Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621
DOI: https://doi.org/10.24867/11BE10Nikic

REALIZACIJA BOOTLOADER-A ZA BEŽIČNO AŽURIRANJE APLIKACIJA PUTEM NB-IOT KOMUNIKACIJE

IMPLEMENTATION OF BOOTLOADER FOR WIRELESS UPDATE OF APPLICATIONS USING *NB*-IOT COMMUNICATION

Vladimir Nikić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu je prikazana realizacija bežičnog ažuriranja aplikacije na mikrokontroleru. Rad sadrži opis aplikacija koje su potrebne za uspostavljanje NB-IoT komunikacije između mikrokontrolera i servera, preuzimanje korisničke aplikacije i njeno pokretanje nakon ažuriranja. Nabrojane i objašnjene su komponente koje su upotrebljene prilikom razvoja, realizacije i testiranja Bootloader-a.*

Ključne reči: *Bootloader, Bežično ažuriranje, NB-IoT*

Abstract – *This paper presents the implementation of wireless application update on a microcontroller. The paper contains a description of applications needed to establish NB-IoT communication between microcontroller and server, download the user application and execute it after the update. The components that are used during the development, implementation and testing of the Bootloader are listed and explained.*

Keywords: *Bootloader, Wireless update, NB-IoT*

1. UVOD

Internet stvari (eng. *Internet of Things*, skraćeno IoT) su objekti koji imaju ugrađenu elektroniku, često mikrokontrolere, koji sadrže softver i sa kojima su povezane različite vrste senzora.

Ovi objekti nisu usamljeni, već se nalaze u sklopu većeg sistema, gde su svi elementi međusobno povezani putem mreže. U okviru ovog sistema svaki objekat je kontrolisan ili nadgledan. Njegovi parametri se prosleđuju u mrežu gde ih drugi objekti koriste i prilagođavaju svoj rad drugima.

Kako je u praksi često potrebno da IoT uređaji budu postavljeni na lokacijama na kojima nije moguće obezbediti stalan izvor napajanja, potrebno je koristiti energetski najefikasnije uređaje, što su uobičajeno mikrokontroleri. Pored toga što na lokacijama nije obezbeđen stalan izvor napajanja, one mogu biti i fizički teško pristupačne. Imajući to u vidu i da se na njima izvršava samo jedna aplikacija, može se zaključiti da je zamena aplikacije komplikovan proces. Konačno, uzimajući u obzir da se u praksi često koriste sistemi koji imaju veći broj ovakvih uređaja, zamena aplikacije podrazumeva da se svakom uređaju treba pristupiti i ručno izvršiti ažuriranje.

Ovo zahteva veliki napor, kako fizički tako i vremenski, u slučajevima kad sistemi imaju nekoliko desetina ili stotina istih uređaja koji trebaju da se ažuriraju.

Navedeni problem predstavlja razlog razvoja uređaja koji imaju sposobnost bežičnog ažuriranja aplikacije, što predstavlja i temu ovog rada.

Kako bi se obezbedilo bežično ažuriranje aplikacije, potrebno je razviti tri aplikacije.

Prva aplikacija se nalazi na serveru i njen zadatak je da mikrokontroleru šalje novu verziju korisničke aplikacije.

Druga aplikacija je korisnička aplikacija čiji jedan deo je namenjen ostvarivanju komunikacije sa serverom, preuzimanje nove verzije i smeštanje u eksternu memoriju.

Konačno, treća aplikacija je Bootloader koja ima zadatak da trenutnu korisničku aplikaciju zameni novom.

U drugom delu rada dat je opis sistema u kojem su predstavljene komponente potrebne za realizaciju sistema. Treći deo rada obrađuje realizaciju gore navedenih aplikacija. Konačno, četvrti deo rada daje pregled mogućih unapređenja sistema.

2. OPIS SISTEMA

Sistem u kojem je realizovan Bootloader sadrži mikrokontroler *ATSAML21J18B* [1], NB-IoT modul *BC68* [2], fleš memoriju *SST25PF040C* [3] i server.

