Tehničko veleučilište u Zagrebu

Stručni prijediplomski studij Mehatronika

Water Buddy

Seminarski rad

Kolegij: Komunikacijske tehnike u mehatronici

Nositelji kolegija: dr. sc. Tomislav Pavlović

Dean Fraj mag. ing. el.

Student: Karlo Mikulaco

JMBAG: 0035241479

Zagreb, veljača 2026.

SAŽETAK

Glavni cilj ovog rada je pokazati razvoj i primjenu energetski učinkovitog sustava koji služi kao podsjetnik korisniku o dnevnom unosu vode, koristeći STM32G031G8U6 mikroupravljač i NFC tehnologiju za praćenje prethodnih unosa koristeći I^2C protokol. Uz pomoć auditivno-vizualnog alarmnog podsustava, zadatak ovog uređaja je pravovremeno kroz dan obavijestiti korisnika o unosu vode u prema njegovim zadanim parametrima kao što su, dnevni broj čaša i interval unosa (pijenja vode).

Prilikom izrade seminarskog rada, uz osnovna svojstva i mogućnosti STM embedded svijeta, susreo sam se s komunikacijskim protokolima kao što su I^2C i jedno žičani NZR protokol. Naposljetku je upravo s tim mogućnostima STM32 mikroupravljača ostvaren je i ovaj rad te njegova željena funkcionalnost. Osim učenja o stranim komunikacijskim protokolima, također je cilj ovoga rada bio razviti nešto što sličnije konačnom proizvodu koji bih se mogao pronaći u rukama korisnika koji također pate od nedovoljnog unosa vode koja je neophodna za zdravlje. Kao izvor napajanja korištena je jedna ne-punjiva litijska baterija CR2032 koja u kombinaciji s niskom potrošnjom rada uređaja, ostvarenog s pomoću RTC funkcionalnosti daje dug radni vijek.

U radu je ukratko i prikazan primjer izrađene same tiskane pločice kako bi se ideja lakše prenijela u nešto približno gotovom proizvodu s što intuitivnijom uporabom.

Ključne riječi: I^2C, STM32, unos vode, NFC, niska potrošnja

Sadržaj

1. Rad uređaja

1.1.Princip rada uređaja

Kako bi se sustav učinio što intuitivnijim i jednostavnim za uporabu, cijelo sučelje je podijeljeno na dva dijela, vizualno-auditivni i onaj taktilni. Vizualno-auditivni dio služi kako bi se korisniku pokazala nekakva informacija i taktilni dio koji služi kako bi korisnik unio upravo neku informaciju u sustav, u ovom je slučaju to doznaka da je popijena čaša vode. RGB LED lampice predstavljaju vizualni dio korisničkog sučelja u obliku osam RGB LED lampica s kojima uređaj „komunicira“ s korisnikom o tome koliko čaša treba popit/koliko je popio. Jedna lampica predstavlja jednu čašu vode (250mL, 500mL….), početno stanje pri prvom paljenju je četiri čaše dnevno te je početni interval namješten na svakih sat vremena.

Prvi je korak dovesti izvor napajanja u sustav što se postiže stavljanjem baterije CR2032 na njeno predviđeno mjesto (držač baterije). Spajanjem baterije dovodi se napon i sustav se pokreće, čime je korisnik dočekan s zvukom zujalice i paljenjem četiri od osam RGB LED lampica (što predstavlja početnu točku i default sustava od četiri čaše), korisnik sada može koristiti lijevo tipkalo za inkrementaciju kroz svih osam LED lampica dok nije zadovoljan s brojem čaša koje bih htio popiti dnevno (četiri čaše predstavlja četiri dnevna alarma) te kada je zadovoljan na sljedeći korak postavka se pristupa pritiskom desnog tipkala pri čemu se aktivira postavljanje dnevnog intervala, tj. interval nakon kojega će zvoniti svaki alarm. Nalik odabiru broja dnevnih čaša vode, sa lijevim tipkalom se bira interval na temelju da jedna LED lampica predstavlja petnaest minuta. Nakon odabira željenog dnevnog intervala alarma, korisnik je dočekan s još jednim postavka no ovaj put u obliku upaljenih svih osam LED lampica, no različitih boja. Dok tirkizna predstavlja željeni dnevni broj čaša vode, crvena interval individualnih alarma, treće postavke u nizu sa svim upaljenim lampicama predstavljaju koliko je trenutno sati te su ove postavke jako nužne kako bi se spriječilo slučajno paljenje alarma usred noći (default noć traje od 22 do 10). Crvene lampice trećeg niza postavki predstavljaju jutarnji period od 00 do 07, zelene popodnevni od 08 do 15 te naposlijetku i plave koje predstavljaju večernji period od 16 do 23. Korisnik lijevim tipkalom bira i postavlja desnim koliko je trenutno sati u trenutku postavljanja alarma. Ako je postavljen dnevni alarm unutar vremena rada (izvan 22 do 10 perioda), prvi se alarm pokreće pet sekundi nakon postavljanja svih postavki. Korisnik svaki alarm može ugasiti samo tako da pritisak lijevog gumba poprati s pritiskom desnog unutar jedne sekunde kako bi se uspješno alarm ugasio i resetirao na ranije zadani interval.

U slučaju pritiska desnog tipkala alarm se ne gasi već se „snooza“ na deset sekundi, nakon čega opet počinje zvoniti. Zvuk alarma i njegovo oglašavanje se ponavlja svakih četiri sekunde te se njegovim uspješnim gašenjem broj popijenih čaša inkrementira i sprema u obliku varijable na internu memoriju. Cilj rada uređaja je obavijestiti korisnika kroz dan postavljenim alarmima da popije vodu ne bi li inače to zaboravio.

1.2.Zašto NFC?

NFC tj. Near Field Communication, igra veliku ulogu ovog rada te je jedan od glavnih razloga zašto ovaj rad izgleda i funkcionira na odabrani način. Kako je jedna od značajki ovog sustava inkrementacija broja popijenih čaša vode od strane korisnika pritiskom tipkala, te kako bi se zadovoljio glavni kriteriji za pisanje ovog seminarskog rada, stvorila se potreba za uporabom nekakvog komunikacijskog protokola, upravo u svrhu da se ta informacija o popijenim čašama vode prenese korisniku na neki drugi uređaj (npr. mobitel prvenstveno).

NFC, uz druge bežične komunikacijske protokole poput Wi-Fi-a i Bluetootha koje je kompliciranije i skuplje za integrirati u sustav uz to da im je potrošnja znatno veća od one koju nudi NFC, no također im je domet znatno veći jer su prave bežične komunikacije za razliku od NFC-a, koji ima mali domet jer njegov sustav ne sadrži aktivni odašiljač. Kako za ovaj uređaj nije potreban veliki domet, tehnologija i specifikacije koje nudi NFC su više nego dovoljne da se zadovolje svi uvjeti rada. Također je jedna od bitnijih prednosti NFC-a za razliku od prethodno navedenih komunikacijskih protokola ona da on najčešće ne zahtijeva nikakvu dodatnu instaliranu aplikaciju i podršku na sustavu koji obavlja očitanje podataka (npr. mobitel), budući da većina sustava već dolazi s instaliranom podrškom za jednostavna NFC očitanja. No glavna i najbitnija prednost izbora NFC-a naspram drugih komunikacijskih protokola je njegova iznimno mala potrošnja, te je izabran integrirani sklop za ovaj rad, koji podržava očitanje i pohranu podataka bez vanjskog napona (baterija), već se sva energija crpi iz antene sustava koji izvršava čitanje. Antena ovog sustava je izvedena oko dugog kontinuiranog voda specifičnog induktiviteta u obliku 2D zavojnice.

Oblik uređaja je diktiran oblikom i dimenzijom antene induktiviteta potrebnog da se NFC antena može se detektirati s drugim uređajem te prenijeti podatke. Veličina i oblik su također određeni proizvoljno kako bi uređaj dimenzijski bio što sličniji kreditnoj kartici kako bi cijeli sustav mogao stati u korisnikov novčanik.

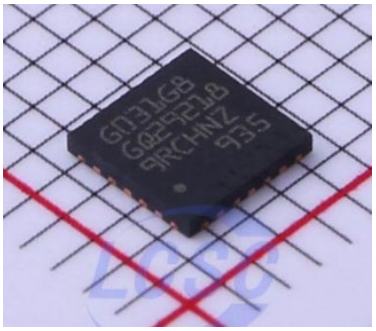
2. Sastavne komponente

2.1.STM32G031G8U6

Pokraj velikog izbora mikroupravljača STM32 podrijetla teško se odlučiti koji odabrati, no ja sam se odlučio upravo na STM32G030F6P6 zbog svojeg malog pakiranja (UFQFN-28) koje je idealno za ostvarivanje PCB-a [1]manjeg formata te i dalje mogućeg ručnog lemljenja pri sastavljanju za razlike od drugih manjih i gušćih pakiranja (BGA). Ovaj mikroupravljač temelji se na ARM Cortex-M0+ procesorskoj jezgri, koja omogućuje rad do 64 MHz. Zahvaljujući širokom naponskom rasponu od 1.7 do 3.6 V i kompaktnim dimenzijama, idealno kombinira energetsku učinkovitost s pouzdanosti. Zbog minimalne kompleksnosti s strane samoga programa uređaja, količina dostupne flash memorije nije bila prioritet pri odlučivanju među izborima, no unatoč tome, s ovim izborom ostvareno je do 64 KB prostora.

