SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1882

PRAĆENJE OSOBNIH AKTIVNOSTI UPORABOM DOSTUPNIH UGRADBENIH RAČUNALA I SENZORA

Lovro Urem

Zagreb, lipanj 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1882

PRAĆENJE OSOBNIH AKTIVNOSTI UPORABOM DOSTUPNIH UGRADBENIH RAČUNALA I SENZORA

Lovro Urem

Zagreb, lipanj 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 3. ožujka 2025.

ZAVRŠNI ZADATAK br. 1882

Pristupnik:	**Lovro Urem (0036548899)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Leonardo Jelenković

Zadatak: **Praćenje osobnih aktivnosti uporabom dostupnih ugradbenih računala i senzora**

Opis zadatka:

Istražiti dostupna i jeftina ugradbena računala i senzore kojima se mogu pratiti različite osobne aktivnosti, npr. otkucaje srca, kretanje, temperaturu i slično. Osmisliti i ostvariti sustav s nekim od istraženih komponenti koje omogućuju praćenje nekih osobnih aktivnosti. Uređaju neka bude moguće pristupiti izravno, npr. pametnim telefonom, ali i neizravno putem poslužitelja kojem uređaj šalje svoja očitanja. Osmisliti nekoliko načina rada uređaja u kojima će on ili omogućiti izravno praćenje očitanja ili će omogućiti dulji samostalni rad. Ocijeniti preciznost korištenih senzora.

Rok za predaju rada: 23. lipnja 2025.

ZAHVALA

Zahvaljujem svom mentoru, dr. sc. Leonardu Jelenkoviću, na stručnom vodstvu i pomoći tijekom izrade ovog rada. Zahvaljujem i profesoru dr. sc. Marinu Golubu na tehničkoj podršci prilikom lemljenja elektroničkih komponenti. Također zahvaljujem svojoj obitelji, djevojci i kolegama na podršci i razumijevanju kroz čitavo vrijeme studija. Na kraju, zahvaljujem i svom stricu na pomoći u izradi kućišta uređaja.

SADRŽAJ

1. UVOD

Nedovoljna fizička aktivnost predstavlja veliku prijetnju ljudskom zdravlju. Način života u današnjem dobu obilježen je sve većom upotrebom tehnologije što doprinosi sjedilačkom ponašanju. Posljedice fizičke neaktivnosti uključuju povećan rizik za razvoj kardiovaskularnih bolesti, moždanog udara, različitih oblika raka te mentalnih bolesti. U tom kontekstu nosivi uređaji predstavljaju učinkovito i dostupno rješenje. Osim što pružaju informacije o tjelesnoj aktivnosti ili trenutačnim fiziološkim podatcima, ovakvi uređaji potiču korisnike na povećanje fizičke aktivnosti, uglavnom putem tehnika promjene ponašanja kao što su samonadzor, postavljanje ciljeva i povratna informacija. Također, pružaju mogućnost za detekciju neočekivanih događaja kao što su padovi, nepokretnost ili nepravilnosti u srčanom ritmu. Trivijalno je za zaključiti kako nosivi uređaji danas sve više postaju osobni digitalni asistenti za zdravlje, prevenciju, sigurnost i skrb.

U ovom radu razvijen je prototip pametnog nosivog uređaja temeljenog na ESP32-S3 mikrokontroleru, sposoban za mjerenje aktivnosti osobe i fizioloških podataka. Uz uređaj izrađena je i prateća web aplikacija kojom je omogućeno upravljanje radnim načinima, pokretanje određenih mjerenja, kao i poslužiteljski sustav koji je zadužen za obradu i pohranu podataka te komunikaciju s uređajem.

Cilj rada bio je istražiti dostupna i jeftina ugradbena računala i senzore kojima se mogu pratiti različite osobne aktivnosti, te implementirati rješenje koje povezuje fizički uređaj s korisničkim sustavom, pritom naglašavajući nekoliko načina rada uređaja u kojima će on ili omogućiti izravno praćenje očitanja ili će omogućiti dulji samostalni rad.

U drugom poglavlju opisana je motivacija te tehnička podloga rješenja. Arhitektura ostvarenog sustava prikazana je u trećem poglavlju. Četvrto poglavlje detaljno prikazuje algoritme i logiku rada, dok se u petom poglavlju analiziraju rezultati testiranja. Šesto poglavlje raspravlja o uočenim problemima i mogućim poboljšanjima, a posljednje donosi zaključak rada.

2. TEHNIČKA PODLOGA I MOTIVACIJA

2.1. Postojeća rješenja

Nosivi uređaji za praćenje tjelesne aktivnosti i zdravlja danas su široko dostupni i popularni. Najpoznatiji primjeri uključuju uređaje kao što su Apple Watch, Fitbit, Xiaomi Mi Band, Garmin satova i ostalih. Funkcionalnosti koje pružaju sve su veće, od praćenja otkucaja srca, broja koraka, pa do prepoznavanja poremećaja sna i zabilježavanja razine stresa. Uz komercijalna rješenja, također postoje i izvedbe otvorenog tipa temeljene na mikrokontrolerima poput ESP32. Ovakvi sustavi mogu se posebno prilagođavati prema ciljevima istraživanja ili potrebama određene korisničke skupine.

Globalno tržište nosivih uređaja bilježi izniman rast u posljednjih deset godina. U odnosu na 2014. godinu, broj isporučenih uređaja u 2024. godini povećao se za više od 1900%, što ukazuje na snažan porast potražnje i sve širu primjenu ovih tehnologija [1].

2.2. Ideja i pristup rješavanju

Kao što je već rečeno, glavni cilj ovog rada je istražiti dostupna i jeftina ugradbena računala i senzore kojima se mogu pratiti različite osobne aktivnosti. Uz to je bilo potrebno osmisliti i ostvariti sustav s nekim od istraženih komponenti. Također, uređaju mora biti moguće pristupiti izravno, npr. pametnim telefonom, ali i neizravno putem poslužitelja kojem uređaj šalje svoja očitanja. Upravo ovakav pristup uklapa se u širi kontekst razvoja „Internet stvari“ (IoT), relativno novi koncept koji omogućuje komunikaciju između elektroničkih uređaja i senzora putem interneta s ciljem olakšavanja svakodnevnog života. IoT koristi pametne uređaje i internetsku povezanost kako bi pružio inovativna rješenja za različite izazove i probleme u poslovnim, državnim, javnim i privatnim sektorima diljem svijeta. Bitnu stavku koju nam pristup temeljen na IoT-u omogućava je razdvajanje slojeva obrade što uvelike povećava fleksibilnost i skalabilnost cijelog sustava [2].

