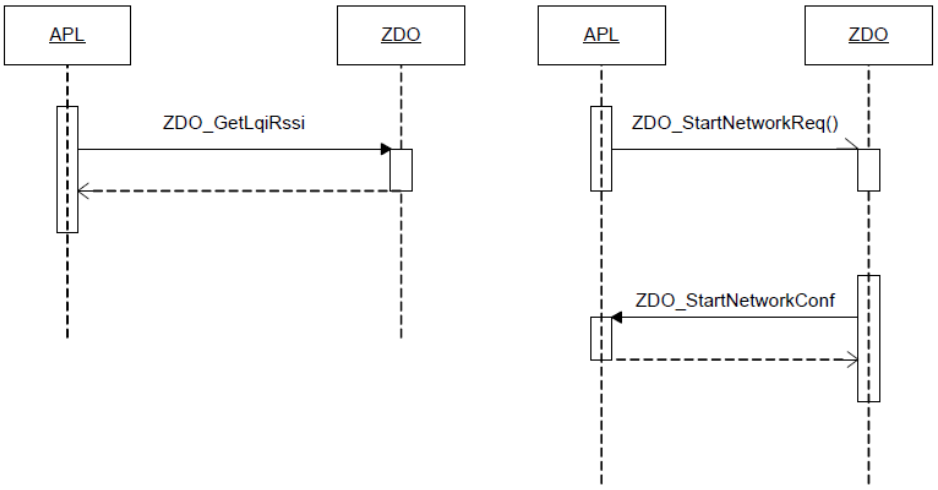
Callback funkcija se realizira tako da se funkciji koja postavlja zahtjev za opsluživanjem kao argument proslijedi pokazivač koji pokazuje na mjesto u memoriji gdje se nalazi *callback* funkcija. Taj pokazivač se prosljeđuje kroz sve funkcije koje funkcija-zahtjev poziva, sve do mjesta gdje se odvija konkretna radnja (npr. fizičko pisanje na sabirnicu). Dovršetak radnje (npr. dovršetak pisanja na sabirnicu) će izazvati poziv *callback* funkcije. Bitno je primijetiti da *callback* funkciju poziva sloj koji je obradio zahtjev (obično sloj niske razine) pa se nakon izvršenja *callback* funkcije kontrola vraća sloju pozivatelju *callbacka* – sloju niske razine.

Kao što je vidljivo iz gornjeg primjera i iz poglavlja 2.2. (pristup sklopovlju preko HAL-a), mehanizam *callbacka* je vrlo pogodan da se slanjem zahtjeva sa sloja visoke razine inicira radnja niske razine (radnja na najnižem sloju) i nakon dovršenja radnje da se obavijesti sloj visoke razine koji je inicirao radnju da je radnja izvršena. Stoga se mehanizam *callbacka* može upotrijebiti i za komunikaciju između slojeva Zigee protokola.

U BitCloudu korisnik ima pristup samo aplikacijskom sloju na kojem piše svoju korisničku aplikaciju. Moguća je posredna komunikacija s ZigBee slojevima APS (funkcije s prefiksom *APS_*) i ZDO (funkcije s prefiksom *ZDO_*). Sva interakcija korisnika prema ZigBee stogu odvija se na način da korisnik s aplikacijskog sloja postavlja zahtjev za uslugom pozivom odgovarajuće funkcije sa sufiksom *Req* (npr. funkcija *ZDO_StartNetworkReq* - zahtjev uspostavom mreže). Funkcija zahtjeva (engl. *request*) obično kao argument prima strukturu čije je jedno od polja pokazivač na *callback* funkciju. Nakon toga kontrolu nad programom preuzima sloj niže razine. Nakon dovršetka obrade zahtjeva poziva se *callback* čime se pozivatelj zahtjeva za uslugom obavještava o završetku usluge. Osim korisnički

definiranih, postoje i *callback* funkcije predefinirane od strane stoga BitCloud. To su funkcije potvrde koje završavaju sufiksom *Conf* (npr. ZDO_StartNetworkConf – obavijest o uspješnosti zahtjeva za uspostavom mreže).

Osim funkcija zahtjeva (*…Req*) i potvrde (*…Conf*), u BitCloudu postoje i funkcije indikacije. Pozivaju se kad se dogodi neki događaj koji nije posljedica korisničkog zahtjeva, već dolazi „izvana“. To su funkcije koje završavaju sufiksom *Ind* – npr. funkcija *APS_DataInd* koja se poziva na aplikacijskom sloju kad ZigBit modul primi podatke. Grupiranje BitCloudovih funkcija na funkcije zahtjeva, *callback* funkcije i funkcije indikacije postignuto je da programski model stoga BitCloud bude potpuno događajima upravljan.



Slika 25: Komunikacija između ZigBee slojeva. Obrada sinkronog zahtjeva (lijevo) i asinkronog zahtjeva (desno) (Atmel, 2009.)

Primjeri komunikacije između slojeva ZigBee protokola u BitCloudu prikazani su slikom 28. Primjer kratkog, vremenski determiniranog, sinkronog, blokirajućeg zahtjeva prikazan je na slici 25, lijevo (čitanje vrijednosti RSSI). Na istoj slici desno je zahtjev neodređenog vremenskog trajanja koji je zato izveden kao, neblokirajući, asinkroni zahtjev korištenjem *callback* funkcije.

4.1.2.2 Upravljanje zadacima, istodobnost izvršavanja, prioriteti

Za ZigBee aplikaciju je karakteristično da postoje dva dijela programskog koda koji se moraju „istovremeno“ izvršavati na istom mikrokontroleru: aplikacijski kod koji sadrži korisničku aplikaciju koja obavlja koristan posao, i programski koda ZigBee stoga koji je potreban za bežičnu komunikaciju. Dakle, potrebno je imati mehanizam koji na

zadovoljavajući način dijeli sistemske resurse (procesorsko vrijeme) između korisničke aplikacije i ZigBee stoga. Podrazumijeva se da postoji samo jedna korisnička aplikacija (zadaća).

4.1.2.3 Upravitelj zadacima – kooperativni višezadaćni sustav

Mehanizam za upravljanje procesorskim vremenom se zove BitCloud *taskmanager*. Kod BitClouda ne postoji klasično cjepkanje procesorskog vremena (engl. *timeslicing*) karakteristično za višezadaćne sustave – simuliranje paralelnog izvršavanja više procesa na način da se procesorsko vrijeme neprestano multipleksira između svih procesa.

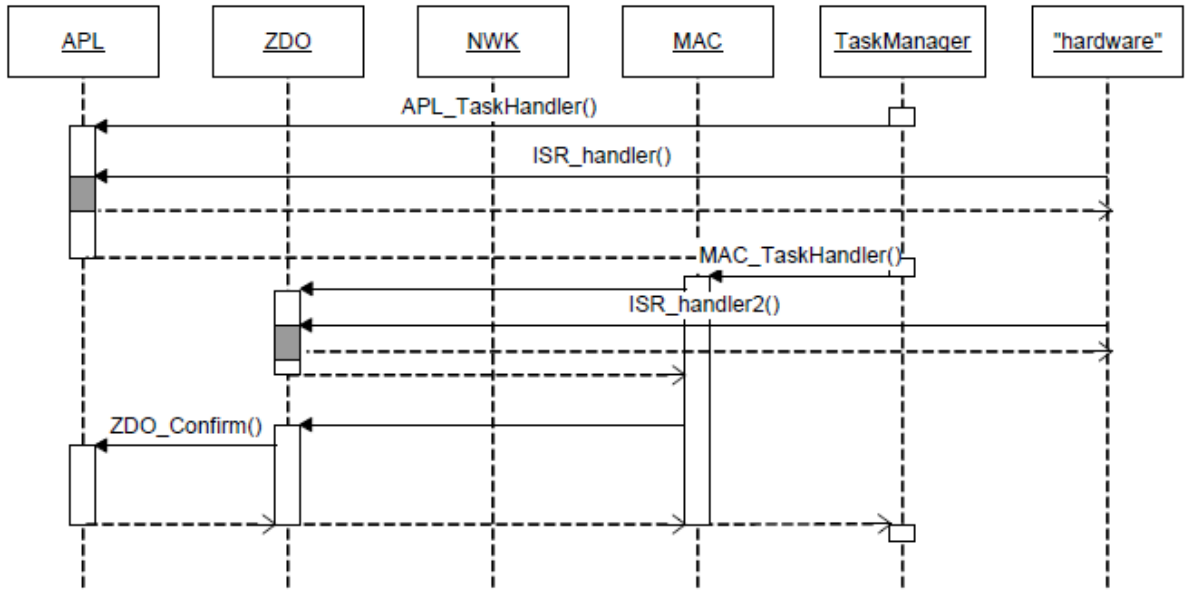
BitCloud predstavlja kooperativni višezadaćni sustav. To znači da neka zadaća/proces (u ovom slučaju ZigBee sloj) toliko dugo ima kontrolu, dok ju svojevoljno ne ponudi nekoj drugoj zadaći/procesu/sloju. Dakle, kod kooperativnih višezadaćnih sustava svi procesi moraju surađivati da bi ostali procesi/zadaće imali pristup procesorskom vremenu. Kooperativni višezadaćni sustavi pogodni su za sustave s malim brojem procesa. Ne zahtjevaju mnogo podatkovne memorije za spremanje konteksta prilikom prenošenja kontrole između zadaća, i izvedba upravitelja zadacima (engl. *taskmanager*) je jednostavnija od višezadaćnih sustava koji koriste vremensko cjepkanje procesorskog vremena. Time su pogodniji za primjenu kod ugradbenih računalnih sustava.

Važnost suradnje slojeva može se vidjeti na sljedećem primjeru: *callback* funkcija koja se poziva iz nižeg sloja obavještava viši sloj (recimo aplikacijski sloj) o dovršenosti nekog procesa. Callback funkcija nasljeđuje prioritet onog sloja iz kojeg je pozvana (niži sloj, obično sloj visokog prioriteta). Ukoliko izvršavanje koda same *callback* funkcije predugo traje, ostatak ZigBee stoga će biti blokiran toliko dugo dok *callback* funkcija ne završi s izvođenjem. Zato je dobra praksa ograničenje duljine izvršanja koda u *callback* funkcijama. U BitCloudu, korisničke *callback* funkcije ne bi smjele trajati dulje od 10 ms. U slučaju potrebe za izvršavanjem dulje sekvence koda u *callback* funkciji, pribjegava se korištenju tehnike odgađanja izvršavanja, koja će biti opisana u poglavlju 4.2.2.

BiCloud *taskmanager* je središnja komponenta u izvršavanju BitCloud aplikacije. Unatoč tome, on djeluje nevidljivo za krajnjeg korisnika iz razloga što krajnji korisnik jedino ima izravni pristup aplikacijskom sloju, tj. većina koda kojeg korisnik piše smješten je u tzv. aplikacijsku *taskhandler* funkciju koja igra ulogu polazne točke na aplikacijskom sloju i bit će detaljnije opisana u poglavlju 4.2.2. *Taskhandler* funkcije postoje i na ostalim ZigBee

slojevima (ZDO, NWK i MAC), no njihov rad je skriven od korisnika. Korisniku so još dostupni taskhandleri slojeva BSP i HAL, no oni se koriste samo kod razvoja programske podrške za vlastite BSP/HAL uređaje.

Dakle, izvršavanje programa započinje od BitCloud *taskmanagera*. On prosljeđuje izvršavanje aplikacijskom sloju (aplikacijskom *taskhandleru*). Kad aplikacijski *taskhandler* završi, kontrola se vraća BitCloud taskmanageru koji predaje vlast sloju (*taskhandleru* onog sloja) koji je zatražio kontrolu, a ima najviši prioritet. Primjer programskog toka prema opisanog pravilima dan je slikom 26.



Slika 26: primjer programskog toka (Atmel, 2009.)

4.1.2.4 Aplikacijski upravitelj zadacima

Glavni dio programskog koda korisničke aplikacije izvodi se u funkciji *APL_TaskHandler*. Prototip dane funkcije glasi

```
void APL_TaskHandler(void)
```

Ovo je polazna točka korisničkog programskog koda na aplikacijskom sloju svake BitCloud aplikacije. To je funkcija u kojoj se izvodi ili iz koje se poziva glavnina korisničkog programskog koda.

Korisnička *taskhandler* funkcija prvi put se poziva automatski prilikom pokretanja aplikacije. Može se pozvati ručno koristeći poziv

```
SYS_PostTask(APL_TASK_ID);
```

Gornjim pozivom BitCloudovom upravitelju zadacima (engl. *taskmanageru*) daje se zahtjev da preusmjeri programski tok (i procesorsko vrijeme) na funkciju APL_TaskHandler (odnosno na aplikacijski sloj). Taj poziv često se koristi za tehniku odgode izvršavanja. Naime, aplikacijski *taskhandler* ima najniži prioritet. Posljedično, kod na aplikacijskom sloju izvršavat će se tek kad su svi ostali slojevi opsluženi i na aplikacijskom sloju (u aplikacijskom *taskhandleru*) programski kod se smije najdulje izvršavati (do 50 ms u komadu). To se često koristi prilikom izvršavanja *callback* funkcija. Podsjetimo, *callback* funkcija ima prioritet sloja koji ju poziva (viši prioritet). Time je vrijeme koje *callback* smije blokirati ostatak ZigBee stoga samo 10 ms. Kako *callback* funkcije predstavljaju događaj koji će potaknuti izvršenje nekog dužeg dijela programskog koda, navedeni kod nije moguće smjestiti unutar same *callback* funkcije. Unutar *callback* funkcije, postavljanjem odgovarajućih korisnički definiranih zastavica i pozivanjem funkcije *SYS_PostTask* unutar *callback* funkcije, može se dio programskog koda iz *callback* funkcije preseliti u aplikacijski *taskhandler* (na aplikacijski sloj). Tamo se dobiva dodatnih 50 ms za izvršavanje, no kod će se izvršiti tek kad *callback* funkcija završi i aplikacijski sloj dođe na red.

Konačno, ovo poglavlje se može sažeti u nekoliko pravila:

1. Korisnička aplikacija organizirana je pomoću *callback* funkcija koje niži slojevi pozivaju na temelju izvršenja zahtjeva poslanog od strane višeg (aplikacijskog sloja).
2. Korisnik samostalno definira koji ga događaji zanimaju definirajući adrese *callback* funkcije proilikom postavljanja zahtjeva.
3. Sve korisničke *callback* funkcije se moraju izvršiti u <10 ms.
4. Prioritet korisničke *callback* funkcije, mada joj se programski kod nalazi u aplikacijskom programskom kodu se naslijeđuje od sloja iz kojeg *callback* biva pozivan (prioritet *callback* funkcije je obično viši od prioriteta aplikacijskog sloja).

5. Aplikacijski upravitelj zadacima (programski kod na aplikacijskom sloju) ima najniži prioritet se izvršava tek kad se izvrši kod sa slojeva višeg prioriteta (niži ZigBee slojevi).
6. Programski kod funkcije aplikacijskog upravitelja zadacima se mora izvršiti u manje od 50 ms (ne smije blokirati ZigBee stog).

4.1.3 Tipična struktura BitCloud aplikacije

U BitCloud aplikaciji funkcija *main* nigdje ne postoji jer je sadržana u stogu BitCloud i njezin je poziv nevidljiv korisniku. Za korisnika programski tok počinje funkcijom *APL_TaskHandler*. Kao što je objašnjeno u poglavlju 4.2.2., to je aplikacijski *taskhandler*. U toj funkciji (i funkcijama koje se iz nje pozivaju) je sadržana većina programskog koda. Ostatak programskog koda nalazi se u *callback* funkcijama. *Callback* funkcije mogu biti funkcije potvrde (engl. *confirmation callback*) koje se pokreću kao odgovor na postavljni asinkroni zahtjev ili funkcije inidikacije (engl. *indication callback*), koje se izvršavaju na neki vanjski „podražaj“.

