Pametni gumb - Napredni haptički kontroler

Radić, Miloš

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE RAČUNARSTVA I INFORMACIJSKIH TEHENOLOGIJA

Stručni studij

PAMETNI GUMB – NAPREDNI HAPTIČKI KONTROLER

Završni rad

Miloš Radić

Osijek, 2024.

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

Obrazac Z1S: Obrazac za ocjenu završnog rada na stručnom prijediplomskom studiju

Ocjena završnog rada na stručnom prijediplomskom studiju

Ime i prezime pristupnika:	Miloš Radić
Studij, smjer:	Stručni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	4450, 07.10.2021.
JMBAG:	0165075876
Mentor:	izv. prof. dr. sc. Tomislav Keser
Sumentor:	prof. dr. sc. Damir Blažević
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	doc. dr. sc. Tomislav Galba
Član Povjerenstva 1:	izv. prof. dr. sc. Tomislav Keser
Član Povjerenstva 2:	izv. prof. dr. sc. Alfonzo Baumgartner
Naslov završnog rada:	%naziv_rada%
Znanstvena grana završnog rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Projektirati, izraditi i testirati sustav haptičkog upravljačkog sučelja temeljen na principu IoT i rotirajućeg pametnog gumba a kojega će karakterizirati: ESP32 upravljačka platforma, IoT principi povezivosti i upravljanja sustavima, rotirajući gumb za odabir i zadavanje upravljačke vrijednosti, haptička povratna veza za dodirnu povratnu vezu i upravljanje rotacijskim svojstvima gumba, okrugao pokaznik integriran u tijelo gumba za prikaz relevantnih veličina, "klik" svojstvo rotirajućeg gumba, dinamička višeboina rasveta za signalizaciju stanja gumba
Datum ocjene pismenog dijela završnog rada od strane mentora:	08.09.2024.
Ocjena pismenog dijela završnog rada od strane mentora:	Izvrstan (5)
Datum obrane završnog rada:	12.09.2024.
Ocjena usmenog dijela završnog rada (obrane):	Izvrstan (5)
Ukupna ocjena završnog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio stručni prijediplomski studij:	12.09.2024.

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

IZJAVA O IZVORNOSTI RADA

Osijek, 12.09.2024.

Ime i prezime Pristupnika:	Miloš Radić
Studij:	Stručni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	4450, 07.10.2021.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Pametni gumb - Napredni haptički kontroler**

izrađen pod vodstvom mentora izv. prof. dr. sc. Tomislav Keser

i sumentora prof. dr. sc. Damir Blažević

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 pristupnika:

SADRŽAJ

1. UVOD

Živimo u dobu rapidnog razvoja i inovacija mnogobrojnih rješenja za postojeće a ponekad i za nepostojeće probleme. Problem većine uređaja je što rješavaju samo jedan problem. Podizanje i spuštanje zastora, kontrola intenziteta rasvjete, kontrola hlađenja i grijanja prostorije, sve zahtjeva po jedan daljinski upravljač nekada i dva. Olakšavanjem tako jednostavnih zadataka stvaramo si novi problem a to je količina daljinskih upravljača za svaki pojedini sustav.

Pametni gumb rješava više problema od jednom. Sustav je prilagodljiv korisnikovim zahtjevima, u potpunosti je otvoren za daljnji razvoj. Kontrola rasvjete, zastora, rashladnih i toplinskih uređaja i svih ostalih sustava za pametnu kuću moguća je pomoću pametnog gumba.

Tržište automatizacije pametnih kuća u 2023. godini, prema [1], procijenjeno je na 50 milijardi američkih dolara, dok je procjena tržišta za 2033. godinu 135 milijardi američkih dolara, što je godišnji porast od skoro 11%. Iz ovih brojki možemo vidjeti da je tržište sustava za pametne kuće još uvijek u početku svog razvoja i da val inovacija u ovom sektoru tek dolazi.

Najveći postotak sustava za automatizaciju pametnih kuća je u sektoru tehničke zaštite, alarmni sustavi za kuće, iza kojih slijedi rasvjeta, zatim audio i video oprema te na kraju sustavi rashlađivanja i grijanja. Na tržištu postoje već gotova rješenja ovakvih problema, međutim kao i svako tehničko rješenje, imaju svoje prednosti i mane. Najveći tržišni udio sustava automatizacije drži tvrtka AMX, međutim njen fokus je na poslovne klijente i luksuzne privatne vile. Nama najpoznatiji sustavi automatizacije su *Google Assistant*, *Amazon Alexa* i *Apple HomeKit* pametni zvučnici. Svaki od uređaja nudi neprimjetnu integraciju u svoj ekosustav, mnogobrojne prednosti, visoku razinu razvijenosti, i mogućnost integracije s ostalim sustavima, međutim želimo li još jedan uređaj koji nas može pratiti ? KNX sustav i protokol za automatizaciju iz godine u godinu širi svoj udio u tržištu. KNX sustav je otvorenog tipa, svatko ga može neograničeno koristiti i unaprijediti međutim većinom je fokusiran na nove instalacije. Adaptacija već postojećih instalacija uz pomoć KNX sustava je poprilično zahtjevna. *Home assistant (HA)* sustav dobiva svoj zamah kao *DIY (Do it yourself)* tj. uradi sam metoda. Cijeli sustav je također otvorenog tipa prvobitno usmjerenog na sigurnost i privatnost korisnika. Sustav je stvoren od strane korisnika koji se njime koriste te s time možemo znati da je većina funkcionalnosti sustava stvarno korisna. Glavna funkcionalnost pametnog gumba je integracija u HA sustav, koja nudi korisniku gotovo bezbroj mogućnosti za prilagodbu svojim potrebama.

Otvaranjem sustava i integracijom u već otvoreni sustav rješavamo probleme spomenute ranije gdje za svaki sustav od svakog proizvođača imamo po jedan ili nekad i više načina kontrole. Rješava se problem korištenja različitih načina upravljanja za svaki sustav koji koristimo.

1.1. Zadatak i struktura rada

Pametni gumb – napredni haptički kontroler, integrira različite sustave kontrole kućnih uređaja, rasvjete, grijanja i hlađenja itd. u jedno sveobuhvatno i jednostavno rješenje. Uz integraciju u Home Assistant sustav ideja je postići jednostavan i lako proširiv sustav automatizacije pametnih kuća. Ova dokumentacija obuhvaća cijeli proces izrade uređaja od početne faze planiranja do završne faze postavljanja gotovog proizvoda i korištenja.

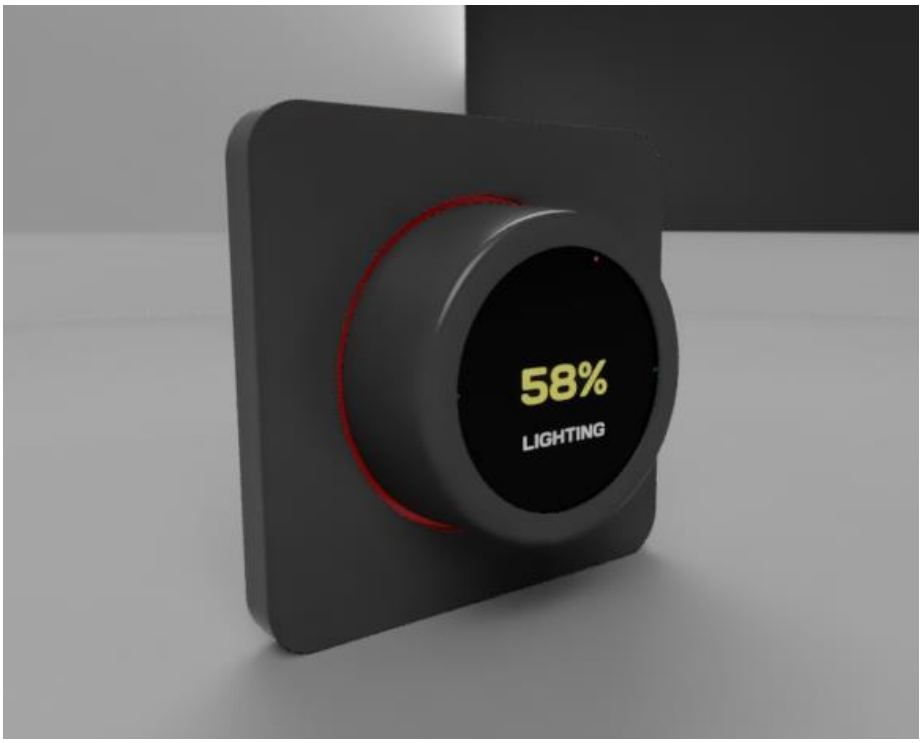
U drugom poglavlju ovog rada razrađena je teorijska problematika koja prethodi samoj izradi projekta. U ovom dijelu se predlaže teorijski okvir za realizaciju sklopovskog i programskog rješenja. Detaljno se objašnjava pristup koji će biti korišten za implementaciju tehničkih aspekata sustava.

Treće poglavlje posvećeno je detaljnom opisu procesa izrade pametnog gumba. Pruženi su uvidi u alate korištene za dizajn sklopovlja, izradu 3D modela proizvoda, kao i alate korištene za razvoj programskih rješenja. Pojašnjen je način dizajna i izrade gumba, arhitekture sklopovskog rješenja, kao i dizajn programskog rješenja, uključujući integraciju s vanjskim sustavima.

U četvrtom poglavlju objašnjen je proces testiranja sustava te analizirani rezultati. Testiranje je provedeno na odabranoj grupi korisnika, a fokus je bio na mjerenju njihovog zadovoljstva korištenjem proizvoda. Rezultati korisničkog zadovoljstva prikazani su u tabelarnom obliku kroz tri različita scenarija korištenja.

2. PAMETNI GUMB – SVEOBUHVATNI HMI SUSTAV.

Pametni gumb je haptički sustav koji ima mogućnost objedini većinu potreba za daljinskim upravljanjem sustava pametnih kuća. Na slici 2.1. možemo vidjeti *render* 3D modela iz softvera za modeliranje. Pametni gumb se sastoji od okruglog LC pokazivača koji pruža napredne sposobnosti grafičkog sučelja sa sustavom. Sadrži *brushless* motor (motor bez četkica) i magnetski enkoder pozicije koji mu omogućuju haptičku povratnu vezu kao dodatan oblik interakcije sa korisnikom. LE diode postavljene na rubu prstena također pružaju interaktivne sposobnosti.



Slika 2.1. Idejni prikaz pametnog gumba

2.1. Teorija funkcionalnosti pametnog gumba

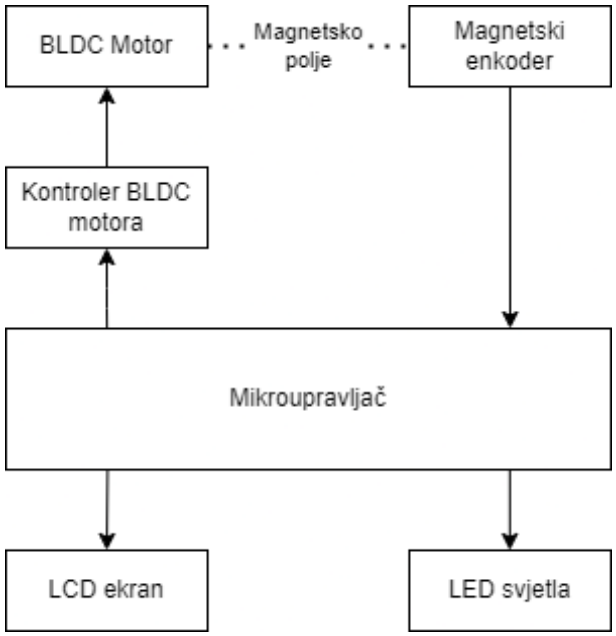
Osnovna značajka pametnog gumba je FOC sustav (*field oriented control*). FOC sustav ili vektorski sustav, koristi imaginarni vektor položaja kao informaciju o trenutnoj poziciji motora. Koristeći vektorsku kontrolu i magnetski enkoder, pozicija motora se može izuzetno precizno kontrolirati te time u teoriji dobivamo neku vrstu servo motora. Kombinirajući spomenuta dva sustava dobivamo

softverski kontrolirani rotacijski enkoder sa mogućnošću kontroliranja njegovog pokreta i povratne veze prema korisniku. Ovisno o softverskoj konfiguraciji kotač gumba se može namjestiti da bude prekidač sa automatskih povratom na sredinu, rotacijski enkoder sa „osjetnim klikovima“ prilikom vrtnje, prekidač sa povratom na sredinu prilikom završetka radnje, i sve ostale haptičke sposobnosti ovakvog sustava.

