STM32 biblioteke za EEPROM i NTAG RFID integrirani sklop

Horvat, Matej

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

STM32 biblioteke za EEPROM i NTAG RFID integrirani sklop

Diplomski rad

Matej Horvat

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 01.07.2021.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Matej Horvat
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1057R, 06.10.2019.
OIB studenta:	14994396915
Mentor:	Izv. prof. dr. sc. Tomislav Matić
Sumentor:	Josip Zidar
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Ivan Aleksi
Član Povjerenstva 1:	Izv. prof. dr. sc. Tomislav Matić
Član Povjerenstva 2:	Doc.dr.sc. Ivan Vidović
Naslov diplomskog rada:	STM32 biblioteke za EEPROM i NTAG RFID integrirani sklop
Znanstvena grana rada:	**Arhitektura računalnih sustava (zn. polje računarstvo)**
Zadatak diplomskog rada:	U radu je potrebno razviti i testirati STM32 biblioteku za EEPROM integrirani sklop i NTAG RFID integrirani sklop. Oba sklopa komuniciraju pomoći I2C sabirnice. Sumentor: Josip Zidar Tema rezervirana za: Matej Horvat
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	01.07.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 21.07.2021.

Ime i prezime studenta:	Matej Horvat
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1057R, 06.10.2019.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: STM32 biblioteke za EEPROM i NTAG RFID integrirani sklop**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Tomislav Matić i sumentora Josip Zidar moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Tema diplomskog rada je razvoj i testiranje STM32 biblioteka za EEPROM (engl. *Electricaly Erasable Programmable Read-Only Memory*) integrirani sklop i NTAG (engl. *Near Field Communication Tag*) integrirani sklop. EEPROM je električno izbrisiva programibilna memorija samo za čitanje, koja se može brisati i ponovno programirati primjenom električnog napona većeg od normalnog. Također, EEPROM ima ograničeni broj brisanja i reprogramiranja te je važno tu činjenicu uzeti u obzir [1]. NTAG predstavlja elektronički uređaj koji je temeljen na RFID (engl. *Radio-frequency Identification*) tehnologiji. Sastoji se od integriranog sklopa i antene te može biti ugrađen u druge uređaje kao što su kartica, boca, privjesak ili može biti izrađen kao samostalna naljepnica [2]. Glavna svrha im je brza i jednostavna izmjena digitalnih informacija. NFC tehnologija je dio, odnosno podskup RFID tehnologije [3].

EEPROM biblioteka se sastoji od raznih funkcija za čitanje i pisanje u memoriju, funkcije za vremensku odgodu, koju je potrebno koristiti zbog ciklusa vremena zapisivanja u memoriju (engl. *write cycle time*). NTAG biblioteka sastoji se od raznih funkcija za čitanje i pisanje u EEPROM te SRAM (engl. *static random access memory*), čitanje ID-a (engl. *identity document*) uređaja, postavljanja statičkih i dinamičkih bitova za zaključavanje memorije, funkcija za promjenu lozinke i mogućnosti pristupa. Osim navedenih funkcija, sastoji se i od većeg broja funkcija za čitanje i postavljanje konfiguracijskih i sesijskih registara te *capability containera*. Za svaku pojedinu funkciju u zaglavnim dokumentima obje biblioteke postoje detaljni opisi na koji način se koriste i opis svakog parametra funkcije. Obje biblioteke koriste I^2C (engl. *Inter-Integrated Circuit*) sabirnicu za komunikaciju sa sklopovima. I^2C je dvosmjerni serijski komunikacijski protokol u kojem se podatci prenose bit po bit pomoću jedne žice (SDA), dok se za sinkronizaciju prijenosa podataka koristi SCL žica koja prenosi signal takta.

2. PREGLED PODRUČJA TEME

EEPROM biblioteke pisane su za interakciju raznih vrsta mikroupravljača s EEPROM-om, odnosno najčešće čitanje te zapisivanje podataka u memoriju korištenjem odabranih mikroupravljača. Razni mikroupravljači te platforme temeljene na mikroupravljačima za koje već postoje EEPROM biblioteke obuhvaćaju Arduino [4], PIC16, PIC18 [5], STM32 [6], ESP8266 [7], NXP ARM [8]. Razlika između već postojeće biblioteke za STM32 koja vrši komunikaciju s EEPROM-om je ta da prethodno navedena sadrži samo ograničeni broj funkcija te je bilo potrebno implementirati više funkcija za različite načine čitanja i zapisivanja memorije te za različiti broj bajtova koje je potrebno pročitati ili zapisati. Uz to, bitno je napomenuti da različiti EEPROM-i imaju drugačiji raspored memorije (npr. broj stranica i bajtova) te se samo adresiranje memorije vrši na različite načine, što je i ovdje slučaj. Testiranje EEPROM biblioteka općenito se vrši izvođenjem svake pojedine funkcije i isprobavanjem rubnih slučajeva.

Biblioteke za NTAG integrirani sklop koriste se za komunikaciju mikroupravljača te platforma koje su temeljene na mikroupravljačima s NTAG integriranim sklopovima. Pomoću njih postavljaju se konfiguracijski bitovi koji utječu na to na koji način će raditi sam integrirani sklop, odnosno koje mogućnosti će mu biti omogućene i onemogućene. Također, moguće je spremati željene podatke u internu memoriju integriranog sklopa. Biblioteke za NTAG integrirani sklop kreirane su za Arduino [9] te NXP ARM [10]. Nije postojala izvedba NTAG biblioteke za STM32 do ove implementacije. Također, u usporedbi s prethodno navedenim bibliotekama koje su izvedene za Arduino i NXP, dodan je veći broj funkcija koje obuhvaćaju različite načine čitanja te zapisivanja u memoriju (EEPROM i SRAM), konfiguraciju sesijskih i konfiguracijskih registara, postavljanje lozinke, konfiguriranje dinamičkih i statičkih bitova za zaključavanje memorije. Osim toga, omogućena je mogućnost čitanja ID-a uređaja, postavljanja željene mogućnosti pristupa memoriji te konfiguracije *capability containera*. Testiranje NTAG biblioteke također se vrši izvođenjem svake pojedine funkcije i isprobavanjem rubnih slučajeva. Uz to, za testiranje biblioteke izvedene za NXP ARM preporučeno je korištenje NTAG I^2C demo aplikacije pomoću koje je moguće paljenje te gašenje svjetlećih dioda (narančaste, plave, zelene), temperaturnog senzora, LCD-a (engl. *Liquid Crystal Display*), slanje i čitanje NDEF (engl. *NFC data exchange format*) poruka iz memorije, testiranje *pass-through* načina rada, čitanje i zapisivanje u interni EEPROM te čitanje i konfiguriranje registara na NXP ARM mikroupravljaču [11]. Većina od tih funkcionalnosti korištene su za testiranje NTAG biblioteke napisane za STM32 mikroupravljač. Funkcionalnosti aplikacije koje nisu korištene su paljenje i gašenje svjetlećih

dioda (narančaste, plave, zelene), temperaturnog senzora te LCD-a iz razloga što mikroupravljač koji je korišten pri razvoju biblioteke ne sadrži navedene funkcionalnosti.

3. NFC, EEPROM I ARM TEHNOLOGIJE

3.1. NFC

NFC je tehnologija bežičnog prijenosa digitalnih podataka koja se koristi na udaljenostima manjim od 10 centimetara. Za uspješnu izmjenu podataka NFC-om, potrebno je imati aktivni uređaj i pasivni NFC uređaj. Aktivni NFC uređaj je odgovoran za stvaranje RF (engl. *radio frequency*) polja, koje koristi frekvenciju od 13.56 MHz te napaja pasivni NFC uređaj. NFC aktivni uređaj najčešće je pametni telefon s ugrađenom NFC tehnologijom, a može biti i pametni sat, odnosno pojedini specijalizirani uređaj za jedinstvenu namjenu. Aktivni uređaj inicira prijenos podataka (slanje i čitanje). Također, važan dio ove tehnologije predstavljaju NFC pametne oznake (engl. *smart tag*). Sastoje se od interne memorije, integriranog sklopa te antene. Podatci se mogu čitati i zapisivati u memoriju pametnih oznaka, koja je veličine od 96 do 4096 bajtova, ovisno o tipu oznake. Pametne oznake su pasivni NFC uređaji, što znači da ne trebaju izvor napajanja te sadrže podatke koje aktivni NFC uređaji mogu čitati iz njihove memorije. Pametne oznake same ne mogu inicirati prijenos podataka. Rade na način da crpe energiju iz uređaja koji ih čita, uz pomoć magnetske indukcije. Bežično punjenje je jedan od glavnih prednosti ove tehnologije [3].

RFID radi na tri frekvencijska raspona:

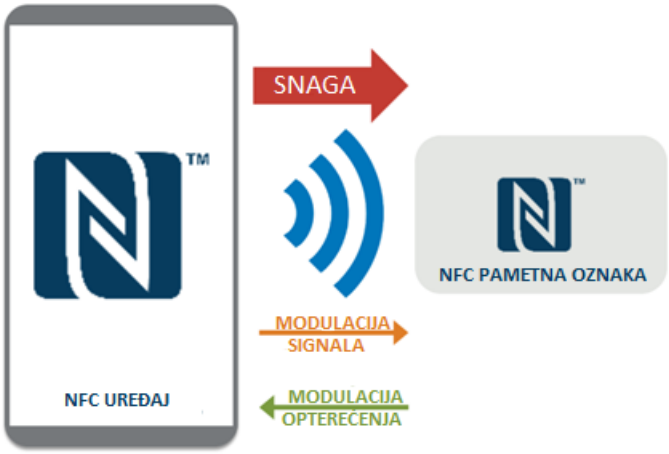
- LF (engl. *low frequency*), koji radi u rasponu 125-134 kHz
- HF (engl. *high frequency*), koji radi na 13.56 MHz
- UHF (engl. *ultra high frequency*) koji radi u rasponu 856-960 MHz.

NFC je dio HF (visokofrekventnih) RFID tehnologija [12]. Dva glavna standarda NFC tehnologije su ISO/IEC 14443 i ISO/IEC 18000-3. Prvi standard propisuje ID kartice koje se koriste za spremanje podataka koje se nalaze u NFC pametnim oznakama. Drugi standard određuje vrstu RFID komunikacije koju koriste NFC uređaji. NFC se može koristiti za izvršavanje transakcija pomoću beskontaktnih kartica, za otključavanje i zaključavanje vrata, za dijeljenje fotografija i punjenje IoT (engl. *Internet of Things*) uređaja [13].

3.1.1. Postupak prijenosa podataka NFC-om

Kada se NFC uređaj s kojim se vrši čitanje ili zapisivanje približi pametnoj oznaci, započinje prijenos podataka. NFC uređaj šalje podatke NFC pametnoj oznaci na način da vrši modulaciju signala RF polja te prima podatke od pametne oznake prepoznavanjem modulacije

opterećenja koje generira pametna oznaka [14]. Cijeli postupak interakcije NFC uređaja i NFC pametne oznake prikazan je na slici 3.1.



Sl. 3.1. Interakcija NFC uređaja i NFC pametne oznake [14].

3.1.2. Vrste NFC pametnih oznaka

Prva vrsta NFC pametne oznake (engl. *type 1*) može imati veličinu EEPROM memorije u rasponu od 96 bajtova do 2 kilobajta. Najveća brzina prijenosa informacija između aktivnih uređaja i ove vrste NFC oznake je 106 kilobita po sekundi. Ovaj tip oznake ne podržava mehanizam protiv kolizije prilikom prijenosa podatka. Druga vrsta NFC pametne oznake (engl. *type 2*) slična je kao prva, osim što može imati veličinu EEPROM memorije u rasponu od 48 bajtova do 2 kilobajta, te prilikom prijenosa podataka ima mehanizam protiv kolizije. Treća vrsta se u odnosu na prethodne dvije razlikuje u tome što joj je veličina EEPROM memorije 2 kilobajta, a brzina prijenosa može biti 212 ili 424 kilobita po sekundi. Također, ima mehanizam protiv kolizije. Četvrta vrsta NFC pametne oznake sadrži do 32 kilobajta EEPROM memorije, a brzina prijenosa podataka može biti 106, 212 ili 424 kilobita po sekundi. Također, sadrži mehanizam protiv kolizije. Peta vrsta NFC pametne oznake sadrži više od 64 kilobajta EEPROM memorije te podržava prijenos podataka od 26.46 kilobita po sekundi kao i mehanizam protiv kolizije. Za razliku od četvrte i pete vrste NFC oznake kod kojih je memorija samo za čitanje, kod prve tri vrste novi podatci mogu biti zapisani u memoriju [13]. U tablici 3.1. prikazane su karakteristike različitih vrsta NFC pametnih oznaka.

Tablica 3.1. Vrste NFC pametnih oznaka

Vrsta NFC pametne oznake	Veličina EEPROM-a	Brzina prijenosa podataka	Mehanizam protiv kolizije	Cijena
tip 1	96 B – 2 kB	106 kB/s	NE	niska
tip 2	48 B – 2 kB	106 kB/s	DA	niska
tip 3	2 kB	212 ili 424 kB/s	DA	visoka
tip 4	< 32 kB	106, 212 ili 424 kB/s	DA	srednja ili visoka
tip 5	> 64 kB	26.46 kB/s	DA	niska

3.1.3. Načini rada NFC tehnologije

Osim osnovnog načina rada, koji koristi NFC aktivni uređaj i NFC pasivni uređaj za interakciju i prijenos podataka, postoji *peer-to-peer* način rada, u kojem se izmjena podataka putem NFC-a vrši između dva aktivna NFC uređaja. Način rada bežičnog napajanja omogućuje bežični prijenos snage do 1 W. Na ovaj način, mogu se napajati IoT uređaji kao što su pametni sat, Bluetooth bežične slušalice i slično. U načinu emulacije kartice, aktivni NFC uređaj ponaša se kao beskontaktna kartica koja vrši interakciju s drugim NFC aktivnim uređajem. Glavne uporabe ovog načina rada su novčane transakcije te emulacija beskontaktnih karti za javni prijevoz [14]. Na slici 3.2. prikazani su različiti načini rada NFC tehnologija.

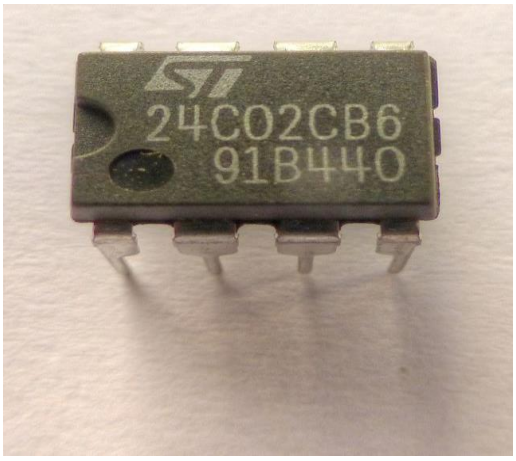
Sl. 3.2. Prikaz različitih načina rada NFC tehnologije [14].

