Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Klemen Škoda

Spremljanje fizičnega stanja oseb v domačem okolju

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Mojca Ciglarič

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V zadnjem času raste delež starajočih se prebivalcev, ki ne more več živeti popolnoma samostojno. Zasnujte prototip nosljive naprave, ki bo omogočala nadzor fizičnega stanja takšnih oseb in v primeru nenormalnih podatkov samodejno obvestila skrbnika. Preglejte različne vrste senzorjev in preučite njihovo primernost, upoštevajte tudi, da mora biti naprava relativno poceni. Izdelajte tudi aplikacijo, ki bo omogočala vizualno spremljanje podatkov in prilagodljivo nastavljanje alarmiranja. Utemeljite izbor tehnologij in arhitekture, končni izdelek pa kritično ovrednotite tako iz tehnološkega kot tudi uporabniškega vidika.

Zahvaljujem se mentorici izr. prof. dr. Mojci Ciglarič za pomoč in hitro odzivnost pri izdelavi diplomske naloge. Zahvaljujem se tudi svoji družini za potrpežljivost in podporo v času študija.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
IoT	Internet of Things	Internet stvari
TSDB	Time series database	Podatkovna baza za časovna zaporedja
SoC	System on chip	Sistem na čipu
ADC	Analog-to-digital converter	Analogno-digitalni pretvornik
GSR	Galvanic skin response	Galvanski odziv kože
RTC	Real time clock	Ura realnega časa
NTP	Network Time Protocol	Protokol omrežnega časa
TLS	Transport Layer Security	Varnost transportnega sloja
ACL	Access control list	Seznam za nadzor dostopa

Povzetek

Naslov: Spremljanje fizičnega stanja oseb v domačem okolju

Avtor: Klemen Škoda

V zadnjih letih so postale nosljive naprave vroča tema v tehnološki industriji. Uporaba teh naprav pa je bolj na strani mlajših uporabnikov. V diplomski nalogi bomo raziskali razloge, zakaj je temu tako, saj je spremljanje fizičnega stanja z vidika zagotavljanja boljšega in bolj proaktivnega zdravljenja ter lažje oskrbe na domu pomembno tudi pri starejših.
V okviru diplomske naloge je bila razvita rešitev spremljanja fizičnega stanja oseb na domu - s poudarkom na enostavnosti uporabe, nizki ceni ter skalabilnosti. Izdelan je bil prototip nosljive naprave ter strežniška storitev, v katero se naprava povezuje in pošilja podatke. Naprava je osnovana na SoC modulu ESP32, njena naloga pa je neprekinjeno spremljanje srčnega utripa in galvanskega odziva kože (čustveno vzburjenje), ki ga pošilja na strežnik. Naprava je povezljiva v strežniško storitev, ki lahko deluje v oblaku ali lokalno. Strežnik shranjuje, obdeluje in vizualizira prejete senzorske podatke in ob anomalijah želenim osebam ta dogodek sporoči.

Ključne besede: nosljiva naprava, fizično stanje, oskrba na domu.

Abstract

Title: Physical monitoring of people in home environment

Author: Klemen Škoda

In recent years, wearable technology has become a hot topic in the tech industry. The use of wearable devices is skewed more towards younger individuals. In the thesis, we will explore the reasons why this is so, since monitoring of physical condition is also important for the elderly in terms of providing better and more proactive treatment and better home care.
Within the thesis we will present the developed solution for monitoring physical condition of people in home environment with emphasis on the ease of use, low price and scalability. We created a wearable device prototype and a data collection service to which wearable device connects and sends data. wearable device is based on SoC module ESP32, its task is to continuously monitor heart rate and galvanic skin response (emotional arousal). Gathered data is sent to the server. Wearable device is connectable to the data collection service that can be deployed in the cloud or locally. The server stores, processes and visualizes received sensor data and notifies caretakers when it detects anomalies in physical condition.

Keywords: wearable device, physical condition, home care.

Poglavje 1

Uvod

V zadnjih letih je prišlo do porasta razvoja naprav, kot so pametne ure, zapestnice za sledenje aktivnosti in podobne naprave, ki sodijo v skupino nosljivih naprav. Vse te naprave imajo skupno to, da nam poskušajo dati neko dodatno funkcionalnost oziroma želijo dati neko dodano vrednost našim napravam, kot je na primer spremljanje spanca in fizičnih aktivnosti. Opazili smo, da je stopnja uporabe takih naprav nižja pri starejših potrošnikih. Poiskali smo glavne razloge za to in jih upoštevali pri implementaciji našega sistema.

Izdelali smo prototip nosljive naprave za spremljanje fizičnega stanja, ki je povezljiva v oblačno ali lokalno storitev. Osredotočili smo se na merjenje utripa ter fiziološkega odziva osebe. Izdelali smo tudi strežniško storitev, ki pregleduje in vizualizira sprejete senzorske podatke in ob zaznanih anomalijah osebam, katerim želimo, to stanje sporoči.

Najprej bomo pregledali področji oskrbe na domu in nosljivih naprav in skušali najti glavne funkcionalnosti našega sistema. Nato bomo predstavili izbrano programsko in strojno opremo, čemur bo sledil opis posameznih delov rešitve. Šesto poglavje naloge pa se navezuje na težave, na katere smo naleteli pri implementaciji.

Poglavje 2

Pregled področja

2.1 Oskrba oseb v Sloveniji

V Sloveniji imamo za oskrbo oseb na voljo domove za starejše občane in tako imenovano pomoč na domu.

Domovi za starejše občane obsegajo osnovno oskrbo, ki vključuje bivanje, organizirano prehrano in socialno oskrbo, kar pomeni varstvo in zdravstveno varstvo. Osebe, ki lahko zaprosijo za sprejem v dom, morajo biti stare najmanj 65 let oziroma zaradi zdravstvenih težav, kroničnih bolezni ali drugih motenj potrebujejo oskrbo, ki nadomešča ali dopolnjuje funkcijo doma ali lastne družine [23]. Domovi ponujajo vse, kar potrebujejo starejše osebe, ki potrebujejo oskrbo ali celodnevni nadzor zaradi zdravstvenih težav. Pri domovih za starejše smo identificirali dve glavni težavi. Prva je zasedenost. V tabeli 2.1, vidimo da imamo skoraj enako število prošenj za nastanitev, kot je vseh mest v domovih v Sloveniji, prostih mest pa imamo zelo malo. Druga težava pa so visoki stroški oskrbe. Stroške vseh storitev krijejo stanovalci sami oziroma s pomočjo svojcev in občin, zaradi česar si marsikdo ne more privoščiti nastanitve v domu. Kot primer smo našli ceno nastanitve v domu Šiška, ki je v letu 2017 znašala 17,58 € na dan [20].

Druga možnost oskrbe je pomoč na domu. To je socialnovarstvena storitev, namenjena osebam, ki imajo zagotovljene bivalne pogoje v svojem okolju, vendar se zaradi starosti, bolezni ali hude invalidnosti ne morejo oskrbovati in negovati sami, njihovi svojci pa take oskrbe ne zmorejo ali znajo opravljati [24, 3]. Razlog za odločitev za oskrbo na domu je velikokrat tudi finančni. Ta vrsta oskrbe ne ponuja celodnevne oskrbe. Zaradi velikega števila oskrbovancev so le-ti v povprečju obiskani le 8 do 21-krat na mesec, ti obiski pa večinoma trajajo le 3 ure in pol (Tabela 2.2 in Tabela 2.3). Zaradi redkih obiskov in kratkega trajanja le-teh, pa oskrbovanec v primeru težav izven časa obiskov nima zagotovljene pomoči.

Kapaciteta	Prosta mesta	Evidentirane prošnje (aktivne)	Aktualne prošnje
20602	4	19583	7434

Tabela 2.1: Pregled prošenj in prostih mest v domovih za starejše in posebnih socialno varstvenih zavodih [19].

Obiski na mesec	Delež uporabnikov
7 obiskov ali manj	26,6%
8 do 21 obiskov	45,9%
22 obiskov in več	27,5%

Tabela 2.2: Število obiskov na mesec pri uporabnikih pomoči na domu [3].

Obseg	Število uporabnikov (stanje 31.12.2016)	
	2016	
Manj kot 3,5 ur na teden	4.434	60.1%
3,5 ure do pod 7 ur na teden	1.681	22.8%
7 ur na teden ali več	1258	17.1%
Skupaj	7373	100%

Tabela 2.3: Obseg pomoči na domu [3].

Leto	Število vseh uporabnikov	Uporabniki glede na starost)					
		0−64 let		64+ let		80+ let	
		N	%	N	%	N	%
2016	7374	746	10,1%	6628	89,9%	4725	64,1%

Tabela 2.4: Število uporabnikov pomoči na domu glede na starost [3].

2.2 Nosljive naprave

V zadnjih letih je prišlo do velikega porasta razvoja pametnih ur in zapestnic za sledenje aktivnosti. Tehnologije pa se razvijajo tudi na medicinskem področju, kjer se uporaba takih naprav uvaja v redno delo. Kljub velikemu porastu teh naprav so le-te namenjene mlajšim posameznikom. Vprašanje, ki se začne postavljati, je, zakaj na trgu ni več takšnih naprav za starostnike, saj jim lahko tovrstne naprave koristijo bolj kot mlajšim.

Eden od stereotipov je, da so starejši ljudje "zaprti" do novih idej in napredka v tehnologiji. A če pogledamo v same začetke nosljivih naprav, so bile ravno starejše osebe prve, ki so sprejele to tehnologijo. Leta 1972 je bil s strani gerontologa Andrewa Dibnerja razvit tako imenovani "Lifeline call button". To je bila brezžična nosljiva naprava z gumbom, ki je ob pritisku poklical nujno službo. Naprava se je obdržala v uporabi vse do danes, vendar pod imenom "Philips Lifeline". Nosljive naprave pa so danes zmožne veliko več kot le obveščanja o potrebni pomoči ob pritisku na gumb. Starejšim osebam lahko pomagajo z enostavnejšim spremljanjem njihovega zdravja, kar pripomore k zgodnejšemu posredovanju in proaktivnem zdravljenju. Informacije o trenutnem stanju oskrbovanca pa bi veliko pomenile tudi negovalcem, saj bi imeli boljši pregled nad stanjem oskrbovancev in boljšo odzivnost, če bi šlo kaj narobe [2, 15].

V raziskavi, ki je zajemala ciljno skupino potrošnikov nad 50 let, je bila za šest tednov dana v uporabo nosljiva naprava v cenovnem rangu 150 $ in manj. Izbrane so bile osebe, ki so imele v svojem domu računalnik z dostopom do interneta, tablični računalnik ali pametni telefon. Izkazalo se je, da so bile uporabnikom najbolj koristne naprave za spremljanje aktivnosti in spanja. Te so jih motivirale k bolj zdravem vedenju. Večina udeležencev je med uporabo naletela na kar nekaj težav. Več kot 80 % udeležencev ni bilo zmožnih uporabljati naprave brez pomoči. Druge večje težave pa so bile tudi nenatančnost izmerjenih podatkov, pomanjkanje navodil za uporabo, nedelovanje naprav, izguba podatkov ter težave s sinhronizacijo [1].