LCD i LE diode pružaju grafičku vezu sa korisnikom. LCD se koristi kao povratna veza sa korisnikom u kombinaciji sa FOC sustavom. Stavke prikazane i meniji prikazani na ekranu se mogu odabirati, mijenjati i odabirati koristeći FOC sustav. Uz LCD koriste se i LE diode kao dodatna komponenta interakcije sa sustavom.

2.2. Sklopovsko rješenje sustava

Slika 2.2 prikazuje sklopovlje sustava koji se može podijeliti u dvije grupe: kontrola motora i kontrola vizualnih sustava (LCD, LED). Kontrola motora se odvija pomoću *brushless motor driver* čipa (čip za kontrolu motora bez četkica). Kao povratna veza o njegovoj poziciji se koristi magnetski enkoder koji mikroupravljaču šalje potrebne informacije.



Slika 2.2. Skica sklopovlja pametnog gumba

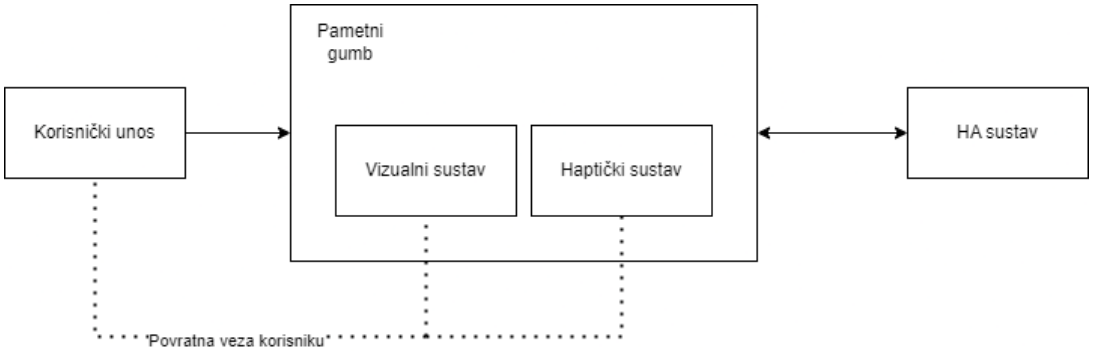
Mikroupravljač dobiva informaciju o poziciji motora pomoću magnetskog enkodera te u skladu s tim informacijama kontrolira poziciju motora koristeći kontroler BLDC motora. BLDC motor i magnetski enkoder su međusobno povezani magnetskim poljem, tj. rotacija magneta određuje poziciju motora. S obzirom da za vizualne sustave nije potrebna nikakva povratna informacija, njihovo sklopovsko rješenje je poprilično jednostavno. Mikroupravljač direktno komunicira sa pojedinom komponentom sustava te ju direktno kontrolira.

2.3. Programsko rješenje sustava

Firmver pametnog gumba bit će zasnovan na Arduino bibliotekama, što pruža praktičan sloj apstrakcije za ubrzani razvoj. Zahvaljujući Arduino bibliotekama, programer se može usredotočiti na ključnu logiku sustava bez gubljenja vremena na rješavanje tehničkih izazova vezanih uz aspekte niske razine, kao upravljanje hardverom. S obzirom na to da će sustav zahtijevati obradu podataka u realnom vremenu, koristi se *FreeRTOS* (*real-time operating system*). Ovaj sustav omogućit će preciznu i pravovremenu obradu ulaznih podataka, osiguravajući da pametni gumb reagira učinkovito i pouzdano u svakom trenutku. Korištenje *FreeRTOS-a* jamči stabilnost i brzinu u kompleksnim scenarijima, gdje su točnost i brzina obrade od ključne važnosti. Slika 2.3. prikazuje slojeve firmvera gdje možemo jasno vidjeti kako *Arduino* biblioteke stvaraju sloj apstrakcije za sve ostale više slojeve sustava. Komunikacija između pametnog gumba i vanjskih sustava može se odvijati pomoću bežične internetske mreže (WiFi) ili *Bluetooth-a* koristeći HTTP komunikacijski protokol.

Slika 2.3. Slojevi firmvera sustava

Na slici 2.4. prikazan je pojednostavljeni prikaz veze između HA sustava i korisničkog unosa. Pametni gumb djeluje kao "most" između korisnika i automatizacijskog sustava, omogućujući intuitivnu interakciju. Svaki unos koji korisnik izvrši putem gumba prolazi kroz obradu u samom uređaju, nakon čega se relevantna informacija prenosi na HA sustav. Haptički i vizualni sustavi osiguravaju trenutnu povratnu informaciju korisniku o statusu njegovog unosa, što povećava osjećaj interaktivnosti i kontrole. Nakon što HA sustav obradi podatke, povratna informacija o statusu unosa također se šalje natrag korisniku putem istih mehanizama, osiguravajući da je korisnik u svakom trenutku svjestan rezultata svojih akcija. Ovaj dvosmjerni proces omogućava glatku i učinkovitu komunikaciju između korisnika i automatizacijskog sustava, čineći iskustvo korištenja jednostavnim i ugodnim.



Slika 2.4. Veza korisničkog unosa i HA sustava

3. REALIZACIJA SUSTAVA PAMETNOG GUMBA

3.1. Korišteni alati i programska okruženja

Korišteni alati i programska okruženja mogu se podijeliti u tri cjeline:

- Alati za dizajn sklopovlja
- Alati za mehanički dizajn
- Alati za razvoj softvera

3.1.1. Alati za dizajn sklopovlja

Prilikom dizajna sklopovlja, sheme i tiskane pločice korišten je softverski paket Altium Designer 21 uz sve popratne internetske portale za nabavku i pretragu elektroničkih komponenti: Octopart, DigiKey, Mouser, itd … Softverski paket Altium Designer je industrijski standard za kreiranje shematskih prikaza, dizajniranje i crtanje tiskanih pločica te također za kreiranje takozvanih *BOM-ova* (*bill of materials*) tj. popisa svih komponenti potrebnih za izradu tiskane pločice.

3.1.2. Alati za mehanički dizajn

Uz 3D sposobnosti Altium Designer sustava, za mehanički dizajn korišten je softverski paket Autodesk Fusion 360. Softverski paket se koristi u industrijske ali i u hobističke svrhe zbog svoje jednostavnosti.

3.1.3. Alati za razvoj softvera

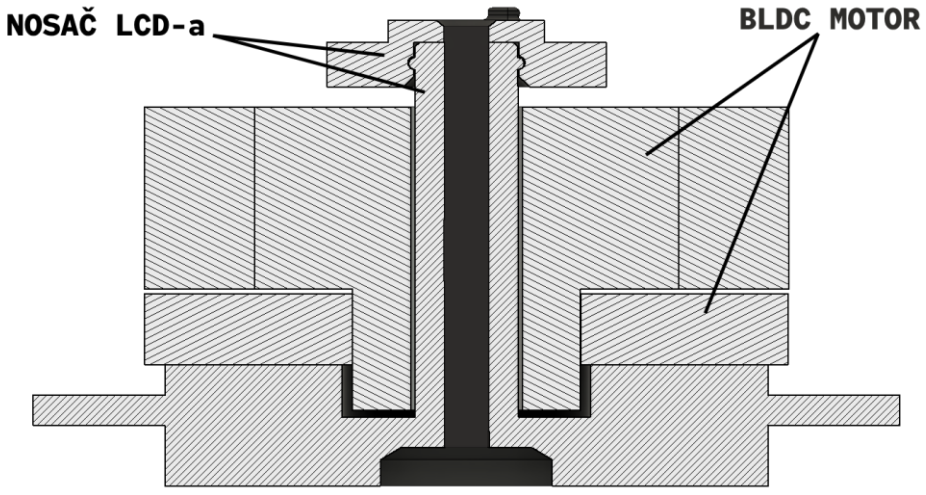
Najviše korišteni alati prilikom izrade pametnog gumba su softverski alati. Program korišten za uređivanje koda je Visual Studio Code uz softverski sustav Arduino kao popratnu bazu firmvera.

3.2. Realizacija sklopovskog rješenja

Realizaciju sustava pametnog gumba možemo podijeliti u dvije glavne kategorije: realizacija mehaničkog sustava gumba i realizacija električkog sklopovlja. Realizacija mehaničkog sustava gumba uključuje dizajniranje i izradu mehaničkih dijelova koji omogućuju funkcionalnost sustava. U okviru električkog sklopovlja obuhvaćeni su dizajn, izrada i sastavljanje printane elektroničke pločice (PCB) te ostalih popratnih elektroničkih komponenti sustava.

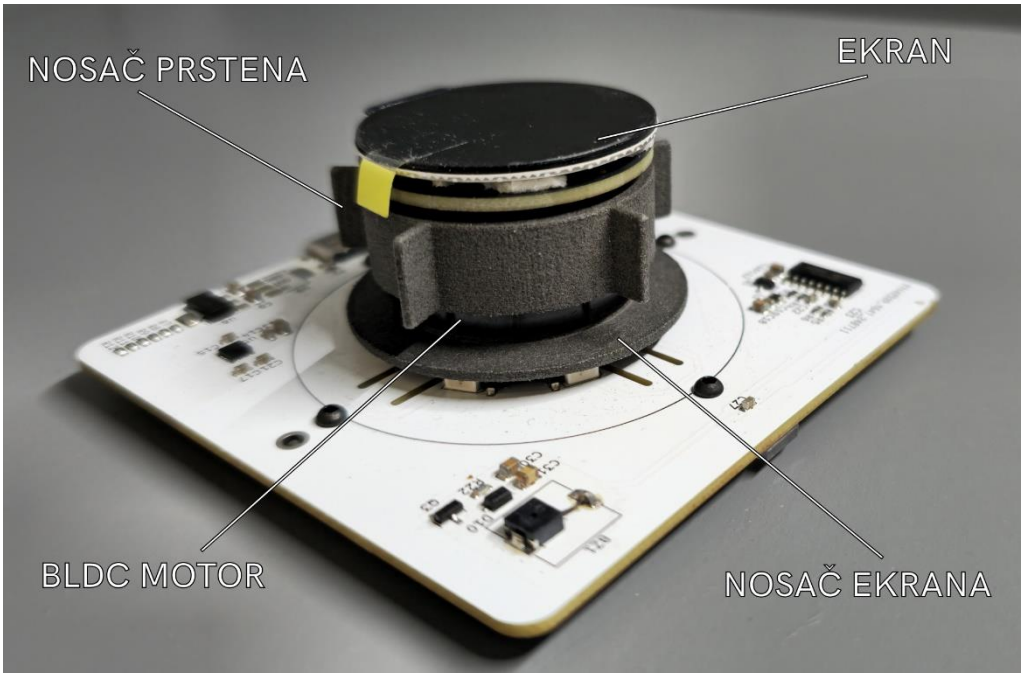
3.2.1. Realizacija mehaničkog sustava gumba

Mehanizam pametnog gumba razvijen je koristeći softverski paket Autodesk Fusion 360, s dizajnom prilagođenim BLDC motoru sa osovinom koja ima unutrašnji kanal. Ovaj motor omogućava da LCD komponenta ostane fiksirana u središtu gumba, dok se vanjsko kućište može slobodno rotirati, što je ključno za ispravno funkcioniranje cijelog sustava. Nosač LCD-a koncipiran je tako da se precizno integrira unutar osovine s unutrašnjim kanalom BLDC motora, čime omogućava stabilizaciju motora i pruža strukturnu potporu samom LCD-u, što jasno možemo vidjeti na slici 3.1. Pored toga, nosač služi kao strukturirani kanal za organizirano sprovođenje električnih vodiča od LCD-a do glavnog PCB-a sustava, čime se osigurava optimalna električna povezanost i funkcionalna integracija unutar sustava. Nosač LCD-a izrađen je pomoću 3D printera koristeći SLS (*selective laser sintering*) tehnologiju, čime se postiže visok stupanj završne kvalitete 3D printane komponente.