3.1.4. Sigurnost NFC tehnologije

Prednost NFC tehnologije je ta što se prijenos podataka vrši na vrlo maloj udaljenosti, koja ne može biti veća od 10 centimetara te to otežava neželjene napade na izmjenu podataka između NFC aktivnog i NFC pasivnog uređaja. Iz tog razloga, sigurnost izmjene podataka NFC tehnologijom je puno veća u odnosu na ostale bežične komunikacijske protokole. Također, u načinu rada *peer-to-peer* prijenosa podataka postoji mehanizam za šifriranje svih informacija koje se razmjenjuju između dva uređaja te to još dodatno otežava neželjene napade na komunikaciju koja se odvija. Uz to, u načinu emulacije kartice prijenos podataka dodatno se može osigurati na dva načina. Prvi način je korištenje SIM (engl. *subscriber identity module*) kartice s omogućenim NFC-om, dok je drugi način korištenje sigurnosnog integriranog sklopa koji je ugrađen u NFC aktivni uređaj. U oba načina informacije primljene od uređaja koji emulira beskontaktnu karticu prosljeđuju se do sigurnosnih elemenata (SIM kartica ili sigurnosni integrirani sklop) za daljnje procesiranje. Na taj način je omogućena visoka razina sigurnosti transakcija. U klasičnom načinu rada, ukoliko se želi povećati sigurnost transakcije, programer, odnosno razvojni inženjer može implementirati razne kriptografske metode vlastitog izbora [14].

3.2. EEPROM

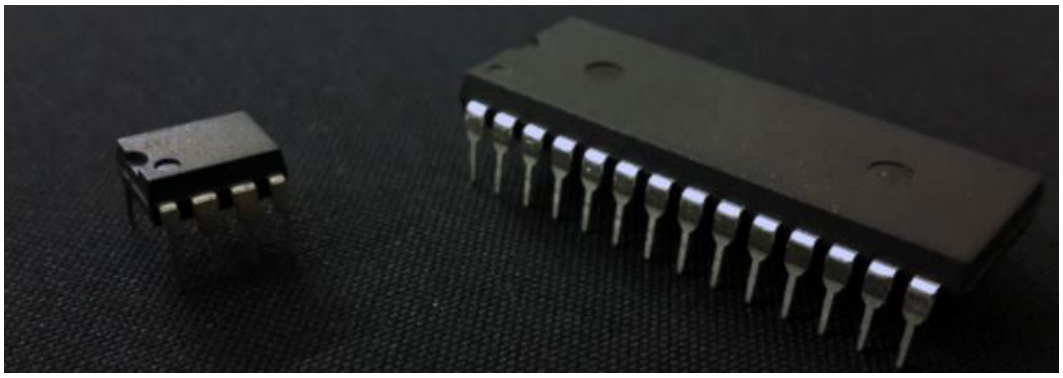
EEPROM je vrsta poluvodičkog memorijskog integriranog sklopa koji se koristi za pohranu podataka. Memorija u EEPROM-u je trajna (engl. *non-volatile*), što znači da se neće obrisati ukoliko izgubi izvor napajanja. Iz EEPROM-a je moguće čitati podatke, brisati ih te zapisivati. Za brisanje podataka potreban je električni napon veće vrijednosti od one koju integrirani sklop uobičajeno koristi. Čitanje i zapisivanje podataka u EEPROM izvršava se puno manjom brzinom u odnosu na čitanje i zapisivanje podataka u RAM. To treba uzeti u obzir kada se koristi EEPROM, iz razloga da ciklusi čitanja i zapisivanja podataka ne usporavaju sustav u koji je EEPROM integriran. Svaki bajt podataka pojedinačno se može reprogramirati ili mu se vrijednost može obrisati. Memorijska ćelija EEPROM-a sastoji se od dva tranzistora s efektom polja. Jedan od ta dva tranzistora je tranzistor za pohranu te se naziva plutajućim vratima (engl. *floating gate*). Elektroni se skladište u tom tranzistoru, a ovisno o njihovoj odsutnosti ili prisutnosti određuje se vrijednost memorijske ćelije. Drugi tranzistor u memorijskoj ćeliji zove se pristupni tranzistor te je potreban za uspješno izvođenje operacija unutar same ćelije [14, 15]. Na slici 3.3. prikazan je EEPROM integrirani sklop.



Sl. 3.3. Prikaz EEPROM integriranog sklopa.

3.2.1. Vrste EEPROM memorije

Postoje dvije vrste EEPROM memorije: serijska i paralelna. Način rada serijske EEPROM memorije je složen te je prijenos podataka unutar te vrste memorije serijski. Također, sadrži manje nožica te i raspored memorijskih ćelija je manje gust naspram paralelnog EEPROM-a. Prednost u odnosu na paralelni EEPROM je ta da su u pravilu serijske izvedenice jeftinije. Serijski EEPROM može koristiti više različitih tipova sučelja: I^2C, SPI, Microwire, UNI/O, 1-Wire. Ova sučelja za izvođenje operacija koriste između jedan i četiri upravljačka signala. Protokol serijskog EEPROM-a sastoji se od 3 faze: OP-Code faza, adresna faza te podatkovna faza. Bitovi unutar OP-Code faze označavaju koji će se tip operacije izvršavati, bitovi unutar adresne faze označavaju nad kojim će se bitovima operacija izvršavati te se na kraju u podatkovnoj fazi ta operacija izvršava. Jedna od prednosti serijskih EEPROM-a je ta da su u pravilu vrlo malih dimenzija. Paralelna EEPROM memorija radi puno brže u odnosu na serijsku izvedenicu. Uz to, raspored memorijskih ćelija je gušći i integrirani sklop ima veću pouzdanost u odnosu na serijski EEPROM. U pravilu sadrži 8-bitnu sabirnicu. Iz tog razloga operacije nad memorijom su brže i jednostavnije u odnosu na serijski. Neki mikroupravljači sadrže integrirane paralelne EEPROM-e radi internog skladištenja podataka. Nedostatak paralelnog EEPROM-a je taj što je puno skuplji u odnosu na serijski i većih je dimenzija zbog većeg broja nožica. Iz tih razloga upotreba paralelne vrste nije česta [15, 16]. Na slici 3.4. prikazani su serijski i paralelni EEPROM.



Sl. 3.4. Usporedba serijskog (lijevo) i paralelnog (desno) EEPROM-a.

3.2.2. Greške u radu EEPROM memorije

Prvu moguću grešku u radu EEPROM memorije predstavlja vrijeme zadržavanja podataka. Do te greške dolazi iz razloga što postoji mogućnost da elektroni koji su ubrizgani u plutajuća vrata ponekad mogu proći kroz izolator. Do drifta elektrona dolazi kao posljedica što ne postoji savršeni izolator. Na taj način dolazi do gubitka naboja i brisanja podataka. U pravilu proizvođači EEPROM-a daju garanciju zadržavanja podataka u vremenu trajanja oko deset godina [15]. Uz to, temperatura okoline igra ulogu u vremenu zadržavanja podataka. Druga moguća greška u radu EEPROM memorije je izdržljivost podataka. Prilikom ponovnog zapisivanja u EEPROM, dolazi do oksidacije plutajućih vrata prilikom čega se neželjeni elektroni nakupljaju u memorijskoj ćeliji. Nakon toga, elektroni koji se trebaju nalaziti u memorijskoj ćeliji i neželjeni elektroni zajedno počinju tvoriti električno polje. To dovodi do stanja u kojem u plutajućim vratima nema elektrona, a i dalje se nalazi zaostalo neželjeno električno polje. Zatim, što se više neželjenih elektrona nastavi nakupljati, neželjeno električno polje jača te u jednom trenutku memorijska ćelija ostaje u trajnom stanju vrijednosti logičke jedinice. U pravilu, iz tog razloga proizvođač EEPROM-a ograničava broj mogućih reprogramiranja integriranog sklopa na oko milijun [15, 16].

3.3. ARM

ARM (engl. *Advanced RISC Machines*) predstavlja obitelj procesorskih arhitektura smanjenog skupa naredbi, RISC (engl. *reduced instruction set computer*), konfiguriranih za različita okruženja. Postoje 32-bitne i 64-bitne verzije ARM-ovih RISC višejezgrenih procesora. Karakteristika RISC procesora je ta da izvodi manji broj naredbi u odnosu na CISC (engl. *complex instruction set computer*) procesore te im to omogućuje da rade na višim brzinama, izvodeći više milijuna instrukcija po sekundi. Također, prilikom rada koriste mnogo manje snage nego CISC

procesori, manje se zagrijavaju, sadrže mali broj tranzistora te su uobičajeno malih dimenzija. ARM procesore karakteriziraju:

- 32-bitna *load/store* arhitektura (nema direktne manipulacije memorijskog sadržaja)
- izvođenje većine naredbi u jednom ciklusu
- ortogonalan niz instrukcija, štednja energije
- podrška vizualizacije sklopovlja
- skalabilne visoke performanse
- višeprocesorski sustavi
- brza memorija i upravljanje memorijom
- korištenje cjevovoda
- veliki broj registara
- Thumb-2 tehnologija.

S obzirom što ARM karakterizira 32-bitna *load/store* arhitektura, jedini mogući pristup memoriji je kada se podatci učitavaju iz memorije ili pohranjuju u memoriju. Koriste se u raznim računalnim i ugradbenim računalnim sustavima, pametnim telefonima, tabletima, laptopima, stolnim računalima, serverima i superračunalima. Jednostavan dizajn ARM procesora omogućuje učinkovito višejezgreno procesiranje i pisanje jednostavnijeg koda programerima. ARM je najraširenija te najčešće korištena računalna arhitektura danas te se u svijetu nalazi oko 180 milijardi ARM integriranih sklopova [17, 18].

3.3.1. Načini rada ARM-a

ARM ima sedam osnovnih načina rada: SVC, FIQ, IRQ, Abort, Undef, System te User. SVC se izvodi na reset sustava ili kada su posebne SVC naredbe pozvane. FIQ se koristi kada je pozvan prekid (engl. *interrupt*) visokog prioriteta. IRQ se upotrebljava kada je pozvan prekid normalnog prioriteta. Abort se koristi prilikom pogreške u radu s memorijom. Undef se koristi kada su pozvane nepoznate naredbe. System se poziva ukoliko se u privilegiranom načinu rada procesora koriste isti registri kao u User načinu rada. User način rada je najčešće korišten način rada, u upotrebi je kada se izvode korisnički te zadatci (engl. *tasks*) operacijskog sustava. Od prethodno navedenih sedam načina rada, svi su privilegirani, osim User načina rada [19].

3.3.2. ARM uređaji

ARM sadrži određene module, koji uključuju [20]:

- GPIO (engl. *general purpose input output*)

- I^2C
- UART (engl. *universal asynchronous receiver-transmitter*)
- DMA (engl. *direct memory access*)
- brojače vremena (engl. *timers*)
- ADC (engl. *analog-to-digital converter*)
- DAC (engl. *digital-to-analog converter*)
- SPI (engl. *serial peripheral interface*)
- USB (engl. *universal serial bus*)
- DCMI (engl. *digital camera memory interface*)
- CAN (engl. *controller area network*)
- LCD-TFT
- RTC (engl. *real time clock*)
- SAI (engl. *serial audio interface*).

Za izradu biblioteka, korišteni su moduli GPIO, I^2C te brojač vremena. Jedna GPIO nožica koristi se za FD pin, koji služi za signaliziranje NFC događaja i sinkronizaciju *pass-through* načina rada NTAG I^2C plus 2K (NT3H2211). I^2C služi za komunikaciju u obje biblioteke, između mikroupravljača te EEPROM-a (CAT24M01), odnosno mikroupravljača i NTAG I^2C plus 2K (NT3H2211). Brojač vremena korišten je u obje biblioteke za generiranje vremenske odgode, zbog generiranja potrebnog vremena zapisivanja u memoriju kod CAT24M01 te NT3H2211.

3.3.3. Verzije ARM-a

Osnovna podjela ARM procesora je na Cortex-A, Cortex-R te Cortex-M procesore.

Cortex-A procesori upotrebljavani su za sustave visokih performansi, koji su najčešće vezani za Android te Linux operacijske sustave. Omogućavaju visoke performanse uz visoku energetsku učinkovitost. Primjenjuju se za rad slušalica, pametnih telefona, računala, opreme za emitiranje te mrežne opreme. Neke od serija Cortex-A procesora su A5, A7, A8, A9, A12, A15, A17, A35, A53. A5 serija omogućava rad s malom potrošnjom energije uz dobre performanse. A7 serija ima sličnu upotrebu energije uz 20 posto veće performanse. Često se koristi u tabletima i pametnim telefonima. A15, A17, A35 i A53 koriste se u slušalicama visoke kvalitete.

Cortex-R serija koristi se u aplikacijama stvarnog vremena, sustavima visokog stupnja sigurnosti, računalnim sustavima stvarnog vremena koji imaju upotrebu u automobilima, zrakoplovima, medicini, vojsci te tehnologijama na strani poslužitelja. Neke od serija Cortex-R

procesora su R4, R5, R7, R8, R52, R82. R4 najčešće se koristi u automobilskoj industriji zbog toga što sadrži sustav prekida s vrlo malim kašnjenjem koji može prekinuti operacije koje traju više ciklusa radi izvođenja nadolazećeg prekida. R5 serija se koristi u mrežnim aplikacijama i aplikacijama za pohranu podataka te ima visoki stupanj učinkovitosti, pouzdanosti i upravljanja pogreškama.

Cortex-M serija koristi se za procesiranje zvuka te slika (DSP), automatizaciju, očitanja senzora, razne IoT upotrebe, pametne zaslone te pametne satove. U odnosu na Cortex-A te Cortex-R procesore puno češće se koristi u svakodnevnoj upotrebi. Neke od serija Cortex-M procesora su M0, M0+, M3, M4, M7. M0 se koristi u sustavima vrlo male cijene i potrošnje. M0+ ima svojstvo najveće energetske učinkovitosti u Cortex-M seriji. M3, M4 i M7 imaju upotrebu u obradi digitalnog signala (DSP). Uz to, M7 omogućuje maksimalnu kontrolu performansi [21, 22].

3.3.4. Organizacija memorije ARM-a

Uobičajena organizacije memorije ARM arhitekture sastoji se od više razina priručne memorije (engl. *cache*). Prva razina memorije sadrži MMU/MPU, TCM, te L1 priručnu memoriju. Daljnje razine ovise o dizajnu samog sustava. Memorijski atributi određuju ponašanje priručne memorije na određenim razinama. Nužno je izvršiti podešavanje programskom podrškom prije nego se priručna memorija koristi. U L1 priručnoj memoriji implementirana je Harvard arhitektura, što znači da su priručne memorije za naredbe i podatke odvojene. Nadalje, ostala svojstva priručne memorije uključuju kod za ispravljanje pogrešaka ili testiranje pariteta. Omogućene su pseudoslučajne i *Round-robin* strategije zamjene. Također, moguće je zaključavanje pričuvne memorije i podešavanje da pričuvna memorija bude neblokirajuća.