Poglavje 3

Izhodišča in opis sistema

3.1 Izhodišča

Ob pregledu področja oskrbe oseb in nosljivih naprav smo razbrali glavne cilje, ki jih bomo v diplomski nalogi skušali doseči. Iz raziskave o mnenjih starejših uporabnikov o nosljivih napravah na trgu smo ugotovili, da so bile največje pomanjkljivosti nenatančnost meritev, težave s sinhronizacijo Bluetooth naprav s pametnim telefonom, težavnost uporabe aplikacije in izguba podatkov. Zato je pomembno, da naša naprava deluje samostojno, brez pomoči pametnega telefona, da je enostavna za uporabo ter da zagotavlja zanesljiv prenos podatkov. Velikokrat je težava oskrbe in spremljanja oseb finančnega razloga, zato je poglavitno, da je naš sistem sestavljen iz cenovno ugodnih komponent in ponuja celovito rešitev, ki ne bo v pomoč le osebam samim, pač pa tudi negovalcem oziroma njihovim bližnjim.

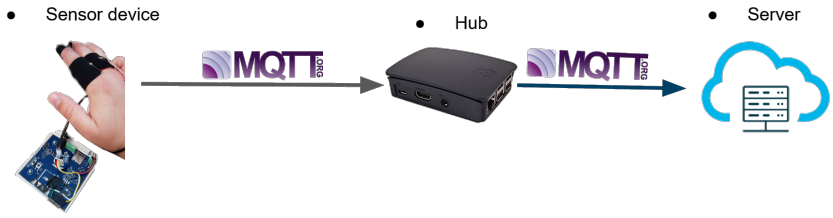
Zadani cilji sistema:

- nizka cena,
- enostavna uporaba,
- spremljanje osebe brez pomoči pametnega telefona,
- celovita rešitev,
- neprekinjeno spremljanje fizičnega stanja,
- zanesljiv prenos podatkov,
- točnost meritev,
- alarmiranje ob zaznanih anomalijah in
- razširljivost (enostavna dodajanja novih senzorjev).

3.2 Opis sistema

Pred implementacijo smo sistem razdelili na potrebne gradnike in jim določiti naloge, ki jih bodo opravljali. Sistem smo razdelili na dve možni arhitekturi. Prvo imamo v primeru lokalnega strežnika, katerega arhitektura je prikazana na sliki 3.1. V tem primeru lahko senzorska naprava komunicira direktno s strežnikom, saj je možnost izpada lokalnega sistema relativno majhna. Druga možna arhitektura pa je predstavljena na sliki 3.2 in uporablja strežnik v oblaku. V tem primeru je verjetnost izpadov večja. Za zagotavljanje zanesljive povezave smo v sistem dodali vozlišče, kot je prikazano na sliki 3.2.

Slika 3.1: Sistem v primeru lokalnega strežnika.



Slika 3.2: Sistem v primeru zunanjega strežnika.

3.2.1 Senzorska naprava

Senzorska naprava v našem sistemu skrbi za glavni del spremljanja fizičnega stanja oseb. Zadolžena je za zajemanje podatkov iz senzorjev in detekcijo srčnega utripa. Zajete podatke senzorja za utrip pretvori v utripe na sekundo. Ob zaznanem utripu naprava pošlje podatke vseh senzorjev proti strežniku.

- Zajemanje senzorskih podatkov.
- Obdelava podatkov senzorja za utrip.
- Pošiljanje podatkov na strežnik ob zaznanem utripu.

3.2.2 Vozlišče

Vozlišče zagotavlja boljšo zanesljivost sistema v primeru strežnika v oblaku. Njegova glavna naloga je zaznavanje prekinitve ali neuspešne internetne povezave. V primeru izpada shrani podatke v lokalni medpomnilnik, dokler povezava ni ponovno mogoča. Sistem je tako zaradi zmanjšane verjetnosti izgube podatkov bolj zanesljiv.

- Zaznavanje padca internetne povezave.
- Shranjevanje podatkov v primeru izpada povezave.
- Pošiljanje shranjenih podatkov ob ponovni vzpostavitvi povezave.
- MQTT posrednik.
- Most do posrednika na strežniku.

3.2.3 Strežnik

Strežnik je osrednji del sistema. Arhitekturo naše strežniške rešitve lahko razdelimo na posrednika sporočil, podatkovnega odjemalca, aplikacijo za vizualizacijo in podatkovno bazo. Posrednik sporočil je zadolžen za komunikacijo med senzorskimi napravami in odjemalci ter avtentikacijo uporabnikov. Podatkovni odjemalec posluša sporočila senzorskih naprav, le-te razčlenjuje v zapis, primeren za našo podatkovno bazo, ter jih vanjo zapisuje. Do shranjenih podatkov dostopamo preko aplikacije za vizualizacijo, ki poleg vizualizacije skrbi tudi za alarmiranje ob prednastavljenih dogodkih.

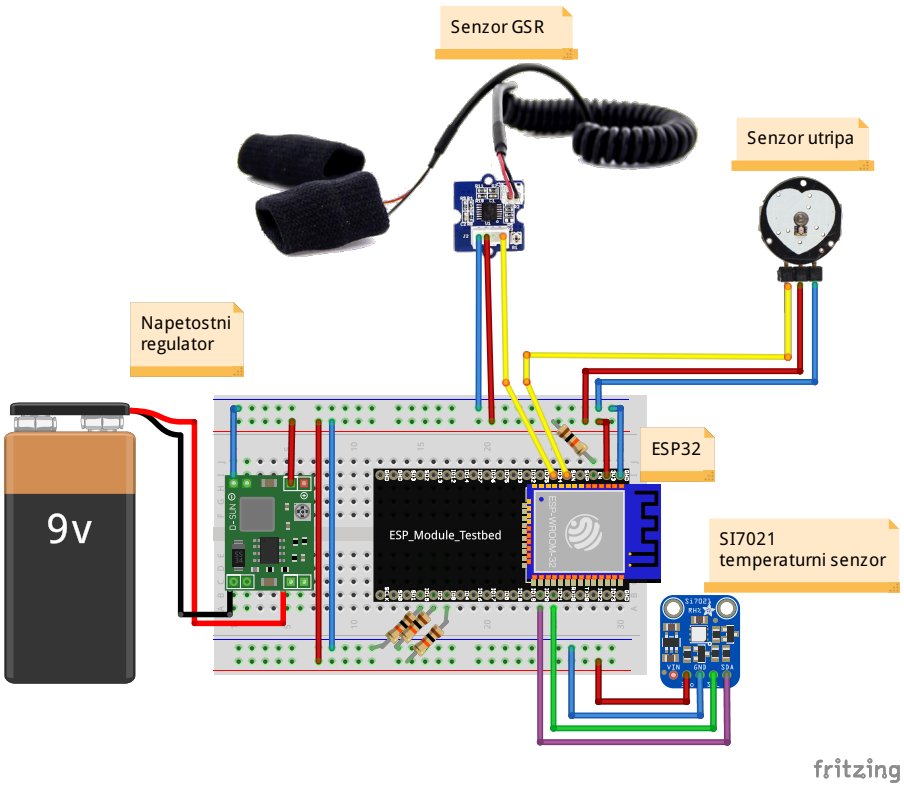
- Sprejemanje in shranjevanje podatkov.
- Obdelava podatkov.
- Vizualizacija podatkov.
- Alarmiranje ob pred nastavljenih dogodkih.
- Avtentikacija uporabnikov.

Poglavje 4

Izbira tehnologij

4.1 Senzorska naprava

Senzorska naprava bo v našem sistemu zadolžena za zbiranje podatkov o fizičnem stanju osebe. Naprava bo sestavljena iz brezžičnega modula, kateri bo zadolžen za branje senzorjev, analizo podatkov iz senzorjev in pošiljanje podatkov na strežnik, in senzorjev za spremljanje fizičnega stanja. Le-ti so senzor srčnega utripa, senzor galvanskega odziva kože in senzor temperature ter vlage kože.



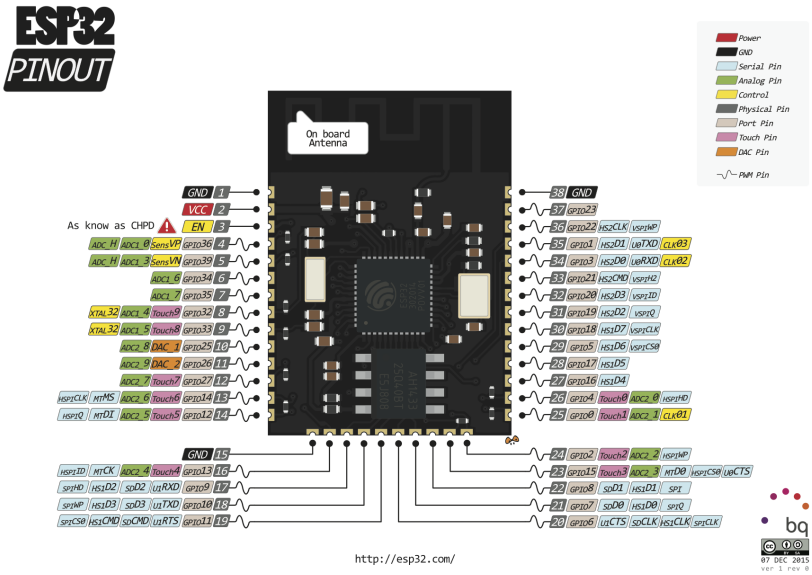
Slika 4.1: Shema senzorske naprave.