Slika 3.1. Presjek 3D modela. Nosač LCD-a

Na rotor BLDC motora pričvršćuje se nosač rotacijskog prstena gumba, na koji se zatim montira sam prsten. Ovim dizajnom ostvaruje se neposredna povezanost između rotacijskog obruča i BLDC motora, čime se omogućuje precizna kontrola i povratna informacija korisniku. Slika 3.2. jasno prikazuje dizajn nosača rotacijskog prstena, gdje možemo vidjeti mehanizam pričvršćivanja rotacijskog prstena. Takav dizajn osigurava izvrsnu usklađenost između rotacijskih kretanja gumba i motornih impulsa, što poboljšava ukupno korisničko iskustvo. Nosač prstena je također proizveden koristeći 3D printer i SLS tehnologiju, čime je osigurana visoka preciznost proizvedenih komponenti.



Slika 3.2. Prikaz nosača rotacijskog prstena

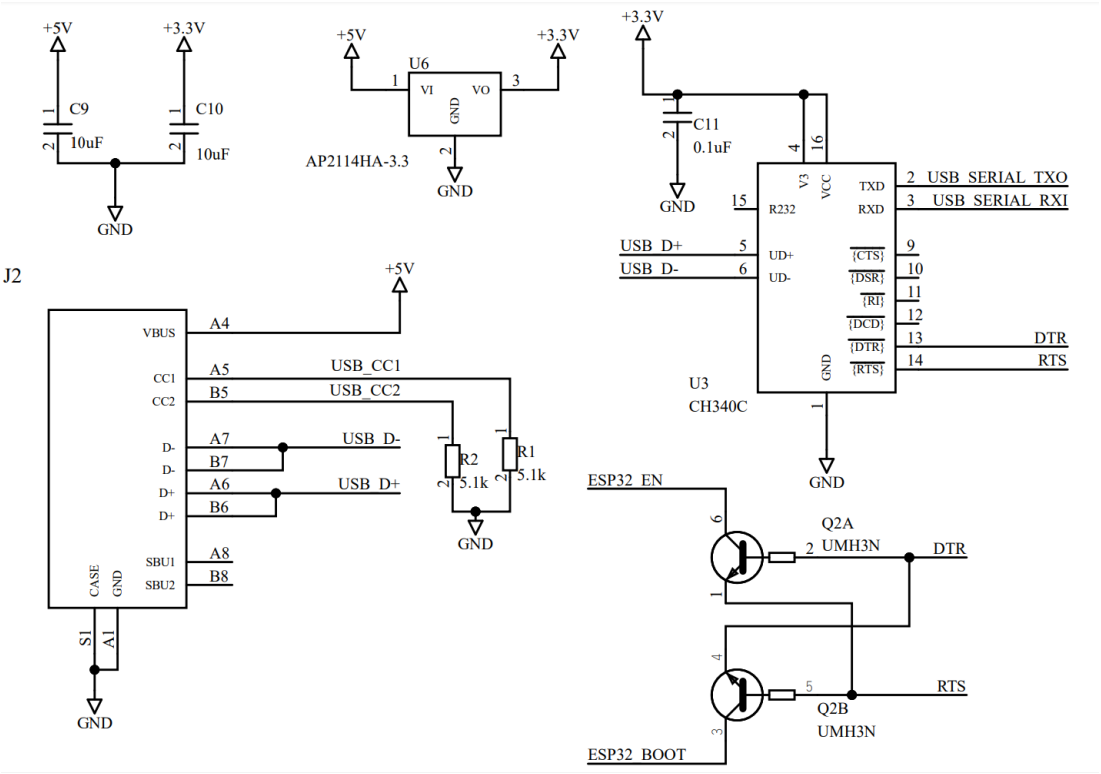
Sklop motora, zajedno s LCD-om i njegovim nosačem, pričvršćuje se na glavni PCB sustava, čime se formira kompletan sustav pametnog gumba. Na glavni PCB-u se zatim ugrađuje vanjsko kućište gumba, što dovodi do završne faze izrade proizvoda. Ovim postupkom pametni gumb postaje gotov proizvod, pri čemu su sve unutarnje komponente sustava skrivene i nevidljive, te se osigurava estetski izgled i funkcionalnost finalnog proizvoda. Rotacijski prsten i vanjsko kućište gumba izrađeni su od aluminija primjenom CNC glodanja i eloksiranja. Ovim postupkom postignuta je izvrsna estetska i taktilna kvaliteta kao i sofisticiran izgled proizvoda što možemo vidjeti na slici 3.3.

Slika 3.3. Završno sastavljen uređaj

3.2.2 Realizacija elektroničkog sklopovlja

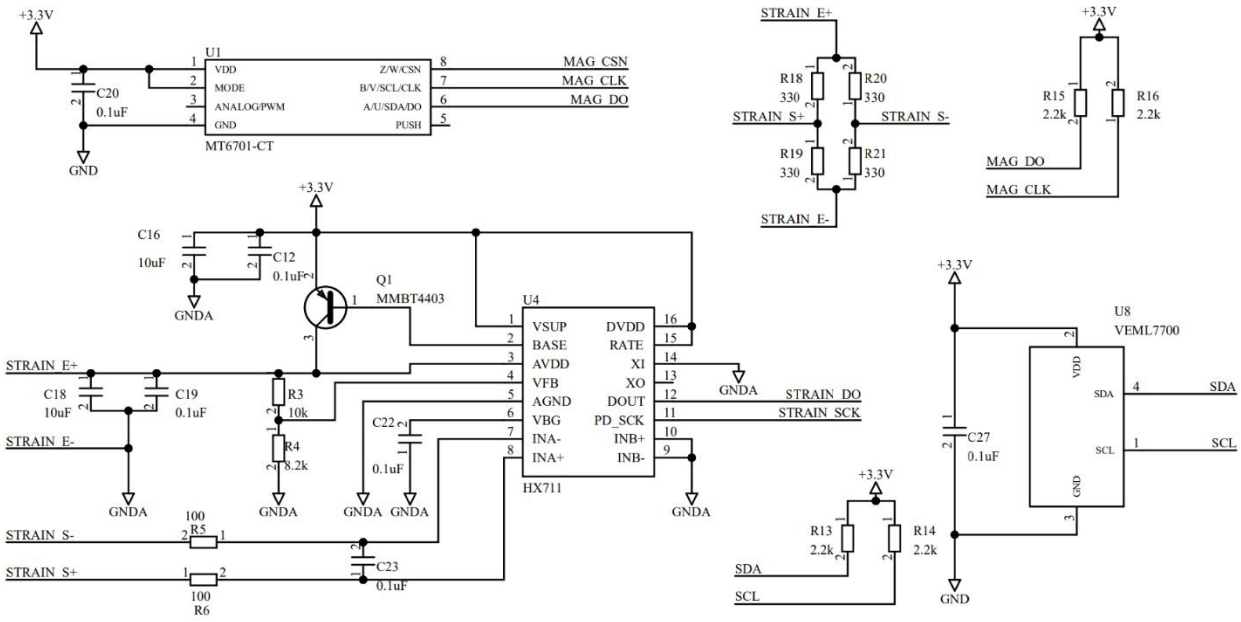
Elektroničko sklopovlje sastoji se od četiri ključne cjeline: USB napajanje i komunikacija, ulazni moduli sustava, izlazni moduli sustava te mikroupravljačka jedinica. Sklopovska shema svih navedenih modula vidljiva je u prilogu P.3.1

USB-C konektor koristi se za pružanje napajanja sustavu, kao i za uspostavljanje komunikacije i upisivanje programa u mikroupravljačku jedinicu putem UART protokola. Komunikacija je implementirana upotrebom CH340G modula, koji služi kao USB-to-UART pretvarač, koji omogućuje pouzdanu serijsku komunikaciju između USB sučelja i pametnog gumba. AP2114HA linearni regulator napona, regulira ulazni napon od 5V na 3.3V potrebnih za rad ostalih sustava. Tranzistori Q2A i Q2B, spojeni na DTR (*data terminal ready*) i RTS (*ready to send*) signale USB-to-UART čipa, resetiraju mikroupravljač u status potreban za prijenos firmvera. Slika 3.4. prikazuje sklopovsku shemu sustava napajanja i komunikacije.