Uz dostupni RTC, za koji je ključno napomenuti da je u sustav dodat vanjski LSE kristal kako bi se ostvarila velika preciznost između pojedinih alarma, te je također je ostvarena potrošnja u veličinama μA[2] upravo zbog uporabe RTC-a. Naposlijetku, kako bi se korisnika obavijestilo o potrebi pijenja vode, koriste se digitalni izlazi u kombinaciji s timer-ima PWM funkcija kako bi se upravljalo vizualnim i auditivnim alarmima.

Uz sve navedeno, glavni razlog odabira ovog mikroupravljača je njegova široka dostupnost na internetu, sa pouzdanih stranica poput Mouser Electronics ili upitnih kao što su LCSC i to za cijenu izrazito prihvatljivog raspona (0,85 do 1,5 €). Također je vrijedno napomenuti da uz postojeći debugger/programatora kao što su ST-LINK, J-Link, J-Link EDU Mini, ovaj mikroupravljač je moguće koristiti bez ikakvih potrebnih pasivnih komponenti što znatno pojednostavljuje integraciju u završni uređaj.



Slika 1 STM32G031G8U6

[1] Printed Circuit Board
[2] Mikroamper

2.2. ST25DV04K

ST25DV04K je jedan od dinamičkih NFC/RFID „tag“ integriranih sklopova u ST25 seriji uređaja, specifično verzija korištena u ovom radu je u stanju pohraniti do 4-Kbit (512 bajta) na svoj interni EEPROM[3]. U ovom radu korištena je specifična verzija ST25DV04K-JFR6D3 koja je u pakiranju UFDFPN12 kako bi se postigla cijeloukupna minimizacija cijelog uređaja, no bitnija stavka ovog pakiranje je to što se nudi dvanaest pinsko pakiranje u malom formatu za razliku od osnovnih osam od čega su dodatnih četiri zaslužni za još nižu potrošnju od osnove verzije ST25 sklopa. Budući da ovaj sklop sadrži EEPROM tj. NVM[4], svi podatci preneseni na ovaj NFC tag, ostaju pohranjeni i bez prisutstva napajanja, te ih je moguće pročitati isto tako.

Komunikacijski protokol zaslužan za upravljanje (pisanje/čitanje) ST25DV04K integriranim sklopom je upravo I^2C[5]. Riječ je o široko prihvaćenom serijskom komunikacijskom protokolu koji omogućuje povezivanje više uređaja (tzv. master-slave arhitektura) putem samo dvije sabirnice: serijske podatkovne linije (SDA) i serijske taktne linije (SCL). Upravo ta jednostavnost i mali broj potrebnih pinova čini ga idealnim za ugrađene sustave gdje je optimizacija prostora ključna, što se savršeno nadopunjuje s odabirom kompaktnog UFDFPN12 pakiranja spomenutog sklopa.

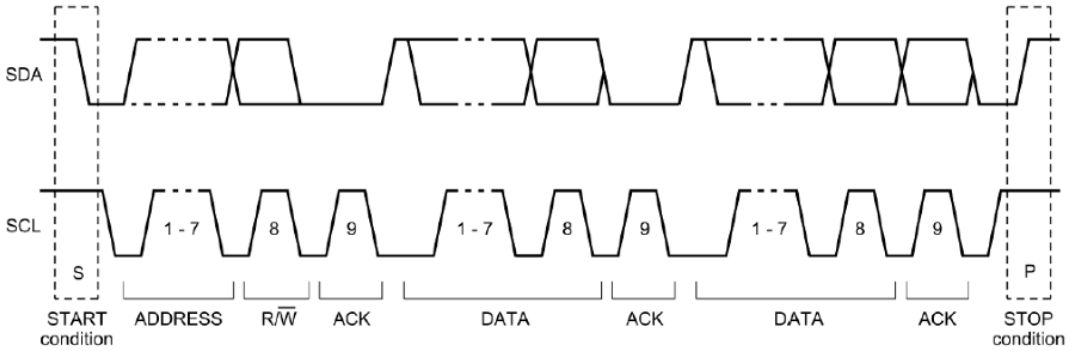
U kontekstu ovog rada, ST25DV04K konfiguriran je kao slave uređaj na I^2C sabirnici, dok mikroupravljač (opisan ranije) preuzima ulogu mastera. Adresa tog slave uređaja hardverski je definirana, a koristi se za slanje naredbi za upis i očitanje podataka u internu EEPROM memoriju.

Posebnost ST25 serije, pa tako i modela DV04K, jest u tome što I^2C sučelje nije namijenjeno isključivo komunikaciji s mikroupravljačem, već ono omogućuje i tzv. energy-harvesting (prikupljanje energije) putem RF polja. To znači da, iako se podatcima upravlja putem I^2C-a, sam sklop može biti napajan bežično putem NFC-a, što dodatno pridonosi energetskoj učinkovitosti cjelokupnog sustava. Energy harvesting u detalje nije opisan u ovom radu budući da je to početno stanje samog sklopa te nije potrebna dodatna konfiguracija putem I^2C.

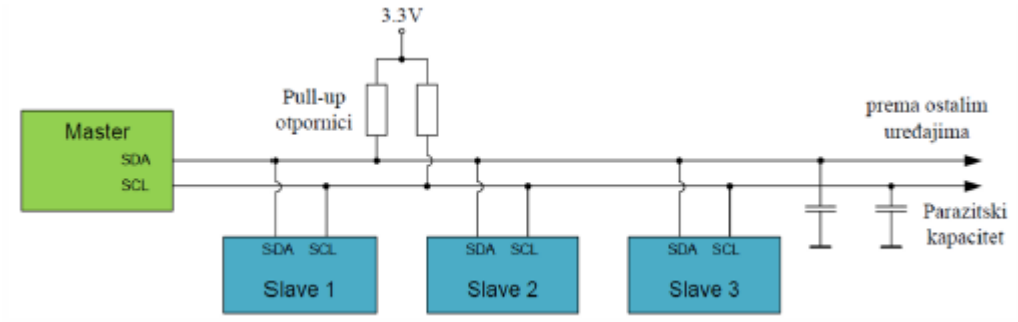
Brzina prijenosa podataka putem I^2C sučelja kod ovog sklopa podržava standardni (do 100 kHz) i brzi (do 1 MHz) mod, čime se postiže dovoljna brzina za prijenos manjih količina podataka kakve se očekuju u ovom uređaju. Sve naredbe, od inicijalizacije do potvrde zapisa, provode se slanjem specifičnih sekvenci bajtova kojima se upravlja stanjem na SDA i SCL linijama, a koje mikroupravljač generira u skladu s I^2C protokolom.

[3] Electrically Erasable Programmable Read-Only Memory
[4] Non-Volatile Memory
[5] Inter-Integrated Circuit

Jedno od temeljnih svojstava I2C sučelja jest njegova open-drain (otvoreni odvod) konfiguracija izlaza. To znači da uređaji na sabirnici mogu povući naponsku razinu na nulu (logička 0), ali je ne mogu samostalno podići na visoku razinu (logička 1). Upravo iz tog razloga, nužno je na obje linije (SDA i SCL) postaviti vanjske pull-up otpornike koji će, u stanju mirovanja, održavati sabirnicu na naponu napajanja (logička 1). Ovi otpornici osiguravaju ispravnu logičku razliku i spriječavaju „lebdeće“ (floating) stanje sabirnice koje bi dovelo do grešaka u komunikaciji. Vrijednost tih otpornika mora biti pažljivo odabrana – dovoljno niska da osigura dovoljno brzo punjenje parazitnih kapaciteta (što je ključno za veće brzine prijenosa), ali istovremeno dovoljno visoka da uređaji mogu savladati struju koja teče kroz njih kako bi liniju uspješno povukli na nulu.