4.1.1 Brezžični modul

ESP32

ESP32 je SoC modul oziroma sistem na čipu, kar pomeni, da na enem samem čipu integrira vse potrebne komponente za njegovo delovanje in uporabo. Vključuje procesor, pomnilnik ter vhode za digitalne in analogne signale, ki so potrebni za priklop senzorike na modul. Zaradi visoke stopnje integriranosti imajo SoC moduli zelo nizko porabo energije. Modul ESP32 je nadgradnja modula ESP8266 podjetja Espressif, ki je zaradi svoje cene in vsestranskosti široko razširjen WiFi modul v IoT svetu. Glede na predhodnika pa naš modul ponuja večje število ter boljšo natančnost vhodov in

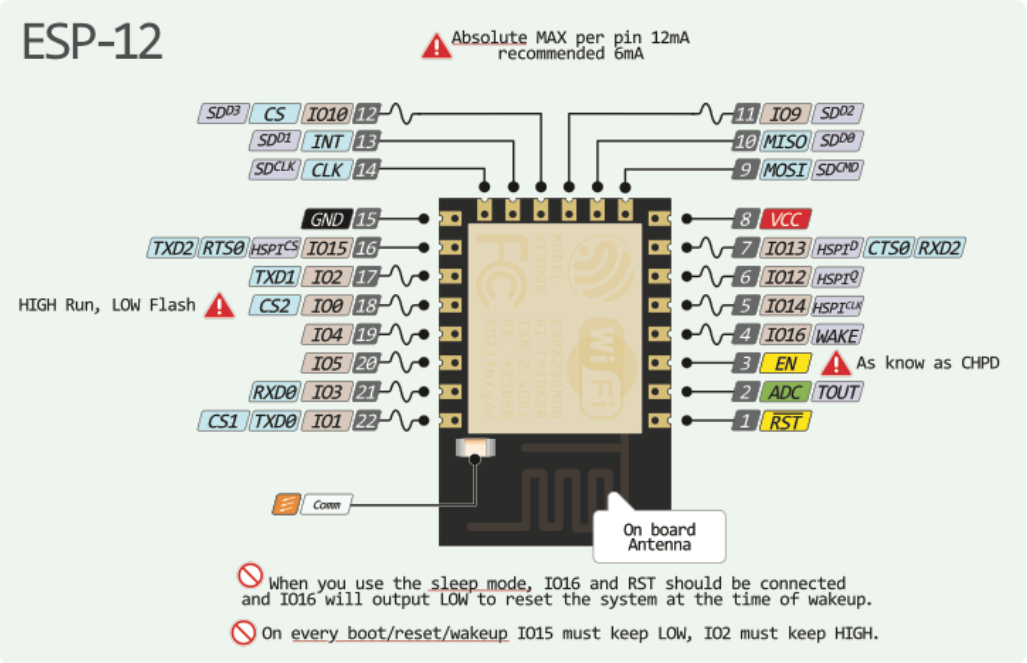
izhodov ter omogoča brezžično povezavo tako preko protokola WiFi kot tudi preko protokola Bluetooth 4.2. Za našo uporabo je zaradi možnosti priklopa večjega števila analognih vhodov ter izboljšane natančnosti le-teh v primerjavi s predhodnikom primernejša izbira.



Slika 4.2: Modul ESP32 [10].

ESP8266 (ESP-12)

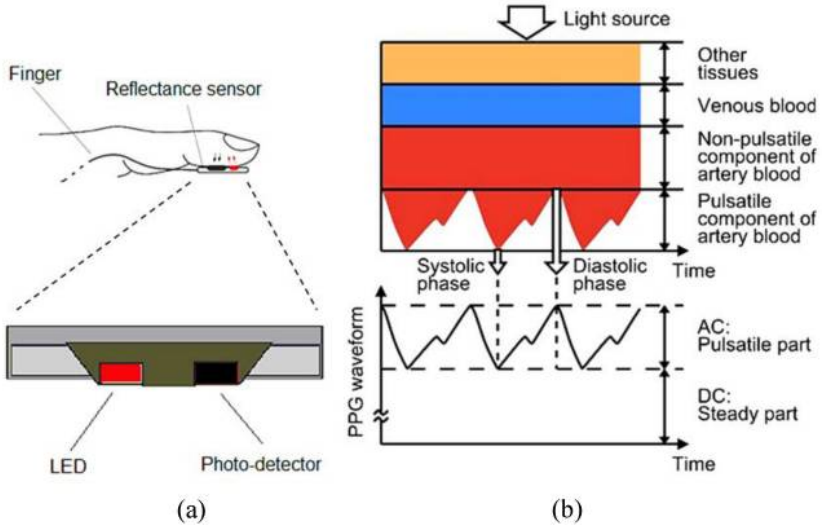
ESP8266 je izredno popularen modul podjetja Espressif v IoT svetu. Je najcenejši certificiran SoC WiFi modul na trgu. Največja pomanjkljivost, ki smo jo opazili pri njem, je bil omejen analogni vhod. Modul ima le en analogni vhod, ki ima vhodno napetost 0-1V. Zaradi tega smo morali signal iz senzorja za utrip peljati skozi napetostni delilnik da smo signal znižali iz območja 0-3.3V na 0-1V. To je privedlo k nenatančnosti meritev, ki jo bomo opisali v nadaljevanju.



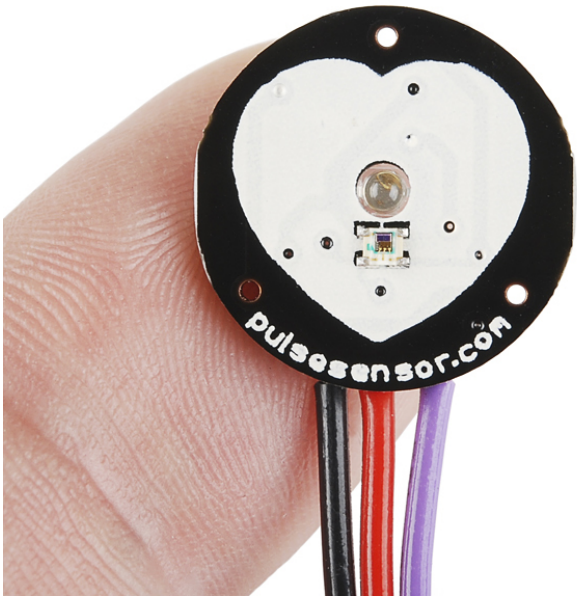
Slika 4.3: Modul ESP8266 [6].

4.1.2 Merjenje srčnega utripa

Za merjenje srčnega utripa smo se odločili za enostavno in cenovno ugodno rešitev, ki jo uporabljajo tudi pametne ure, kot je na primer Apple Watch. Merjenje se opravlja z zaznavanjem odboja svetlobe. Postopek merjenja je prikazan na sliki 4.4. Senzor zaznava količino odbite svetlobe, ki na dobro prekrvavljenih mestih, kot so na primer konice prstov, ušesne mečice ali notranja stran zapestja, sovpada s srčnim utripom. Senzor, ki smo ga uporabili, je tako imenovani "Pulse Sensor", slika 4.5. Vsebuje ojačevalnik signala in odstranjevanje šuma. Te funkcionalnosti senzorja omogočajo enostavnejše vezje in programsko logiko, saj je signal na analognem vhodu ESP32 že ojačan in filtriran za nadaljnjo procesiranje in obdelavo.



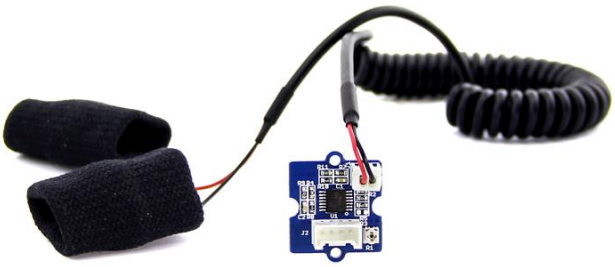
Slika 4.4: Prikaz delovanja odbojnega senzorja utripa [4].



Slika 4.5: Senzor srčnega utripa [13].

4.1.3 Merjenje galvanskega odziva kože(GSR)

Senzor galvanskega odziva kože oz. GSR senzor smo izbrali za zaznavanje čustvene vzburjenosti osebe. Meritev temelji na količini toka, ki lahko prehaja skozi naše telo. Senzor prikazuje, kako lahko (prevodnost) oziroma kako težko (upornost) tok prehaja skozi naše telo, kar temelji na Ohmovem zakonu [17]. Čustvena vznemirjenost se kaže v obliki globokega dihanja, stresa, mišične aktivnosti. Ti pojavi vplivajo na prevodnost kože, kar bomo v naši aplikaciji beležili. Skupaj s senzorjem utripa pa pridobimo dodatno informacijo o stanju uporabnika. V naši aplikaciji ga bomo uporabili za zaznavanje velikih sprememb v prevodnosti kože, ki bi lahko bile znak stresa ali kakšne druge čustvene vzburjenosti osebe.



Slika 4.6: Senzor GSR [21].

4.2 Strežnik

4.2.1 Podatkovna baza (InfluxDB)

Podatki, ki prihajajo iz senzorske naprave, so vedno časovno zvezni in sosledni. Za podatke take vrste so najbolj primerne baze za časovna zaporedja. Ena izmed priljubljenih odprto kodnih rešitev te vrste baz je InfluxDB.

InfluxDB [9] je podatkovna baza za časovna zaporedja oz. TSDB. Osnovana in optimizirana je za obdelovanja velike količine časovno zveznih podatkov, kot sta na primer realno časovna analitika in podatki IoT senzorjev. Za poizvedbe se uporablja jezik podoben SQL-u, kar omogoča enostavno poizvedovanje po agregiranih podatkih. Ponuja tudi možnost izvajanja avtomatičnih poizvedb, ki podatke agregirajo, s tem pa zmanjšajo potrebo po pogostih poizvedbah, proženje uporabniško določenih funkcij za procesiranje alarmov in podobno. Za naš sistem je zanimiva z vidika zapisovanja velike količine podatkov. Z njeno uporabo dosežemo visoko frekvenco zajemanja podatkov ter enostavno dodajanje in odvzemanje senzorjev s senzorskih naprav, pri čemer ni potrebna sprememba tabel, saj ponuja podobno fleksibilnost, kot jo najdemo v NoSQL bazah.

4.2.2 Vizualizacija (Grafana)

Za vizualizacijo podatkov smo skušali najti orodje, ki podpira izbrano podatkovno bazo iz prejšnjega poglavja, poleg tega pa omogoča enostavno uporabo in implementacijo alarmiranja nad podatki. Primerna rešitev je vizualizacijsko orodje Grafana.

Grafana [8] je odprto kodno orodje za metrično analitiko in vizualizacijo. Podpira veliko število podatkovnih baz, kot so InfluxDB, ElasticSearch, Graphite, OpenTSB, MySQL ter še mnoge druge. Orodje omogoča enostavno upravljanje z uporabniki, katerim lahko omejujemo dostop do organizacij, določamo njihove naloge (ali lahko urejajo poizvedbe, spreminjajo postavitev pod strani in podobno). Omogoča nam enostavno in hitro pot do kreiranja vizualizacije za uporabnika, saj lahko z nekaj kliki pridemo do želenega prikaza podatkov. Nad podatki imamo možnost vzpostavitve alarmiranja, s katerim lahko ob preseženih vrednostih določenim uporabnikom pošljemo e-mail ali SMS.

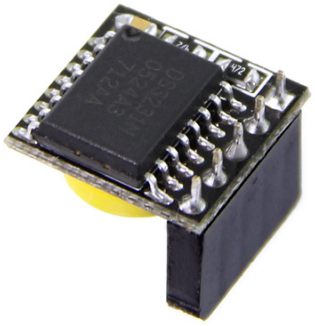
4.2.3 Strežnik (Raspberry Pi 3)

Kot strežnik našega sistema je bil uporabljen Raspberry Pi 3, saj zaradi izbranih tehnologij ne potrebujemo zmogljivega strežnika. Zaradi želje po razširljivosti sistema smo izbrali verzijo 3, ki ima zaradi nadgradnje centralne procesne enote izboljšano učinkovitost delovanja, za nas najbolj pomembna pa je dodana brezžična povezljivost. Modul vsebuje v tej verziji podporo za WiFi in Bluetooth 4.1 protokol, kar nam omogoča Raspberry Pi uporabljati kot dostopno točko brez dodatne opreme - v primeru, da na lokaciji, kjer želimo spremljati stanje osebe, nimamo dostopa do brezžične dostopne točke - in ponuja povezljivost naprav preko Bluetooth-a.