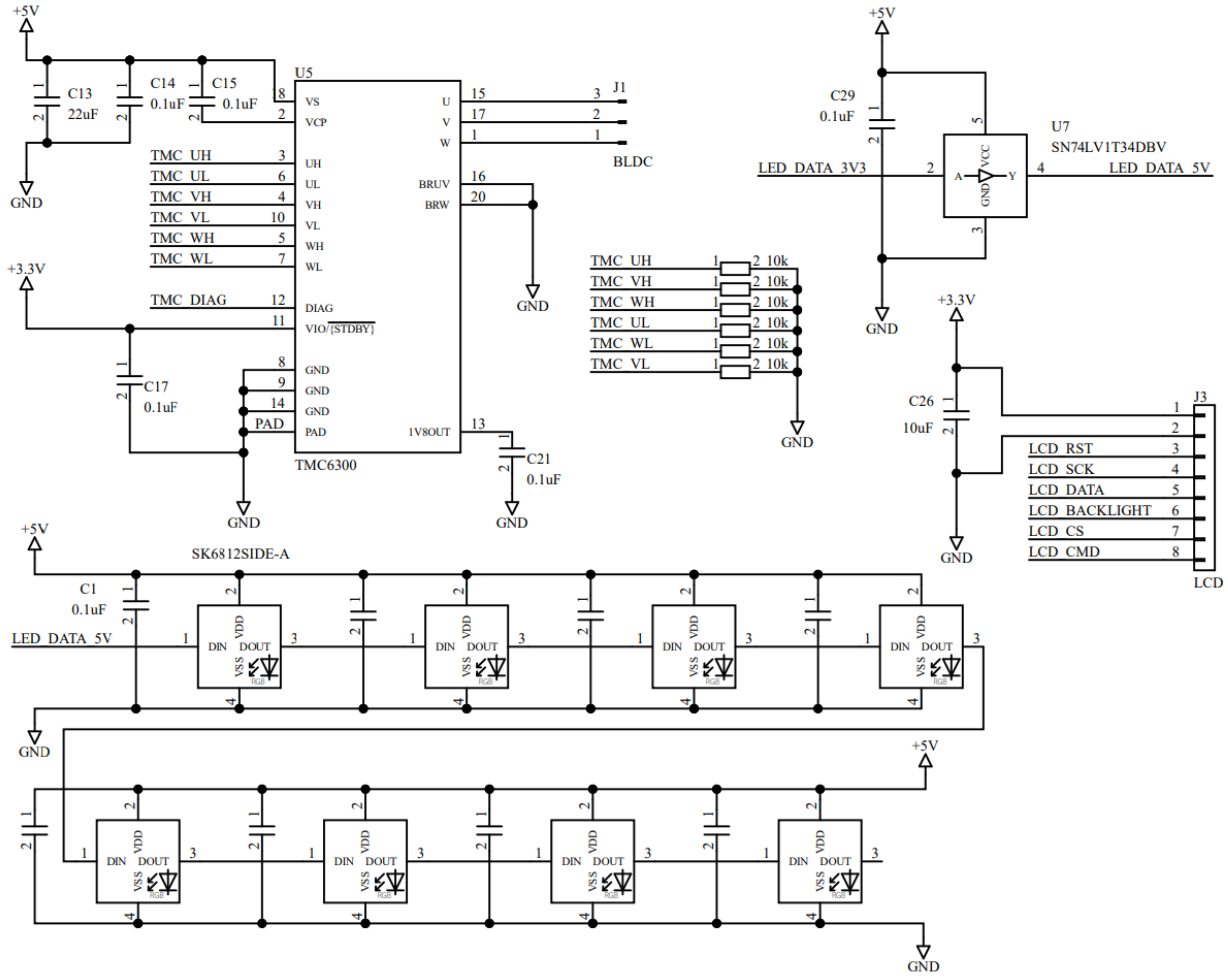
Slika 3.4. Shema USB sklopa

Ulazni moduli sustava, prikazani sklopovskom shemom na slici 3.5. osiguravaju mikroupravljačkoj jedinici potrebne ulazne signale za pravilno funkcioniranje i obradu ulaznih veličina. Magnetski enkoder MT6701 precizno detektira kut osovine BLDC motora i pruža povratnu informaciju mikroupravljaču o rotacijskom položaju prstena gumba. Komunikacija između mikroupravljača i magnetskog enkodera realizirana je koristeći SPI (*serial peripheral interface*) komunikacijski protokol. Senzor ambijentalnog osvjetljenja VEML7700 mjeri razinu svjetlosti u prostoriji i omogućava automatsku prilagodbu intenziteta osvjetljenja LCD-a i prstena LE dioda. Komunikacijski protokol korišten za prijenos podataka između senzora ambijentalnog osvjetljena i mikroupravljača je I2C (*inter-integrated circuit*). Senzor naprezanja (*strain sensor*) realiziran je koristeći otpornike R18, R19, R20 i R21 spojene u Wheatstoneov most. Prilikom deformacije njihov otpor se mijenja. Promjena otpora se detektira i digitalizira koristeći HX711 senzor težine. Ovim pristupom dobivamo kvantitativnu reprezentaciju sile koja djeluje na rotacijski prsten gumba. Komunikacijski protokol korišten za prijenos podataka između HX711 senzora težine i mikroupravljača je SPI.



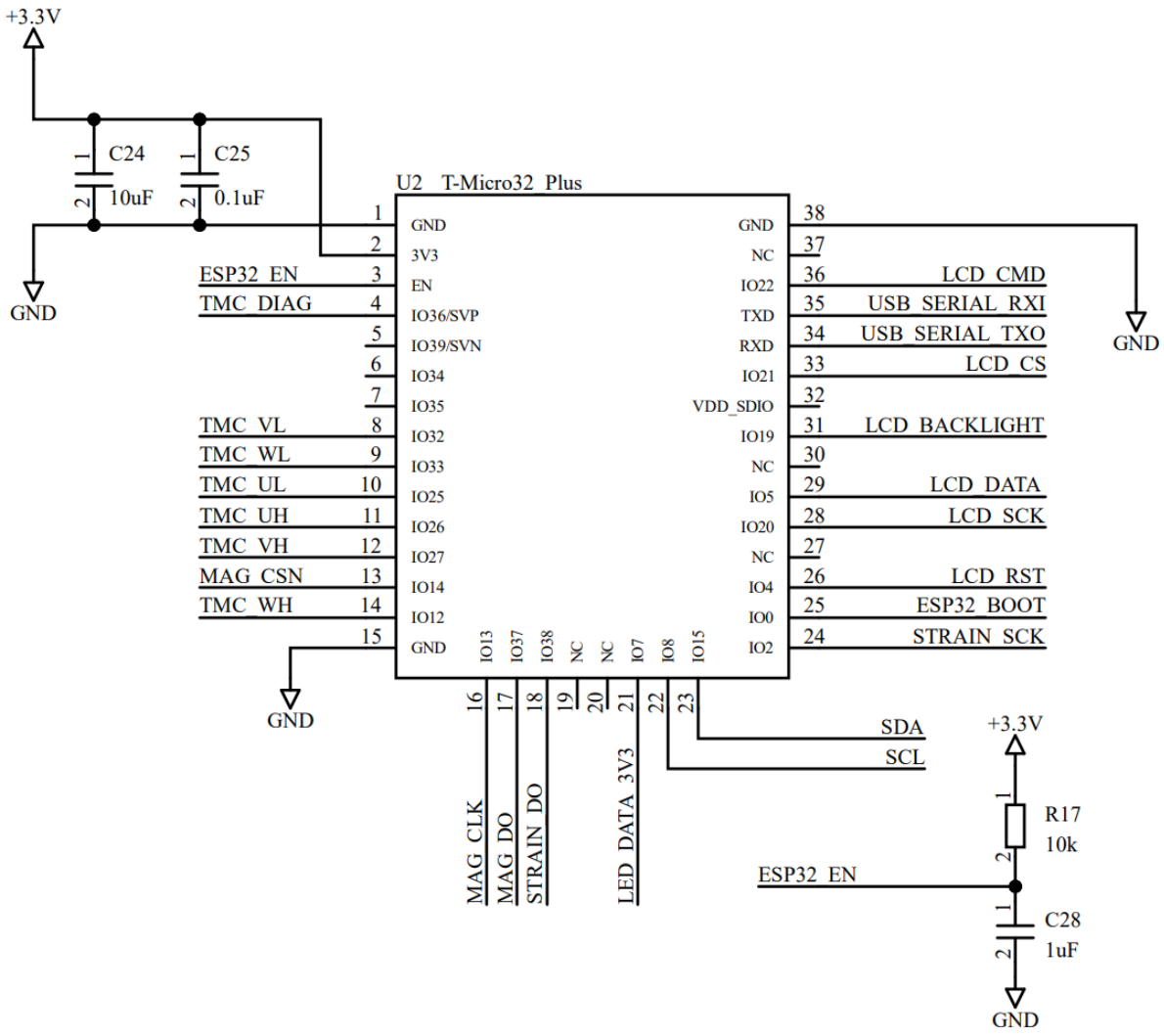
Slika 3.5. Shema sklopa ulaznih modula

Izlazni moduli sustava, prikazani sklopovskom shemom na slici 3.6. pružaju korisniku važne povratne informacije o stanju sustava. Kontroler BLDC motora, sa oznakom TMC6300, upravlja rotacijom motora s visokim stupnjem preciznosti, omogućujući korisniku da osjeti taktilnu povratnu informaciju prilikom interakcije s gumbom. S obzirom na zahtjeve sustava, kontroleru nisu bili potrebni vanjski tranzistori za upravljanje motorom. Mikroupravljač šalje kontroleru signale u realnom vremenu o stanju svake zavojnice motora. Prsten LE dioda realiziran je koristeći SK6812 individualno adresibilne RGB LE diode. Upotrebom ovih LE dioda osiguravamo kontrolu nad njihovim individualnim bojama. S obzirom da je naponska razina izlaza mikrupravljača 3.3V, a LE dioda 5V, potreban je pretvarač razine napona logičkih signala između mikroupravljača i LE dioda. LCD korišten u sustavu je GC9A01 okrugli ekran, dijagonale 1.27 inča. Ekran komunicira sa mikroupravljačem koristeći SPI komunikacijski protokol. LE diode i LCD služe kao vizualne indikacije stanja gumba, omogućujući korisniku jasnu i neposrednu percepciju informacija. Integriranjem taktilnih i vizualnih izlaznih modula, korisniku se omogućuje potpuno i sveobuhvatno iskustvo interakcije s proizvodom.



Slika 3.6. Shema sklopa izlaznih modula

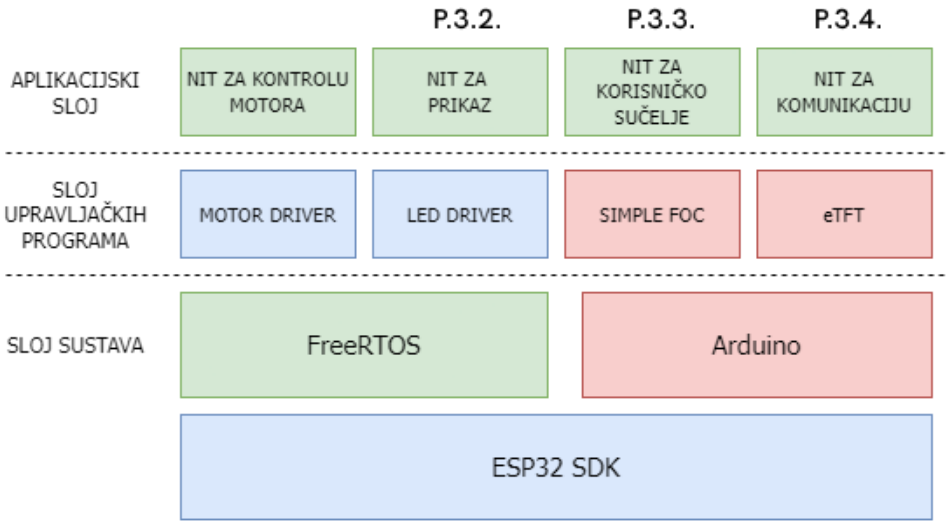
Mikroupravljačka jedinica korištena u sustavu je T-Micro32 Plus. Ova upravljača jedinica je bazirana na ESP32-PICO-D4 modulu koji omogućava bežičnu komunikaciju putem WiFi i Bluetooth tehnologija. Modul sadrži 4 MB flash memorije, sa taktom procesora do 240 MHz i dvije jezgre. U ovu jedinicu integrirani su svi ulazni moduli, čije se informacije obrađuju. Na temelju obrade podataka, sustav koristi izlazne module za generiranje povratne informacije korisniku, čime se omogućava iskustvo interakcije i kontrole pametnog gumba. Na slici 3.7. možemo vidjeti sklopovsku shemu upravljače jedinice sustava.