4. STM32 BIBLIOTEKE

STM32 je obitelj 32-bitnih mikroupravljača koje izrađuje tvrtka STMicroelectronics. Temelje se na 32-bitnim RISC ARM jezgrama. Svaki mikroupravljač sastoji se od procesorske jezgre, statičkog RAM-a, *flash* memorije, sučelja za debugiranje te periferije. Razvojna ploča korištena za izradu biblioteka za EEPROM i NTAG NFC integrirani sklop u sebi sadrži mikroupravljač STM32F407VET6. Navedeni mikroupravljač dio je serije F4 STM32 mikroupravljača temeljenih na Cortex-M4F jezgri. Serija F4 je prva STM32 serija koja ima DSP te naredbe s pomičnim zarezom [23].

Karakteristike ovog mikroupravljača su [23]:

- frekvencija CPU (engl. *central processing unit*) maksimalne vrijednosti 168 MHz
- VDD od 1.8 V do 3.6 V
- 512 KB *flash* memorije
- 192+4 kilobajta SRAM-a uključujući 64 kilobajta CCM (engl. *core coupled memory*) podatkovnog RAM-a
- GPIO s mogućnostima vanjskih prekida (82)
- 12-bitni ADC sa 16 kanala (3)
- 12-bitni DAC sa 2 kanala (1)
- RTC
- brojači vremena (14)
- I^2C sučelja (3)
- I^2S (engl. *Inter-IC Sound*) sučelja (2)
- USART (4)
- SPI (3)
- USB 2.0
- CAN (2).

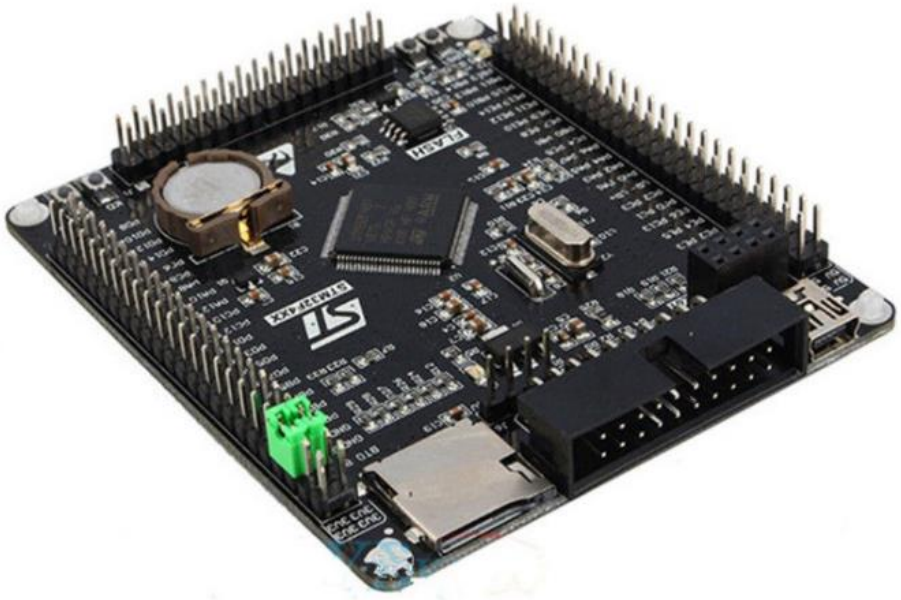
Za programiranje ove razvojne ploče potreban je ST-LINK/V2-1.

Razvojna ploča sadrži [23]:

- JTAG/SWD zaglavlje
- Micro SD
- 16 Mbit SPI Flash

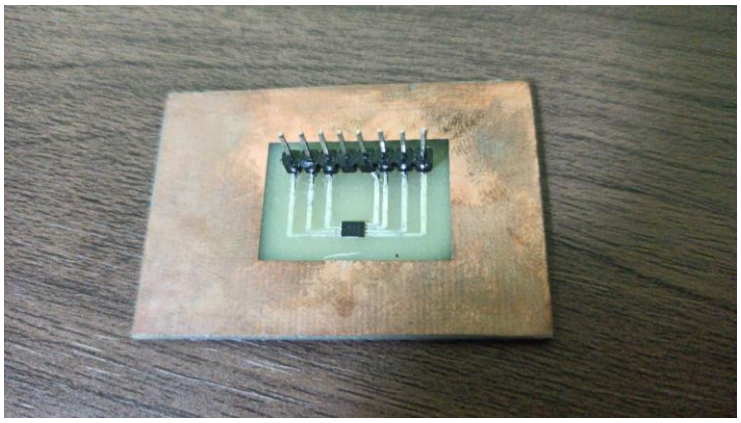
- RTC bateriju CR1220
- 10/100 Ethernet MAC (engl. *medium access control*)
- 3.3 V regulator napona
- mini USB priključak
- LED diodu za napajanje
- 2 korisničke LED diode
- tipku za resetiranje
- 2 korisničke tipke K0 i K1
- 84 pin-a
- LCD sučelje.

Na slici 4.1. nalazi se prikaz korištene razvojne ploče.



Sl. 4.1. Prikaz korištene razvojne ploče [23].

EEPROM za kojeg je razvijena biblioteka je CAT24M01 serijski EEPROM, veličine 1 Mb. Organiziran je u 131 072 riječi, od kojih je svaka veličine 8 bita. Sadrži međuspremnik koji omogućuje zapisivanje stranica veličine 256 bajtova. Podržava Standardni (100 kHz), Brzi (400 kHz) te Brzi-Plus (1 MHz) I^2C protokol. Ukoliko je WP pin EEPROM-a vrijednosti logičko '1', nije moguće vršiti zapisivanje u memoriju. Pomoću pinova A1 i A2 moguće je adresirati 4 različita CAT24M01 uređaja na istoj I^2C sabirnici. Također, integrirani sklop sadrži ECC (engl. *error code correction*) što omogućava visoku pouzdanost. Početne vrijednosti svih bajtova CAT24M01 su FFh [24]. Na slici 4.2. nalazi se prikaz CAT24M01 EEPROM-a zalemljenog na tiskanoj pločici.

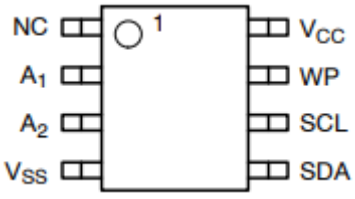


Sl. 4.2. Prikaz CAT24M01 EEPROM-a zalemljenog na tiskanoj pločici.

Ostale karakteristike ovog integriranog sklopa su [24]:

- izvor napajanja 1.8 V do 5.5 V
- Schmitt-ovi okidači te filteri za uklanjanje šuma na I^2C pin-ovima (SCL i SDA)
- CMOS tehnologija male snage
- 1 000 000 ciklusa zapisivanja/brisanja
- vrijeme zadržavanja podataka od 100 godina
- industrijski i prošireni temperaturni rasponi
- 8-pinski SOIC (engl. *small outline integrated circuit*), TSSOP (engl. *thin shrink small outline package*) te UDFN paket od 8 nožica.

Na slici 4.3. prikazana je konfiguracija pin-ova CAT24M01.



Sl. 4.3. Konfiguracija pin-ova CAT24M01 [24].

U tablici 4.1. prikazane su funkcije pin-ova CAT24M01.

Tablica 4.1. Funkcije pin-ova CAT24M01 [24].

Naziv pina	Funkcija
A1,A2	adresa uređaja
SDA	podatkovna linija I^2C
SCL	linija signala takta I^2C
WP	blokiranje zapisa u memoriju
VCC	napajanje
VSS	uzemljenje

NXP NTAG I^2C plus je obitelj NFC pametnih oznaka koje kombiniraju pasivno NFC sučelje sa kontaktnim I^2C sučeljem. Korištena pametna oznaka u praktičnom dijelu završnog rada je NTAG I^2C plus 2K (NT3H2211). Na slici 4.4. prikazan je NTAG I^2C plus 2K (NT3H2211).

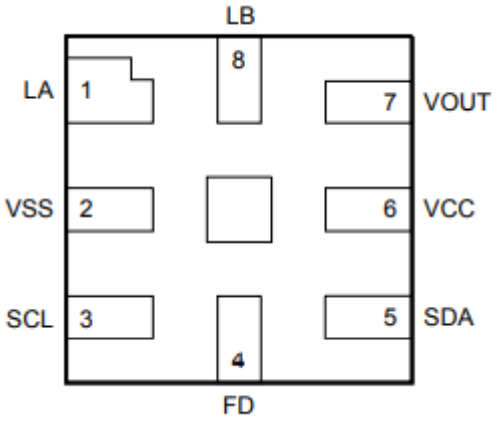
Sl. 4.4. Prikaz NTAG I^2C plus 2K (NT3H2211).

Karakteristike NTAG I^2C plus 2K (NT3H2211) su:

- pametna oznaka je sukladna s ISO/IEC 14443 2 i 3 standardima
- jedinstveni 7-bajtni UID
- I^2C *slave* sučelje, koje podržava Standardni (100 kHz) i Brzi (400 kHz) način rada
- NFC sučelje brzine prijenosa podataka 106 kilobita po sekundi, osiguranih CRC-om (engl. *cyclic redundancy check*) koje sadrži mehanizam protiv kolizije
- podesivi pin za otkrivanje polja (FD pin) zasnovan na implementaciji otvorenog odvoda za signaliziranje NFC događaja i sinkronizaciju *pass-through* načina rada

- EEPROM veličine 2k bajta, uz vrijeme zadržavanja podataka 20 godina i 500 000 ciklusa zapisivanja/brisanja
- *pass-through* način rada sa 64-bitnim SRAM međuspremnikom te FAST_WRITE i FAST_READ NFC naredbe za veću propusnost podataka
- opcije potpunog pristupa, pristupa samo za čitanje ili bez pristupa memoriji putem NFC sučelja uz mogućnost uporabe 32-bitne lozinke
- opcije potpunog pristupa, pristupa samo za čitanje ili bez pristupa memoriji putem I^2C sučelja
- podatci zaštićeni lozinkom imaju ograničeni broj pristupa s pogrešnim pokušajem autentifikacije radi prevencije neovlaštenog pristupa podatcima
- potpis orginalnosti zasnovan na kriptografiji eliptičke krivulje (ECC) za jednostavnu autentifikaciju
- izvlačenje snage iz NFC polja do 15 mW koje omogućuje napajanje spojenih uređaja (npr. mikroupravljač)
- radi na temperaturama od -40 °C do 105 °C.

NTAG I^2C plus 2K (NT3H2211) može se koristiti kao IoT čvor (u pametnoj kući, kućna automatizacija), za napajanje NFC uređaja (slušalice, zvučnici), u fitnes opremi, za zdravstvene svrhe, konfiguriranje potrošačkih aplikacija, prijenos podataka sa raznih senzora, pametni printeri i slično [25]. Na slici 4.5. prikazana je konfiguracija pinova NTAG I^2C plus 2K (NT3H2211).



Sl. 4.5. Konfiguracija pin-ova NTAG I^2C plus 2K (NT3H2211).

Tablica 4.2. prikazuje funkcije pin-ova NTAG I^2C plus 2K (NT3H2211).

Tablica 4.2. Funkcije pin-ova NTAG I²C plus 2K (NT3H2211) [25].

Naziv pina	Funkcija
LA	Spoj antene LA
VSS	Uzemljenje
SCL	Linija signala takta I²C
FD	Detekcija polja
SDA	Podatkovna linija I²C
VCC	Napajanje
VOUT	Izvlačenje energije iz polja
LB	Spoj antene LB

I²C je serijski komunikacijski protokol, koji se koristi za komunikaciju mikroupravljača s perifernim uređajima na malim udaljenostima. Prijenos podataka odvija se relativno malim brzinama. Protokol koristi sabirnicu od dvije žice za komunikaciju, od kojih žica SCL služi za generiranje signala takta, dok SDA žica služi za prijenos podataka. Obje žice spojene su na *pull-up* otpornike vrijednosti između 4 i 10 kΩ. I²C komunikacija odvija se između glavnog uređaja (engl. *master*) i pomoćnog uređaja (engl. *slave*). Signal takta uvijek je generiran od strane *mastera*. Na I²C sabirnici može biti spojeno do 128 uređaja. Postoji Standardni način rada (100 kHz), Brzi način rada (400 kHz), Brzi-Plus način rada (1 MHz), Vrlo brzi način rada (3.4 MHz) te Ultra brzi način rada (5 MHz) [26].

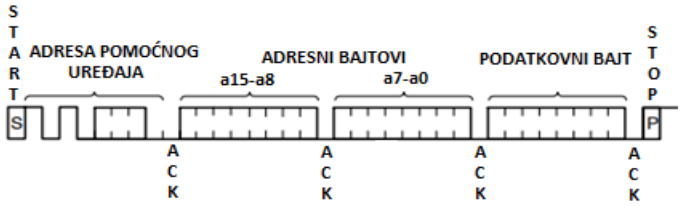
Platforma za razvoj programske podrške koja je korištena za razvoj, implementaciju te testiranje biblioteka je Keil uVision 5. Biblioteke su napisane u programskom jeziku C.

4.1. EEPROM biblioteka

EEPROM biblioteka pisana je za CAT24M01 serijski EEPROM. CAT24M01 podržava I²C komunikacijski protokol, putem kojeg je povezan sa STM32 mikroupravljačem te se koristeći taj protokol u ovoj biblioteci vrši komunikacija. Protok podataka upravljan je od strane STM32 mikroupravljača, koji se ponaša kao *master* i generira signal takta te START i STOP uvjete. CAT24M01 se ponaša kao *slave* uređaj. Oba uređaja mogu biti i pošiljatelji i primatelji podataka. Prijenos podataka putem I²C-a funkcionira na način da uređaj koji šalje podatke spušta vrijednost SDA žice u logičku '0', a otpušta ju da šalje logičku '1' (iz razloga što je žica spojena na *pull-up* otpornik). Prilikom prijenosa podataka, SDA mora biti stabilna dok SCL poprima vrijednost logičke '1', a promjena vrijednosti SDA iz logičke '1' u logičku '0' dok je SCL u logičkoj '1' interpretira se kao START uvjet. STOP uvjet je kada SDA pređe iz logičke '0' u logičku '1', dok je

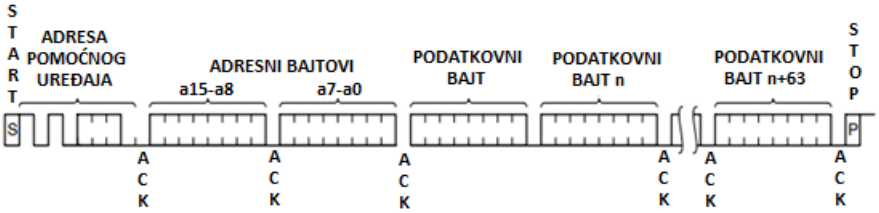
SCL u logičkoj '1'. Adresiranje CAT24M01 od strane STM32 mikroupravljača vrši se na način da mikroupravljač započinje komunikaciju kreiranjem START uvjeta i zatim šalje 8-bitnu *slave* adresu gdje su prva 4 bita vrijednosti 1010. Peti i šesti bit (vrijednosti pin-ova A2 i A1) odabiru jedan od četiri moguća CAT24M01 uređaja koji su povezani na I^2C sabirnicu. Sedmi bit predstavlja vrijednost najznačajnijeg bita interne adrese (a16). Zadnji bit označava radi li se o operaciji čitanja (ukoliko je vrijednost '1'), ili pisanja (ukoliko je vrijednost '0'). Zatim se šalju 2 bajta koja predstavljaju vrijednost memorijske lokacije unutar EEPROM-a.

