Analiza EKG-signala prikupljenog integriranim sklopom MAX30003 u stvarnom vremenu

Bošnjak, Eugen

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1520

ANALIZA EKG-SIGNALA PRIKUPLJENOG INTEGRIRANIM SKLOPOM MAX30003 U STVARNOM VREMENU

Eugen Bošnjak

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1520

ANALIZA EKG-SIGNALA PRIKUPLJENOG INTEGRIRANIM SKLOPOM MAX30003 U STVARNOM VREMENU

Eugen Bošnjak

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1520

Pristupnik:	**Eugen Bošnjak (0036539172)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Hrvoje Džapo

Zadatak: **Analiza EKG-signala prikupljenog integriranim sklopom MAX30003 u stvarnom vremenu**

Opis zadatka:

Istražiti metode mjerenja elektrokardiografskog (EKG) signala. Istražiti mogućnosti postojećeg razvojnog sustava za mjerenje EKG-signala temeljenog na integriranom sklopu MAX30003 upravljanim mikrokontrolerom. Nadograditi postojeću programsku potporu koja će omogućiti eksperimentalno prikupljanje i snimanje EKG-signala u različitim uvjetima rada i aktivnosti ispitanika. Istražiti postojeće metode i implementirati odabrane algoritme za obradu EKG-signala u svrhu procjene sljedećih parametara: srčani ritam (heart rate, HR), varijacija srčanog ritma (heart rate variability, HRV) i estimacija ritma disanja iz EKG signala (respiration rate, RR). Usporediti performanse različitih algoritama i provesti usporedbu s referentnim mjernim sustavima. Odabrane algoritme implementirati na mikrokontroleru s ograničenim računalnim resursima i prilagoditi ih za izvođenje u stvarnom vremenu.

Rok za predaju rada: 14. lipnja 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1520

ANALIZA EKG-SIGNALA PRIKUPLJENOG INTEGRIRANIM SKLOPOM MAX30003 U STVARNOM VREMENU

Eugen Bošnjak

Zagreb, lipanj, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1520

Pristupnik:	**Eugen Bošnjak (0036539172)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Hrvoje Džapo

Zadatak: **Analiza EKG-signala prikupljenog integriranim sklopom MAX30003 u stvarnom vremenu**

Opis zadatka:

Istražiti metode mjerenja elektrokardiografskog (EKG) signala. Istražiti mogućnosti postojećeg razvojnog sustava za mjerenje EKG-signala temeljenog na integriranom sklopu MAX30003 upravljanim mikrokontrolerom. Nadograditi postojeću programsku potporu koja će omogućiti eksperimentalno prikupljanje i snimanje EKG-signala u različitim uvjetima rada i aktivnosti ispitanika. Istražiti postojeće metode i implementirati odabrane algoritme za obradu EKG-signala u svrhu procjene sljedećih parametara: srčani ritam (heart rate, HR), varijacija srčanog ritma (heart rate variability, HRV) i estimacija ritma disanja iz EKG signala (respiration rate, RR). Usporediti performanse različitih algoritama i provesti usporedbu s referentnim mjernim sustavima. Odabrane algoritme implementirati na mikrokontroleru s ograničenim računalnim resursima i prilagoditi ih za izvođenje u stvarnom vremenu.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem mentoru prof. dr. sc. Hrvoju Džapi i asistentici Karli Salamun, mag. ing. na potpori i pomoći pri rješavanju tehničkih i svih ostalih poteškoća. Također zahvaljujem doc. dr. sc. Josipu Lončaru na ustupljenoj razvojnoj pločici s mikrokontrolerom.

Sadržaj

1. Uvod

Elektrokardiografija je proces bilježenja električne aktivnosti srca u obliku elektrokardiograma (EKG) koji predstavlja zapis ovisnosti akcijskih potencijala srca u vremenu. Za mjerenje potencijala koriste se elektrode postavljene na kožu, a sama dijagnostička metoda je neinvazivna i praktična. Osim što je od izuzetne važnosti u medicinskoj dijagnostici, elektrokardiografija se koristi i u sportu za praćenje performansi i opterećenja sportaša. Primarni parametar koji se koristi u dijagnostici i praćenju aktivnosti je srčani ritam (engl. *Heart Rate*, HR), koji predstavlja broj otkucaja u jedinici vremena (tipično u minuti). Drugi parametar koji se također često prati je varijabilnost srčanog ritma (engl. *Heart Rate Variability*, HRV), koji može pružiti uvid i u psihofizički status ispitanika [1].

Da bi se iz sirovog signala elektrokardiograma odredio srčani ritam (HR), potrebno ga je obraditi koristeći prikladne algoritme. Jedan od najpoznatijih algoritama koji se koriste u tu svrhu je Pan-Tompkinsov algoritam [2] koji se temelji na detekciji tzv. QRS kompleksa. Dodatnom obradom EKG signala može se odrediti varijabilnost srčanog ritma (HRV). Međutim, osim samih parametara rada srca, EKG signal može se koristiti i za estimaciju drugih fizioloških parametara od interesa, a jedan od njih je i posredna procjena ritma disanja (engl. *Respiration Rate*, RR), koja se također može dobiti primjenom naprednijih algoritama obrade signala. Zbog prikladnosti primjene na ispitaniku, jednostavnosti postava za prikupljanje podataka i razmjerno niske računalne složenosti algoritamske obrade snimljenog signala, EKG se često koristi na manjim nosivim (engl. *wearable*) elektroničkim uređajima koji su prikladni u svakodnevnim aktivnostima. Iako su takvi EKG nosivi uređaji manje točni od bolničkih elektrokardiografa ili holtera, mogu pružiti sasvim dobre rezultate primjerice kod praćenja aktivnosti sportaša koji žele u stvarnom vremenu pratiti rad srca pri opterećenju ili kada se takvi uređaji koriste za praćenje po-

našanja srca u aktivnostima u svakodnevnom životu.

Cilj ovog rada bio je istražiti mogućnosti primjene mikrokontrolera iz porodice *Arm Cortex-M* za implementaciju algoritama obrade EKG signala u svrhu određivanja srčanog ritma, varijabilnosti srčanog ritma i ritma disanja u stvarnom vremenu, obradom signala dobivenog sa senzora *MAX30003*. Razvijena programska potpora radi u stvarnom vremenu i dobiveni rezultati šalju se putem asinkronog serijskog (UART) sučelja na osobno računalo, gdje se prikazuju u grafičkom korisničkom sučelju implementiranom u programskom jeziku Python. Rad je podijeljen u cjeline kako slijedi. U drugom poglavlju rada opisana su načela elektrokardiografije. U trećem poglavlju prikazan je detaljan opis korištenog sklopovlja (integriranog sklopa *MAX30003* za snimanje EKG signala i mikrokontrolera *STM32L412KB*). U četvrtom poglavlju prezentiraju se algoritmi korišteni za određivanje parametara od interesa (HR, HRV i RR), kao i programska potpora za prikaz podataka korisniku na osobnom računalu.

2. Elektrokardiografija

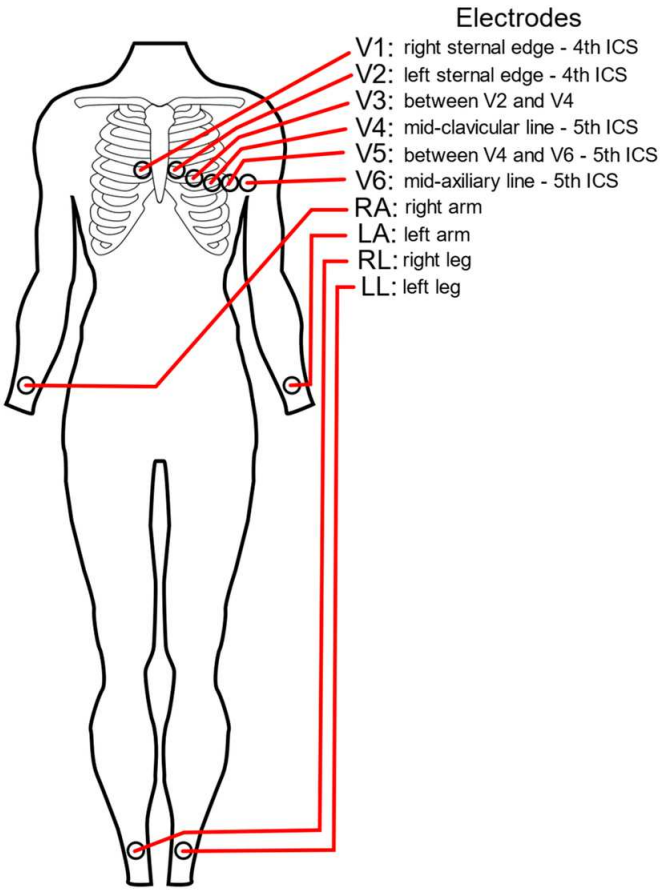
Kod provođenja postupka elektrokardiografije [3] elektrode su smještene na koži i mjere električnu aktivnost srca, koja se odnosi na malene električke promjene uzrokovane depolarizacijom i repolarizacijom srčanog mišića. Za medicinsku dijagnostiku obično se koristi 10 elektroda, od kojih se 4 postavlja na udove, a preostalih 6 na područje prsa, kao što je prikazano na slici 2.1. Iz tih se elektroda mjeri 12 naponskih signala, stoga se ovakav EKG naziva 12-kanalni EKG. Premda opisani način mjerenja EKG signala predstavlja standard u kliničkoj dijagnostici, on je nepraktičan za primjenu u nosivim uređajima, kod kojih se zbog praktičnosti koristi značajno manje točan jednokanalni EKG.