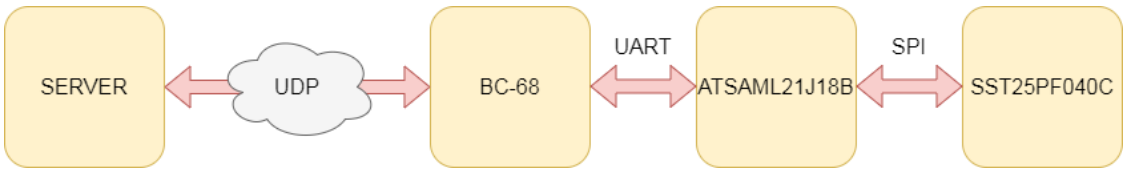
2.1. Mikrokontroler ATSAML21J18B

Centralnu komponentu sistema predstavlja *Microchip*-ov mikrokontroler ATSAML21J18B. Ovaj 32-bitni mikrokontroler ima osobinu vrlo male potrošnje prilikom rada na frekvenciji do 48 MHz zbog čega je idealan za primenu u IoT-u.

Za rad sa ovim mikrokontrolerom, *Microchip* je obezbedio besplatno softversko razvojno okruženje *Atmel Studio 7.0*. U okviru ovog okruženja nalazi se alat *Atmel Start* koji omogućava konfigurisanje mikrokontrolera, gde se na jednostavan način aktiviraju komponente, pinovi i povezuju izvori taktnog signala.

Takođe ovaj alat generiše kostur projekta na osnovu konfiguracije u kojem su uključene biblioteke potrebne za rad sa aktiviranim komponentama. Pored ovog alata, prisutni su kompajler i programator, koji putem *PowerDebugger*-a ažurira aplikaciju na mikrokontroleru.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Ivan Mezei, vanr.prof.



Slika 1. *Povezivanje sistema*

2.2. NB-IoT modul BC68

BC68 je modul kompanije *Quectel* koji omogućava komunikaciju putem *Narrowband IoT* (skraćeno NB-IoT). Kao i ATSAML21J18B, ovaj modul ima osobinu male potrošnje. Za uspostavljanje komunikacije ovaj modul koristi SIM karticu. BC68 omogućava komunikaciju putem širokog spektra protokola, uključujući UDP koji je upotrebljen u ovom radu.

2.3. Fleš memorija SST25PF040C

Fleš memorija SST25PF040C kapaciteta 4Mbit-a je komponenta koja omogućava skladištenje preuzete aplikacije pre njenog smeštanja u programsku memoriju. Ova memorija omogućava upis podataka na nivou strane (eng. *page*). Postoje mogućnosti običnog čitanja podataka kao i visoko performanskog čitanja. Konačno, fleš memorija ima mogućnost brisanja na nivou bloka (64KB) ili sektora (4KB).

2.4. Povezivanje sistema

Sistem je podeljen na dva dela, prvi deo predstavlja server na kojem se nalazi aplikacija i drugi deo koji čine komponente zadužene za preuzimanje aplikacije i njeno pokretanje. To su mikrokontroler, NB-IOT modul i fleš memorija. Ova dva dela povezana su putem NB-IOT mreže koristeći UDP protokol (eng. *Universal Datagram Protocol*).

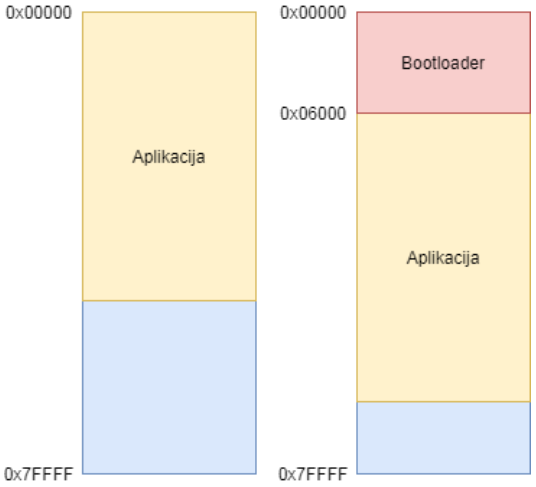
Drugi deo sistema se može nalaziti na istoj štampanoj ploči ili je realizovan pomoću diskretnih komponenti. Bez obzira na način implementacije, mikrokontroler je povezan sa perifierijama putem *SERCOM*-a koji predstavlja univerzalni serijski interfejs. Koriste se dva *SERCOM*-a, jedan za komunikaciju sa SST25PF040C konfigurisan da implementira Serijski periferni interfejs (eng. *Serial peripheral interface*, skraćeno SPI). Drugi *SERCOM* se koristi za komunikaciju sa BC68 i implementira Univerzalni asinhroni prijemnik i predajnik (eng. *Universal asynchronous reciever-transimitter*, skraćeno UART).