CAT24M01 omogućava dvije opcije zapisivanja podataka u memoriju, zapisivanje bajta ili zapisivanje stranice. Zapisivanje bajta odvija se na način da STM32 mikroupravljač generira START uvjet, nakon kojeg šalje 4 bajta: *slave* adresu, dva adresna bajta i bajt podataka koji trebaju biti zapisani. CAT24M01 potvrđuje primanje svakog od četiri bajta (ACK) te na kraju komunikacije mikroupravljač generira STOP uvjet. Nakon toga, pokreće internu operaciju zapisivanja podataka. Ovaj proces prikazan je na slici 4.6.



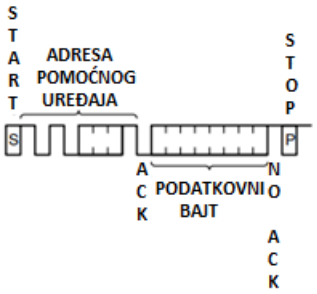
Sl. 4.6. Operacija zapisivanja bajta u CAT24M01 [24].

CAT24M01 sastoji se od 131 072 bajtova podataka, koji su raspoređeni u 512 stranica, od kojih svaka sadrži 256 bajtova. Devet najznačajnijih bitova adrese označuju stranicu, a zadnjih osam bitova označuju bajt unutar stranice kojem se pristupa. Drugi način zapisivanja podataka u CAT24M01 je zapisivanje cijele stranice te stoga u jednom ciklusu zapisivanja može biti zapisano 256 bajtova. Ukoliko mikroupravljač pošalje više od 256 bajtova, raniji bajtovi biti će prepisani kasnijim bajtovima, no u napisanoj biblioteci je taj problem izbjegnut. Na slici 4.7. nalazi se proces zapisivanja stranice.



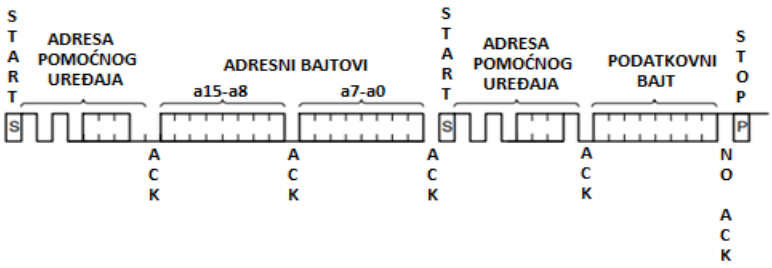
Sl. 4.7. Operacija zapisivanja stranice u CAT24M01 [24].

Čitanje podataka iz CAT24M01 može se izvršiti na više različitih načina. Prvi način je direktno čitanje bajta. Unutarnji adresni brojač od CAT24M01 pokazuje na zadnji bajt kojem je pristupljeno prilikom zadnje operacije čitanja ili pisanja podataka. Operacija direktnog čitanja bajta izvršava se na način da mikroupravljač generira START uvjet, nakon kojega šalje bajt *slave* adrese, čija je vrijednost najmanje značajnog bita jednaka logičkoj '1' što, kao prethodno objašnjeno, označava operaciju čitanja. Nakon toga, CAT24M01 potvrđuje primanje bajta (ACK) te mikroupravljaču šalje bajt na čiju adresu pokazuje unutarnji adresni brojač. Mikroupravljač tada šalje NoACK i generira STOP uvjet za završetak komunikacije. Operacija direktnog čitanja nema samostalnu praktičnu upotrebu. Operacija direktnog čitanja bajta prikazana je na slici 4.8.



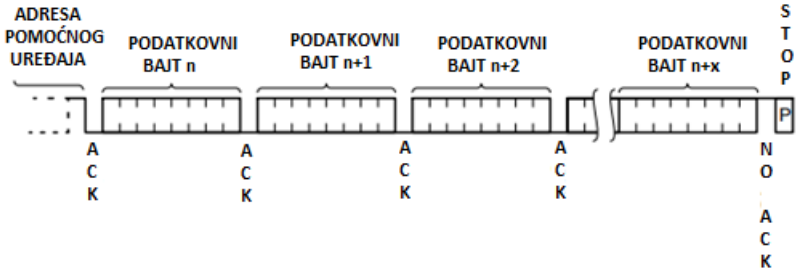
Sl. 4.8. Operacija direktnog čitanja bajta iz CAT24M01 [24].

Operacija selektivnog čitanja izvršava se na način da mikroupravljač počne izvoditi *'dummy'* operaciju zapisivanja podataka na adresu s koje se želi čitati bajt podataka. Izvršava se na sljedeći način: mikroupravljač generira START uvjet, nakon kojeg šalje *slave* adresu, s vrijednosti najmanje značajnog bita jednakoj logičkoj '0', što označava da se radi o operaciji zapisivanja podataka, a zatim šalje 2 bajta adrese. Nakon toga, umjesto da se nastavi operacija zapisivanja, mikroupravljač ponovo generira START uvjet, nakon kojeg slijedi operacija direktnog čitanja te se podatak sa željene adrese uspješno pročita. Na slici 4.9. prikazana je operacija selektivnog čitanja bajta.



Sl. 4.9. Operacija selektivnog čitanja bajta iz CAT24M01 [24].

Sljedeći način čitanja podataka iz CAT24M01 je sekvencijalno čitanje. Prilikom procesa sekvencijalnog čitanja, umjesto da kao kod direktnog ili selektivnog čitanja nakon što mikroupravljač primi bajt podataka pošalje NoACK i generira STOP uvjet, on šalje ACK (potvrđuje primanje jednog bajta podatka) te nakon što CAT24M01 primi tu potvrdu, nastavlja slati bajtove podataka slijedno mikroupravljaču, sve dok mikroupravljač sa NoACK i STOP ne označi završetak komunikacije. Ukoliko je kraj memorije CAT24M01 dosegnut, mikroupravljaču se počinju slati podatci s početka memorije. Na slici 4.10. nalazi se prikaz sekvencijalnog čitanja bajta [24].



Sl. 4.10. Operacija sekvencijalnog čitanja bajta iz CAT24M01 [24].

Biblioteka za CAT24M01 sastoji se od 10 funkcija, od kojih su 5 funkcije za čitanje podataka, 4 funkcije za zapisivanje podataka, a preostala funkcija služi za generiranje vremenske odgode zbog potrebnog vremena za izvođenje internog zapisivanja u memoriju. Cjelokupni kod te opisi pojedinih funkcija i njihovih parametara nalaze se u prilogu diplomskog rada. Na slici 4.11. prikazi su prototipovi funkcija za čitanje podataka.

```
int ImmediateReadByte(uint8_t slave_address, uint8_t *data);

int SelectiveReadByte(uint8_t slave_address, uint8_t page_address, uint8_t
byte_address, uint8_t *data);
int SelectiveReadNBytes(uint8_t slave_address, uint8_t page_address,
uint8_t byte_address, uint8_t *data, uint32_t numOfBytes);
int SelectiveReadNBytesNoWrapAround(uint8_t slave_address, uint8_t
page_address, uint8_t byte_address, uint8_t *data, uint32_t numOfBytes);
int SelectiveReadPage(uint8_t slave_address, uint8_t page_address, uint8_t
*data);
```

Sl. 4.11. Prikaz svih prototipova i parametara funkcija iz biblioteke za čitanje podataka iz CAT24M01.

Na slici 4.12. prikazani su prototipovi funkcija za zapisivanje podataka.

```
int WriteByte(uint8_t slave_address, uint8_t page_address, uint8_t
byte_address, uint8_t *data);

int WriteNBytes(uint8_t slave_address, uint8_t page_address, uint8_t
byte_address, uint8_t *data, uint32_t numOfBytes);

int WriteNBytesNoWrapAround(uint8_t slave_address, uint8_t page_address,
uint8_t byte_address, uint8_t *data, uint32_t numOfBytes);

int WritePage(uint8_t slave_address, uint8_t page_address, uint8_t *data);
```

Sl. 4.12. Prikaz svih prototipova i parametara funkcija iz biblioteke za zapisivanje podataka u CAT24M01.

Na slici 4.13. prikazan je prototip funkcije za vremensku odgodu.

```
void DelayMs(uint32_t delay);
```

Sl. 4.13. Prikaz funkcije za vremensku odgodu.

Sljedeći isječak iz koda na slici 4.14. prikazuje na koji način STM32 mikroupravljač vrši komunikaciju s CAT24M01 prilikom zapisivanja bajta podataka. Ovaj postupak napisan je prema uputama iz STM32F40x Reference Manuala [20, str.849].

Linija	Kod
67:	`I2C1->CR1 |= START_GEN;`
68:	`while(!(I2C1->SR1 & START_BIT)){}`
69:	`I2C1->DR = slave_address << 1;`
70:	
71:	`while(!(I2C1->SR1 & ADDR_SENT)) {}`
72:	`tmp = I2C1->SR2;`
73:	
74:	`while(!(I2C1-> SR1 & TxE)) {}`
75:	`I2C1->DR = page_address;`
76:	`while(!(I2C1-> SR1 & TxE)) {}`
77:	`I2C1->DR = byte_address;`
78:	`while(!(I2C1-> SR1 & TxE)) {}`
79:	`I2C1->DR = *data;`
80:	`while(!(I2C1-> SR1 & TxE)) {}`
81:	
82:	`I2C1->CR1 |= STOP_GEN;`
83:	
84:	`DelayMs(5);`

Sl. 4.14. Zapisivanje bajta podataka mikroupravljačem u EEPROM.

Sljedeći isječak iz koda na slici 4.15. prikazuje na koji način STM32 mikroupravljač vrši komunikaciju s CAT24M01 prilikom selektivnog čitanja bajta podataka. Ovaj postupak napisan je prema uputama iz STM32F40x Reference Manuala [20, str.850].

```
Linija      Kod
9:          I2C1->CR1 |= START_GEN;
10:         while(!(I2C1->SR1 & START_BIT)){}
11:         I2C1->DR = slave_address << 1;
12:
13:         while(!(I2C1->SR1 & ADDR_SENT)) {}
14:         tmp = I2C1->SR2;
15:
16:         while(!(I2C1-> SR1 & TxE)) {}
17:         I2C1->DR = page_address;
18:         while(!(I2C1-> SR1 & TxE)) {}
19:         I2C1->DR = byte_address;
20:         while(!(I2C1-> SR1 & TxE)) {}
21:
22:         I2C1->CR1 |= START_GEN;
23:         while(!(I2C1->SR1 & START_BIT)){}
24:         I2C1->DR = slave_address << 1 | 1;
25:
26:         while(!(I2C1->SR1 & ADDR_SENT)) {}
27:         I2C1->CR1 &= ~ACK;
28:         tmp = I2C1->SR2;
29:
30:         while(!(I2C1-> SR1 & RxE)) {}
31:         *data = I2C1->DR;
32:
33:         I2C1->CR1 |= STOP_GEN;
```

Sl. 4.15. Čitanje bajta podataka mikroupravljačem iz EEPROM-a.

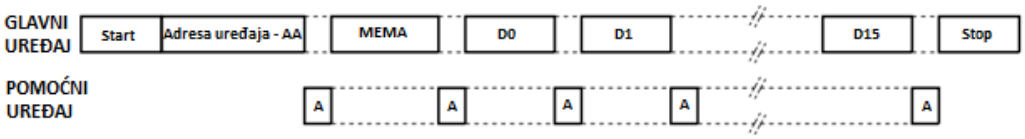
4.2. NFC NXP biblioteka

NTAG I^2C plus 2K (NT3H2211) podržava I^2C protokol te se prilikom korištenja ove biblioteke tim protokolom vrši komunikacija između STM32 mikroupravljača i NT3H2211. Prijenos podataka iniciran je od strane *master* uređaja, koji također vrši sinkronizaciju pomoću signala takta. U ovom slučaju, *master* uređaj je STM32 mikroupravljač, a *slave* uređaj je NT3H2211. START uvjet je identificiran od strane NT3H2211, ukoliko se dogodio padajući brid na SDA, dok je vrijednost SCL u logičkoj '1'. Stop uvjet je identificiran od strane NT3H2211, ukoliko je došlo do rastućeg brida na SDA, a vrijednost SCL je u logičkoj '1'. NT3H2211 vrši očitanje vrijednosti bita s SDA za rastući brid SCL prilikom unosa podataka. Također, SDA mora biti stabilna prilikom rastućeg brida SCL te treba mijenjati vrijednost samo na padajući brid SCL. Za označavanje uspješnog prijenosa bajta podatka, koristi se bit potvrde (ACK), koji primatelj podataka, bio on *master* ili *slave* uređaj, generira povlačenjem linije SDA u logičku '0' u devetom ciklusu signala takta. Početak komunikacije počinje START uvjetom, te nakon toga mikroupravljač šalje bajt adrese uređaja, naziva *Slave Address*. Vrijednost adrese uređaja je jednaka heksadekadskoj vrijednosti ABh, ukoliko je operacija koja se treba izvršiti čitanje podataka ili vrijednosti AAh ukoliko se treba izvršiti zapisivanje podataka (najmanje značajan bit

je jednak 1 za čitanje i najmanje značajan bit je jednak 0 za zapisivanje). Operacije zapisivanja i čitanja NT3H2211 uvijek se izvršavaju na način da se zapiše ili iščita jedan blok podataka. Blok podataka sastoji se od 16 bajtova.

Operacija zapisivanja podataka u NT3H2211 pomoću STM32 mikroupravljača vrši se na način da mikroupravljač generira START uvjet i zatim šalje bajt heksadekadske vrijednosti AAh (adresa uređaja, zapisivanje podataka). Primanje bajta adrese uređaja NT3H2211 potvrđuje bitom potvrde (ACK). Nakon toga mikroupravljač šalje bajt adrese lokacije u EEPROM-u ili SRAM-u gdje se treba izvršiti zapisivanje podataka. NT3H2211 potvrđuje primanje adrese ACK bitom i zatim mikroupravljač šalje 16 bajtova podataka (jednog bloka podataka). NT3H2211 primanje svakog od tih bajtova pojedinačno potvrđuje ACK bitom. Nakon potvrde primanja zadnjeg podatkovnog bajta, mikroupravljač generira STOP uvjet. Na slici 4.16. prikazana je operacija zapisivanja u NT3H2211.