Početni zahtjevi ovog završnog rada rezultirali su idejom izrade prototipa narukvice čime su obuhvaćeni svi željeni elementi. Rješavanju problema se pristupa evolucijskim modelom koji uključuje iterativni (ponavljajući ciklusi razvoja) i inkrementalni (nove verzije proizvoda su sve bolje) pristup razvoju. Područje određenog dijela problema se prvo istražuje, a stečeno znanje se koristi u svrhu rješavanja problema, i tako u krug [3].

Kao početna ideja sustav je osmišljen tako da:

- prima naredbe od korisnika putem web aplikacije
- sadrži podršku za više radnih modova (npr. Full, Half, Low)
- analizira podatke o kretanju
- detektira duboki san
- podatke obrađuje lokalno i na poslužitelju
- rezultate prikazuje korisniku na korisničkom sučelju.

Nastavno na Projekt R potrebno je bilo istražiti postojeće senzore i ugradbena računala, proučiti dostupne komponente, njihove cijene, recenzije, kompatibilnost te podršku u vidu programskih knjižnica. Nakon konačnog odabira komponenti, provedeno je testiranje svake od njih kroz pisanje primitivnog koda. Cilj ove faze je bilo uvjeriti se u ispravnost mjerenja, razumjeti kako knjižnice rade, itd. Sljedeći korak odnosi se na pisanje konkretnog koda kojim se osposobljava uređaj, a paralelno s time razvoj korisničkog sučelja i poslužitelja, ponovno kroz različite faze (početne postavke, jednostavna mjerenja, uspostava komunikacije, pisanje algoritama itd.). Kao zaključak praktičnog dijela provode se poboljšanja sustava i algoritama za obradu podataka te izrada jednostavnog kućišta 3D printerom.

2.3. Pregled tehnologija

2.3.1. Ugradbeni sustav i senzori

U ovome radu, kao glavni upravljački uređaj korišten je ESP32-S3, u izvedbi kompanije Seeed Studio (model Seeed Studio XIAO ESP32S3) [4]. Odgovoran je za povezivanje sa senzorima, prikupljanje podataka, lokalnu obradu te slanje podataka na poslužitelj putem Wi-Fi-a. ESP32 općenito predstavlja seriju mikrokontrolera poznatih po niskoj cijeni, visokoj dostupnosti i ugrađenoj Wi-Fi/Bluetooth podršci. Velik spektar podržanih komunikacijskih protokola, kao i analognih i digitalnih ulaza/izlaza, uz bogatu razvojnu zajednicu i otvorene alate omogućuju brzu izradu prototipova i široku primjenu u IoT području [5].

Za potrebe mjerenja osobnih aktivnosti korišteni su sljedeći senzori:

- MAX30102 – senzor za mjerenje otkucaja srca. Koristi fotopletizmografiju (eng. Photoplethysmography, PPG) za prikupljanje podataka. (DFRobot, SEN0344) [6]
- MLX90614 – infracrveni senzor za beskontaktno mjerenje temperature. Mjeri temperaturu objekta ispred sebe. (DFRobot, SEN0206) [7]
- LIS3DH – akcelerometar tri osi. Mjeri ubrzanje u tri međusobno okomite osi X, Y i Z. Koristi se za detekciju koraka na temelju promjene ubrzanja u sve tri osi i prepoznavanje naglih pokreta. (Adafruit, 2809) [8].

Kao izvor napajanja koristi se se Li-Po baterija, napona 3.7 V i kapaciteta 250 mAh (bežično) ili putem USB sučelja spajanjem na vanjski izvor napajanja (žično). Male dimenzije i mogućnost punjenja čine ju pogodnom za ugradnju u prijenosne uređaje. Za potrebe punjenja baterije koristi se modul za punjenje temeljen na punjačkom čipu MCP73831. (Adafruit, 4410) [9].

2.3.2. Razvojni alati

Sustav je podijeljen na tri ključne komponente: ugrađeni program za mikrokontroler, poslužitelj i korisničko sučelje. Za razvoj koda na mikrokontroleru korišten je PlatformIO,

ekstenzija za Visual Studio Code koja omogućuje kompilaciju i učitavanje programa na uređaj. Poslužitelj je razvijen korištenjem Spring Boot frameworka, Java okvira koji omogućuje brzo postavljanje REST API-ja i obradu zahtjeva s uređaja. Korisničko sučelje izrađeno je korištenjem Reacta, JavaScript okvira za izradu korisničkih sučelja. React osigurava dinamičko prikazivanje podataka i jednostavnu integraciju s poslužiteljskim programskim sučeljem (API-jem).

2.4. Motivacija

Motivacija za izradu ovog rada proizlazi iz želje za edukacijom i stjecanjem novih znanja iz područja IoT tehnologija, ugradbenih računala i senzora, a sve to tako da se stečeno teorijsko znanje primjeni na praktičan projekt. Ova tematika činila se interesantnom zbog svoje raširenosti i sveprisutnosti u svakodnevnom životu, kao i zbog činjenice da spaja nekoliko zanimljivih domena djelovanja: elektroniku, programiranje i zdravlje.

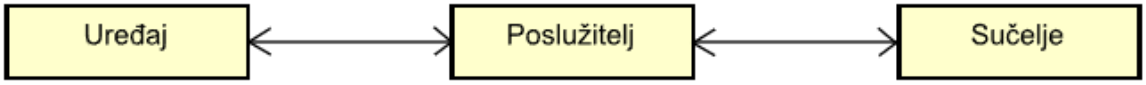
3. ARHITEKTURA OSTVARENA SUSTAVA

3.1. Uređaj – poslužitelj – korisničko sučelje

Programsko rješenje ovog zadatka izrađeno je na načelu troslojne arhitekture gdje entitete predstavljaju:

- uređaj tj. glavna upravljačka jedinka (ESP32)
- poslužitelj
- korisničko sučelje.

Načelo rada je jednostavno za opisati: postoje dvije dvosmjerne komunikacijske veze, veza uređaj – poslužitelj i veza poslužitelj – sučelje prema slici Slika 3.1. Prikaz arhitekture sustava



Slika 3.1. Prikaz arhitekture sustava

Ovakvo načelo razdvajanja slojeva je vrlo važna stavka u dizajnu sustava, osigurava samostalni razvoj i testiranje svake komponente. Odabir ovakvog pristupa i razvojnih alata navedenih u poglavlju 2.3.2. proizlazi iz prijašnjih iskustava u radu s njima.

3.2. Struktura API-ja

Sustav koristi REST API (Representational State Transfer), gdje REST predstavlja skup pravila koja definiraju način razmjene podataka između klijenta i poslužitelja. Dohvaćanje i manipulacija podatcima su omogućene korištenjem standardnih HTTP metoda kao što su GET, POST, PUT i DELETE. Kada klijent (npr. uređaj ili web aplikacija) pošalje zahtjev REST API-ju, poslužitelj vraća odgovor u nekom formatu (npr. JSON, HTML, XML ili plain text). U kontekstu ovog rada govorit će se o komunikaciji putem HTTP protokola, gdje se podaci razmjenjuju u JSON formatu.