Elektrokardiogram predstavlja ovisnost napona srčane aktivnosti o vremenu na svakom od kanala. Oblik signala drukčiji je ovisno o kanalu na kojem se mjeri, prema sljedećim osnovnim načelima:

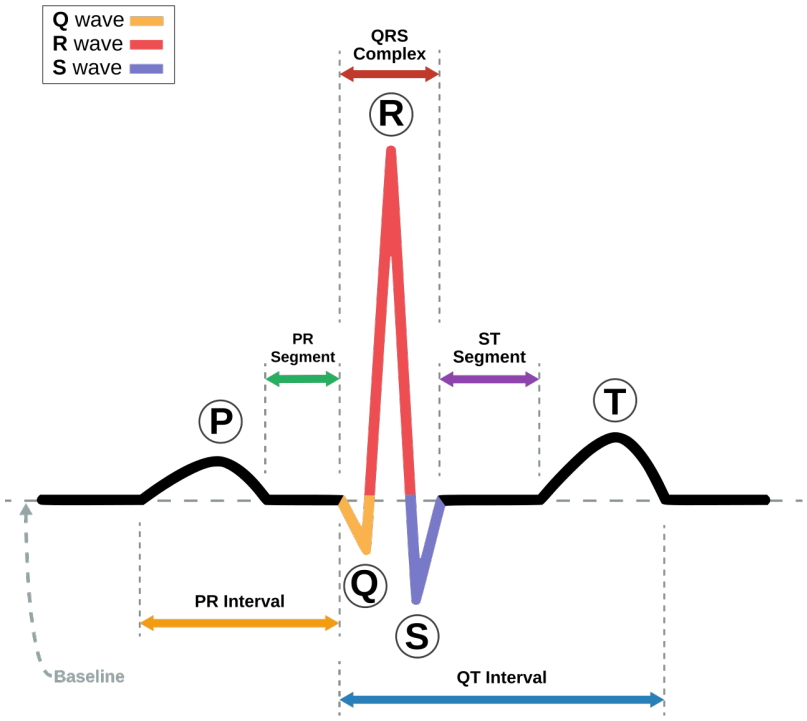
- depolarizacija srca prema pozitivnoj elektrodi kanala daje pozitivan napon,
- depolarizacija srca od pozitivne elektrode kanala daje negativan napon,
- repolarizacija srca prema pozitivnoj elektrodi kanala daje negativan napon,
- repolarizacija srca od pozitivne elektrode kanala daje pozitivan napon.

Glavne značajke signala prikazane su na slici 2.2.

- P val predstavlja depolarizaciju pretklijetki,
- QRS kompleks predstavlja depolarizaciju klijetki,
- T val predstavlja repolarizaciju klijetki.



Slika 2.1. Način spajanja elektroda za 12-kanalni EKG [4]



Slika 2.2. Izgled i značajke EKG signala [5]

3. Sklopovski dio sustava

3.1. Integrirani sklop MAX30003

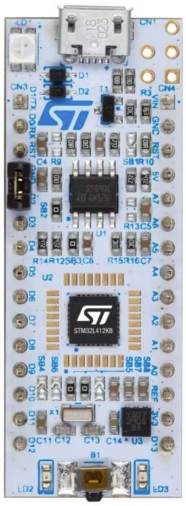
MAX30003 [6] je sklop koji sadrži analogno sučelje za jednokanalno mjerenje EKG signala, namijenjen za primjenu u nosivoj elektronici. Osim uzorkovanja EKG signala, također ima i hardverski implementiran algoritam za detekciju otkucaja srca temeljen na Pan-Tompkins algoritmu [2], detekciju odspajanja elektroda i filtriranje elektromagnetskih smetnji. Komunikacija s *MAX30003* odvija se putem protokola SPI. *MAX30003* uzorkuje EKG signal frekvencijom 128 uzoraka po sekundi (ta se frekvencija može promijeniti u konfiguracijskim registrima sklopa) i uzorke sprema u svoj FIFO spremnik. Opisi svih registara, funkcionalnosti te vremenski dijagrami komunikacijskog protokola mogu se pronaći u [6]. Ovdje valja napomenuti da se ugrađeno prepoznavanje otkucaja srca u ovom radu ne koristi, s obzirom da je cilj algoritme implementirati u softveru na mikrokontroleru.

Konačno, budući da je *MAX30003* integrirani sklop koji dolazi u TQFN (*Thin Quad Flat No-Lead*) i WLP (*Wafer-Level Package*), radi jednostavnosti povezivanja s razvojnim sustavom koristi se gotova razvojna pločica za ovaj čip *ECG 3 Click* proizvođača *MikroElektronika* na kojoj je već montiran sklop *MAX30003*, uz ostale pomoćne elektroničke komponente, te su na vanjske pinove izvedene linije sklopa [7].

3.2. Mikrokontroler STM32L412KB

STM32L4 je porodica mikrokontrolera temeljenih na *Arm Cortex-M4* arhitekturi koje proizvodi *STMicroelectronics*. U ovom radu koristi se razvojna pločica *NUCLEO-L412KB* s mikrokontrolerom *STM32L412KB*, prikazana na slici 3.1. Korisna značajka ovog razvojnog sustava jest proslijeđivanje jednog od UART sučelja mikrokontrolera na virtu-

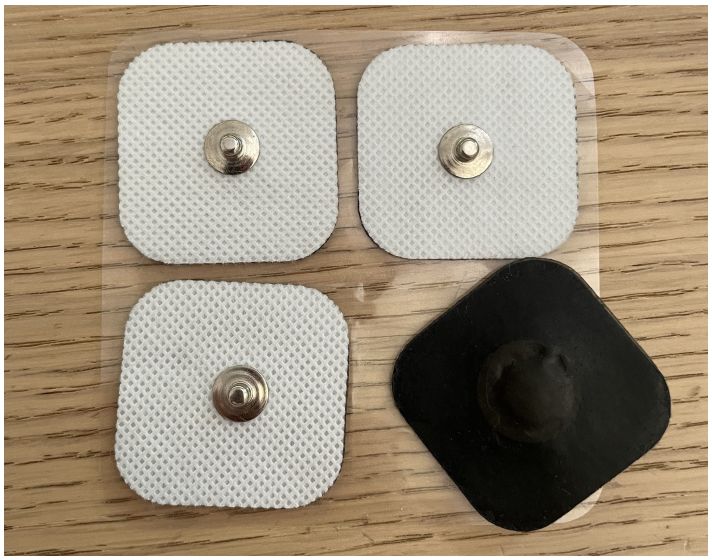
alni COM priključak putem sučelja ST-Link, stoga nije potreban dodatni UART na USB adapter za komunikaciju s računalom.



Slika 3.1. Razvojni sustav *NUCLEO-L412KB* [8]

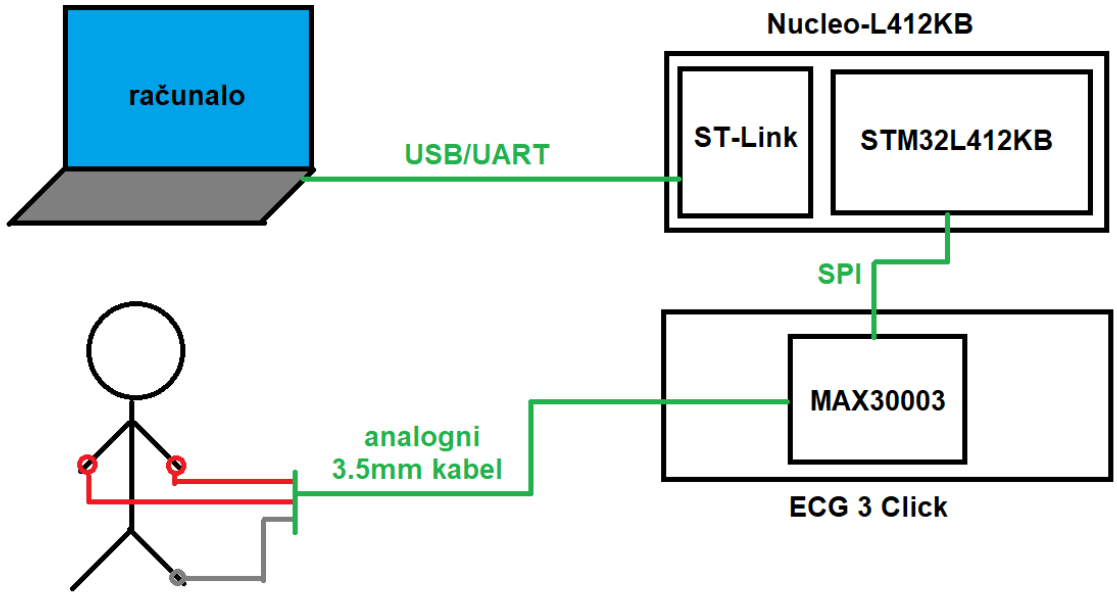
Mikrokontroler *STM32L412KB* podržava frekvencije takta do 80 MHz, a koji se može dovesti iz internog RC oscilatora (*Low Speed Internal - LSI* te *High Speed Internal - HSI*) ili iz vanjskog kristalnog oscilatora (*Low Speed External - LSE* te *High Speed External - HSE*). *Nucleo* pločica ima na sebi ugrađen kristalni oscilator od 8 MHz koji se *solder bridge* konfiguracijom može dovesti na ulaz *OSCIN* mikrokontrolera te koristiti kao *HSE* takt.

3.3. Električna shema sustava



Slika 3.2. Polusuhe kontaktne EKG elektrode

Slika 3.3. Struktura 3.5mm audio kabela



Slika 3.4. Električna shema sklopovlja sustava

Blok shema sustava prikazana je na slici 3.4. Prvo, elektrode se na osobu postavljaju na idući način. Elektroda koja će predstavljati masu stavlja se na jednu nogu iznad gležnja. Preostale dvije se stavljaju simetrično svaka na jednu ruku, s unutarnje strane zapešća. Koriste se jednokratne polusuhe kontaktne elektrode, koje se lijepe na kožu uz pomoć vodljivog gela, a čiji je izgled prikazan na slici 3.2. Za primanje EKG signala, pločica *ECG 3 Click* ima konektor za 3.5mm audio kabel s tri vodiča. On je praktičan za ovu svrhu zbog svoje strukture, prikazane na slici 3.3. Vodič za masu spaja se na elektrodu na nozi, a preostala dva spajaju se na elektrode na rukama. Ovakav način spajanja korišten je u radu [9] i pružio je prihvatljive rezultate. Ako se primijeti da je valni oblik dobivenog signala obratan od onoga kako bi EKG trebao izgledati, to znači da su vodiči za lijevi i desni kanal audio kabela obratno spojeni na elektrode na rukama te im samo treba

zamijeniti mjesta.