Slika 4.7: Raspberry Pi 3 [7].

4.2.4 Ura realnega časa DS3231(RTC)

Ker Raspberry Pi nima svoje ure realnega časa, smo uporabili RTC DS321. Izbrali smo ga zaradi nizke cene, polnilne baterije in enostavne implementacije. RTC potrebujemo v primeru težav z internetom, saj takrat Raspberry Pi ob vključitvi ne pridobi ure časovnega (NTP) strežnika. V tem primeru Raspberry Pi privzame uro, ki jo ima shranjeno od zadnjega izklopa. Če bi prišlo do te težave, podatki ne bi bili shranjeni v bazi s pravo časovno značko. Z RTC modulom pa se ta težava odpravi, saj le-ta šteje čas, tudi ko je Raspberry Pi izključen.



Slika 4.8: Real time clock DS3231 modul [22].

4.3 Brezžična tehnologija in komunikacijski protokol

4.3.1 Brezžična tehnologija

Ena izmed glavnih odločitev je bila izbira brezžične tehnologije senzorske naprave. Odločali smo se med tehnologijo Bluetooth ter WiFi.

V večini aplikacij nosljivih naprav na trgu je trenutno uporabljen Bluetooth, predvsem zaradi nizke porabe energije. Pri uporabi smo opazili, da je eden od največjih problemov nosljivih naprav enostavnost uporabe. Tehnologija Bluetooth v primerjavi z WiFi potrebuje pametni telefon in sinhronizacijo z njim, ali pa večje število sprejemnikov zaradi slabše prodornosti signala. V primeru WiFi omrežja ima večina uporabnikov dostopno točko že doma. WiFi v večini primerov nudi dovolj dobro pokritost, zato ne potrebujemo dodatnih sprejemnikov za povečanje dosega. V našem primeru, ko želimo uporabo čim bolj poenostaviti, smo se tako odločili za tehnologijo WiFi.

4.3.2 Komunikacijski protokol

Pri izbiri komunikacijskega protokola je bila odločitev nekoliko lažja. Zaradi tehnologije WiFi za senzorsko napravo potrebujemo protokol, ki za svoje delovanje ne porabi veliko sistemskih sredstev, saj bo implementiran na njej. Eden izmed trenutno najbolj priljubljenih protokolov je MQTT. Za razliko od drugih lahkih protokolov ima podporo pri največjih ponudnikih oblačnih storitev, kot so Amazonov AWS IoT, Microsoftov Azure, Google Cloud Platform in IBMov Watson IoT. V primeru ponujanja našega sistema kot oblačne storitve je pomembna kompatibilnost, zato je bil MQTT očitna izbira.

Posrednik MQTT (Mosquitto)

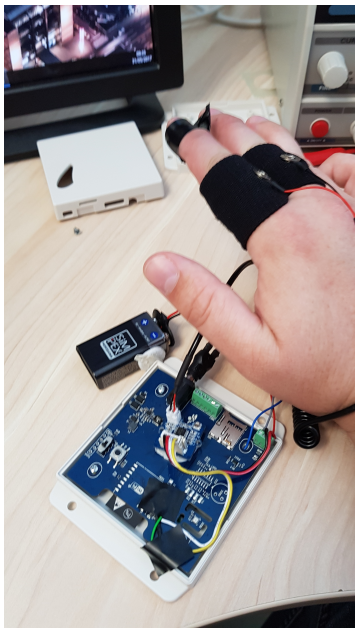
Mosquitto MQTT [5] je odprtokodni MQTT posrednik, ki implementira MQTT protokol verzije 3.1 in 3.1.1. MQTT oz. Message Queue Telemetry Transport je publish-subscribe "lahki" sporočilni protokol. Protokol ustreza napravam z majhno količino sistemskih sredstev in na lokacijah s slabo oz. omejeno internetno povezavo. Omogoča tudi veliko prepustnost podatkov v primerjavi z HTTP protokolom. V testiranjih smo prišli tudi do nekaj tisoč sporočil na sekundo, kar omogoča nadaljno skalabilnost sistema. Za delovanje uporablja posrednika, ki skrbi za dostavo sporočil odjemalcem, ki so nanje naročeni. Odjemalec se na želena sporočila naroči preko teme. Preko te teme posrednik ve, komu mora sporočilo posredovati. Zadolžen je tudi za avtentikacijo uporabnikov. Avtentikacija uporabnikov je narejena na več načinov. Najbolj odprt način zahteva samo enolični identifikator za vsakega odjemalca. Naslednja stopnja je avtentikacija z uporabniškim imenom in geslom za večjo varnost. MQTT podpira tudi TLS verzije 1.2. Odjemalcem je mogoče omejiti dostop tudi preko seznama za nadzor dostopa. S seznamom uporabnikom omejimo naročanje ter pošiljanje sporočil na želene teme.

Poglavje 5

Opis rešitve

5.1 Senzorska naprava

Glavni cilj pri implementaciji senzorske naprave je bila enostavnost uporabe. Ob zagonu se naprava poveže na privzeto WiFi dostopno točko. Po vzpostavljeni povezavi z dostopno točko naprava vzpostavi povezavo na strežnik preko MQTT protokola ter začne z zajemanjem analognega signala senzorja utripa. Ta je poslan v funkcijo, zadolženo za obdelavo signala utripa. Ob zaznanem utripu zajamemo še stanje senzorja galvanskega odziva kože ter obe vrednosti pošljemo proti strežniku.



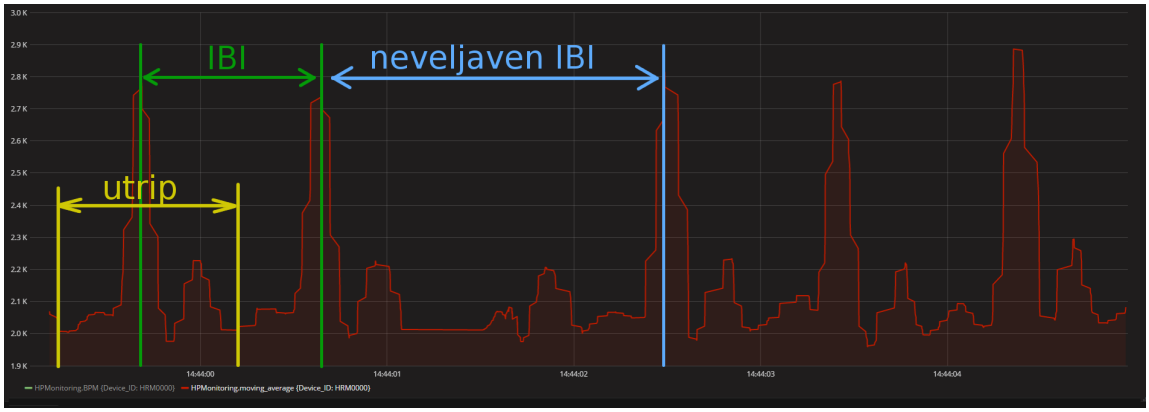
Slika 5.1: Prototip senzorske naprave.

Implementacija senzorja utripa

Za implementacijo senzorja utripa smo se odločili za izdelavo lastne kode obrazložitev za to odločitev pa se nahaja v poglavjih 6.1.2 in 6.2.1.

Pri meritvi utripa je za nas pomemben čas med utripi - imenovan IBI(Inter Beat Interval). Kot je razvidno iz slike 5.2, imamo ob vsakem utripu dva vrhova. Za nas je pomemben višji izmed njiju. Za zaznavanje teh vrhov smo potrebovali nekaj omejitev. Postavili smo si spodnjo in zgornjo mejo, med katerima smo iskali vrh. Meje predstavljajo vrednosti, med katerimi so za nas vrhovi veljavni. Ob testiranju smo opazili, da signal skozi čas niha, vendar je to nihanje počasno. To pomeni, da se vsak zaporedni signal ne spremeni nenadoma, ampak počasi preide v drugo območje. Zaradi tega smo si določili, da mora vsak zaporedno zaznani vrh biti v $\pm 15\%$ prejšnje zaznanega veljavnega vrha, s tem smo z veliko gotovostjo izločili šume signala. Pomembno je tudi, da čas IBI beležimo samo med dvema veljavnima vrhovoma, drugače bi le-ta negativno vplival na natančnost naših meritev. Kot veljavne vrhove

imenujemo vse vrhove, ki se nahajajo znotraj zgoraj omenjenih omejitev. Za izračun utripa smo uporabili zadnjih deset veljavnih meritev IBI, ki smo jih shranjevali v krožni seznam. Ob vsakem veljavnem IBI času smo izračunali utrip z enačbo "$BPM = \frac{60000*10}{IBI1+IBI2+\cdots+IBI10}$". Kot je razvidno iz enačbe smo za izračun vzeli povprečje vseh desetih veljavnih meritev IBI, s čimer smo zagotovili manjše nihanje izračunanega utripa. Ker je vrednost IBI merjena v milisekundah, je v števcu ulomka minuta pretvorjena v milisekunde.



Slika 5.2: Signal utripa in čas IBI.

Implementacija GSR senzorja

Implementacija senzorja je zelo enostavna. Ob vsakem zaznanem utripu izmerimo analogno vrednost senzorja. Le-to poleg ostalih podatkov pošljemo proti strežniku. Trenutno GSR vrednost uporabljamo zgolj kot indikacijo spremembe v prevodnosti kože, ko je utrip šel izven mejnih vrednosti. Indikacija, ki bi pomenila, da je bila oseba čustveno vzburjena, je velika in hitra sprememba v vrednosti signala GSR senzorja.

Iz podatkov bi lahko razbrali, kaj kakšna izmed sprememb v signalu pomeni, vendar bi za to potrebovali veliko večjo količino podatkov. Te podatke bi morali nato analizirati, da bi lahko določili, kaj kakšna sprememba pomeni.

5.2 Vozlišče

V primeru strežnika v oblaku lahko pride do izpada internetne povezave. Za rešitev tega problema lahko v sistem dodamo vozlišče. Vozlišče ima svojega MQTT posrednika, kamor se povezujejo naprave v lokalnem omrežju. Le-ta ima enake nastavitve kot strežnik.