Osnovna ideja rada API-ja jest:

- uređaj šalje podatke prema API-ju
- poslužitelj podatke prima
- frontend aplikacija šalje zahtjeve API-ju kako bi prikazala podatke.

Ovakav opis rada vrlo je štur, ali s razlogom. Naime, načelo rada se razlikuje ovisno o načinu rada u kojem se nalazi, ali i o promatranoj funkcionalnosti narukvice. Shodno tome, detaljniji osvrt nalazi se u nastavku ovog dokumenta (poglavlje 4.5.).

U nastavku slijede neki od primjera krajnjih točaka u sustavu.

Tablica 3.1. Primjeri krajnjih točaka

Endpoint	HTTP metoda	Opis funkcionalnosti	Tko šalje zahtjev
/mode	POST	Postavljanje novo odabranog načina rada	Korisnik
/mode	GET	Dohvaćanje trenutačnog načina rada	Korisnik / Uređaj
/steps	POST	Ažuriranje broja zabilježenih koraka	Uređaj
/steps	GET	Dohvaćanje broja zabilježenih koraka	Korisnik
/status	GET	Dohvaćanje informacije u trenutačnoj povezanosti uređaja i poslužitelja	Korisnik
/checkin	POST	Postavljanje vremenske oznake kada se uređaj spojio na poslužitelj	Uređaj

3.3. Programska potpora na poslužitelju

Kao što je već spomenuto, poslužitelj (engl. *backend*) je izrađen korištenjem radnog okvira Spring Boot, koji nam olakšava i ubrzava stvaranje REST API-ja. Aplikacija je organizirana po standardnom modelu MVC (Model – View – Controller), gdje je upravljački (kontrolerski) dio aplikacije zadužen za prihvat HTTP zahtjeva, uslužni (servisni) dio upravlja logikom, a klase DTO (Data Transfer Object) služe za prijenos podataka. Spremanje u trajnu bazu podataka nije ostvareno već je ostavljeno za buduća proširenja.

Na primjer, klasa „CommandDTO“ sadrži varijable poput: *mode, command, result, steps* i *sleep*, a koristi se za jednostavno pakiranje podataka. U servisnom sloju postoji klasa „CommandService“ koja je zadužena za upravljanje trenutačnim stanjem uređaja, dakle mod rada, aktivna naredba, rezultat naredbe, broj zabilježenih koraka itd. Odgovarajući upravljač, „CommandController“, ponaša se kao posrednik između HTTP zahtjeva i opisanog servisnog sloja. Jedan primjer rada ove tri klase bio bi sljedeći:

Kod 3.1. Primjer metode u upravljaču

```
@PostMapping("/mode")
    public ResponseEntity<Void> setMode(@RequestBody CommandDTO dto) {
        commandService.setMode(dto.getMode());
        return ResponseEntity.ok().build();
    }
```

- upravljač prima HTTP zahtjev
- validira ulazni JSON pomoću DTO objekta
- mapira ulazni JSON u „CommandDTO“

- šalje podatke u servisni sloj
- vraća odgovor klijentu.

3.4. Sučelje prema korisniku

Sučelje prema korisniku (engl. frontend) je izrađeno korištenjem React JavaScript knjižnice i podijeljeno je na dvije glavne stranice: početnu stranicu (Homepage) i stranicu za postavke (Settings). Aplikacija koristi razvojno okruženje Vite, module za oblikovanje stilova CSS i React Router za navigiranje u aplikaciji, a sva komunikacija s poslužiteljem se odvija preko *fetch* poziva.

Početna stranica sadrži 6 kartica, njihova zadaća je da prikažu sljedeće:

- trenutačna vrijednost pulsa (mjerenje na zahtjev korisnika)
- trenutačna vrijednosti tjelesne temperature (mjerenje na zahtjev korisnika)
- količina napravljenih koraka u tom danu (resetiranje vrijednosti u ponoć)
- podaci o spavanju korisnika (odnosi se na posljednji ciklus spavanja)
- paljenje i gašenje ciklusa spavanja korisnika
- procjena potrošenih kalorija i prijeđene kilometraže.

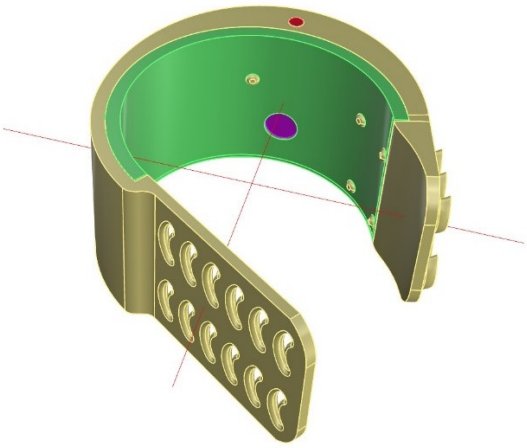
Pritiskom na gumb za postavke otvara se druga stranica na kojoj je moguće odabrati jedan od tri ponuđena načina rada. Konkretan primjer komunikacije opisan je u poglavlju 4.5..

Slika 3.2. Prikaz korisničkog sučelja

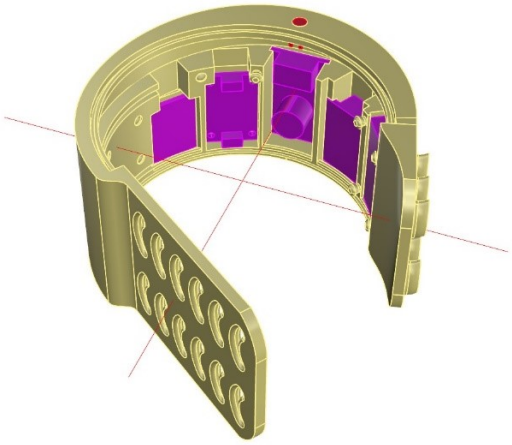
3.5. Kućište

Razvoj kućišta za uređaj proveden je u nekoliko faza, a modeliranje je obavljeno uz program Rhinoceros 8. Za pripremu modela za 3D ispis korišten je Pruca Slicer, dok je sam ispis izvršen na 3D printeru Prusa XL, koristeći fleksibilni filament tipa TPU (Thermoplastic Polyurethane). Korištena je mlaznica promjera 0.4 mm i visina sloja od 0.2 mm. Temperatura ispisa bila je 235 °C, uz hotbed 50 °C.