3. KONCEPT REŠENJA I REALIZACIJA

U narednim poglavljima dat je opis aplikacija koje su potrebne za implementaciju bežičnog ažuriranja.

3.1. Programska memorija mikrokontrolera

Mikrokontroler *ATSAML21J18B* poseduje 256KB programske fleš memorije u kojoj se nalazi aplikacija koja se izvršava na njemu. U običnom režimu rada mikrokontrolera, kad ne postoji Bootloader, čitav sadržaj memorije je na raspolaganju korisničkoj aplikaciji (prikazano u prvom delu slike 2). Međutim, ukoliko je na mikrokontroleru potrebno implementirati Bootloader, sadržaj programske memorije se deli na dve celine (prikazano u drugom delu slike 2). Uobičajno je da se prvi deo memorije dodeli Bootloader-u, dok se drugi deo memorije ostavlja na raspolaganju korisničkoj aplikaciji. Praksa je da se Bootloader-u dodeli minimalna količina prostora koja je potrebna da bi on bio implementiran, a da se što više prostora ostavi korisniku za implementaciju željenih funkcionalnosti.



Slika 2. *Programska memorija*

Kako su Bootloader i korisnička aplikacija dve aplikacije koje treba da se nađu u istoj programskoj memoriji, potrebno je podesiti početne adrese na kojima se svaka od njih treba naći. Ovo se postiže konfigurisanjem linkera, tako da Bootloader počinje od adrese 0x00000, dok se korisnička aplikacija nalazi na adresi 0x06000.

3.2. Drajver za SST25PF040C

Komunikacija sa SST25PF040C fleš memorijom se realizuje putem SPI interfejsa. Ovaj interfejs poseduje taktni signal SCK, signale za prenos podataka MOSI i MISO i konačno signal za selektovaje CE. Signal MOSI se najčešće koristi za slanje podataka od mikrokontrolera ka memoriji, dok se MISO signal koristi prenos podataka u suprotnom smeru.
Memorija se deli na strane koje predstavljaju sukcesivne celine od 256B, zatim sektore koji se sastoje od 4KB i konačno blokova koji se sastoje od 64KB memorijskog prostora. Ova podela je značajna zato što se brisanje podataka vrši na nivou nabrojanih celina.

Komande koje se koriste prilikom rada sa fleš memorijom su:

Read – komanda za čitanje podataka iz memorije. Mikrokontroler šalje kod komande i početnu adresu sa koje želi da učita neodređen broj podataka. Fleš šalje podatke redom od željene lokacije pa do kraja memorijskog prostora sve dok mikrokontroler ne postavi signal CE na vredost 1.

Write enable/disable – komande koje omogućavaju ili zabranjuju korišćenje komandi koje menjaju sadržaj memorijskih lokacija, to su Page-Program i Erase naredbe.

Page-Program – je komanda koja služi za upisivanje sadržaja u memoriju. Fleš memorija SST25PF040C omogućava upis podataka samo na nivou strane. Mikrokontroler šalje kod komande, zatim adresu na koju želi da upiše podatke i na kraju niz podataka koji treba da se upišu počevši od željene adrese.

Erase komande – su komande koje brišu sadržaj memorije. Postoje **Sector-Erase, Block-Erase** i **Chip-Erase** koje brišu sadržaj sektora, bloka i celog čipa.