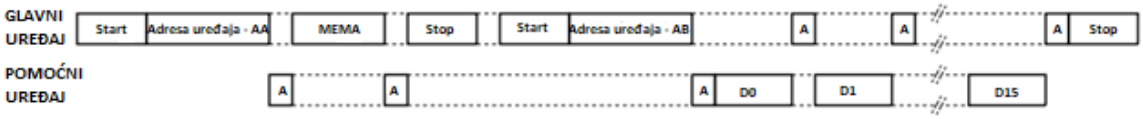
Operacija čitanja NT3H2211 pomoću STM32 mikroupravljača putem I^2C



Sl. 4.16. Operacija zapisivanja u NT3H2211 [25].

komunikacijskog protokola vrši se na način da mikroupravljač generira START uvjet te zatim šalje bajt heksadekadske vrijednosti AAh (adresa uređaja, zapisivanje podataka). Razlog iz kojeg se šalje AAh umjesto ABh (adresa uređaja, čitanje podataka) je što unutarnji brojač adrese u NT3H2211 prilikom procesa čitanja podataka, pokazuje na zadnji adresni blok kojem je pristupljeno u prethodnoj operaciji čitanja ili zapisivanja. Stoga je potrebno izvesti *'dummy'* operaciju zapisivanja da bi se moglo pristupiti željenoj adresi unutar EEPROM-a ili SRAM-a. NT3H2211 potvrđuje primanje bajta adrese uređaja pomoću potvrdnog bita ACK. Nakon toga mikroupravljač šalje bajt adrese lokacije u EEPROM-u ili SRAM-u sa koje se treba izvršiti čitanje podataka. Zatim, NT3H2211 potvrđuje primanje bajta adrese ACK bitom. Nakon toga mikroupravljač generira STOP uvjet. Zatim, mikroupravljač generira novi START uvjet, nakon kojeg šalje bajt heksadekadske vrijednosti ABh (adresa uređaja, čitanje podataka). NT3H2211 potvrđuje primanje adrese uređaja potvrdnim bitom ACK te započinje prijenos 16 bajtova podataka (jednog bloka podataka). Svaki bajt podataka, mikroupravljač zasebno potvrđuje ACK

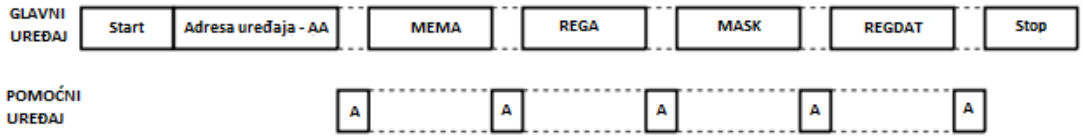
bitom, te nakon potvrde zadnjeg bajta podataka generira STOP uvjet. Na slici 4.17. prikazana je operacija čitanja iz NT3H2211.



Sl. 4.17. Operacija čitanja iz NT3H2211 [25].

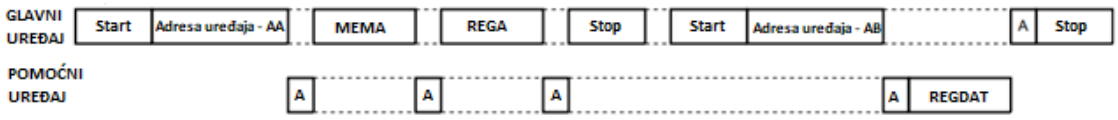
Operacije čitanja i zapisivanja u NT3H2211, na ovaj način vrše se za čitanje i zapisivanje u EEPROM, SRAM te sve registre uređaja, osim sesijskog registra. Za komunikaciju sa sesijskim registrom uređaja, postoje zasebne operacije čitanja i zapisivanja, u kojima se dohvaća i šalje vrijednost pojedinačnog bajta podataka, za razliku od uobičajenog načina dohvaćanja i slanja bloka podataka u NT3H2211.

Operacija zapisivanja u sesijski registar izvršava se na sljedeći način. Mikroupravljač generira START uvjet i šalje bajt heksadekadske vrijednosti AAh (adresa uređaja, operacija zapisivanja). NT3H2211 potvrđuje primanje adrese uređaja ACK potvrdnim bitom. Zatim, mikrupravljač šalje bajt memorijske adrese sesijskog registra koja je jednaka heksadekadskoj vrijednosti FEh te ju NT3H2211 potvrđuje ACK bitom. Nakon toga mikroupravljač šalje bajt adrese registra, što predstavlja adresu bajta unutar sesijskog registra čija će se vrijednost mijenjati. Vrijednosti mogu biti od 00h do 07h. NT3H2211 ponovo potvrđuje primanje bajta ACK-om. Zatim, mikroupravljač šalje tzv. MASK bajt, koji predstavlja masku koja označava bitove unutar bajta čija vrijednost može biti promijenjena. Ponovo, NT3H2211 potvrđuje primanje bajta potvrdnim bitom ACK. Na kraju, mikroupravljač šalje novu vrijednost bajta koji se mijenja (REGDAT), NT3H2211 ju potvrđuje ACK bitom te mikroupravljač generira STOP uvjet. Na slici 4.18. prikazana je operacija zapisivanja u sesijski registar.



Sl. 4.18. Operacija zapisivanja u sesijski registar NT3H2211 [25].

Operacija čitanja vrijednosti iz sesijskog registra izvršava se na sljedeći način. Mikroupravljač generira START uvjet i zatim šalje bajt heksadekadske vrijednosti AAh (adresa uređaja, zapisivanje podataka). Razlog iz kojeg se šalje AAh umjesto ABh (adresa uređaja, čitanje podataka) je što unutarnji brojač adrese u NT3H2211 prilikom procesa čitanja podataka, pokazuje na zadnji adresni blok kojem je pristupljeno u prethodnoj operaciji čitanja ili zapisivanja. Stoga se izvodi *'dummy'* operacija zapisivanja da bi se moglo pristupiti željenoj adresi. NT3H2211 potvrđuje primanje bajta adrese uređaja ACK. Nakon toga, mikroupravljač šalje željenu adresu (adresu sesijskog registra koja je jednaka heksadekadskoj vrijednosti FEh) te ju NT3H2211 potvrđuje. Zatim, mikroupravljač šalje bajt adrese registra, koja predstavlja bajt unutar sesijskog registra koji se treba pročitati te ga NT3H2211 potvrđuje. Mikroupravljač generira STOP uvjet te nakon njega ponovni START uvjet. Zatim, šalje bajt heksadekadske vrijednosti ABh, koji predstavlja adresu uređaja te da se izvodi operacija čitanja. NT3H2211 potvrđuje primanje adrese uređaja potvrdnim bitom ACK te šalje odabrani bajt podataka (REGDAT) iz sesijskog registra. Mikroupravljač potvrđuje primanje bajta ACK bitom te generira STOP uvjet [25]. Na slici 4.19. prikazana je operacija čitanja iz sesijskog registra.



Sl. 4.19. Operacija čitanja iz sesijskog registra NT3H2211 [25].

Biblioteka za NTAG I^2C plus 2K (NT3H2211) sastoji se od 94 funkcije, od kojih je 1 za generiranje vremenske odgode, 18 za zapisivanje u memoriju (EEPROM-a ili SRAM-a), 9 za čitanje iz memorije, 4 za podešavanje statičkih bitova za zaključavanje, 4 za podešavanje dinamičkih bitova za zaključavanje, 2 za interakciju s *capability containerom*, 1 za čitanje UID-a uređaja, 12 za upravljanje registrom za lozinku i pristupnu konfiguraciju, 19 za podešavanje konfiguracijskog registra te 24 za podešavanje sesijskog registra. Cjelokupni kod te opisi pojedinih funkcija i njihovih parametara nalaze se u prilogu diplomskog rada. Na slici 4.20. prikazan je prototip funkcije za generiranje vremenske odgode.

```
void DelayMs(uint32_t delay);
```

Sl. 4.20. Prikaz funkcije za vremensku odgodu.

Na slici 4.21. prikazani su prototipovi funkcija za zapisivanje u memoriju.

```
int WriteDataEEPROM(uint8_t block_address, uint8_t *data);

int WriteDataSRAM(uint8_t block_address, uint8_t *data);

int WriteNBytesDataEEPROM(uint8_t block_address, uint8_t *data, uint32_t
NumOfBytes);
int WriteNBytesDataEEPROMAddZeros(uint8_t block_address, uint8_t *data,
uint32_t NumOfBytes);
int WriteNBytesDataEEPROMSelectByte(uint8_t block_address, uint8_t
start_byte, uint8_t *data, uint32_t NumOfBytes);
int WriteNBytesDataEEPROMAddZerosSelectByte(uint8_t block_address, uint8_t
start_byte, uint8_t *data, uint32_t NumOfBytes);
int WriteNBytesDataSRAM(uint8_t block_address, uint8_t *data, uint32_t
NumOfBytes);
int WriteNBytesDataSRAMAddZeros(uint8_t block_address, uint8_t *data,
uint32_t NumOfBytes);
int WriteNBytesDataSRAMSelectByte(uint8_t block_address, uint8_t
start_byte, uint8_t *data, uint32_t NumOfBytes);
int WriteNBytesDataSRAMAddZerosSelectByte(uint8_t block_address, uint8_t
start_byte, uint8_t *data, uint32_t NumOfBytes);
int WriteNBytesDataEEPROMNoWrapAround(uint8_t block_address, uint8_t *data,
uint32_t NumOfBytes);
int WriteNBytesDataEEPROMAddZerosNoWrapAround(uint8_t block_address,
uint8_t *data, uint32_t NumOfBytes);
int WriteNBytesDataEEPROMSelectByteNoWrapAround(uint8_t block_address,
uint8_t start_byte, uint8_t *data, uint32_t NumOfBytes);
int WriteNBytesDataEEPROMAddZerosSelectByteNoWrapAround(uint8_t
block_address, uint8_t start_byte, uint8_t *data, uint32_t NumOfBytes);
int WriteNBytesDataSRAMNoWrapAround(uint8_t block_address, uint8_t *data,
uint32_t NumOfBytes);
int WriteNBytesDataSRAMAddZerosNoWrapAround(uint8_t block_address, uint8_t
*data, uint32_t NumOfBytes);
int WriteNBytesDataSRAMSelectByteNoWrapAround(uint8_t block_address,
uint8_t start_byte, uint8_t *data, uint32_t NumOfBytes);
int WriteNBytesDataSRAMAddZerosSelectByteNoWrapAround(uint8_t
block_address, uint8_t start_byte, uint8_t *data, uint32_t NumOfBytes);
```

Sl. 4.21. Prikaz svih prototipova i parametara funkcija iz biblioteke za zapisivanje podataka u NT3H2211.

Na slici 4.22. prikazani su prototipovi funkcija za čitanje iz memorije.

```
int ReadData(uint8_t block_address, uint8_t *data);


int ReadNBytesDataEEPROM(uint8_t block_address, uint8_t *data, uint32_t
NumOfBytes);
int ReadNBytesDataEEPROMSelectByte(uint8_t block_address, uint8_t
start_byte, uint8_t *data, uint32_t NumOfBytes);
int ReadNBytesDataSRAM(uint8_t block_address, uint8_t *data, uint32_t
NumOfBytes);
int ReadNBytesDataSRAMSelectByte(uint8_t block_address, uint8_t start_byte,
uint8_t *data, uint32_t NumOfBytes);
int ReadNBytesDataEEPROMNoWrapAround(uint8_t block_address, uint8_t *data,
uint32_t NumOfBytes);
int ReadNBytesDataEEPROMSelectByteNoWrapAround(uint8_t block_address,
uint8_t start_byte, uint8_t *data, uint32_t NumOfBytes);
int ReadNBytesDataSRAMNoWrapAround(uint8_t block_address, uint8_t *data,
uint32_t NumOfBytes);
int ReadNBytesDataSRAMSelectByteNoWrapAround(uint8_t block_address, uint8_t
start_byte, uint8_t *data, uint32_t NumOfBytes);
```

Sl. 4.22. Prikaz svih prototipova i parametara funkcija iz biblioteke za čitanje podataka iz NT3H2211.

Svaka 4 bajta u memoriji predstavljaju jednu stranicu, iz perspektive NFC sučelja. Pomoću statičkih bitova za zaključavanje može se podesiti zaključavanje pojedine stranice u memoriji (stranice 0-15), da imaju pristup samo za čitanje. Također, stranice koje su konfigurirane da su samo za čitanje mogu biti konfigurirane natrag da se u njih ponovo može zapisivati. Osim toga, moguće je blokirati statičko zaključavanje određenog raspona stranica u memoriji. Na slici 4.23. prikazani su prototipovi funkcija za podešavanje statičkih bitova za zaključavanje.

```
int ConfigNFCStaticLockBytesReadOnly(uint8_t NumberOfPage);
int ConfigNFCStaticLockBytesReadWrite(uint8_t NumberOfPage);
int BlockNFCStaticLocking(uint8_t BlockLockPageRange);
int UnblockNFCStaticLocking(uint8_t UnblockLockPageRange);
```

Sl. 4.23. Prikaz svih prototipova i parametara funkcija za podešavanje statičkih bitova za zaključavanje.

Dinamički bitovi za zaključavanje funkcioniraju na isti način kao statički bitovi za zaključavanje, osim što su korišteni za zaključavanje raspona stranica (stranice imaju pristup samo za čitanje), umjesto pojedine stranice. Blokirani rasponi stranica također mogu biti odblokirani, odnosno u njih se ponovo može zapisivati. Osim toga, moguće je blokirati dinamičko zaključavanje određenog raspona stranica u memoriji.

Na slici 4.24. prikazani su prototipovi funkcija za podešavanje dinamičkih bitova za zaključavanje.

```
int ConfigNFCDynamicLockBytesReadOnly(uint8_t ReadOnlyPageRange);
int ConfigNFCDynamicLockBytesReadWrite(uint8_t ReadWritePageRange);
int BlockNFCDynamicLocking(uint8_t BlockLockPageRange);
int UnblockNFCDynamicLocking(uint8_t UnblockLockPageRange);
```

Sl. 4.24. Prikaz svih prototipova i parametara funkcija za podešavanje dinamičkih bitova za zaključavanje.

U *capability containeru* nalaze se kontrolni podatci za podešavanje veličine dijela memorije korištenog za pohranu NDEF poruka. *Capability container* potrebno je podesiti na vrijednost 2032 bajta, ili 176 + 128*N bajtova, gdje je vrijednost N manja ili jednaka 14. Na slici 4.25. prikazani su prototipovi funkcija za interakciju s *capability containerom*.

```
int GetCC(uint8_t *CC_4_bytes);
int SetCC(uint8_t *CC_4_bytes);
```

Sl. 4.25. Prikaz svih prototipova i parametara funkcija za interakciju s *capability containerom*.