Slika 3.7. Shema sklopa mikroupravljačke jedinice

3.3. Realizacija programskog rješenja

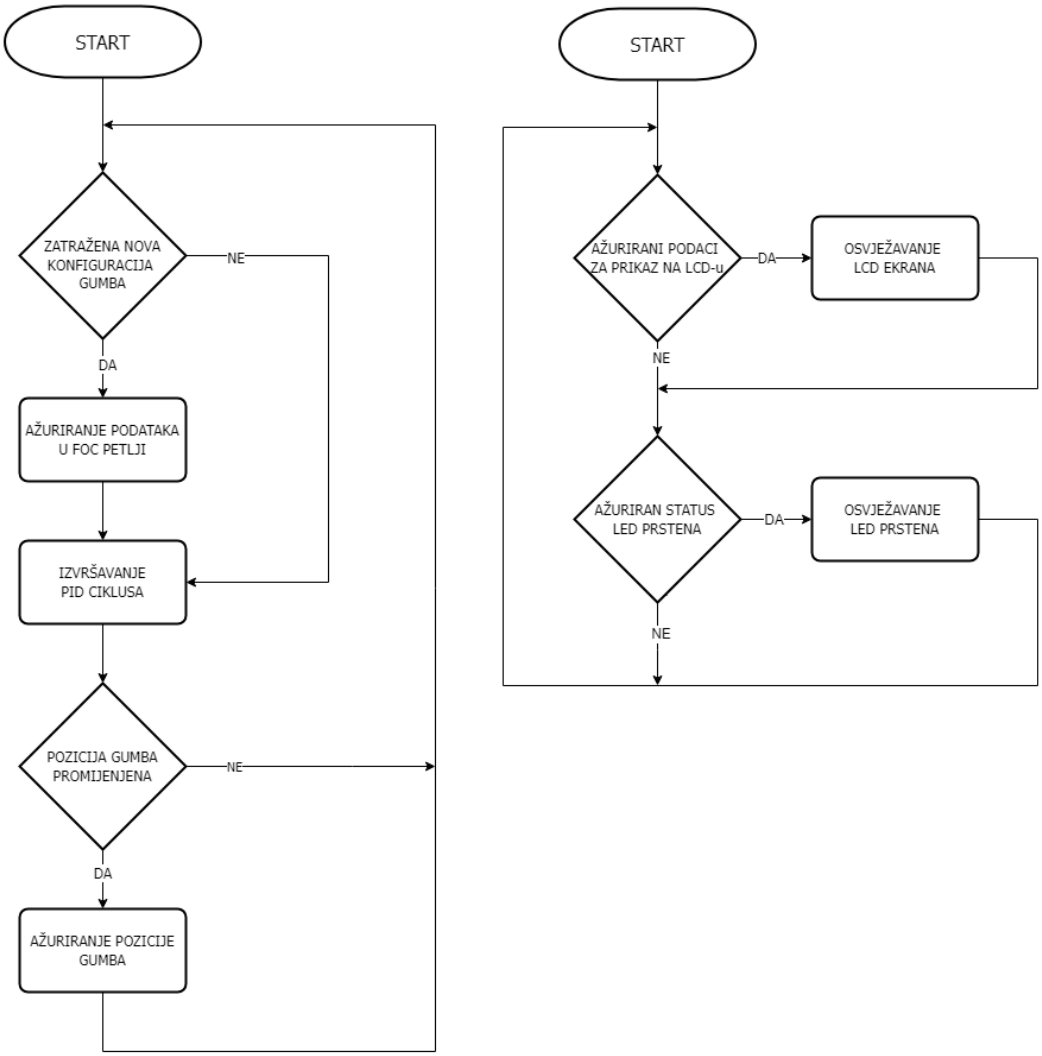
Korištenjem realnog vremenskog operacijskog sustava FreeRTOS (*real time operating system*), softversko rješenje osigurava visoku razinu determinizma i pouzdanosti. Četiri niti, svaka s jasno definiranim zadatkom, omogućuju paralelno izvršavanje različitih funkcionalnosti. Četiri niti rade paralelno, svaka optimizirana za specifičan zadatak. Nit za kontrolu motora osigurava precizno pozicioniranje, nit za prikaz ažurira stanja vizualnih modula sustava, nit za korisničko sučelje obrađuje korisničke ulaze, a nit za komunikaciju upravlja mrežnim povezivanjem. Slika 3.8. prikazuje arhitekturu slojeva sustava gdje su jasno prikazane granice različitih modula firmvera.



Slika 3.8. Arhitektura sustava

Nit za kontrolu motora, implementirana u [2], u realnom vremenu prikuplja podatke o kutnom položaju s magnetskog enkodera. Koristeći biblioteku SimpleFOC [3] (*field oriented control*), zasnovanu na vektorskom upravljanju, sustav kvantificira kutnu pogrešku između stvarnog i željenog položaja rotora. Implementirana PID kontrolna petlja djeluje korektivno, smanjujući uočenu pogrešku putem generiranja impulsa za motorni pogon. Time se osigurava precizno pozicioniranje rotora u željenu kutnu poziciju. Prikazano dijagramom na slici 3.9. (lijevo) nit za kontrolu motora, koristeći mehanizam reda čekanja (*queue*), učitava trenutnu konfiguraciju u sustav, ukoliko je ona zatražena, te svakim ciklusom beskonačne petlje izvršava FOC algoritam i popratno PID kontrolni algoritam.

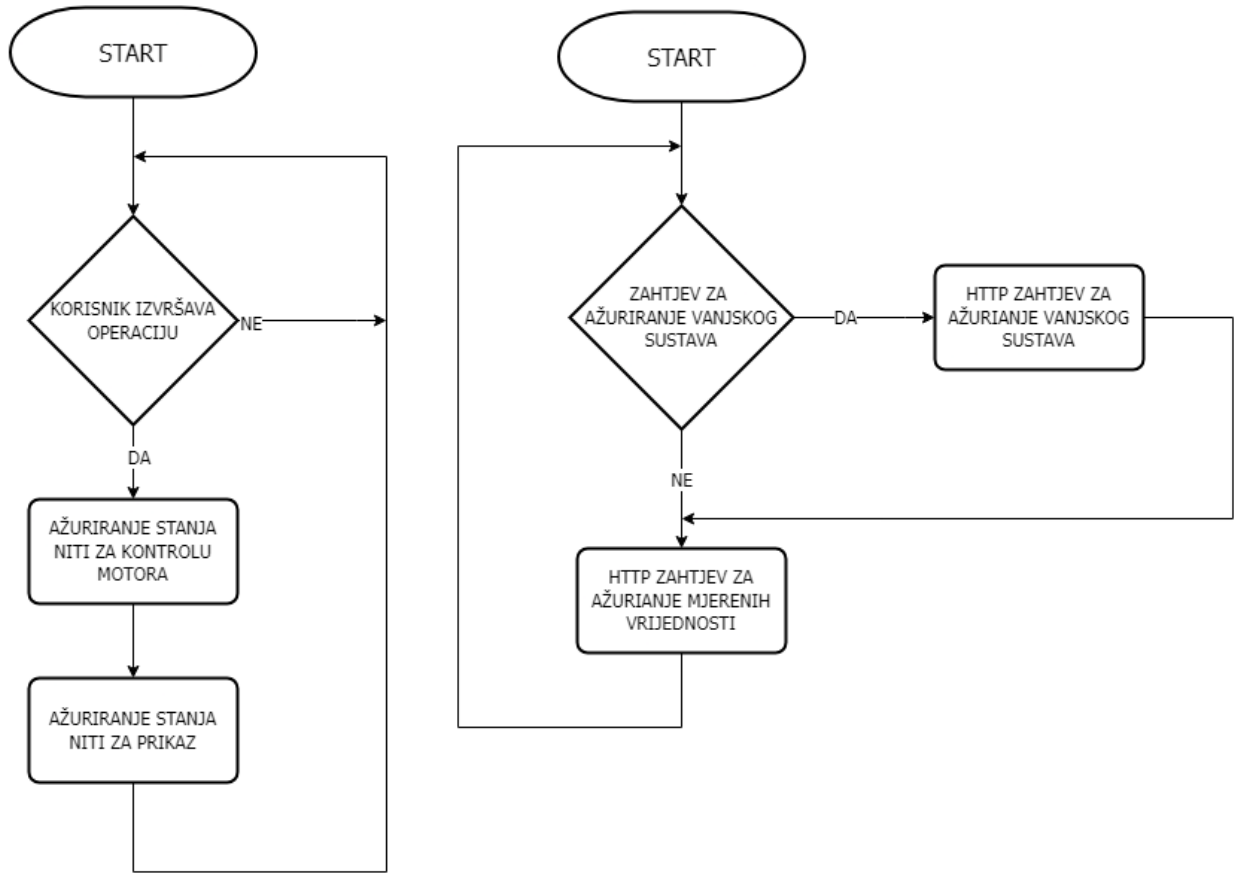
Nit za prikaz djeluje kao vizualni predstavnik sustava, dinamički ažurirajući LCD i LED prsten sukladno promjenama u korisničkom sučelju. Na temelju podataka primljenih od ostalih niti, ova nit u realnom vremenu prikazuje trenutno stanje sustava. Korištenjem mehanizma reda čekanja (*queue*), nit za prikaz je obaviještena o promjenama u korisničkom sučelju te odmah inicira ažuriranje prikaza. Slika 3.9. (desno) prikazuje dijagram tijeka niti za prikaz, iz kojeg možemo vidjeti da se komunikacija između LCD-a i LE dioda izvršava jedino u slučaju kada se podaci promjene. Detaljan prikaz logike niti za prikaz vidljiv je u izvornom kodu u prilogu P.3.2. Ovim pristupom štedi se procesorsko vrijeme i prepušta se ostalim prioritetnijim nitima.



Slika 3.9. Dijagram tijeka niti za kontrolu motora (*lijevo*) i ažuriranje prikaza (*desno*)

Nit korisničkog sučelja, posredujući između korisnika i sustava, prima korisničke ulaze i prenosi ih u obliku upravljačkih signala ostalim nitima, koje zatim izvršavaju zadatke u skladu s tim signalima. Prikazano dijagramom na slici 3.10. (lijevo) nit u beskonačnoj petlji, koristeći red čekanja, iščekuje korisnički ulaz, te prilikom događanja istog ažurira stanja niti za kontrolu motora i niti za prikaz. Izvorni kod niti za korisničko sučelje prikazan je u prilogu P3.3.

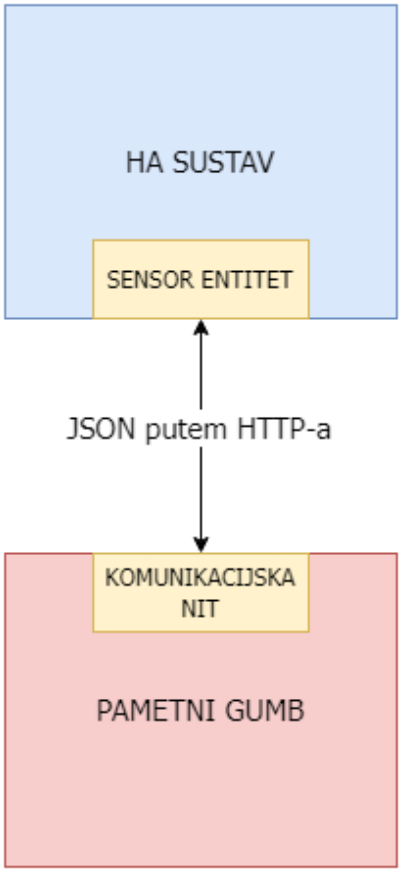
Nit za komunikaciju je zadužena za uspostavljanje i održavanje HTTP veza s vanjskim sustavom Home Assistant, omogućujući time razmjenu podataka i upravljanje vanjskim uređajima. Dijagram tijeka niti za komunikaciju, prikazanim na slici 3.10. (desno), opisuje princip komunikacije sa vanjskim sustavom. Nit u beskonačnoj petlji, u slučaju prisutnosti zahtjeva za ažuriranje vanjskog sustava, šalje HTTP zahtjev za promjenu. Mjerene vrijednosti sustava se ažuriraju u svakoj iteraciji beskonačne petlje. Izvorni kod niti za komunikaciju prikazan je u prilogu P3.24



Slika 3.10. Dijagram tijeka niti za korisničko sučelje (*lijevo*) i komunikaciju (*desno*)

3.3.1. Integracija u Home Assistant sustav

Pametni gumb, napredni haptički kontroler, integriran je u Home Assistant (HA) platformu za upravljanje pametnih domova. Konfiguracija korisničkog sučelja na pametnom gumbu u potpunosti je personalizirana putem HA sustava. Integracija s HA sustavom ostvarena je koristeći HTTP protokol koji služi kao temeljni komunikacijski sloj. HA sustav opskrbljuje pametni gumb podacima o trenutnom stanju relevantnih parametara u domu (temperature, intenzitet osvjetljenja), dok pametni gumb, šalje HA sustavu željene vrijednosti za kontrolu tih parametara. Slika 3.11. detaljnije prikazuje arhitekturu integracije u Home Assistant sustav, gdje se može vidjeti da je pametni gumb integriran kao *sensor entitet* u HA sustav. Koristeći HTTP protokol i mehanizam API-ja (*application programming interface*), pametni gumb prenosi informacije o trenutnom stanju prema HA sustavu.