3.3. Drajver za BC68

Za komunikaciju između modula BC68 i mikrokontrolera koristi se UART interfejs. Ovaj interfejs omogućava komunikaciju preko dva signala, Rx koji služi za prijem podataka i Tx koji služi za slanje podataka.

Upravljanje modulom BC68 vrši se slanjem poruka koje predstavljaju stringove oblika „AT+<cmd>␣", gde se na mesto <cmd > postavlja niz karaktera koji označava određenu komandu. Simbol ␣ se može zameniti sa „=?", „?", „=<...>" ili „" u zavisnosti od toga koja funkcionalnost treba da se realizuje.

Za potrebe bežičnog ažuriranja aplikacije, od BC68 se očekuje da omogući komunikaciju putem UDP protokola. UDP protokol nema implementiranu sinhronizaciju između pošiljaoca i primaoca poruke. Komunikacija se odvija asinhrono, tako što pošiljalac šalje podatke kad on to želi. Ovim protokolom se dobija na brzini, jer ne postoji nikakva procedura rukovanja (eng. *handshake*), međutim gubi se na pouzdanosti zbog čega je potrebno izvršiti proveru primljenih podataka.

Najznačajnije komande koje su korišćene za realizaciju UDP komunikacije su date u nastavku.

NSOCR – komanda koja služi za pravljenje socket-a. Kao parameter prima tip soketa koji je potrebno napraviti što je UDP, zatim port koji treba da sluša.

NSOCL - predstavlja komandu koja zatvara parametrom prosleđen socket.

NSOST – komanda koja služi za slanje podataka koristeći UDP protocol. Prosleđuju joj se kao parametri socket preko kojeg se podaci šalju, zatim broj podataka koji se šalju i konačno sami podaci.

NSORF – komanda čija je uloga da primi podatke preko UDP protokola. Prosleđuju joj se kao parametri socket i broj podataka koje treba preuzeti.

3.4. Serverska aplikacija

Serverska aplikacija je zadužena za slanje izvršnog koda. Ovaj proces podrazumeva učitavanje izvršnog koda iz .hex fajla, iz kog se određuju parametri koda, početna adresa na kojoj kod treba da se nađe u programskoj memoriji i dužina samog koda. Zatim se vrši preprocesiranje koda u vidu određivanja CRC32 vrednosti koja štiti integritet prilikom slanja. Kada je proces učitavanja i preprocesiranja završen, server otvara soket na koji mikrokontroler može da se poveže.

Komunikacija između mikrokontrolera i servera se vrši putem UDP protokola. Mikrokontroler šalje 7 bajtove koje server učitava i dekoduje. Prvi bajt poruke označava tip komande koja sledi. Koriste se dve komande:

Info - komanda koja zahteva od server da pošalje informacije o kodu. Prvi bajt ove komande ima vrednost 0x42, dok ostali bajtovi mogu imati proizvoljnu vrednst. Ukoliko server primi ovu komandu odgovara sa 12 bajtova, gde bajtovi 0-3 predstavljaju vrednost početne adrese memorije. Bajtovi 4-7 predstavljaju broj bajtova koji kod zauzima, odnosno dužinu koda. Konačno, bajtovi 8-11 sadrže vrednost CRC32 koja je izračunata za ceo kod. Sve vrednosti su predstavljene u *BigEndian* zapisu.

Data – je komanda koja zahteva od servera da mikrokontroleru pošalje kod zadate dužine počevši od određene adrese. Prvi bajt komande ima vrednost 0xD1, bajtovi 1-4 predstavljaju početnu adresu od koje kod treba da se šalje. Na kraju, poslednja dva bajta, 5-6, predstavljaju dužinu koda koji treba da se pošalje. Obe vrednosti su predstavljene u *BigEndian* zapisu. Server zatim šalje odgovarajuću količinu koda mikrokontroleru.

3.5. Korisnička aplikacija

Korisnička aplikacija se razvija tako da ima mogućnost da iskoristi postojeći Bootloader za potrebe bežičnog ažuriranja. Detaljan opis Bootloadera dat je u sekciji 3.6. Ova aplikacija je zadužena za preuzimanje nove verzije sa servera, proveru integriteta i konačno smeštanje u eksternu fleš memoriju. Nabrojane funkcionalnosti su implementirane u sklopu veće aplikacije, gde one predstavljaju samo jednu podcelinu namenjenu za ažuriranje.