5.2.1 Most

Da vse podatke prejme tudi strežnik, moramo med vozliščem in strežnikom vzpostaviti povezavo. Ta povezava se imenuje most. Most potrebuje za dostop do strežnika enake parametre kot odjemalec; to so unikatni identifikator, uporabniško ime in geslo ter certifikat za TLS. Na njem tudi določimo, kateri izmed podatkov se pošiljajo proti strežniku in katere od njega sprejemamo. Nastavitve, ki smo jih uporabili v našem sistemu so predstavljene v kodi 5.1.

```
connection bridge
#certificate used for TLS
bridge_cafile /etc/mosquitto/certs/ca.crt
try_private false
#server URI and port
address api.firefly-iot.com:8192
start_type automatic
#username and password
remote_username hpm
remote_password hpm9318
#unique idenfitier
remote_clientid bridge_hpm
#time between restart attempts
restart_timeout 5
#time between checks if connection is still up
keepalive_interval 60
#allow all topics in both ways
topic # both
```

Koda 5.1: Nastavitve za vzpostavitev mostu.

5.2.2 Zaznavanje padca povezave

Na vozlišču teče skripta, ki zaznava, ali je most med MQTT posrednikoma vzpostavljen. Ko skripta vzpostavi povezavo na lokalni MQTT posrednik, se sproži funkcija on_connect(), v kateri se naroči na dve temi. Kot prvo "$SYS/broker/connection/#", preko katere dobi obvestilo "1", če je povezava vzpostavljena, in "0", če povezave ni. Druga je tema "data/#", na katero prihajajo vsi lokalni senzorski podatki. Ob prihodu podatka iz katerekoli od naročenih tem, se sproži funkcija on_message(). V primeru prihoda podatka o spremembi stanja mostu osvežimo globalno boolean spremenljivko "bridge". Če dobimo sporočilo, da je most vzpostavljen, pokličemo funkcijo read_buffer(). Funkcija read_buffer(), v primeru, da so v datoteki "buffer.csv" podatki, te pošlje na strežnik in medpomnilnik izprazni. V primeru senzorskega podatka pa se preveri, ali je most vzpostavljen. Če most ni vzpostavljen, se podatku doda časovna značka in pokliče funkcija write_to_buffer(). Ta funkcija podatek - skupaj s temo, na katero je bil poslan - zapiše na konec datoteke "buffer.csv", če ta ni večja kot 20MiB. Omejitev datoteke v našem sistemu zadošča za 60 ur podatkov iz enega senzorja.

```
def on_message(client, userdata, msg):
    global bridge
    if(msg.topic[:4] == "$SYS"): #check if bridge is active
        if str(msg.payload) == '0':
            bridge = False
        elif str(msg.payload) == '1': #if bridge is active check buffer
            bridge = True
            read_buffer()
    elif(msg.topic[:4] == "data"):
        if bridge == False: #if bridge is not active
            #save incoming data into a buffer
            payload = json.dumps(msg.payload)
            payload['timestamp'] = datetime.datetime.utcnow().isoformat()
            write_to_buffer(str(payload), msg.topic)

def write_to_buffer(data, topic):
    #fill buffer if it's smaller than 20MiB
    write = open('/usr/bin/app/buffer.csv', 'a')
    if write.tell() < 20000000:
        write.write(data + '$' + topic + '\n')
        write.close()
    else:
        f.close()

def read_buffer(): #send all data from the buffer to the server
    try:
        read = open('/usr/bin/app/buffer.csv', 'r')
        for line in read:
            spline = line[:-1].split("$")
            print spline[0]
            client.publish(spline[1], spline[0])
    except:
        pass
```

Koda 5.2: Nastavitve za vzpostavitev mostu.

5.3 Strežnik

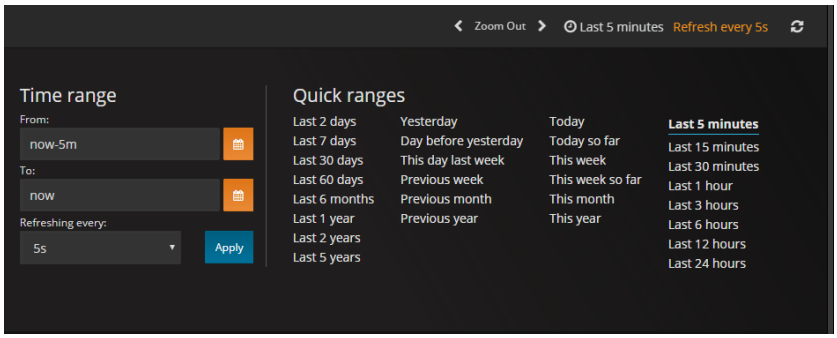
5.3.1 Uporabniški vmesnik (Grafana)

Za vizualizacijo senzorskih podatkov smo uporabili odprto kodno orodje Grafana. Samo orodje je zelo enostavno za uporabo, saj lahko samo z nekaj kliki dostopamo do podatkov z naših senzorjev. Za uporabnika smo postavili enostavno stran, na kateri lahko hitro vidi podatke iz svoje senzorske naprave.

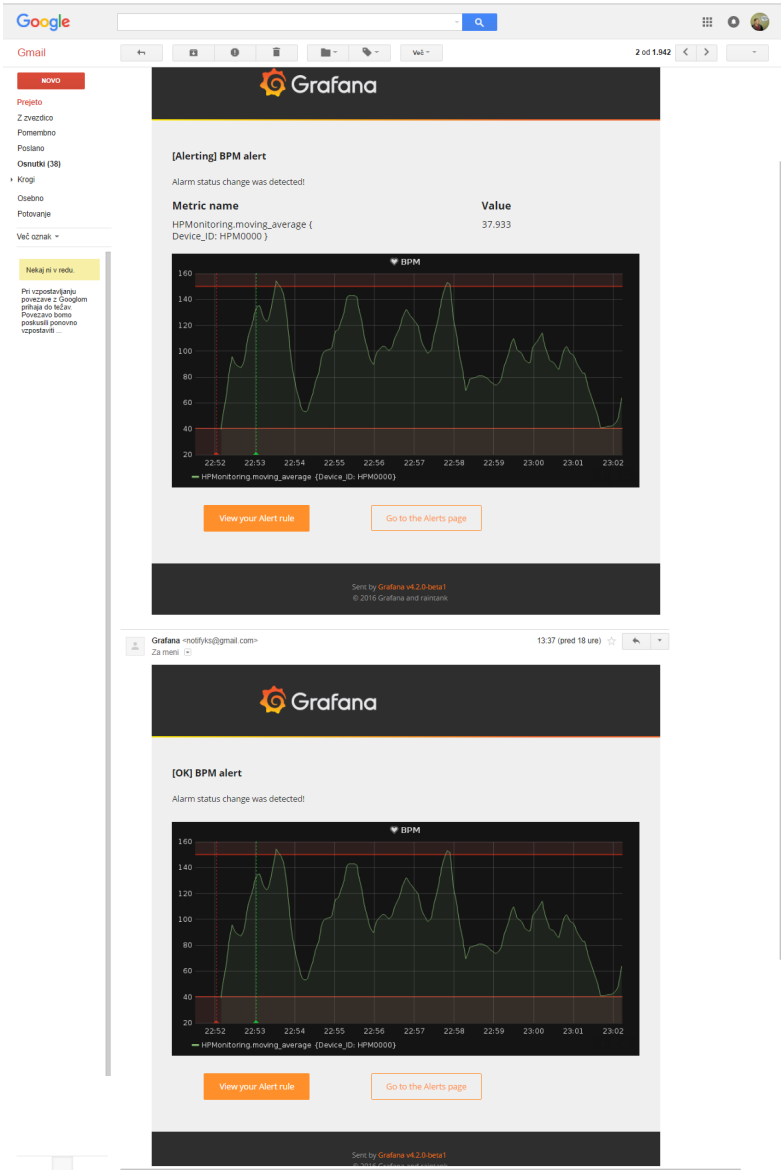
Kot je razvidno iz slike 5.3, imamo na vrhu merilnik, ki prikazuje zadnji prejeti podatek senzorja utripa. Pod njim pa se nahajajo grafi senzorjev za izbrano obdobje. Uporabnik s klikom v zgornji desni kot strani odpre meni, kjer lahko spreminja obdobje prikazanih podatkov, slika 5.4.

Na grafu senzorja za utrip smo dodali alarmiranje s spodnjo in zgornjo mejo. Meje so nastavljene za starostno skupino uporabnika, podatke zanjo pa smo pridobili iz raziskave o variabilnosti srčnega utripa glede na starost in spol [16]. V primeru prekoračitve ali izpada podatkov prejmejo določene osebe elektronsko pošto s tem dogodkom (primer prejete elektronske pošte lahko vidimo na sliki 5.5).

Slika 5.3: Vizualizacija podatkov v Grafani.



Slika 5.4: Izbira obdobja vizualiziranih podatkov.



Slika 5.5: Alarmiranje ob prednastavljenih dogodkih.

5.3.2 MQTT posrednik Mosquitto

Protokol MQTT je sam po sebi zelo odprt. V osnovni konfiguraciji, ki jo imamo po vzpostavitvi MQTT posrednika, je vse, kar uporabnik potrebuje za vzpostavitev povezave, unikatni identifikator, IP na katerem se posrednik nahaja, in port. Uporabnik lahko vse podatke bere in pošilja na katerokoli temo. Zato je bilo potrebno omogočiti nekaj dodatnih nastavitev - za večjo varnost uporabnikov ter za varovanje uporabniških podatkov.

Uporabniško ime in geslo

Kot prvo, smo za večjo varnost implementirali uporabniška imena in gesla. Na ta način omogočimo dostop do strežnika preko MQTT povezave samo napravam, ki imajo veljavno uporabniško ime in geslo.

Za implementacijo funkcionalnosti smo ustvarili novo datoteko z gesli in prvega uporabnika. To smo storili z ukazom mosquitto_passwd, ki je prikazan v kodi 5.3. Tu je pomemben parameter "-c", ki ustvari novo datoteko z gesli in prvim uporabnikom. V primeru, da datoteka s tem imenom že obstaja, le-to prepiše. Pri dodajanju dodatnih uporabnikov želimo ohraniti obstoječo datoteko, zato uporabimo parameter "-b", kot je to razvidno v kodi 5.4.

Po kreiranju datoteke moramo onemogočiti dostop do MQTT posrednika brez uporabniškega imena, kar naredimo z dodajenjem vrstice 2 v kodi 5.5. Dodati moramo tudi pot do datoteke z gesli, kot je to vidno v vrstici 5 v kodi 5.5.

```
1 mosquitto_passwd -c filename username
```

Koda 5.3: Ukaz za kreiranje nove datoteke z gesli[11].

```
1 mosquitto_passwd -b ime_datoteke username password
```

Koda 5.4: Ukaz za kreiranje novega uporabnika.

```
#Users can not connect to our server without a username and
    password
allow_anonymous false

# location of the file with all users. Passwords are encrypted.
password_file /etc/mosquitto/passwd
```

Koda 5.5: Onemogočenje povezovanja brez uporabniškega imena in gesla ter nastavitev lokacije datoteke z gesli.