Slika 2 I2C podatkovni okvir



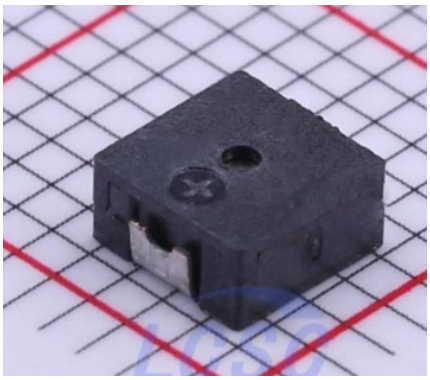
Slika 3 Prikaz I2C protokola i spajanja Master-Slave uređaja na sabirnicu

2.3. Alarm indikatori

Kako bih uređaj prenio korisniku informaciju da je potrebno popiti vode te koliko je još mora popiti, korištene su komponente koje omogućuju brz, jasan i učinkoviti prijenos jednostavnih informacija. Budući da je cilj projekta da bude što jednostavniji i intuitivnija uporaba, prijenos tih informacija je ostvaren uporabom sljedećih komponenti (u malenom SMD pakiranju zbog cilja postizanja što manjeg uređaja):

2.3.1. Zujalica

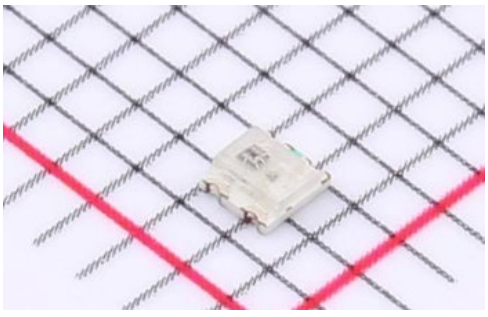
Auditivni podražaj, koji služi kako bi se korisnika obavijestilo da je vrijeme za popiti vode ostvaren je uporabom pasivne zujalice. Pojam pasivnosti u se odnosi na činjenicu da ova komponenta ne sadrži nikakvu internu elektroniku već se ponaša kao zvučnik te stvaranje zvuka ovisi isključivo o vanjskom podražaju (PWM).



Slika 4 FUET-4020; Zujalica

2.3.2. LED

Vizualni podražaj ostvaren je uporabom osam adresabilnih RGB[6] LED-ica čije se boje lako mogu povezati sa određenim postavkama i načinima rada uređaja (npr. tirkizno za čaše vode te crveno za alarm).



Slika 5 XL-1615RGBC-2812B-S

[6] Red Green Blue (odnosi se na boju emisije)

2.3.2.1. Adresabilne LED-ice

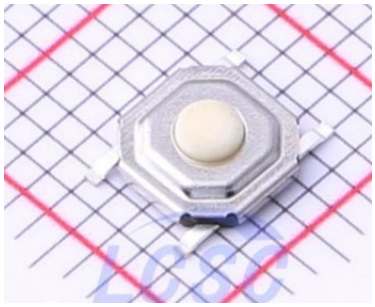
U ovom sustavu korištene su tzv. adresabilne WS2812 LED lampice malog pakiranja koje za razliku od standardnih RGB LED-ica zahtijevaju uz napajanje samo jedan vod (žicu) za komunikaciju kojom se određuje boja i najbitnije, specifično koja LED-ica svijetli u tom trenutku. Za ovaj uređaj jako je bitno da se održi malen format uz što više funkcija, stoga ovaj izbor.

Naziv "adresabilne" dolazi od činjenice da se svakoj LED diodi u lancu može pristupiti zasebno putem jedinstvenog redoslijeda u nizu. Iako fizički ne posjeduju klasičnu adresu poput memorijske lokacije, njihova pozicija u serijskom spoju određuje kojoj se od njih šalje određeni podatak. Mikroupravljač šalje niz podataka (kontrolne signale) u kojem je točno definirano koja će LED dioda primiti koju vrijednost boje. Drugim riječima, prva primljena naredba namijenjena je prvoj LED diodi u nizu, druga naredba drugoj, i tako dalje. To omogućuje neovisno upravljanje bojom i svjetlinom svake pojedine LED jedinice unutar lanca, bez potrebe za zasebnim upravljačkim linijama za svaku od njih.

Za razliku od konvencionalnih RGB dioda koje za promjenu boje zahtijevaju tri zasebna upravljačka voda (jedan za svaku boju), WS2812 serija koristi jednolinijski serijski protokol temeljen na NZR (Non-Return-to-Zero). NZR protokol označava način prijenosa podataka pri kojem se signal ne vraća na nulu (referentnu razinu) između prijenosa dva bita. Razina napona na komunikacijskom vodu tijekom cijelog trajanja bita je konstantna i ovisi isključivo o tome prenosi li se logička '1' ili logička '0'. Kod WS2812 LED dioda, logička '1' definira se visokim naponskim stanjem u trajanju od otprilike 0.8 µs, nakon čega slijedi nisko stanje u trajanju od 0.45 µs. S druge strane, logička '0' definira se visokim stanjem u trajanju od 0.4 µs i niskim stanjem od 0.85 µs. Ovaj precizan vremenski slijed (timing) omogućuje prijenos 24-bitne informacije (8 bita za zelenu, 8 bita za crvenu i 8 bita za plavu boju) za svaku LED diodu u nizu. Nakon što se podaci pošalju za sve LED jedinice, mikrokoupravljač šalje signal reseta (nisko stanje u trajanju od najmanje 50 µs) čime se lanac priprema za sljedeći prijenos podataka.

2.3.3. Tipkala

Kako bi korisnik mogao inkrementirati da je popio čašu vode, tj. ugasiti alarm, na uređaj su dodata dva tipkala malog SMD formata.

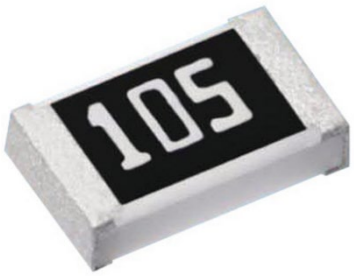


Slika 6 SMD-4P,5.2x5.2mm x1.5mm; SMD tipkalo

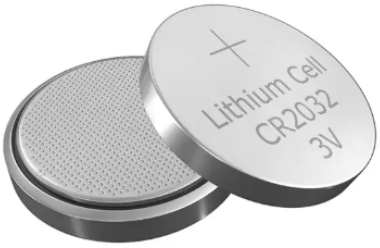
2.4.Ostale komponente

Osim navedenih komponenti koje pružaju vizualni/auditivni podražaj i povratnu informaciju prilikom pritiska tipkala, kako bi sve funkcioniralo kao cjelina potrebne su i ostale komponente poput: kondenzatora, otpornika, mosfeta/tranzistora, dioda, kristala, držač baterije, Schmitt trigger podsustav.

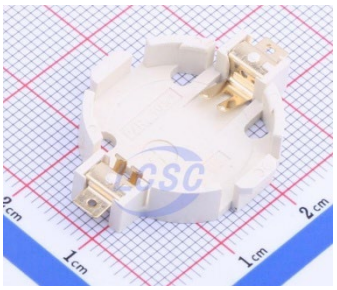
Slika 8 keramički SMD kondenzatori



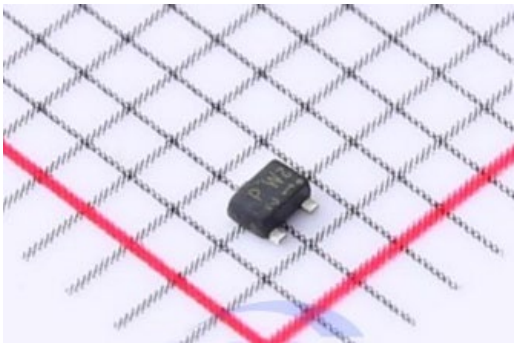
Slika 7 0603 SMD otpornik



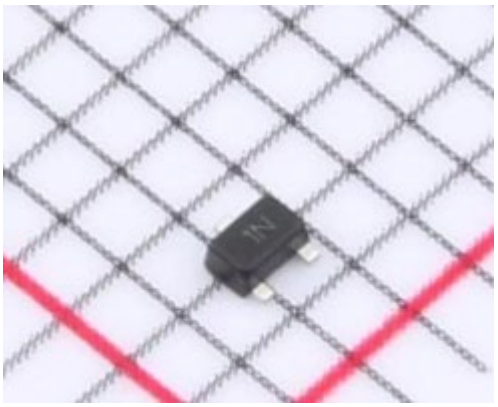
Slika 10 Baterija tip CR2032



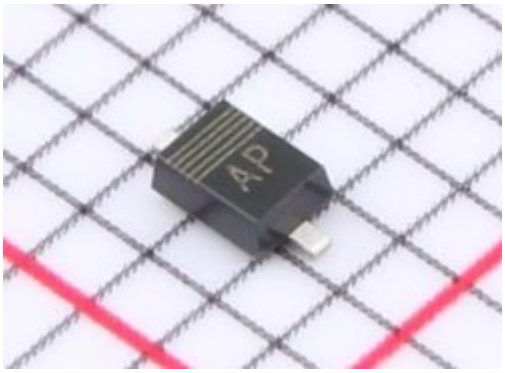
Slika 9 Držač za CR2032 tip baterije



Slika 11 P-kanalni mosfet



Slika 13 NPN tranzistor



Slika 12 Schottky dioda

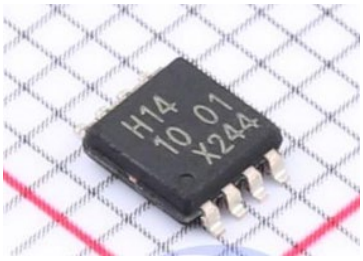
2.4.1. Schmitt trigger podsusatv

Pri radu s mehaničkim tipkalima, neizbježno se javlja pojava zvana bouncing (titranje kontakata). Naime, zbog mehaničke prirode tipkala, pri njegovom pritisku ili otpuštanju, kontakti se ne zatvore ili otvore trenutno i čisto, već nekoliko milisekundi titraju prije nego što se stabiliziraju u konačnom stanju. Posljedica toga je da umjesto jednog, mikrokontroler registrira više uzastopnih pritisaka, što dovodi do neželjenog ponašanja uređaja i netočnog očitavanja korisnikovih namjera.