3.5.1. Preuzimanje nove verzije korisničke aplikacije

Uzimajući u vidu da je za potrebe komunikacije sa serverom potrebno koristiti BC68 modul, potrebno ga je na početku aktivirati slanjem komande za inicijalizaciju. Zatim je potrebno poslati komandu za otvaranje sockete i povezivanje na server. Nakon toga se šalje komanda koja zahteva informacije o kodu. Mikrokontroler zatim čeka da stigne odgovor od servera koji učitava iz BC68 i parsira ga tako da dobija vrednosti početne adrese koda, dužine i CRC32 vrednost. Kako nije moguće preuzti ceo kod, zato što je za njegovo skladištenje potrebno onoliko memorije koliko on ima bajtova, što nije racionalno korišćenje memorije, preuzimanje koda se vrši u stranama od 256 bajtova, što odgovara mogućnostima upisa u fleš memoriju.

Mikrokontroler zatim sukcesivno šalje serveru Data komandu počevši od početne adrese, svaki put povećavajući za 256 bajtova ili, ukoliko je na redu poslednji paket, onoliko koliko je preostalo. Kod koji dobije kao odgovor privremeno skladišti u SRAM memoriji. Nakon prijema svih bajtova, vrši se njihovo prebacivanje u fleš memoriju koristeći komandu Page-Program. Kod se u fleš memoriju smešta tako da jedna strana sledi drugu, tj. ne postoje praznine između dve strane u kojima se skladišti kod. Takođe potrebno je ispoštovati redosled koda u memoriji.

3.5.2. Provera integriteta korisničke aplikacije

Provera integriteta podrazumeva proveru validnosti koda nakon njegovog preuzimanja i smeštanja u memoriju. Potrebno je detektovati da li je došlo do promene u kodu prilikom prenošenja bežičnim putem. Ovo se može postići različitim metodama. Metoda koja je odabrana u ovom radu je CRC (eng. *Cyclic Redundancy Check*).

Ona se zasniva na ostatku pri deljenju sadržaja poruke odgovarajućim polinomom. Korišćen je CRC32 kod koji se zasniva na polinomu 32-og reda. Standardan oblik polinoma dat je u nastavku.

$$P_n(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad (1)$$

Algoritam za izračunavanje CRC32 se izvršava prvo na serveru. Server svoju dobijenu vrednost šalje mikrokontroleru kao odgovor na Info komandu. Nakon što mikrokontroler smesti ceo kod u fleš memoriju, počinje da ga učitava bajt po bajt i prosleđuje ga ugrađenoj komponenti mikrokontrolera čija je funkcionalnost računanje CRC32. Nakon što izračuna vrednost CRC32 posle prosleđivanja poslednjeg bajta koda, upoređuje izračunatu vrednost sa onom koju je dobio od servera i utvrđuje da li se poklapaju. Ukoliko se ne poklapaju, kod nije validan i potrebno je ponovo preuzeti i informacije o kodu i sam kod jer se ne može utvrditi da li je došlo do greške u prenosu koda ili informacija. Ukoliko se poklapaju, završava se deo korisniče aplikacije namenjen za preuzimanje nove verzije i ona nastavlja svoje izvršavanje.

3.6. Bootloader

Bootloader predstavlja aplikaciju koja je zadužena za ažuriranje i pokretanje korisničke aplikacije. Kako se on nalazi na početnoj adresi programske memorije, izvršava se odmah nakon dovođenja napona napajanja na mikrokontroler ili resetovanja mikrokontrolera.

3.6.1. Pokretanje Bootloader-a

Kao što je već rečeno, Bootloader se može pokrenuti resetovanjem ili dovođenjem napajanja na mikrokontroler. Resetovanje se vrši koristeći taster koji je povezan na pin namenjen za reset mikrokontrolera. Potrebno ga je povezati na takav način da u slobodnom režimu ne dovodi do reseta. Pritiskom ovog tastera pokreće se Bootloader.