ECG 3 Click se na mikrokontroler spaja pomoću šest vodiča, od kojih četiri služe za prijenos signala protokola SPI, a preostala dva za napajanje. Masa pločice se spaja na masu mikrokontrolera, a napajanje pločice na 3,3V priključak mikrokontrolera (označen kao *3V3*). *Chip select* pločice spaja se na proizvoljno odabrani GPIO priključak mikrokontrolera. Ovdje je za to iskorišten priključak 5 na GPIO vratima A, kao što je prikazano na slici 4.2. *SCK* na pločici spaja se na priključak gdje je doveden signal *SCK* na mikrokontroleru. Priključci *SDO* i *SDI* spajaju se redom na priključke mikrokontrolera gdje se nalaze signali *MISO* i *MOSI*. S obzirom da priključci pločice *Nucleo-L412KB* nisu označeni oznakama GPIO vrata i priključaka koji im pripadaju (npr. PA3, PB7 itd.), već oznakama koje koristi *Arduino Nano*, potrebno je u dokumentaciji [10] vidjeti za svaki GPIO priključak kojem fizičkom priključku na pločici *Nucleo* on odgovara. Konkretno, ovdje su relevantni:

- *SCK*, koji je na *PA1*, odgovara priključku s oznakom *A1*,
- *CS*, koji je na *PA5*, odgovara priključku s oznakom *A4*,
- *MISO*, koji je na *PA6*, odgovara priključku s oznakom *A5*, te
- *MOSI*, koji je na *PA7*, odgovara priključku s oznakom *A6*.

Konačno, za spajanje pločice *Nucleo* s računalom, koristi se obični USB na Micro-USB kabel. Nakon spajanja, potrebno je odrediti na kojem se virtualnom priključku odvija komunikacija između računala i mikrokontrolera. To se može napraviti pomoću programa *Device Manager* na operacijskom sustavu *Windows*, dok se na operacijskom sustavu *Linux* mogu iskoristiti alati kao što su `lsusb` ili `dmesg`.

4. Programska potpora za mikrokontrole i računalo

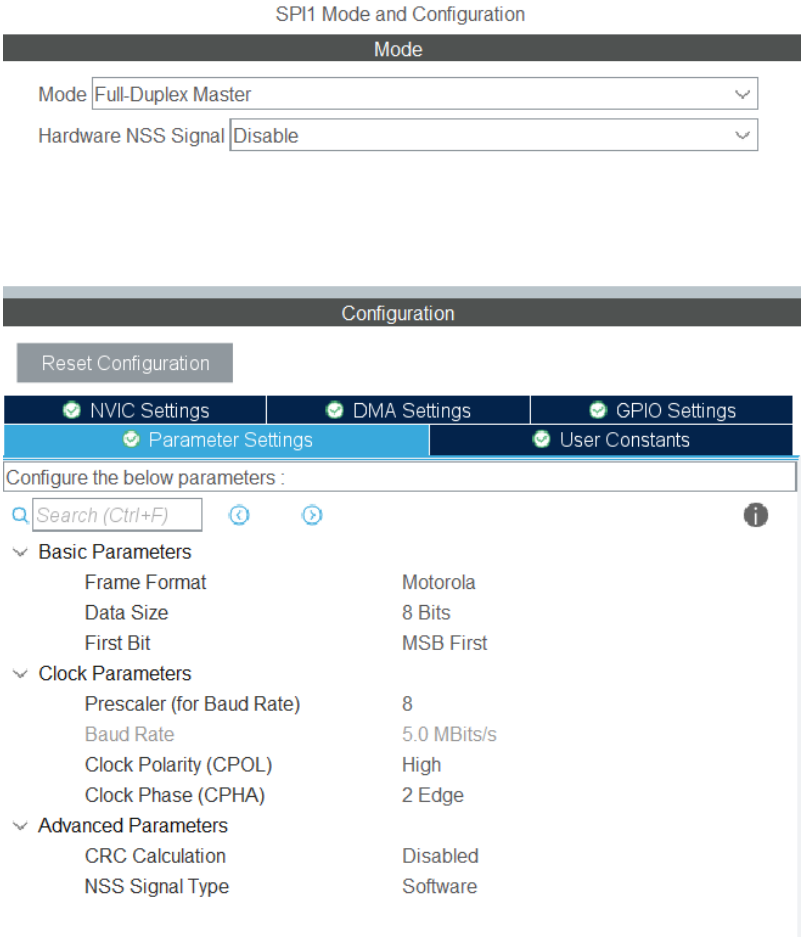
4.1. Programska potpora za mikrokontroler

Za razvoj programske potpore za mikrokontroler koristi se programski jezik *C* i razvojno okruženje *STM32CubeIDE*. Ono je temeljeno na okruženju *Eclipse*, a sadrži dodatne alate specifične za programiranje *STM32* mikrokontrolera, kao što je *cross-compiler* za *Arm* arhitekturu, debugger i alat za konfiguraciju upravljačkih programa *STM32CubeMX*. Sve navedeno omogućuje stvaranje projekata sa svim potrebnim inicijalizacijskim kodom na temelju odabira mikrokontrolera i njegove konfiguracije, bez potrebe da se ti zadaci ručno rade. Također, uz razvojno okruženje dolazi i skup *HAL* (engl. *Hardware Abstraction Layer*) upravljačkih programa, koji omogućuju jednostavno upravljanje periferijama bez direktnog upisivanja numeričkih vrijednosti u registre. U nastavku će biti ukratko opisane najvažnije cjeline programske potpore za mikrokontroler, a cjelokupan kod nalazi se u privitku ovog rada.

4.1.1. Konfiguracija mikrokontrolera

Na početku je potrebno konfigurirati taktove mikrokontrolera. Uz pravilnu *solder bridge* konfiguraciju (tablica 6. u dokumentaciji [10]), točnije uz kratko spajanje *SB17*, na ulaz *OSCIN* bit će doveden takt vanjskog kristalnog 8 MHz oscilatora sa *Nucleo* pločice. Tada se u konfiguracijskoj *.ioc* datoteci može omogućiti *HSE* takt, te ga je u postavkama taktova potrebno dovesti na sklop PLL. Odabirom odgovarajućih parametara za sklop PLL, primjerice dijeljenje s 2, množenje s 40 i dijeljenje s 2, na *PLLCLK* dobiva se takt od 80 MHz koji se jednostavno dovodi do jezgre te do AHB i APB sabirnica. Ovim se koracima dobiva maksimalna brzina jezgre radi izvođenja algoritama, dok brži takt na APB sabirnici omogućuje komunikaciju većom brzinom preko UART-a prema računalu.

Komunikacija sa sklopom *MAX30003* odvija se putem protokola SPI. Odabrani mikrokontroler ima jedno sučelje za SPI. Njegove se postavke moraju namjestiti tako da odgovaraju onome što podržava *MAX30003*, a primjer postavki prikazan je na slici 4.1.

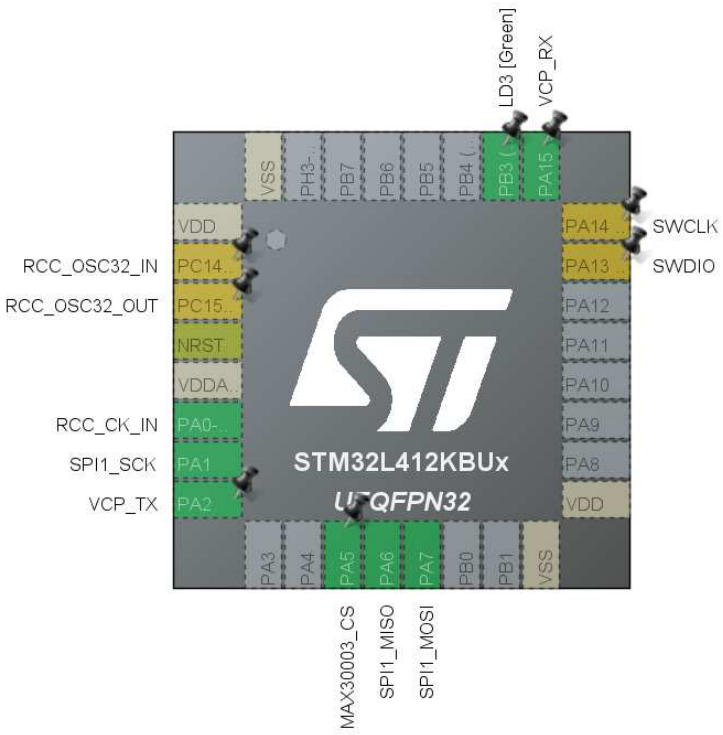


Slika 4.1. Postavke sučelja *SPI*

Još nedostaje *chip select* signal za SPI. Za njega se rezervira jedan GPIO izlaz koji će se na početku postaviti u logičku jedinicu i tijekom SPI transakcije aktivirati spuštanjem u logičku nulu.

Konačno, za komunikaciju s računalom mora se omogućiti USART2 periferija. To je sučelje povezano na virtualni COM port, dok je USART1 povezan na Rx i Tx pinove na pločici. Za *baud rate* zadaje se jedna od standardnih vrijednosti, na primjer 115200 bps.

Raspored pinova mikrokontrolera prikazan je na slici 4.2. U [10] moguće je vidjeti koji pin na pločici odgovara određenom pinu mikrokontrolera.