Kako bi se taj problem u potpunosti eliminirao, u ovom je radu primijenjen dvostupanjski pristup koji kombinira hardversko i softversko filtriranje. Dok softverski debouncing (npr. u obliku kašnjenja ili provjera stanja) pruža dodatnu sigurnost, glavninu posla preuzima hardverska komponenta u obliku Schmitt triggera.

Schmitt trigger je komparator s histerezom, što znači da ima dva različita naponska praga: jedan za uključivanje (pri rastućem naponu) i jedan za isključivanje (pri padajućem naponu). Ova histereza sprječava neželjene promjene izlaznog stanja uslijed malih naponskih oscilacija koje nastaju upravo zbog mehaničkog titranja kontakata. Drugim riječima, Schmitt trigger "čisti" signal neposredno na ulazu, pretvarajući potencijalno šumni i oscilirajući signal iz tipkala u čisti, oštri digitalni signal koji mikroupravljač može pouzdano interpretirati.

Ovakav pristup, gdje se problem rješava već na razini hardvera prije nego što signal uopće stigne do mikroupravljača, osigurava maksimalnu pouzdanost i brzinu odziva. Time se postiže da uređaj reagira točno onako kako korisnik očekuje svaki pritisak tipke registrira se samo jednom i bez iznimke.

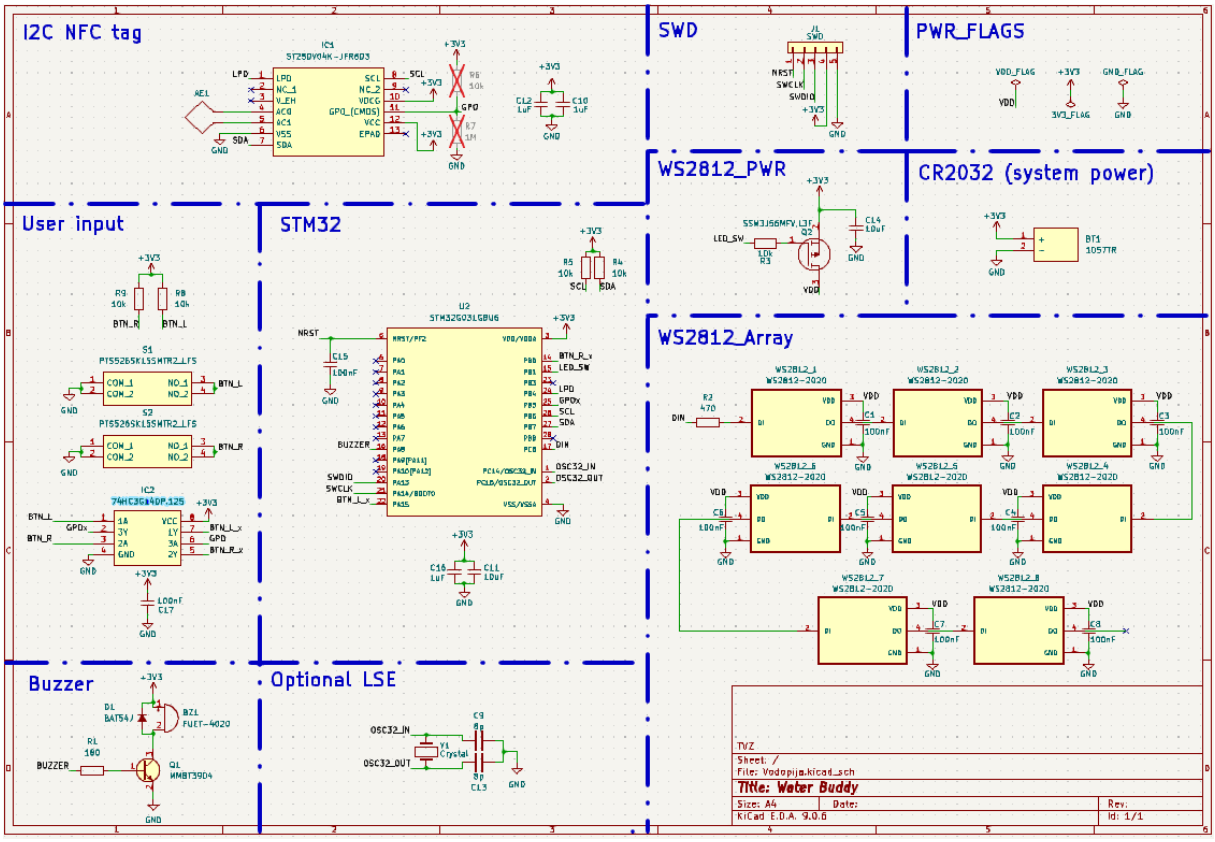


Slika 14 Trostruko invertirajući schmittov prekidač

3. Shema spajanja

Za postizanje željenog rada ovog uređaja, potrebno je sve elektroničke komponente međusobno spojiti vodovima, čime se omogućuje povezanost samog mikroupravljača s radnim dijelovima (LED lampica, zujalica, tipkala).

Shema spajanja je konstruirana digitalnim pristupom, putem „KiCAD“ alata.



Slika 15 Shematski prikaz spajanja

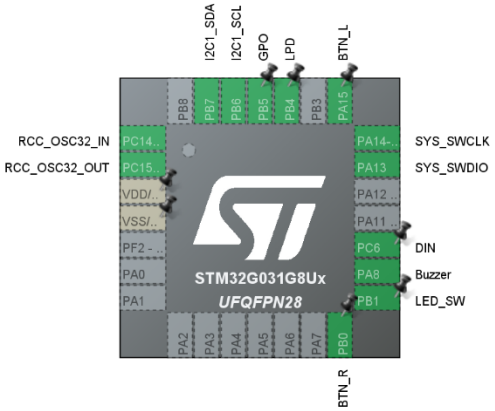
4. STM32CubeIDE

4.1.Postavljanje okruženja

Razvojno okruženje s pomoću kojeg je računalni dio ovog rada ostvaren je STM32CubeIDE. Kroz sljedećih par točaka raspravljati će se kako i zašto su tako postavljeni dijelovi okruženja.

4.1.1. Odabir željenog mikroupravljača i postavljanje funkcija izlaza/ulaza

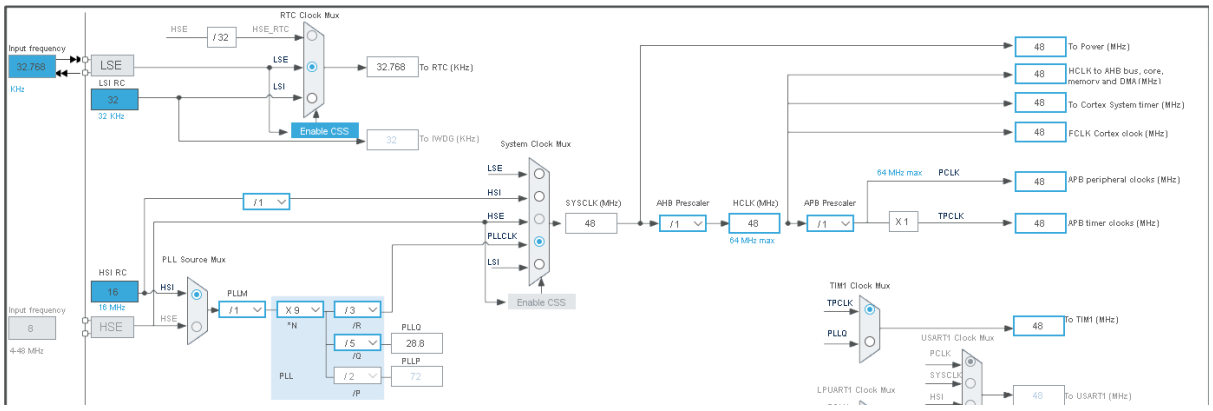
Slika u privitku prikazuje odabrani spomenuti mikroupravljač s imenovanim ulazima i izlazima na način koji odgovara onome opisanom shematskim prikazom.



Slika 16 Odabrani MCU u IDE-u

4.1.2. CLOCK

Cilj je postići malenu potrošnju uz efikasni precizni rad, stoga je u postavkama clock-a frekvencija RC oscilator s pomoću prescaler[7]-a umanjena na 48 MHz.