U prvom bajtu EEPROM memorije, nalazi se jedinstveni 7-bitni UID, koji je napisan prema standardu ISO/IEC 14443-3. Na slici 4.26. prikazan je prototip funkcije za dohvaćanje UID-a.

```
int ReadUID(uint8_t *data);
```

Sl. 4.26. Prikaz funkcije za dohvaćanje UID-a NT3H2211.

Registar za lozinku i pristupnu konfiguraciju koristi se za konfiguraciju lozinke te uvjeta pristupa NT3H2211. Također, moguće je podesiti da su određena područja memorije zaštićena lozinkom. Ukoliko se lozinka omogući, zapisivanje u bajtove koji služe za konfiguraciju lozinke i uvjeta pristupa, moguće je izvršiti samo nakon uspješne autentifikacije lozinkom. Ukoliko se lozinka pokuša pročitati putem NFC ili I^2C sučelja, NT3H2211 će vratiti bajtove heksadekadske vrijednosti jednake 00h.

Na slici 4.27. prikazani su prototipovi funkcija za upravljanje registrom za lozinku i pristupnu konfiguraciju.

```
int SetPasswordPasswordACK(uint32_t password, uint16_t passwordAck);
int SetAuthenticationPointer(uint8_t NFCpageValue);
int SetNFC_PROT(uint32_t password, uint16_t passwordAck);
int ResetNFC_PROT(uint32_t password, uint16_t passwordAck);
int SetNFC_DIS_SEC1(uint32_t password, uint16_t passwordAck);
int ResetNFC_DIS_SEC1(uint32_t password, uint16_t passwordAck);
int SetAUTHLIM(uint8_t AuthLim, uint32_t password, uint16_t passwordAck);
int Set2K_PROT(uint32_t password, uint16_t passwordAck);
int Reset2K_PROT(uint32_t password, uint16_t passwordAck);
int SetSRAM_PROT(uint32_t password, uint16_t passwordAck);
int ResetSRAM_PROT(uint32_t password, uint16_t passwordAck);
int SetI2C_PROT(uint8_t value, uint32_t password, uint16_t passwordAck);
```

Sl. 4.27. Prikaz svih prototipova i parametara funkcija za upravljanje registrom za lozinku i pristupnu konfiguraciju.

Podešavanje ponašanja NTAG I^2C plus (NT3H2211) može biti izvedeno pristupom konfiguracijskom registru, te promjenom željenih bajtova u registru, nakon čega je potrebno resetirati NT3H2211 da bi došlo do samih promjena u ponašanju NT3H2211. Konfiguracijskom registru može biti pristupljeno putem NFC sučelja ili putem I^2C sučelja, za koje je biblioteka pisana.

Na slici 4.28. prikazani su prototipovi funkcija za podešavanje konfiguracijskog registra.

```
int SetNFCS_I2C_RST_ON_OFF();
int ResetNFCS_I2C_RST_ON_OFF();
int SetPTHRU_ON_OFF();
int ResetPTHRU_ON_OFF();
int SetSRAM_MIRROR_ON_OFF();
int ResetSRAM_MIRROR_ON_OFF();
int SetTRANSFER_DIR();
int ResetTRANSFER_DIR();
int SetFD_OFF(uint8_t value);
int SetFD_ON(uint8_t value);
int SetLAST_NDEF_BLOCK(uint8_t block_address);
int SetSRAM_MIRROR_BLOCK(uint8_t block_address);
int SetWDT_LS(uint8_t value);
int SetWDT_MS(uint8_t value);
int SetI2C_CLOCK_STR();
int ResetI2C_CLOCK_STR();
int SetREG_LOCK_I2C();
int SetREG_LOCK_NFC();
int ResetREG_LOCK_NFC();
```

Sl. 4.28. Prikaz svih prototipova i parametara funkcija za podešavanje konfiguracijskog registra.

Sesijski registar služi za istu svrhu kao konfiguracijski registar, a glavna razlika između ta dva registra je ta što se prilikom konfiguriranja sesijskog registra promjene u ponašanju počinju događati odmah (unutar te komunikacijske sesije), za razliku od konfiguracijskog registra gdje je potrebno resetirati NT3H2211 da bi promjene bile vidljive. Nakon resetiranja uređaja, sesijski registar poprima vrijednosti konfiguracijskog registra. Uz to, razlika između ova 2 registra je ta što su REG_LOCK bitovi dostupni jedino u konfiguracijskom registru, dok su NS_REG bitovi dostupni jedino u sesijskom registru.

Na slici 4.29. prikazani su prototipovi funkcija za podešavanje sesijskog registra.

```
int ReadSessionRegisterNC_REG(uint8_t *data);
int ReadSessionRegisterLAST_NDEF_BLOCK(uint8_t *data);
int ReadSessionRegisterSRAM_MIRROR_BLOCK(uint8_t *data);
int ReadSessionRegisterWDT_LS(uint8_t *data);
int ReadSessionRegisterWDT_MS(uint8_t *data);
int ReadSessionRegisterI2C_CLOCK_STR(uint8_t *data);
int ReadSessionRegisterNS_REG(uint8_t *data);
int SetSessionRegisterNFCS_I2C_RST_ON_OFF();
int ResetSessionRegisterNFCS_I2C_RST_ON_OFF();
int SetSessionRegisterPTHRU_ON_OFF();
int ResetSessionRegisterPTHRU_ON_OFF();
int ConfigSessionRegisterFD_OFF(uint8_t regdat);
int ConfigSessionRegisterFD_ON(uint8_t regdat);
int SetSessionRegisterSRAM_MIRROR_ON_OFF();
int ResetSessionRegisterSRAM_MIRROR_ON_OFF();
int SetSessionRegisterTRANSFER_DIR();
int ResetSessionRegisterTRANSFER_DIR();
int ConfigSessionRegisterLAST_NDEF_BLOCK(uint8_t regdat);
int ConfigSessionRegisterSRAM_MIRROR_BLOCK(uint8_t regdat);
int ConfigSessionRegisterWDT_LS(uint8_t regdat);
int ConfigSessionRegisterWDT_MS(uint8_t regdat);
int SetSessionRegisterI2C_LOCKED();
int ResetSessionRegisterI2C_LOCKED();
int ResetSessionRegisterEEPROM_WR_ERR();
```

Sl. 4.29. Prikaz svih prototipova i parametara funkcija za podešavanje sesijskog registra.

Sljedeći isječak iz koda na slici 4.30. prikazuje na koji način STM32 mikroupravljač vrši komunikaciju s NT3H2211 prilikom zapisivanja bloka podataka. Ovaj postupak napisan je prema uputama iz STM32F40x Reference Manuala [20, str.849] te NT3H2211 Data Sheeta [25, str, 44].

```
Linija      Kod
142:        I2C1->CR1 |= START_GEN;
143:        while(!(I2C1->SR1 & START_BIT)){}
144:        I2C1->DR = SLAVE_ADDRESS_WRITE;
145:        while(!(I2C1->SR1 & ADDR_SENT)) {}
146:        tmp = I2C1->SR2;
147:
148:        while(!(I2C1-> SR1 & TxE)) {}
149:        I2C1->DR = block_address;
150:        while(!(I2C1-> SR1 & TxE)) {}
151:        for(i=0;i<16;i++){
152:          I2C1->DR = *data++;
153:          while(!(I2C1-> SR1 & TxE)) {}
154:        }
155:
156:        I2C1->CR1 |= STOP_GEN;
157:        DelayMs(5);
```

Sl. 4.30. Zapisivanje bloka podataka mikroupravljačem u NT3H2211.

Sljedeći isječak iz koda na slici 4.31. prikazuje na koji način STM32 mikroupravljač vrši komunikaciju s NT3H2211 prilikom čitanja bloka podataka. Ovaj postupak napisan je prema uputama iz STM32F40x Reference Manuala [20, str.850] te NT3H2211 Data Sheeta [25, str, 44].

```
Linija      Kod
17:         I2C1->CR1 |= ACK;
18:         I2C1->CR1 |= START_GEN;
19:         while(!(I2C1->SR1 & START_BIT)){}
20:         I2C1->DR = SLAVE_ADDRESS_WRITE;
21:         while(!(I2C1->SR1 & ADDR_SENT)) {}
22:         tmp = I2C1->SR2;
23:
24:         while(!(I2C1-> SR1 & TxE)) {}
25:         I2C1->DR = block_address;
26:         while(!(I2C1-> SR1 & TxE)) {}
27:         I2C1->CR1 |= STOP_GEN;
28:         I2C1->CR1 |= START_GEN;
29:         while(!(I2C1->SR1 & START_BIT)){}
30:         I2C1->DR = SLAVE_ADDRESS_READ;
31:
32:         while(!(I2C1->SR1 & ADDR_SENT)) {}
33:         tmp = I2C1->SR2;
34:
35:         for(i=0;i<16;i++){
36:           while(!(I2C1-> SR1 & RxE)) {}
37:           *data++ = I2C1->DR;
38:         }
39:         I2C1->CR1 |= STOP_GEN;
```

Sl. 4.31. Čitanje bloka podatka mikroupravljačem iz NT3H2211.

Sljedeći isječak iz koda na slici 4.32. prikazuje na koji način STM32 mikroupravljač vrši komunikaciju s NT3H2211 prilikom konfiguracije sesijskog registra. Ovaj postupak napisan je prema uputama iz STM32F40x Reference Manuala [20, str.849] te NT3H2211 Data Sheeta [25, str.46]. U ovom primjeru vrši se konfiguracija drugog bajta sesijskog registra (SRAM_MIRROR_BLOCK).

Linija	Kod

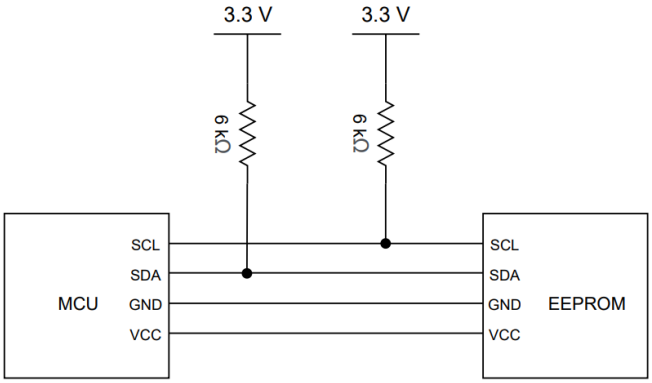
```
5643:     I2C1->CR1 |= START_GEN;
5644:     while(!(I2C1->SR1 & START_BIT)){}
5645:     I2C1->DR = SLAVE_ADDRESS_WRITE;
5646:     while(!(I2C1->SR1 & ADDR_SENT)) {}
5647:     tmp = I2C1->SR2;
5648:
5649:     while(!(I2C1-> SR1 & TxE)) {}
5650:     I2C1->DR = SESSION_REG;         //MEMA
5651:     while(!(I2C1-> SR1 & TxE)) {}
5652:     I2C1->DR = 2;                   //REGA
5653:     while(!(I2C1-> SR1 & TxE)) {}
5654:     I2C1->DR = 0b11111111;          //MASK
5655:     while(!(I2C1-> SR1 & TxE)) {}
5656:     I2C1->DR = regdat;              //REGDAT
5657:     while(!(I2C1-> SR1 & TxE)) {}
5658:
5659:     I2C1->CR1 |= STOP_GEN;
5660:     DelayMs(5);
```

Sl. 4.32. Konfiguracija sesijskog registra NT3H2211 mikroupravljačem.

5. TESTIRANJE

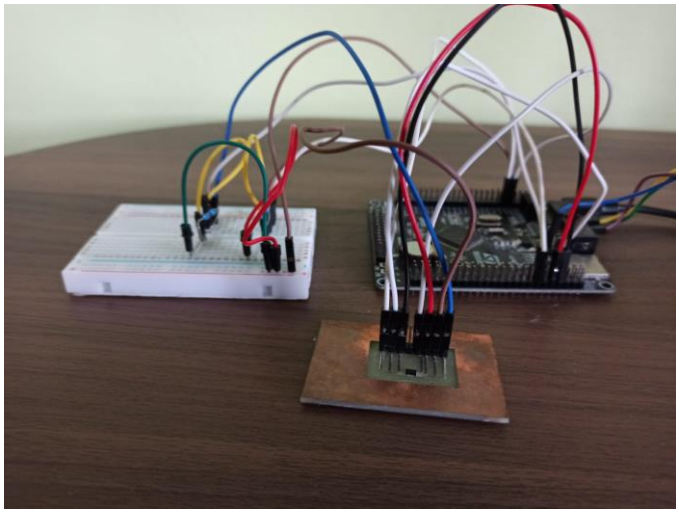
5.1. Testiranje EEPROM biblioteke

Testiranje EEPROM biblioteke je izvedeno na način da se nakon implementacije svake pojedine funkcije ta funkcija pokrenula na razvojnom sustavu Keil uVision 5. Ukoliko je funkcija uspješno radila s nasumičnim uobičajenim vrijednostima parametrima, slijedilo je testiranje funkcije s parametrima rubnih vrijednosti. Ako je i tada funkcija uspješno radila, unešeni su parametri s namjerom da funkcija netočno radi, odnosno ne izvršava predviđenu funkciju. Ukoliko nakon bilo kojeg od prethodno navedenih koraka testiranja funkcija nije radila, izvršavao se postupak debugiranja. Debugiranje je izvođeno u istom programu za razvoj programske podrške Keil uVision 5 u kojem je izvršen razvoj kao i testiranje biblioteke. Zatim su postupkom debugiranja uočene te ispravljene sve nastale pogreške. Na slici 5.1. prikazana je shema spajanja mikroupravljača i EEPROM-a.



Sl. 5.1. Shema spajanja mikroupravljača i EEPROM-a.

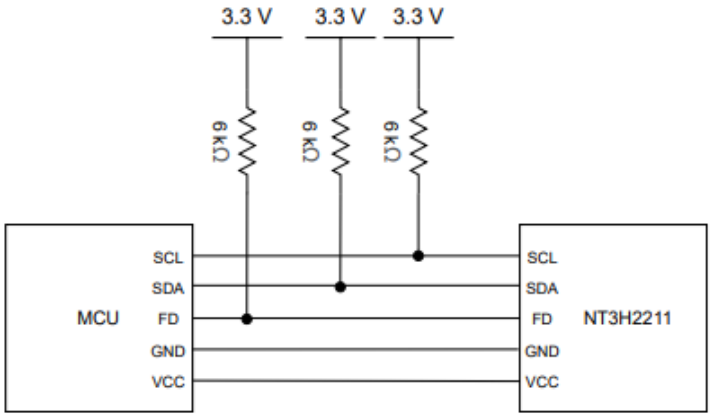
Na slici 5.2. prikazano je spajanje mikroupravljača s EEPROM-om prilikom testiranja.