Kao što je već spomenuto, trajanje izvršavanja koda u *callback* funkcijama ne smije prelaziti 10 ms pa se najčešće mora koristiti tehnika odgađanja izvršenja prebacivanjem koda iz *callback* funkcija u aplikacijski *taskhandler*. To logično vodi do zaključka da je BitCloud aplikaciju najbolje organizirati kao stroj s konačnim brojem stanja. Pritom *callback* funkcije služe samo za promjenu stanja i pozivanje aplikacijskog *taskhandlera*. U tom slučaju aplikacijski *taskhandler* je centralna funkcija u kojoj se u *switch-case* strukturi provjerava u kojem se stanju nalazi aplikacija i prema tome poziva odgovarajuća funkcija. Za čuvanje aplikacijskog stanja potrebno je definirati posebnu globalnu varijablu enumeracijskog tipa.

U nastavku je dan odsječak koda iz kojeg je vidljivo kak izgleda temelja BitCloud aplikacija. Vidljivo je definirana globalna varijabla *appState* koja sadrži stanje aplikacijskog stroja s konačnim brojem stanja. Nakon toga je realiziran aplikacijski upravitelj zadacima (engl. *task handler*) u kojem se tok u *switch-case* strukturi preusmjerava ovisno o aplikacijskom stanju. Nakon aplikacijskog upravitelja zadacima dani su primjeri dviju *callback* funkcija potvrde. Na kraju su dane dvije callback funkcije indikacije. Vidljivo je da se u i funkcijama samo mijenja aplikacijsko stanje i poziva *APL_TaskHandler* gdje će se obaviti koristan posao.

```
/******************************
#include directives
******************************/
...
#include <taskManager.h>
#include <zdo.h>
#include <configServer.h>
#include <aps.h>
/******************************
Prototipovi funkcija
******************************/
...
/******************************
Globalne varijable
******************************/
AppState_t appState = APP_INITING_STATE;
...

/******************************
Aplikacijski taskhandler
******************************/
void APL_TaskHandler()
{
	switch (appState)
	{
		case APP_IN_NETWORK_STATE:
			...
			break;
		case APP_INITING_STATE: //node has initial state
			...
			break;
		case APP_STARTING_NETWORK_STATE:
			...
			break;
	}
}
/******************************
Callback funkcije potvrde
******************************/
static void ZDO_StartNetworkConf(ZDO_StartNetworkConf_t *confirmInfo)
{
	...
	if (ZDO_SUCCESS_STATUS == confirmInfo->status)
	{
		appState = APP_IN_NETWORK_STATE;
		...
		SYS_PostTask(APL_TASK_ID);
	}
}
void ZDO_LeaveResp(ZDP_ResponseData_t *zdpRsp)
{
	...
}
/******************************
Callback funkcije indikacije
******************************/
void ZDO_WakeUp_Ind(void)
{
	...
	if (APP_IN_NETWORK_STATE == appState)
	{
		appState = APP_STARTING_NETWORK_STATE;
		...
		SYS_PostTask(APL_TASK_ID);
	}
}
```

4.1.4 Sučelje ConfigServer

U okviru sotga BitCloud postoji posebno sučelje preko kojeg se namještaju mnogi parametri funkcioniranja bežične mreže i pojedinačnih čvorova. Ovo sučelje naziva se *Config Server Interface*. Parametri koji se njime namiještaju prepoznaju se po prefiksu *CS_*. Postoje dvije vrste CS parametara:

- Trajni (engl. *persistent*) – spremaju se u EEPROM memoriju mikrokontrolera i ostaju nakon reseta ZigBit modula
- Privremeni (engl. *nonpersistent*) – spremaju se u RAM pa se gube se nakon gašenja modula

Najčešće korišteni CS parametri su:

- CS_EXT_PANID – PAN ID adresa mreže
- CS_CHANNEL_MASK – postavljanje filtra kanala
- CS_END_DEVICE_SLEEP_PERIOD - duljina perioda spavanja krajnjeg čvora
- CS_NETWORK_KEY – enkripcijski ključ mreže
- CS_RF_TX_POWER – izlazna snaga odašiljača
- CS_DEVICE_TYPE – uloga čvora (krajnji čvor/koordinator/usmjernik)

Tip svakog parametra (privremeni/trajni) dokumentiran je u datoteci izvornog koda *configServer.h*. Parametri *Config Servera* se mogu mijenjati na dva načina - statički u Makefile datoteci prije prevođenja ili dinamički u programskom kodu:

Statičko namještanje CS parametara:

```
CSFLAGS += -D CS_RF_TX_POWER = 3
```

Dinamički se CS parametri čitaju i pišu pomoću sljedećih funkcija:

```
CS_WriteParameter(CS_RF_TX_POWER_ID, &new_txPwr);
CS_ReadParameter(CS_RF_TX_POWER_ID, &curr_txPwr);
```

4.1.5 Mrežna komunikacija

4.1.5.1 Uspostava mreže

ZigBee mrežu uspostavlja ZigBee koordinator. Uloge čvorova postavljaju se CS parametrom CS_DEVICE_TYPE.

Prije uspostave mreže potrebno je podesiti sljedeće CS parametre:

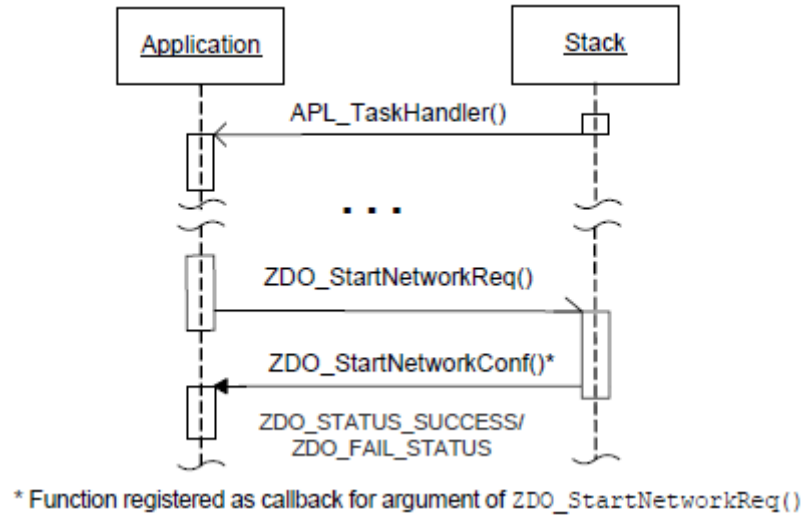
- CS_CHANNEL_MASK – izbor kanala

- CS_EXT_PAN_ID – PAN ID adresa mreže

Parametar CS_CHANNEL_PAGE služi za izbor modulacijskog postupka, no za frekvencijsko područje 2,4 GHz se zanemaruje jer se koristi samo OQPSK (engl. *offset quadrature phase-shift keying*) modulacija.

Sam proces uspostave mreže odvija se pomoću standardnog programskog sučelja. Iz aplikacijskog upravitelja zadacima (funkcije *APL_TaskHandler*) postavlja se zahtjev za uspostavom mreže pomoću funkcije *ZDO_StartNetworkReq*. Nakon uspostave mreže, dobiva se callback potvrde *ZDO_StartNetworkConf*.

Poruka da se u mrežu spojio novi čvor, čvor roditelj (npr. koordinator) dobiva preko *callback* funkcije *ZDO_MgmtNwkUpdateNotf*. Slika 27 prikazuje proces uspostave ZigBee mreže.



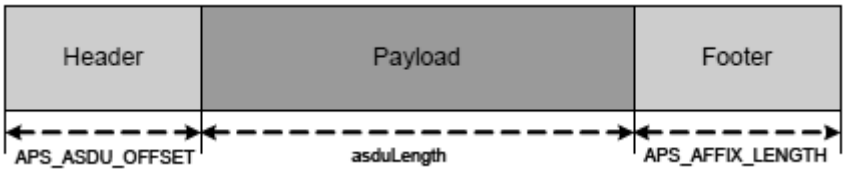
Slika 27: Proces uspostave ZigBee mreže (Atmel, 2009.)

4.1.5.2 Slanje poruka

Preduvjet za slanje poruka je da je definiran barem jedan *endpoint*. *Endpoint* je ZigBee primitiv. Korisnički *endpoint* definiran je konstantom iz skupa 1 do 240, dok je *endpoint* 0 rezerviran za poruke sloja ZDO. *Endpoint* je opisan tzv. *endpoint descriptorom*. To je struktura tipa *SimpleDescriptor_t*. Njome se definira konstanta pridružena *endpointu*, pripadnost ZigBee profilu te broj i popis ulaznih i izlaznih ZigBee *clustera*. *Endpoint* descriptor polje je strukture *APS_RegisterEndpointReq_t*. Ova struktura proslijeđuje se

kao argument funkciji *APS_ReqisterEndPoint* koja registrira endpoint. Nakon registracije *endpointa* moguće je slanje poruka.

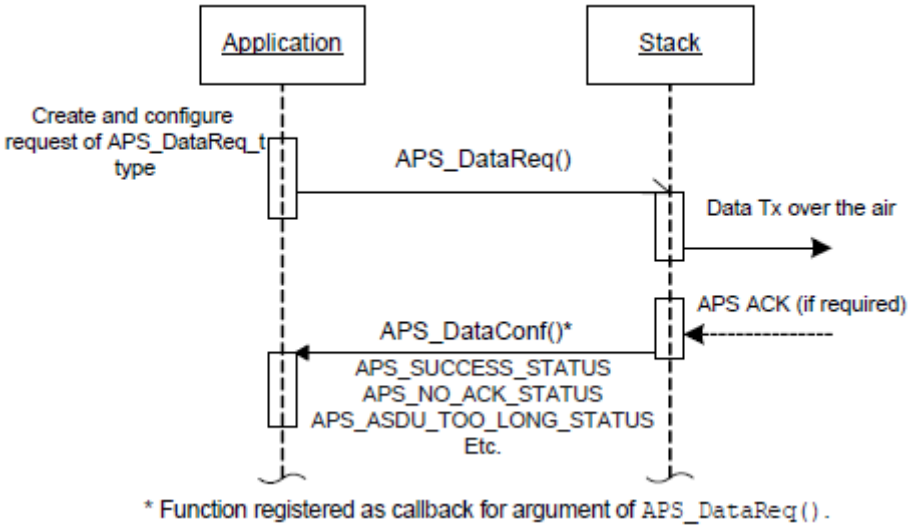
Na aplikacijskom sloju, slanje poruka se inicira pozivom funkcije *APS_DataReq()*. Ova funkcija kao argument prima strukturu tipa *APS_DataReq_t*. Ovom strukturom definirana je pripadnost pripadnost ZigBee profilu, postoji li potvrda prijenosa, koja se funkcija koristi kao *callback* za potvrdu prijenosa te pokazivač na programsku strukturu koja se koristi za prijenos korisnički definiranih podataka (tzv. *application message buffer*). Slika 28 prikazuje izgled spremnika korištenog za prenošenje korisničkih poruka:



Slika 28: Spremnik za prenošenje korisničkih poruka (Atmel, 2009.)

Najveća duljina spremnika za prenošenje korisničkih poruka (zajedno s *poljima* header i *footer*) je 84 bajta. Ukoliko se uključi enkripcija podataka, moguće je prenositi najviše 53 bajta.

Slikom 29 prikazan je proces slanja podataka od iniciranja zahtjeva funkcijom *APS_DataReq* do primitka potvrde prijenosa preko registrirane *callback* funkcije *APS_DataConf* :



Slika 29: Slanje ZigBee paketa (Atmel, 2009.)

Ako se podaci šalju pojedinačnim čvorovima, potrebno je prije slanja polje *.dstAddress.shortAddress* strukture tipa *APS_DataReq_t* popuniti kratkom, 16-bitnom adresom čvora odredišta. Za slanje svim čvorovima u mreži navedeno polje popunjava se konstantom BROADCAST_ADDR_ALL (ili numeričkom konstantom 0xFFFF). Kod *broadcast* prijenosa potrebno je isključiti potvrdu prijenosa postavljanjem .txOptions.acknowledgedTransmission u nulu.

4.1.5.3 Primanje poruka

Da bi se mogle primati poruke na aplikacijskom sloju, potrebno je definirati strukturom tipa APS_RegisterEndpointReq_t definirati barem jedan ZigBee *endpoint*. Polje *.APS_DataInd* definira adresu inidikacijske *callback* funkcije koja će se pozvati nakon što ZigBee čvor primi poruku te se utvrdi da je poruka namijenjena registriranom *endpointu*. *Callback* funkcija kao parametar prima strukturu tipa *APS_DataInd_t*. Najkorisniji parametri koji se mogu iščitati iz ove strukture su:

- uint8_t* asdu – pokazivač na spremnik (prikazan slikom 4) koji sadrži korisničke podatke primljene preko mreže
- APS_Address_t srcAddress - mrežna adresa čvora pošiljatelja
- Endpoint_t srcEndpoint - ZigBee *endpoint* pošiljatelja
- ClusterId_t clusterId – ZigBee *cluster* kojem pripada poslana poruka

Na temelju posljednja tri parametra moguće je razlikovanje pristiglih poruka unutar indikacijske *callback* funkcije.

4.1.6 Upravljanje potrošnjom

U BitCloudu je moguće upravljati potrošnjom krajnjih čvorova ZigBee mreže. Pretpostavlja se da su koordinator i usmjernik (*router*) uvijek uključeni.

Glavna metoda upravljanja potrošnjom je postavljanje krajnjeg čvora u neaktivno stanje (spavanje). U tom stanju mikrokontroler ostaje uključen ali radi u stanju smanjene potrošnje, dok se komunikacijski sklop potpuno gasi.

Krajnji čvor se u stanje spavanja šalje postavljanjem asinkronog zahtjeva ZDO_SleepReq. Nakon što se dobije odgovor preko callback funkcije ZDO_SleepConf, čvor odlazi na spavanje. Postoje dva načina buđenja iz stanja spavanja.

1. Buđenje nakon vremena određenog CS parametrom CS_END_DEVICE_SLEEP_PERIOD. Bitno je napomenuti da se period buđenja ne može mijenjati dinamički tokom rada aplikacije. Prilikom buđenja, čvor se javlja izvršavanjem callback funkcije *ZDO_wakeUpInd*.
2. Buđenje na prekid – svaki registrirani prekid može probuditi krajnji čvor. Pritom jedino valja voditi računa o tome da su prekidi dio HAL sloja te da ostali slojevi BitClouda neće biti obaviješteni o buđenju. Zato je prilikom buđenja na prekid potrebno obavijestiti ostatak stoga da se čvor probudio postavljanjem zahtjeva *ZDO_WakeUpReq*.

Također, moguće je ostaviti mikrokontroler potpuno uključenim i isključiti samo komunikacijski sklop. To može pomoći smanjiti potrošnju u aktivnom stanju. Komunikacijski sklop se zasebno može isključiti postavljanjem CS parametra CS_AUTO_POLL na *false*. To je prilikom rada čvora izvodi funkcijskim pozivom ZDO_StopSyncReq(). Komunikacijski sklop se ponovo uključuje funkcijom ZDO_StartSyncReq().

Potrošnja se može pokušati smanjiti i smanjenjem snage odašiljača. Snaga odašiljača može se namiještati CS parametrom CS_RF_TX_POWER ili API pozivom ZDO_SetTxPowerReq().