Tijekom razvoja printane su tri inačice kućišta. U drugoj verziji primijenjene su izmjene na remenu kako bi se bolje prilagodio ruci korisnika uz preinake vezane za dodavanje prostora za bateriju. U trećoj verziji prilagodbe su izvršene s ciljem optimizacije prostora za kablova i bolji smještaj uređaja. Modeliranje se temeljilo na preciznim mjerenjima dimenzija komponenti s pomoću šublera. Prvo je nacrtan 2D poprečni presjek u koji su smještene komponente po širini, a zatim je taj presjek ekstrudiran u visinu. Za kraj je izdubljen prostor potreban za smještanje komponenti unutar narukvice, te su dodana izdubljena i rupe na remenu.



Slika 3.3. Kućište s poklopcem

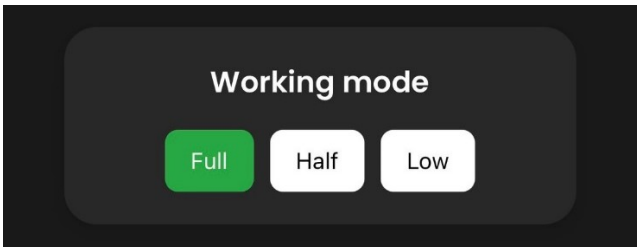


Slika 3.4. Kućište bez poklopca

4. ALGORITMI I LOGIKA RADA

4.1. Upravljanje načinima rada

Jedna od bitnijih stavki nosivog uređaja je njegova potrošnja baterije. Kako bi se osigurala što duža upotreba uređaja bez potrebe za punjenjem, u sustavu je izrađen mehanizam za upravljanje tri različita načina rada. Korisnik način rada bira kroz korisničko sučelje, na stranici za postavke.



Slika 4.1. Prikaz stranice za postavke

Ovaj problem programski je riješen tako da je implementirana funkcija `loop()`, njena je zadaća da radi upravo ono što joj ime sugerira, ponavlja se uzastopno te omogućuje programu da se mijenja i reagira. Na temelju trenutačnog stanja sustava koje je spremljeno u varijablu `state`, glavna petlja poziva odgovarajuću funkciju.

Kod 4.1. Glavna petlja

```
void loop() {
  if (state == Initial) {
    handleInitialState();
  }
  else if (state == Full) {
    loopFullMode();
  }
  else if (state == Half) {
    loopHalfMode();
  }
  else if (state == Low) {
    loopLowMode();
  }
  ...
}
```

Polazeći od načina rada s najnižom potrošnjom baterije prvo se razmatra mod *Low,* koji je osmišljen na sljedeći način. Potrošnja baterije mora biti minimalna, a želi se postići da uređaj ne radi praktički ništa i to sve dok ne dobije konkretnu naredbu od korisnika. Shodno tome, u ovom načinu rada pogodno je koristiti tzv. „Deep-Sleep“, mod ugrađen u korišteni mikrokontroler u kojem se isključuje CPU, većina RAM memorije i sve digitalne periferne

jedinice koje koriste APB_CLK takt [11]. Mikrokontroler se iz ovog stanja može probuditi na nekoliko načina, a u ovoj implementaciji koristi se vanjsko buđenje (External Wake-up). ESP32 se u ovom slučaju budi vanjskim signalom, npr. pritiskom na tipku. Kako bi se ovo omogućilo, na prototip narukvice dodana je tipka s tom svrhom. Pritiskom na tipku za vrijeme rada u *Low* modu, mikrokontroler se budi, a stanje sustava prelazi u *Initial*.

Sljedeći način rada bio bi *Half*, koji je zamišljen na način da konstantno bilježi podatke o broju prijeđenih koraka, a na poslužitelj se spaja u dva slučaja. Prvi slučaj je kada brojač koraka dođe do 500, u tom trenutku uređaj se spaja na poslužitelj, šalje HTTP zahtjev, poslužitelj obrađuje zahtjev, ažurira svoju vrijednost brojača koraka te se ažurirana vrijednost prikazuje na korisničkom sučelju. Drugi slučaj je situacija u kojoj korisnik pritisne tipku opisanu u prethodnom paragrafu, koja i u ovom slučaju mijenja stanje sustava u *Initial*. U *Half* načinu rada korišten je tzv. „Modem-Sleep“, također jedan važan način rada s niskom potrošnjom energije na ESP32. U ovom modu, Wi-Fi/Bluetooth modul ulazi u stanje mirovanja, dok CPU jezgre ostaju aktivne. To omogućuje mikrokontroleru da zadrži određenu razinu bežične povezanosti uz značajno smanjenje potrošnje energije.

Posljednji od tri načina rada je *Full*, zamišljen tako da u svakom trenutku omogućuje sve interakcije korisnika s korisničkim sučeljem, bez potrebe za pritiskanjem tipke na uređaju. Osim što je veza između uređaja, poslužitelja i korisničkog sučelja konstantna, podatak o broju koraka se ažurira svakih 10 koraka.

Initial se odnosi na prijelazno stanje u koje ulazimo iz *Low* i *Half* modova, a koje nam omogućuje privremeno aktiviranje uređaja i zadavanje željenih naredbi. Uređaj se nakon pritiska tipke u ovom stanju zadržava 5 sekundi, nakon čega se vraća u prethodni način rada ili prelazi u novo zadani preko korisničkog sučelja.

4.2. Detekcija koraka

Detekcija koraka jedna je od glavni komponenti pametnih senzorskih sustava za praćenje tjelesne aktivnosti korisnika. Kao osnovni senzor za detekciju koristi se akcelerometar jer omogućuje praćenje ubrzanja u tri smjera. Glavna ideja temelji se na prepoznavanju lokalnih maksimuma signala ubrzanja, gdje se korak bilježi kao prelazak vrijednosti ubrzanja preko određenog praga.

Za početak se sirovi podaci akcelerometra iz sve tri osi pretvaraju u jednu skalarnu vrijednost koja predstavlja ukupnu količinu gibanja. Ta vrijednost naziva se magnituda, a računa se po sljedećoj formuli:

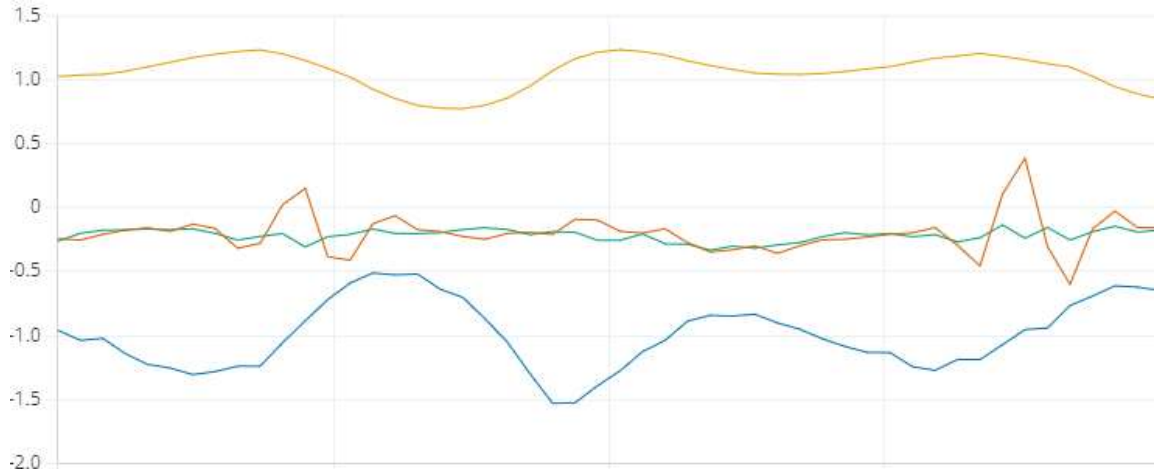
$$M = \sqrt{x^2 + y^2 + z^2} \tag{4.1.}$$

Sljedeći korak je filtriranje i zaglađivanje magnitudne vrijednosti. Naime, sirovi podaci s akcelerometra mogu biti poprilično bučni (Slika 4.2. Prikaz magnitude prije filtriranja), što znači da će se vrijednosti signala nepredvidivo mijenjati zbog okolnih utjecaja kao što su vibracije i blagi pokreti. Filtriranje provodimo po sljedećoj formuli:

$$Filtered = \alpha \times M \times (1 - \alpha) \times Filtered \quad (4.2.)$$

M predstavlja trenutačno izmjerenu magnitudu, Filtered se odnosi na prethodno filtriranu vrijednost, a α određuje koliko brzo se reagira na promjene (manji α = signal se sporije mijenja, veći α = brže reagira, ali više šuma) [12] [13]. Na sljedećim grafovima žutom bojom je prikazana promjena iznosa magnitude kroz vrijeme, dok su narančastom, zelenom i plavom bojom označeni iznosi ubrzanja u pojedinim smjerovima (X, Y i Z osi).

Slika 4.2. Prikaz magnitude prije filtriranja



Slika 4.3. Prikaz magnitude nakon filtriranja

Nadalje, nije dovoljno samo pratiti prelazak magnitude preko određenog praga, time bi svaki lokalni maksimum bio prepoznat kao korak. Iz tog razloga provodi se odbacivanje ostalih pokreta. Kada se naiđe na lokalni maksimum koji prijelazi prag, zabilježava se taj trenutak u varijablu *now*. Ukoliko je od zadnjeg zabilježenog pokreta prošlo manje od 500 ms ili više od 1400 ms, taj pokret karakteriziran je kao nešto što nije korak. Međutim ako prethodni uvjet nije ispunjen, a vremenski razmak je veći od 500 ms taj pokret zabilježen je kao korak. Ovim jednostavnim postupkom značajno se povećava preciznost ovog algoritma.

Kod 4.2. Odbacivanje ostalih pokreta

```
...
if (prev1 > prev2 && prev1 > filtered){
    if (prev1 > 1.2){
      unsigned long now = millis();
      if(now - lastMovement < 500 || now - lastMovement > 1400){
        lastMovement = now;
        lastStepTime = now;
      }
      else if(now - lastStepTime > 500){
        lastStepTime = now;
        lastMovement = now;
        return true;
      }
    }
    else if(prev1 > 1.0){
      unsigned long now = millis();
      lastMovement = now;
    }
  }
...
```

4.3. Praćenje aktivnosti za vrijeme spavanja

Poznata je činjenica kako je preporučeno spavati između sedam i devet sati svake noći, a da spavanje kontroliraju biološke, socijalne i okolinske vremenske oznake koju uključuju svjetlost kojoj smo izloženi, doba dana u koje jedemo, vježbamo ili smo u interakciji s drugima te mnoge druge. Faze sna dijele se na NREM (Non-Rapid Eye Movement) i REM (Rapid Eye Movement), a tijekom noći se izmjenjuju [14] [15]. Ukratko:

- faza 1 i 2 (NREM) – lagani san, prijelaz iz budnosti, usporavanje disanja i otkucaja
- faza 3 i 4 (NREM) – duboki san, tijelo se regenerira, teško je probuditi osobu
- faza 5 (REM) – lagani san u kojem se sanja, a mozak je vrlo aktivan.

Za vrijeme dubokog sna u tijelu se događaju različiti procesi, sjećanja se konsolidiraju, pojavljuje se smanjenje za 10 do 30 % krvnog tlaka, disanja i intenziteta bazalnog metabolizma. Valovi moždane aktivnosti su sinkronizirani, velikih amplituda i male frekvencije, a spavač je miran i tek ponekad mijenja položaj tijela. U zdravih odraslih osoba, oko 13 do 23 % sna je upravo duboki san [16] [17].

Algoritam praćenja spavanja je zamišljen na sljedeći način, uređaj u intervalima pokreće senzor za mjerenje pulsa, uzima 5 rezultata te računa njihovu srednju vrijednost. Uz srednju vrijednost sprema i vremensku oznaku kada je taj ciklus mjerenja obavljen. Nakon što skupi 20 mjerenja i njihove vremenske oznake, spaja se na poslužitelj i šalje te podatke. Uz otkucaje srca, prvobitno je zamišljeno da akcelerometar na sličan način kao kod detekcije

koraka zabilježava pokrete osobe te da poslužitelj na temelju podataka o otkucajima srca i korisnikovim pokretima procjenjuje vrijeme provedeno u dubokom snu. Međutim, zbog baterijskih ograničenja odlučeno je kako će se rad s akcelerometrom izostaviti.

Naime, rad akcelerometra zahtjeva praktički konstantno zabilježavanje podataka i provjeravanje je li prepoznat lokalni maksimum. U ovoj situaciji potrebno je da mikrokontroler bude u normalnom načinu rada, a to stvara problem s potrošnjom baterije. Za vrijeme testiranja prvi pokušaj rješavanja ovog problema bilo je korištenje „Modem-Sleep“ načina rada između slanja podataka, u kojem Wi-Fi/Bluetooth modul ulazi u stanje spavanja, dok CPU ostaje aktivan. Međutim, ovakav pristup nije dao zadovoljavajuće rezultate. Nakon toga pokušano je s potpunom deaktivacijom Wi-Fi veze sve do trenutka kada je potrebno spajanje kako bi se podaci poslali na poslužitelj, uz povećanje vremenskog intervala između pojedinih slanja podataka. Ponovno rezultati nisu bili dovoljno dobri s obzirom na to da mikrokontroler u načinu rada bez Wi-Fi konekcije svejedno troši oko 30 mA. U kombinaciji s aktiviranjem pulsnog senzora, koji tijekom mjerenja može povući i do 50 mA, te stalnim radom akcelerometra, ukupna potrošnja ostala je previsoka.