Sl. 5.2. Spajanje mikroupravljača s EEPROM-om prilikom testiranja.

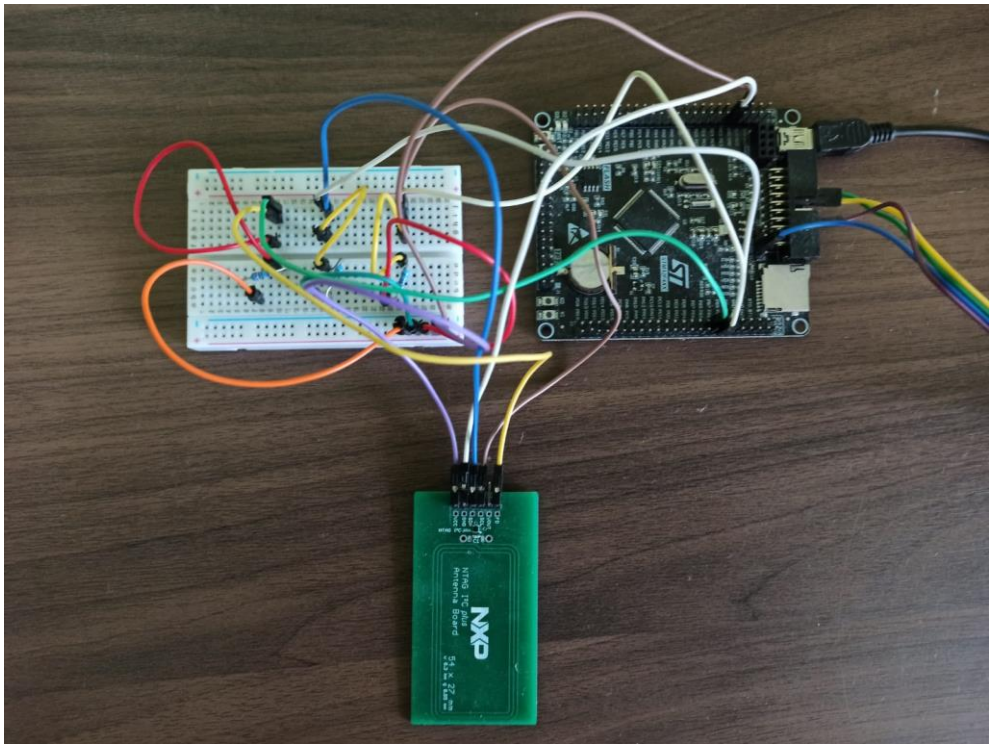
5.2. Testiranje NFC NXP biblioteke

Prvi dio procesa testiranja NFC NXP biblioteke bio je izveden na identičan način kao testiranje EEPROM biblioteke, u istom programu za razvoj programske podrške. Uz to, za testiranje NFC NXP biblioteke također su korištene određene funkcionalnosti aplikacije NTAG I^2C demo. Nakon što je utvrđeno da svaka pojedinačna funkcija radi uspješno, bilo je potrebno implementirati te testirati *pass-through* način rada NT3H2211. Na slici 5.3. prikazana je shema spajanja mikroupravljača i NT3H2211.



Sl. 5.3. Shema spajanja mikroupravljača i NT3H2211.

Na slici 5.4. prikazano je spajanje mikroupravljača s NT3H2211 prilikom testiranja.

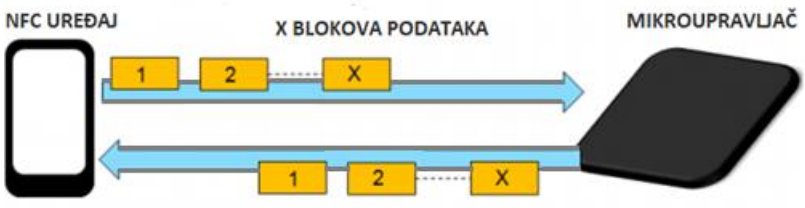


Sl. 5.4. Spajanje mikroupravljača s NT3H2211 prilikom testiranja.

5.2.1. Testiranje pass-through način rada

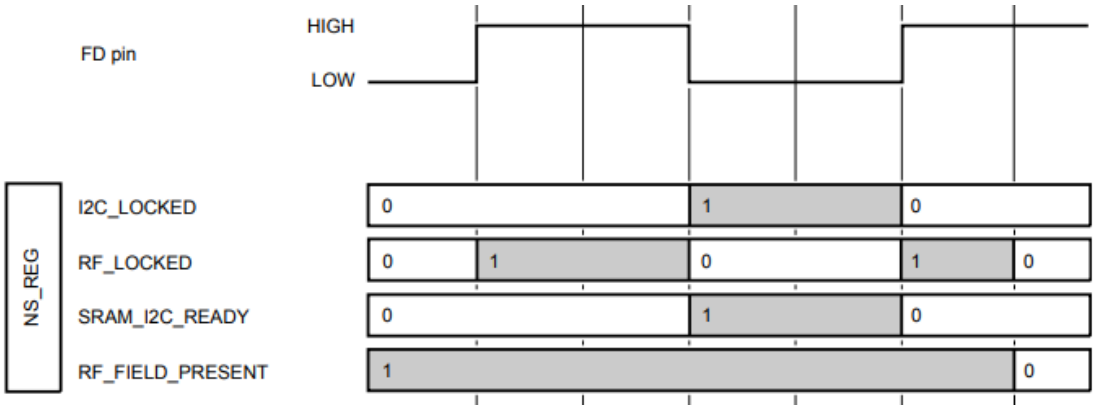
Ovaj način rada koristi se ukoliko je potrebno prenijeti veliku količinu podataka iz jednog sučelja u drugo sučelje. Prijenos podataka odvija se na način da jedno sučelje (NFC ili I^2C, ovisi koje je odabrano putem sesijskog registra) šalje blokove od 64 bajtova u SRAM, koji je u ovom načinu rada korišten kao međuspremnik. SRAM se koristi kao međuspremnik iz razloga što je veličine 64 bajta, u njega je omogućen neograničen broj zapisivanja podataka i omogućava korištenje jednostavnog mehanizma rukovanja (engl. *handshake*) između dva sučelja. Podatci se mogu slati s NFC sučelja prema I^2C sučelju ili u drugome smjeru, ovisno na koju je vrijednost postavljen bit TRANSFER_DIR sesijskog registra. Ako mu je vrijednost 1, podatci se šalju s NFC sučelja prema I^2C sučelju, a ukoliko je vrijednost 0 prijenos se odvija u drugom smjeru. Oba sučelja pristupaju SRAM međuspremniku na isti način. *Pass-through* način rada se omogućuje na način da se bit sesijskog registra PTHRU_ON_OFF postavi na vrijednost 1.

Za testiranje ovog načina rada korištena je NTAG I^2C demo aplikacija te njena značajka *SRAM speed test*, koja putem NFC sučelja šalje blokove od 64 bajta prema NT3H2211, koji te podatke putem I^2C sučelja prosljeđuje mikroupravljaču. Zatim, mikroupravljač primljene podatke putem I^2C sučelja prosljeđuje nazad prema NT3H2211, a aplikacija putem NFC sučelja čita podatke. Integritet prenešenih podataka u oba smjera se provjerava na način da je CRC32 (engl. *cyclic redundandcy check*) vrijednost nadodana na zadnji blok podataka koji se prenosi. CRC32 je izračunat za ukupne podatke (sve blokove podataka) koji se prenose. Ukoliko je CRC32 od primljenih podataka točan, aplikacija će prikazati poruku „Integrity of the data: OK“, a u suprotnom će prikazati poruku „Integrity of the data: ERROR“ za oba smjera prijenosa podataka pojedinačno. Prilikom testiranja aplikacija se ponekad samostalno ugasila te je dolazilo do *bug-a* gdje je pisalo da je NT3H2211 zaštićen lozinkom iako nije [11]. Na slici 5.5. prikazano je testiranje *pass-through* načina rada pomoću aplikacije NTAG I^2C demo.



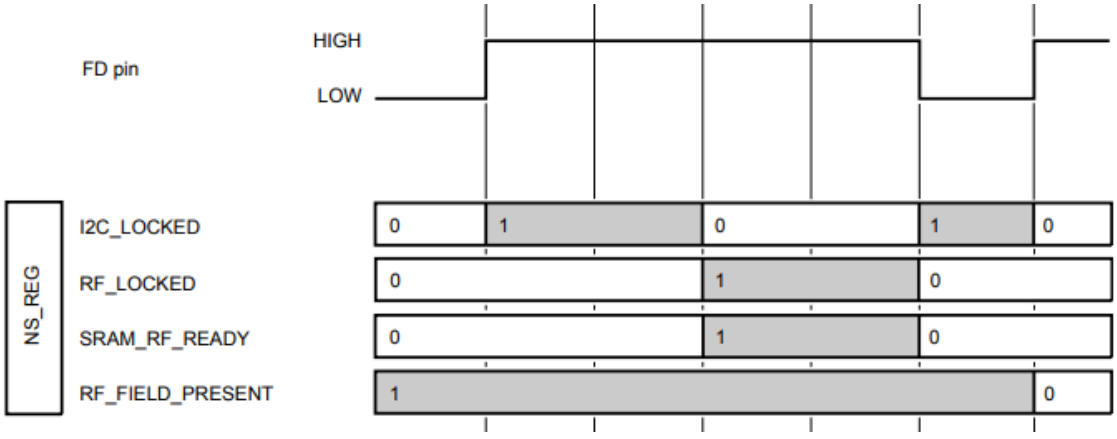
Sl. 5.5. Prikaz testiranja *pass-through* načina rada pomoću aplikacije NTAG I^2C demo [11].

Prijenos podataka s NFC prema I^{2}C sučelju odvija se na sljedeći način. Prije samog prijenosa podataka vrijednosti FD_OFF te FD_ON u sesijskom registru (oba su veličine 2 bita) moraju biti postavljene na 3. Uz to, bit SRAM_MIRROR_ON_OFF sesijskog registra mora biti postavljen na vrijednost 0, dok bit TRANSFER_DIR sesijskog registra mora biti postavljen na 1. FD_OFF i FD_ON su postavljeni na tu vrijednost iz razloga da se FD pin može koristiti kao dio mehanizma rukovanja *pass-through* načina rada. Bit SRAM_MIRROR_ON_OFF mora biti postavljen na 0, zato što ukoliko je uključen *SRAM mirror* način rada, *pass-through* način rada ne može biti uključen. Bit TRANSFER_DIR je postavljen na vrijednost 1 zbog smjera prijenosa podataka (NFC prema I^{2}C). Ukoliko NT3H2211 detektira prisutnost RF polja, bit sesijskog registra RF_FIELD_PRESENT automatski se postavlja na vrijednost 1. Na početku prijenosa podataka, na rastući brid FD pina, bit RF_LOCKED (pristup SRAM međuspremniku omogućen je samo NFC sučelju) automatski se postavlja na vrijednost 1, a potrebno je putem I^{2}C sučelja postaviti vrijednost PTHRU_ON_OFF (omogućen *pass-through* način rada) bita na vrijednost 1. Zatim, NFC sučelje započinje sa zapisom podataka u SRAM međuspremnik. Zatim, na padajući brid FD pina, vrijednost RF_LOCKED bita automatski se postavlja na 0, a vrijednosti I2C_LOCKED (pristup SRAM međuspremniku omogućen je samo I^{2}C sučelju) i SRAM_I2C_READY (podatci u SRAM međuspremniku su spremni za čitanje od strane I^{2}C sučelja) bitova automatski se postavljaju u 1. Zapisivanje 64 bajta podataka od strane NFC sučelja u međuspremnik je završeno te je pomoću I^{2}C sučelja potrebno započeti s čitanjem podataka iz SRAM međuspremnika. Na sljedeći rastući brid FD pina, bitovi I2C_LOCKED i SRAM_I2C_READY se automatski postavljaju u 0, a bit RF_LOCKED se automatski postavlja u 1 te završava čitanje podataka iz međuspremnika od strane I^{2}C sučelja [25, 27]. Na slici 5.6. prikazan je postupak prijenosa podataka *pass-through* načinom rada, s NFC prema I^{2}C sučelju.



Sl. 5.6. Postupak prijenosa podataka *pass-through* načinom rada, s NFC prema I^{2}C sučelju [25].

Prijenos podataka s I^2C prema NFC sučelju odvija se na sljedeći način. FD_OFF i FD_ON u sesijskom registru moraju biti postavljeni na 3. SRAM_MIRROR_ON_OFF mora biti postavljen na vrijednost 0, a TRANSFER_DIR mora biti postavljen na 0 zbog smjera prijenosa podataka (I^2C prema NFC). Kada NT3H2211 detektira prisutnost RF polja, bit RF_FIELD_PRESENT automatski se postavlja na vrijednost 1. Na početku prijenosa podataka, na rastući brid FD pina, bit I2C_LOCKED se automatski postavlja na vrijednost 1, a pomoću I^2C sučelja potrebno je postaviti vrijednost PTRHU_ON_OFF na 1 te započeti sa zapisivanjem 64 bajta podataka u SRAM međuspremnik. Nakon toga, kada je završeno zapisivanje podataka u međuspremnik, bit I2C_LOCKED se automatski postavlja u vrijednost 0, dok se bitovi RF_LOCKED i SRAM_RF_READY (podatci u SRAM međuspremniku su spremni za čitanje od strane NFC sučelja) automatski postavljaju u 1 te započinje čitanje podataka iz međuspremnika od strane NFC sučelja. Zatim, na padajući brid FD pina, bitovi RF_LOCKED i SRAM_RF_READY se automatski postavljaju u 0, bit I2C_LOCKED se automatski postavlja u 1, a NFC sučelje završava s čitanjem podataka iz međuspremnika. Važno je napomenuti, da je prethodno opisan način prijenosa podataka s I^2C prema NFC sučelju objašnjen prema dokumentaciji koja opisuje *pass-through* način rada te je nepotpun. Za uspješan prijenos podataka s I^2C prema NFC prilikom testiranja, na svaki rastući i padajući brid FD pina, također je bilo potrebno postavljati bitove TRANSFER_DIR te SRAM_MIRROR_ON_OFF na vrijednost 0 putem I^2C sučelja. U suprotnom, tijekom prijenosa podataka, nakon određenog vremena pojedini od ta dva bita se automatski postavi na 1 te dolazi do greške prilikom prijenosa podataka [25, 27]. Na slici 5.7. prikazan je postupak prijenosa podataka *pass-through* načinom rada, s I^2C prema NFC sučelju.



Sl. 5.7. Postupak prijenosa podataka *pass-through* načinom rada, s I^2C prema NFC sučelju [25].