Slika 4.2. Raspored omogućenih pinova na mikrokontroleru

4.1.2. Komunikacija sa sklopom *MAX30003*

Napravljene su dvije pomoćne funkcije, prikazane u odsječku 4..1, koje se koriste za komunikaciju s *MAX30003* - jedna čita registar sklopa, a druga zapisuje u registar sklopa. Obje operacije obavljaju se putem protokola SPI u skladu s vremenskim dijagramom iz [6]. Transakcija preko SPI se jednostavno obavlja koristeći funkcije *HAL_SPI_Transmit* i *HAL_SPI_TransmitReceive* iz *STM-ove HAL* biblioteke.

```
void MAX30003_Write(uint8_t addr, uint32_t data) {
    addr = (addr << 1) | 0x00;
    uint8_t _data[3] = {(data >> 16) & 0xFF, (data >> 8) & 0xFF, data & 0xFF};

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
    if (HAL_SPI_Transmit(&hspi1, (uint8_t*) &addr, 1, 100) != HAL_OK) {
        printf("Error: Failed to write address\r\n");
    }
    while (hspi1.State == HAL_SPI_STATE_BUSY);
    if (HAL_SPI_Transmit(&hspi1, (uint8_t*) &_data, 3, 100) != HAL_OK) {
        printf("Error: Failed to write data\r\n");
```

```
    }
    while (hspi1.State == HAL_SPI_STATE_BUSY);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
}

uint32_t MAX30003_Read(uint8_t addr) {
    addr = (addr << 1) | 0x01;
    uint8_t data[4];

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
    if (HAL_SPI_TransmitReceive(&hspi1, (uint8_t*) &addr, &data[0], 4, 100)
        != HAL_OK) {
        printf("Error: Failed to read\r\n");
    }
    while (hspi1.State == HAL_SPI_STATE_BUSY) ;
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);

    uint32_t ret = ((uint32_t) data[1] << 16) | ((uint32_t) data[2] << 8) |
        data[3];
    return ret;
}
```

Odsječak koda 4..1 Pomoćne funkcije za komunikaciju s *MAX30003*

Da bi se ispitala ispravnost komunikacije, na početku izvođenja iz registra *INFO* sklopa *MAX30003* čita se vrijednost čiji bitovi 23-20 (prema [6]) moraju biti 0101, tj. dekadski 5. Ako to nije tako, došlo je do greške i sklop ne odgovara kako bi trebao. Ovdje valja napomenuti da se prije čitanja spomenutog registra mora napraviti još neka (bilo koja osim reseta) operacija nad sklopom kako bi čitanje bilo valjano.

Nakon uspješne provjere funkcionalnosti, sklop se koristi tako da se čita iz njegovog FIFO spremnika. Ovdje je to učinjeno pomoću radnog čekanja sve dok čitanje iz FIFO-a ne vrati valjanu vrijednost. Najniža tri bita pročitane vrijednosti sadrže dodatne informacije o uzorku, kao što su: je li došlo do preljeva FIFO spremnika, je li pročitani uzorak zadnji u FIFO spremniku, je li FIFO prazan i slično. Ako je vrijednost valjana, iznos uzorka

dobiva se predznačnim proširivanjem vrijednosti u bitovima 23-6 i on se šalje u obradu. Implementacija čitanja uzorka sa sklopa prikazana je u odsječku 4..2

```
// poll the sensor until we get a valid sample back
do {
    val = MAX30003_Read(0x21);
    flag = (val >> 3) & 0x7;

    if (flag == 7) {
        MAX30003_Write(0x0A, 0x00);  // fifo reset
    } else if (flag == 2 || flag == 3) {
        HAL_Delay(10);               // last sample in fifo
    }
} while (flag > 3);

sample = (val >> 6);      // remove the flags from the value

// sign extend the sample value
sample <<= 14;
sample >>= 14;

PT_ProcessSample(sample);
```

Odsječak koda 4..2 Dobivanje EKG uzorka s *MAX30003*

4.1.3. Komunikacija s računalom

Na početku, radi lakšeg uhodavanja sustava potrebno je ostvariti ispis formatiranih znakovnih nizova na UART. Najjednostavniji i najpraktičniji način za to jest *I/O retargeting*. Na računalima s operacijskim sustavima, funkcije za ulaz i izlaz, kao što je *printf*, svoju funkcionalnost postižu tako da obavljaju sistemski poziv prema operacijskom sustavu za ispis na standardni izlaz. Dakle, kako bi te funkcije radile na mikrokontroleru, potrebno je definirati "sistemski poziv" (koji nije zaista pravi sistemski poziv jer se kod ne izvodi na OS, već je obična funkcija) koji će biti pozvan pri ispisu. Pri stvaranju projekta, dodana je datoteka *syscalls.c* koja upravo definira opisane funkcije. No, u toj datoteci postoje dvije

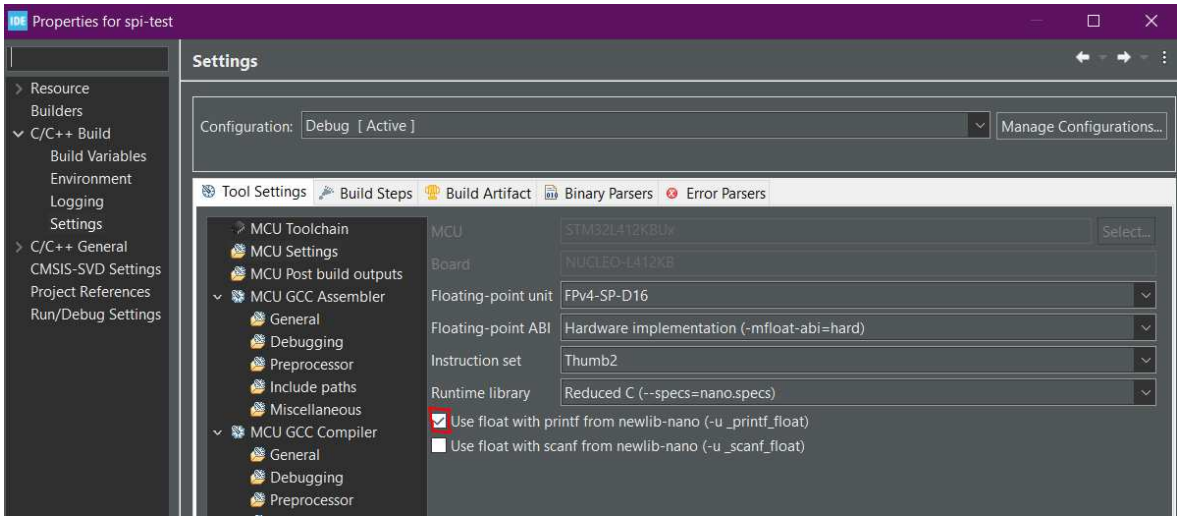
funkcije koje se koriste, a samo su deklarirane: *__io_getchar* te *__io_putchar*; stoga bi pokušaj poziva *printf* rezultirao pogreškom, jer on poziva *_write* koji poziva *__io_putchar*. Jasno je da spomenute funkcije mora programer definirati zbog toga što on zna kamo se ispisuje i odakle se unosi. To se radi tako da se jednostavno u nekoj *.c* datoteci napiše definicija, kao što je prikazano u odsječku 4..3

```
int __io_putchar(int ch) {
    HAL_UART_Transmit(&huart2, (uint8_t*) &ch, 1, 100);
    return ch;
}
```

Odsječak koda 4..3 *Retargeting* funkcije za ispis znaka

Sada se *printf* može pozvati i njegov će se ispis slati na računalo putem UART-a. Valja primijetiti da su u *syscalls.c* i neke funkcije koje su već ondje definirane također označene oznakom *__attribute__((weak))*. To je napomena linkeru da tu definiciju zanemari ako je programer napisao drugu, što znači da je moguće i te funkcije "nadjačati".

Na koncu, kako bi se mogli ispisivati brojevi s pomičnim zarezom, potrebno je pod *"Project - Properties - C/C++ Build - Settings - MCU Settings"* omogućiti opciju *"Use float with printf from newlib-nano"*, sukladno slici 4.3.



Slika 4.3. Postavke za ispis brojeva s pomičnim zarezom

Nakon što je ispitivanjem utvrđeno da je funkcionalnost sustava odgovarajuća, brišu se svi pozivi izlaznih funkcija koji su služili za uhodavanje, te se komunikacija prema koris-

ničkom sučelju implementira u binarnom formatu. Iako je ispis formatiranih znakovnih nizova praktičan za uhodavanje, binarni je ispis znatno bolja opcija za konačnu verziju aplikacije. Naime, mikrokontroler šalje pet podataka primitivnog tipa prema korisničkom sučelju:

- iznos zadnjeg uzorka EKG signala prije prolaska kroz Pan-Tompkins algoritam (`int32_t`),
- varijabilnost srčanog ritma (`float`)
- trenutni iznos procjene disanja (`int32_t`),
- trenutni prosjek trajanja R-R intervala (`int32_t`),
- *boolean* zastavicu koja označava detekciju QRS kompleksa (`uint8_t`).

To je ukupno 17 bajtova podataka. Kad bi se ispis radio u obliku formatiranog ASCII znakovnog niza, tj. pomoću koda iz odsječka 4..4, nepotrebno bi se prenosilo znatno više podataka. Na primjer, niz `"143 102.13 100 100 0\r\n"` ima 22 bajta. Nadalje, pokazalo se da su ovi cijeli brojevi po apsolutnoj vrijednost uvijek manji od 1000, stoga se umjesto `int32_t` može koristiti i `int16_t`, čime bi se veličina binarnog paketa podataka smanjila na 11 bajtova, dok bi ASCII formatirani znakovni niz i dalje imao 22 bajta. Najgori problem kod ispisa formatiranih znakovnih nizova su brojevi s pomičnim zarezom. Broj `123.4567` ima 7 znamenki i točku, što je 8 znakova, tj. 8 bajtova. To je čak dvostruko više nego veličina zapisa tog broja u IEEE754 formatu, a ovaj omjer postaje sve lošiji za brojeve većeg reda veličine ili s više decimala.

```
int32_t sample, edr, rr;
float hrv;
uint8_t is_peak;

printf("%ld %f %ld %ld %hhu\r\n", sample, hrv, edr, rr, is_peak);
```

Odsječak koda 4..4 Neučinkovit način prijenosa podataka prema računalu

S obzirom na sve navedeno, ispis se radi u binarnom formatu, istim redoslijedom kako su podatci nabrojani iznad: zadnji EKG uzorak, HRV, procjena disanja, RR prosjek, zas-