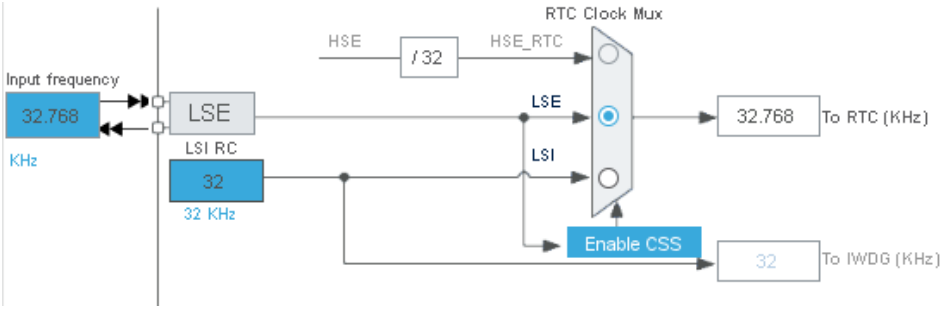


Slika 17 HSI Clock postavke

Slika 17 opisuje konfiguraciju HSI takta tako da je izlazna frekvencija ona SYSCLK-a sa djeliteljem frekvencije namještenog na 1 (nepromijenjeno).

[7] Djelitelj frekvencije

Slika 18 pokazuje postavke RTC[8]-a tako da se koristi LSE umjesto LSI



Slika 18 Konfiguracija LSE

4.1.3. SYS Mode and Configuration

Prilikom odabira „Serial Wire“ opcije automatski se generiraju dosutpni ulazi na MCU-u odgovorni za komunikaciju s vanjskom jedinicom odgovornom za programiranje.

4.1.4. Timer-i

Pod postavkama Timer-a odabrani su TIM1 i TIM3 zbog lakšeg naknadnog slaganja samog PCB dizajn. TIM1 i TIM3 su postavljeni s prescalers na 0 no njihov Counter Period (ARR) je postavljen različito ovisno o primjeni.

Budući da zujaliza radi najefikasnije na 2 kHz, njen TIM1 je postavljen na 23999 što nakon pretvorbe odgovara tih poželjnih 2 kHz.

TIM3 je zadužen za generiranje takta vezanog uz komunikacijski protokol za WS2812 ledice, kako one rade na 800 kHz signalu podataka, onda sam koristio ARR od 59 što je otprilike upravo ta brzina.

[8] Real Time Clock

4.1.5. DMA

Za upravljanje WS2812 LED diodama, ključno je generirati signale točno određenih vremenskih trajanja (timing) kakvi su opisani u prethodnom poglavlju. Budući da je riječ o zahtjevnom vremenskom slijedu (reda veličine mikrosekundi), softversko generiranje takvih signala (engl. bit-banging) bilo bi nepouzdano i opterećivalo bi procesor u mjeri koja bi onemogućila obavljanje drugih zadataka.

Iz tog razloga, primijenjeno je hardverski potpomognuto generiranje signala korištenjem timera i DMA kontrolera. DMA (Direct Memory Access) kontroler omogućuje prijenos podataka između periferija i memorije bez sudjelovanja procesorske jezgre. U konkretnom slučaju, DMA je zadužen za automatsko slanje podataka o bojama LED dioda iz memorije (RAM) u registar timera (TIM3), i to izravno, bez opterećenja CPU-a. CPU samo inicijalno postavi podatke u memoriju i pokrene DMA prijenos, a potom može obavljati druge zadatke dok DMA samostalno prosljeđuje podatke timeru. Bez DMA-a, procesor bi morao za svaki pojedini bit (a njih je 24 po LED diodi, odnosno 192 bita za osam LED dioda) ručno ažurirati vrijednost timera, što bi bilo neučinkovito i sklono greškama.

4.1.6. RTC

Za RTC timer su odabrane i konfigurirane postavke tako da se periodično generiraju prekidi uz pomoć dva alarma, Alarm A i Alarm B, koja su oba postavljena da igrnoriraju dan u tjednu te da se samo upale kada se poklope sati. Alarm A služi za dnevno pokretanje prvog alarma, u 10 sati ujutro a B alarm služi kako bi se iza 22 navečer stavila zabrana na alarme. Ove postavke su lako podesive kroz varijable u samom CubeMX-u.

Uz osnovne alarme A i B koristi se i prekidni alarm koji se koristi za stvaranje prekida u intervalima koje je korisnik ranije odabrao u postavkama drugim po redu. Također je uz pomoć istog internog sustava, napravljeno da funkcije zadužene za alarm, ne koriste delay i pauze već se cijeli sustav stavlja u najniži način potrošnje kako bi se produljio vijek baterije.

4.1.7. GPIO[9] postavke

Pod izbornikom NVIC[10] postavljena je postavka „EXTI line 4 to 15 interrupts“ , EXTI line 0 and line 1 interrupts“ te „RTC and TAMP interrupts throught EXTI lines 19 and 21“ što omogućuje da se tipkala, RTC i NFC koriste kao izvor vanjske prekidne rutine .

[9] General Purpose Input/Output
[10] Nested Vectored Interrupt Controller

5. Biblioteke

Nakon što su se postavile željene navedene postavke, potrebno je stvoriti generirati dio programa „main“ u odabranom programskom jeziku (u ovom slučaju to je C). U nastavku ovoga poglavlja, najviše je pažnje posvećeno objašnjenju rada ručno pisanih biblioteka vezanih za rad. Kako bi ovaj uređaj bio što intuitivniji za koristit, glavni kod je izrazito velik i pre opsežan za obuhvatiti u jednom ovakvom radu, te se on u cijelosti nalazi na GitHub linku na stranici s linkovima na kraju ovog dokumenta. Kako bi se uspješno i efikasno integrirale funckije vezane za dodatne komunikacijske protokole opisane do sada u ovom radu, funckije su svedene u istoimene ručno pisane biblioteke koje se kasnije ubacuju u glavni kod (main).

5.1. ST25DV.c

Datoteka st25dv.c implementira upravljački sloj za NFC/RFID tag ST25DV04K putem I^2C sučelja. Biblioteka pruža apstrakciju komunikacije s EEPROM memorijom ST25D-om te omogućuje pohranu i dohvat podataka u formatu čitljivom NFC uređajima. Sastoji se od nekoliko funkcionalnih cjelina: osnovne funkcije za inicijalizaciju (ST25DV_Init) i provjeru spremnosti (ST25DV_WaitReady), funkcije za pristup memoriji (ST25DV_WriteBytes i ST25DV_ReadBytes) koje omogućuju čitanje i upis proizvoljnog broja bajtova, pri čemu se veliki upisi automatski dijele na manje blokove zbog ograničenja I2C protokola, te funkcije za formatiranje podataka u NDEF strukturu (ST25DV_WriteNDEFText) i pomoćna funkcija ST25DV_SendCasaCount specifična za ovaj projekt.

Komunikacija se odvija putem I2C protokola, gdje svaki upis zahtijeva slanje adrese (2 bajta) nakon koje slijede podaci. Zbog vremena potrebnog za upis u EEPROM, nakon svakog upisa poziva se ST25DV_WaitReady koja provjerava je li čip spreman za novu operaciju. Čitanje se izvodi slanjem adrese nakon čega slijedi očitavanje podataka. Funkcija ST25DV_WriteNDEFText formatira tekst prema NFC specifikaciji u TLV formatu s jezičnom oznakom "hr", čime su podaci automatski prepoznatljivi pametnim telefonima. Biblioteka koristi HAL biblioteke za I^2C komunikaciju, a upisi se izvode u normal modu kako bi se spriječilo neželjeno ponavljanje podataka, uz maksimalnu veličinu pojedinačnog upisa definiranu konstantom CHUNK_MAX.

```c
/*
 * st25dv.c
 *
 *  Created on: Dec 7, 2025
 *      Author: Admin
 */
/**
 * @file    st25dv.c
 * @brief   ST25DV NFC driver
 */

/* Includes ------------------------------------------------------------------*/
#include "st25dv.h"

/* Private variables ---------------------------------------------------------*/
static I2C_HandleTypeDef *hi2c_st25dv = NULL;  /* Pokazivač na I2C strukturu */

/* Private function prototypes -----------------------------------------------*/
static HAL_StatusTypeDef st25dv_write_chunk(uint16_t addr, uint8_t *data,
        uint16_t len, uint32_t timeout);  /* Upis jednog bloka u memoriju */

/* Public functions ----------------------------------------------------------*/

/**
 * @brief   Inicijalizira ST25DV uređaj
 * @param   hi2c - Pokazivač na I2C strukturu
 * @retval  HAL_OK ako je uređaj spreman, inače HAL_ERROR
 */
HAL_StatusTypeDef ST25DV_Init(I2C_HandleTypeDef *hi2c) {
    if (hi2c == NULL) {
        return HAL_ERROR;
    }

    hi2c_st25dv = hi2c;

    // Provjera prisutnosti uređaja na I2C sabirnici
    if (HAL_I2C_IsDeviceReady(hi2c_st25dv, ST25DV_DEV_ADDR, 1, 200) != HAL_OK) {
        return HAL_ERROR;
    }

    return HAL_OK;
}
```