4.2 Programska podrška za senzore –HAL i Maslinet BSP

Programski stog BitCloud koristi programski model u kojem se pristup ugrađenoj periferiji mikrokontrolera (brojila, A/D pretvornik, serijska komunikacijska sučelja poput I^2C, SPI, USART…) i vanjskim jedinicama (senzori) apstrahira na način da na najvišem sloju - sloju korisničke aplikacije programer pristupa svim uređajima na uniformni način. Tipično je to preko jednostvnih asinkronih funkcijskih poziva kojim se inicira pristup uređaju. Nakon dovršetka operacije periferna/vanjska jedinica poziva *callback* funkciju u kojoj korisničkoj aplikaciji dojavlja rezultat operacije. Ovakav neblokirajući model omogućava optimalno iskorištenje procesorskog vremena (a time i uštedu energije). Također, na HAL i BSP slojevima postoje upravitelji zadacima (engl. *taskhandler*) po funkcionalnosti analogni aplikacijskom upravitelju zadacima.

4.2.1 HAL

HAL (engl. *hardware abstraction layer*) je sučelje za pristup unutarnjim perifernim jedinicama ZigBit modula. Upravitelj zadacima na HAL sloju definiran je u datoteci *BitCloud**Components**HAL**avr**common**src**halTaskManager.c*. HAL se prema skripti iz datoteke *BitCloud**Components**HAL**Makefile* prevodi u statičke biblioteke smještene u mapi *BitCloud**lib*. Navedene statičke biblioteke uključuju se prilikom prevođenja korisničke aplikacije.

U okviru projekta Maslinet postojećim HAL sučeljima dodano je sučelje za upravljanje prekidima. Definirano je u datotekama *BitCloud**Components**HAL**include**maslinetIrq.h* i *BitCloud**Components**HAL**avr**common**src**maslinetIrq.c*. Implementira funkcije za pridjeljivanje prekidnih funkcija na promjene razina na pojedinim GPIO pinovima ZigBit modula. GPIO prekidi su mogući jedino na pinovima ZigBita 4, 37, 38, 42 i 43. Funkcije su opisane u tablici 12:

Tablica 12: Programsko sučelje za rad s GPIO prekidima

funkcija	**opis**
`uint8_t MHAL_RegisterIrq(MHAL_IrqNumber_e, MHAL_IrqMode_e irqMode, MHAL_IrqCallback_t)`	Registrira prekid na pinu MHAL_IrqNumber_e (ZigBit pinovi broj 4, 37, 38, 42 i 43), na željeni brid i definira prekidnu funkciju.
`uint8_t MHAL_EnableIrq(MHAL_IrqNumber_e)`	Omogućuje prekid na pinu

	MHAL_IrqNumber_e. Prekid se ne može dogoditi prije nego se uključi ovom funkcijom.
`uint8_t MHAL_DisableIrq(MHAL_IrqNumber_e)`	Onemogućuje postavljanje prekida na pinu MHAL_IrqNumber_e.
`uint8_t MHAL_UnregisterIrq(MHAL_IrqNumber_e)`	Uklanja prekid na pinu MHAL_IrqNumber_e (omogućava redefiniciju prekidne funkcije za neki izvor prekida)

4.2.2 Maslinet BSP

BSP (engl. *board support package*) je programska podrška za vanjske jedinice (engl. *device drivers*) koje se spajaju na ZigBit modul, a nalaze se ili na tiskanoj pločici osjetilnog čvora ili se spajaju na njega izvana kabelom. Kako ova programska podrška ovisi o sklopovlju koje je dodano modulu ZigBit, za potrebe projekta Maslinet je razvijen Maslinet BSP za sklopovlje osjetilnog čvora FER Čvorak v2.0 i sve senzore koji se spajaju na osjetilni čvor. Programska podrška je potpuno kompatibilna sa sklopovljem osjetilnog čvora FER Čvorak v2.0b opisanog u ovom radu.

Programski kod BSP sloja općenito se nalazi u mapi *BitCloud**Components**BSP* i prevodi prema skripti *BitCloud**Components**BSP**Makefile* prilikom svakog prevođenja korisničke aplikacije. Prevođenje BSP sloja u svakom prevođenju korisničke aplikacije je pogodno zbog toga što se postavljanjem odgovarajući predprocesorskih labela definiranih u datotekama *BitCloud**lib**MakerulesBc_xxx*, koje se uključuju u *Makefile* skriptu korisničke aplikacije navodi koji dijelovi koda BSP-a se moraju uključiti (kojim vanjskim jedinicama pristupa korisnička aplikacija) pa se selektivnim prevođenjem štedi programska memorija.

Programska podrška za pristupanje vanjskim jedinicama korištenim u mreži Maslinet nalazi se u mapi *BitCloud**Components**BSP**MASLINET_ZIGBIT*\. Upravitelj zadacima na BSP sloju zajednički svim vanjskim jedinicama nalazi se u datoteci *BitCloud**Components* *BSP**MASLINET_ZIGBIT**src**bspTaskManager.c*.

Tražena (grupa)vanjskih jedinica se uključuje u korisničku aplikaciju uključivanjem željene datoteke iz mape \BitCloud\Components\BSP\include. Za senzore je potrebno uključiti datoteku \BitCloud\Components\BSP\include maslinetSensors.h. U nastavku slijedi opis razvijenih sučelja za senzore iz navedene datoteke.

4.2.2.1 SHT75

Ovo je sučelje za pristup senzoru temperature i relativne vlažnosti zraka (Sensirion, 2010.) .Senzor dakle može mjeriti dva modaliteta. Komunicira preko dvožičnog sučelja sličnog, no nekompatibilnog s I^2C protokolom. Funkcije za pristup senzoru dane su u tablici 13.

Tablica 13: Sučelje senzora SHT75

funkcija	**opis**
`result_t BSP_OpenTempHumiditySensor(void)`	Uključuje senzor (dovodi mu napajanje VCC_Sens).
`result_t BSP_CloseTempHumiditySensor(void)`	Isključuje senzor.
`result_t BSP_ReadAirTemperatureData(void (*f)(bool result, int16_t data))`	Inicira čitanje temperature. Rezultat vraća preko *callback* funkcije f koja vraća podatak o uspješnosti čitanja result i sam podatak dana.
`result_t BSP_ReadAirHumidityData(void (*f)(bool result, int16_t data))`	Pokreće čitanje vlage zraka. Rezultat vraća preko *callback* funkcije f koja vraća podatak o uspješnosti čitanja result i sam podatak data.

4.2.2.2 LM73

LM73 je dijagnostički senzor temperature na pločici. Radi preko I^2C sučelja (National Semiconductor, 2009.). Programsko sučelje za rad sa senzorom prikazano je u tablici 14.

Tablica 14: Programsko sučelje za pristup senzoru LM73

funkcija	**opis**
`result_t BSP_OpenOnBoardTemperatureSensor (void)`	Uključuje napajanje senzora.
`result_t BSP_CloseOnBoardTemperatureSensor (void)`	Iskjučuje npajanje senzora.
`result_t BSP_ReadOnBoardTemperatureData(void (*f)(bool result, int16_t data)`	Pokreće čitanje temperature. Rezultat vraća preko *callback* funkcije f koja vraća podatak o uspješnosti čitanja result i sam podatak data.

4.2.2.3 ISL29013

ISL29013 je senzor za mjerenje osvjetljenja. Spaja se na I^2C sabirnicu. (Intersil, 2008.) Sučelje za pristup senzoru prikazano je u tablici 15.

Tablica 15: Sučelje za pristup senzoru ISL29013

funkcija	**opis**
`result_t BSP_OpenLightSensor(void)`	Uključuje napajanje senzora.
`result_t BSP_CloseLightSensor(void)`	Isključuje napajanje senzora.
`result_t BSP_ReadLightData(void (*f)(bool result, int16_t data))`	Pokreće čitanje temperature. Rezultat vraća preko *callback* funkcije f koja vraća podatak o uspješnosti čitanja result i sam podatak data.

4.2.2.4 Mjerenje napona akumulatora

Napon akumulatora mjeri se pomoću A/D pretvornika ugrađenog u ZigBit. Napon VBAT se na A/D pretvornik dovodi preko sklopke U6 opisane u poglavlju 3.1.1.4. Funkcije za mjerenje napona akumulatora popisane su u tablici 16.

Tablica 16: Programsko sučelje za mjerenje preostalog napona akumulatora VBAT

funkcija	**opis**
`result_t BSP_OpenBatterySensor(void)`	Uključuje sklopku
`result_t BSP_CloseBatterySensor(void)`	Isključuje napajanje senzora.
`result_t BSP_ReadBatteryData(void (*callback)(uint8_t data))`	Pokreće čitanje napona akumulatora VBAT. Rezultat vraća preko *callback* funkcije f koja vraća podatak data.

4.2.2.5 MS5540B

MS5540B je senzor tlaka zraka. Za komunikaciju koristi 3-žično serijsko sučelje opisano u (Intersema, 2008.), na pinove konektora X1 kako je opisano u poglavlju 2.2.6, tablici 8. U nastavku je dana tablica 17 s korisničkim funkcijama za pristup senzoru.

Tablica 17: Sučelje za pristup senzoru MS5540B

funkcija	opis
`result_t BSP_OpenAirPressureSensor(void)`	Uključuje napajanje senzora.
`result_t BSP_CloseAirPressureSensor(void)`	Isključuje napajanje senzora.
`result_t BSP_ReadAirPressureData(void (*f)(bool success, int16_t data1, int16_t data2, int16_t c1, int16_t c2, int16_t c3, int16_t c4, int16_t c5, int16_t c6))`	Pokreće čitanje tlaka zraka. Rezultat vraća preko *callback* funkcije f koja vraća podatak o uspješnosti čitanja result i podatak u neobrađenom obliku podatka proporcionalnog tlaku (napetosti membrane), temperaturi okoline i 6 koeficijenata c1…c6 koji služe za daljnju obradu kojom se na temelju svih 8 vraćenih vrijednosti dobije podatak o tlaku .

4.2.2.6 RTC

Na pločici osjetilnog čvora koristi se RTC sklop DS1337 (Dallas Semiconductor, 2009.). Sklop komunicira sa modulom ZigBit I^2C sučeljem i prekidnom linijom kojom dojavljuje alarm. Programsko sučelje prikazano je u tablici 18. Omogućuje upisivanje točnog vremena, čitanje trenutnog vremena, postavljanje periodičkog ili jednokratnog brojila te postavljanje alarma u predefinirano vrijeme. Sve funkcije rezultat zahtjeva vraćaju preko pridruženih callback funkcija. Generalno, RTC je prije upotrebe potrebno inicijalizirati. Nakon postavljanja alarma npr. funkcijom AlarmSet() potrebno je uključiti odbrojavanje do alarma funkcijom AlarmEnable().

Tablica 18: Programsko sučelje za rad s RTC sklopom

Funkcija	Opis
`result_t rtc_Init(rtc_callback_f)`	Uključuje, inicijalizira sučelje prema RTC sklopu.
`result_t rtc_TimeSet(rtc_time_t *, rtc_callback_f)`	Postavlja (zadaje) periodički alarm.
`result_t rtc_TimeGet(rtc_time_t *, rtc_callback_f)`	Dohvaća trenutno vrijeme.
`result_t rtc_AlarmSet(rtc_time_t *, rtc_alarm_f, rtc_callback_f)`	Postavlja alarm u predefinirano vrijeme.
`result_t rtc_AlarmGet(rtc_time_t *, rtc_callback_f)`	Dohvaća trenutno postavljeno vrijeme alarma.
`result_t rtc_TimerSet(unsigned short, rtc_timer_type_e, rtc_timer_f, rtc_callback_f)`	Postavlja brojilo.Brojilo može biti periodičk ili jednokratno (definirano parametrom `rtc_timer_type_e`). Kad brojilo odbroji vrijeme u sekundama zadano prvim parametrom, poziva se a `rtc_callback_f`.

`unsigned int rtc_TimerGet();`	Dohvaća stanje brojila.
`result_t rtc_AlarmEnable(rtc_callback_f)`	Pokreće mjerenje vremena do alarma.
`result_t rtc_AlarmDisable(rtc_callback_f)`	Zaustavlja (gasi) alarm.
`result_t rtc_TimerEnable(rtc_callback_f)`	Pokreće odbrojavanje brojila.
`result_t rtc_TimerDisable(rtc_callback_f)`	Zaustavlja odbrojavanje brojila.

4.3 Mrežna aplikacija

U ovom radu korištena je bežična mreža osjetila razvijena za potrebe projekta Maslinet. U trenutnoj fazi razvoja, mreža se sastoji od 3 čvora:

- krajnjeg čvora koji mjeri mikroklimatske parametre zraka – temperaturu, relativnu vlažnost, osvjetljenje i tlak
- krajnjeg osjetilnog čvora s kamerom
- koordinatora koji upravlja mrežom, krajnjim čvorovima proslijeđuje zahtjeve za podacima, agregira primljene podatke i dalje ih prosljeđuje ugradbenom mrežnom serveru/GPRS *gatewayju* (modulu Wavecom)

U ovom radu nisu razmatrani dijelovi programske podrške za akviziciju slike i serijski komunikacijski protokol za prijenos podataka između zigBee koordinatora i modula Wavecom.

4.3.1 Zajednički dio mrežne aplikacije

4.3.1.1 Definicija potrebnih ZigBee primitiva

Cluster

Clusteri odgovaraju "tipu poruke" koji se šalje. *Cluster* je logički kontejner u koji se prima ili iz kojeg se šalje neka vrsta poruka. *Clusteri* su neophodni za razlikovanje koju vrstu poruke je primio pojedini čvor. Svaki *cluster* može biti ulazni ili izlazni. Tako npr. krajnji čvor za slanje slike ima 3 izlazna *clustera*, CLUSTER_IMAGE_HEADER, CLUSTER_IMAGE_DATA i CLUSTER_IMAGE_EOF te 2 ulazna *clustera* CLUSTER_IMAGE_HEADER_REQ i CLUSTER_IMAGE_DATA_REQ jer je za slanje slike potrebno ovih 5 vrsta poruka. Ideja je da oba čvora između kojih se prenosi neka vrsta poruke imaju za tu vrstu poruka definirane iste clustere. Dakle, npr. za prijenos slike bi onda i na koordinatorskom čvoru trebali postojati istih 5 gorenavedenih *clustera*.

Određena vrsta poruke se, dakle, zapravo uvijek šalje između 2 ista *clustera*. Zato se u programskom kodu niti ne govori o izvorišnim i odredišnim *clusterima*, nego se jednostavno navodi jedinstveni *clusterId* za neku vrstu poruke.

Endpoint

Logički gledano, *endpoint* je primitiv za jednu razinu viši od clustera, ali iz ZigBee specifikacije (ZigBee Alliance, 2008.) nije do kraja definirana hijerarhijska veza između *endpointa* i ostalih entiteta (*profile*, *device* i *cluster*). Kod svake transakcije se definira pošiljateljski srcEndpoint i primateljski dstEndpoint. Endpointi su predstavljeni numeričkim identifikatorima 1-240. 0 predstavlja posebnu vrstu *endpointa* koji služi za uspostavljanje i upravljanje ZigBee mrežom. Za ovdje opisanu mrežnu aplikaciju *endpointi* nisu strogo neophodni za razlikovanje vrsta poruka. Svejedno, mogu se iskoristiti pa je dogovoreno da postoje 4 vrste endpointa:

- ENDPOINT_COORDINATOR - sve poruke koje se šalju koordinatoru, šalju se na ovaj i. Sve poruke koje koordinator šalje, šalje s ovog *endpointa*.
- ENDPOINT_CAMERA - ovaj *endpoint* imaju krajnji čvor s kamerom.
- ENDPOINT_SENSOR - imaju ga krajnji čvorovi sa senzorima. Kad se komunicira sa senzorima na nekom krajnjem čvoru, s ovog *endpointa* krajnji čvor šalje podatke odn. na tom *endpointu* prima zahtjev za slanjem podataka.
- ENDPOINT_SERVICE - postoji na svim krajnjim čvorovima. Služi za primanje/slanje servisnih informacija. Trenutno je implementirano samo slanje slanje servisnih podataka - temperature, stanja baterije itd. koordinatoru. Ovaj endpoint predviđen je za implementaciju mrežnog protokola za distribuciju programskog koda prilikom udaljene nadogradnje programske podrške.