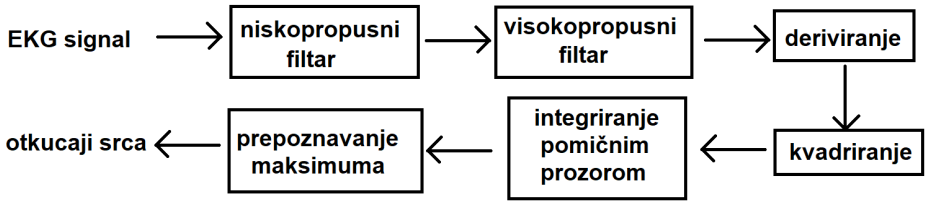
tavica prepoznatog QRS kompleksa. Ovdje je važno da zastavica za QRS dolazi na kraju zbog načina kako Pythonov `struct` modul dodaje *padding*, kao što je opisano u potpoglavlju 4.2. Također, potrebno je na neki način signalizirati početak komunikacije kako bi korisničko sučelje znalo prepoznati kada treba krenuti ispisivati. Stoga se prije početka obrade signala ispisuje niz znakova `"!!!!!"`. Implementacija komunikacije prema računalu prikazana je u odsječku 4..5

```
printf("!!!!!");   // pocetak komunikacije
fflush(stdout);   // obavezno napraviti kako bi string zaista otisao na UART

while (1) {
    // dohvati uzorak
    // ...

    // obradi uzorak
    PT_ProcessSample(sample);

    // dohvati izracunate vrijednosti
    uint8_t is_peak = PT_GetPeak();
    float hrv = PT_GetCurrentHRV();
    int32_t edr = PT_GetCurrentEDR();
    int32_t rr = PT_GetCurrentRR();

    // posalji prema racunalu
    HAL_UART_Transmit(&huart2, (uint8_t*) &sample, 4, 100);
    HAL_UART_Transmit(&huart2, (uint8_t*) &hrv, 4, 100);
    HAL_UART_Transmit(&huart2, (uint8_t*) &edr, 4, 100);
    HAL_UART_Transmit(&huart2, (uint8_t*) &rr, 4, 100);
    HAL_UART_Transmit(&huart2, (uint8_t*) &is_peak, 1, 100);
}
```

Odsječak koda 4..5 Slanje podataka prema korisničkom sučelju

4.1.4. Detekcija QRS kompleksa

Za detekciju otkucaja srca, tj. QRS kompleksa u elektrokardiogramu, koristi se Pan-Tompkins algoritam [2]. Algoritam se sastoji od primjene nekoliko filtara na EKG signal, nakon čega se provodi postupak prepoznavanja maksimuma koji odgovaraju QRS kompleksima. Slika 4.4. opisuje tijek obrade signala pomoću blok dijagrama.



Slika 4.4. Blok dijagram Pan-Tompkins algoritma

Niskopropusni filtar zadan je jednadžbom diferencija:

$$y[n] = 2y[n-1] - y[n-2] + x[n] - 2x[n-6] + x[n-12]$$

Njegova je granična frekvencija oko 11 Hz, pojačanje 36, a kašnjenje 5 uzoraka. Visokopropusni filtar zadan je jednadžbom diferencija [11]:

$$y[n] = y[n-1] - \frac{x[n]}{32} + x[n-16] - x[n-17] + \frac{x[n-32]}{32}$$

Granična frekvencija je oko 5 Hz, pojačanje je 1, a kašnjenje 16 uzoraka. Filter derivacije [11] zadan je s:

$$y[n] = \frac{1}{8}(-2x[n-4] - x[n-3] + x[n-1] + 2x[n])$$

i ima kašnjenje od 2 uzorka. Kvadriranje se postiže trivijalno:

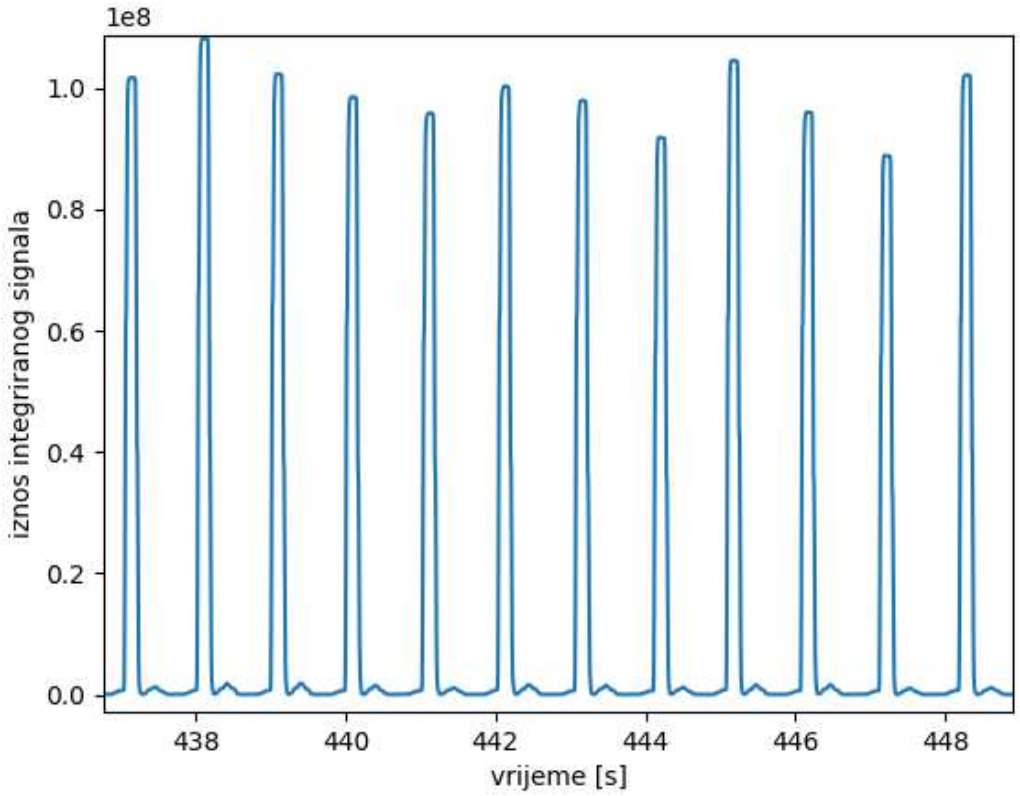
$$y[n] = (x[n])^2$$

Dok se zadnji korak, integriranje pomičnim prozorom, postiže filtrom:

$$y[n] = \frac{1}{N}(x[n] + x[n-1] + x[n-2] + ... + x[n-N+1])$$

pri čemu se veličina prozora N bira tako da je prozor širok oko 150 ms. U ovom radu,

frekvencija uzorkovanja je 128 Hz, čemu odgovara širina prozora od 19 uzoraka.



Slika 4.5. Rezultat integriranja pomičnim prozorom

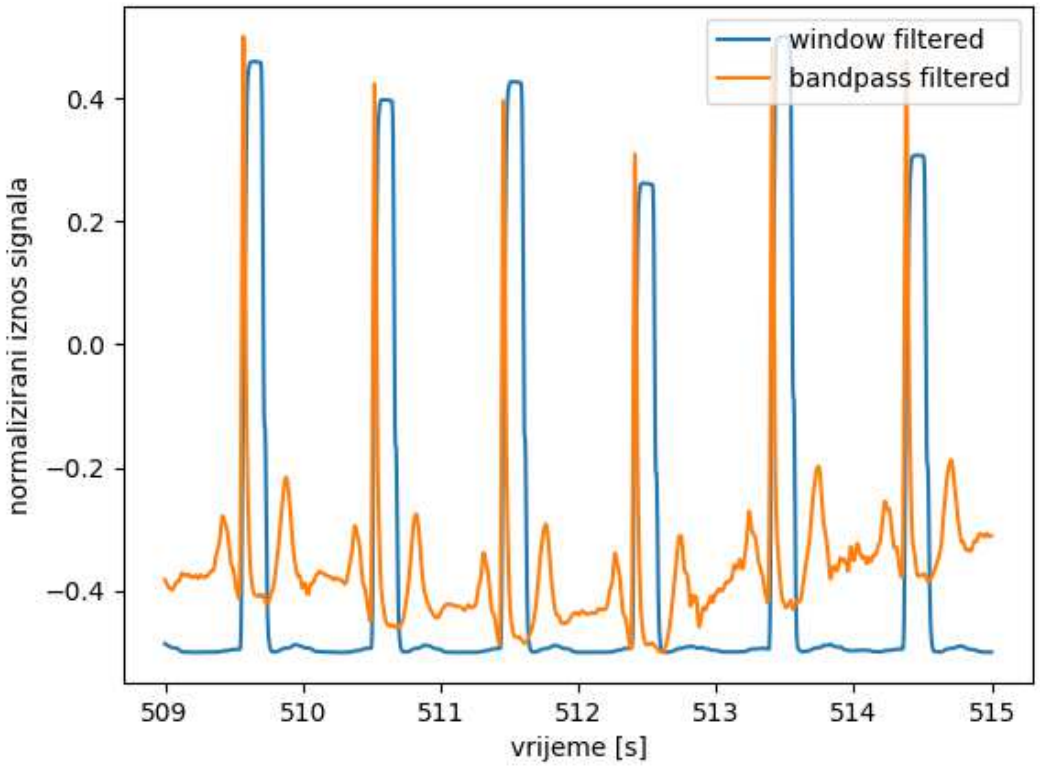
Nakon ovog postupka, dobiva se signal čiji je oblik prikazan na slici 4.5. Svaki rastući brid u tom signalu odgovara jednom QRS kompleksu (slika 4.6. te slika 5. u [2]). Algoritam će prepoznavati maksimume u dobivenom signalu. U tu svrhu koristi nekoliko varijabli: razinu signala, razinu šuma i graničnu vrijednost. Potrebno je razdoblje učenja kako bi se te varijable pravilno inicijalizirale. U ovom je radu korišteno razdoblje učenja od 2 sekunde, tj. 256 uzoraka. Kao početne vrijednosti varijabli uzimaju se:

$$\text{razina šuma} := \frac{1}{2} \cdot \frac{x[0] + x[1] + \cdots + x[255]}{256}$$

$$\text{razina signala} := \frac{1}{4} \cdot \max(x[0], x[1], \ldots, x[255])$$

$$\text{granica} := \text{razina šuma} + \frac{1}{4} \cdot (\text{razina signala} - \text{razina šuma}),$$

gdje je x signal dobiven integriranjem pomičnim prozorom. Sada se svaki potencijalni maksimum (uzorak $x[i]$ za koji vrijedi $x[i-1] < x[i] > x[i+1]$) uspoređuje s granič-



Slika 4.6. Položaj rastućih bridova u odnosu na QRS komplekse filtriranog signala

nom vrijednošću. Ako je veći od nje, zaista se radi o maksimumu koji odgovara QRS kompleksu, a ako nije, radi se o šumu. Nakon toga moraju se ažurirati pomoćne varijable. Ako je uzorak $x[i]$ iznad granične vrijednosti, tada:

$$\text{razina signala} := \frac{1}{8} \cdot x[i] + \frac{7}{8} \cdot \text{razina signala},$$

a ako je ispod granične vrijednosti, tj. ako je posljedica šuma:

$$\text{razina šuma} := \frac{1}{8} \cdot x[i] + \frac{7}{8} \cdot \text{razina šuma}.$$

Granična se vrijednost u oba slučaja ažurira po formuli:

$$\text{granica} := \text{razina šuma} + \frac{1}{4} \cdot (\text{razina signala} - \text{razina šuma}).$$

Nakon toga, točnije u narednih 200 ms, prepoznavanje QRS kompleksa je onemogućeno zbog toga što je u tom razdoblju fiziološki nemoguće da se dogodi novi otkucaj.