Enkripcija TLS

Naslednji korak za večjo varnost je implementacija TLS enkripcije. Najprej smo z uporabo orodja OpenSSL ustvarili vse potrebne certifikate, z ukazi, navedenimi v kodi 5.6. Ustvarjene certifikate smo dodali v konfiguracijsko datoteko MQTT posrednika ter določili verzijo TLS enkripcije, ki jo želimo uporabiti, kot je prikazano v kodi 5.7. V naši implementaciji smo uporabili TLS verzije 1.2, ki je trenutno najvišja verzija ter podprta na vseh nivojih našega sistema.

```
Generate a certificate authority certificate and key.
openssl req -new -x509 -days <duration> -extensions v3_ca -
    keyout ca.key -out ca.crt

Generate a server key.
openssl genrsa -des3 -out server.key 2048

Generate a certificate signing request to send to the CA.
openssl req -out server.csr -key server.key -new

Sign CSR with your CA key.
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -
    CAcreateserial -out server.crt -days <duration>
```

Koda 5.6: Ukazi za generiranje certifikatov[12].

```
#Here we define locations of generated certificates for TLS.
cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/server.crt
keyfile /etc/mosquitto/certs/server.key

#TLS version we use is 1.2 .
tls_version tlsv1.2
```

Koda 5.7: Nastavitev lokacije certifikatov ter verzije TLS.

Seznam za nadzor dostopa (access control list)

Za varovanje uporabniških podatkov smo omejili pravice uporabnikom. Ustvarili smo datoteko "aclfile.conf", ki vsebuje omejitve dostopa za vse uporabnike v sistemu. V našem sistemu smo imeli dva. Prvi z imenom "hpm" je bil uporabljen s strani senzorske naprave in ima dovoljeno samo pisanje na temo "data". Preko te teme smo pošiljali senzorske podatke proti strežniku. Drugi uporabnik z imenom "hpm_server" pa je bil uporabljen samo na strežniku. Ima pravico branja senzorskih podatkov, ki prihajajo preko teme "data". Nastavitve so prikazane v kodi 5.8. Pot do datoteke smo dodali v konfiguracijsko datoteko MQTT posrednika, koda 5.9. S tem smo dosegli, da nihče razen uporabnika, ki je v uporabi na strežniku, ne more brati senzorskih podatkov. S tem smo zagotovili, da uporabniki ne morejo poslušati prometa med seboj in preprečili poslušanje podatkov v primeru vdora.

```
user hpm
topic write data/#

user hpm_server
topic read data/#
```

Koda 5.8: Omejitev dostopa uporabnika hpm.

```
1 #File contains rules for each user. These rules define what the
    user can do on the broker. What subscriptions they can make
    and what topics can they publish to.
2 acl_file /etc/mosquitto/aclfile
```

Koda 5.9: Nastavitev lokacije certifikatov datoteke s seznamom za nadzor dostopa.

5.3.3 Shranjevanje podatkov (InfluxDB)

Na strežniku teče skripta, ki skrbi za shranjevanje podatkov v bazo. Po začetni inicializaciji povezave na MQTT posrednik se ob vsakem prispelem podatku na temo "data/#", na katero smo naročeni, sproži funkcija on_message(). Prispeli podatek s klicem funkcije "json.loads(msg.topic)" dekodiramo v tabelo, katere elemente lahko naslavljamo.

Podatke iz te tabele preoblikujemo v obliko, ki je primerna za bazo Influx. Doda se tudi časovna značka, v primeru, da le-ta ni prišla s senzorskim podatkom. V primeru, da podatek pri zapisu v bazo časovne značke nima, jo influxdb knjižnica doda sama - pred zapisom v bazo. Za zapis v bazo uporabljamo funkcijo write_points(). Ta funkcija v bazo zapiše seznam točk. Sprva je bila implementacija narejena na način, da smo vsako točko posebej zapisali v bazo, vendar se je ob dodatnih testih, kjer je bila količina podatkov velika, to izkazalo za prepočasno, saj smo bili omejeni na le okoli 16 točk na sekundo, kar pri skaliranju uporabnikov ter večji frekvenci zajemanja podatkov ni sprejemljivo. Po testiranju več opcij implementacije smo izbrali način z zapisovanjem seznama točk. Saj je hitrost zapisovanja ene točke s funkcijo write_points() skoraj enaka kot pri zapisovanju množice točk, kot je to razvidno iz slike 5.6.

Podatke po preoblikovanju shranjujemo v seznam. Pri vsakem prejetem podatku preverjamo, ali je od zadnjega zapisa bilo že več kot pol sekunde; če je to res, seznam zapišemo v bazo, ga izpraznimo in ponastavimo časovnik. Tu pa imamo težavo. Funkcija on_message() se sproži samo ob novem pre-

Čas pisanja enega podatka	Čas pisanja 100 podatkov	Čas pisanja seznama 100 točk
0.0600382971764	6.040320158	0.0998501777649
0.0608256173134	6.11775612831.0	0.0530087947845
0.0610306882858	6.13981294632	0.0943048000336
0.0613311886787	6.17115998268	0.0667631626129
0.0630457615852	6.34228110313	0.0679502487183
0.0607687973976	6.11626505852	0.0715427398682
0.0623398113251	6.27373290062	0.0855429172516
0.0616766023636	6.20601820946	0.0555663108826
0.0642437958717	6.46115994453	0.0579309463501
0.0631590723991	6.35363316536	0.0572154521942

Slika 5.6: Testiranje hitrosti zapisovanja funkcije write_points() (rezultati so merjeni v sekundah).

jetem sporočilu preko MQTT protokola. Pridemo lahko v stanje, kjer nekaj časa ne bo novega podatka, kar lahko privede do tega, da podatki ostanejo v seznamu.

To rešimo s funkcijo looping(), ki teče v svoji niti. Ta funkcija vsako sekundo preverja, ali so v seznamu kakšni podatki. Če so, le-te shrani v bazo. Po opravljenih testih se je izkazalo, da se zapisovanje v bazo, ki se vrši v on_message() funkciji, izvede hitreje kot v posebni niti. Razlog je, da ne glede na to, da se zapis kliče v posebni niti, mora funkcija on_message() vseeno čakati na zaključitev zapisa, preden polni seznam, da ne pride do izgube podatkov, ker se seznam po zapisu izprazni. S tem načinom zapisovanja smo dosegli hitrost okoli 1000 sporočil na sekundo.

```
#Callback when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    try:
        payload = json.dumps(msg.payload)
        #data structure for influxdb
        outdata = {}
        outdata['measurement'] = measurement
        #here are tags by which we will search for data and also group data by
        outdata['tags'] = {}
        outdata['tags']['deviceID'] = payload['ID']
        #here we store sensor data from incoming messages
        outdata['fields'] = {}
        outdata['fields']['BPM'] = payload['BPM']
        outdata['fields']['GSR'] = payload['GSR']
        #add timestamp to data
        try:
            outdata['time'] =  payload['timestamp']
        except:
            outdata['time'] = str(datetime.datetime.utcnow().isoformat())
        #append data to a list
        global data
        data.append(outdata)
        time1 = time.time()
        global time_overall
        #store data only every half a second, this is because write_points function is slow
        if (time1-time_overall) > 0.5:
            db.write_points(data)
            data = []
            time_overall = time1
    except Exception as e:
        print e
```

Koda 5.10: Glavna funkcija za shranjevanje podatkov.

```
def looping(threadtmp):
    #check every second if there is any data left in the list if
     so save it and empty the list
    while threadtmp.end == False:
        time.sleep(1)
        global time_overall
        timeatm =  time.time()
        global data
        if (timeatm - time_overall) > 1 and len(data) != 0:
            print len(data)
            threadtmp.db.write_points(data)
            data = []
            time_overall = timeatm
```

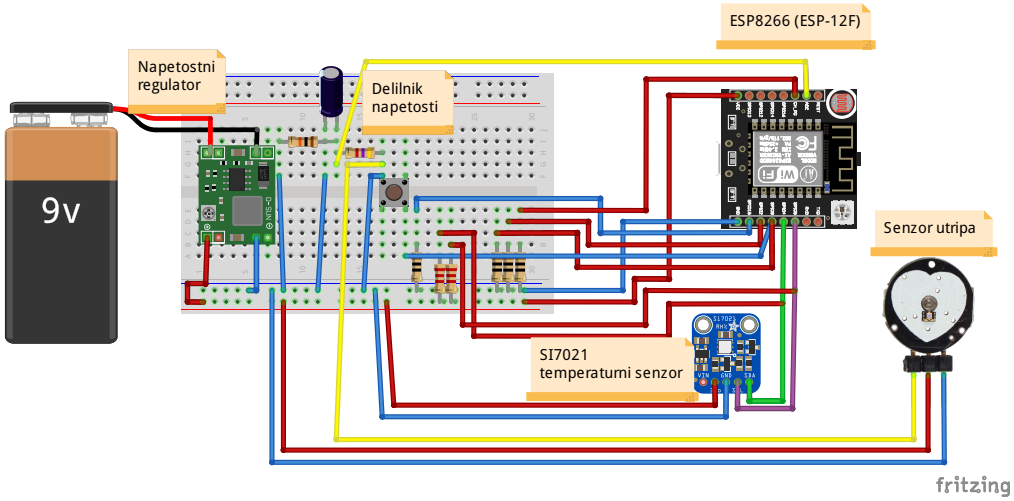
Koda 5.11: Funkcija, ki skrbi za zanesljivo praznenje seznama točk.

Poglavje 6

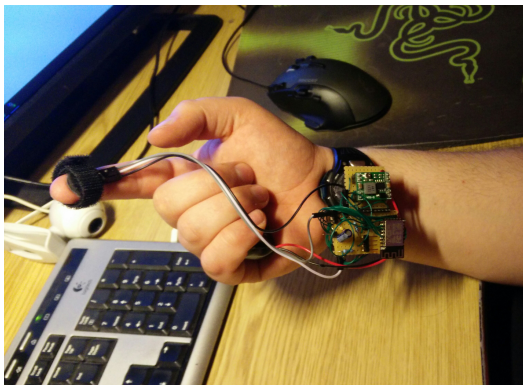
Testiranje in težave

6.1 Prva različica prototipa

Pri prvem prototipu, viden na sliki 6.2, smo se odločili za zajemanje telesne temperature, vlage in srčnega utripa. Izbira senzorjev in želja po čim manjši ceni naprave nas je pripeljala do modula ESP8266 na sliki 4.3, in senzorjev, ki so prikazani na sliki 6.1.



Slika 6.1: Shema prve različice prototipa.



