Prekidač s varijabilnim hodom i pozicijom aktivacije koristeći elektromagnet

Fletcher, Joshua Lee

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Joshua Lee Fletcher

PREKIDAČ S VARIJABILNIM HODOM I POZICIJOM AKTIVACIJE KORISTEĆI ELEKTROMAGNET

ZAVRŠNI RAD

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Joshua Lee Fletcher

Matični broj: 0016146902

Studij: Informacijski i poslovni sustavi

PREKIDAČ S VARIJABILNIM HODOM I POZICIJOM AKTIVACIJE KORISTEĆI ELEKTROMAGNET

ZAVRŠNI RAD

Mentor :

Doc. dr. sc. Boris Tomaš

Varaždin, rujan 2023.

Joshua Lee Fletcher

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U radu se opisuje implementacija proizvoda prekidača koji omogućava varijabilni hod i točku aktivacije koristeći elektromagnet (u daljnjem kontekstu, "elektromagnetski gumb"). Zbog šuma signala na senzoru implementirana su rješenja za filtriranje signala, njenu noramlizaciju u raspon koji se može lako čitati i koristiti te je omogućena kalibracija u stvarnom vremenu kako bi korisnik mogao preko sučelja na računalu jednostavno mijenjati parametre za točku aktivacije i posljedice aktivacije prekidača. Elektromagnetski gumb je uspoređen sa sličnim proizvodima na tržištu, na njima je objašnjen koncept gumba koji radi s magnetima te na toj osnovi je objašnjen princip rada elektromagnetskog gumba i navedeni su primjeri korištenja ovakvog proizvoda. Konačno, pokrivena je programska podrška u sklopu kojeg je stvoren protokol za komunikaciju s uređajem, program koji emulira server koja šalje odgovore na zahtjeve stvorene sukladno s API-em i lokalna SPIFFS-bazirana baza podataka te sklopovlje proizvoda izrađeno 3D printerom. Naposljetku je napravljena analiza troškova i mogućnosti za budući razvoj.

Ključne riječi: elektromagnet; prekidač; arduino; bluetooth; API

Sadržaj

1. Uvod 1
1.1. Motivacija 1
1.2. Ciljevi 2
1.2.1. Izazov: Pregrijavanje 2
1.2.2. Izazov: Ograničenja sklopovlja 3
1.3. Hipoteze 3
1.4. Tehnologija 4
1.4.1. Programi 4
1.4.2. Biblioteke 4
1.4.2.1. BleKeyboard 4
1.4.2.2. ArduinoJSON 5
1.4.3. Sklopovlje 5
1.4.3.1. ESP32 Dasduino Wrover 6
1.4.3.2. OH49E senzor Hallovog efekta 6
1.4.3.3. Elektromagnet 7
1.4.3.4. OLED monitor 8
1.4.3.5. Kučište 8

2. Pregled tržišta 9
2.1. Slični uređaji na tržištu 9
2.1.1. SteelSeries OmniPoint prekidač 9
2.1.2. 3D-printani prekidači 9
2.1.3. Upravljači za videoigrice 10
2.1.4. Razlike s elektromagnetskim gumbom 11

3. Razvoj uređaja 13
3.1. Arhitektura 13
3.1.1. Sklopovlje 13
3.1.1.1. Mehanizam kliznog zaključavanja 14
3.1.2. Konačni model 16
3.1.3. Programska podrška 18
3.1.3.1. Šum Hall senzora 18
3.1.3.2. Upravljanje elektromagnetom 21
3.1.3.3. Struktura koda i podataka 21
3.1.3.4. Emuliranje servera i API 23
3.1.3.5. Dvosmjerna komunikacija s računalom 26

4. Konačan proizvod 33
4.1. Analiza troškova 33
4.1.1. Troškovi materijala 33
4.1.2. Troškovi programskih alata 34
4.1.3. Troškovi osoblja 34
4.2. Korisničke upute 34

5. Zaključak 36
5.1. Budući rad 36

Popis literature 38

Popis slika 39

1. Uvod

Od raznih vrsta prekidača, oni bazirani na magnetima imaju brojne prednosti. Prekidači bazirani na magnetima reagiraju na prisutnost magnetskog polja te mogu raditi beskontaktno. Kao posljedica toga, ne troše se ni približno onoliko brzo koliko prekidači koji su bazirani na mehaničkim dijelovima. Mogu biti otporni na velike raspone temperature.

Prekidač koristeći elektromagnet će biti ostvaren na sličnom načinu kao prekidač baziran na magnetu, samo što će imati mogućnosti ažuriranja snage elektromagneta i time i magnetskog polja, s ciljem pružanja većeg/manjeg otpora prtiisci tipke (koji u sebi ima ugrađen magnet) te omogućio varijabilni hod (više/manje prostora) obzirom na snagu elektromagneta.

1.1. Motivacija

Prekidač koristeći elektromagnet može omogućiti visoku razinu prilagodljivosti (engl. customizability) za krajnjeg korisnika, pomoću mogućnosti postavljanja i ažuriranja specifičnih parametara:

1. jačina elektromagneta
2. preciznost odgovora senzora (raspon normalizacije signala)
3. posljedica/e pritiska gumba
4. pozicija/e aktivacije gumba

Jačina elektromagneta može varirati u rasponu od 0-100. Preciznost odgovora senzora je odgovor koji senzor vraća nakon što dobiva zahtjev za podacima (engl. reading) u obliku cjelobrojne nenegativne vrijednosti. Npr.:

$$[0-10], [0-100]$$

Posljedice pritiska gumba se mogu isprogramirati za određene domene raspona vrijednosti, npr.:

$$Akcija1 : a$$

kada je odgovor između domene

$$[0, 5]$$

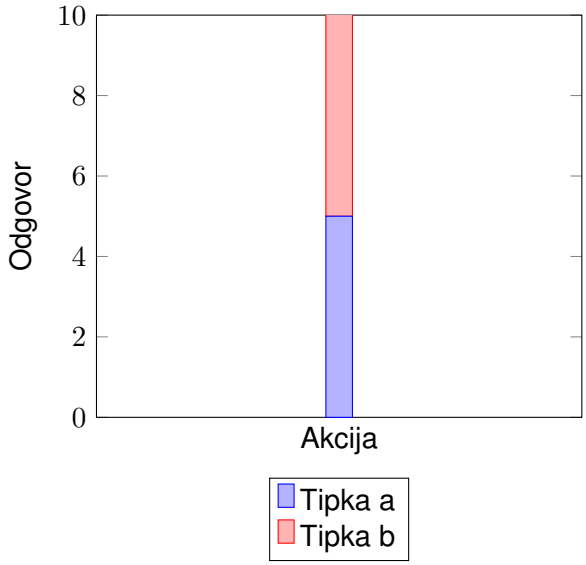
te posljedica

$$Akcija2 : b$$

kada je odgovor u domeni

$$< 5, 10]$$

Na taj način, moguće je izvesti različite zadatke koristeći jedan gumb.



Slika 1: Prikaz odnosa akcija/odgovor [autorski rad]

1.2. Ciljevi

Glavni je cilj stvoriti funkcionalan prototip elektromagnetskog gumba. Veličina nije bitna, ali idealno je da sam gumb ne prelazi veličinu od 10x10x10cm.

Treba izraditi sučelje za komunikaciju s uređajem za pridobivanje informacija o trenutnom statusu uređaja i stanju senzora.

Uređaj treba biti izrađen na način da može potpuno samostalno funkcionirati (što se tiče pohrane podataka i aktivacije; odnosno, da nije potrebna dodatna softverska podrška za funkcioniranje uređaja) te je potrebna mogućnost kalibracije uređaja (preliminarna mjerenja se moraju izvesti kako bi se uređaj mogao automatski prilagoditi okolini u kojoj je aktiviran) i mogućnost ažuriranja vrijednosti na uređaju (ažuriranje specifičnih parametara).

Cilj je implementirati rješenje koje osigurava standardni format poruka (poruka zahtjeva i odgovora) sličan onome za HTTP zahtjeve (implementirati tzv. STP protokol, Serial Transfer Protocol) kako bi se programski kod mogao primijeniti i na drugim projektima u budućnosti.

1.2.1. Izazov: Pregrijavanje

Pretpostavlja se da će najveći izazovi u projektu biti pregrijavanje elektromagneta - što će biti veći parametar za jačinu elektromagneta (što se više bliži prema 100), to će se brže pregrijavati, Pregrijavanje je potencijalno problematično ne samo zbog moguće destrukcije ili istrošavanje komponenti, nego i zbog same funkcionalnosti elektromagneta - magneti izgube svoju magnetičnost na visokim temperaturama. Budući da elektromagneti najbolje funkcioniraju na sobnoj temperaturi, nije optimalno koristiti jačinu od 100% u duljim vremenskim razdobljima.

Ovaj se izazov može djelomično riješiti osiguravanjem dobre ventilacije za dio uređaja koji sadrži elektromagnet i osiguravanjem sredstava za hlađenje uređaja (toplinski odvodnik).

Dodatna rješenja su npr. koristeći tzv. cooldown timer koji će automatski ugasiti uređaj nakon nekog vremena kako bi se spriječilo pregrijavanje ili upotreba dodatnog senzora. Upotrebom senzora za daljinu, pokret ili upotrebom fotorezistora, bilo bi moguće ugasiti elektromagnet dok korisnik nije u poziciji za njeno korištenje. To bi rješenje također smanjilo količinu lažnih pozitivnih rezultata, gdje bi možda senzor inače registrirao pritisak gumba iako korisnik nije ništa učinio.

Moguće je kompletno izbjeći problem pregrijavanja upotrebom supravodljivog materijala, koji se mora rashladiti na vrlo nisku temperaturu kako bi funkcionirao. Međutim, uzimajući u obzir dodatnu opremu i troškove koji bi tada bili potrebni za njegovo funkcioniranje ovo rješenje nije praktično osim ako se ne razvije supravodljiv materijal koji funkcionira pri sobnoj temperaturi [1], [2].

1.2.2. Izazov: Ograničenja sklopovlja

Varijabilni hod će biti limitiran ovisno o specifikacijama komponenti (magnet, elektromagnet i kučište). Ako elektromagnet nije dovoljno snažan, moguće je da će ga magnet nadjačati te da će se magnet spojiti na metal elektromagneta. Također ako elektromagnet nije dovoljno snažan, prostor za varijabilan hod će biti značajno ograničen.