Kao krajnje rješenje iskorišten je „Light-Sleep“ način rada mikrokontrolera između pojedinih mjerenja pulsa, čime se značajno smanjuje potrošnja energije. No, ova odluka onemogućuje rad akcelerometra potreban za detektiranje trenutačnih pokreta korisnika.

4.4. Zabilježavanje i pohrana podataka

Sustav je temeljen na kombinaciji privremene lokalne pohrane podataka unutar mikrokontrolera i slanja podataka na udaljeni poslužitelj kada su zadovoljeni uvjeti za slanje. Kako trajna baza podataka nije implementirana, podaci se spremaju na poslužitelju gdje se redovito čiste. Npr. podaci o broju koraka bilježe se za taj dan te se resetiraju svakog dana u ponoć, podaci o kvaliteti sna korisnika ostaju spremljeni na poslužitelju sve dok se ne pokrene novi ciklus spavanja, podaci o mjerenjima trenutačnih vrijednosti otkucaja srca i temperature se brišu svakim novim zahtjevom za mjerenjem. Ovakav pristup je odabran kako se izvedba rada ne bi previše zakomplicirala zbog velikog opsega različitih područja.

Za zabilježavanje podataka sa senzora korištene su sljedeće vanjske knjižnice:

- ArduinoJson – serijalizacija podataka u JSON
- Adafruit_MLX90614 – zabilježavanje podataka temperaturnog senzora
- DFRobot_BloodOxygen_S – zabilježavanje podataka o otkucajima srca
- Adafruit_Sensor – generička knjižnica za rad s Adafruit senzorima
- Adafruit_LIS3DH – zabilježavanje podataka o akceleraciji.

4.5. Komunikacija

Dio sustava zadužen za komunikaciju omogućuje povezivanje krajnjeg korisnika, mikrokontrolera i poslužitelja u jednu cjelinu. Kao što je prikazano u poglavlju 3.1.,

komunikacija se odvija višeslojno, na razini korisničkog sučelja, poslužitelja te mikrokontrolera i senzora. Kroz sljedeća potpoglavlja daju se jednostavni primjeri komunikacija različitih slojeva.

4.5.1. Komunikacija s korisnikom

Komunikacija s korisnikom (frontend – backend) sadrži jednostavan načelo gdje se HTTP zahtjevi ostvaruju putem `fetch()` poziva. Klikom korisnika na određeni gumb generira se GET ili POST zahtjev prema krajnjim točkama na poslužitelju. Prije generiranja HTTP zahtjeva provjerava se povezanost uređaja s poslužiteljem korištenjem klasičnog „heartbeat“ mehanizma, gdje se bilježi posljednje vrijeme kada se uređaj „javio“ poslužitelju. Ukoliko od zadnjeg javljanja nije prošlo više od 5 sekundi pretpostavlja se aktivna veza poslužitelja i uređaja.

Ukoliko je uređaj trenutno u komunikaciji s poslužiteljem korisniku je dozvoljeno zadati naredbu putem korisničkog sučelja, u protivnom se na sučelju pojavljuje poruka s upozorenjem kako naredbu nije moguće zadati, te da je potrebno uređaj povezati s poslužiteljem pritiskom tipke na narukvici. U nastavku slijedi jedan primjer komunikacije.

Kod 4.3. Fetch poziv za provjeru povezanosti uređaja i poslužitelja

```
const checkDeviceStatus = async () => {
        try {
            const res = await fetch("http://<IPadress:port>/status");
            if (!res.ok) throw new Error("Status check failed");
            const status = await res.text();
            return status === "connected";
        } catch (error) {
            console.error("Device status check error:", error);
            return false;
        }
    };
```

Kada na rutu `/status` stigne HTTP GET zahtjev, upravljač poziva metodu servisnog sloja za provjeru povezanosti. Metoda `isDeviceConnected()` provjerava da li se uređaj javljao u posljednjih 5 sekundi te vraća vrijednosti `true` ili `false`. Na temelju ovog rezultata, upravljač slijedno vraća odgovor frontendu u obliku običnog teksta.

Kod 4.4. Provjera povezanosti uređaja putem upravljača

```
@GetMapping("/status")
    public ResponseEntity<String> getDeviceStatus() {
        boolean connected = commandService.isDeviceConnected();
        return ResponseEntity.ok(connected ? "connected":"disconnected");
    }
```

4.5.2. Komunikacija uređaj – poslužitelj

U svrhu komunikacije uređaja s poslužiteljem koristi se knjižnica HTTPClient. GET zahtjevi koriste se za dohvaćanje načina rada i naredbi, dok se POST koristi za slanje podataka poslužitelju. Kao primjer možemo uzeti sljedeći tok događaja. Uređaj šalje GET zahtjev na rutu `/command` kako bi provjerio postoji li nova naredba. Ako je odgovor uspješan, uređaj dohvaća tekst odgovora te ovisno o sadržaju tog teksta poziva odgovarajuću funkciju.

Kod 4.5. Dohvat i izvršavanje naredbe s poslužitelja

```
HTTPClient httpCommand;
httpCommand.begin("http://<IPadress:port>/command");
int CmdCode = httpCommand.GET();

if (CmdCode == 200) {
      String command = httpCommand.getString();
      if (command == "measure_heartbeat") {
        measureHeartbeat();
      }
      else if(command == "measure_temperature"){
        measureTemperature();
      }
}
httpCommand.end();
```

4.5.3. Komunikacija mikrokontroler – senzori

Za komunikaciju mikrokontrolera i senzora koristi se I2C sabirnica, a senzori su definirani sljedećim adresama:

- MAX30102 (mjerenje pulsa): 0x57
- MLX90614 (mjerenje temperature): 0x5A
- LIS3DH (akcelerometar): 0x18.

Prijenos podataka događa se putem Wire protokola s definiranim pinovima SDA i SCL, pri čemu SDA služi za prijenos podataka, dok SCL služi za sinkronizaciju prijenosa podataka. Senzori koriste odgovarajuće funkcije iz svojih knjižnica za čitanje vrijednosti:

- `oximeter.getHeartbeatSPO2();` – čitanje pulsa
- `mlx.readObjectTempC();` – čitanje tjelesne temperature
- `lis.read();` – očitanje akcelerometra

4.5.4. Kompletan primjer komunikacije

U nastavku je dan kompletan primjer komunikacije u situaciji kada se uređaj nalazi u *Half* načinu rada, a korisnik želi provesti mjerenje pulsa.