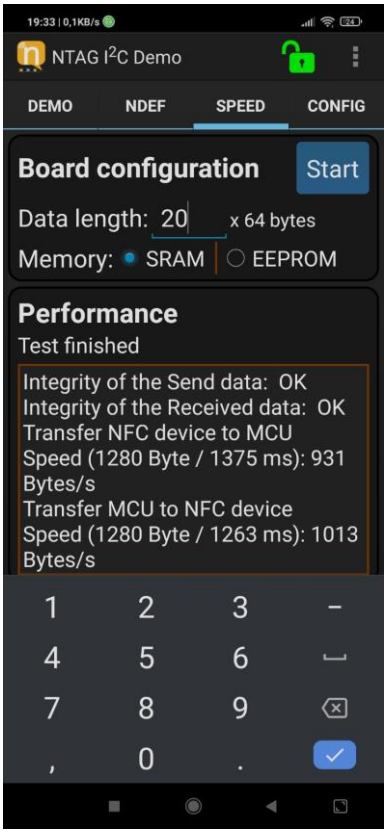
Na slici 5.8. prikazan je isječak koda pomoću kojeg je testiran *pass-through* način rada u oba smjera.

Linija Kod

```
35:     ConfigSessionRegisterFD_OFF(3);
36:     ConfigSessionRegisterFD_ON(3);
37:     ResetSessionRegisterSRAM_MIRROR_ON_OFF();
38:     SetSessionRegisterTRANSFER_DIR();
39:     while(1){
40:       if(!flag){
41:         if(GPIOB->IDR & FD){
42:           ReadSessionRegisterNS_REG(&data);
43:           if(data == RF_WRITING){
44:             SetSessionRegisterPTHRU_ON_OFF();
45:           }
46:         }
47:         if(!(GPIOB->IDR & FD)){
48:           ReadSessionRegisterNS_REG(&data);
49:           if(data == I2C_READING){
50:             ReadNBytesDataSRAM(SRAM_FIRST_BLOCK,
                dataArr+(i++)*BLOCK_SIZE,BLOCK_SIZE);
51:           }
52:         }
53:         if(GPIOB->IDR & FD){}
54:         if(i==n+1){
55:           flag=1;
56:           i=0;
57:           ResetSessionRegisterTRANSFER_DIR();
58:         }
59:       }
60:       if(flag){
61:         if(GPIOB->IDR & FD){
62:           ResetSessionRegisterTRANSFER_DIR();
63:           ResetSessionRegisterSRAM_MIRROR_ON_OFF();
64:           ReadSessionRegisterNS_REG(&data);
65:           if(data == I2C_WRITING){
66:             SetSessionRegisterPTHRU_ON_OFF();
67:             WriteNBytesDataSRAM(SRAM_FIRST_BLOCK,
                dataArr+((i++)+1)*BLOCK_SIZE,BLOCK_SIZE);
68:           }
69:         }
70:         if(!(GPIOB->IDR & FD)){
71:           ResetSessionRegisterTRANSFER_DIR();
72:           ResetSessionRegisterSRAM_MIRROR_ON_OFF();
73:         }
74:         if(i==n){
75:           flag=0;
76:           i=0;
77:           SetSessionRegisterTRANSFER_DIR();
78:         }
79:       }
80:     }
```

Sl. 5.8. Kod testiranja *pass-through* načina rada.

Na slici 5.9. prikazan je uspješan rezultat testiranja korištenjem aplikacije NTAG I^{2}C demo.



Sl. 5.9. Prikaz prozora aplikacije NTAG I^{2}C demo u *pass-through* načina rada.

6. ZAKLJUČAK

Za uspješnu izradu praktičnog dijela diplomskog rada, odnosno STM32 biblioteka za EEPROM i NTAG NFC integrirane sklopove, bilo je potrebno proučiti NFC, EEPROM te ARM tehnologije. Osim toga, bilo je potrebno pročitati svu dokumentaciju koja opisuje uređaje za koje je bilo potrebno implementirati biblioteke, CAT24M01 (EEPROM) te NT3H2211 (NTAG NFC). Uz to je bilo nužno proučiti dokumentaciju vezanu za STM32F407VET6 mikroupravljač s kojim prethodno navedeni uređaji u bibliotekama komuniciraju putem I^2C sučelja. Za razvoj navedenih biblioteka odabrana je platforma za razvoj programske podrške Keil uVision 5. Zatim je započela izrada biblioteka, prvo za CAT24M01, a onda za NT3H2211. Tijekom izrade biblioteka, svaka funkcija pojedine biblioteke bila je detaljno testirana. Za svaku funkciju obje biblioteke napisani su jasni opisi, koji uključuju detaljno objašnjenje funkcije te svakog pojedinog parametra funkcije.

EEPROM biblioteka sadrži ukupno 10 funkcija, od kojih se većina koristi za razne načine čitanja i zapisivanja u memoriju. S mikroupravljačem, EEPROM komunicira putem I^2C sučelja. NFC NXP biblioteka sastoji se od 94 funkcije, od kojih se određeni broj koristi za čitanje i zapisivanje u memoriju, a većina funkcija se odnosi na interakciju te upravljanje pojedinih registara da bi se postiglo željeno ponašanje NTAG NFC uređaja. NTAG NFC uređaj također komunicira s mikroupravljačem putem I^2C sučelja. Važna značajka ovog uređaja je *pass-through* način rada koji omogućuje brzi prijenos velike količine podataka s NFC sučelja prema I^2C sučelju i obratno te je bilo potrebno pomoću NFC NXP biblioteke uspješno implementirati i testirati ovaj način rada. Biblioteke su napisane u programskom jeziku C.

Pomoću NFC tehnologije moguće je kreirati pametne kartice za razne uporabe (poslovne posjetnice, novčane transakcije), otključavanje te zaključavanje vrata, dijeljenje multimedijskog sadržaja (slike, video), očitavanje vrijednosti senzora, punjenje IoT uređaja i slično. EEPROM tehnologija primarno je korištena za pohranu podataka. Kombiniranjem ove dvije tehnologije mogu se postići razni sustavi u kojima se željeni podatci putem NFC sučelja mogu poslati u EEPROM, gdje će biti trajno pohranjeni. Također, te podatke moguće je ponovo pročitati koristeći NFC sučelje. Medijator između dva uređaja treba biti određena vrsta mikroupravljača. Jedan primjer je očitanje vremenskih čimbenika (temperatura, tlak, vlaga) u određenoj meteorološkoj stanici svakih 6 sati kroz određeno vremensko razdoblje. Ti podatci mogu biti jednostavno očitani pomoću NFC sučelja, spremljeni u EEPROM i po potrebi analizirani.

LITERATURA

[1] EEPROM, https://whatis.techtarget.com/definition/EEPROM-electrically-erasable-programmable-read-only-memory, pristupljeno 26.4.2021.

[2] NFC tag, https://www.pcmag.com/encyclopedia/term/nfc-tag, pristupljeno 26.4.2021.

[3] NFC tag, https://electronics.howstuffworks.com/nfc-tag.htm, pristupljeno 26.4.2021.

[4] Arduino EEPROM biblioteka, https://www.arduino.cc/en/Reference/EEPROM, pristupljeno 27.4.2021.

[5] PIC EEPROM biblioteka, https://download.mikroe.com/documents/compilers/mikroc/pic/help/eeprom_library.htm, pristupljeno 27.4.2021.

[6] STM32 EEPROM biblioteka, https://github.com/nimaltd/ee24/commit/560e5932af826a25bb8249c9de58400708ed242c, pristupljeno 27.4.2021.

[7] ESP8266 EEPROM biblioteka, https://www.arduino.cc/reference/en/libraries/esp_eeprom/, pristupljeno 27.4.2021.

[8] NXP EEPROM biblioteka, https://www.nxp.com/design/microcontrollers-developer-resources/lpcopen-libraries-and-examples/lpcopen-software-development-platform-lpc15xx:LPCOPEN-SOFTWARE-FOR-LPC15XX, pristupljeno 27.4.2021.

[9] NXP NTAG NFC biblioteka, https://community.nxp.com/t5/NFC/Sofware-library-for-NTAG-I2C-plus/m-p/1033187, pristupljeno 27.4.2021.

[10] Arduino NTAG NFC biblioteka, https://github.com/LieBtrau/arduino-ntag, pristupljeno 27.4.2021.

[11] NTAG I^2C plus Exploler Kit – Android Demo, 2020.

[12] RFID vs NFC, https://www.atlasrfidstore.com/rfid-insider/rfid-vs-nfc/, pristupljeno 7.6.2021.

[13] NFC, https://www.bluebite.com/nfc/how-does-nfc-work, pristupljeno 28.4.2021.

[14] NFC, https://nfc-forum.org/what-is-nfc/about-the-technology/, pristupljeno 28.4.2021.

[15] EEPROM, https://www.electronics-notes.com/articles/electronic_components/semiconductor-ic-memory/eeprom-e2prom-technology.php, pristupljeno 28.4.2021.

[16] EEPROM, https://ecomputernotes.com/fundamental/input-output-and-memory/eeprom, pristupljeno 28.4.2021.

[17] ARM, https://whatis.techtarget.com/definition/ARM-processor, pristupljeno 29.4.2021.

[18] ARM, https://www.geeksforgeeks.org/arm-processor-and-its-features/, pristupljeno 29.4.2021.

[19] R.Grbić, PR-3 Uvod u ARM arhitekturu, Ugradbeni računalni sustavi

[20] STM32F40x Reference Manual, 2019.

[21] ARM, https://gsasindia.com/newsroom/cortex-a-cortex-r-and-cortex-m/, pristupljeno 7.6.2021.

[22] Kategorije ARM procesora, https://sirinsoftware.com/blog/the-arm-processor-a-r-and-m-categories-and-their-specifics/, pristupljeno 7.6.2021.

[23] STM32F407VET6, https://os.mbed.com/users/hudakz/code/STM32F407VET6_Hello/, pristupljeno 1.5.2021.

[24] CAT24M01 EEPROM Serial 1-Mb I^2C, 2018.

[25] NTAG I^2C plus: NFC Forum T2T with I^2C interface, password protection and energy harvesting, 2019.

[26] I^2C, https://learn.sparkfun.com/tutorials/i2c/all, pristupljeno 3.5.2021.

[27] How to use the NTAG I^2C plus for bidirectional communication, 2018.

SAŽETAK

Cilj ovog diplomskog rada bila je implementacija STM32 biblioteka za EEPROM i NTAG NFC integrirane sklopove. Uređaji koji su se koristili prilikom izrade diplomskog rada bili su mikroupravljač STM32F407VET6, EEPROM CAT24M01 te NTAG I^2C plus (NT3H2211). Za razvoj biblioteka korišteno je razvojno okruženje za razvoj programske podrške Keil uVision 5. Bilo je potrebno napraviti da EEPROM te NTAG I^2C plus u implementiranim bibliotekama komuniciraju s mikroupravljačem putem I^2C sučelja. Također, svaka funkcija unutar pojedine biblioteke morala je biti detaljno testirana. Svaka funkcija obje biblioteke sadrži detaljan opis ponašanja, uz opis svakog pojedinog parametra. EEPROM biblioteka sastoji se od 10 funkcija, od kojih je većina za čitanje i zapisivanje podataka. NFC NXP biblioteka sastoji se od 94 funkcija, gdje se dio funkcija odnosi na čitanje i zapisivanje podataka, a dio funkcija na interakciju s pojedinim registrima da bi NTAG NFC uređaj radio prema zahtjevanim specifikacijama.

Nakon uspješnog razvoja NFC NXP biblioteke, koristeći njene funkcije trebalo je implementirati *pass-through* način rada, koji služi za prijenos velike količine podataka u malome vremenu s NFC sučelja uređaja prema I^2C sučelju odnosno s I^2C sučelja prema NFC sučelju. Uz to, trebalo je testirati funkcionalnost tog načina rada pomoću NTAG I^2C demo aplikacije. Korištenjem ovog načina rada u kombinaciji s EEPROM-om, moguće je ostvariti sustave u kojima korisnik šalje podatke putem NFC sučelja svog pametnog telefona/uređaja, a ti podatci se preko I^2C sučelja i mikroupravljača spremaju u EEPROM. Također, moguće je napraviti sustave u kojima mikroupravljač očitava određene senzore, te ih sprema u EEPROM, a korisnik ih zatim očitava pomoću NFC sučelja kada su mu potrebni. Primjer takvog sustava je meteorološka stanica.

Ključne riječi: RFID, NFC, I^2C, EEPROM, NTAG, STM32.

ABSTRACT

The aim of this thesis was to implement the STM32 library for EEPROM and NTAG NFC integrated circuits. The devices used in the preparation of the thesis were the microcontroller STM32F407VET6, EEPROM CAT24M01 and NTAG I^2C plus (NT3H2211). The Keil uVision 5 software development environment was used to develop the libraries. It was necessary to create that EEPROM and NTAG I^2C plus in the implemented libraries communicate with the microcontroller via the I^2C interface. Also, every function within a particular library had to be tested in detail. Each function of the libraries contains a detailed description of the behavior along with a description of each individual parameter. The EEPROM library consists of 10 functions, most of which are for reading and writing data. The NFC NXP library consists of 94 functions, where part of the functions is related to reading and writing data, and part of the functions interact with individual registers to make the NTAG NFC device work according to demanding specifications.

After the successful development of the NFC NXP library, using its functions, a pass-through mode was to be implemented, which serves to transfer large amounts of data in a short period of time via NFC interface of the device to the I^2C interface or from the I^2C interface of the device to the NFC interface. In addition, the functionality of this mode had to be tested using NTAG I^2C demo application. By using this mode in combination with EEPROM, it is possible to create systems in which the user sends data via the NFC interface of his smartphone/device, and then this data gets transferred to the EEPROM via the I^2C interface and microcontroller. Also, it is possible to create systems in which the microcontroller reads certain sensors, then propagates data in the EEPROM and the user then reads the stored data using the NFC interface when he wants to. An example of such a system is a meteorological station.

Key words: RFID, NFC, I^2C, EEPROM, NTAG, STM32.

ŽIVOTOPIS

Matej Horvat rođen je 20. travnja 1996. godine u Našicama. Završio je Osnovnu školu Dore Pejačević, te Srednju školu Isidora Kršnjavoga u Našicama (prirodoslovno-matematičku gimnaziju). Tijekom srednjoškolskog školovanja sudjelovao je na brojnim natjecanjima iz područja matematike, informatike te fizike uz najbolje ostvareni rezultat 2. mjesto iz matematike. Trenutno je redovan student 2.godine diplomskog studija, smjera računalno inženjerstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Tečan je govornik engleskog jezika. Nagrađen je od strane FERIT-a za uspješnost u studiranju 2018. te 2021. godine. Radio je kao demonstrator iz predmeta Digitalna elektronika te kao instuktor iz stručnih predmeta iz područja računarstva i elektrotehnike u centru instrukcija Edukos. Od 2020. stipendist je tvrtke Atos.

PRILOZI

Prilozi na CD-u:

- aplikacija NTAG I^2C demo
- kod projekta STM32 libraries.