Slabi magnet može "lebdjeti" kao što je potrebno za oponašanje tipke gumba koji je podržan oprugom, ali magnet koji je nedovoljno snažan također može uzrokovati probleme što se tiče manjka otpora pri pritisku tipke. Snažan elektromagnet ničemu ne služi ako je magnet koji se koristi nedovoljno snažan, jer količina otpora koju ona može pružiti je ograničena ovisno o snazi magneta koja joj se suprotstavlja. Napraviti snažniji magnet se može prilično jednostavno spajanjem magneta, no time se onda mora ponovno izraditi kučište za različite veličine magneta. Također postoji granica održivosti ovog rješenja jer ako su magneti sami po sebi nedovoljno snažni, onda će njihova težina prevagnuti nad snagom zajedničkog magnetskog polja koje stvaraju.

Naravno, samo kučište uređaja najviše utječe na ograničenja varijabilnog hoda te treba pažljivo izmjeriti i odrediti mjerenja za kučište kako bi se mogao najbolje iskoristiti sav prostor koji je na raspolaganju (prostor na raspolaganju je određen mjerenjem najmanje daljine između elektromagneta i magneta iznad njega, gdje elektromagnet počinje pružati otpor magnetu dovoljno velik da ono lebdi unatoč svojoj masi).

1.3. Hipoteze

Približavanjem istih polova dvaju magneta počinje se osjetiti otpor. Taj otpor se može iskoristiti da se postigne fenomen magnetske levitacije. Isti se fenomen dogodi kada je jedan od tih magneta elektromagnet, te se to može iskoristiti da se ostvari varijabilnost u daljini levitacije ili, ako se smanji prostor za varijabilni hod, varijabilnost u osjećaju otpora na pritisak tipke.

Međutim, za registraciju "događaja" pritiska tipke, treba se ta vrijednost izmjeriti. Udaljenost između dvaju magneta se može izmjeriti na razne načine, ali je najjednostavnije izmjeriti

stupanj promjene magnetskog polja prilikom približavanja magneta koji će služiti kao tipka. Tu je vrijednost moguće izmjeriti koristeći senzor Hallovog efekta koju je moguće normalizirati na određeni raspon prema kojemu se mogu izvesti različite akcije, ostvarujući funkcionalnost elektromagnetskog gumba.

1.4. Tehnologija

Kučište koje se izrađuje služit će kao prototip te je potrebno izraditi model koji će se moći lako nadograditi, s komponentima koji se mogu lako rastaviti i zamijeniti, umjesto da se mora zamijeniti cjelokupno kučište za svaku iteraciju.

Takav modularni dizajn će omogućiti i lakšu promjenu i prilagodbu dizajna u budućnosti i na taj način dati povećani stupanj prilagodljivosti uređaja za buduće korisnike i posebice za "makere" (hrv. tvoritelji, ljudi koji pokušavaju sami izrađivati vlastite proizvode i projekte).

1.4.1. Programi

Tehnologije koje će se koristiti za realizaciju programa samog uređaja su programski jezik C++ za programiranje samog uređaja (uključujući razne biblioteke), PlatformIO ekstenzija za Visual Studio Code IDE, 3D printer i Fusion360 za izradu modela za kučište.

Za realizaciju računalne aplikacije za promjenu postavki uređaja koristit će se Unity3D engine s jezikom C#. Računalna aplikacija će demonstrirati korištenje i stvaranje naredbi (API zahtjeva), parsiranje odgovora na API zahtjeve te će nuditi sučelje preko kojeg se mogu generirati naredbe i kopirati za vlastite svrhe.

1.4.2. Biblioteke

Glavne biblioteke koje će se koristiti za uspješno ostvarivanje ciljeva projekta su BleKeyboard za ostvarivanje pritisaka tipki na računalu i ArduinoJSON za slanje zahjteva i primanje odgovora, za C++.

Za računalni program najvažnija je biblioteka SerialPortStream koju je napisao Jason Curl. SerialPortStream biblioteka za C# omogućava pregled i pretraživanje portova, pregled pristiglih poruka preko specifičnog porta te slanje vlastitih zahtjeva uređaju kako bi se ostvarila dvosmjerna komunikacija između računala i uređaja.

1.4.2.1. BleKeyboard

BleKeyboard je potrebno koristiti zbog ograničenja sklopovlja. Naime, ESP32 Dasduino Wrover uređaj ne podržava serial (USB) baziranu komunikaciju za uzrokovanje efekta (izvršavanje akcije) pritiska tipke na računalu. Inače bi se za to koristila biblioteka Keyboard, ali ona nema podršku za ESP32 uređaje.

Moguće je napisati program za računalo koja će osluškivati promjene na portu te na osnovi toga uzrokovati pritisak tipki na računalu, ali se tim pristupom ne ispunjava cilj samostalnog rada uređaja i s aspekta softverskog pogleda nije efikasno. Taj pristup također otežava opće korištenje uređaja za prosječnog korisnika, jer obvezuje instalaciju desktop softvera za njeno funkcioniranje.

1.4.2.2. ArduinoJSON

ArduinoJSON biblioteka je korištena za ostvarivanje STP protokola time da se na serial portu čeka "zahtjev" (bilo kakva povratna informacija ili poruka) od računala.

Pri dobivanju zahtjeva, ono se obrađuje prema sljedećim koracima:

1. Provjera da zahtjev ima STP1.0 na početku stringa (primjer pravilno strukturiranog STP zahtjeva: 'STP1.0{"route":"/","method":"GET"}', metoda može biti napisana malim ili velikim slovima)
2. Stvori se substring gdje je početni položaj novog stringa od indeksa prve vitičaste zagrade "{"
3. Rezultirajući se substring parsira u JSON objekt
4. Prema JSON objektu se odredi metoda (GET, POST, PUT, DELETE) i ruta (npr. "/", "/device/status/", "/device/calibrate/")
5. Prema ruti i metodi obavlja se određena akcija
6. Uređaj vraća odgovor s odgovarajućim statusom (status je obilježen kodovima sličnima HTTP protokolu)

Ovime je moguće na organiziran, standardiziran i strukturiran način omogućiti nove funkcionalnosti i jednostavnu komunikaciju s elektromagnetskim gumbom preko računala. Koristeći ArduinoJSON moguće je lako parsirati zahtjeve i ispisati strukturirane odgovore koje će sučelje na računalu znati interpretirati zahvaljujući prepoznatljivom JSON formatu poruka.

1.4.3. Sklopovlje

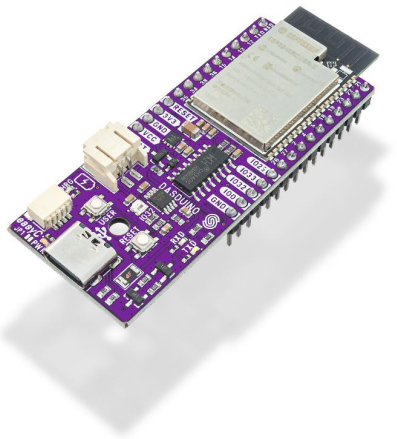
Sastavni komponenti uređaja su:

- ESP32 Dasduino Wrover
- OH49E senzor Hallovog efekta
- elektromagnet (250N)
- OLED display
- breadboard

- kučište
- magnet
- žice
- otpornici
- MOSFET tranzistor

1.4.3.1. ESP32 Dasduino Wrover

Arduino uređaj koji ima ugrađene mogućnosti za Bluetooth komunikaciju te pohranu ograničene količine podataka (između ostalog), za koju je moguće koristiti razne biblioteke za relativno jednostavno programiranje složenih interakcija komponenti, senzora i aktuatora. Ti su faktori osnova za ostvarivanje svih drugih funkcionalnosti elektromagnetskog gumba.



Slika 2: ESP32 Dasduino Wrover [3]

1.4.3.2. OH49E senzor Hallovog efekta

Jeftini senzor Hallovog efekta (u daljnjem kontekstu: Hall senzor). Koristeći "analogRead" funkciju obično je moguće očitati vrijednosti sa senzora u domeni

$$x \in \mathbb{N} \cap [0, 4096]$$

što se može normalizirati na raspon vrijednosti koristeći funkciju map

```
1     (int) (map(reading, DEFAULT_VALUE, 4096, DEFAULT_VALUE, max_normalized)
          );
```

Isječak koda 1.1: Normalizacija vrijednosti senzora pomoću map funkcije

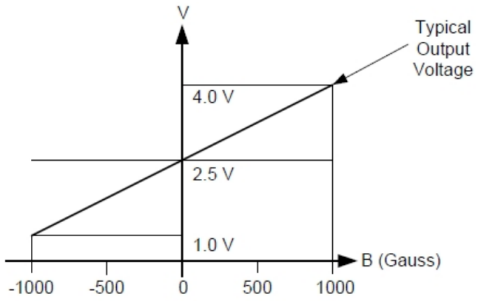
ili koristeći formulu

```
1     (int) ((reading - DEFAULT_VALUE) * max_normalized / (4096 -
          DEFAULT_VALUE));
```

Isječak koda 1.2: Normalizacija vrijednosti senzora pomoću formule

gdje je "reading" vrijednost očitan sa senzora pomoću "analogRead" funkcije, a "max_normalized" granica za normalizaciju vrijednosti te "DEFAULT_VALUE" početna vrijednost senzora dok je uređaj u stanju mirovanja (dok gumb nije pritisnut). Promjenom vrijednosti varijable "max_normalized" moguće je dobiti preciznija ili manje precizna normalizirana mjerenja.

Međutim, Hall senzor u svojim specifikacijama opisuje da je minimalna vrijednost koju može vratiti 1024, a ne 0. To je prikazano na grafu.



Slika 3: Raspon vrijednosti Hall senzora [4]

Preračunavajući ove brojeve u vrijednosti najbliže potenciji broja 2, možemo odrediti da kada nema magnetskog polja, senzor vraća vrijednost od 2,560. Kako se senzor približava južnom polu magneta, vrijednost na senzoru raste do maksimalne vrijednosti od 4096, a ako detektira sjeverni pol, vrijednost se smanjuje do minimalne vrijednosti od 1024. [4]

Obzirom da se u projektu koristi elektromagnet koji će se nalaziti neposredno ispod Hall senzora, treba postaviti zadane početne vrijednosti mjerenja pri pokretanju/ažuriranju uređaja kako bi se temelji ostalih očitanih vrijednosti bili valjani (potrebno je imati mogućnost kalibracije uređaja). Onda se mjeri razlika (stupanj promjene magnetskog polja obzirom na početnu vrijednost) i na osnovi toga izvode akcije koje je korisnik odredio.

1.4.3.3. Elektromagnet

U sklopu projekta koristi se industrijski elektromagnet koji ima nosivost od 250 N (25 kg) zbog problema s nedovoljno snažnim "KS0320 Keyestudio Electromagnet" modulom. Međutim, ovaj industrijski elektromagnet zahtijeva napajanje od 12 V.

Naime, Keyestudiov modul je nedovoljno snažan da bi podržao težinu magneta. Samo magnetsko polje magneta je dovoljno jako da pridržava magnet na elektromagnetu, unatoč odbijanju koje uzrokuju isti polariteti elektromagneta i magneta.