1. Korisnik pritiskom na tipku na uređaju aktivira uređaj, koji se iz *Half* načina rada prebacuje u prijelazno stanje *Initial*. Wi-Fi modul se ponovno aktivira, a uređaj se

spaja na poslužitelj putem HTTP zahtjeva prema ruti `/checkin`. Uz to, na korisničkom sučelju se pojavljuje iskočni prozor da se onemogući slanje novih zahtjeva dok uređaj provodi mjerenje.

2. Korisnik na korisničkom sučelju odabire opciju mjerenja pulsa. Ukoliko poslužitelj može korisničkom sučelju potvrditi povezanost uređaja, pokreće se POST zahtjev na rutu `/command` s tijelom : `{ "command": "measure_heartbeat" }`.
3. Poslužitelj prima naredbu te ju sprema u servisni sloj tako što ažurira vrijednost varijable `command`.
4. Uređaj u stanju *Initial* šalje GET zahtjev na rutu `/command`. Ako primi odgovor u kojem se nalazi vrijednost „measure-heartbeat“, poziva se funkcija `measureHeartbeat()` koja aktivira senzor za mjerenje pulsa.
5. Nakon završetka mjerenja, uređaj šalje POST zahtjev na rutu `/result` s JSON objektom: `{ "result": "82" }`. Poslužitelj prima rezultat te ga sprema. Paralelno, čisti polje command tako da se ista naredba ne izvrši ponovno.
6. Korisničko sučelje dohvaća rezultat tako što koristi GET zahtjev prema `/result` kako bi dohvatila posljednje dostupno mjerenje, a rezultat se prikazuje unutar kartice za puls.

5. TESTIRANJE I ANALIZA

Testiranje je provedeno u tri faze: testiranje senzora, testiranje algoritama, testiranje cjelokupnog sustava, a sve faze su testirane na tri osobe. Prva faza odnosi se na izolirano testiranje korištenih senzora kako bi se provjerio njihov odziv i ponašanje u različitim uvjetima.

- Pulsni senzor – utvrđeno je kako najpreciznija mjerenja daje kada se puls mjeri pritiskom prsta na senzor, dok mjerenja obavljena preko zapešća uglavnom daju očitanja uvećana za faktor od 1.2, a u slučaju lošeg pozicioniranja može se dogoditi da senzor ne očitava ništa. Također je potvrđeno kako su mjerenja kod osoba s tamnijom boje kože nepreciznija u odnosu na osobe sa svjetlijom boje kože. Ovo se događa zbog činjenice da tamnija koža sadrži više melanina, pigmenta koji apsorbira više svjetlosti, čime se smanjuje intenzitet reflektirane svjetlosti [18].
- Akcelerometar – testiranjem utvrđena visoka preciznost i zadovoljavajući rezultati.
- Temperaturni senzor – testiranjem utvrđeno kako je temperatura izmjerena na zapešću uglavnom manja od središnje tjelesne temperature za otprilike 2 °C.

Druga faza odnosi se na testiranje implementiranih algoritama, glavni dio ove faze bila je kalibracija algoritama za detekciju koraka i dubokog sna. Koraci poduzeti za kalibraciju ovih algoritama opisani su u poglavljima 4.2. i 4.3..

Zadnja faza podrazumijeva testiranje kompletnog sustava, koje je provođeno u višesatnim razdobljima u stvarnim uvjetima. Testiranje je potvrdilo funkcionalnost, ali i ukazalo na ograničenja koja proizlaze iz korištenja višestrukih komponenti na općem mikrokontroleru.

5.1. Potrošnja baterije

Optimizacija potrošnje baterije predstavlja jedan od ključnih izazova prilikom dizajna ovakvih sustava. Shodno tome, analiza potrošnje baterije je bila nužna kako bi se osigurao što duži rad po jednom punjenju. Uređaj koristi Li-Po bateriju kapaciteta 250 mAh, a potrošnja ovisi o načinu rada mikrokontrolera, broju aktivnih senzora te trajanju i frekvenciji mjerenja. Potrošnja pojedinih komponenti sustava prikazana je u tablici Tablica 5.1.

Tablica 5.1. Potrošnja struje pojedinih komponenti

Komponenta	Potrošnja za vrijeme rada (mA)
Pulsni senzor	50
Temperaturni senzor	1.2
Akcelerometar	1
Mikrokontroler (Wi-Fi omogućen)	100
Mikrokontroler (Modem-Sleep)	30
Mikrokontroler (Light-Sleep)	2
Mikrokontroler (Deep-Sleep)	0.014

Trajanje baterija u ovisnosti o načinu rada:

- *Full*: oko 2 sata
- *Half*: oko 5 sati
- *Low*: oko 7 dana (procjena).

6. PROBLEMI I MOGUĆA POBOLJŠANJA

Za vrijeme izrade uređaja putem se nailazilo na razne probleme, međutim uspješno je ostvarena zadovoljavajuća razina funkcionalnosti cijelog sustava. Jedan od glavnih izazova bio je ostvariti energetsku učinkovitost, što je naposljetku uspješno provedeno. Drugi problem stvarao je senzor za mjerenje pulsa, koji se pokazao vrlo osjetljiv na pozicioniranje i razinu pritiska na kožu, što je ponekad uzrokovalo da algoritam predugo čeka na dovoljno valjanih uzoraka mjerenja. Ovaj problem je djelomično riješen uvođenjem gornje granice broja uzoraka za određeni ciklus mjerenja, pri čemu se u slučaju da unutar te granice nema dovoljno valjanih mjerenja za trenutačnu vrijednost uzima rezultat prethodnog ciklusa. Svi ostali funkcionalni zahtjevi ispunjeni su vrlo zadovoljavajuće. Glavno poboljšanje što se tiče ovog dijela bilo bi korištenje boljeg pulsnog senzora, kao i izrada vlastite PCB pločice. Ovim potezom bi bilo moguće smanjiti dimenzije uređaja, ali i značajno smanjiti potrošnju energije.

Kako primarni fokus ovog rada nije bila sigurnost, već funkcionalnost sustava, nisu implementirani mehanizmi za nadzor integriteta podataka.Web-aplikacija i poslužitelj nisu javno dostupni putem interneta, već se izvršavaju lokalno na računalu unutar iste mreže kao i uređaj. S obzirom na to da rješenje nije produkcijski orijentirano, komunikacija se obavlja putem HTTP protokola bez primjene enkripcije. Sljedeći logičan korak na ovom polju bio bi uvođenje autentikacije te implementacija HTTPS protokola što bi uvelike podiglo razinu sigurnosti kompletnog sustava. Za približavanje sustava krajnjem korisniku bilo bi potrebno provesti izradu mobilne aplikacije i postavljanje produkcijske verzije. Uz sve spomenuto bilo bi potrebno i stvoriti bazu podataka za stabilno i dugoročnije pohranjivanje podataka.