Slika 19 Funckije u ST25DVO4.c

```c
/**
 * @brief   Upisuje više bajtova u ST25DV memoriju
 * @param   addr - Početna adresa upisa
 * @param   data - Pokazivač na podatke za upis
 * @param   len - Dužina podataka u bajtovima
 * @param   timeout - Maksimalno vrijeme čekanja u ms
 * @retval  HAL_OK ako je upis uspio, inače HAL_ERROR ili HAL_TIMEOUT
 */
HAL_StatusTypeDef ST25DV_WriteBytes(uint16_t addr, uint8_t *data, uint16_t len,
        uint32_t timeout) {
    if (hi2c_st25dv == NULL || data == NULL || len == 0) {
        return HAL_ERROR;
    }

    // Provjera adresnog područja
    if (addr + len - 1 > ST25DV_MAX_ADDR) {
        return HAL_ERROR;
    }

    uint16_t offset = 0;
    while (len > 0) {
        uint16_t chunk_len = (len > CHUNK_MAX) ? CHUNK_MAX : len;

        HAL_StatusTypeDef status = st25dv_write_chunk(addr + offset,
                data + offset, chunk_len, timeout);
        if (status != HAL_OK) {
            return status;
        }

        // Čekanje da EEPROM završi upis
        if (ST25DV_WaitReady(3000) != HAL_OK) {
            return HAL_TIMEOUT;
        }

        offset += chunk_len;
        len -= chunk_len;
    }

    return HAL_OK;
}
```

Slika 20 Funckije u ST25DVO4.c

```c
/**
 * @brief   Čita više bajtova iz ST25DV memorije
 * @param   addr - Početna adresa čitanja
 * @param   rx - Pokazivač na spremnik za primljene podatke
 * @param   len - Broj bajtova za čitanje
 * @param   timeout - Maksimalno vrijeme čekanja u ms
 * @retval  HAL_OK ako je čitanje uspjelo, inače HAL_ERROR
 */
HAL_StatusTypeDef ST25DV_ReadBytes(uint16_t addr, uint8_t *rx, uint16_t len,
        uint32_t timeout) {
    if (hi2c_st25dv == NULL || rx == NULL || len == 0) {
        return HAL_ERROR;
    }

    uint8_t addrbuf[2];
    addrbuf[0] = (uint8_t) (addr >> 8);     /* Visoki bajt adrese */
    addrbuf[1] = (uint8_t) (addr & 0xFF);   /* Niski bajt adrese */

    // Slanje adrese za čitanje
    if (HAL_I2C_Master_Transmit(hi2c_st25dv, ST25DV_DEV_ADDR, addrbuf, 2,
            timeout) != HAL_OK) {
        return HAL_ERROR;
    }

    // Primanje podataka
    return HAL_I2C_Master_Receive(hi2c_st25dv, ST25DV_DEV_ADDR, rx, len,
            timeout);
}

/**
 * @brief   Čeka da ST25DV bude spreman za novu operaciju
 * @param   timeout_ms - Maksimalno vrijeme čekanja u ms
 * @retval  HAL_OK ako je uređaj spreman, HAL_TIMEOUT ako nije
 */
HAL_StatusTypeDef ST25DV_WaitReady(uint32_t timeout_ms) {
    if (hi2c_st25dv == NULL) {
        return HAL_ERROR;
    }

    uint32_t start = HAL_GetTick();
    while ((HAL_GetTick() - start) < timeout_ms) {
        // I2C adresiranje uspijeva samo kad je EEPROM spreman
        if (HAL_I2C_IsDeviceReady(hi2c_st25dv, ST25DV_DEV_ADDR, 1, 10)
                == HAL_OK) {
            return HAL_OK;
        }
        HAL_Delay(5);
    }
    return HAL_TIMEOUT;
}
```

Slika 21 Funckije u ST25DVO4.c

```c
/**
 * @brief   Upisuje NDEF tekstualni zapis u ST25DV memoriju
 * @param   text - Tekst za pohranu
 * @retval  HAL_OK ako je upis uspio, inače HAL_ERROR
 */
HAL_StatusTypeDef ST25DV_WriteNDEFText(const char *text) {
    if (text == NULL || hi2c_st25dv == NULL) {
        return HAL_ERROR;
    }

    size_t text_len = strlen(text);
    if (text_len == 0) {
        return HAL_ERROR;
    }

    const char lang[] = "hr";              /* Jezična oznaka */
    uint8_t lang_len = (uint8_t) strlen(lang);
    uint32_t payload_len = 1 + lang_len + (uint32_t) text_len;
    uint8_t SR = (payload_len <= 255) ? 1 : 0;  /* Short Record zastavica */

    // Izračun veličina NDEF zapisa
    uint32_t ndef_record_overhead = 1 + 1 + (SR ? 1 : 4) + 1;
    uint32_t ndef_len = ndef_record_overhead + payload_len;

    // Provjera prostora
    if (NDEF_START_ADDR + ndef_len + 1 > ST25DV_MAX_ADDR) {
        return HAL_ERROR;
    }

    uint8_t header[16];
    uint32_t h = 0;

    // TLV format: Tip
    header[h++] = 0x03;  /* NDEF poruka TLV tip */

    // TLV format: Duljina
    if (ndef_len < 0xFF) {
        header[h++] = (uint8_t) ndef_len;
    } else {
        header[h++] = 0xFF;
        header[h++] = (uint8_t) ((ndef_len >> 8) & 0xFF);
        header[h++] = (uint8_t) (ndef_len & 0xFF);
    }

    // NDEF zaglavlje zapisa
    uint8_t rec_header = 0;
    rec_header |= (1 << 7);  /* MB - početak poruke */
    rec_header |= (1 << 6);  /* ME - kraj poruke */
    if (SR)
        rec_header |= (1 << 4);  /* SR - kratka duljina */
    rec_header |= 0x01;  /* TNF = 0x01 (NFC forum tip) */
    header[h++] = rec_header;

    header[h++] = 0x01;  /* Duljina tipa (T) */
```

Slika 22 Funckija u ST25DVO4.c

```
    // Duljina payloada
    if (SR) {
        header[h++] = (uint8_t) payload_len;
    } else {
        header[h++] = (uint8_t) ((payload_len >> 24) & 0xFF);
        header[h++] = (uint8_t) ((payload_len >> 16) & 0xFF);
        header[h++] = (uint8_t) ((payload_len >> 8) & 0xFF);
        header[h++] = (uint8_t) (payload_len & 0xFF);
    }

    header[h++] = 'T';  /* Tip zapisa: tekst */
    header[h++] = (uint8_t) (lang_len & 0x3F);  /* Duljina jezika */

    // Jezična oznaka
    for (uint8_t i = 0; i < lang_len; ++i) {
        header[h++] = (uint8_t) lang[i];
    }

    uint16_t addr = NDEF_START_ADDR;
    HAL_StatusTypeDef st;

    // Upis zaglavlja
    st = ST25DV_WriteBytes(addr, header, h, 100);
    if (st != HAL_OK)
        return st;

    addr += h;

    // Upis teksta u dijelovima
    const uint8_t *tp = (const uint8_t*) text;
    size_t rem = text_len;
    uint8_t chunk_buf[CHUNK_MAX];

    while (rem) {
        uint16_t c = (rem > CHUNK_MAX) ? CHUNK_MAX : (uint16_t) rem;
        memcpy(chunk_buf, tp, c);

        st = ST25DV_WriteBytes(addr, chunk_buf, c, 100);
        if (st != HAL_OK)
            return st;

        addr += c;
        tp += c;
        rem -= c;
    }

    // TLV terminator (kraj)
    uint8_t term = 0xFE;
    st = ST25DV_WriteBytes(addr, &term, 1, 100);
    if (st != HAL_OK)
        return st;

    return HAL_OK;
}
```

Slika 23 Funckija u ST25DVO4.c

```c
/**
 * @brief   Pomoćna funkcija za slanje broja čaša vode na NFC tag
 * @param   casa_vode - Broj čaša za pohranu
 */
void ST25DV_SendCasaCount(uint16_t casa_vode) {
    char buffer[32];
    snprintf(buffer, sizeof(buffer), "%u čaša vode", casa_vode);
    ST25DV_WriteNDEFText(buffer);
}

/* Private functions ---------------------------------------------------------*/

/**
 * @brief   Upisuje jedan blok podataka u ST25DV memoriju
 * @param   addr - Adresa upisa
 * @param   data - Pokazivač na podatke
 * @param   len - Duljina podataka (maksimalno CHUNK_MAX)
 * @param   timeout - Maksimalno vrijeme čekanja u ms
 * @retval  HAL_OK ako je upis uspio, inače HAL_ERROR
 */
static HAL_StatusTypeDef st25dv_write_chunk(uint16_t addr, uint8_t *data,
        uint16_t len, uint32_t timeout) {
    if (len > CHUNK_MAX || len == 0) {
        return HAL_ERROR;
    }

    uint8_t buf[2 + CHUNK_MAX];
    buf[0] = (uint8_t) (addr >> 8);     /* Visoki bajt adrese */
    buf[1] = (uint8_t) (addr & 0xFF);   /* Niski bajt adrese */
    memcpy(&buf[2], data, len);         /* Kopiranje podataka iza adrese */

    return HAL_I2C_Master_Transmit(hi2c_st25dv, ST25DV_DEV_ADDR, buf, 2 + len,
            timeout);
}
```

Slika 24 Funckije u ST25DVO4.c

5.2. ST25DV.h

Datoteka st25dv.h je zaglavlje koje definira sučelje za upravljanje ST25DV NFC čipom. Sadrži sve potrebne definicije konstanti, adresa i prototipova funkcija koje se koriste u pripadajućoj st25dv.c implementaciji.