Rješavanje izazova pregrijavanja elektromagneta će stoga biti složenije, jer ako elektro-

magnet nema dovoljno snage, neće pružiti dovoljan otpor magnetima da uređaj funkcionira kao prekidač, što znači da će elektromagnet morati konstantno primati visoku jačinu napajanja da bi uređaj uopće funkcionirao. Posljedica toga je brzo pregrijavanje ako elektromagnet radi dulje vremensko razdoblje.

1.4.3.4. OLED monitor

Sve interakcije i povratne informacije su preko porta kojim je uređaj spojen na računalo za napajanje. Međutim, posljedica toga je da ako korisnik nije spojen nekim programom na "serial monitor" za taj port onda korisnik ne može znati informacije o trenutnom statusu uređaja kao što su to:

- aktiviranost uređaja
- status Bluetooth veze
- trenutno stanje senzora
- trenutno aktivna akcija i sl.

Zbog toga se upotrebljava OLED monitor kako bi se prikazivale te vrijednosti korisniku u stvarnom vremenu.

1.4.3.5. Kučište

Kučište podrazumijeva 3D-printani model koji će sadržavati sve glavne komponente elektromagnetskog gumba (elektromagnet, Hall senzor, magnet) te svojom strukturom od njih tvoriti funkcionalni gumb.

2. Pregled tržišta

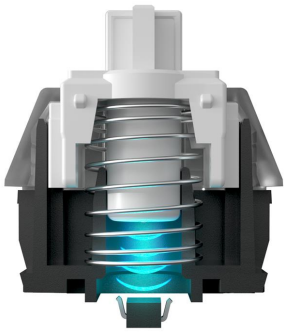
Na tržištu već postoje uređaji koji korste kombinaciju magneta i senzora Hallovog efekta kako bi mogli brzo i precizno otkriti događaje promatrajući promjene u magnetskom polju.

2.1. Slični uređaji na tržištu

Ima nekoliko uređaja sličnih konceptu elektromagnetskog gumba na tržištu, ali nijedan od njih nema mogućnost promjene postavke varijabilnog hoda tipke.

2.1.1. SteelSeries OmniPoint prekidač

SteelSeriesov OmniPoint 2.0 prekidač koristi magnet, Hall senzor i oprugu.



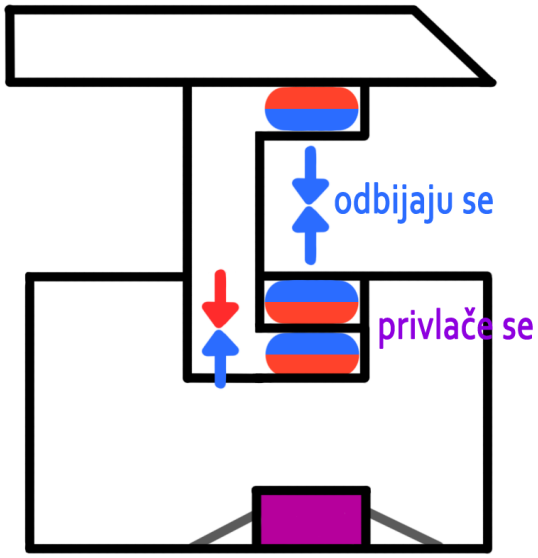
Slika 4: OmniPoint 2.0 prekidač [5]

Senzor je ispod tipke i reagira na promjene u magnetskom polju koje uzrokuje pritisak tipke u kojoj se nalazi sitan magnet. Ovo omogućuje varijabilnu točku aktivacije uz pomoć visoke preciznosti koju nudi senzor, ali ne i varijabilni hod tipke.

Međutim, OmniPoint 2.0 tipke ne koriste princip magnetske levitacije kako bi ostvarili funkcionalnost gumba, nego oprugu.

2.1.2. 3D-printani prekidači

Postoje sheme i modeli za tzv. "DIY" (engl. Do it Yourself, hrv. izradi sam) projekte za one koji žele jeftino izraditi vlastite tipke koje rade na principu magnetske levitacije. Jedan od njih zahtijeva 3D-printanje vlastite tipke te uklapa tri sitna magneta u svoj dizajn. Jedan je u tzv. kapici tipke (engl. keycap), drugi je u bazi tipke, a treći je na dnu šipke koja se miče pritiskom na tipku. Sve to skupa sa senzorom Hallovog efekta na dnu tvori relativno jeftinu tipku koju može svatko samostalno izraditi. [6]



Slika 5: Maglev prekidač [autorski rad]

Ovaj prekidač radi na bazi magnetske levitacije, ali obzirom da se radi o permanentnim magnetima (magneti kojima se ne mijenja snaga) onda nema varijabilnosti u hodu. U najboljem slučaju može samo imati istu primjenu kao SteelSeriesov OmniPoint 2.0 prekidač.

2.1.3. Upravljači za videoigrice

Hall senzori su već našle svoje mjesto na tržištu upravljača za videoigrice.

Ostarjeli analog upravljači počinju reagirati na nepostojeće unose, odnosno počinju samim sobom upravljati. To je zbog istrošenosti upravljača. Međutim, Hall senzori ne zahtijevaju kontakt kako bi funkcionirali te se ne istroše i mogu puno dulje te pouzdanije funkcionirati.

Neki primjeri upravljača koji koriste Hall senzore (tzv. Hall efekt upravljači) su [7]:

- GuliKit KingKong
- NYXI Switch Controller
- SteelSeries Stratus+ Wireless [8]

Ovi upravljači su konkurentni na tržištu zbog preciznosti koju nudi Hall senzor te neistrošivosti. Hall senzori se primjenjuju gdje je bitna varijabilnost u snazi unosa, npr. u volanima za igre utrke.

2.1.4. Razlike s elektromagnetskim gumbom

Elektromagnetski gumb se razlikuje od svih ovih proizvoda zbog varijabilnosti jačine elektromagneta, odnosno zbog prilagodljivosti uređaja potrebama korisnika. Ono nudi sve koristi koje nude postojeći uređaji na tržištu i više.

Primjerice, niz elektromagnetskih gumbova je moguće upotrijebiti kao tipkovnicu (u daljnjem kontekstu "elektromagnetska tipkovnica"). Preko sučelja za interakcije s elektromagnetskim gumbovima na individualnoj razini ili kao cjelinu, moguće je onda prilagoditi visinu određenih tipki ili kompletno isključiti određene tipke s visokom razinom preciznošću. To će omogućiti da npr. igrači videoigrica ne pritisnu često krive gumbove (engl. fat-finger), omogućiti korisnicima da smanje ili povećaju otpor na njihov pritisak za prilagodbu taktilnosti tipki i sl.

Uzmemo li primjer tipične "gamer" tipkovnice ovih dana ta tipkovnica će vrlo vjerojatno svjetliti i imati razne animacije. Za korisnike, elektromagnetska tipkovnica nudi mogućnost stvaranja tih animacija na fizičkoj razini. Naravno, to bi onemogućilo samu upotrebu tipkovnice te bi se ta mogućnost najvjerojatnije nudila u obliku koji nalikuje čuvaru zaslona.

Za programere elektromagnetska tipkovnica nudi jednostavnu interakciju sa stvarnim svijetom, obzirom da je ona u suštini ulazno-izlazni tip uređaja. Moguće je praviti jednostavne igrice, npr.:

- "Whack-a-mole!" gdje korisnik treba pritisnuti gumbove koji se nasumično ispupčaju na tipkovnici i na taj način zaradio bodove
- "Snake" gdje korisnik treba strijelicama preusmjeravati zmiju da pojede "voće" u obliku ispupčanih tipki na tipkovnici
- igrice za treniranje u daktilografiji, gdje korisnik uči pravilno tipkati, time da se fizički spuštaju sve tipke osim one koju treba trenutno pritisnuti, stvarajući motoriku i naviku korisnika da pritisne samo one potrebne tipke pravilnim prstima za to

Postojeće igrice mogu koristiti sučelje s elektromagnetskom tipkovnicom da fizički izražavaju vrijednosti igraču, npr.:

- umor lika koju igrač igra se može izražavati tako da se pojačava otpor za pritisak tipke što je lik umorniji, ili na način da se smanji varijabilni hod te na taj način igraču ograniči brzina trčanja u igrici
- vrijednosti kao što su životni bodovi lika u igrici se mogu fizički izraziti na tipkovnici (npr. brojevima 1-0 za 1-10) kako bi se smanjila zatrpanost zaslona takvim informacijama i povećao osjećaj imerzivnosti

Osim ovih malo neozbiljnih koncepata, ono može naći svoju primjenu kao prekidač visoke razine detalja gdje je to potrebno:

- za ažuriranje veličine kista u digitalnim alatima za umjetnost

- za pružanje detaljne alternacije dinamike u digitalnim alatima za skladanje glazbe ili uređivanje audiosnimki (npr. za emuliranje glazbene tehnike "vibrato")

3. Razvoj uređaja

Rad na uređaju trajao je otprilike dva mjeseca. To uključuje vrijeme utrošeno na specificiranje zahtjeva i ciljeva projekta, istraživanje sličnih projekata, istraživanje potrebnih materijala, osmišljavanje rješenja, prototipiranje, programiranje i usavršavanje rješenja.

3.1. Arhitektura

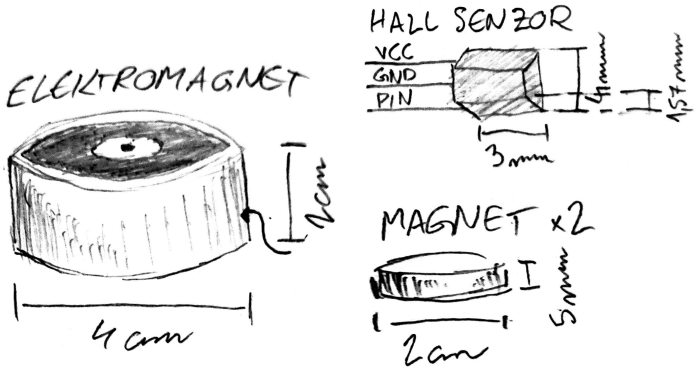
Arhitekturu uređaja moguće je podijeliti na dva dijela:

- sklopovolje
- programska podrška

U sklopu poglavlja o programskoj podršci, pokriven je i razvoj softvera za računalo koje može ostvariti dvosmjernu komunikaciju s uređajem za jednostavan pregled njegovog statusa i jednostavno ažuriranje podataka na uređaju.