Ovaj je algoritam pojednostavljen u odnosu na algoritam opisan u [2]. Naime, potonji istovremeno promatra signal dobiven integracijom pomičnim prozorom i signal dobiven prolaskom kroz niskopropusni i visokopropusni filtar, te prepoznaje QRS kompleks samo ako je odgovarajući maksimum pronađen u oba promatrana signala. U ovom radu pokazalo se dovoljno pronaći maksimume u signalu dobivenom integracijom pomičnim prozorom. Također valja napomenuti da iako rastući brid odgovara QRS kompleksu u filtriranom signalu, traženje maksimuma umjesto rastućih bridova pokazalo se robusnijim. Naravno, kako bi se položaj QRS-a, tj. rastućeg brida, odredio pravilno, od položaja maksimuma trebat će oduzeti polovicu širine prozora integriranja, tj. 9 uzoraka. Dodatno, s obzirom da se QRS označava na originalnom EKG signalu, trebat će oduzeti i ukupno kašnjenje filtara koje iznosi 23 uzorka.

Za praćenje srčanog ritma (RR intervala), algoritam prati dva RR prosjeka $\overline{RR}_1$ i $\overline{RR}_2$. Prvi je definiran kao prosjek trajanja zadnjih 8 RR intervala bez obzira na njihove vrijednosti, a drugi je definiran kao prosjek zadnjih 8 RR intervala čija su trajanja unutar granica određenih pomoćnim varijablama:

$$RR_L = 0,92 \cdot \overline{RR}_2$$

$$RR_H = 1,16 \cdot \overline{RR}_2.$$

Svrha računanja dvaju prosjeka je mogućnost da se prilagodi brzim promjenama u otkucajima ili nepravilnim otkucajima, no za normalni ritam će vrijediti $\overline{RR}_1 = \overline{RR}_2$.

Potrebno je još promotriti kako riješiti lažno pozitivne i lažno negativne rezultate. Pan-Tompkins algoritam rješava lažno negativne, tj. promašene QRS komplekse na način da koristi drugu graničnu vrijednost:

$$\text{granica}' = \frac{1}{2} \cdot \text{granica},$$

i pomoćnu varijablu:

$$RR_M = 1,66 \cdot \overline{RR}_2.$$

Ako u vremenu RR_M nije pronađen nijedan QRS kompleks, tada se algoritam vraća unatrag i traži ga na isti način kao što inače prepoznaje QRS komplekse, samo što koristi

drugu graničnu vrijednost. U implementaciji je i ovaj korak izbačen jer se ispostavilo da algoritam nema poteškoća s lažno negativnim rezultatima. No, ono s čime ima problema su lažno pozitivni rezultati, ponajprije uzrokovani T valovima. Odbacivanje T valova radi se na idući način. Ako je prepoznat QRS kompleks, a prošlo je manje od 360 ms od prethodnog QRS kompleksa (i naravno, više od 200 ms u kojima se ne može dogoditi otkucaj), potencijalno se radi o T valu. Tada se računa najveći nagib u zadnjih 80 ms (što je polovica širine T vala [12]) i ako je on manji od polovine najvećeg nagiba u zadnjem QRS kompleksu, radi se o T valu, a ako je veći, radi se o novom otkucaju. Nagibi su već dobiveni prolaskom signala kroz filtar derivacije, stoga se ova provjera jednostavno implementira.

4.1.5. Mjerenje varijabilnosti srčanog ritma

Postoje razne mjere varijabilnosti srčanog ritma [13]. U ovom se radu koristi standardna devijacija RR intervala (SDRR) izražena u milisekundama. S obzirom da mikrokontroler obrađuje uzorke u stvarnom vremenu, potrebno je na *online* način računati varijancu. Taj problem rješava Welfordov algoritam koji to obavlja efikasno i uz očuvanje numeričke stabilnosti, i to pomoću rekurzivne relacije:

$$M_{2,n} = M_{2,n-1} + (x_n - \overline{x}_{n-1})(x_n - \overline{x}_n), \quad M_{2,0} = 0.$$

Za praćenje srednje vrijednosti služi rekurzivna relacija:

$$\overline{x}_n = \overline{x}_{n-1} + \frac{x_n - \overline{x}_{n-1}}{n}.$$

Varijanca uzoraka se iz varijable M_2 računa kao:

$$s_n^2 = \frac{M_{2,n}}{n-1}.$$

U odsječku 4..6 dan je primjer implementacije Welfordovog algoritma.

```
float running_variance(float new_val) {
    static int count = 0;
    static float mean = 0.0f, M2 = 0.0f;
```

```
    count++;
    float delta = new_val - mean;
    mean += delta / count;
    float delta2 = new_val - mean;
    M2 += delta * delta2;
    if (count >= 2)
        return sqrtf(M2 / (count - 1));
    else
        return 0.0f;
}
```

Odsječak koda 4..6 Implementacija Welfordovog algoritma

4.1.6. Mjerenje ritma disanja

Metode za procjenu ritma disanja iz EKG signala su brojne, a usporedba nekih može se pronaći u [14]. Implementacija metoda iz [14] u ovom radu nije bila uspješna, stoga je odabrano procijeniti disanje pomoću RSA (engl. *Respiratory sinus arrythmia*), pojave zbog koje je ritam srca brži pri udisaju nego pri izdisaju [15]. Iako su metode navedene u [14] pouzdanije, s obzirom na to da je RSA izražena najviše kod mladih i zdravih ljudi [16], u ovom se radu koristi procjena disanja pomoću RSA zbog jednostavnosti implementacije uz već obavljeno prepoznavanje RR intervala. Kao parametar za procjenu disanja se stoga koristi razlika između trajanja trenutnog RR prosjeka i posljednjeg RR intervala.

4.2. Programska potpora za računalo

Mikrokontroler je spojen putem USB-a na računalo. S obzirom da korištena pločica ima na sebi ugrađeni ST-Link preko kojega se *USART2* periferija mikrokontrolera prosljeđuje na virtualni COM port (kao što je rečeno u 3.2.), moguće je koristiti modul `serial` u programskom jeziku Python kako bi se čitali podatci. Na početku je potrebno pročitati i odbaciti sve bajtove dok se ne prepozna niz znakova `"!!!!"` koji označava početak komunikacije, kako je opisano u potpoglavlju 4.1.3. Nakon toga se čitaju segmenti po 17 bajtova te se svaki od njih rastavlja na podatke koje sadrži koristeći modul `struct`, suk-

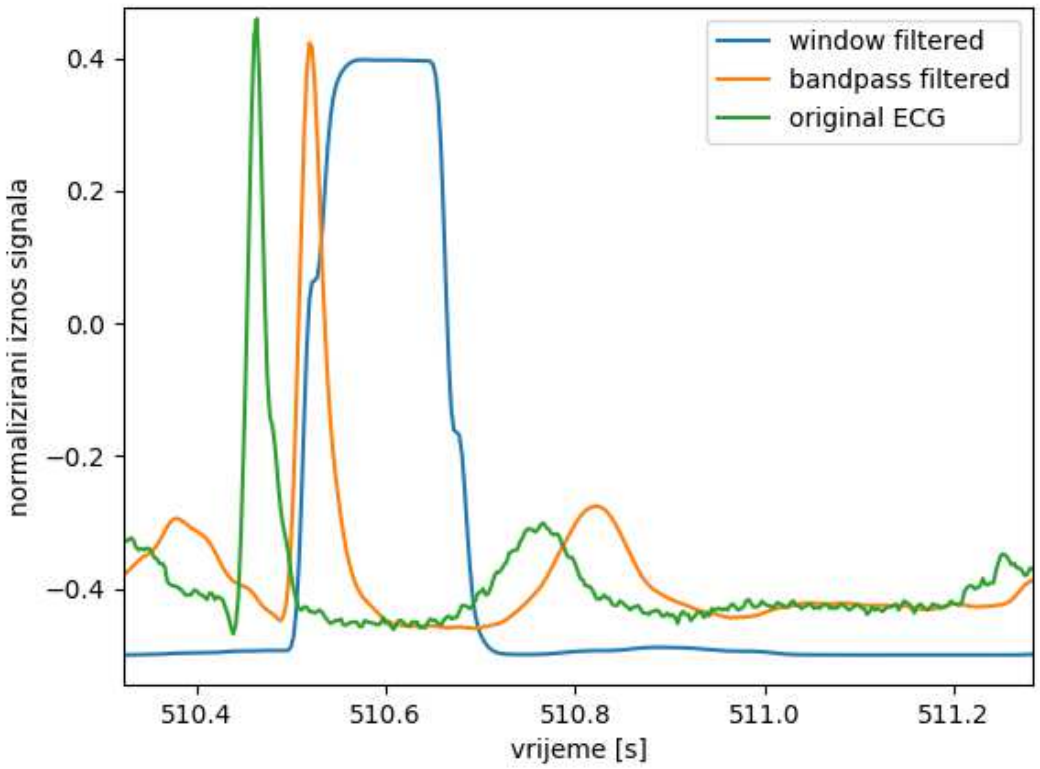
ladno kodu iz odsječka 4..7 Bitno je napomenuti da taj modul, osim ako je zadano drukčije, funkcionira na isti način kao i `struct` u programskom jeziku C, u smislu da dodaje *padding* kako bi udaljenost u bajtovima svakog podatka od početka niza bajtova bio višekratnik broja 4 [17][18]. *Padding* se ne dodaje na početak i na kraj strukture. Stoga, pokušaj dekodiranja niza od 17 bajtova s formatom `(int32_t, float, int32_t, int32_t, uint8_t)` funkcionira, ali s formatom `(int32_t, uint8_t, float, int32_t, int32_t)` ne uspijeva uz pogrešku `"struct.error: unpack requires a buffer of 20 bytes"`.

```
import serial
import struct

ser = serial.Serial(
    port='COM4',
    baudrate=115200,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
)

ser.read_until("!!!!".encode("ASCII"))

while True:
    while self.ser.in_waiting >= 17:
        data = self.ser.read(17)
        sample, hrv, edr, rr, is_peak = struct.unpack('ifiiB', data)

        # ...
```

Odsječak koda 4..7 Čitanje podataka sa serijskog priključka

Za implementaciju korisničkog sučelja koristi se biblioteka *PyQtGraph* [19]. *PyQtGraph* je biblioteka za crtanje grafova i prikazivanje grafičkih korisničkih sučelja namijenjena za primjene u matematici, znanosti i inženjerstvu. Njezin se opis može pronaći u [20]. Interno se temelji na upotrebi biblioteke *NumPy* [21] za obavljanje izračuna i biblioteke *PyQt* [22] za konstruiranje grafičkog korisničkog sučelja.