Profile

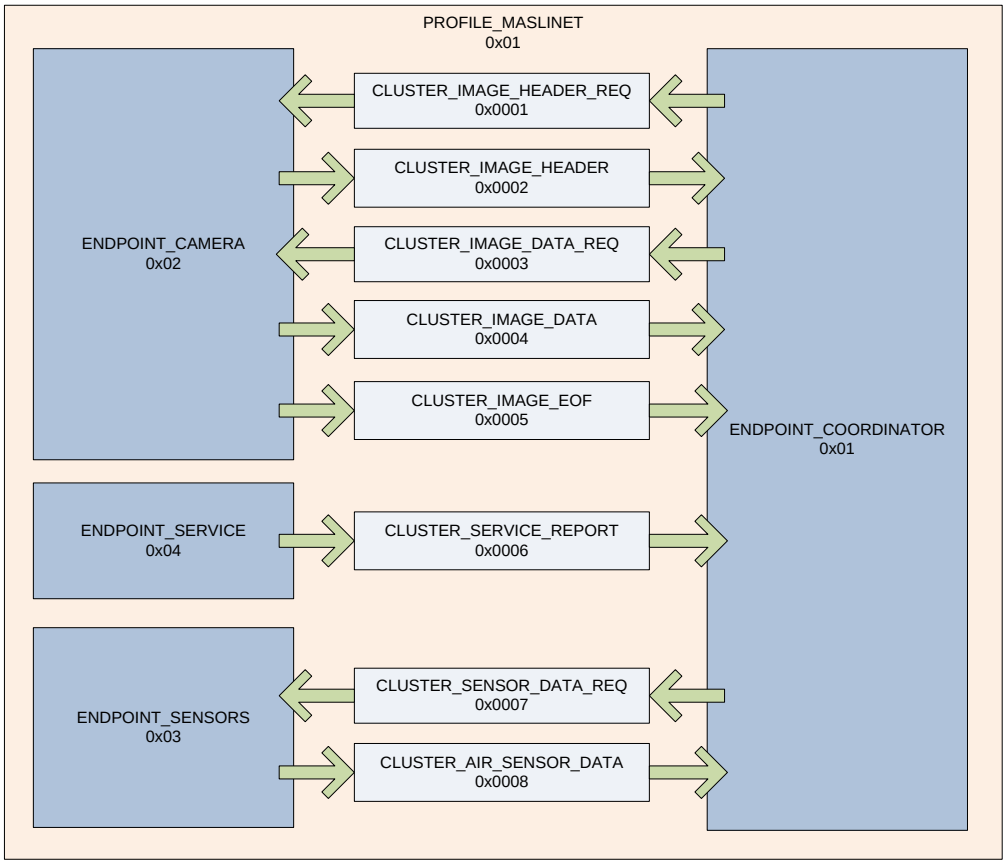
Hijerarhijski najviši ZigBee primitiv. Ovdje je jedino bitno da svi čvorovi unutar mreže imaju isti APP_PROFILE_ID.

Attribute

Hijerarhijski gledano, ovo je primitiv najniže razine u hijerarhiji koji definira samu vrijednost podatka koji se šalje preko mreže (npr. 16-bitna izmjerena vrijednost sa senzora). Zbog programskog sučelja visoke razine na aplikacijskom sloju preko kojeg se definira sadržaj poruke koja se šalje (engl. *payload*) - sadržaj poruke se enkapsulira obično

u strukturu čija se polja popune, a slanje se izvodi prosljeđivanjem odgovarajuće strukture funkciji koja obavi zahtjev za slanjem podataka, atributi su suvišni. Tim više, što u BitCloudu na aplikacijskom sloju nije moguće birati između ZigBee KVP i MSG formata poruka. U BitCloudu korisnik jednostavno uvijek šalje generički *payload* propisane maksimalne duljine, no slobodnog sadržaja. Ovo je posljedica činjenice što je KVP format poruka je ukinut u novijim inačicama ZigBee standarda (ZigBee Alliance, 2008.). Dakle, entitet attribute ne postoji u BitCloudu.

Ukratko, potrebno je više različitih clustera jer je to jedini način razlikovanja poruka bez slanja dodatnih podataka u sadržaju korisničkih poruka. Nadalje, koordinator sve podatke prima/šalje s jednog endpointa ENDPOINT_COORDINATOR, dok neki krajnji čvor može imati više endpointova (npr. ENDPOINT_SERVICE i ENDPOINT_CAMERA). Slika 30 prikazuje odnose između prikazanih ZigBee primitiva u napisanoj mrežnoj aplikaciji. Iz slike se mogu vidjeti numeričke konstante kojima su pojedini primitivi predstavljeni.



Slika 30: Korišteni ZigBee primitivi

4.3.1.2 Implementacija ZigBee primitiva u BitCloudu

Definicija ZigBee primitiva opisanih u prošlom poglavlju nalazi se u datoteci *common.h* svih sudionika mrežne komunikacije (koordinator, senzorski čvor, čvor kamere) i dana je sljedećim kodom:

```
// ZigBee profiles
#define PROFILE_MASLINET 0x01
// ZigBee endpoints
#define ENDPOINT_COORDINATOR 0x01
#define ENDPOINT_CAMERA 0x02
#define ENDPOINT_SENSORS 0x03
#define ENDPOINT_SERVICE 0x04
// ZigBee clusters
// image sending
#define CLUSTER_IMAGE_HEADER_REQ 0x0001
#define CLUSTER_IMAGE_HEADER 0x0002
#define CLUSTER_IMAGE_DATA_REQ 0x0003
#define CLUSTER_IMAGE_DATA 0x0004
#define CLUSTER_IMAGE_EOF 0x0005
// service report
#define CLUSTER_SERVICE_REPORT 0x0006
// sensor data sending
#define CLUSTER_SENSOR_DATA_REQ 0x0007
#define CLUSTER_AIR_SENSOR_DATA 0x0008
```

Razlikovanje primljenih poruka se može izvesti pomoću pokazivača *APS_DataInd_t *ind* kojeg kao argument prima funkcija za primanje podataka *dataInd(APS_DataInd_t *ind)*. Čitanjem elemenata strukture tipa *APS_DataInd_t* mogu se doznati sljedeće informacije o adresama čvorova:

- ***srcAddress*** – adresa čvora koji je poslao poruku. Ovo se može iskoristiti na koordinatorskoj aplikaciji.
- ***dstAddress*** – može se iskoristiti za provjeru je li poruka bila stvarno namijenjena čvoru na koji je stigla. U načelu se ovo ne provjerava na aplikacijskom sloju, jer to rade niži slojevi.

Također, iz *APS_DataInd_t *ind* može se zaključiti o korištenim ZigBee primitivima u pristigloj poruci:

- ***dstEndpoint*** – kojem *endpointu* je namijenjena poruka
- ***srcEndpoint*** – s kojeg *endpointa* je poslana poruka
- ***clusterId*** – koji je korišteni *cluster*. **Ovaj element je najbitniji za razlikovanje poruka**.

Veza između profila, *endpointa* i *clustera* koji se definiraju kao numeričke konstante te programskog koda na aplikacijskoj razini ostvarena je pomoću funkcije *APS_RegisterEndpointReq()*. Ova funkcija služi registriranju jednog *endpointa* na aplikacijskoj razini. Funkcija se poziva prilikom kreiranja mreže. Kao argument prima strukturu tipa *APS_RegisterEndpointReq_t*. Dva najbitnija polja te strukture su:

- ***apsRegisterEndpointReq.APS_DataInd*** - definira *callback* funkciju koja realizira primanje podataka namijenjenih registriranom *endpointu*. Znači, svaki registrirani endpoint ima svoju posebnu tzv. *data indication* funkciju. Unutar svake takve funkcije potrebno je razlikovati samo clustere podržane na tom *endpointu*.
- ***apsRegisterEndpointReq.simpleDescriptor*** - pokazivač na strukturu tipa *SimpleDescriptor_t*. Struktura tipa *SimpleDescriptor_t* sadrži polja u kojima se definira naziv *endpointa* koji će biti registriran u funkciji *APS_RegisterEndpointReq()* i profile ID kojem će endpoint pripadati. Također, *SimpleDescriptor_t* sadrži još listu i broj ulaznih i izlaznih *clustera* na pojedinom čvoru. Iz (Atmel, 2009) se može zaključiti da ako se ta polja ostave nepopunjena ili se inicijaliziraju na 0 odn. NULL, da su svi *clusteri* dozvoljeni. Kako u pripadnim funkcijama za primanje podataka (engl. *data indication*) svakog *endpointa* ionako treba filtrirati *clustere*, specificiranje polja *SimpleDescriptor_t* koja se tiču *clustera* se čini nepotrebnim.

Ukratko, za svaki *endpoint* treba definirati vlastitu strukturu *SimpleDescriptor_t* i vlastitu *callback* funkciju za primanje poruka (engl. *data indication*) i onda s njima za svaki *endpoint* pozvati funkciju *APS_RegisterEndpointReq()*.

Ako se za svaku vrstu poruke koja se šalje deklarira zaseban primitiv za zahtjev za slanje tipa *APS_DataReq_t*, moguće je za svako mrežno slanje definirati posebnu *callback* funkciju *APS_DataConf(APS_DataConf_t *conf)*. To je funkcija koja se poziva na

aplikacijskom sloju kad nakon slanja svake poruke stigne ZigBeejeva MAC-potvrda prijenosa (engl. *acknowledge*), ako je potvrda prijenosa uključena. Različite funkcije *APS_DataConf* mogu se iskoristiti za razlikovanje od koje vrste poslane poruke se vratila potvrda prijenosa i prema tome preusmjeravanje programskog toka po stanjima aplikacijskog upravitelja zadacima (engl. *application task handler*).

Struktura *APS_DataReq_t* koja implementira zahtjev za slanjem poruke na aplikacijskom sloju (engl. *data request*) i koja definira funkciju za potvrdu prijenosa *APS_DataConf* nije u vezi sa strukturom *APS_RegisterEndpointReq_t* pomoću kojeg se definira hoće li se slati potvrda prijenosa na aplikacijskom sloju. Dakle, ako se prilikom definicije polja strukture postavi *APS_RegisterEndpointReq_t stavi .txOptions.acknowledgedTransmission = 0*, *callback* funkcija *APS_DataConf* će se svejedno pozivati. Razlika će jedino biti manji mrežni promet jer se neće slati dodatna automatska potvrda prijenosa (osim standardne ZigBeejeve generirane na MAC sloju).

4.3.1.3 Aplikacijski spremnik za poruke (engl. *message buffer*)

Na aplikacijskom sloju, slanje poruka se inicira pozivom funkcije *APS_DataReq()*. Ova funkcija kao argument prima strukturu tipa *APS_DataReq_t*. Ovom strukturom definirana je pripadnost pripadnost ZigBee profilu, postoji li potvrda prijenosa, koja se funkcija koristi kao *callback* za potvrdu prijenosa te pokazivač na programsku strukturu koja se koristi za prijenos korisnički definiranih podataka (tzv. *application message buffer*). Spremnik za prijenos korisničkih podataka već je prikazan slikom 29. U programskom kodu se struktura sa slike 29 realizira kako je prikazano odsječkom koda u nastavku teksta. *Header* je realiziran poljem *header* duljine APS_ASDU_OFFSET. *Footer* je također polje nazvano *footer* duljine APS_AFFIX_LENGTH - APS_ASDU_OFFSET. Sam sadržaj poruke (*payload*) može biti raznolikog sadržaja i duljine pa je realiziran pomoću unije *msg*. Pritom u kodu treba voditi računa o tome da je duljina unije jednaka duljini njezinog najvećeg elementa. Zato prilikom svakog slanja poruka treba voditi računa o duljini elementa unije kojom se popunjava vrijednost *.asduLength* polja strukture *APS_DataReq_t*. U suprotnom, ako se ako se to polje popuni veličinom cijele unije *msg*, svaki će se put preko mreže slati korisničke poruke duljine najvećeg člana unije.

```
// Aplication message buffer
BEGIN_PACK
typedef struct
{
    // Auxilliary header (stack required)
    uint8_t header[APS_ASDU_OFFSET];
    // Actually application message
    union
    {
        uint8_t imageHeaderReq;
        AppImageHeader_t imageHeader;
        uint16_t imageDataReq;
        AppImageData_t imageData;
        uint8_t imageEOF;
        AppServiceReport_t serviceReport;
        uint8_t sensorDataReq;
        AppAirSensorsData_t airSensorsData;
    } msg;
    // Auxilliary footer (stack required)
    uint8_t footer[APS_AFFIX_LENGTH - APS_ASDU_OFFSET];
} PACK AppMessageBuffer_t;
END_PACK
```

Slijede objašnjenja pojedinih vrsta poruka (članova unije *msg*):

AppServiceReport_t serviceReport

Svaki put kad se krajnji čvor probudi, koordinator indikaciju o statusu. Dakle, servisne izvještaje periodički šalju svi krajnji čvorovi. Servisni izvještaj sadrži podatke RSSI LQI o snazi i kvaliteti radio-signala, trenutni iznos namještene snage odašiljača, temperaturu na pločici osjetilnog čvora (izmjerena senzorom LM73), napon akumulatora i trenutnu verziju programske podrške. Porukama servisnog izvještaja se na koordinatorskom čvoru koristeći RTC sklop dodaje podatak o vremenu primitka (engl. *timestamp*). Na temelju toga se u slučaju prestanka rada mreže zna kad se posljednji put koji od čvorova javio. Ovi podaci se dalje na zahtjev proslijeđuju Wavecom ugradbenom mrežnom poslužitelju. U nastavku je dana deklaracija strukture AppServiceReport_t.

```
// service mode report sent on request from enddevice to coordinator
BEGIN_PACK
typedef struct
{
    int8_t rssi;
    uint8_t lqi;
    uint8_t txPower;
    int8_t internalTemperature;
    uint8_t batteryVoltage;
    char softwareVersion[SOFTWARE_VERSION_STRING_SIZE];
} PACK AppServiceReport_t;
END_PACK
```

Sadržaj strukture tipa *AppServiceReport_t* za slanje servisnog izvještaja prema gornjem isječku programskog koda:

- *rssi* - RSSI (received signal strength indicator)
- *lqi* - LQI (link quality indicator)
- *txPower* - izlazna snaga krajnjeg čvora
- *internalTemperature* - temperatura unutar kućišta
- *batteryVoltage* - preostali napon akumulatora VBAT
- *softwareVersion* – trenutno korištena verzija programske podrške

Pripadni ZigBee primitivi prikazani su u tablici 19.

Tablica 19: ZigBee primitivi poruke *serviceReport*

smjer	krajnji čvor → koordinator
polazišni endpoint	ENDPOINT_SERVICE
odredišni endpoint	ENDPOINT_COORDINATOR
cluster	CLUSTER_SERVICE_REPORT

Uint8_t imageHeaderReq

Zahtjev koordinatora za početkom slanja slike. Na ovu poruku čvor s kamerom šalje poruku *imageHeader* (zaglavlje slike) ukoliko je sadržaj poruke konstanta 0xFF. Korišteni ZigBee primitivi nalaze se u tablici 20.