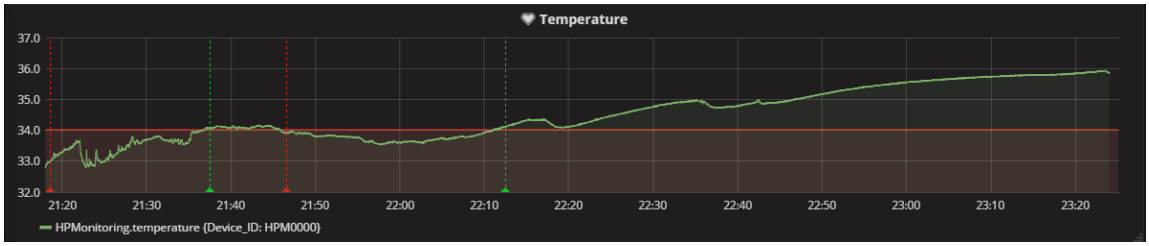
Slika 6.2: Prvi prototip z merjenjem utripa, temperature in vlage z modulom ESP8266.

6.1.1 Testiranje in implementacija merjenja temperature in vlage

Prvotna želja je bila spremljanje telesne temperature in vlage na zapestju osebe. Za ta namen smo se odločili za senzor SI7021. Senzor ima obratovalno napetost $1.9 - 3.6V$ in nizko porabo. Temperaturna natančnost senzorja je reda ± 0.4°C v območju -10 do 85 °C. Natančnost vlage pa reda $\pm 3\%$ v območju 0-80 % Rh. Kar nam za merjenje telesne temperature in vlage zadošča.

Za implementacijo senzorja smo uporabili knjižnico SI7021 LowPowerLab-a [18]. Knjižnica omogoča enostavno branje senzorja. Potrebujemo le inicializacijo objekta SI7021, s klicem funkcije begin(), ki inicializira senzor, nato pa že lahko beremo temperature in vlago s senzorja. Za našo implementacijo smo uporabili funkcijo getHumidityAndTemperature(), ki vrne tako vlago kot tudi temperaturo senzorja.

Za mesto merjenja temperature smo se odločili za zapestje, saj tam dobimo najboljši kontakt s kožo. Po nekaj opravljenih testih smo opazili, da je izmerjena temperatura okoli štiri do pet stopinj nižja od realne temperature. Vpliv na izmerjeno temperaturo je imela tudi temperatura okolja, v katerem smo opravljali merjenje. Blizu realne temperature smo prišli le v primeru, ko smo testiranje opravljali med spanjem, saj je odeja zadrževala toploto, zaradi česar je bila tudi temperatura na koži veliko bližje realni (ta meritev je prikazana na sliki 6.3 od 22:20 naprej). Opravili smo tudi meritve temperature z laserskim merilnikom toplote (prikazan na sliki 6.4), ki je pokazal še nižje temperature kot naš senzor. Predvidevamo, da je do razlike v temperaturah prišlo zaradi načina namestitve senzorja za temperaturo, za kar smo uporabili pas zapestne ure.

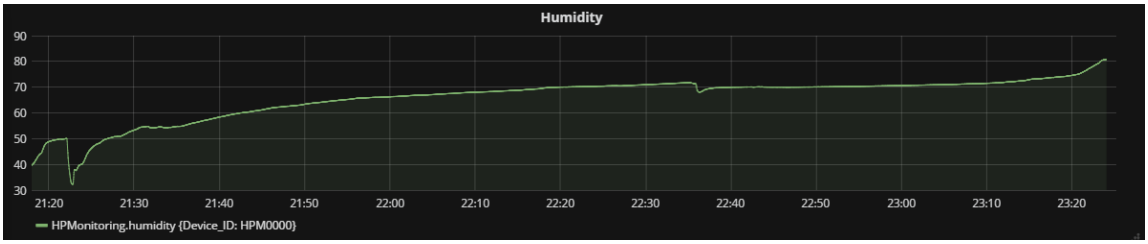


Slika 6.3: Testiranje merjenja temperature pred in med spanjem.

Merjenje vlage smo opravljali na istem mestu, saj izbrani senzor omogoča meritev obeh merilnih količin. Pri opravljanju meritev smo opazili, da je bila v veliko primerih vlaga zelo visoka, prikazano na sliki 6.5. Ugotovili smo, da zaradi načina namestitve senzorja prihaja na mestu meritve do potenja, kar nam je dalo nepravilne rezultate.

Ugotovitve testiranja temperature in vlage so bila, da zaenkrat opustimo to senzoriko in dodamo kakšen drug senzor, ki nam lahko pomaga pri spremljanju fizičnega stanja oseb. To je bil eden izmed ključnih razlogov za izdelavo druge različice prototipa.

Slika 6.4: Meritev temperature z laserskim merilnikom.



Slika 6.5: Testiranje merjenja vlage pred in med spanjem.

6.1.2 Testiranje in implementacija senzorja utripa

Za implementacijo senzorja utripa smo v prvi različici prototipa uporabili primer Yakira Malke[14].

S prvim testom smo skušali ugotoviti najboljšo lokacijo za merjenje utripa. Seveda bi si želeli narediti čimbolj kompaktno napravo, saj bi bilo to veliko boljše za končnega uporabnika. Testirali smo dve lokaciji - na prstu in na zapestju.

Pri testu na zapestju smo opazili, da so meritve veliko bolj nihale kot pa meritve na prstu. Ugotovili smo, da je težava v samem načinu merjenja utripa. Ker utrip merimo kot količino odbite svetlobe in s tem zaznavamo širjenje in krčenje tkiva ob utripu, pridemo do težave, da ob vsakem premiku prstov in zapestja generiramo signal, ki je močnejši od samega utripa. Tu imamo kar veliko težavo, saj lahko meritve izvajamo samo ob mirovanju, ki ga težko zaznamo brez dodatne senzorike. Kot potrditev te teorije smo opravili test z uro "Apple Watch", ki za merjenje utripa uporablja enako senzoriko. Za test je uporabnik hitro krčil in raztezal dlan. Meritev je pokazala, da tudi ura obravnava signal, ki ga ustvarja krčenje in raztezanje dlani, kot utrip. Izmerjeni utrip je bil 215, prikazano na sliki 6.6, kar je daleč od realne vrednosti, ki je bila v mirovanju okoli 75 utripov na minuto.

Pri testu na prstu smo imeli veliko bolj konsistentne meritve - tudi ob premikanju prstov in roke. Zaradi veliko manjšega vpliva premikanja na meritev smo se odločili, da bomo meritve utripa izvajali na prstu.

Ne glede na to, da smo imeli veliko boljše zaznavanje utripa na prstu kot zapestju, je še vedno prihajalo do nenadnega skoka v izmerjenem utripu za 50 do 100 utripov v minuti. Odločili smo se, da odkrijemo vzrok težav. Naredili smo testno kodo, ki je direktno izpisovala vrednost signala senzorja utripa. V kodi je bila izbrana fiksna meja utripa ravno na sredini resolucije ADC vhoda modula ESP8266, ki ima 10bitni ADC. Opazili smo dve težavi. Prva je bila, da smo na signalu imeli zelo veliko šuma. Kar smo skušali rešiti z dodajo 10μF kondenzatorja, preden signal pride do modula ESP8266. S tem smo hoteli doseči zmanjšanje šuma, vendar to ni pomagalo, saj je bil

Slika 6.6: Testiranje anomalije merjenja utripa z Apple Watch.

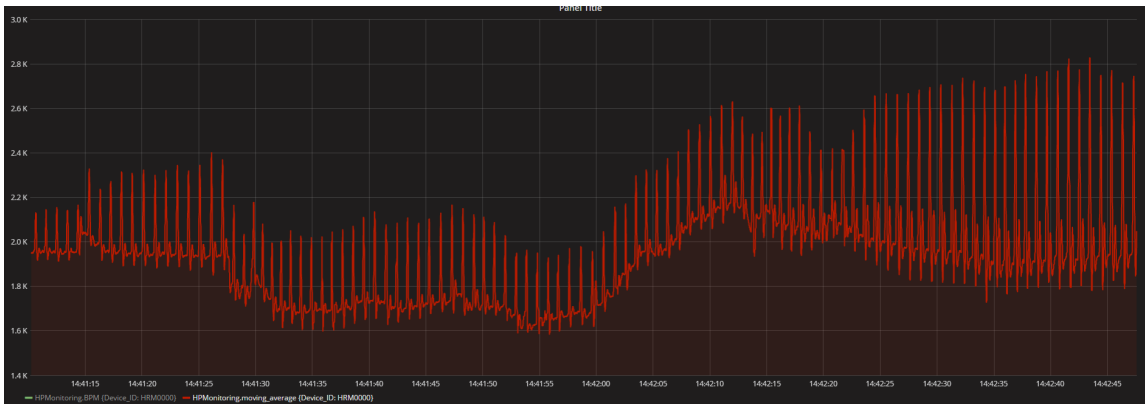
šum pri detekciji utripa še vedno moteč. Predvidevamo, da je k težavi s šumom doprineslo tudi to, da je razpon ADC vhoda samo 0-1V, zaradi česar smo morali narediti delilnik napetosti, saj je signal s senzorja utripa 0-3,3V. Kar lahko doprinese k samem šumu. Kot drugo pa, da signal utripa ni bil vedno na sredini resolucije ADC vhoda, ki je 512, ampak se je gibal med 300 in 600. Iz teh testiranj smo ugotovili, da imamo težavo s signalom in s samo implementacijo kode, ki smo jo uporabili za zaznavanje utripa. To je bil dodaten razlog za izdelavo druge različice prototipa.

6.2 Druga različica prototipa

Poglavitna razloga za izdelavo drugega prototipa sta bila težava s šumom pri zajemu signala senzorja utripa in želja po dodatnih analognih senzorjih. Po pregledu možnosti je bila najboljša izbira modul ESP32, ki je nadgradnja modula, uporabljenega pri prvem prototipu. Le-ta vsebuje tako več analognih vhodov za senzorje kot tudi večjo resolucijo signala.

6.2.1 Testiranje senzorja utripa

Pri prvem prototipu smo opazili veliko šuma pri zajemu analognega signala. Pri drugi verziji zaradi nadgradnje ADC vhodov na modulu ESP32 nismo potrebovali delilnika napetosti, saj je razpon vhodne napetosti enak signalu senzorja utripa. Preden smo se lotili implementacije zaznavanja utripa, smo opravili test, pri katerem smo opazovali sam signal senzorja. Testna koda je zajemala signal z 200Hz in jih pošiljala direktno na naš strežnik. S testiranjem smo ugotovili, da je signal zelo čist in lepo berljiv, kar je razvidno iz slike 6.7. Opazili smo tudi, da signal s časom narašča in pada, vidno na sliki 6.8, kar pomeni, da fiksna meja, kot jo je imela prva koda, ki smo jo uporabili, ni najboljša za zaznavanje utripa. Zato smo se odločili za svojo implementacijo zaznavanja utripa, ki je opisana v poglavju 5.1.



Slika 6.8: Nihanje signala utripa.

Slika 6.7: Test signala senzorja utripa.