Napravljeno sučelje prikazano je na slici 4.8., a izvorni je kod prilagođen iz [23]. Sastoji se od četiriju grafova u *grid* razmještaju koji se ažuriraju u stvarnom vremenu: EKG signal s označenim prepoznatim QRS kompleksima, prosjek srčanog ritma, varijabilnost srčanog ritma te procjena disanja. Svi se grafovi crtaju jednostavno kao linijski grafovi iz pročitanih podataka, a prepoznati QRS kompleksi prikazani su križićima na EKG grafu. Za označavanje prepoznatih QRS kompleksa na grafu koristi se *boolean* zastavica koju mikrokontroler postavlja kada je obrada trenutnog uzorka detektirala otkucaj. No, postavljena zastavica nakon obrade trenutnog uzorka ne znači da je trenutni uzorak R zubac, već da se R zubac nalazi određeni broj uzoraka prije zbog toga što filtri unose kašnjenje u signal, što se jasno vidi iz slike 4.7. Kao što je opisano u potpoglavlju 4.1.4., taj broj uzoraka iznosi 32. Stoga kada je zastavica postavljena, na grafu treba staviti oznaku na mjesto $(t[n-32], y[n-32])$, gdje je t niz trenutaka uzorkovanja, a y niz uzoraka EKG signala, kao što je prikazano u odsječku 4..8



Slika 4.7. Kašnjenje signala uzrokovano filtrima

```
self.time.append(self.time[-1] + self.time_step)
self.ecg_y.append(sample)
```

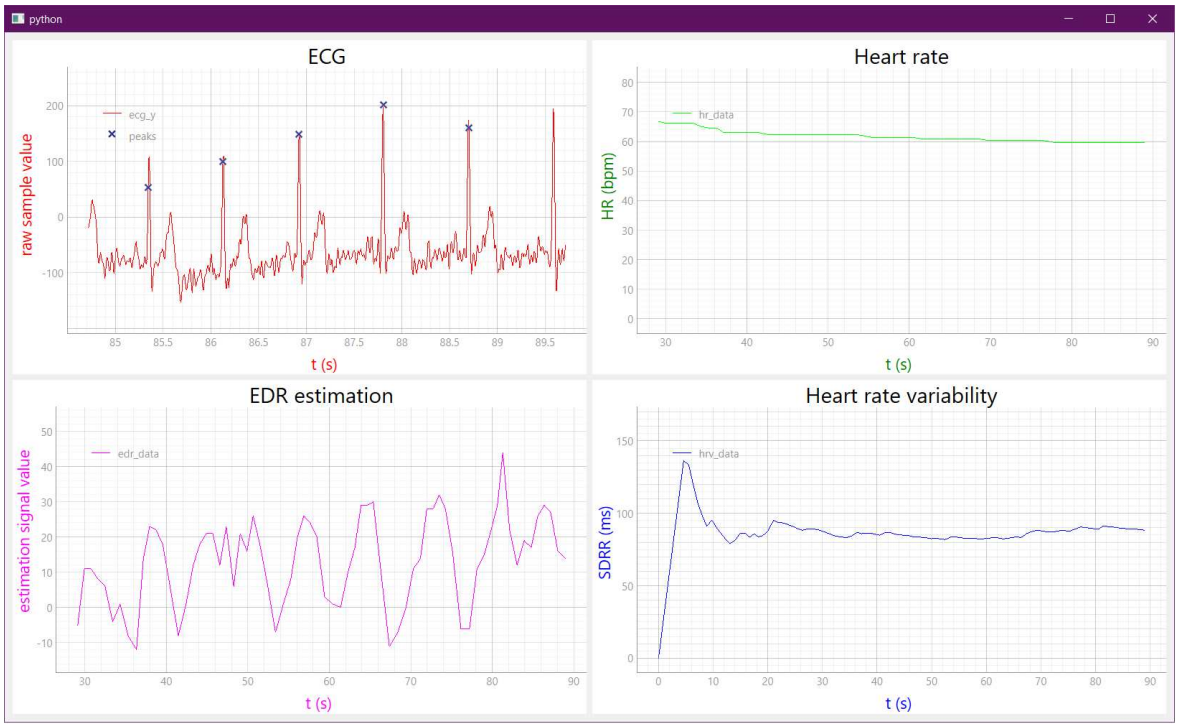
```
if len(self.time) >= self.DELAY_SAMPLES and is_peak:
    self.peaks_x.append(self.time[-self.DELAY_SAMPLES])
    self.peaks_y.append(self.ecg_y[-self.DELAY_SAMPLES])
```

Odsječak koda 4..8 Pravilno dodavanje oznake otkucaja na graf

Također treba imati na umu da je frekvencija ažuriranja grafova različita od frekvencije kojom mikrokontroler šalje podatke, što znači da se ne smije za svako ažuriranje grafova pročitati jedan segment od 17 bajtova sa serijskog priključka, već se treba čitati 17 po 17 bajtova dok god je to moguće, tj. dok se ne pročitaju svi u međuvremenu pristigli segmenti. Konačno, kako bi graf bio čitljiv, podatci stariji od nekog vremena se odbacuju, tj. prikazuje se samo prozor određene vremenske širine na grafu, a ne cijela povijest od početka mjerenja. Naravno, za graf EKG signala bi ovaj prozor trebao biti kraći kako bi se jasno vidio valni oblik te prepoznati otkucaji, a za srčani ritam i njegovu varijabilnost bi prozor trebao biti duži jer su to iznosi koji se najbolje prate kroz duže vrijeme.

Slika 4.8. Izgled korisničkog sučelja

5. Rezultati i rasprava



Slika 5.1. Prikaz rezultata na korisničkom sučelju

Primjer rezultata prikazan je na slici 5.1. Pan-Tompkinsov algoritam vrlo pouzdano prepoznaje otkucaje. Iako bi se na dovoljno dugom razdoblju vjerojatno pojavili, pri testiranju uopće nisu primijećeni lažno negativni otkucaji, što znači da je njihova učestalost pojavljivanja izuzetno mala. Lažno pozitivni otkucaji su primijećeni vrlo rijetko. Također, sam položaj R zubaca biva određen s visokom točnošću. Naravno, povremeno se dogodi da oznaka R zubca bude postavljena na dnu Q ili S vala ili pak na sredini R zubca, ali generalno gledajući to je i dalje vrlo dobra točnost s obzirom na to da je sam R zubac vrlo uzak te se ne radi o velikoj vremenskoj razlici. Posljedica pouzdanosti Pan-Tompkins algoritma za detekciju otkucaja jest i točnost praćenja srčanog ritma i njegove varijabilnosti. Na primjeru na slici srčani ritam iznosi oko 60 otkucaja po minuti, što je očekivana vrijednost, a varijabilnost srčanog ritma kreće se od 50 do 100 ms, što također

okvirno odgovara očekivanim vrijednostima iz [1].

Procjena disanja iz EKG signala pokazuje slabu točnost rezultata. Iz valnog oblika na slici 5.1. mogu se ugrubo iščitati udisaji i izdisaji, ali to je postignuto uz spore i duboke udisaje i izdisaje (na slici je 8 ili 9 udisaja/izdisaja u jednoj minuti). Uz normalan ili brži ritam disanja, signal nije dovoljno čist da bi se iz njega mogao razaznati ritam disanja. Jedan način na koji bi se mogli postići bolji rezultati jest obradom ovog signala. Problem je u tome što frekvencija uzorkovanja nije pravilna, već ovisi o srčanom ritmu jer se signal procjene disanja uzorkuje na svaki otkucaj srca. To se teoretski može riješiti interpoliranjem signala te ponovnim uzorkovanjem većom (i konstantnom) frekvencijom, kao što je u radu [14] učinjena interpolacija kubičnim splajnovima nad signalom koji se sastoji od trajanja RR intervala. Slično bi se moglo primijeniti i na ovaj signal, te zatim nad dobivenim signalom primijeniti pojasno propusni filtar kojim bi se odbacile komponente s frekvencijama izvan pojasa normalnih frekvencija disanja. Mana ovakvog poboljšanja jest u tome što se uvodi kašnjenje jer bi se trebao čekati novi uzorak (otkucaj), interpolirati s novim uzorkom te provesti sve novodobivene uzorke interpoliranog signala kroz filtar, a dodatno sama interpolacija kubičnim splajnom podrazumijeva rješavanje linearnog sustava, stoga sve navedeno predstavlja relativno veliko računsko opterećenje za izvođenje na mikrokontroleru u stvarnom vremenu.