Tablica 20: ZigBee primitivi poruke *imageHeaderReq*

smjer	coordinator → enddevice
polazišni endpoint	ENDPOINT_COORDINATOR
odredišni endpoint	ENDPOINT_CAMERA
cluster	CLUSTER_IMAGE_HEADER_REQ

AppImageHeader_t imageHeader

Za sad se kao jedini član strukture za slanje zaglavlja slike *imageHeader* šalje samo polje *imageFilename* duljine IMAGE_FILENAME_SIZE. Ovo polje sadrži tekstualni naziv slike koja se šalje.

```
// image header sent as first message of image transfer from enddevice to
coordinator
BEGIN_PACK
```

```
typedef struct
{
    char imageFilename[IMAGE_FILENAME_SIZE];
} PACK AppImageHeader_t;
END_PACK
```

Tablica 20 sadrži korištene primitive.

Tablica 21: ZigBee primitivi poruke *imageHeader*

smjer	enddevice → coordinator
polazišni endpoint	ENDPOINT_CAMERA
odredišni endpoint	ENDPOINT_COORDINATOR
cluster	CLUSTER_IMAGE_HEADER

Uint16_t imageDataReq

Zahtjev koordinatora za paketom slike. Sadržaj paketa je indeks zahtjevanog paketa. Na ovu poruku čvor s kamerom odgovara porukom *imageData* (paket slike). Korišteni ZigBee primitivi nalaze se u tablici 22.

Tablica 22: ZigBee primitivi poruke *imageDataReq*

smjer	coordinator → enddevice
polazišni endpoint	ENDPOINT_COORDINATOR
odredišni endpoint	ENDPOINT_CAMERA
cluster	CLUSTER_IMAGE_DATA_REQ

AppImageData _t imageData

```
// packet of image data sent from enddevice to coordinator
BEGIN_PACK
typedef struct
{
    uint16_t imagePacketIndex;
    uint8_t imagePacketSize;
    uint8_t imagePacketData[MAX_BYTES_PER_PACKET];
} PACK AppImageData_t;
END_PACK
```

Polja strukture za slanje slike tipa AppImageData_t popisana su u nastavku, a korišteni ZigBee primitivi nalaze se u tablici 23.

- *imagePacketIndex* - redni broj poslanog paketa slike

- *imagePacketSize* – veličina slike u bajtovima. Potrebno da bi se na prijemnoj strani moglo odrediti koliko bajtova treba pročitati u primljenom paketu.
- *imagePacketData* - polje bajtova, maksimalne veličine MAX_BYTES_PER_PACKET. Sadrži bajtove slike koji se šalju u jednoj ZigBee transakciji.

Tablica 23: ZigBee primitivi poruke *imageData*

smjer	enddevice → coordinator
polazišni endpoint	ENDPOINT_CAMERA
odredišni endpoint	ENDPOINT_COORDINATOR
cluster	CLUSTER_IMAGE_DATA

Uint8_t imageEOF

Poruka o kraju prijenosa slike. Šalje ju krajnji čvor koordinatoru kad primi zahtjev za sljedećim paketom slike, a već je bio poslao zadnji paket slike. Sadržaj poruke je numerička konstana 0x00. Korišteni ZigBee primitivi nalaze se u tablici 24.

Tablica 24: ZigBee primitivi poruke *imageEOF*

smjer	coordinator → enddevice
polazišni endpoint	ENDPOINT_COORDINATOR
odredišni endpoint	ENDPOINT_CAMERA
cluster	CLUSTER_IMAGE_DATA_REQ

Uint8_t sensorDataRequest

Zahtjev koordinatora za mjernim podacima. Sadržaj paketa je numerička konstanta 0xFF. Na ovu poruku čvor s kamerom odgovara npr. porukom air*SensorsData* (mjerni podaci čvora koji mjeri parametr zraka). Korišteni ZigBee primitivi nalaze se u tablici 25.

Tablica 25: ZigBee primitivi poruke *imageHeaderReq*

smjer	coordinator → enddevice
polazišni endpoint	ENDPOINT_COORDINATOR
odredišni endpoint	ENDPOINT_CAMERA
cluster	CLUSTER_SENSOR_DATA_REQ

AppAirSensorsData_t airSensorsData

Senzorski podaci krajnjeg čvora koji mjeri mikroklimatske parametre zraka (temperatura i relativna vlažnost zraka, razina osvjetljenja, tlak zraka). Struktura *AppAirSensorsData_t* je opisana sljedećim isječkom koda:

```
// air sensor data sent on request from enddevice to coordinator
BEGIN_PACK
typedef struct
{
    bool airTemperatureSuccess;
    int16_t airTemperatureData;
    bool airHumiditySuccess;
    int16_t airHumidityData;
    bool lightSuccess;
    int16_t lightData;
    bool airPressureSuccess;
    int16_t airPressureData1;
    int16_t airPressureData2;
    int16_t airPressureC1;
    int16_t airPressureC2;
    int16_t airPressureC3;
    int16_t airPressureC4;
    int16_t airPressureC5;
    int16_t airPressureC6;
} PACK AppAirSensorsData_t;
END_PACK
```

Iz strukture je vidljivo da krajnji čvor koordinatoru šalje neobrađene podatke, u onom obliku u kakvom ih je izmjerio, kao što je opisano u poglavlju 4.3.2.1.

Korišteni ZigBee primitivi nalaze se u tablici 26.

Tablica 26: ZigBee primitivi poruke *imageHeaderReq*

smjer	coordinator → enddevice
polazišni endpoint	ENDPOINT_COORDINATOR
odredišni endpoint	ENDPOINT_CAMERA
cluster	CLUSTER_IMAGE_DATA_REQ

4.3.2 Krajnji senzorski čvor

4.3.2.1 Specifikacije programske podrške krajnjeg senzorskog čvora

Funkcije senzorskog krajnjeg čvora:

- Većinu vremena provodi u stanju smanjene potrošnje
- Periodički (u fiksnim vremenskim intervalima) prelazi u budno stanje i javlja se koordinatoru slanjem servisnog izvještaja koji sadrži informacije o podešenju snage odašiljača krajnjeg čvora, RSSI, LQI, temperaturu na tiskanoj pločici, napon akumulatora te informaciju o verziji programske podrške
- Kad je u budnom stanju, može primiti zahtjev od koordinatora za akvizicijom i slanjem mjernih podataka – mjeri se temperatura i vlažnost zraka (senzor SHT75), razina osvjetljenja (senzor ISL29013) i tlak zraka (senzor MS5540B).
- Ako nije primljen takav zahtjev, odlazi na spavanje.
- Ako je primljen zahtjev za mjernim podacima, šalje mjerne podatke.
- Nakon što je poslao podatke odlazi na spavanje.

4.3.2.2 Implementacija programske podrške krajnjeg senzorskog čvora

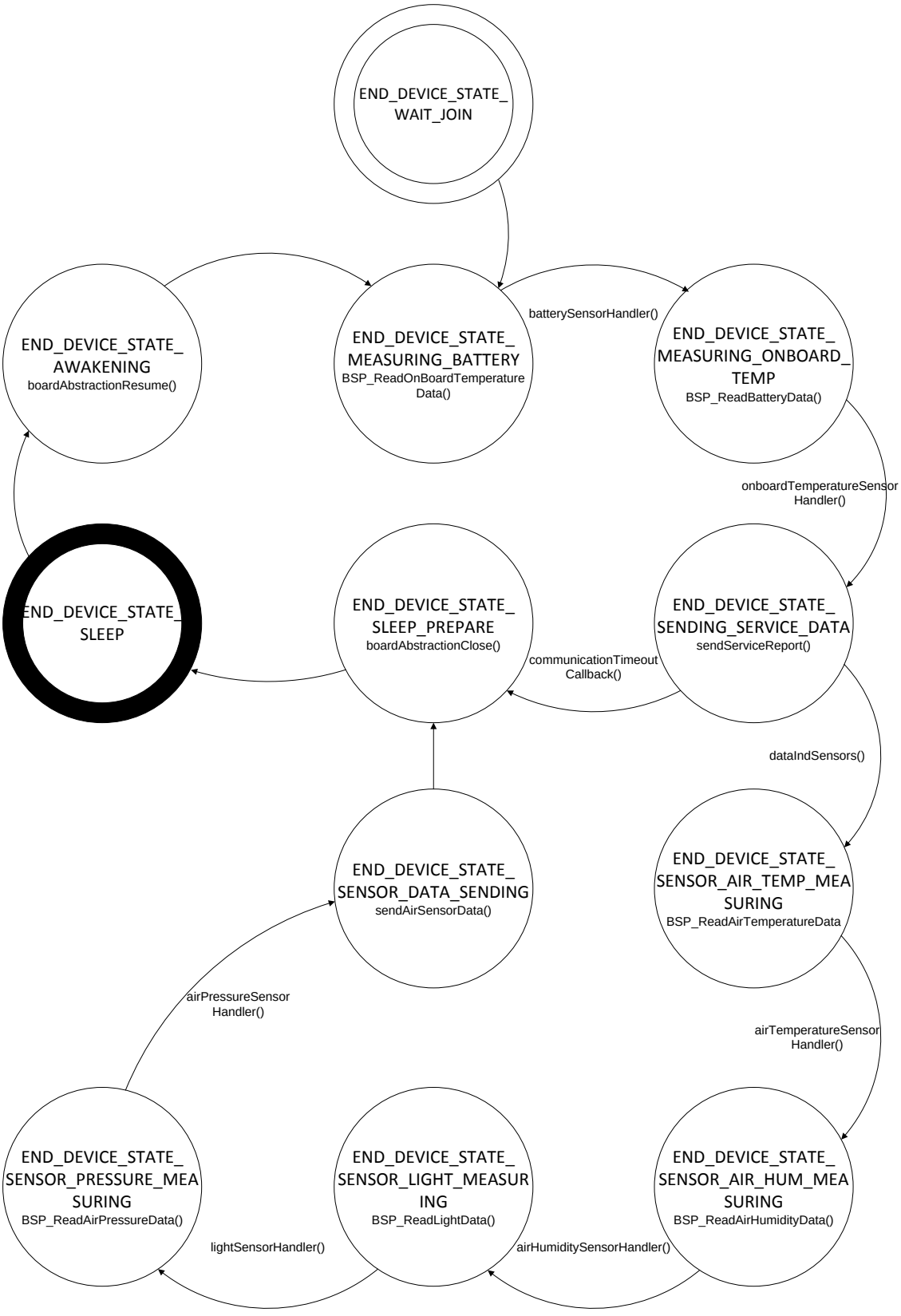
Programski kod aplikacijskog sloja mrežnih aplikacija svih čvorova napisan je u skladu s događajima upravljanim programskim modelom stoga BitCloud. U skladu s tim, programski kod mrežnih aplikacija organiziran je kao stroj s konačnim brojem stanja (engl. *finite state machine*) opisan je dijagramom stanja. Stanja su popisana tablicom 27.

Tablica 27: Popis stanja aplikacijskog upravitelja zadacima

stanje	**opis**
END_DEVICE_STATE_WAIT_JOIN	Čekanje da se krajnji čvor spoji u mrežu.
END_DEVICE_STATE_MEASURING_BATTERY	Dijagnostičko mjerenje napona akumulatora u svrhu servisnog izvještaja.
END_DEVICE_STATE_MEASURING_ONBOARD_TEMP	Krajnji čvor se probudio, dijagnostičko mjerenje temperature pločice u svrhu servisnog izvještaja.
END_DEVICE_STATE_SENDING_SERVICE_DATA	Slanje servisnog izvještaja.
END_DEVICE_STATE_SENSOR_AIR_TEMP_MEASURING	Mjerenje temperature zraka.
END_DEVICE_STATE_SENSOR_AIR_HUM_MEASURING	Mjerenje relativne vlažnosti zraka.
END_DEVICE_STATE_SENSOR_LIGHT_MEASURING	Mjerenje intenziteta svjetla.

END_DEVICE_STATE_SENSOR_PRESSURE_MEASURING	Mjerenje tlaka zraka.
END_DEVICE_STATE_SENSOR_DATA_SENDING	Slanje izmjerenih podataka.
END_DEVICE_STATE_SLEEP_PREPARE	Priprema za odlazak na spavanje (gašenje periferije).
END_DEVICE_STATE_SLEEP	Stanje spavanja.
END_DEVICE_STATE_AWAKENING	Krajnji čvor se probudio.

Slika 31 prikazuje automat po kojem se izvodi programski kod aplikacijskog sloja korisničke mrežne aplikacije čvora koji mjeri parametre zraka. Nakon paljenja čvor se nalazi u početnom stanju END_DEVICE_STATE_WAIT_JOIN u kojem inicijalizira sva potrebna sučelja i početne vrijednosti te poziva funkcije za spajanje u ZigBee mrežu. Nakon spajanja u mrežu čvor ulazi u ciklus periodičke izmjene budnih stanja i stanja spavanja. Duljina spavanja određena je *Config Server* konstantom CS_END_ DEVICE_SLEEP_PERIOD (u milisekundama). U budnom stanju čvor najprije šalje servisni izvještaj. U tu svrhu u stanjima END_DEVICE_STATE_MEASURING_BATTERY i END_DEVICE_STATE_MEASURING_ONBOARD_TEMP čvor mjeri potrebne dijagnostičke podatke koje nakon toga šalje u stanju END_DEVICE_STATE_SENDING_SERVICE_DATA.
Pritom pokreće vremensko brojilo communicationTimeoutTimer. Ako do isteka brojila ne primi zahtjev za slanjem podataka prikupljenih od mikroklimatskih senzora, gasi periferne sklopove i odlazi na spavanje (END_DEVICE_STATE_SLEEP_PREPARE). Ako je primio zahtjev za mjernim podacima, mjeri mikroklimatske parametre (stanja END_DEVICE_STATE_SENSOR_AIR_TEMP_MEASURING, END_DEVICE_STATE_SENSOR_ AIR_HUM_MEASURING, END_DEVICE_STATE_SENSOR_LIGHT_MEASURING, END_ DEVICE_STATE_SENSOR_PRESSURE_MEASURING), šalje ih koordinatoru u stanju END_DEVICE_STATE_SENSOR_DATA_SENDING i odlazi na spavanje (END_DEVICE_ STATE_SLEEP_PREPARE). Nakon buđenja čvor odlazi u stanje END_DEVICE_ STATE_AWAKENING i ciklus se ponavlja.
Mrežna adresa senzorskog čvora za mjerenje mikroklimatskih parametara zraka NWK_NODE_ADDRESS je 0x0001.



Slika 31: Stroj s konačnim brojem stanja aplikacijskog upravitelja zadacima čvora za mjerenje mikroklimatskih parametarar zraka

4.3.2.3 Implementirani mehanizmi za upravljanje pogreškama

communicationTimeoutTimer - to je sigurnosni jednokratni vremensko brojilo (engl. *one-shot timer*) koji nakon isteka šalje čvor na spavanje. Ovim vremenskim brojilom sprječava se da čvor u slučaju greške u komunikaciji ostane stalno budan i potroši cijelu bateriju. Pokreće se u 2 slučaja:

1. Krajnji čvor se probudio, no od koordinatora nije primio zahtjev za mjernim podacima. Zapravo definira koliko dugo krajnji čvor čeka koordinatorove naredbe prije nego ode na spavanje.
2. Krajnji čvor je poslao paket *image data* i čeka da mu koordinator pošalje zahtjev za sljedećim *image data* paketom.