Slika 3.11. Arhitektura integracije

3.3.2. Primjer regulacije razine osvjetljenja i temperature

Na slikama 3.12. i 3.13. prikazana je maketa pametnog doma koja realistično simulira uvjete upotrebe pametnog gumba u svakodnevnim scenarijima. Sustav je demonstriran na primjeru regulacije razine osvjetljenja i temperature, što omogućava prikaz ključnih funkcionalnosti uređaja u kontroliranom okruženju. Ova simulacija pruža korisnicima mogućnost interakcije sa sustavom, omogućujući im da se upoznaju s osnovnim konceptom pametnog gumba, kao i da testiraju njegove performanse i učinkovitost. Također, simulacijsko okruženje omogućava evaluaciju potencijalne korisničke upotrebljivosti u različitim scenarijima.

Slika 3.12. Primjer regulacije intenziteta rasvjete

Slika 3.13. Primjer regulacije temperature

4. TESTIRANJE I REZULTATI

4.1. Metodologija testiranja

Definirano je nekoliko scenarija upotrebe koji se primjenjuju na grupu odabranih korisnika koji su se susreli s razvijenim sustavom, kako bi ga koristili prema testnim scenarijima i izrazili svoje iskustvo upotrebe kroz deskriptivnu procjenu funkcionalnosti sustava, prema [4]. Kvalitativna procjena temelji se isključivo na iskustvu krajnjeg korisnika i osjećaju upotrebljivosti. Testiranje i validacija provode se na grupi od 17 osoba, s jednakim brojem muškaraca i žena, u kontroliranom okruženju koje simulira pametni dom u kojem se primjenjuju scenariji upotrebe.

Način upotrebe br. 1: Individualna upotreba

U ovom scenariju, pametni haptički gumb koristili bi pojedinci ili obitelji u svojim domovima za upravljanje različitim aspektima svog sustava automatizacije doma. To bi moglo uključivati upravljanje svjetlima, temperaturom, glazbom i drugim pametnim uređajima. Haptička povratna informacija pružila bi intuitivnije i zadovoljnije korisničko iskustvo, olakšavajući korisnicima upravljanje svojim domaćim okruženjem bez potrebe za gledanjem na ekran.

Način upotrebe br. 2: Grupna upotreba

U ovom scenariju, pametni haptički gumb bi koristilo više ljudi u zajedničkom prostoru, kao što je ured. Gumb bi se mogao koristiti za upravljanje zajedničkim uređajima, kao što su svjetla, grijanje, hlađenje i zvučni sustav. Haptička povratna informacija pomogla bi spriječiti korisnike od slučajnog mijenjanja postavki, te bi također pružila zajedničko korisničko iskustvo.

Način upotrebe br. 3: Upotreba u hotelu

U ovom scenariju, pametni haptički gumb koristili bi gosti u hotelskim sobama za upravljanje različitim aspektima svog prostora u sobi, kao što su svjetla, temperatura i zavjese. Haptička povratna informacija pružila bi luksuznije i intuitivnije korisničko iskustvo, olakšavajući gostima upravljanje prostorom u sobi bez potrebe za mukotrpnim radom s nepoznatim kontrolama.

4.2. Rezultati testiranja

Testiranje i postupci validacije provedeni su u skladu s izjavama navedenim u prethodnom poglavlju. Scenariji upotrebe primijenjeni su u simuliranom okruženju pametnog doma, a testna skupina krajnjih korisnika izrazila je svoje mišljenje na sljedeći način.

Način upotrebe br. 1: Individualna upotreba

Od sudionika je zatraženo da zamisle svoje domove opremljene pametnim haptičkim gumbom koji bi se nesmetano uklopio u njihov životni prostor. Vođeni su kroz scenarije u kojima bi uređaj jednostavno kontrolirao različite aspekte njihovog kućnog okruženja, od upravljanja svjetlom i temperaturom do upravljanja glazbom i drugim pametnim uređajima. Kroz ova pitanja, sudionici su procijenili svoju lakoću prihvaćanja takvog sustava u svojim domovima. Njihovo iskustvo i procjena kvalitete sustava sažeti su u tablici 4.1.

Tab 4.1. Procjena kvalitete i iskustvo u korištenju taktilno-haptičkog kontrolera.

Ocjena kvalitete	Sudionici
Vrlo lako	7
Relativno lako	5
Neutralno	5
Relativno teško	0
Vrlo teško	0

Način upotrebe br. 2: Grupna upotreba

Od sudionika je zatraženo da zamisle zajedničke prostore, poput ureda, opremljene pametnim haptičkim gumbom koji bi se skladno uklopio u njihov kolaborativni radni prostor. Bili su vođeni kroz scenarije u kojima bi uređaj jednostavno kontrolirao različite aspekte zajedničkog okruženja, od podešavanja svjetla i temperature do upravljanja glazbom i drugim zajedničkim uređajima. Kroz ova pitanja, sudionici su procijenili svoju lakoću prihvaćanja takvog sustava u svojim zajedničkim prostorima. Njihovo iskustvo i procjena kvalitete sustava sažeti su u tablici 4.2.

Tab 4.2. Procjena kvalitete i iskustvo u korištenju taktilno-haptičkog kontrolera.

Ocjena kvalitete	Sudionici
Vrlo lako	4
Relativno lako	4
Neutralno	5
Relativno teško	4
Vrlo teško	0

Način upotrebe br. 3: Upotreba u hotelu

Od sudionika je zatraženo da zamisle hotelske sobe opremljene pametnim haptičkim gumbom, koji bi se skladno uklopio u njihov privremeni smještaj. Vođeni su kroz scenarije u kojima bi uređaj jednostavno kontrolirao različite aspekte okoliša hotelske sobe, od upravljanja svjetlima i temperaturom do upravljanja zavjesama i drugim hotelskim sadržajima. Kroz ova pitanja, sudionici su bili potaknuti da procjene svoju percipiranu lakoću prihvaćanja takvog sustava u hotelskim sobama. Tablica 4.3. jasnije prikazuje opće zadovoljstvo krajnjih korisnika, s naglaskom na lakoću prihvaćanja sustava.

Tab 4.3. Procjena kvalitete i iskustvo u korištenju taktilno-haptičkog kontrolera.

Ocjena kvalitete	Sudionici
Vrlo lako	6
Relativno lako	3
Neutralno	4
Relativno teško	3
Vrlo teško	1

Korisnici su pozitivno reagirali na koncept taktilno-haptičkih kontrolera u pametnim domovima. Ovi kontroleri pružaju korisniku osjetljivu povratnu informaciju o stanju sustava, čime se podiže razina interakcije s pametnim kućnim uređajima. Jednostavna mrežna povezivost omogućuje još veću razinu automatizacije i udobnosti u domu. Općenito, rezultati istraživanja potvrđuju potrebu za daljnjim razvojem i integracijom ovakvih sustava u koncept pametnog življenja, jer korisnici sve više traže intuitivnije i prirodnije načine interakcije s tehnologijom u svakodnevnom životu.

5. ZAKLJUČAK

Sa napretkom tehnologije i sve većom potrebom za automatizacijom, moramo težiti obnovi ravnoteže između digitalne kontrole i fizičke interakcije. Uvođenje haptičkih sučelja, kao što je pametni haptički gumb, nudi obećavajuće rješenje za ovaj izazov, omogućujući korisnicima da ponovno steknu osjećaj kontrole u svojim digitaliziranim okruženjima. Pozitivne ocjene sudionika i njihov entuzijazam svjedoče o potencijalu haptičkih sučelja da osvježe domenu korisničkog iskustva, omogućavajući intuitivnu kontrolu i vraćajući osjećaj fizičke povezanosti u sve digitalnijem svijetu [5]. Jednostavnom integracijom ovih intuitivnih mehanizama kontrole u pametne uređaje i okruženja, možemo stvoriti skladniji i privlačniji model interakcije čovjeka i računala koji zadovoljava i udobnost digitalne kontrole i urođenu ljudsku želju za opipljivim iskustvima [6]. Ponovni uspon haptičkih sučelja označava značajan pomak u interakciji korisnika, pružajući obećavajuće rješenje za rastući raskorak između pojedinaca i njihove fizičke okoline. Osim rastuće potražnje za haptičkim sučeljima u kućnoj automatizaciji, potencijalne primjene sežu daleko izvan granica osobne upotrebe. Svestranost i intuitivna priroda haptičkih mehanizama kontrole čine ih prikladnima za raznoliku paletu okruženja, od hotela i ugostiteljskih postavki do industrijskih i komercijalnih primjena. Tržište za taktilno-haptičke kontrolere pametnih domova još je u ranoj fazi razvoja, s ogromnim potencijalom za rast i inovacije. Kako se tehnologija nastavlja unaprjeđivati, a potražnja za intuitivnim, korisniku prijateljskim sučeljima raste, možemo očekivati širenje haptičkih kontrolnih rješenja u širokom spektru primjena, transformirajući način na koji komuniciramo s našim sve više digitaliziranim svijetom.

LITERATURA

[1] Fortune Business Insighs, [Mrežno], https://www.fortunebusinessinsights.com/industry-reports/home-automation-market-100074

[2] Scott Bezek, GitHub, [Mrežno], https://github.com/scottbez1/smartknob

[3] Antun Skuric, GitHub [Mrežno] https://github.com/simplefoc/Arduino-FOC

[4] Radić, M., Keser, T., Blažević, D. (2024). An Advanced Tactile-Haptic Controller for Smart Home. (OTO 2023). OTO 2023. https://doi.org/10.1007/978-3-031-51494-4_35

[5] Scorcia, M. S., Leotta, A., & Spano, L. D. Haptic Knobs for Smart Home Appliances: A User-Centered Design Approach. p. 4 (2022).

[6] Jones, L., & Cross, S. A. N. Haptics: Neuroscience, Devices, Modeling, and Applications. pp. 3, 6 (2018).

SAŽETAK

Ideja pametnih domova temelji se na automatizaciji gotovo svih aspekata svakodnevnog života. Iako automatizacija primarno upravlja kućanskim sustavima, kao što su grijanje, hlađenje i rasvjeta, ona također omogućuje komunikaciju s drugim sustavima za sveobuhvatno upravljanje domom. Većina pametnih sustava usredotočena je na upravljanje izvršnim elementima, poput upravljanjem temperaturom, intenzitetom svjetla i nadzorom potrošnje energije. Međutim, malo je rješenja koja se fokusiraju na intuitivnu interakciju između korisnika i uređaja. Ovaj rad predstavlja razvoj naprednog kontrolera za pametne domove koji koristi taktilno-haptičku povratnu informaciju kako bi simulirao osjećaj i iskustvo korištenja tradicionalnih fizičkih prekidača. Koristeći IoT platformu, razvijeno je rješenje koje omogućuje precizno upravljanje temperaturom i rasvjetom. U okviru ovog rada provedeno je istraživanje optimalnih taktilno-haptičkih odnosa za različite vrste interakcija kako bi se osiguralo intuitivno i ugodno korisničko iskustvo.