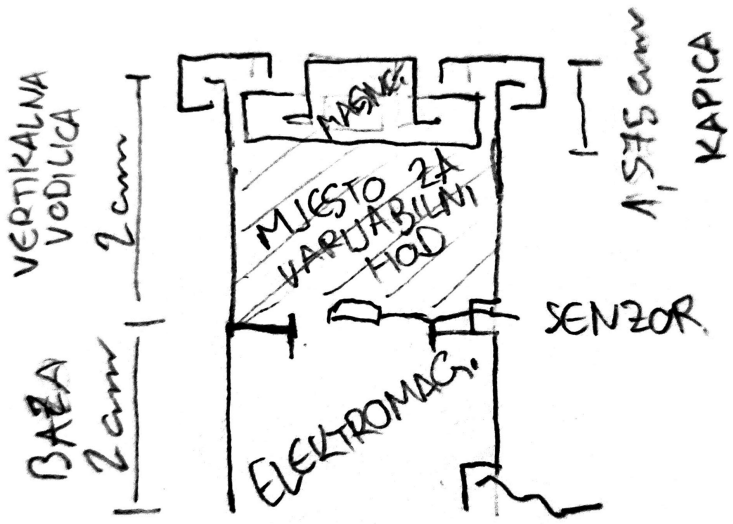
3.1.1. Sklopovlje

Sklopovlje je izrađeno kroz iterativni proces (prototipiranje). Prvi korak bio je izraditi detaljnu skicu koja bi sadržavala sve potrebne informacije za planirani 3D model pa su sve dimenzije glavnih komponenti koje će biti uključene u kućište uređaja (elektromagnet, magnet i Hall senzor) bile izmjerene.



Slika 6: Skica veličina komponenti [autorski rad]

Prema veličinama komponenti određene su dimenzije modela koja će sadržati bazu s elektromagnetom i mali otvor kroz koji će se progurati Hall senzor tako da se pozicionira točno iznad sredine elektromagneta. Prostor za varijabilni hod je bio izmjeren priključivanjem elektromagneta u struju te držanjem magneta iznad elektromagneta dok se nije osjetio dovoljno snažan otpor.

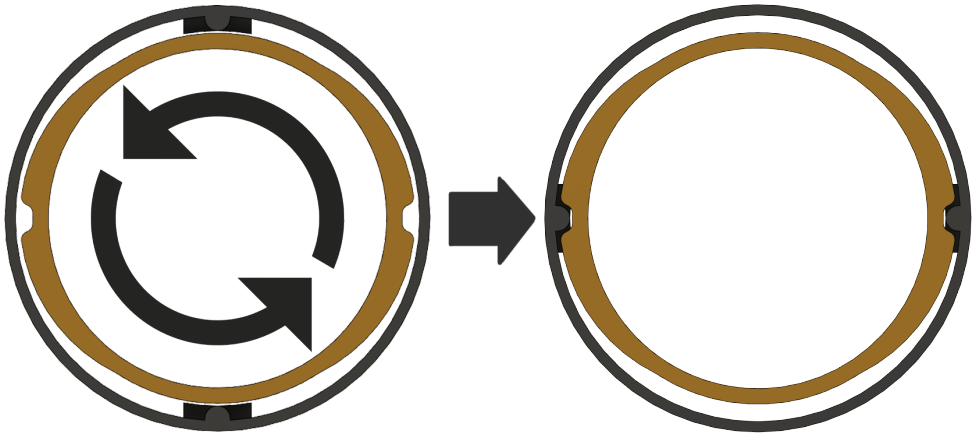


Slika 7: Preliminarna skica modela elektromagnetskog gumba [autorski rad]

Dizajn koji je osmišljen je modularan - omogućit će dekomponiranje modela na dijelove za jednostavno sastavljanje i nadogradnju tijekom faze prototipiranja. Također je dizajniran s nakanom da se svi dijelovi mogu ponovno dohvatiti bilo kada, bez oštećenja uređaja.

Kao što se vidi iz skice na slici 7, plan obuhvaća dizajn "košare" s "kapicom" koja će sadržavati magnet, dizajn "prstena" koji će biti držač koji ograničava vertikalno kretanje magneta te baze koja iznad sebe ima rupu kako bi omogućilo maksimalno širenje magnetskog polja koju generira elektromagnet. Na sredini između elektromagneta i košarice s magnetom se nalazi pregrada koja će sprječavati direktan kontakt sa Hall senzorom.

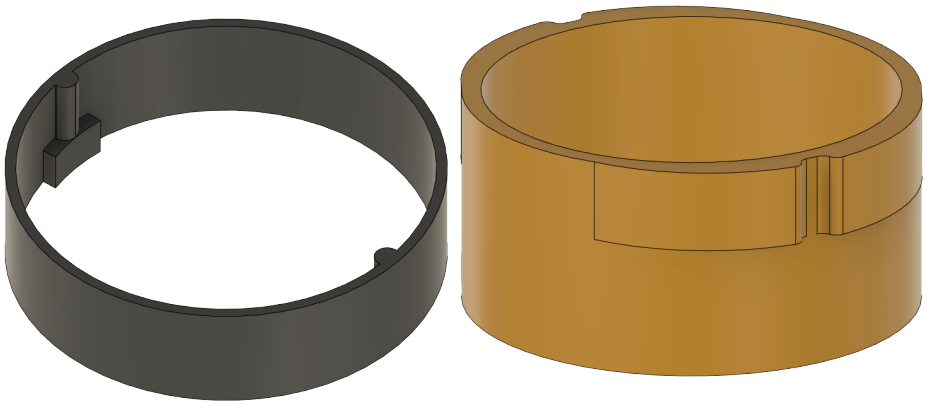
3.1.1.1. Mehanizam kliznog zaključavanja



Slika 8: Pogled odozgo na rotirajući mehanizam kliznog zaključavanja [autorski rad]

Za ostvariti modularnost, sastavni dijelovi kučišta su dizajnirani kako bi se mogli rotirati na mjesto. Čim se rotiraju na odgovarajuće mjesto, one se "zaključavaju" i bez većeg napora nije ih moguće razdvojiti. Ovakav dizajn je moguć zahvaljujući elastičnosti tanke plastike. Kako

bi se ostvario, potrebno je dizajnirati "rampe" po kojima će "šipke" kliziti dok ne dođu u odgovarajuću poziciju (rupu).



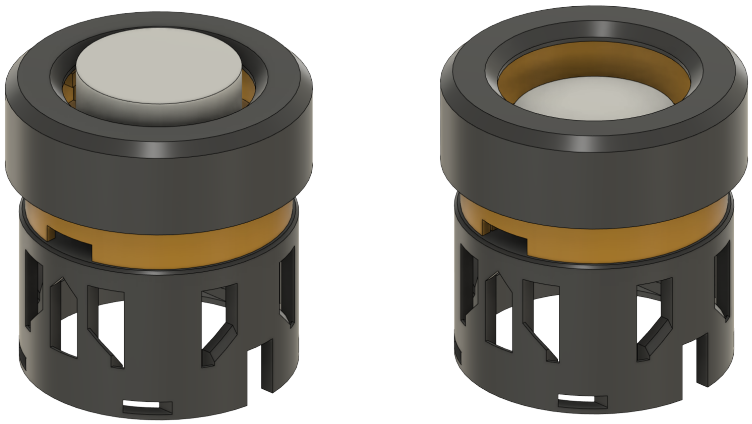
Slika 9: Primjer dijelova za rotirajući mehanizam kliznog zaključavanja [autorski rad]

Za realizaciju komponenti koje koriste mehanizam kliznog zaključavanja potrebno je implementirati sljedeće:

- klizni dio koji ima:
    1. šipku koja ograničava horizontalno kretanje čim je zaključano
    2. blokadu na dnu koja onemogućava vertikalno kretanje dijela
- statični dio koji ima:
    1. površinu s blagim nagibom po kojemu će kliziti šipka
    2. rupu u koju će upasti šipka

Pritom šipka mora biti minimalno iste visine kao i rupa u statičnom dijelu. Za svrhe ovog projekta je korišten rotirajući klizni mehanizam zaključavanja, ali klizni mehanizam zaključavanja ne mora nužno biti ostvaren kao rotirajući model.

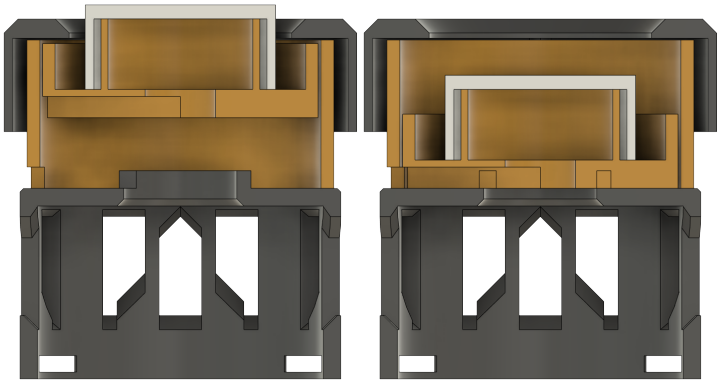
3.1.2. Konačni model



Slika 10: Prikaz gumba dok nije pritisnut i dok je pritisnut [autorski rad]

Konačni trodiobni model tipke se sastoji od:

- kapice tipke
- vertikalne vodilice
- baze tipke

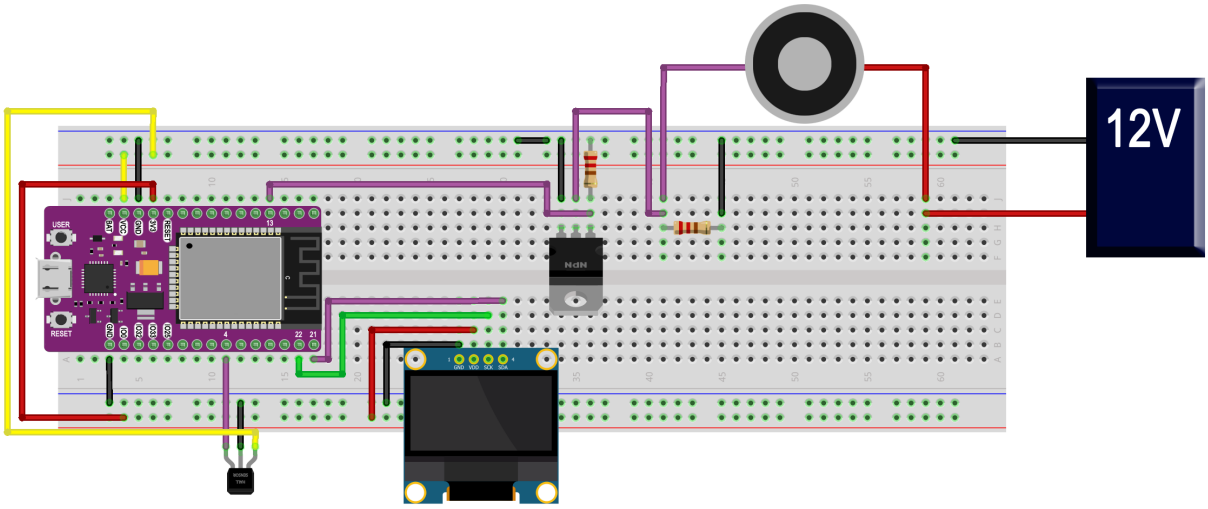


Slika 11: Vertikalni presjek gumba dok nije i dok je pritisnut [autorski rad]

Kapica tipke sadrži magnete koje će lebdjeti nad elektromagnetom. To je dio tipke koju će korisnik pritisnuti. Ovdje je modularan dizajn od posebne važnosti jer osoba koja sastavlja uređaj može ju slučajno sastaviti s magnetima okrenutim s krivim polom prema bazi gumba.