Što se tiče sklopovske potpore, sam mikrokontroler pokazao se dovoljnim u smislu izračunske moći i nije bilo poteškoća s kašnjenjem ili gubitkom uzoraka zbog njegove sporosti. Također, sklop *MAX30003* bio je dovoljan i pravilno je davao uzorke čak i bez korištenja mogućnosti efikasnijeg prijenosa podataka u *burst* načinu rada [6]. Ovo je bilo i za očekivati s obzirom na to da je frekvencija rada jezgre mikrokontrolera 80 MHz, a sklop *MAX30003* uzorkuje EKG frekvencijom 128 Hz, što je čak pet redova veličine manja frekvencija, te je stoga nemoguće da mikrokontroler propusti neki uzorak ili da dođe do preljeva FIFO spremnika na sklopu. Programsku potporu za mikrokontroler moguće je unaprijediti tako da se sa sklopom *MAX30003* radi u prekidnom načinu rada. U ovoj implementaciji mikrokontroler nije imao druge zadatke te je stoga mogao uzorke dohvaćati radnim čekanjem, no u slučaju potrebe za višezadaćnošću, prekidni način rada bio bi neophodan. Također, tada bi bilo moguće iskoristiti i prekide koje *MAX30003* daje na prepoznate otkucaje srca pomoću interno implementiranog Pan-Tompkins algoritma, te

dobivene otkucaje usporediti s onima koji su dobiveni algoritmom u softveru mikrokontrolera.

Programska potpora za računalo je također zadovoljila sve potrebe. Grafovi su bili ažurirani u stvarnom vremenu uz vizualno prihvatljivu brzinu, ali i dovoljnu brzinu da ne dolazi do preljeva ulaznog spremnika i gubitka podataka koji dolaze s mikrokontrolera. Mogućnost prikaza četiriju grafova u jednom prozoru pokazala se vrlo praktičnom, a sve je ostvareno uz relativno jednostavan i jasan kod. Jedno moguće proširenje za korisničko sučelje jest opcija kojom se pristigli podatci mogu snimati u datoteku. Također, korisno poboljšanje bila bi i mogućnost *debug* načina rada koji bi prikazivao pomoćne varijable iz Pan-Tompkinsovog algoritma, tj. razinu signala i šuma te granicu, kako bi se vidjelo kako se oni mijenjaju, što bi bilo korisno za slučajeve visokih smetnji.

6. Zaključak

U ovom radu realiziran je sustav za procjenu parametara srčanog ritma u stvarnom vremenu, temeljen na vlastitoj razvijenoj, jednostavnoj i pouzdanoj implementaciji Pan-Tompkins algoritma na mikrokontroleru. Izvedba omogućuje procjenu srčanog ritma, varijabilnosti srčanog ritma i ritma disanja. Rezultati mjerenja pokazali su da implementirani algoritmi mogu pouzdano odrediti srčani ritam i varijabilnost srčanog ritma, ali da trenutna implementacija nije u mogućnosti dati kvalitetnu procjenu ritma disanja te da bi za bolje performanse trebalo implementirati složenije algoritme obrade signala. Izračun varijabilnosti srčanog ritma realiziran je korištenjem Welfordovog algoritma za *online* računanje varijance, koji omogućuje efikasan izračun bez numeričke nestabilnosti, koristeći rekurzivne relacije. Predložena su potencijalna rješenja za poboljšanje procjene ritma disanja iz EKG signala. Rad se fokusirao primarno na razvoj algoritama prikladnih za implementaciju na mikrokontroleru, pri čemu su se koristile gotove sklopovske komponente za mjerenje EKG signala i obradu na mikrokontroleru, koje su međusobno komunicirale putem protokola SPI. Komunikacija s osobnim računalom izvedena je korištenjem UART protokola putem USB sučelja, a razvijena programska potpora za osobno računalo omogućuje jednostavnu kontrolu i praćenje rada razvijenog rješenja.

Ovaj rad predstavlja podlogu za daljnje istraživanje i razvoj jer omogućuje jednostavniju implementaciju drugih algoritama obrade EKG signala bez potrebe da se cjelokupna programska potpora za mikrokontroler razvija ispočetka. Također, korisničko sučelje razvijeno za osobno računalo omogućuje lakši rad sa sustavom, a može se koristiti i za druge tipove biomedicinskih signala ili druge primjene izvan ovog područja.

Literatura

[1] "Heart rate variability", https://en.wikipedia.org/wiki/Heart_rate_variability, [mrežno, pristup 14.06.2024.].

[2] J. Pan i W. J. Tompkins, "A Real-Time QRS Detection Algorithm", *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, sv. BME-32, br. 3, str. 230–236, ožujak 1985.

[3] "Electrocardiography", https://en.wikipedia.org/wiki/Electrocardiography, [mrežno, pristup 13.06.2024.].

[4] A. Butchy, U. Jain, M. Leasure, V. Covalesky, i G. Mintz, "Importance of Electrode Selection and Number in Reconstructing Standard Twelve Lead Electrocardiograms", *Biomedicines*, sv. 11, str. 1526, svibanj 2023. https://doi.org/10.3390/biomedicines11061526

[5] A. Atkielski, "Schematic diagram of normal sinus rhythm for a human heart as seen on ECG", https://en.wikipedia.org/wiki/File:SinusRhythmLabels.png, [mrežno, pristup 13.06.2024.].

[6] Analog Devices, "Ultra-Low Power, Single-Channel Integrated Biopotential (ECG, R-to-R Detection) AFE", https://www.analog.com/media/en/technical-documentation/data-sheets/MAX30003.pdf, [mrežno, pristup 25.05.2024.].

[7] "ECG 3 Click", https://www.mikroe.com/ecg-3-click, [mrežno, pristup 25.05.2024.].

[8] "STM32 Nucleo-L412KB - slika", https://www.st.com/bin/ecommerce/api/image.PF266995.en.feature-description-include-personalized-no-cpn-large.jpg, [mrežno,

pristup 13.06.2024.].

[9] M. Srpak, “Bežični čvor za mjerenje EKG signala temeljen na sklopu MAX30003 i ESP32 platformi”, srpanj 2023.

[10] STMicroelectronics, “User manual UM1956”, https://www.st.com/resource/en/user_manual/um1956-stm32-nucleo32-boards-mb1180-stmicroelectronics.pdf, [mrežno, pristup 31.05.2024.].

[11] W. J. Tompkins, “A Real-Time QRS Detection Algorithm - Errata”, srpanj 1985.

[12] L. Rosenthal, “Normal Electrocardiography (ECG) Intervals”, https://emedicine.medscape.com/article/2172196-overview, [mrežno, pristup 02.07.2024.].

[13] F. Shaffer i J. P. Ginsberg, “An Overview of Heart Rate Variability Metrics and Norms”, *Frontiers in Public Health*, sv. 5, rujan 2017. https://doi.org/10.3389/fpubh.2017.00258

[14] C. Varon, J. Morales, J. Lázaro, M. Orini, M. Deviaene, S. Kontaxis, D. Testelmans, B. Buyse, P. Borzée, L. Sörnmo, P. Laguna, E. Gil, i R. Bailón, “A Comparative Study of ECG-derived Respiration in Ambulatory Monitoring using the Single-lead ECG”, *Scientific Reports*, sv. 10, br. 1, str. 5704, ožujak 2020. https://doi.org/10.1038/s41598-020-62624-5

[15] F. Yasuma i J.-I. Hayano, “Respiratory sinus arrhythmia”, *Chest*, sv. 125, br. 2, str. 683–690, veljača 2004.

[16] G. B. Moody, “ECG-Derived Respiration”, https://archive.physionet.org/physiotools/edr/, [mrežno, pristup 28.05.2024.].

[17] “The Python Standard Library”, https://docs.python.org/3/library/struct.html, [mrežno, pristup 03.07.2024.].

[18] “Why does mixing types in Python struct.pack uses more space than needed?” https://stackoverflow.com/questions/21332956/why-does-mixing-types-in-python-struct-pack-uses-more-space-than-needed, [mrežno, pristup 03.07.2024.].

[19] “PyQtGraph - Scientific Graphics and GUI Library for Python”, https://www.pyqtgraph.org/, [mrežno, pristup 03.07.2024.].

[20] “PyQtGraph dokumentacija”, https://pyqtgraph.readthedocs.io/en/latest/index.html, [mrežno, pristup 03.07.2024.].

[21] “NumPy”, https://numpy.org/, [mrežno, pristup 03.07.2024.].

[22] “Qt for Python”, https://doc.qt.io/qtforpython-6/index.html, [mrežno, pristup 03.07.2024.].

[23] “Plotting With PyQtGraph”, https://www.pythonguis.com/tutorials/pyqt6-plotting-pyqtgraph/, [mrežno, pristup 03.07.2024.].

Sažetak

Analiza EKG-signala prikupljenog integriranim sklopom MAX30003 u stvarnom vremenu

Eugen Bošnjak

Razvijena je programska potpora za mikrokontroler za obradu EKG signala u stvarnom vremenu, dobivenog iz vanjskog sklopa. Programska potpora izračunava srčani ritam (HR), varijabilnost srčanog ritma (HRV) i ritam disanja (RR). Izračunate vrijednosti šalju se serijskim sučeljem prema osobnom računalu gdje se prikazuju u grafičkom korisničkom sučelju implementiranom u programskom jeziku Python. Točnost izračuna srčanog ritma i varijabilnosti srčanog ritma je visoka, dok je točnost procjene ritma disanja značajno manja zbog potrebe za korištenjem bitno naprednijih algoritama obrade signala. Sklopovska potpora sastoji se od razvojne pločice *Nucleo-L412KB* s mikrokontrolerom *STM32L412KB* temeljenim na arhitekturi *Arm Cortex-M*, razvojne pločice *ECG 3 Click* s integriranim sklopom *MAX30003*, koji sadrži analogno sučelje za mjerenje EKG signala, te osobnog računala.

Ključne riječi: EKG, biomedicinska elektronika, nosivi elektronički uređaj, digitalna obrada signala

Abstract

Real-time analysis of the ECG signal acquired by the MAX30003 integrated circuit

Eugen Bošnjak

Software for microcontroller for real-time processing of ECG signal acquired via an external circuit was developed. Software calculates heart rate (HR), heart rate variability (HRV) and respiratory rate (RR). Calculated values are sent via a serial interface to a personal computer, where these values are displayed in a graphical user interface implemented in the Python programming language. The accuracy of the calculation of the heart rate and heart rate variability is high, while the accuracy of the estimation of the respiration rate is significantly lower due to the need for using significantly more advanced signal processing algorithms. The circuit consists of a development board *Nucleo-L412KB* with a microcontroller *STM32L412KB* based on the *Arm Cortex-M* architecture, a development board *ECG 3 Click* with an integrated circuit *MAX30003* , which contains an analog interface for measuring ECG signals, and a personal computer.

Keywords: ECG, biomedical electronics, wearable electronics, digital signal processing