4.3.3 Krajnji čvor s kamerom

4.3.3.1 Specifikacije programske podrške krajnjeg čvora s kamerom

Funkcije krajnjeg čvora s kamerom:

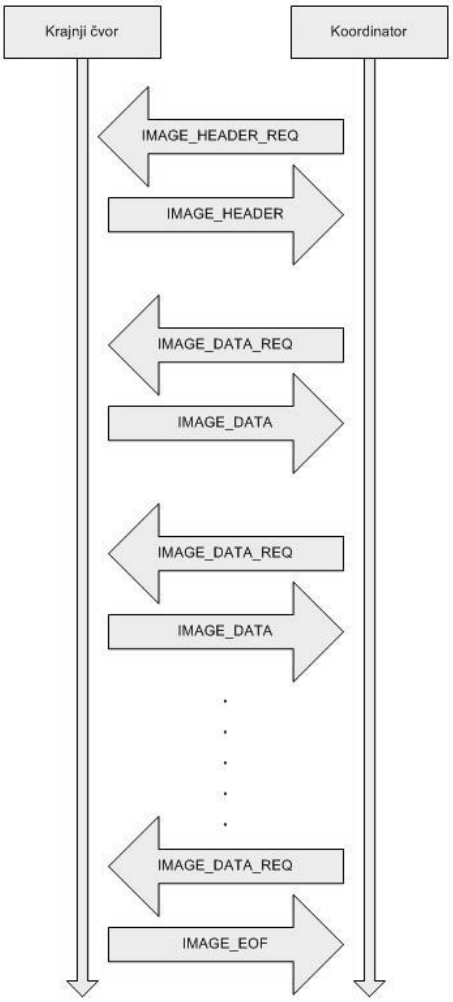
- Većinu vremena provodi u stanju smanjene potrošnje
- Periodički (u fiksnim vremenskim intervalima) prelazi u budno stanje i javlja se koordinatoru slanjem servisnog izvještaja koji sadrži informacije o podešenju snage odašiljača krajnjeg čvora, RSSI, LQI, temperaturu na tiskanoj pločici, napon akumulatora te informaciju o verziji programske podrške
- Kad je u budnom stanju, može primiti zahtjev od koordinatora za akvizicijom i slanjem slike.
- Ako nije primljen takav zahtjev, odlazi na spavanje.
- Ako je primljen zahtjev za obradom slike, šalje sliku paket po paket prema protokolu prijenosa slike.
- Nakon što je poslao cijelu sliku odlazi na spavanje.

4.3.3.2 Implementacija programske podrške krajnjeg čvora s kamerom

Mrežna adresa krajnjeg čvora s kamerom je NWK_NODE_ADDRESS = 0x0002.

Krajni čvor s kamerom namijenjen je akviziciji i prijenosu slike. Zigbee protokol nije namijenjen prenošenju velikih količina podataka niti velikim brzinama podataka. Najveća veličina paketa korisničkih podataka unutar jedne transakcije je 84 bajta. Obzirom na građu strukture AppImageData _t, prenosi se najviše 80 bajtova po paketu. Potrebno je

osigurati prijenos bez gubitaka paketa. U dokumentu (Jeličić, 2009) razmatraju se protokoli sigurnog paketnog prijenosa slike bez gubitaka. Protokol korišten u ovoj aplikaciji razlikuje se od njih utolio što se nakon svakog paketa čeka potvrda prijenosa i nakon svakog paketa u slučaju pogreške može slijediti retransmisija. Sljedeći se paket prenosi tek kad je osigurano da je trenutni uspješno prenesen. U tu svrhu je razvijen jednostavan protokol prikazan slikom 32.



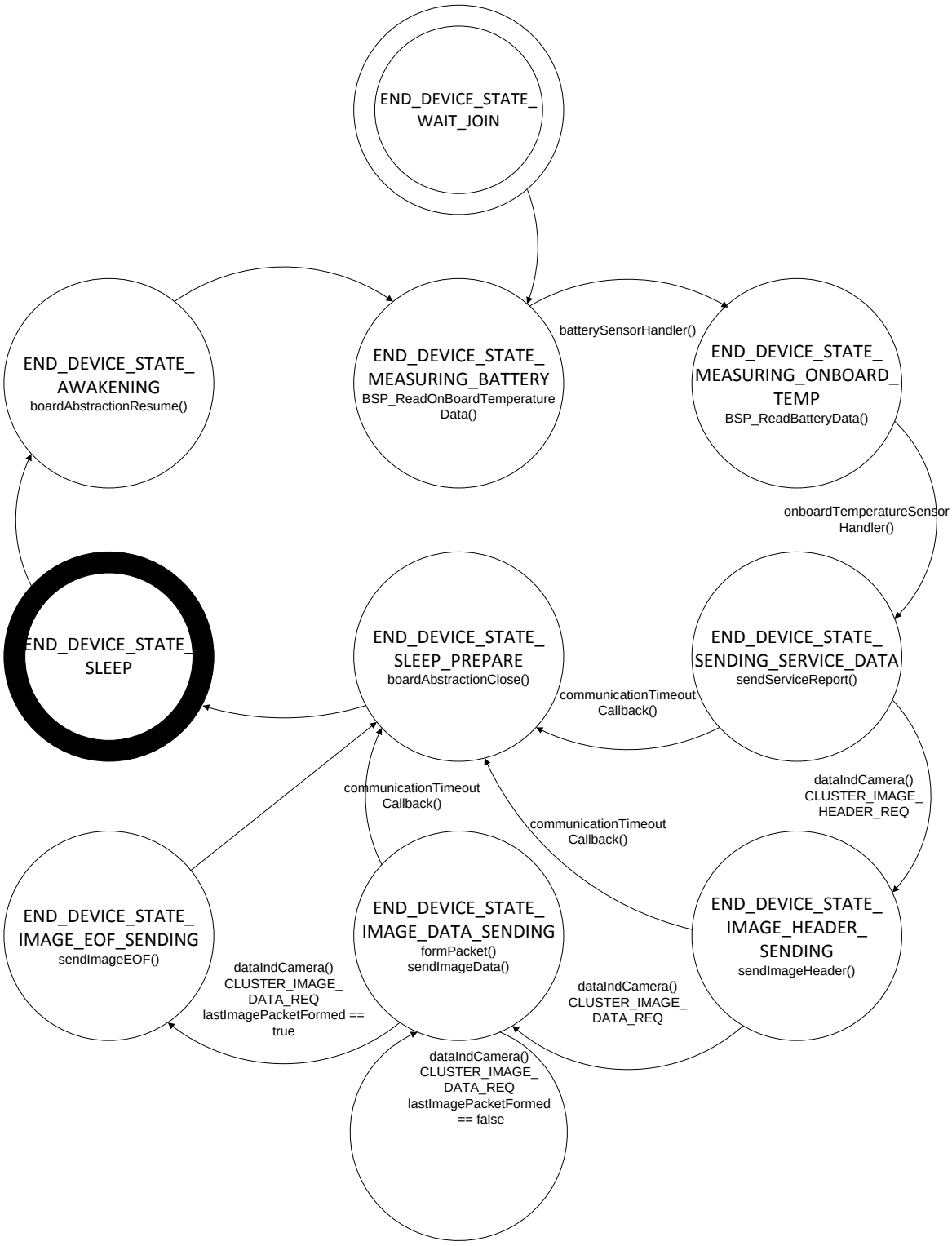
Slika 32: Implementirani protokol prijenosa slike

Protokol radi na način da koordinator postavlja zahtjev za određenim paketom. Zahtjev za sljedećim paketom slike postavlja se tek kad je prethodni paket bio uspješno prenesen. U slučaju da neki paket nije bio uspješno prenesen, koordinator će ponoviti zahtjev za traženim paketom. Krajnji čvor ne vodi računa o uspješnosti prijenosa već samo isporučuje traženi paket. Stanja aplikacijskog upravitelja zadacima dana su u tablici 28.

Tablica 28: Stanja aplikacijskog upravitelja zadacima krajnjeg čvora s kamerom

stanje	opis
END_DEVICE_STATE_WAIT_JOIN	Čekanje da se krajnji čvor spoji u mrežu.
END_DEVICE_STATE_MEASURING_BATTERY	Krajnji čvor se probudio, dijagnostičko mjerenje napona akumulatora u svrhu servisnog izvještaja.
END_DEVICE_STATE_MEASURING_ONBOARD_TEMP	Dijagnostičko mjerenje temperature pločice u svrhu servisnog izvještaja.
END_DEVICE_STATE_SENDING_SERVICE_DATA	Slanje servisnog izvještaja.
END_DEVICE_STATE_IMAGE_HEADER_SENDING	Slanje zaglavlja slike.
END_DEVICE_STATE_IMAGE_DATA_SENDING	Slanje paketa slike.
END_DEVICE_STATE_IMAGE_EOF_SENDING	Slanje EOF znaka za završetak prijenosa slike.
END_DEVICE_STATE_SLEEP_PREPARE	Priprema za odlazak na spavanje (gašenje periferije).
END_DEVICE_STATE_SLEEP_PREPARE	Stanje spavanja.
END_DEVICE_STATE_AWAKENING	Krajnji čvor se probudio.

Slika 33 prikazuje automat po kojem se izvodi programski kod aplikacijskog sloja korisničke mrežne aplikacije čvora s kamerom. Nakon paljenja čvor se nalazi u početnom stanju END_DEVICE_STATE_WAIT_JOIN. Nakon spajanja u mrežu čvor ulazi u ciklus periodičke izmjene budnih stanja i stanja spavanja. U budnom stanju čvor najprije šalje servisni izvještaj. U tu svrhu u stanjima END_DEVICE_STATE_MEASURING_BATTERY i END_DEVICE_STATE_MEASURING_ONBOARD_TEMP čvor mjeri potrebne dijagnostičke podatke koje nakon toga šalje u stanju END_DEVICE_STATE_SENDING_SERVICE_DATA. Pritom pokreće vremensko brojilo communicationTimeoutTimer. Ako do isteka brojila ne primi zahtjev za slanjem slike, gasi periferne sklopove i odlazi na spavanje (END_DEVICE_STATE_SLEEP_PREPARE). Ako je u međuvremenu primio zahtjev za akvizicijom i slanjem slike, prelazi u stanje END_DEVICE_STATE_ IMAGE_HEADER_SENDING u kojem šalje zaglavlje slike. Nakon toga čvor iterativno ulazi u stanje END_DEVICE_STATE_IMAGE_DATA_SENDING toliko dugo dokle od koordinatora prima zahtjev za novim paketima i ima još paketa slike koje nije poslao. Ako koordinator predugo ne postavlja zahtjev za novim paketom, krajnji čvor odlazi na spavanje.



Slika 33: Stroj s konačnim brojem stanja aplikacijskog upravitelja zadacima krajnjeg čvora s kamerom

Valja napomenuti da su paketi slike spremljeni u memoriji kamere, a ne krajnjeg čvora i čitaju se serijski bajt po bajt jer krajnji čvor nema dovoljno memorije za pohranu slike reda veličine 100 kilobajta (JPEG kompresija). Nakon što pošalje zadnji paket slike, a ponovo primi zahtjev za sljedećim paketom, krajnji čvor šalje EOF znak u stanju END_DEVICE_STATE_IMAGE_EOF_SENDING. Nakon poslane slike odlazi na spavanje (END_DEVICE_STATE_SLEEP_PREPARE). Nakon buđenja čvor odlazi u stanje END_DEVICE_STATE_AWAKENING i ciklus se ponavlja.

4.3.3.3 Mehanizmi za upravljanje pogreškama

communicationTimeoutTimer – funkcionira analogno vremenskom brojilu na krajnjem čvoru s kamerom. Pokreće se u 2 slučaja:

1. Krajnji čvor se probudio, no od koordinatora nije primio zahtjev za slanjem zaglavlja slike. Zapravo definira koliko dugo krajnji čvor čeka koordinatorove naredbe prije nego ode na spavanje.
2. Krajnji čvor je poslao paket tipa *image header* i čeka da mu koordinator pošalje zahtjev *image data request*.
3. Krajnji čvor je poslao paket tipa *image data* i čeka da mu koordinator pošalje zahtjev za sljedećim paketom *image data request*.

4.3.4 Koordinatorski čvor

4.3.4.1 Specifikacije programske podrške koordinatorskog čvora

U okviru bežične ZigBee mreže razvijene za potrebe projekta Maslinet, zadaće koordinatora su:

- Odgovoran je za uspostavu i održavanje mreže. Prema (ZigBee Alliance, 2008.), krajnji čvor nikad ne spava. To je jedini čvor u mreži koji je stalno budan. Služi kao centralna jedinica nadređena krajnjim čvorovima i GPRS modulu (engl. *parent node*).
- Po potrebi budi GPRS modem kad je potrebno poslati podatke preko interneta.
- Prosljeđuje krajnjim čvorovima zahtjev za akvizicijom i slanjem slike i zahtjev za dohvatom senzorskih podataka primljen od GPRS modula ili generiran alarmom RTC sklopa
- Sinkroniziranje sata RTC sklopa sa satom GPRS modema na zahtjev.
- Od krajnjih čvorova periodički prima izvještaj o statusu svakog čvora.
- Na zahtjev šalje GPRS modulu aktualne servisne izvještaje.
- Od krajnjeg čvora s kamerom prima pakete slike i prosljeđuje ih GPRS modemu koji ih šalje na internetski poslužitelj.
- Od krajnjih čvorova (nakon poslanog zahtjeva) prima podatke sa senzora i prosljeđuje ih po potrebi GPRS modemu koji ih šalje na internetski poslužitelj.

4.3.4.2 Implementacija programske podrške koordinatorskog čvora

Koordinator ima mrežnu adresu NWK_NODE_ADDRESS = 0x0000. Popis stanja aplikacijskog upravitelja zadacima koordinatora dan je u tablici 29.

Tablica 29: Popis stanja aplikacijskog upravitelja zadacima koordinatora

stanje	**opis**
COORD_STATE_WAIT_JOIN	Čekanje da se uspostavi ZigBee mreža.
COORD_STATE_IDLE	Koordinator čeka na interakciju (ništa se ne događa).
ERROR_STATE_SEND_ERROR	Dogodila se pogreška. Slanje koda pogreške Wavecom GPRS modulu.
SENS_STATE_ZIGBEE_SEND_AIR_SENSOR_DATA_REQ	Koordinator šalje krajnjem čvoru zahtjev za mjernim podacima.

SENS_STATE_WCOM_SEND_AIR_SENSOR_DATA	Koordinator šalje Wavecom GPRS modulu prikupljene senzorske podatke.
IMG_STATE_USART_RX	Obrada poruke pristigle od Waveccoma preko serijskog sučelja.
IMG_STATE_GPRS_CONNECT_REQ	Slanje zahtjeva Wavecomu za uspostavu GPRS veze.
IMG_STATE_ZIGBEE_SEND_HEADER_REQ	Slanje zahtjeva za zaglavljem slike krajnjem čvoru s kamerom (inicira sekvencu prijenosa slike).
IMG_STATE_WCOM_SEND_HEADER	Slanje zaglavlja slike Wavecom GPRS modulu.
IMG_STATE_ZIGBEE_SEND_DATA_REQ	Slanje zahtjeva za mjernim podacima krajnjem čvoru za mjernje mikroklimatskih parametara zraka (inicira sekvencu prijenosa mjernih podataka).
IMG_STATE_WCOM_SEND_DATA	Slanje paketa slike Wavecomu.
IMG_STATE_WCOM_SEND_EOF	Slanje EOF paketa Wavecom modulu.