Ključne riječi: Pametni dom, IoT sustavi, Taktilno-haptičko upravljanje

ABSTRACT

Smart Knob – Advanced haptic controller

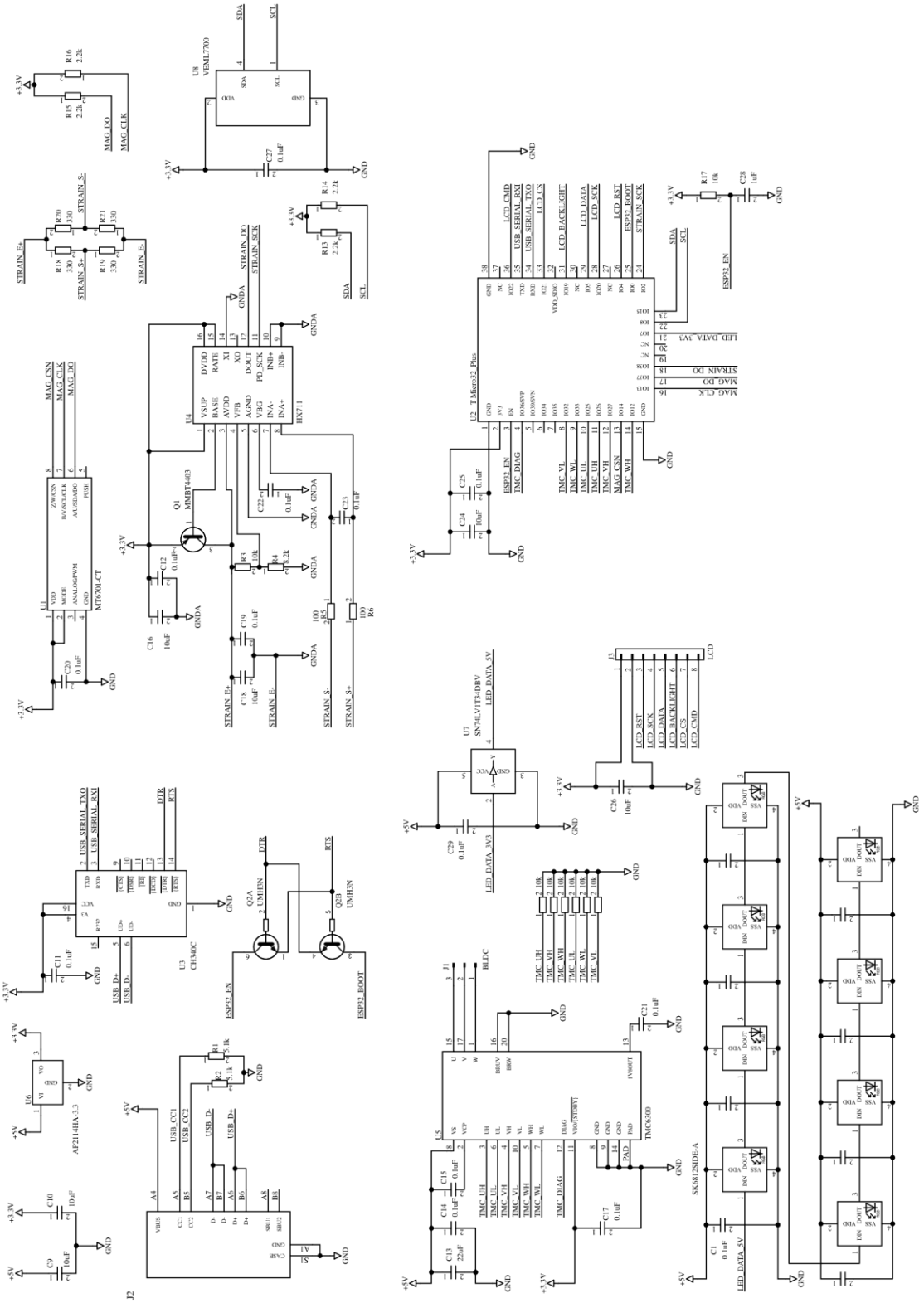
The concept of smart homes hinges on automating nearly every aspect of daily life. While automation primarily controls household systems such as heating, cooling, and lighting, it also enables communication with other systems for comprehensive home management. Most smart systems focus on controlling output devices, such as adjusting temperature, light intensity, and monitoring energy consumption. However, few solutions focus on intuitive interaction between the user and devices. This work presents the development of an advanced controller for smart homes that uses tactile-haptic feedback to simulate the feel and experience of using traditional physical switches. Using an IoT platform, a solution has been developed that enables precise control of temperature and lighting. As part of this work, research has been conducted on optimal tactile-haptic relationships for different types of interactions to ensure an intuitive and pleasant user experience.

Keywords: Smart home, IoT systems, Tactile-haptic control.

ŽIVOTOPIS

Miloš Radić, rođen 16.07.1998 godine. Odrastao u prigradskom naselju Tenja te tamo završio Osnovnu školu Tenja 2014. godine. U osmom razredu osnovne škole pohađa državno natjecanje iz predmeta tehničke kulture u kategoriji elektronika. Nakon osnovne škole upisuje Elektrotehničku i prometnu školu u Osijeku, smjer Tehničar za elektroniku. Tijekom srednjoškolskog obrazovanja, pohađa nekoliko natjecanja u inovacijama i elektronici. Godine 2016. na sajmovima inovacija „Inventum“ u Iloku i „Inova“ u Zagrebu, osvaja dvije zlatne medalje za učenički rad pod nazivom „Računalo za električni auto“. Sa istim radom nastupa kao izlagač na prvom „Osijek mini Maker Faire-u“ 2017. godine. Tijekom četvrte godine srednje škole aktivno je volontirao u Zajednici tehničke kulture grada Osijeka. Nakon završene srednje škole upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, na prijediplomski studijski program Elektrotehnika. Od 2020. godine profesionalno se bavi razvojem softvera za ugradbena računala.

PRILOZI



P.3.1. Sklopovska shema sustava

```cpp
#include "display_task.h"
#include "semaphore_guard.h"
#include "wifi_task.h"
#include "font/roboto_light_60.h"

// ESP32 Led Control kanal za upravljanje LED-diodama
static const uint8_t LEDC_CHANNEL_LCD_BACKLIGHT = 0;

// Kreiranje niti kao nasljeđena klasa
DisplayTask::DisplayTask(const uint8_t task_core) : Task{"Display", 2048, 1, task_core}
{
 // Inicijalizacija queue mehanizma
 knob_state_queue_ = xQueueCreate(1, sizeof(PB_SmartKnobState));
 assert(knob_state_queue_ != NULL);

 // Inicijalizacija semaphore mehanizma
 mutex_ = xSemaphoreCreateMutex();
 assert(mutex_ != NULL);
}

DisplayTask::~DisplayTask()
{
 // Brisanje kontrolnih mehanizama
 vQueueDelete(knob_state_queue_);
 vSemaphoreDelete(mutex_);
}

void DisplayTask::run()
{
 // Inicijalizacija LCD
 tft_.begin();
 tft_.invertDisplay(1);
 tft_.setRotation(0);
 tft_.fillScreen(TFT_DARKGREEN);

 // Inicijalizacija LED Control mehanizma
 ledcSetup(LEDC_CHANNEL_LCD_BACKLIGHT, 5000, 16);
 ledcAttachPin(PIN_LCD_BACKLIGHT, LEDC_CHANNEL_LCD_BACKLIGHT);
 ledcWrite(LEDC_CHANNEL_LCD_BACKLIGHT, UINT16_MAX);

 spr_.setColorDepth(8);

 // Generiranje LCD sprite-a
 if (spr_.createSprite(TFT_WIDTH, TFT_HEIGHT) == nullptr)
 {
  log("ERROR: sprite allocation failed!");
  tft_.fillScreen(TFT_RED);
 }
 else
 {
  log("Sprite created!");
  tft_.fillScreen(TFT_PURPLE);
 }

 spr_.setTextColor(0xFFFF, TFT_BLACK);

 PB_SmartKnobState state;

 const int RADIUS = TFT_WIDTH / 2;
 const uint16_t FILL_COLOR = spr_.color565(90, 18, 151);
 const uint16_t DOT_COLOR = spr_.color565(80, 100, 200);

 spr_.setTextDatum(CC_DATUM);
 spr_.setTextColor(TFT_WHITE);

 // Povezivanje na WiFi mrežu
 while (1)
 {
  // Prikaz statusa povezivanja
  if (WiFiTask::instance().getStatus() == WIFI_CONNECTING)
  {
   spr_.fillSprite(TFT_BLACK);
   spr_.setFreeFont(&DESCRIPTION_FONT);
   spr_.drawString("CONNECTING", TFT_WIDTH / 2, TFT_HEIGHT / 2, 1);
   spr_.pushSprite(0, 0);
   delay(100);
   continue;
  }

  // Čekanje na promjenu statusa sustava
  if (xQueueReceive(knob_state_queue_, &state, portMAX_DELAY) == pdFALSE)
  {
   continue;
  }

  // Ažuriraj trenutnu vrijednost u komunikacijskoj niti
  WiFiTask::instance().setValue(0, state.current_position);

  // Očisti LCD
  spr_.fillSprite(TFT_BLACK);

  // Prikazivanje namještene vrijednosti kontrolirane veličine
```

```
  spr_.setTextColor(TFT_PINK);
  spr_.setFreeFont(&DESCRIPTION_FONT);
  spr_.drawString(String() + WiFiTask::instance().getConfig(0).valueState.currentValue, TFT_WIDTH / 2, TFT_HEIGHT / 2 - VALUE_OFFSET, 1);

  spr_.setTextColor(TFT_WHITE);
  spr_.setFreeFont(&Roboto_Light_60);
  spr_.drawString(String() + WiFiTask::instance().getConfig(0).valueState.setValue, TFT_WIDTH / 2, TFT_HEIGHT / 2 + VALUE_OFFSET, 1);
  spr_.setFreeFont(&DESCRIPTION_FONT);

  // Prikaz naziva kontrolirane veličine
  int32_t line_y = TFT_HEIGHT / 2 + DESCRIPTION_Y_OFFSET;
  char *start = WiFiTask::instance().getConfig(0).text;
  char *end = start + strlen(WiFiTask::instance().getConfig(0).text);

  while (start < end)
  {
   char *newline = strchr(start, '\n');
   if (newline == nullptr)
   {
    newline = end;
   }

   char buf[sizeof(WiFiTask::instance().getConfig(0).text)] = {};
   strncat(buf, start, min(sizeof(buf) - 1, (size_t)(newline - start)));
   spr_.setTextColor(TFT_WHITE);
   spr_.drawString(String(buf), TFT_WIDTH / 2, line_y + 50, 1);
   start = newline + 1;
   line_y += spr_.fontHeight(1);
  }

  // Izračun pozicije indikacijske točke u ovisnosti o trenutnoj konfiguraciji minimalne i maksimalne vrijednosti kontrolirane vrijednosti
  float left_bound = PI / 2;
  float adjusted_sub_position = state.sub_position_unit * state.config.position_width_radians;
  if (state.config.num_positions > 0)
  {
   if (state.current_position == 0 && state.sub_position_unit < 0)
   {
    adjusted_sub_position = -logf(1 - state.sub_position_unit * state.config.position_width_radians / 5 / PI * 180) * 5 * PI / 180;
   }
   else if (state.current_position == state.config.num_positions - 1 && state.sub_position_unit > 0)
   {
    adjusted_sub_position = logf(1 + state.sub_position_unit * state.config.position_width_radians / 5 / PI * 180) * 5 * PI / 180;
   }
  }


  // Iscrtavanje indikacijske točke na LCD.
  float raw_angle = left_bound - state.current_position * state.config.position_width_radians;
  float adjusted_angle = raw_angle - adjusted_sub_position;
  if (state.config.num_positions > 0 && ((state.current_position == 0 && state.sub_position_unit < 0) || (state.current_position ==
state.config.num_positions - 1 && state.sub_position_unit > 0)))
  {
   spr_.fillCircle(TFT_WIDTH / 2 + (RADIUS - 10) * cosf(raw_angle), TFT_HEIGHT / 2 - (RADIUS - 10) * sinf(raw_angle), 5, DOT_COLOR);
   if (raw_angle < adjusted_angle)
   {
    for (float r = raw_angle; r <= adjusted_angle; r += 2 * PI / 180)
    {
     spr_.fillCircle(TFT_WIDTH / 2 + (RADIUS - 10) * cosf(r), TFT_HEIGHT / 2 - (RADIUS - 10) * sinf(r), 2, DOT_COLOR);
    }
    spr_.fillCircle(TFT_WIDTH / 2 + (RADIUS - 10) * cosf(adjusted_angle), TFT_HEIGHT / 2 - (RADIUS - 10) * sinf(adjusted_angle), 2, DOT_COLOR);
   }
   else
   {
    for (float r = raw_angle; r >= adjusted_angle; r -= 2 * PI / 180)
    {
     spr_.fillCircle(TFT_WIDTH / 2 + (RADIUS - 10) * cosf(r), TFT_HEIGHT / 2 - (RADIUS - 10) * sinf(r), 2, DOT_COLOR);
    }
    spr_.fillCircle(TFT_WIDTH / 2 + (RADIUS - 10) * cosf(adjusted_angle), TFT_HEIGHT / 2 - (RADIUS - 10) * sinf(adjusted_angle), 2, DOT_COLOR);
   }
  }
  else
  {
   spr_.fillCircle(TFT_WIDTH / 2 + (RADIUS - 10) * cosf(adjusted_angle), TFT_HEIGHT / 2 - (RADIUS - 10) * sinf(adjusted_angle), 5, DOT_COLOR);
  }


  // Pošalji iscrtani sprite na LCD.
  spr_.pushSprite(0, 0);

  // Zaključaj semafor za kontrolu pristupa LED Control mehanizmu
  {
   SemaphoreGuard lock(mutex_);
   ledcWrite(LEDC_CHANNEL_LCD_BACKLIGHT, brightness_);
  }

  delay(2);
 }
}
```

```
QueueHandle_t DisplayTask::getKnobStateQueue()
{
 return knob_state_queue_;
}

void DisplayTask::setBrightness(uint16_t brightness)
{
 SemaphoreGuard lock(mutex_);
 brightness_ = brightness;
}
```