U zaglavlju su definirane ključne konstante: I2C adresa uređaja (ST25DV_7BIT_ADDR i ST25DV_DEV_ADDR), početna adresa NDEF podataka u EEPROM memoriji (NDEF_START_ADDR) te maksimalna veličina pojedinačnog I2C upisa (CHUNK_MAX). Također su navedene granice adresnog prostora za tri različita modela ST25DV čipova (04, 16 i 64), pri čemu je aktivna definicija ST25DV_MAX_ADDR postavljena na model ST25DV04KC (512 bajta) koji se koristi u ovom projektu, uz napomenu da se ta vrijednost može promijeniti ovisno o korištenom modelu.

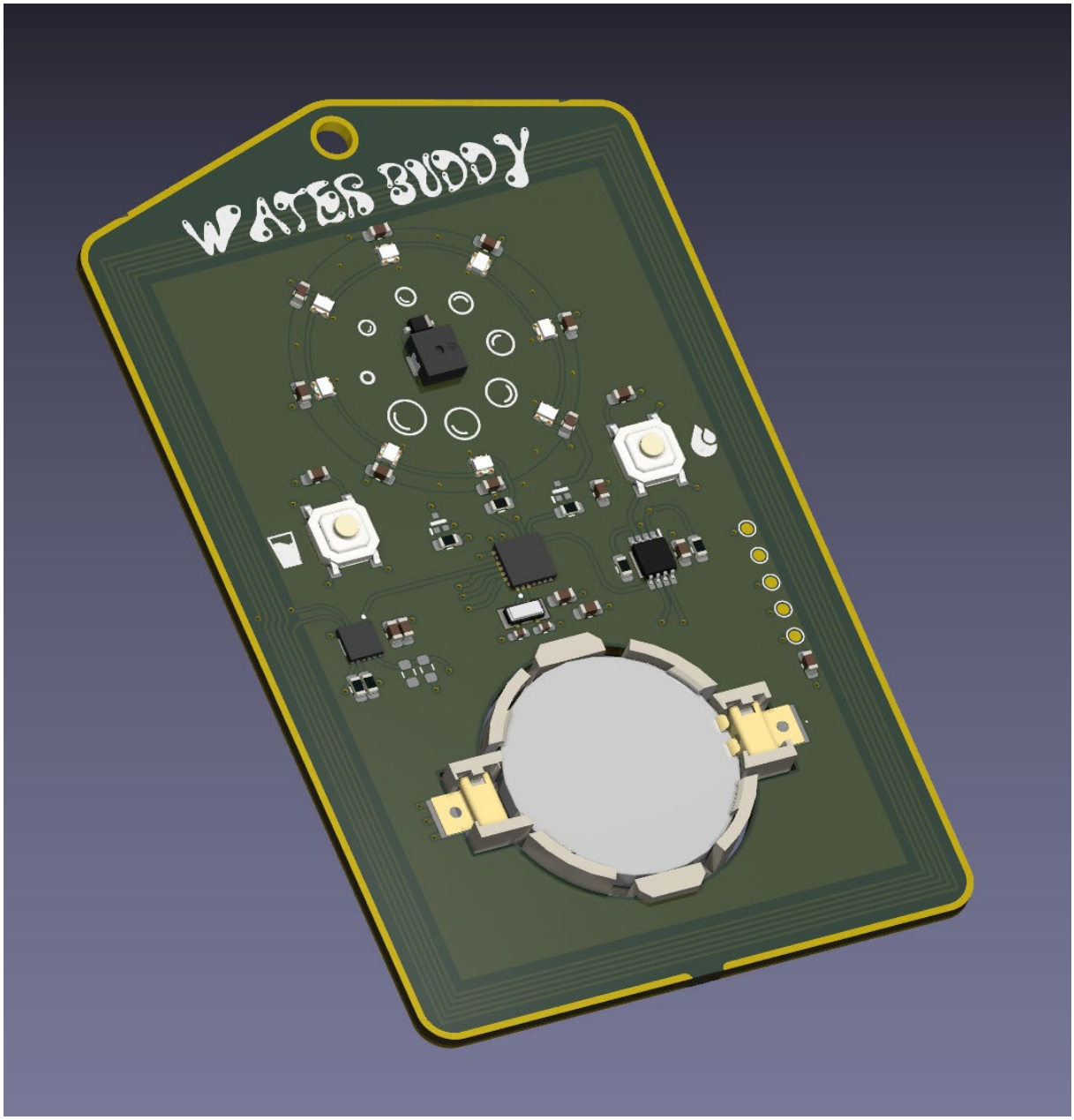
Prototipi funkcija pružaju potpuno sučelje za rad s čipom: inicijalizaciju, čitanje i upis memorije, provjeru spremnosti te funkcije za formatiranje podataka u NDEF oblik. Ovo zaglavlje omogućuje drugim dijelovima programa jednostavno korištenje ST25DV funkcionalnosti bez potrebe za poznavanjem detalja I2C komunikacije ili NDEF formatiranja.

```
/**
  ******************************************************************************
  * @file    st25dv.h
  * @brief   Header file for ST25DV EEPROM driver
  ******************************************************************************
  * @attention
  *
  * Created on: Dec 7, 2025
  *     Author: Admin
  *
  ******************************************************************************
  */

#ifndef ST25DV_H
#define ST25DV_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "main.h"        /* Glavna HAL biblioteka */
#include <string.h>      /* Funkcije za rad sa stringovima */
#include <stdio.h>       /* Standardne ulazno-izlazne funkcije */

/* Defines -------------------------------------------------------------------*/
#define ST25DV_7BIT_ADDR    0x53        /* 7-bitna I2C adresa ST25DV čipa */
#define ST25DV_DEV_ADDR     (ST25DV_7BIT_ADDR << 1)  /* 8-bitna I2C adresa za HAL */
#define NDEF_START_ADDR     0x0004      /* Početna adresa NDEF podataka u EEPROM-u */
#define CHUNK_MAX           32          /* Maksimalan veličina paketa I2C write */

/* Granične adresne vrednosti za različite modele ST25DV */
#define ST25DV04_MAX_ADDR   0x01FF      /* Zadnja adresa za ST25DV04KC (4 Kbit = 512 B) */
#define ST25DV16_MAX_ADDR   0x07FF      /* Zadnja adresa za ST25DV16KC (16 Kbit = 2 KB) */
#define ST25DV64_MAX_ADDR   0x1FFF      /* Zadnja adresa za ST25DV64KC (64 Kbit = 8 KB) */

/* Odabir maksimalne adrese prema o korišćenom modelu */
#define ST25DV_MAX_ADDR     ST25DV04_MAX_ADDR  /* <-- PROMJENI OVDJE za drugi model */

/* Function prototypes -------------------------------------------------------*/
/**
  * @brief   Inicijalizacija ST25DV upravljača
  * @param   hi2c - Pokazivač na I2C strukturu
  * @retval  HAL_OK ako je inicijalizacija uspjela
  */
HAL_StatusTypeDef ST25DV_Init(I2C_HandleTypeDef *hi2c);

/**
  * @brief   Upis podataka u ST25DV memoriju
  * @param   addr - Početna adresa upisa
  * @param   data - Pokazivač na podatke
  * @param   len - Broj bajtova za upis
  * @param   timeout - Maksimalno vrijeme čekanja u ms
  */
HAL_StatusTypeDef ST25DV_WriteBytes(uint16_t addr, uint8_t *data, uint16_t len, uint32_t timeout);

/**
  * @brief   Čitanje podataka iz ST25DV memorije
  * @param   addr - Početna adresa čitanja
  * @param   rx - Pokazivač na međumemoriju za primljene podatke
  * @param   len - Broj bajtova za čitanje
  * @param   timeout - Maksimalno vrijeme čekanja u ms
  */
HAL_StatusTypeDef ST25DV_ReadBytes(uint16_t addr, uint8_t *rx, uint16_t len, uint32_t timeout);

/**
  * @brief   Čekanje da ST25DV bude spreman
  * @param   timeout_ms - Maksimalno vrijeme čekanja u ms
  */
HAL_StatusTypeDef ST25DV_WaitReady(uint32_t timeout_ms);

/**
  * @brief   Upis NDEF tekstualnog zapisa
  * @param   text - Tekst za pohranu na NFC tag
  */
HAL_StatusTypeDef ST25DV_WriteNDEFText(const char *text);

/**
  * @brief   Pomoćna funkcija za slanje broja ispita preko UART
  * @param   case_code - Broj ispita za pohranu
  */
void ST25DV_SendCaseCount(uint16_t case_code);

#ifdef __cplusplus
}
#endif

#endif /* ST25DV_H */
```