Vertikalna vodilica služi za fiksiranje kapice tipke na mjesto, omogućavajući joj slobodno vertikalno kretanje i sprječavajući horizontalno kretanje. Vertikalna vodilica također čvrsto pridržava Hall senzor na svome mjestu odmah iznad elektromagneta u samoj sredini modela.

Baza tipke sadrži elektromagnet i ima rešetke za brže rashlađenje elektromagneta nakon što se ono zagrije.



Slika 12: Shema povezivanja za elektromagnetski gumb [autorski rad]

Uređaj koristi MOSFET kako bi upravljao jačinom elektromagneta pomoću "analogWrite" funkcije u kodu, ali elektromagnet svoje napajanje ne dobiva od samog uređaja zato što je njeno napajanje previše zahtjevno za ESP32. Elektromagnet dobiva svoje napajanje od adaptera od 12V.

Slika 13: Slika konačnog modela elektromagnetskog gumba [autorski rad]

3.1.3. Programska podrška

Pisanje programskog koda je preuzelo približno 63% vremena od sveukupnog trajanja razvojnog procesa. Programski isječci koji će se prikazati su preuzeti sa službenog repozitorija projekta.

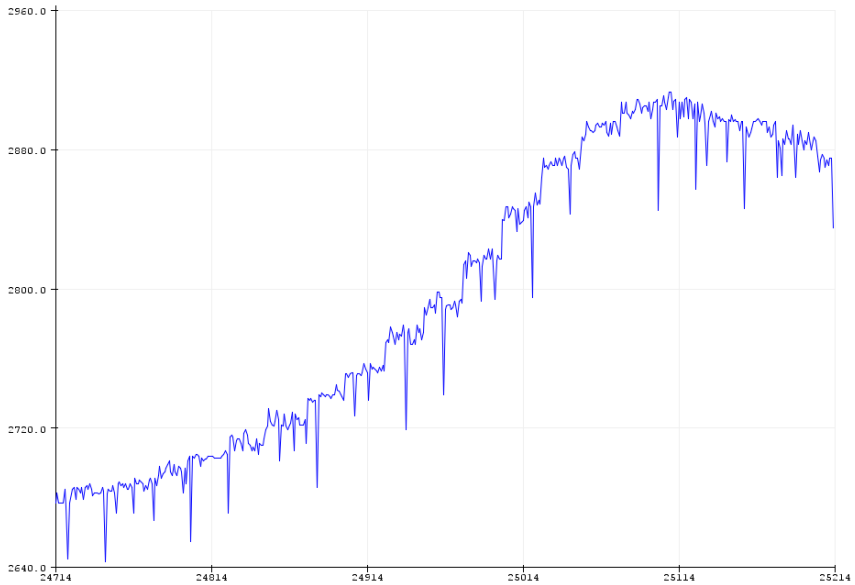
Potreban softver treba riješiti niz problema i izazova:

1. problem šuma Hall senzora
2. problem ažuriranja snage elektromagneta
3. problem strukture koda i podataka
4. problem perzistencije podataka (pohrana postavki)
5. problem ažuriranja postavki i programibilnsti
6. izazov standardizirane dvosmjerne komunikacije

Najveći problemi tijekom razvoja su predstavili bugovi u PlatformIO ekstenziji za Visual Studio Code te programiranje lokalnog sustava za upravljanje bazom podataka, ali je prvi izazov bio filtriranje vrijednosti Hall senzora.

3.1.3.1. Šum Hall senzora

Koristeći "analogRead" funkciju može se uočiti značajna varijacija u vrijednostima koje vraća Hall senzor. Razlika u svakom trenu može varirati u rasponu od ± 200 jedinica od centralne vrijednosti. Zbog toga je važno implementirati programski filter koji će uzimati prosjek od više mjerenja i prikazivati razliku samo dok je ona dovoljno blaga, a ne ako značajno varira od prošle.



Slika 14: Približavanje magneta senzoru bez filtera [autorski rad]

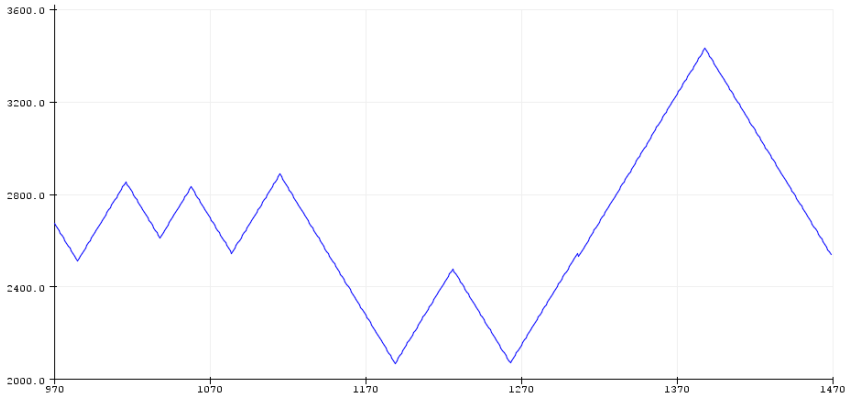
Programski kod koji ispisuje vrijednost dobivenog sa Hall senzora u "loop" funkciji dobro prikazuje šum na Hall senzoru.

```
#define HALL_PIN 4
void loop() {
    Serial.println(analogRead(HALL_PIN));
}
```

Isječak koda 3.1: Čitanje vrijednosti sa senzora bez filtera

Za elektromagnetski gumb koji ovisi o preciznim mjerenjima to može predstaviti veliki problem te je potrebno regulirati tu vrijednost. Normalizacija vrijednosti koju vraća senzor je jedan od načina da se ono regulira, jer će promjena od 200, uzimajući u obzir da su na raspolaganju vrijednosti od 2560-4096, biti manje značajna kada se zaokružuje na broj koji je npr. između 1-10. Međutim, to samo daje prividan osjećaj kao da je problem riješen. Zato je potreban algoritam koji će pravilno filtrirati šum i od nje razabrati informaciju te na taj način dati čitanja koja su "čista".

Potrebno je, dakle, implementirati programski kod takav da će uzeti neku početnu vrijednost (u ovom primjeru 2560, ali kasnije će to biti vrijednost koja je određena pri kalibraciji senzora) te uzeti prozor vrijednosti oko nje. Tada treba izračunati prosjek svih vrijednosti u tom prozoru i usporediti apsolutnu vrijednost rezultata razlike između trenutne vrijednosti na senzoru i prosjeka vrijednosti u prozoru. Ako je razlika prevelika (veća od vrijednosti pohranjene u makrou "THRESHOLD"), može se pretpostaviti da se radi o šumu i ignorirati najnoviju vrijednost, ali s druge strane, ako je razlika unutar dopuštene domene vrijednosti (razlika je ili za 10 veća ili manja od prethodno pohranjene vrijednosti) onda se radi o novom podatku koju je senzor primio iz okolice te ju možemo pohraniti i koristiti.



Slika 15: Približavanje i udaljavanje magneta senzoru s filterom [autorski rad]

Alatom "serial plotter" se može uočiti graf sa čistim vrijednostima uz kod za "denoise" (hrv. ukloni šum) funkciju.

```
#define THRESHOLD 20
```

```
#define WINDOW_SIZE 40

void denoise(int HALL_PIN) {

    int values[WINDOW_SIZE];
    int sum = 0;
    int index = 0;

    int past = 2560;

    for (int i = 0; i < WINDOW_SIZE; i++) {
        values[i] = 2560 + i;
        sum += values[i];
    }

    while (true) {
        int value = analogRead(HALL_PIN);

        sum -= values[index];
        sum += value;
        values[index] = value;
        index = (index + 1) % WINDOW_SIZE;
        int average = sum / WINDOW_SIZE;

        int diff = abs(value - average);
        if (diff > THRESHOLD) {
            // the value is noisy, skip it
            continue;
        }

        // the value is not noisy, print it
        if(average > past + 10 || average < past - 10) {
            Serial.println(average);
            past = average;
        }
    }
}
```

Isječak koda 3.2: Čitanje vrijednosti sa senzora s filterom

Vrijednost dobivena "denoise" funkcijom se može dodatno normalizirati kao što je to prethodno specificirano. Normalizacijom je moguće odrediti udaljenost magneta od samog senzora. Obzirom na izračunatu udaljenost moguće je onda i odrediti kada je gumb "pritisnut" i izazvati slijed događaja na uređaju koji će izvesti specifičnu akciju na računalu koji je povezan s uređajem preko Bluetootha.

3.1.3.2. Upravljanje elektromagnetom

Elektromagnetom se može upravljati pomoću tranzistora (u ovom slučaju je upotrijebljen MOSFET tranzistor) i "analogWrite" funkcije koja prima dva argumenta.

Prvi argument te funkcije predstavlja pin na kojemu treba "pisati" vrijednost, a drugi argument je brojčana vrijednost u domeni od $x \in \mathbb{Z} \cap [0, 255]$ (što znači da postoje 2^8 različith vrijednosti na raspolaganju).

Međutim, zahvaljujući magnetima koji moraju lebdjeti iznad elektromagneta, vrijednosti koje su upotrebljive za svrhe ovog projekta su samo one u domeni $x \in \mathbb{Z} \cap [245, 255]$.