6.2.2 Testiranje senzorja galvanskega odziva kože

Zaradi želje po zaznavanju čustvenega vzburjenja osebe smo v drugi različici prototipa dodali senzor galvanskega odziva kože. Kot test smo si izbrali igranje igre, katerega namen je bil ugotoviti, ali z uporabljenim senzorjem lahko zaznavamo spremembe v signalu, katere bi lahko pomenile, da je oseba čustveno vzburjena. Kot je razvidno iz slike 6.11 smo ob igranju opazili dve večji spremembi v signalu. Prva se je zgodila med 18:40 in 18:55, druga pa ob 20:10, kar sovpada z zabeleženimi stresnimi dogodki ob igranju. Opazili smo, da je obe spremembi v signalu spremljal skok v izmerjenem utripu, le-to nam lahko pomaga pri izdelavi algoritma za zaznavanje čustvene vzburjenosti. Iz pridobljenih rezultatov lahko rečemo, da bi bilo s tako senzoriko možno spremljati čustveno vzburjenost oseb. Za izdelavo zanesljivega algoritma za zaznavanje čustvenega vzburjenja, pa bi morali podobne teste opraviti z več osebami hkrati in beležiti, kaj in kdaj se je zgodil določen dogodek na primer ob gledanju filma ali igranju iger. S pomočjo zbranih podatkov bi lahko nato izdelali algoritem za zaznavanje in določanje čustvenega vzburjenja osebe.

Slika 6.9: Test signala senzorja galvanskega odziva kože.

6.2.3 Rezultati testiranja

Rezultati testiranja so nas pripeljali do končnega prototipa, katerega lahko vidimo na sliki 6.10. Končni prototip vsebuje senzor utripa ter senzor GSR. Opravili smo test realne uporabe. Za test smo med vadbo na sebi imeli naš prototip ter pametno uro Apple Watch. Med vadbo smo primerjali meritve pametne ure in našega prototipa. Izkazalo se je, da so bile izmerjene vrednosti zelo blizu, nekje v območju ±5 utripov v minuti. Kot smo opisali v poglavju testiranja senzorja GSR 6.2.2, prototip še ne vsebuje algoritma za zaznavanje čustvene vzburjenosti, vendar lahko na grafu na sliki 6.11 opazimo kar velike skoke v vrednosti, ki bi lahko pomenili čustveno vzburjenost osebe. S testom smo tako dokazali dobro delovanje našega prototipa senzorske naprave.

Slika 6.10: Končni prototip senzorske naprave.

Slika 6.11: Test končnega prototipa med vadbo.

Poglavje 7

Sklepne ugotovitve

V diplomskem delu smo izdelali celovito rešitev spremljanja fizičnega stanja oseb na domu. Preučili smo ciljno skupino, iz česar smo ugotovili funkcionalnosti, ki jih je naš sistem potreboval. Prav tako smo na trgu pregledali naprave s podobno funkcionalnostjo in našli našo konkurenčno prednost, ceno. Sistem je sestavljen iz treh delov. Prvi je senzorska naprava, za katero smo napisali algoritem za zaznavanje utripa, meri pa tudi galvanski odziv kože ter te podatke pošilja proti strežniku. Drugi je vozlišče, ki skrbi za zaznavanje padca internetne povezave in zagotavlja boljšo zanesljivost sistema s pomočjo medpomnilnika. Kot zadnji del imamo strežnik. Le-ta skrbi za shranjevanje podatkov, njihovo vizualizacijo in obveščanje oziroma alarmiranje ob prednastavljenih dogodkih. Celoten lokalni sistem bi tako stal približno 150 EUR, medtem ko bi sama senzorska naprava, ki je edini obvezni element našega Sistema, stala le okoli 50 EUR - v primeru sistema v oblaku. Z našo implementacijo smo pokazali, da je mogoče narediti cenovno zelo ugodno napravo, ki ima potrebno funkcionalnost za osnovno spremljanje stanja kolikor toliko samostojnega oziroma zdravega starostnika v domačem okolju. To omogoča skrbniku vpogled, ali je z varovano osebo vse vredu. Seveda prototip zahteva izboljšave, pokaže pa, da je pred takšnimi napravami svetla prihodnost.

7.1 Možne nadgradnje

- Najbolj pomembna izmed njih je izboljšava algoritma za zaznavanje srčnega utripa. Trenutno kot veljaven signal vzame le signal med prednastavljeno zgornjo in spodnjo mejo. Pri testiranju smo ugotovili, da lahko signal ob slabi postavitvi senzorja pade izven teh meja, kar povzroči, da signal obravnavamo kot neveljaven, kar pa v teh primerih ni res. Zato bi te meje morale biti dinamične.

- Naša senzorska naprava zajema tudi signal iz senzorja GSR. Napravo bi bilo treba testirati na množici ljudi - na primer ob dnevnem nošenju naprave ter med dodatnimi stresnimi okoliščinami. S pomočjo teh zajetih podatkov bi lahko razbrali vzorce signala ob stresnih dogodkih, ki bi bili naša učna množica, preko katere bi ugotavljali, ali je oseba, ki ima na sebi senzorsko napravo, pod stresom ali ne.

- Pomembna je tudi optimizacija delovanja senzorske naprave z vidika nižanja porabe. Cilj diplomske naloge je bil dokazati izdelavo cenovno dostopnega sistema za spremljanje oseb na domu. Seveda smo izbrali modul in senzorje, ki imajo nizko porabo energije, vendar bi bilo le-to možno še precej zmanjšati tako s pomočjo optimizacije kode kot tudi z novo, bolj optimizirano senzorsko napravo.

- Naslednja nadgradnja bi bila možnost nastavitve dostopne točke, na katero se naša naprava povezuje. Ta nadgradnja je dokaj enostavna, saj obstajajo knjižnice, ki ponujajo funkcionalnost za naš izbrani modul.

Literatura

[1] AARP. Building a better tracker: Older consumers weigh in on activity and sleep monitoring devices. Dosegljivo: https://www.aarp.org/content/dam/aarp/home-and-family/personal-technology/2015-07/innovation-50-project-catalyst-tracker-study-AARP.pdf?kbid=62750&tag=safewicom-20. [Dostopano januar 2018].

[2] Paul Adams. Seniors shy away from wearable devices for health — but shouldn't. Dosegljivo: https://www.statnews.com/2016/11/29/wearable-devices-older-americans/. [Dostopano januar 2018].

[3] Nadja Kovač Aleš Kenda, Lea Lebar. Izvajanje pomoČi na domu, analiza stanja v letu 2016. Dosegljivo: http://www.mddsz.gov.si/fileadmin/mddsz.gov.si/pageuploads/dokumenti__pdf/sociala/Izvajanje_PND_za_leto_2016_3.pdf. [Dostopano september 2017].

[4] Abdullah Alzahrani, Sijung Hu, Vicente Azorin-Peris, Laura Barrett, Dale Esliger, Matthew Hayes, Shafique Akbare, Jérôme Achart, and Sylvain Kuoch. A multi-channel opto-electronic sensor to accurately monitor heart rate against motion artefact during exercise. *Sensors*, 15(10):25681–25702, 2015.

[5] Mosquitto - an open source mqtt v3.1/v3.1.1 broker. Dosegljivo: https://mosquitto.org/. [Dostopano september 2017].

[6] Espressif. Esp8266 pinout. Dosegljivo: http://simba-os.readthedocs.io/en/latest/boards/esp12e.html. [Dostopano februar 2018].

[7] Technologijos expertams. Raspberry pi 3. Dosegljivo: http://tx.lt/raspberry-pi-prapletimai/ds3231-rtc-modulis-raspberry-pi.html. [Dostopano februar 2018].

[8] Grafana - the open platform for beautiful analytics and monitoring. Dosegljivo: https://grafana.com/. [Dostopano september 2017].

[9] InfluxData. Influxdata (influxdb) — time series database for monitoring and analytics. Dosegljivo: https://www.influxdata.com/. [Dostopano september 2017].

[10] Fernando Koyanagi. Esp32 pinout. Dosegljivo: http://www.instructables.com/id/ESP32-Internal-Details-and-Pinout/. [Dostopano februar 2018].

[11] Roger Light. mosquitto passwd — manage password files for mosquitto. Dosegljivo: https://mosquitto.org/man/mosquitto_passwd-1.html. [Dostopano Julij 2017].

[12] Roger Light. mosquitto-tls — configure ssl/tls support for mosquitto. Dosegljivo: http://mosquitto.org/man/mosquitto-tls-7.html. [Dostopano Julij 2017].

[13] World Famous Electronics llc. Pulse sensor. Dosegljivo: https://pulsesensor.com/. [Dostopano februar 2018].

[14] Yakir Malka. Wearable heart beat sensor esp8266 pulse sensor. Dosegljivo: http://www.instructables.com/id/Wearable-heart-beat-sensor-ESP8266Pulse-sensor/. [Dostopano Julij 2017].

[15] Steve Mann. Wearable computing: A first step toward personal imaging. *Computer*, 30(2):25–32, 1997.

[16] Jason Moore. Hrv demographics, part 1 – age and gender. Dosegljivo: https://hrvcourse.com/hrv-demographics-age-gender/. [Dostopano december 2017].

[17] A Noraziah, Muhd Azrulnizam Suna Abdullah, Nurzety Aqtar, MohammedAdam Ibrahim Fakhreldin, and Muhammad Nubli Abd Wahab. Greenvec game for skin conductivity level (scl) biofeedback performance simulator using galvanic skin response (gsr) sensor. *International Journal of Software Engineering and Computer Systems*, 1(1):41–53, 2015.

[18] Felix Rusu. Lowpowerlab si7021. Dosegljivo: https://github.com/LowPowerLab/SI7021. [Dostopano Julij 2017].

[19] Skupnost socialnih zavodov Slovenije. Pregled prošenj in prostih mest v domovih za starejše in posebnih socialno varstvenih zavodih. Dosegljivo: https://servis.ssz-slo.si/porocilo.pdf. [Dostopano november 2017].

[20] Dom starejših občanov Ljubljana Šiška. Cenik socialno varstvenih storitev. Dosegljivo: http://www.dso-siska.si/data/file/dom/Cenik_marec_2016.pdf. [Dostopano oktober 2017].

[21] Seeed Studio. Grove - gsr sensor. Dosegljivo: http://wiki.seeed.cc/Grove-GSR_Sensor/. [Dostopano februar 2018].

[22] Seeed Studio. Mini rtc module. Dosegljivo: https://www.seeedstudio.com/Mini-RTC-Module-p-1702.html. [Dostopano februar 2018].

[23] Ministrstvo za javno upravo. Domovi za starejše. Dosegljivo: https://e-uprava.gov.si/podrocja/delo-upokojitev/upokojitev/domovi-za-ostarele.html. [Dostopano oktober 2017].

[24] MINISTRSTVO ZA DELO DRUŽINO SOCIALNE ZADEVE IN ENAKE ZMOŽNOSTI. Pomoč na domu. Dosegljivo:

http://www.mddsz.gov.si/si/delovna_podrocja/sociala/izvajalci/pomoc_na_domu/. [Dostopano oktober 2017].