P.3.2. Izvorni kod niti za prikaz

```
#include <FastLED.h>
#include <HX711.h>
#include <Adafruit_VEML7700.h>
#include "interface_task.h"
#include "util.h"
#include "wifi_task.h"

#define COUNT_OF(A) (sizeof(A) / sizeof(A[0]))

HX711 scale;
Adafruit_VEML7700 veml = Adafruit_VEML7700();

static PB_SmartKnobConfig configs[] = {

    // Postavljanje haptičkih konfiguracija
    {
        4,              // Broj podioka
        0,              // Početna pozicija
        40 * PI / 180, // Širina podioka u radijanima
        0.2,            // Tvrdoća podioka
        3,              // Sila odupiranja na min i max ekstremima
        1.1,            // Točka preskakanja
        "CONFIG",
    },
    {
        100,
        0,
        2.5 * PI / 180,
        0.7,
        3,
        1.1,
        "CONTROL",
    }};

// Kreiranje niti kao nasljeđena klasa
InterfaceTask::InterfaceTask(const uint8_t task_core, MotorTask &motor_task, DisplayTask *display_task) : Task("Interface", 3000, 1, task_core),
                                                                                                           stream_(),
                                                                                                           motor_task_(motor_task),
                                                                                                           display_task_(display_task),
                                                                                                           plaintext_protocol_(stream_,
motor_task_),
                                                                                                           proto_protocol_(stream_, motor_task_)
{

  assert(display_task != nullptr);

  // Inicijalizacija queue
  knob_state_queue_ = xQueueCreate(1, sizeof(PB_SmartKnobState));
  assert(knob_state_queue_ != NULL);
}

void InterfaceTask::run()
{
  // Inicijalizacija I2C komunikacije sa senzorom ambijentalnog osvjetljenja
  Wire.begin(PIN_SDA, PIN_SCL);
  Wire.setClock(400000);
  if (veml.begin())
  {
    veml.setGain(VEML7700_GAIN_2);
    veml.setIntegrationTime(VEML7700_IT_400MS);
  }
  else
  {
    log("ALS sensor not found!");
  }

  // Inicijalizacija komunikacije sa senzorom težine
  scale.begin(38, 2);

  // Petlja korisničkog sučelja
  while (1)
  {
    PB_SmartKnobState state;
```

```
    // Provjera promjene ambijentalnog osvjetljenja kroz vrijeme
    const float LUX_ALPHA = 0.005;
    static float lux_avg;
    float lux = veml.readLux();
    lux_avg = lux * LUX_ALPHA + lux_avg * (1 - LUX_ALPHA);
    static uint32_t last_als;
    if (millis() - last_als > 1000)
    {
      last_als = millis();
    }

    // Koeficijent mjerne jedinice za količinu pritiska
    float press_value_unit = 0;
    if (scale.wait_ready_timeout(100))
    {
      // Isčitavanje vrijednosti pritiska na prsten
      int32_t reading = scale.read();

      const int32_t lower = -150000;
      const int32_t upper = -90000;
      if (reading >= lower - (upper - lower) && reading < upper + (upper - lower) * 2)
      {
        long value = CLAMP(reading, lower, upper);
        press_value_unit = 1. * (value - lower) / (upper - lower);

        static bool pressed;

        // Simulacija pritiska na gumb
        if (!pressed && press_value_unit > 0.75)
        {
          motor_task_.playHaptic(true);
          pressed = true;
          // Ažuriranje trenutne konfiguracije
          changeConfig(true);
        }
        else if (pressed && press_value_unit < 0.25)
        {
          motor_task_.playHaptic(false);
          pressed = false;
        }
      }
    }

    // Ažuriranje razine osvjetljenja LCD-a
    uint16_t brightness = UINT16_MAX;
    brightness = (uint16_t)CLAMP(lux_avg * 13000, (float)1280, (float)UINT16_MAX);
    display_task_->setBrightness(brightness);

    delay(1);
  }
}

// Metoda za ažuriranje haptičkih konfiguracija
void InterfaceTask::changeConfig(bool next)
{
  if (next)
  {
    current_config_ = (current_config_ + 1) % COUNT_OF(configs);
  }
  else
  {
    if (current_config_ == 0)
    {
      current_config_ = COUNT_OF(configs) - 1;
    }
    else
    {
      current_config_--;
    }
  }
  motor_task_.setConfig(configs[current_config_]);
}
```

P.3.3. Izvorni kod niti za korisničko sučelje

```cpp
#include "wifi_task.h"
#include "WiFi.h"
#include "ArduinoJson.h"

// Kreiranje niti kao nasljeđena klasa
WiFiTask::WiFiTask() : Task{"WiFi Task", 4096, 4, 1}
{
}

// Ulazna funkcija za nit
void WiFiTask::run()
{
    // Inicijalizacija WiFi mehanizma
    status = WIFI_CONNECTING;
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    delay(100);

    // Započinjanje povezivanja sa AP
    WiFi.begin(WIFI_SSID, WIFI_PASS);

    while (WiFi.status() != WL_CONNECTED)
    {
        if (WiFi.status() == WL_CONNECT_FAILED)
        {
            // U slučaju neuspjelog povezivanja nit iskače iz beskonačne petlje
            // što rezultira ponovnim pokretanjem uređaja
            log("Device failed to connect, cannot proceed, rebooting");
            return;
        }
        delay(5000);
    }

    // U slučaju neuspjelog iščitavanja konfiguracije
    // nit iskače iz beskonačne petlje
    // što rezultira ponovnim pokretanjem uređaja
    if (!loadConfigs())
    {
        log("Device failed to configure, cannot proceed");
        return;
    }

    // Ažuriraj status povezanosti
    status = WIFI_CONNECTED;

    // Beskonačna petlja niti
    while (1)
    {
        for (int i = 0; i < configCount; i++)
        {
            // Ukoliko se namještena vrijednost promjenila, pošalji zahtjev prema vanjskom sustavu
            if (menuConfigs[i].valueState.isChanged)
            {
                httpPost("/api/states/" + menuConfigs[i].setEntity,
                         "{\"enity_id\":\"" + menuConfigs[i].setEntity + "\",   \"state\":\"" + String(menuConfigs[i].valueState.setValue) + "}");
                menuConfigs[i].valueState.isChanged = false;
            }

            // Ažuriranje trenutne vrijednosti mjerenih veličina
            String entityJson = httpGet("/api/states/" + menuConfigs[i].getEntity);
            int entitnyState = parseStateFromJson(entityJson);
            if (entitnyState < 0)
                continue;

            menuConfigs[i].valueState.currentValue = entitnyState;
            log(String(menuConfigs[i].text + ": " + String(menuConfigs[i].valueState.currentValue)).c_str());
        }

        delay(500);
    }
}

// Metoda za parsiranje stanja varijable iz JSON formata
int WiFiTask::parseStateFromJson(String json)
{
    JsonDocument document;
    DeserializationError error = deserializeJson(document, json);
    if (error.code() != DeserializationError::Ok)
        return -1;

    if (!document.containsKey("state"))
        return -1;

    return document["state"].as<int>();
}

// Metoda za učitavanje konfiguracije gumba iz JSON formata
bool WiFiTask::loadConfigs()
{
    String entityText = httpGet("/api/states/sensor.smart_knob");
    JsonDocument deviceEntity;
    DeserializationError error = deserializeJson(deviceEntity, entityText);
```

```
    if (error.code() != DeserializationError::Ok)
        return false;

    String deviceConfigJson = deviceEntity["attributes"]["configuration"].as<String>();
    JsonDocument deviceConfig;

    error = deserializeJson(deviceConfig, deviceConfigJson);
    if (error.code() != DeserializationError::Ok)
        return false;

    JsonArray jsonConfigs = deviceConfig["menus"].as<JsonArray>();

    int configCnt = 0;
    for (configCnt = 0; configCnt < jsonConfigs.size(); configCnt++)
    {
        JsonVariant menuItem = jsonConfigs[configCnt];

        menuConfigs[configCnt].text = menuItem["text"].as<String>();
        menuConfigs[configCnt].min = menuItem["min"].as<int>();
        menuConfigs[configCnt].max = menuItem["max"].as<int>();
        menuConfigs[configCnt].unitOfMeasurement = menuItem["unitOfMeasurement"].as<String>();
        menuConfigs[configCnt].setEntity = menuItem["setEntity"].as<String>();
        menuConfigs[configCnt].getEntity = menuItem["getEntity"].as<String>();
    }

    configCount = configCnt + 1;

    return true;
}

// Metoda za slanje HTTP GET zahtjeva
String WiFiTask::httpGet(String route)
{
    http.begin(client, String(SERVER_NAME) + route);
    http.addHeader("Authorization", SERVER_TOKEN, false, true);

    int responseCode = http.GET();

    if (responseCode != 200)
        return String();

    String result = http.getString();

    http.end();

    return result;
}

// Metoda za slanje HTTP POST zahtjeva
String WiFiTask::httpPost(String route, String body)
{
    http.begin(client, String(SERVER_NAME) + route);
    http.addHeader("Authorization", SERVER_TOKEN, false, true);
    http.addHeader("Content-Type", "application/json", false, true);

    int responseCode = http.POST(body);

    if (responseCode != 200)
        return String();

    String result = http.getString();

    http.end();

    return result;
}




// Upit u status povezanosti WiFi
WiFiStatus WiFiTask::getStatus()
{
    return status;
}

// Methoda za ažuriranje vrijednosti varijable
void WiFiTask::setValue(int index, int value)
{
    menuConfigs[index].valueState.setValue = value;
    menuConfigs[index].valueState.isChanged = true;
}

// Metoda za dohvaćanje konfiguracije sustava
WiFiMenuConfig WiFiTask::getConfig(int index)
{
    return menuConfigs[index];
}
```

P.3.4. Izvorni kod niti za komunikaciju