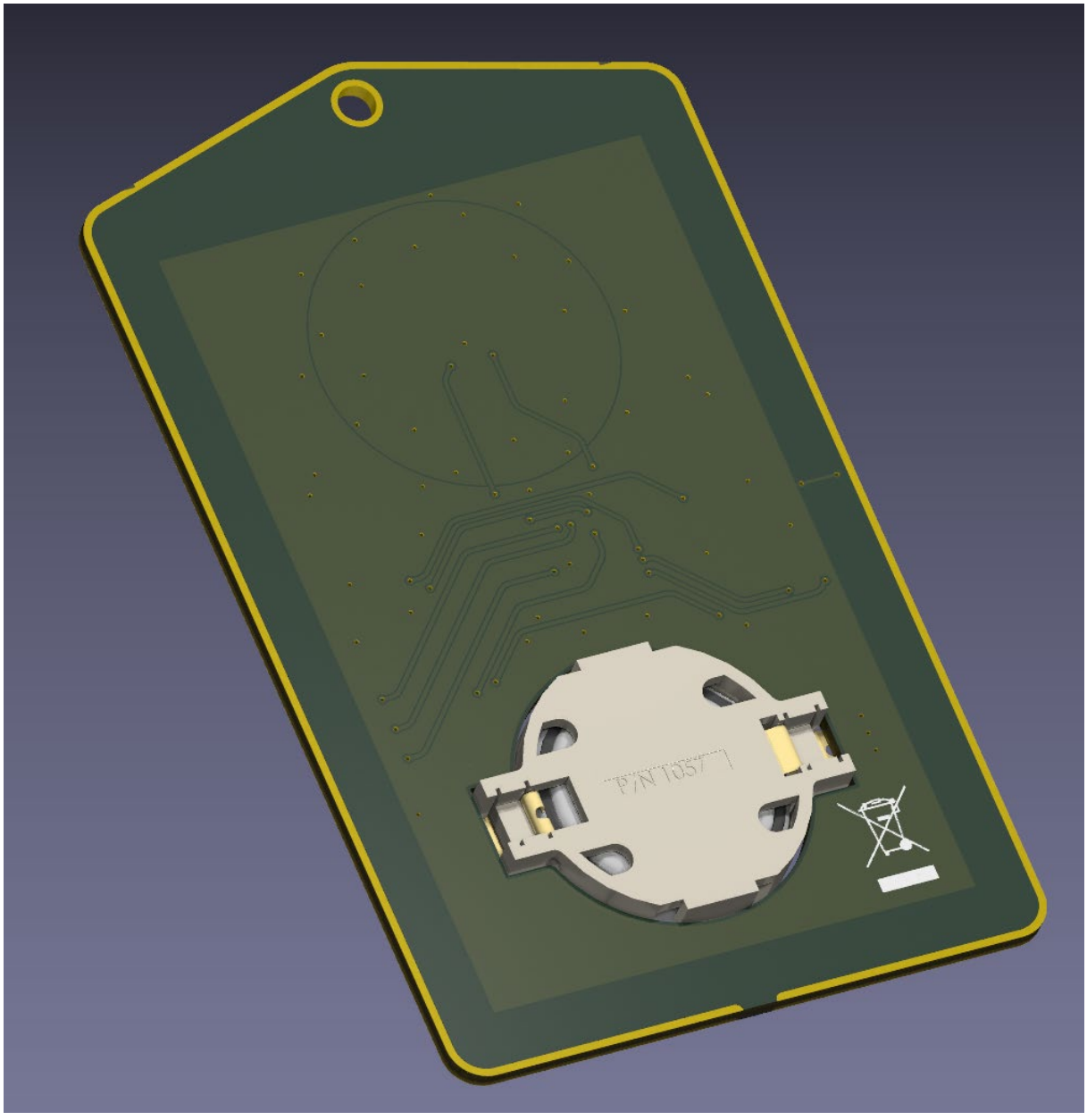
Slika 25 Deklaracije funckija u ST25DVO4.h

6. Prikaz uređaja i stavke

6.2. 3D prikaz uređaja



Slika 26 Prednja strana 3D prikaza uređaja



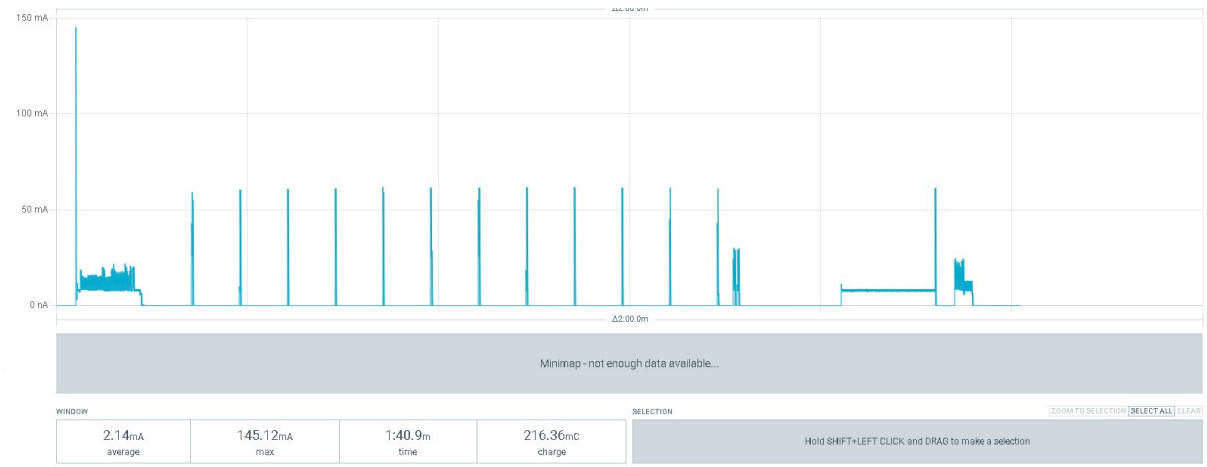
Slika 27 Stražnja strana 3D prikaza uređaja

6.2. Potrošnja i vijek rada

Kako je program (kod) osmišljen da je u radnom stanju samo kratko, kada zvoni alarm, a ostalo vrijeme u stanju male potrošnje (STOPMode), uz baterijski izvor napajanja ovaj uređaj ovisi o maloj potrošnji kako bi ispunio što duži vijek trajanja. Kada je u normalnom načinu rada sa upaljenim indikatorima (zujalica i LED lampice), sustav koristi ≈24mA ali samo po 100mS svake 4 sekunde(interval aktiviranog alarma).

Prethodno navedeni način rada niske potrošnje koristi pak **samo ≈4 µA** struje što je ključno za dug vijek rada. S ovakvim opisanim načinom rada, **očekivani rad uređaja je do 23 mjeseci na jednoj CR2032 bateriji**

U privitku se nalazi graf generiran uz pomoć PowerProfiler alata od Nordic Semiconductor proizvođača. Na grafu se nalazi potrošnja jednog kompletnog paljenja, što uključuje postavljanje postavki pomoću tipkala, aktivacija alarma od trajanja jedne minute, pauziranje alarma na deset sekundi te naposlijetku inkrementacija/gašenje alarma.



Slika 28 Potrošnja na grafu

7. Zaključak

Uz brz tempo današnjeg svijeta oko nas, čovjek često uđe u zaborav samoga sebe dok se fokusira na druge stvari oko njega. Stoga ovaj uređaj, često sam se doveo u situaciju gdje bih taj dan nedovoljno vode popi zbog drugih i raznih briga koje su mi bile na pameti.

Svjestan sam da postoje aplikacije koje možda čine slično i rješavaju isti problem, no ideja ovoga rada je bilo naučiti nešto novo, integrirati komunikacijski protokol te pri čemu i stvoriti nešto zanimljivo i zabavno za koristi, nešto neovisno o mobitelima, na kojima ionako provodimo previše našeg dragocjenog vremena.

8. GitHub

https://github.com/kmikulaco/KTM_Seminar.git

9. Tablica slika

10. Literatura

[1] https://www.st.com/en/microcontrollers-microprocessors/stm32g031g8.html#documentation
[2]https://wiki.st.com/stm32mcu/wiki/Getting_started_with_PWR#:~:text=Stop%20mode%20achieves%20the%20lowest,LSI%20can%20be%20kept%20running.

[3] https://www.arrow.com/en/research-and-events/articles/protocol-for-the-ws2812b-programmable-led

Napomena:
Za izradu seminara su također korišteni dostupni materijali sa stranice kolegija na LMS-u.