7. ZAKLJUČAK

U ovom radu razvijen je funkcionalan prototip nosivog uređaja temeljenog na ESP32-S3 mikrokontroleru, sposoban za praćenje osobne aktivnosti korisnika, s pripadajućom web aplikacijom i lokalnim poslužiteljem. Iako nije riječ o produkcijskom rješenju, ostvarena je visoka razina integracije između sklopovskog i programskog dijela sustava. Uređaj omogućuje rad u više načina rada s različitim profilima potrošnje energije, a tijekom testiranja istaknuto je nekoliko važnih područja za daljnje poboljšanje.

Za vrijeme izrade ovog završnog rada prošlo se kroz čitavi razvojni ciklus, od dizajna arhitekture sustava te izrade programske podrške i korisničkog sučelja, do programiranja mikrokontrolera i optimizacije energetske učinkovitosti. Na ovaj način obuhvaćeno je cjelokupno iskustvo integracije senzora, mikrokontrolera, mrežne komunikacije i korisničke aplikacije. Kroz cjelokupni proces stečeno je vrijedno i praktično iskustvo u radu s ugradbenim računalima, upravljanju potrošnjom energije te dizajnu i testiranju sustava u stvarnim uvjetima.

LITERATURA

[1] F. Laricchia, »Wearables unit shipments worldwide from 2014 to 2028,« 7 Listopad 2024. [Mrežno]. Available: https://www.statista.com/statistics/437871/wearables-worldwide-shipments/. [Zadnji pristup 6 Lipanj 2025].

[2] A. R. Sfar, Z. Chtourou i Y. Challal, »A systemic and cognitive vision for IoT security: A case study of military live simulation and security challenges,« u *2017 International Conference on Smart, Monitored and Controlled Cities (SM2C)*, Sfax, Tunisia, 2017.

[3] »Modeli procesa programskog inženjerstva,« Fakultet elektrotehnike i računarstva (FER), 2024.

[4] S. Studio, »Getting Started with Seeed Studio XIAO ESP32S3 Series,« Seeed Studio, [Mrežno]. Available: https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/. [Zadnji pristup 6 Lipanj 2025].

[5] E. Systems, »ESP32 Product Overview,« Espressif Systems, [Mrežno]. Available: https://www.espressif.com/en/products/socs/esp32. [Zadnji pristup 6 Lipanj 2025].

[6] DFRobot, »Heart Rate and Oximeter Sensor V2 SKU SEN0344,« DFRobot, [Mrežno]. Available: https://wiki.dfrobot.com/Heart_Rate_and_Oximeter_Sensor_V2_SKU_SEN0344. [Zadnji pristup 6 Lipanj 2025].

[7] DFRobot, »IR Thermometer Sensor MLX90614 SKU SEN0206,« DFRobot, [Mrežno]. Available: https://wiki.dfrobot.com/IR_Thermometer_Sensor_MLX90614_SKU__SEN0206. [Zadnji pristup 6 Lipanj 2025].

[8] Adafruit, »Adafruit LIS3DH Triple-Axis Accelerometer,« Adafruit, [Mrežno]. Available: https://www.adafruit.com/product/2809. [Zadnji pristup 6 Lipanj 2025].

[9] Adafruit, »Adafruit Micro-Lipo Charger for LiPoly Batt with USB Type C Jack,« Adafruit, [Mrežno]. Available: https://www.adafruit.com/product/4410. [Zadnji pristup 6 Lipanj 2025].

[10] »Arhitektura programa,« Fakultet elektrotehnike i računarstva (FER), 2024.

[11] S. Studio, »XIAO ESP32S3 Sense Sleep Modes,« Seeed Studio, [Mrežno]. Available: https://wiki.seeedstudio.com/XIAO_ESP32S3_Consumption/. [Zadnji pristup 6 Lipanj 2025].

[12] A. Abadleh, E. Al-Hawari, E. Alkafaween i H. Al-Sawalqah, »Step detection algorithm for accurate distance estimation using dynamic step length,« u *2017 18th IEEE International Conference on Mobile Data Management (MDM)*, Daejeon, Korea (South), 2017.

[13] H.-H. Lee, S. Choi i M.-J. Lee, »Step Detection Robust against the Dynamics of Smartphones,« *Sensors,* svez. 15, br. 10, 2015.

[14] J. Leavitt, »How Much Deep, Light, and REM Sleep Do You Need?,« 18 Siječanj 2024. [Mrežno]. Available: https://www.healthline.com/health/how-much-deep-sleep-do-you-need?c=469581512162#stages-of-sleep. [Zadnji pristup 6 Lipanj 2025].

[15] B. Bei, S. Rajaratnam, S. Drummond i R. Manber, »Savjeti Američkog društva za istraživanje spavanja o spavanju tijekom razdoblja izolacije,« 2020. [Mrežno]. Available: https://www.imi.hr/wp-content/uploads/2020/04/Savjeti-za-spavanje-tijekom-razdoblja-izolacije_SRS_novo.pdf. [Zadnji pristup 6 Lipanj 2025].

[16] J. E. Hall i M. E. Hall, Guyton i Hall: Medicinska fiziologija, Medicinska naklada, 2022.

[17] M. Judaš i I. Kostović, Temelji neuroznanosti, Zagreb: MD, 1997.

[18] B. A. Fallow, T. Tarumi i H. Tanaka, »Influence of skin type and wavelength on light wave reflectance,« *Journal of Clinical Monitoring and Computing,* pp. 313-317, 2013.

SAŽETAK

Naslov rada: Praćenje osobnih aktivnosti uporabom dostupnih ugradbenih računala i senzora

U ovom radu izrađen je prototip nosivog uređaja za praćenje osobnih aktivnosti koristeći ESP32-S3 mikrokontroler i senzore za puls, temperaturu i pokret. Sustav se sastoji od fizičkog uređaja, web aplikacije izrađene u Reactu i poslužitelja temeljenog na Spring Bootu. Uređaj podržava više načina rada radi optimizacije potrošnje energije, a podaci se prenose lokalno putem Wi-Fi mreže. Funkcionalnosti uključuju mjerenje pulsa i temperature na zahtjev, brojanje koraka i analizu spavanja.

Ključne riječi: ESP32, nosivi uređaji, praćenje aktivnosti, senzori, IoT

SUMMARY

Title: Monitoring personal activities with low-cost embedded computers and sensors

This thesis presents a wearable device prototype based on the ESP32-S3 microcontroller and sensors for tracking heart rate, temperature, and motion. The system includes a hardware unit, a React-based web app, and a Spring Boot backend. It supports multiple power modes to improve battery life and communicates locally via Wi-Fi. Features include on-demand pulse and temperature measurement, step counting, and sleep monitoring. Focus was placed on energy efficiency and system modularity.

Keywords: ESP32, wearable device, activity tracking, sensors, IoT