To je posljedica snage korištenih neodimijskih magneta. Zbog velike snage tih magneta, sve vrijednosti u domeni $x \in \mathbb{Z} \cap [0, 244]$ imaju istu efektivnost kao vrijednost 0. To je zato što je elektromagnet izrađen od željeza, što privlači magnete, te kad magneti nadjačaju elektromagnet one se prilijepe za njenu površinu i narušen je efekt magnetske levitacije. Formula za postavljanje jačine elektromagneta je onda:

$$electromagnet\_power \in \mathbb{R} \cap [0, 1]$$

$$y = electromagnet\_power * 10 + 245$$

U programskom kodu, ako u varijabli "electromagnet_power" pohranimo vrijednost kao postotak između 0 i 100 koji predstavlja jačinu koju želimo dodijeliti elektromagnetu, onda će programski kod za dodjelu jačine elektromagnetu od 75% izgledati ovako:

```
1 #define ELECTROMAGNET_PIN 13
2 double electromagnet_power = 0.75;
3 void loop() {
4       analogWrite(ELECTROMAGNET_PIN, (int) (electromagnet_power * 10) +
            245);
5 }
```

Isječak koda 3.3: Upravljanje elektromagnetom

Budući da formula može rezultirati decimalnom vrijednošću, potrebno ju je pretvoriti u cjelobrojnu vrijednost.

3.1.3.3. Struktura koda i podataka

Kako bi kod bio održiv, koristi se enkapsulacija za sve varijable i funkcije koje ne bi trebale biti javno dostupne. Upotrebom datoteka zaglavlja moguće je djelomično sakriti te podatke, ali obzirom na visoku razinu među-referenciranja između datoteka koda bilo je potrebno svrstati u klase sve što je moguće.

- DisplayManager - prikazuje podatke o trenutnom statusu uređaja u stvarnom vremenu na

OLED monitoru te omogućava njihovu pretvorbu u JSON informacije za objavu statusa uređaja preko porta

- KeyboardLogic - statična klasa koja upravlja statusom Bluetooth veze uređaja i logikom za izvođenje akcija
- HallFilter - filtrira i normalizira vrijednosti Hall senzora te identificira koje je od akcija trenutno aktivno (ako nijedna akcija nije aktivna vraća -1, kao indeks "nepostojeće akcije")
- STP - statična klasa koja služi za objavlivanje poruka o nadolazećim zahtjevima te stvaranje zahtjeva protokola STP1.0
- EmbButtonDB - SPIFFS-bazirani sustav za upravljanje lokalnom bazom podataka sa svim osnovnim operacijama za to; baza podataka je zapravo datoteka koja pohranjuje niz JSON objekata za koju vrijedi pravilo "novi red, novi zapis"
- EmbServer - omogućava dvosmjernu komunikaciju s uređajem emulirajući pravi server, primajući zahtjeve i šaljujući odgovore te obradom zahtjeva prema prethodno specificiranim pravilima u API-u; zaslužno je za kalibraciju uređaja, ažuriranje podataka na uređaju te osvježavanje podataka na OLED monitoru kad je to potrebno
- STPDBConnection - povezuje EmbServer s bazom podataka, omogućavajući komunikaciju s podatkovnim slojem

Podaci koji se čitaju iz baze pri pokretanju uređaja ili pohranjuju tijekom aktivnog rada uređaja su pohranjeni primarno u slogove, nakon čega su dodijeljeni prethodno navedenim klasama po potrebi.

- EmbKeyBlock - sadrži podatke o statusu Bluetooth povezanosti uređaja, koliko je puta gumb pritisnut (ne vodi se posebno za svaku akciju) te vrijeme čekanja prije sljedeće registracije pritiska gumba; EmbKeyBlock je vidljiv samo klasi KeyboardLogic te se podatak o Bluetooth povezanosti pohranjuje i u EmbConnectionState slogu
- EmbConnectionState - sadrži podatak o statusu Bluetooth povezanosti koju dobiva od EmbKeyBlock
- EmbAction - sadrži identifikacijski ključ akcije (ako je ključ -1, akcija se računa kao da je prazna i neće se izvršiti), atribut ključ koji predstavlja vrijednost tipke odnosno akciju koja će se izvesti te decimalni broj između 0 i 1 koji predstavlja poziciju aktivacije
- EmbButton - sadrži identifikacijski ključ gumba (identifikacijski ključ je potreban u slučaju da su povezana više gumbova na računalo ili na uređaj), vrijednosti za pinove Hall senzora i elektromagneta te popis triju EmbAction-a
- Emb - sadrži metapodatke o samom uređaju kao što su njen Bluetooth naziv, proizvođač, verzija softvera uređaja te referencu na status povezanosti EmbConnectionState, podatke prema kojima treba registrirati akcije iz sloga EmbButton i referencu na objekt preko kojeg se registriraju akcije, BleKeyboard

3.1.3.4. Emuliranje servera i API

Kako bi se riješili problemi perzistencije podataka, njihovog ažuriranja te omogućila visoka razina programibilnosti uređaja, implementirana je EmbServer klasa koja emulira ponašanje stvarnog servera.

Glavna prednost u emuliranju servera je njena brzina - sve komunikacije su preko žice te su uvijek iste brzine ukoliko se sam uređaj ne uspori. Također je manje zahtjevno za procesor uređaja i olakšava njeno korištenje krajnjem korisniku.

Server dohvaća i provjerava svaku nadolazeću poruku za oznaku "STP1.0" prije obrade.

```
1  void serialLogic() {
2      if (Serial.available() > 0) {
3          String c = Serial.readString();
4          c.trim();
5          if (c.startsWith("STP1.0")) {
6              String jsonData = c.substring(6);
7              DynamicJsonDocument doc(1024);
8              deserializeJson(doc, jsonData);
9              STP::announceRequest(doc);
10             handleRequest(doc["route"], doc);
11             /* example request (method isn't case sensitive):
12             STP1.0{"route":"/","method":"GET"}
13             */
14         }
15     }
16 }
17
18 void localRequestLogic(String request) {
19     if(request.startsWith("STP1.0")) {
20         String jsonData = request.substring(6);
21         DynamicJsonDocument doc(1024);
22         deserializeJson(doc, request);
23         STP::announceRequest(doc);
24         handleRequest(doc["route"], doc);
25     }
26 }
```

Isječak koda 3.4: Obrada zahtjeva na serveru

Metoda "handleRequest(String route, DynamicJsonDocument json)" tada obrađuje zahtjev, provjeravajući njenu rutu i metodu.

```
1 enum class STPMethod { GET, POST, PUT, DELETE };
```

Isječak koda 3.5: Dostupni tipovi metoda

```
// for simpler access to route description via enum key index
enum Routes {
    ROOT,
    DB,
    ROUTES,
    CALIBRATE,
    ENABLE,
    DISABLE,
    DATA,
    ELECTROMAGNET,
    ELECTROMAGNET_POWER,
    HALLSENSOR,
    HALLSENSOR_NORMALIZED,
};

const String routes[] = {
    "/", // GET
    "/db/", // GET, POST, PUT, DELETE
    "/device/routes/", // GET
    "/device/calibrate/", // GET
    "/device/enable/", // PUT
    "/device/disable/", // PUT
    "/device/data/", // GET
    "/device/electromagnet/", // GET
    "/device/electromagnet/power/", // PUT
    "/device/hallsensor/", // GET
    "/device/hallsensor/normalized/", // GET
}; // example: STP1.0{"route":"/db/","method":"get"}
```

Isječak koda 3.6: Dostupne rute na API-u

Ako postoji funkcija za tu kombinaciju, onda obrada nastavlja dalje, bilo to na prezentacijskom sloju, podatkovnom sloju ili na sloju poslovne logike uređaja. U protivnom će server izbaciti poruku greške. Također u slučaju da tip metode nije validan - nije sadržan u enumeraciji STPMethod - server će izbaciti poruku greške.

U samome kodu, svaka ruta ima svoju funkciju koja se pozove kada su ispunjeni svi uvjeti za tu rutu.

```
void getCalibrateRoute() {
    Serial.println(STP::createResponse(200, "Calibrating the sensor"));
    calibrateFilter();
}
void calibrateFilter() {
    *enableDevice = false;
    delete filter;
    filter = new HallFilter(filter->emb);
```

```
    *enableDevice = true;
}
```

Isječak koda 3.7: Funkcija za rutu "/device/calibrate/" s metodom "GET"

Svaki odgovor ima povratni status i povratnu poruku. Odgovor može sadržavati i dodatne podatke, npr. ruta "/device/hallsensor/" vraća najnoviju očitanu vrijednost sa Hall senzora.

```
void getHallSensorDataRoute() {
    Serial.println(STP::createResponse(200, "Accessed hall sensor data", "
        current_value", String(filter->getDisplayValue())));
}
```

Isječak koda 3.8: Funkcija za rutu "/device/hallsensor/" s metodom "GET"

Podaci se preko API-a spremaju u bazu podataka koja funkcionira na sličan način kao i normalna baza podataka, s metodama za uklanjanje ili ažuriranje specifičnih zapisa, dodavanje novih zapisa, dohvaćanje zapisa ili brisanje svih zapisa.

```
String* getAll() {
    // print all lines in the document (each newline is a new record) (
        records are all json strings)
    String* database = new String[100];

    if(!SPIFFS.exists(this->file)) {
        String response = STP2::createResponse(404, "File does not exist");
        Serial.println(response);
        return database;
    }

    File file = SPIFFS.open(this->file, "r");
    if(!file) {
        String response = STP2::createResponse(500, "Failed to open file
            for reading");
        Serial.println(response);
        return database;
    }

    int i = 0;
    Serial.print("STP1.0{\"status\":200,\"data\":[");
    while(file.available()) {
        database[i++] = file.readStringUntil('\n');
        Serial.print(database[i - 1]);
        Serial.print(",");
    }
```

```
    Serial.println("]}");

    file.close();
    return database;
}
```

Isječak koda 3.9: Ruta "/db/" s metodom "GET" vraća sve zapise

Međutim, važno je pitanje je li to uopće upotrebljivo ili korisno, jer ako uređaj može samostalno raditi zašto bi bio potreban STP1.0 protokol?

3.1.3.5. Dvosmjerna komunikacija s računalom

STP1.0 protokol skupa s EmbServer klasom značajno pojednostavljuje proces stvaranja novih funkcionalnosti za elektromagnetski gumb, ali je još bitnija činjenica da je do njih tada moguće doći strukturiranim, standardiziranim zahtjevima koje se mogu slati s bilo kojeg uređaja ukoliko je ono direktno povezano s elektromagnetskim gumbom. To je prava vrijednost STP1.0 protokola.

Za demonstraciju dvosmjerne komunikacije je odabrana tehnologija Unity. Postoje jednostavnije tehnologije u kojima se može izraditi demonstracija, ali obzirom da je veći broj primjera slučajeva korištenja elektromagnetskog gumba bila na temi videoigrica, demo će biti izrađen u Unityu.

Kako bi se omogućila komunikacija između računala i elektromagnetskog gumba, korištena je biblioteka SerialPortStream.