4.3.4.3 Mehanizmi za upravljanje pogreškama u komunikaciji prema ZigBee mreži

Slično kao i na krajnjem čvoru, treba osigurati dojavu greške u slučaju da nakon zahtjeva poslanog s koordinatora krajnji čvor u razumnom vemenu ne vrati odgovor. Za to se brinu sljedeća vremenska brojila:

imageHeaderRequestErrorTimer - nakon poslane poruke *image header request* krajnji čvor do isteka brojila nije vraćena poruka *image header*

imageDataRequestErrorTimer - nakon poslane poruke *image request* nije vraćena poruka *image data*

sensorDataRequestErrorTimer - nakon poslane poruke *sensor request* nisu vraćena očitanja sa senzora

U slučaju da okine bilo koji od gornjih brojila, to je znak da postoje problemi u ZigBee komunikaciji. U tom slučaju ista se transakcija pokušava ponoviti ZIGBEE_NO_RESPONSE_TIMEOUT puta. Tome služe sljedeća brojila:

zigbeeNoResponseTimeoutCounter - radi na način da je početno napunjen na vrijednost ZIGBEE_NO_RESPONSE_TIMEOUT. ZIGBEE_NO_RESPONSE_TIMEOUT je zaliha koja govori koliko puta se transakcija smije ponoviti prije nego se proglasi greška. Svaka uspješna transakcija resetira zalihu na ZIGBEE_NO_RESPONSE_TIMEOUT. Uzastopne neuspješne transakcije smanjuju zalihu. Kad zaliha dođe do 0, treba zaustaviti trenutnu sekvencu (npr. prijenos slike) i proglasiti stanje pogreške.

imageIndexTimeoutCounter - u slučaju da se zahtijevani i primljeni indeks paketa slike ne poklapaju, trenutni paket se zahtjeva ZIGBEE_IMAGE_DATA_INDEX_TIMEOUT puta prije proglašenja pogreške. Sva gore navedena i u nastavku opisana brojila funkcioniraju isto kao **zigbeeNoResponseTimeoutCounter**.

4.3.4.4 Mehanizmi za upravljanje pogreškama u serijskoj komunikaciji prema Wavecomu

U slučaju pogreški u serijskoj komunikaciji aktiviraju se brojila:

- **GPRSConnectTimeoutCounter** - ponavljanje spajanja na GPRS maksimalno WCOM_GPRS_CONNECT_TIMEOUT puta u slučaju da cb_GprsConnect ne uspije
- **imageHeaderTimeoutCounter** - ponavljanje prijenosa image header paketa slike na Wavecom maksimalno WCOM_IMAGE_HEADER_TIMEOUT puta ako ne uspije
- **imageDataTimeoutCounter** - ponavljanje prijenosa headera slike na Wavecom maksimalno WCOM_IMAGE_DATA_TIMEOUT puta ako ne uspije iz prve
- **imageEOFTimeoutCounter** - ponavljanje prijenosa EOF znaka za kraj slike na Wavecom maksimalno WCOM_IMAGE_EOF_TIMEOUT puta ako ne uspije iz prve

U slučaju detekcije pogreške, implementirano je da koordinator odlazi u centralno stanje za dojavu pogreški stanje ERROR_STATE_SEND_ERROR i tamo ovisno o tipu pogreške dojavljuje jednu od sljedećih poruka:

NO_ERROR
RTC_ERROR_INITIALIZATION_FAILED
RTC_ERROR_TIMER_SETTING_FAILED
RTC_ERROR_TIME_SETTING_FAILED
RTC_ERROR_TIME_GETTING_FAILED
RTC_ERROR_GETTING_TIME_FOR_ TIMESTAMP_FAILED
RTC_ERROR_TIMER_ENABLING_FAILED
RTC_ERROR_RTC_NOT_READY
ZIGBEE_ERROR_NO_IMAGE_HEADER_RESPONSE
ZIGBEE_ERROR_NO_IMAGE_DATA_OR_EOF_RESPONSE
ZIGBEE_ERROR_NO_SENSOR_DATA_RESPONSE
IMAGE_ERROR_WRONG_PACKET_INDEX
IMAGE_ERROR_TRANSFER_ALREADY_IN_PROGRESS
WCOM_ERROR_COULDNT_SEND_IMAGE_HEADER
WCOM_ERROR_COULDNT_SEND_IMAGE_DATA
WCOM_ERROR_COULDNT_SEND_IMAGE_EOF
WCOM_ERROR_COULDNT_SEND_AIR_SENSOR_DATA

4.4 Sustav za udaljenu nadogradnju programske podrške

U nastavku je opisan scenarij udaljenog programiranja kakav bi se mogao primijeniti u bežičnoj osjetilnoj mreži Maslinet obzirom na njenu topologiju, mogućnosti i ograničenja. U prvoj inačici sustava za nadogradnju programske podrške najbolje je koristiti čim jednostavnija rješenja i pošto se radi o zadaći visokog rizika izbjegavati nepotrebni automatizam.

4.4.1 Sučelje na strani poslužitelja

Slanje i pokretanje nadogradnje iniciraju sa udaljenog mrežnog poslužitelja. Potrebno je razviti korisničko sučelje na preko kojeg korisnik započinje i nadzire nadogradnju programske podrške. Sučelje mora sadržavati sljedeće elemente:

- prikaz verzije programske podrške koja je trenutno instalirana na kojem čvoru – ova informacija već je ugrađena u poruke servisnog izvještaja pa se lako može prikazati
- prikaz verzije programske podrške koja se nalazi u vanjskoj FLASH memoriji kojeg čvora i čeka da bude instalirana – ova informacija se ne šalje u trenutnoj inačici mreže – treba ju uvesti u sljedeću inačicu, u poruku servisnog izvještaja
- naredbu za pojedinačnim slanjem nove verzije programske podrške u vanjsku memoriju svakog čvora
- naredbu za pojedinačnim pokretanjem instalacije nadogradnje (pokretanje *bootloader* programa)
- mehanizme zaštite od pokretanja slanja i instalacije programske podrške koji koriste podatke o količini energije u sustavu. U tu svrhu predlaže se da se iskoriste 2 podatka:
    - doba dana (dan/noć) – korisno kod koordinatorskog čvora napajanog solarno jer je po noći nema pritoka energije pa je rizik prestanka rada tijekom nadogradnje veći
    - napon akumulatora – ne dozvoliti nadogradnju ako je napon akumulatora niži od nekog graničnog napona

Na strani poslužitelja ne koristi se nikakvo kodiranje (npr. inkrementalno nadogradnja prenošenjem samo dijelova programske slike koji se u novoj inačici programske podrške razlikuju). Programski kod šalje se u izvornom obliku kao binarna datoteka. Nedostatak ovog pristupa je što se mora slati puna veličina programske slike, a prednost što

omogućava jednostavnu izvedbu *bootloader* aplikacije na krajnjem čvoru jer pristigli blokovi ne zahtijevaju nikakvu obradu. Iz tog razloga se odustalo od korištenja tekstualnih SREC datoteka.

4.4.2 Propagacija programskog koda kroz mrežu Maslinet

Potrebno je dizvesti sigurni protokol mrežnog prijenosa programske slike u kojem se sljedeći paket toliko dugo ne šalje sa Wavecoma dok prethodni nije uspješno stigao na odredište.

Maslinet je *single-hop* zvjezdasta mreža s koordinatorom u središtu i samo 3 krajnja čvora. Obzirom na *single-hop* strukturu, nije potrebno razmatrati složene algoritme propagacije kroz mrežu. Za krajnje ZigBee čvorove, koordinatorski čvor je uvijek izvor programske slike, a krajnji čvorovi odredište. Za koordinatorski čvor, izvor programske slike je GPRS modul Wavecom.

Proces propagacije programske slike započinje na udaljenom serveru odabirom programske slike i čvora kojem se slika želi poslati. Prijenos je dopušten ako su zadovoljeni minimalne energetske pretpostavke (ako se šalje koordinatorskom čvoru – dan, ako se šalje krajnjem čvoru – minimalni napon akumulatora). Najprije slika FTP protokolom GPRS-om propagira do lokalnog ugradbenog poslužitelja Wavecom. Cijela programska slika se pohranjuje u njegovoj memoriji. Daljnji tijek procesa ovisi o odredištu.

4.4.2.1 Prijenos programskog koda ZigBee koordinatoru

Ako je programska slika bila namijenjena koordinatoru, koordinator serijski prenosi sliku u vlastitu vanjsku FLASH memoriju. Prijenos započinje tako da Wavecom modul koordinatoru postavlja zahtjev za prijenosom programske slike. Zahtjev sadrži i identifikator verzije nove programske slike. Ukoliko je spreman, koordinator dopušta prijenos. Serijski prijenos je organiziran tako da koordinator postavlja upit za svakim paketom programske slike na što Wavecom odgovara slanjem paketa preko serijskog sučelja. Tek kad koordinator uspješno pohrani trenutni paket u svoju FLASH memoriju, zahtjeva sljedeći paket slike od Wavecoma. Ako nije uspješno primio paket, prijenos se ponavlja N puta. Ako ni nakon toga paket nije uspješno prenesen, cijela sekvenca prijenosa programske slike se prekida i koordinator Wavecom modulu javlja poruku o greški koja se prosljeđuje do udaljenog poslužitelja. Prijenos završava na način da

koordinator zahtjeva sljedeći paket programske slike, a Wavecom modul mu odgovori poruku EOF jer je bio poslao sve pakete.

Nakon što je koordinator uspješno primio EOF poruku i time završio sekvencu upisivanja slike u svoju FLASH memoriju, šalje Wavecomu poruku da je dozvoljeno pokretanje nadogradnje koji poruku prosljeđuje na udaljeni mrežni poslužitelj. Kad korisnik na udaljenom poslužitelju pokrene proces nadogradnje, ZigBee koordinator će preko serijskog sučelja primiti odgovarajuću poruku na koju će pokrenuti *bootloader* program i započeti nadogradnju.

4.4.2.2 Prijenos programskog koda do krajnjih ZigBee čvorova

Slanje programske kroz ZigBee mrežu se izvodi u *unicast* načinu. To znači da se svakom čvoru programsku kod šalje odvojeno. To je razumljivo obzirom na činjenicu da se mreža sastoji od 3 različita krajnja čvora.

Protokol serijskog prijenosa između Wavecom uređaja i koordinatora ostaje isti u i u ovom slučaju. Razlika je jedino u tome što sve poruke moraju propagirati do krajnjeg čvora ZigBee mreže kojemu je programska slika namijenjena. Paketi koji se prenose serijskim protokolom između Wavecom modula i koordinatora te koordinatora i krajnjeg čvora ZigBee mreže iste su veličine (80 bajtova) tako da ne zahtijevaju prepakiravanje.

Prvi paket (zahtjev za nadogradnjom) ostavlja krajnji čvor budnim sve do primanja EOF paketa.

4.4.3 Programska podrška za nadogradnju na čvorovima ZigBee mreže

Upravljanje verzijama programskog koda trebalo bi realizirati kako je objašnjeno u poglavlju 2.3.3. – u memoriju se spremaju dvije slike od kojih je jedna uvijek trenutna, a druga nova i gdje trenutna služi mehanizmima za povratak na prošlu verziju u slučaju nestabilnosti nove aplikacije. Takav mehanizam koji koristi WDT brojilo također je već opisan u poglavlju 2.3.3. Kritična aplikacija u ovom scenariju je *bootloader* aplikacija koja se ne može nadograditi na daljinu.

5 Zaključak

Proizveden je prototip bežičnog osjetilnog čvora FER Čvorak v2.0b izrađen za korištenje u budućoj inačici mreže Maslinet prema opisu sklopovlja danom u ovom radu. Čvor ispravlja nedostatke postojeće inačice osjetilnog čvora, dodaje mogućnost udaljene nadogradnje programske podrške i ima manje dimenzije. Čvor je razvijen tako da bude kompatibilan sa svim trenutno korištenim senzorima te budućom inačicom mrežnog poslužitelja/GPRS modula Wavecom i novim sučeljem za spajanje kamere. Čvor FER Čvorak v2.0b je trenutno u fazi testiranja sklopovlja po podsustavima.

Paralelno sa sklopovljem novog čvora razvijana je programska podrška za pristup korištenim senzorima i mrežna aplikacija za postojeće čvorove ugrađene u mrežu Maslinet – krajnji senzorski čvor, krajnji čvor s kamerom i koordinatorski čvor. Napisana programska podrška kompatibilna je s novim osjetilnim čvorom FER Čvorak v2.0b zbog iste korištene platforme i istih ključnih komponenata i sučelja. Nakon testa ispravnosti sklopovlja u planu je isprobati mrežne aplikacije na novom osjetilnom čvoru.

Također, potrebno je izmjeriti potrošnju novog bežičnog osjetilnog čvora u karakterističnim stanjima – neaktivnom stanju („spavanju“) te slanju, primanju podataka i obradi informacija u aktivnom („budnom“) stanju.

Također, trenutno je u razvoju programska podrška koja iskorištava mogućnost udaljene nadogradnje bežičnog osjetilnog čvora novom verzijom programske podrške preko bežične mreže. Potrebno je još dovršiti *bootloader* program za samoprogramiranje osjetilnog čvora čitanjem programskog koda iz vanjske FLASH memorije i spremanjem u unutarnju programsku memoriju mikrokontrolera. Nakon toga potrebno je proširiti mrežnu aplikaciju bežične osjetilne mreže Maslinet tako da podržava scenarij udaljene nadogradnje predložen u ovom radu, vodeći računa o izboru trenutka nadogradnje ovisno o količini dostupne energije u mreži i stabilnosti nove verzije programskog koda nakon nadogradnje.

6 Literatura

Atmel AT25DF041A: 4-megabit 2.3-volt or 2.7-volt Minimum SPI Serial Flash Memory. - 9. 2008..

Atmel BitCloud Stack Documentation [Help file]. - [s.l.] : Atmel, 2009.

Atmel BitCloud: User Guide [Mrežno] // Atmel. - 5. 2009.. - 13. 11. 2009.. - http://www.atmel.com/dyn/resources/prod_documents/doc8199.pdf.

Atmel ZigBit™ 2.4 GHz Amplified Wireless Modules Datasheet. - [s.l.] : Atmel, 2009..

Bačan Andro Komunikacijski modul bežičnog osjetilnog čvora [Radovi]. - Zagreb : Fakultet elektrotehnike i računarstva, 2007..

Brown S. Updating Software in Wireless Sensor Networks: A Survey [Izvješće]. - Maynooth : Dept. of Computer Science, National University of Ireland, 2006..

Brunelli D. [i dr.] Photovoltaic scavenging systems: Modeling and optimization [Časopis]. - [s.l.] : Microelectronics Journal, 2008.

Brunelli Davide Electronic systems for ambient intelligence [Knjiga]. - Bologna : Department of electronics, computer science and systems of University of Bologna, 2007.

Dallas Semiconductor DS1337: I2C Serial Real-Time Clock. - 7. 2009..

Dallas Semiconductor DS2411R: Silicon Serial Number with VCC Input.

Diodes Incorporated AP2280: Single Channel Slew Rate Controlled Load Switch. - 2. 2009..

Holger Karl i Willig Andreas Protocols and Architectures for Wireless Sensor Network [Knjiga]. - [s.l.] : Wiley, 2005.

Intersema MS5540B (RoHS*) MINIATURE BAROMETER MODULE. - 1. 2008..

Intersil ISL29013: Light-to-Digital Output Sensor with High Sensitivity, Gain Selection, Interrupt Function and I2C Interface. - 9. 2008..