Potrebno je koristiti i drugi taster koji je povezan na jedan od GPIO (eng. *General Purpose Input Output*) pinova. Ovaj pin mora biti konfigurisan kao ulazni digitalni pin. Bootloader koristi ovaj taster za proveru da li pri njegovom pokretanju treba da ažurira aplikaciju ili samo da je pokrene.

3.6.2. Zamena korisničke aplikacije

Za upisivanje novih vrednosti u programskoj fleš memoriji neophodno je koristiti drajver koji se aktivira kao posebna komponenta. Ovaj drajver poseduje funkcije upisa i čitanja, kao i provera veličine strane programske fleš memorije.

Nakon učitavanja veličine strane programske fleš memorije, ova vrednost se koristi za učitavanje te količine podataka iz eksterne fleš memorije. Ovi podaci se zatim upisuju u internu programsku memoriju počevši od adrese 0x6000. Svaki put se adrese sa koje se učitavaju podaci iz eksterne memorije i na koju se upisuju u programskoj memoriju inkrementuju za vrenost veličine strane programske memorije. Ovo se radi sve dok se ceo kod ne prabaci iz eksterne u programsku memoriju.

3.6.3. Pokretanje korisničke aplikacije

Na kraju potrebno je pokrenuti kod. Prvo je potrebno postaviti *stack pointer* na vrednost početne adrese aplikacionog dela programske memorije, što je u ovom slučaju 0x6000.

Takođe neophodno je postaviti početnu adresu tabele vektora prekida. Konačno potrebno je proslediti novu vrednost programskom brojaču (eng. *Program counter*) na vrednost za 4 veću od početne adrese aplikacionog dela, što je 0x6004.

Nakon ovoga izvršava se korisnička aplikacija koja da bi se ponovo ažurirala mora imati implementiranu funkcionalnost namenjenu za to. U suprotnom nije moguće ponovo ažurirati aplikaciju novom verzijom. Ove funkcionalnosti se najčeše realizuju korišćenjem asemblerskih naredbi.

4. ZAKLJUČAK

U ovom radu prikazan je osnovni način za bežično ažuriranje aplikacije na mikrokontroleru. Objašnjene su komponente potrebne za realizaciju i testiranje. Opisan je izgled i funkcionalnost sistema, kao i softvera koji je potreban da se nalazi na serverskoj i klijentskoj strani. Objašnjeni su protokoli i komande koje služe za komunikaciju između delova sistema, kao i metod zaštite pri prenosu podataka bežičnim putem.

Kako je ovo osnovni način za realizaciju, postoje mnoga unapređenja koja je moguće ostvariti. Neke od mogućnosti su da klijent proverava sopstvenu verziju koda i onog koji se nalazi na serveru i na osnovu toga određuje da li je potrebno ažuriranje. Zatim postoji mogućnost unapređenja bezbednosti prenosa u vidu kriptovanja podataka koristeći TLS protokol (eng. *Transport Layer Security*). Konačno, moguće je smanjiti broj podataka koji se prenose od servera ka mikrokontroleru ako se način ažuriranja implementira kao diferencijalnoažuriranje.

5. LITERATURA

[1] *Microchip* ATSAML21J18B Datasheet: http://ww1.microchip.com/downloads/en/DeviceDoc/SAM_L21_Family_DataSheet_DS60001477C.pdf (pristupljeno u septembru 2020.)

[2] *Quectel* NB68 Datasheet: https://www.quectel.com/UploadFile/Product/Quectel_BC68_NB-IoT_Specification_V1.8.pdf (pristupljeno u septembru 2020.)

[3] *Microchip* SST25PF040C Datasheet: https://eu.mouser.com/datasheet/2/268/20005397B-967551.pdf (pristupljeno u septembru 2020.)

Kratka biografija:

Vladimir Nikić rođen je u Novom Sadu 1996. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Embeded sistemi i algoritmi odbranio je 2019.god.
kontakt: nikic.vladimir@gmail.com