Pri pokretanju programa pokreće se funkcija "initSerialPort()" koja traži port na koju je povezan uređaj. Ono se pohrani u varijablu te se otvori, a ako ne postoji ili je taj port već otvoren prikazuje odgovarajuću poruku pogreške.

```
string deviceName = "USB-SERIAL CH340";
private int baudRate = 115200;

SerialPort port;
PortDescription findSerialPort() => SerialPortStream.GetPortDescriptions().
    FirstOrDefault(p => p.Description.Contains(deviceName));

void initSerialPort() {
    PortDescription discoveredPort = findSerialPort();
    if (discoveredPort == null) {
        portNotFound();
        return;
    }
    portDataComponent.text = portName(discoveredPort.ToString());
    currentMonitorData["port"] = portDataComponent.text;
    openPort(discoveredPort);
```

```
16 }
17
18 private string _portNotFound = "Port not found. Please plug in the device."
       ;
19 public void openPort(PortDescription discoveredPort) {
20     port = new SerialPort(discoveredPort.Port, baudRate);
21     port.ReadTimeout = 1; // this line of code is crucial
22     attemptToOpenPort();
23 }
24 private void attemptToOpenPort() {
25     try {
26         portDataComponent.color = initialPortDataComponentColor;
27         port.Open();
28     } catch {
29         portDataComponent.text = "ERROR: Port is in use by another process.
               ";
30         monitorDataComponent.text = "ERROR\n\nThe port is currently in use
               by another process.\n"
31             + "Close any other programs and click the settings button to
                   refresh.";
32         portDataComponent.color = Color.red;
33     }
34 }
35
36 public void portNotFound() {
37     portDataComponent.text = _portNotFound;
38     currentMonitorData["port"] = portDataComponent.text;
39 }
```

Isječak koda 3.10: Otkrivanje i inicijalizacija porta

Vrlo je važno da se postavi "ReadTimeout" atribut porta, inače će program vječno čekati nekakvu poruku. U primjeru je postavljeno na maksimalno čekanje od 1s. Valja spomenuti da otvaranje prethodno neotvorenog porta ponovno pokreće elektromagnetski gumb, a ako je port otvoren to će uzrokovati grešku te je zbog toga poziv metode "open" u try bloku.

Poruke se vrlo jednostavno čitaju. U primjeru su sve poruke pohranjene u zapisnik kako bi se kasnije mogle sve pregledati ako je to potrebno. Zapisnik "logs" je pohranjen kao tip podataka "Dictionary<float, string>" i može omogućiti povratak postavki uređaja na starije stanje, pregled povijesti naredbi korisnika i dr.

```
1 private void _log() {
2     try {
3         _latestLogVal = port.ReadLine();
4     } catch {
5         return;
6     }
```

```
7       if (_latestLogVal.Length > 1 && _latestLogVal.StartsWith("STP1.0")) {
8           logs.Add(Time.time, _latestLogVal);
9           print(_latestLogVal);
10      }
11  }
12
13  void Update() {
14      timer -= Time.deltaTime;
15      serialDataManagement();
16  }
17
18  private void serialDataManagement() {
19      if (timer <= 0) {
20          _port(); // manages display feedback for active port
21          if (port != null) _log();
22          timer = timerDuration;
23      }
24  }
```

Isječak koda 3.11: Otkrivanje i spremanje poruka u zapisnik

Pri svakom novom ažuriranju logova, može se onda registrirati događaj pristigle poruke. Budući da će sve poruke biti u JSON obliku, moguće ih je serijalizirati u zasebne klase ovisno o njihovom sadržaju. Klase koje se mogu serijalizirati moraju imati atribut "System.Serializable" te njihovi atributi moraju biti identični nazivima atributa iz poruke.

Potrebno je dodatno napisati i "toDictionary()" metodu koja će pomoći da se vrijednosti klase preoblikuju u JSON format sa svrhom njihovog parsiranja u podatke za zahtjeve koje će se slati elektromagnetskom gumbu.

```
1  using System.Collections.Generic;
2
3  /// <summary>
4  ///  Base response class
5  /// </summary>
6  [System.Serializable]
7  public class STPResponseModel {
8      public int status;
9      public STPDataModel data;
10     public virtual Dictionary<string, object> toDictionary() {
11         Dictionary<string, object> result = new Dictionary<string, object
               >();
12         result.Add("status", status);
13         result.Add("data", data.toDictionary());
14         return result;
15     }
16 }
```

```
17 [System.Serializable]
18 public class STPDataModel {
19     public string message;
20     public virtual Dictionary<string, object> toDictionary() {
21         Dictionary<string, object> result = new Dictionary<string, object
               >();
22         result.Add("message", message);
23         return result;
24     }
25 }
```

Isječak koda 3.12: Osnovna klasa za sve pristigle STP poruke

Klase STPResponseModel i STPDataModel čine osnovu za sve ostale tipove odgovora i svaki odgovor mora njih naslijediti ili mora bar imati njihovu osnovnu strukturu.

```
1  [System.Serializable]
2  public class KeyPressResponseModel : STPResponseModel {
3      public new KeyPressDataModel data;
4      public override Dictionary<string, object> toDictionary() {
5          Dictionary<string, object> result = new Dictionary<string, object
               >();
6          result.Add("status", status);
7          result.Add("data", data.toDictionary());
8          return result;
9      }
10 }
11
12 [System.Serializable]
13 public class KeyPressDataModel : STPDataModel {
14     public int keyId, actionId;
15     public int times_pressed;
16     public float time_when_pressed;
17     public override Dictionary<string, object> toDictionary() {
18         Dictionary<string, object> result = new Dictionary<string, object
               >();
19         result.Add("message", message);
20         result.Add("keyId", keyId);
21         result.Add("actionId", actionId);
22         result.Add("times_pressed", times_pressed);
23         result.Add("time_when_pressed", time_when_pressed);
24         return result;
25     }
26     public enum Keys {
27         keyId, actionId, activation_point, times_pressed, time_when_pressed
28     }
29 }
```

Isječak koda 3.13: Klasa za obradu poruke o pritisnutoj tipki

Primjer kako bi se realizirala serijalizacija pristigle informacije o novom registriranom pritisku tipke ima dodatnu enumeraciju Keys koja nije potrebna.

Otvoren je port, računalo prima i obrađuje poruke od elektromagnetskog gumba i pohranjuje povijest o pristiglim podacima. Jedino što preostaje je slanje zahtjeva uređaju.

Taj je posao značajno pojednostavljen zahvaljujući metodi "toDictionary", te je moguće koristiti slog koji će nadjačavanjem metode "toString" obraditi podatke koje su joj prosljeđene u zahtjev STP1.0 protokola koji elektromagnetski gumb može primiti preko otvorenog porta te obraditi.

```
1  using System.Collections.Generic;
2
3  public enum STPMethod { GET, POST, PUT, DELETE };
4
5  public struct STPCommand {
6      public string route;
7      public STPMethod method;
8      public Dictionary<string, string> data;
9
10     public override string ToString() => _prefix + "{" + _apiRoute + "," +
           _apiMethod + _apiData + "}";
11
12     private const string _prefix = "STP1.0";
13     private string _apiRoute { get { return "\"route\":\"" + route + "\"";
           } }
14     private string _apiMethod { get { return "\"method\":\"" + method.
           ToString() + "\""; } }
15     private string _apiData {
16         get {
17             if (data == null) {
18                 data = new Dictionary<string, string>();
19             }
20             if (data.Count < 1) return "";
21             string result = ",";
22             foreach (string key in data.Keys) {
23                 string value = data[key];
24                 if (int.TryParse(value, out _)) {
25                     result += "\"" + key + "\":" + value + ",";
26                 } else if (double.TryParse(value, out _)) {
27                     result += "\"" + key + "\":" + value + ",";
28                 } else if (bool.TryParse(value, out _)) {
29                     result += "\"" + key + "\":" + value.ToLower() + ",";
30                 } else {
```

```
                    result += "\"" + key + "\":\"" + value + "\",";
                    if (value.StartsWith("[{")) result = result.Replace("
                        \"[{", "[{");
                    if (value.EndsWith("}]")) result = result.Replace("}]\"
                        ", "}]");
                }
            }
            return result.Substring(0, result.Length - 1);
        }
    }
}
```

Isječak koda 3.14: Slog za obradu zahtjeva

U onoj klasi gdje je definirana varijabla "port" možemo onda dodati funkciju za slanje naredbe i time je uspostavljena dvosmjerna komunikacija između računala i elektromagnetskog gumba.

```
public void sendCommand(STPCommand command) {
    try {
        if (!port.IsOpen) {
            port.Open();
    } else
        port.Write(command.ToString());
    } catch {
        Debug.LogError("Failed to send command. Is the port initialized and
            open?");
    }
}
```

Isječak koda 3.15: Metoda za slanje zahtjeva

Slanje naredbi je sada vrlo jednostavno. Bez STPCommand klase, trebali bi svakog puta ponovno obraditi podatke prije slanja zahtjeva.

```
using UnityEngine;
using UnityEngine.UI;

public class RestartButton : MonoBehaviour {
    public SerialMonitor serialMonitor;
    private void Awake() {
        GetComponent<Button>().onClick.AddListener(callRestartRoute);
    }
    private void callRestartRoute() {
        serialMonitor.sendCommand(new STPCommand { method = STPMethod.GET,
            route = "/" });
```

```
11          }
12  }
```

Isječak koda 3.16: Implementacija gumba za ponovno pokretanje uređaja

4. Konačan proizvod

Konačan proizvod je upotrebljiv. Ispunjava osnovne ciljeve varijabilne točke aktivacije i varijabilnog hoda, nudi sučelje za jednostavno ostvarenje dvosmjerne komunikacije s uređajem, može samostalno raditi bez računala te ne predstavlja nikakvu značajnu opasnost za korištenje ukoliko je korisnik odgovoran u njegovom korištenju.

Taktilnost uređaja je vrlo zadovoljavajuće. Njegovo korištenje te desktop aplikacija izrađena za prilagodbu postavki na njoj je dovoljno intuitivno i bez dodatnih objašnjenja.

Glavni nedostatak u korisničkom iskustvu je sigurno potreba za priključivanjem uređaja na dva različita izvora napajanja, jedan za ESP32 Dasduino Wrover i jedan za elektromagnet, ali se to može riješiti povezivanjem uređaja na prijenosnu bateriju. To je moguće zato što se akcije registriraju na krajnjoj točki preko Bluetooth veze (ovo je izvedivo u slučajevima gdje se korisnik želi povezati na računalo, a da nema namjeru koristiti dodatni softver za prilagodbu postavki uređaja ili čitanje vrijednosti s uređaja). Kalibracija je moguća i bez računala pritiskom na gumb za "reset" uređaja te ono može kompletno samostalno funkcionirati ukoliko ima dostatno napajanje.

4.1. Analiza troškova

Ukupni troškovi projekta: 3,619.11 €.

4.1.1. Troškovi materijala

- 13.94 € - Dasduino CONNECTPLUS (ESP32)
- 8.99 € - OH9E Hall effect linear IC, 10 pieces (samo 1 korišten)
- 13.49 € - Heschen electromagnet DC 12V 25kg
- 3.50 € - Set of cables for breadboard, 65 pieces (korišteno: 20)
- 3.95 € - Set of female-female cables, 40 pieces (korišteno: 3)
- 12.00 € - RS-AB03J00-B adapter 12V 3A
- 0.10 € - Set of carbon-film resistors, 10 pieces (korišteno: 2)
- 4.00 € - OLED 128x64 I2C Monochrome Display SSD1306
- 6.57 € - Breadboard
- 0.88 € - Diotec DIT050N06 MOSFET

Ukupni troškovi materijala bi iznosili 67.42 €, ali s obzirom na korištene materijale stvarni su troškovi: 53.17 €.

4.1.2. Troškovi programskih alata

- 64.77 € - Fusion360 1-month subscription (korišteno: 2)
- 3.7 € - GitHub PRO 1-month subscription (korišteno: 2)

Ukupni troškovi za softverske alate: 136.94 €.

4.1.3. Troškovi osoblja

Troškovi osoblja će se izračunati prema estimacijama uloženih sati u toj ulozi te prema prosječnoj plaći za tu ulogu u Republici Hrvatskoj koristeći formulu:

$$((placa\_godisnje_{(HRK)} \ / \ 246) \ / \ 8) * sati\_rada \ / \ 7.5345 = placa_{(EUR)}$$

- 186 € - 24h rada - osoblje za planiranje i istraživanje (prema najnižoj plaći) [9]
- 1002 € - 96h rada - osoblje za modeliranje modela kućišta (prema prosjećnoj plaći) [10]
- 2023 € - 160h rada - osoblje za programiranje programske podrške za ugrađene sustave (prema prosjećnoj plaći) cite [11]
- 218 € - 48h rada - osoblje za programiranje računalnog programa za demonstraciju u Unity3D engine-u (prema najnižoj plaći) [12]

Ukupni troškovi osoblja: 3,429 €.

U sklopu troškova za softverske alate se nalazi GitHub PRO. To je omogućilo korištenje GitHub Copilot-a koji je ubrzao proces izrade programske podrške za uređaj (poboljšao efikasnost osoblja za programiranje programske podrške za ugrađene sustave) za 55.8%, a developeri za demonstraciju u Unity3D engine-u nisu koristili Copilot te su bili značajno sporiji u razvoju aplikacije. [13]

4.2. Korisničke upute

Uređaj se automatski kalibrira kad se uključi, ali je preporučena dodatna kalibracija uređaja preko računalne aplikacije kako bi se osigurale najbolje performanse ili resetirati uređaj jednom nakon što se prvi put uključi, držući tipku gumba gore kako se uređaj ne bi slučajno kalibrirao s gumbom u "pritisnutoj" poziciji.

Nakon što je sastavljen uređaj prema shemi spajanja, treba:

1. ponovno pokrenuti računalo
2. uključiti 12V adapter
3. priključiti USB-om uređaj na računalo

4. uključiti Bluetooth na računalu
5. spojiti se preko Bluetootha na uređaj
6. ostaviti uređaj u stanju mirovanja (gumb nije pritisnut) i kalibrirati ga aplikacijom za računalo

Uređaj će nakon toga raditi. Ako se podešavaju postavke uređaja, kao što su to pinovi za Hall senzor i elektromagnet, jačina elektromagneta ili popis akcija koje se izvode, predloženo je za svaki slučaj ponovno pokrenuti uređaj kako bi ono potpuno osvježilo svoje podatke.

5. Zaključak

Ciljevi projekta su ispunjeni i proizvod elektromagnetskog gumba je uspješno izrađen.

Omogućena je dvosmjerna komunikacija koja nudi opciju virtualizacije prikaza gumba u stvarnom vremenu, gumb može dodatno izvršiti više različitih akcija ovisno o pozicijama aktivacije te naposljetku uređaj nudi varijabilni hod promjenom jačine elektromagneta.

Međutim, procesom izrade uređaja postalo je jasno zašto takav proizvod još ne postoji na tržištu.

Naime, jedan od glavnih problema je veličina elektromagneta - teško je razviti dovoljno snažan elektromagnet koji je dovoljno malen za ovakve svrhe koji si može priuštiti svatko. Dok minijaturizacija elektromagnetskog gumba nije nemoguća, ne može se reći sa sigurnošću da će to biti lagan ili isplativ zadatak.

Također postoje problemi pregrijavanja elektromagneta i cijena njenog napajanja. Sami po sebi ovi problemi nisu pretjerano veliki dok se radi o jednom jedinom elektromagnetu, no kad bi se pokušala implementirati ova tehnologija u većem broju ovi će izazovi postati puno značajniji.

5.1. Budući rad

Ima mnogo potencijala za budući razvoj projekta. Prva mogućnost na koju bi se moglo raditi je mogućnost da jedna akcija može izvesti niz naredbi. Trenutno može registrirati samo pritisak jedne tipke, ali je moguće napraviti jednostavnu promjenu u kodu kako bi postalo moguće pohraniti niz naredbi u jednu akciju te bi se mogle npr. tipkati složene kombinacije poput lozinki ili bi se mogli snimiti makroi ako se uključi dodatni parametar za kašnjenje prije registracije pritiska tipki.

S druge strane, moguće je preinačiti elektromagnetski gumb u maglev gumb prilično jednostavnim remodeliranjem.

Također je moguće umjesto remodeliranja u maglev gumb izraditi hibridni model elektromagnetskog gumba i maglev gumba - elektro-maglev gumb - u kojemu bi dio tereta stvaranja snažnog magnetskog polja s elektromagneta preuzeo magnet prilijepljen na vrh elektromagneta. Time bi se mogao ublažiti problem pregrijavanja, gumb bi mogao funkcionirati i bez dodatnog napajanja te bi cjelokupni rang vrijednosti dostupnih za promjenu otpora mogao značajno porasti.

Moguće je i rješenje koje može realizirati sličan efekt kao što ima elektromagnetski gumb upotrebom pokretnog klipa umjesto trenutnog industrijskog elektromagneta. Ako se na vrh klipa stavi magnet, onda je moguće pomaknućem klipa postići isti efekt koji se trenutno postiže elektromagnetom.

Ovo bi moglo riješiti problem pregrijavanja u potpunosti, omogućilo bi varijabilan hod i varijabilan otpor tipke te bi teoretski bilo manje zahtijevnije što se tiče pitanja napajanja.

STP protokol 1.0 koji je razvijen u sklopu projekta će sigurno biti koristan u budućim projektima te će također omogućiti jednostavnu nadogradnju uređaja u budućnosti. Sam programski kod bi također mogao biti koristan u edukacijama o razvoju pozadinske strane programskih sustava.

SPIFFS-bazirana baza podataka može biti od velike koristi onima koji žele implementirati vlastita rješenja za lokalne baze podataka bez dodatnih biblioteka i koristit će se i u budućim projektima dok je usavršen.

Potencijal za računalne programe koju nudi elektromagnetski gumb je ogroman. Interesantan je izazov razviti neka od rješenja spomenutih pred kraj poglavlja 2.1.4. ili nešto slično koristeći API koji je ponuđen za to, a mogućnost virtualiziranog prikaza gumba može se rabiti u programima prividne stvarnosti ili običnim igricama kako bi se povećao osjećaj imerzije interakcijama s fizičkim modelom kojeg predstavlja virtualni objekt.

Popis literature

[1] P. Grant i T. Sheahen, „Cost Projections for High Temperature Superconductors," teh. izv., ožujak 2002.

[2] D. Garisto, „LK-99 isn't a superconductor — how science sleuths solved the mystery," *Nature*, 16. kolovoza 2023. (pogledano 28. 8. 2023.).

[3] Soldered, *Dasduino CONNECTPLUS s muškim headerima*. (pogledano 28. 8. 2023.).

[4] L. NanjingOuzhuoTechnologyco., *OH49E Series Linear Hall Effect IC*. (pogledano 26. 6. 2023.).

[5] C. C.
bibinitperiod Electronics, „SteelSeries Apex Pro Mini Keyboard - OmniPoint 2.0 - Double Shot PBT Keycaps Black," teh. izv. (pogledano 5. 7. 2023.).

[6] K. Panos, „Mag-Lev Switches Are the Future of Clacking," 2. kolovoza 2021. (pogledano 19. 1. 2023.).

[7] R. Sam, „Hall Effect Joysticks - Everything You Need to Know," 14. lipnja 2023. (pogledano 13. 7. 2023.).

[8] J. van der Merwe, „3 of the best game controllers for PC and console that eliminate joystick drift with hall-effect sensors," 5. srpnja 2023. (pogledano 13. 7. 2023.).

[9] S. Explorer, „Average Engineering Research and Development Manager Salary in Croatia for 2023," 2023. (pogledano 28. 8. 2023.).

[10] S. Explorer, „Average 3D Designer Salary in Croatia for 2023," 2023. (pogledano 28. 8. 2023.).

[11] S. Explorer, „Average Embedded Software Developer Salary in Croatia for 2023," 2023. (pogledano 28. 8. 2023.).

[12] S. Explorer, „Average Game Developer Salary in Croatia for 2023," 2023. (pogledano 28. 8. 2023.).

[13] S. Peng, E. Kalliamvakou, P. Cihon i M. Demirer, „The Impact of AI on Developer Productivity: Evidence from GitHub Copilot," Massachusetts Institute of Technology Sloan School of Management, teh. izv., 2023.

Popis slika

1. Prikaz odnosa akcija/odgovor [autorski rad] 2
2. ESP32 Dasduino Wrover [3] 6
3. Raspon vrijednosti Hall senzora [4] 7
4. OmniPoint 2.0 prekidač [5] 9
5. Maglev prekidač [autorski rad] 10
6. Skica veličina komponenti [autorski rad] 13
7. Preliminarna skica modela elektromagnetskog gumba [autorski rad] 14
8. Pogled odozgo na rotirajući mehanizam kliznog zaključavanja [autorski rad] . . . 14
9. Primjer dijelova za rotirajući mehanizam kliznog zaključavanja [autorski rad] . . . 15
10. Prikaz gumba dok nije pritisnut i dok je pritisnut [autorski rad] 16
11. Vertikalni presjek gumba dok nije i dok je pritisnut [autorski rad] 16
12. Shema povezivanja za elektromagnetski gumb [autorski rad] 17
13. Slika konačnog modela elektromagnetskog gumba [autorski rad] 17
14. Približavanje magneta senzoru bez filtera [autorski rad] 18
15. Približavanje i udaljavanje magneta senzoru s filterom [autorski rad] 19