Jeličić Vana Modifikacija protokola ZigBee za energetski učinkovit i pouzdan prijenos slike u bežičnim multimedijskim mrežama osjetila [Radovi]. - Zagreb : Fakultet elektrotehnike i računarstva, 2009.

Jeong J. Incremental Network Programming for Network Sensors [Radovi] // 1st Annual IEEE Communications Society Conf. on Sensor and Ad Hoc Networks and Communications. - [s.l.] : IEEE, 2004..

Koshy J. i Pandy R. Remote Incremental Linking for Energy-Efficient Reprogramming of Sensor [Radovi] // 2nd European Workshop on Wireless Sensor Networks (EWSN'05). - [s.l.] : IEEE, 2005..

Linear Technology LTC1540: Nanopower Comparator with Reference. - 1997..

Maslinet Maslinet [Mrežno] // Maslinet. - 2009.. - 25. 4. 2010.. - http://www.maslinet.com/main.html.

National Semiconductor LM73: 2.7V, SOT-23, 11-to-14 Bit Digital Temperature Sensor with 2-Wire Interface. - 5. 2009..

Reijers N. i Langendoen K. Efficient Code Distribution in Wireless Sensor Networks [Radovi] // Second ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA'03). - San Diego : ACM, 2003..

Sensirion Datasheet SHT7x (SHT71, SHT75) Humidity and Temperature Sensor. - 5. 2010..

Shaikh R.P., Thakare V.M. i Dharaskar R.V. Efficient Code Dissemination Reprogramming Protocol for WSN [Časopis]. - [s.l.] : (IJCNS) International Journal of Computer and Network Security, 2010. - 2 : Svez. 2.

ST Microelectronics STM809, STM810, STM811, STM812: Reset Circuit. - 1. 2010..

Texas Instruments SN54AHC08, SN74AHC08: QUADRUPLE 2-INPUT POSITIVE-AND GATES. - 6. 2003..

Texas Instruments SN74AHC1G00: Single 2-input positive-NAND Gate. - 9. 2008..

Texas Instruments TPS61221: LOW INPUT VOLTAGE STEP-UP CONVERTER IN 6 PIN SC-70 PACKAGE. - 1. 2009..

Texas Instruments TS5A3159: 1-Ohm SPDT analog switch. - 8. 2004..

ZigBee Alliance ZigBee Specification [Mrežno] // ZigBee Alliance. - 17. 1. 2008.. - 11. 11. 2009.. - http://www.zigbee.org/ZigBeeSpecificationDownloadRequest/tabid/311/Default.aspx.

Sažetak

Bežični osjetilni čvor nove generacije

Od 2007. u toku je razvojno istraživački projekt Maslinet. Cilj projekta je razviti prototip multimodalne bežične mreže osjetila za primjenu u maslinarstvu. Namjena mreže je vizualno praćenje broja maslinovih muha te praćenje mikroklimatskih parametara - temperature i mokrine tla, temperature, tlaka i relativne vlažnosti zraka te razine osvjetljenja. Ovaj rad fokusira se na osjetilne čvorove ZigBee bežične mreže korištene u projektu Maslinet.

Najprije je definiran koncept bežičnog osjetilnog čvora, njegove karakteristične komponente, specifičnosti napajanja. Opisani su temeljni koncepti udaljene nadogradnje programske podrške. Opisani su dosad korišteni osjetilni čvorovi. Zatim je detaljno opisano sklopovlje novorazvijenog čvora FER Čvorak v2.0b po podsustavima – napajanje, sklopovlje za upravljanje perifernim uređajima, sklopovlje za reset, podsustav za obradu podataka i radiokomunikaciju, vanjska memorija za trajnu pohranu. Opisane su specifičnosti tiskane pločice osjetilnog čvora. Nakon toga dan je opis programske podrške razvijene za lokalni dio bežične mreže Maslinet sastavljene od čvorova FER Čvorak v2.0b – opisano je korišteno programsko sučelje BitCloud, programska podrška za senzore te su opisane mrežne aplikacije pojedinih čvorova razvijene u okviru projekta Maslinet. Iznesen je koncept programske podrške za udaljeno programiranje za mrežu Maslinet.

Na kraju su predložene aktivnosti vezane uz daljnji rad: testiranje sklopovlja čvora, testiranje rada ssenzora i mrežne aplikacije s novim čvorom, mjerenje potrošnje te dovršavanje i testiranje programske podrške za udaljeno programiranje čvorova.

Ključne riječi:

Bežični osjetilni čvor, bežične mreže osjetila, programiranje na daljinu

Summary

New generation of wireless sensor node

Since 2007 there is an ongoing research and development project Maslinet. The project aims to develop a prototype of multi-modal wireless sensor network for use in olive production. The purpose of the network is to enable visual tracking of the olive fly and monitoring of microclimatic parameters – soil temperature and moisture, air temperature, pressure, relative humidity and light intensity. This paper focuses on the sensor nodes of ZigBee wireless network used in the project Maslinet.

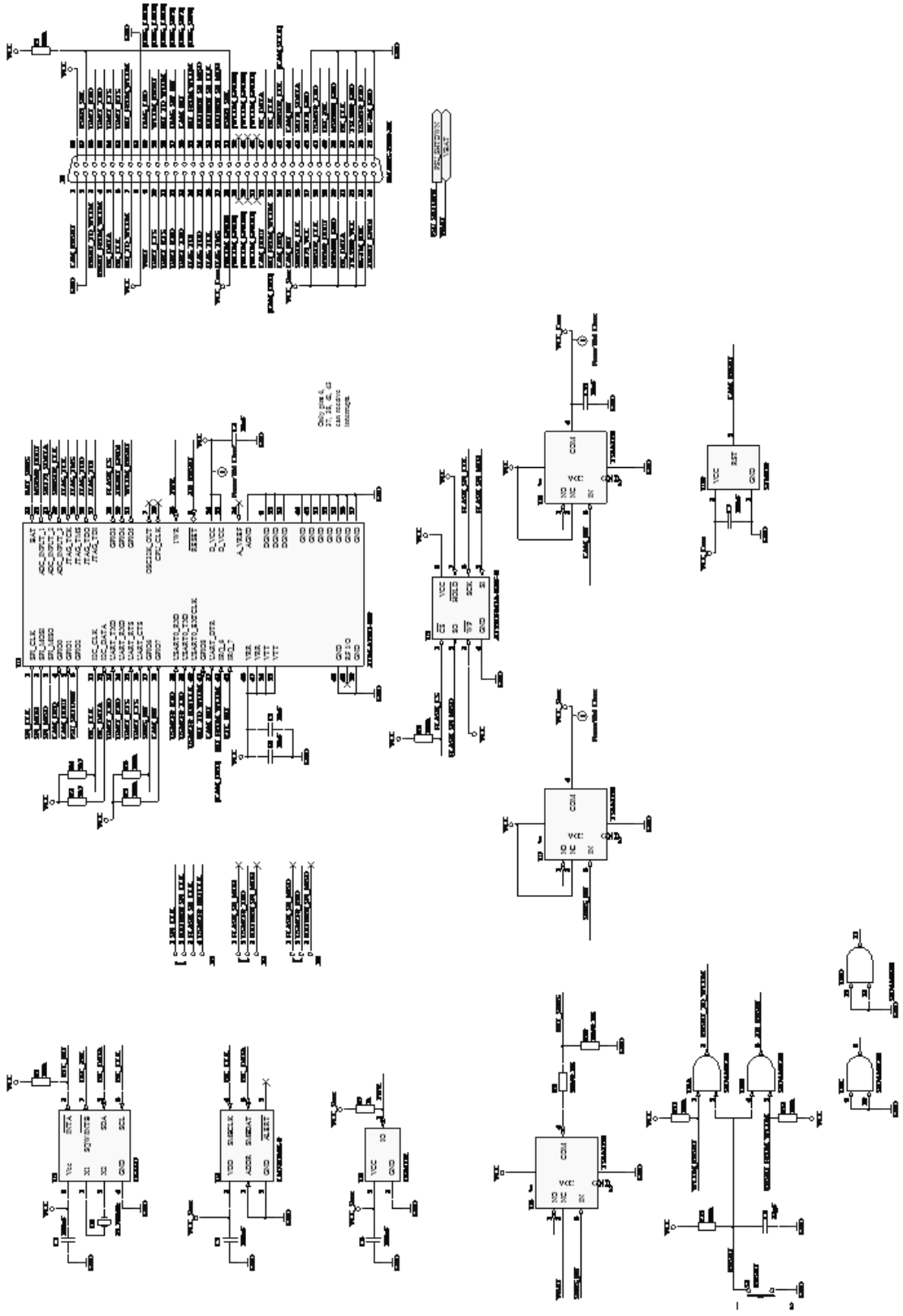
Firstly, a concept of wireless sensor node, its characteristic components and specifics of power supply system are being defined. Key concepts of firmware download over the air are being described. Then, existing wireless sensor nodes used so far are described. After that hardware of all subsystems of a newly developed node FER Čvorak v2.0b are being described - the power system, peripheral control system, reset hardware, data processing and radio-communications subsystems, external memory for permanent storage. Specifics of the printed circuit board design are also being described. Then, description of the software for wireless sensor network consisting of nodes Maslinet FER Čvorak v2.0b is given – BitCloud programming interface, sensor drivers, and networking application developed within the project Maslinet. Concept of over the air download scenario for Maslinet wireless sensor network is being proposed.

At the end activities related to further work are being proposed: testing of the hardware subsystems of the newly developed node, testing with sensors and networking application, power consumption testing and further work on software for remote programming of nodes.

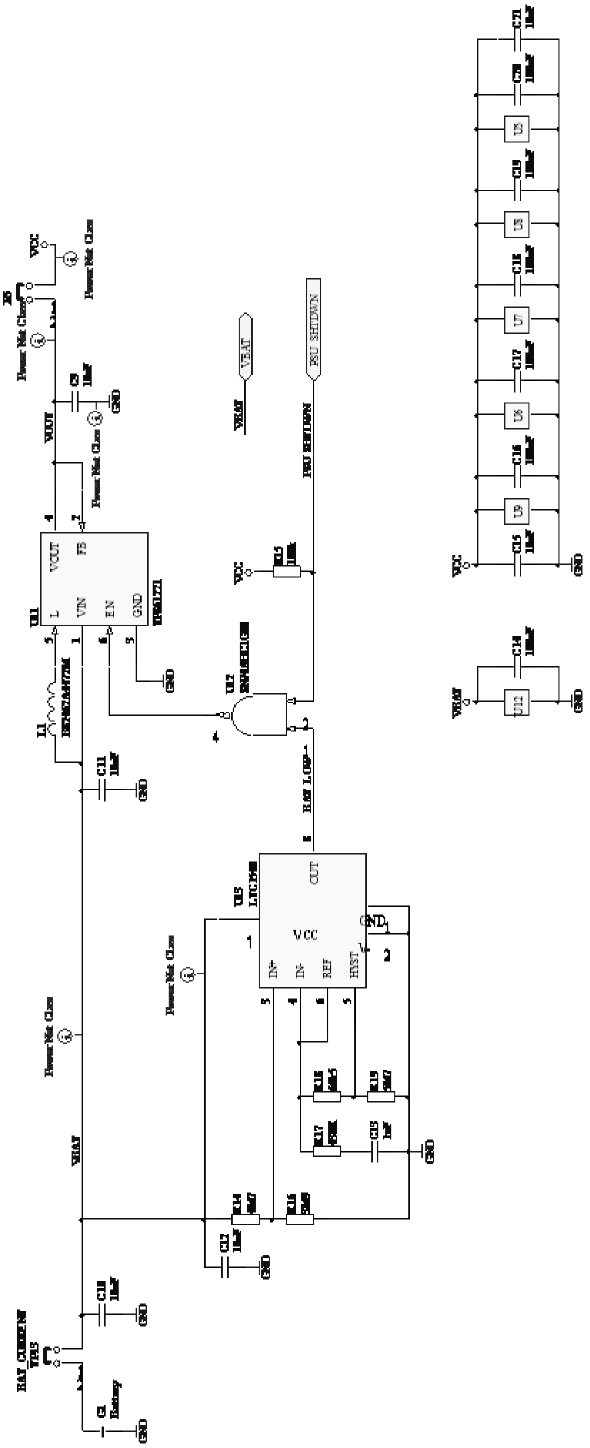
Keywords:

Wireless sensor node, mote, wireless sensor network, over the air programming

Prilog A: Shema bežičnog osjetilnog čvora FER Čorak v2.0b

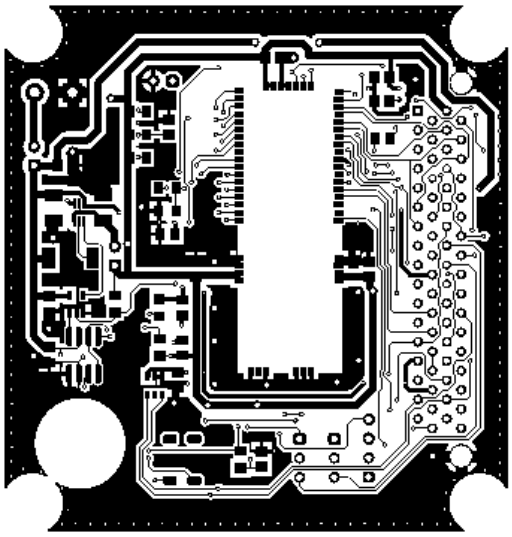


Prilog B: Shema napajanja osjetilnog čvorra FER Čvorak v2.0b

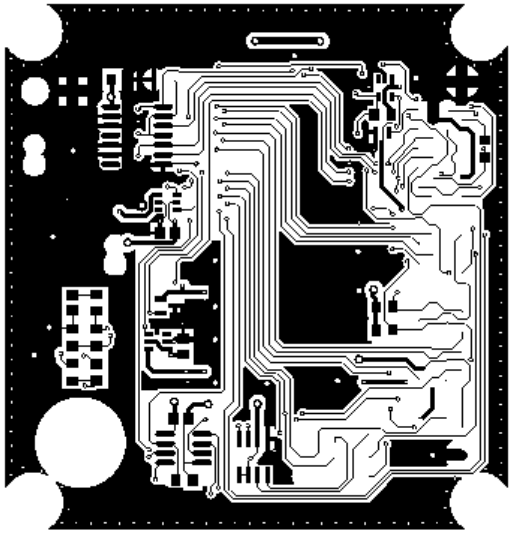


Prilog C: Nacrt tiskanih veza bežičnog osjetilnog čvora FER Čvorak v2.0b

Gornja strana:

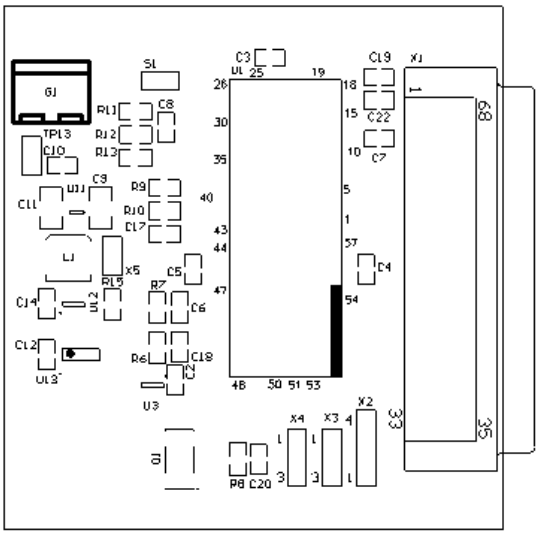


Donja strana:



Prilog D: Položajni nacrt komponenata bežičnog osjetilnog čvora FER Čvorak v2.0b

Gornja strana:



Donja strana: