SADRŽAJ

1. Uvod

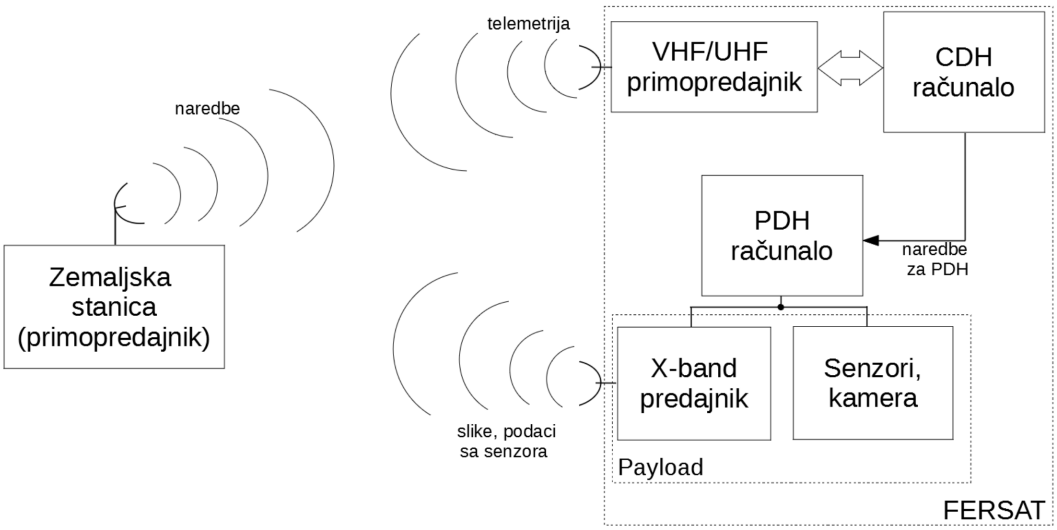
Projekt FERSAT, koji se od 2018. godine provodi na Fakultetu elektrotehnike i računarstva, uključuje izradu i lansiranje CubeSat satelita te korištenje satelita u svrhu prikupljanja informacija o svjetlosnom zagađenju i debljini ozonskog omotača. Satelit u izradi dimenzija je približno 10 cm x 10 cm x 10 cm, volumena jedne litre i ne teži od 4/3 kilograma, što ga svrstava u skupinu satelita formata CubeSat 1U [13]. Očekivani životni vijek satelita je 3 godine, a bit će postavljen u Zemljinoj orbiti na visini između 500 i 600 kilometara. Planirani korisni teret (engl. *payload*) FERSAT-a podijeljen je na tri podsustava:

- kamera za snimanje površine Zemlje i zemaljskog horizonta,
- detektori svjetla u vidljivom i ultraljubičastom dijelu spektra za mjerenje svjetlosnog onečišćenja i debljine stupca ozona,
- komunikacijski sustav u radijskom X-pojasu (10.45 GHz) za prijenos podataka na Zemlju.

Radom korisnog tereta upravlja *Payload Data Handler* (PDH) računalo. Zadaća je PDH računala prikupiti podatke iz senzorskog podsustava i kamere, pohraniti ih u trajnu memoriju (engl. *non-volatile memory*) te poslati prikupljene podatke na Zemlju korištenjem komunikacijskog podsustava. Kao PDH računalo odabran je mikrokontroler STM32L4P5VET6 proizvođača ST Microelectronics.

Za rad ostalih podsustava satelita koji nisu direktno vezani uz koristan teret (npr. upravljanje položajem satelita, slanje telemetrijskih podataka na Zemlju) brine se *Command and Data Handler* (CDH) računalo. CDH računalo također upravlja napajanjem korisnog tereta i šalje naredbe PDH računalu. Komunikacija CDH i PDH računala odvija se korištenjem sučelja CAN (*Controller Area Network*). Konkretno CDH računalo u trenutku pisanja ovog teksta još nije odabrano.

Slika 1.1 prikazuje blok dijagram cijelog sustava. U okviru ovog rada razvijena je programska potpora PDH računala za prikupljanje i obradu podataka senzorskog podsustava.



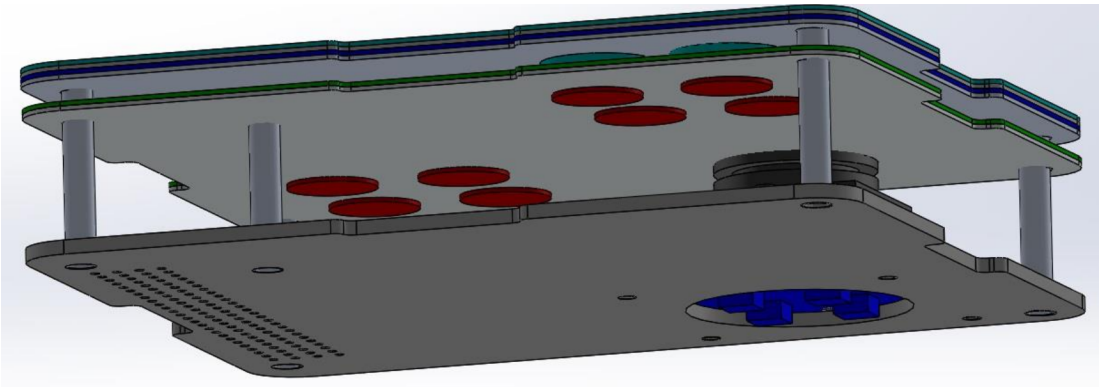
Slika 1.1: Blok dijagram FERSAT-a i komunikacija sa zemaljskom postajom [5]

Senzorski podsustav ima dvije temeljne zadaće. Prva od njih je korištenjem fotosenzora koji rade u vidljivom dijelu elektromagnetskog spektra prikupiti podatke na temelju kojih će biti moguće odrediti udio LED rasvjete u naseljenim mjestima u odnosu na konvencionalnu natrijevu, metal-halidnu i fluorescentnu javnu rasvjetu. Razvijen je algoritam koji na temelju obrade signala multispektralnog svjetla sa Zemlje može odrediti ovu informaciju [11]. Mjerenje udjela LED rasvjete zanimljivo je zbog mogućih negativnih utjecaja plavog svjetla na ljudsko zravlje, kojeg LED rasvjeta emitira u znatno većem intenzitetu nego konvencionalna [3].

Druga je zadaća senzorskog podsustava mjerenje propusnosti i refleksije atmosfere za ultraljubičasto svjetlo u svrhu određivanja debljine ozonskog omotača. Za mjerenje se koriste PureB detektori ultraljubičastog zračenja razvijeni na FER-u [2] i algoritmi razvijeni za tu namjenu [9]. Uspješna mjerenja po prvi put bi potvrdila mogućnost korištenja ove tehnologije u mjerenjima debljine ozonskog omotača iz svemira.

Upravljačko sklopovlje potrebno za rad PDH računala već je razvijeno [4]. Tiskana pločica PDH računala, osim mikrokontrolera STM32L4P5VET6, sadrži i vanjsku *Flash* memoriju, sustav za napajanje, sklop za kontrolu izvođenja programa (engl. *watchdog*), upravljački sklop za CAN komunikaciju i konektore za povezivanje s ostalim dijelovima sustava. PDH pločica bit će smještena ispod senzorske pločice, u takozvanoj *stack-up* konfiguraciji (slika 1.2).

Također, u sklopu projekta FERSAT razvijen je i dio programske potpore PDH računala [5]. No, kako je u međuvremenu došlo do promjene izbora mikrokontrolera PDH računala i promjene dijela sklopovlja senzorskog podsustava, dijelove te pro-



Slika 1.2: Trodimenzionalni model korisnog tereta FERSAT-a. PDH računalo smješteno je na donjoj pločici, a senzorski podsustav na srednjoj [4].

gramske potpore bilo je potrebno prilagoditi ili ponovno razviti.

Nastavak rada strukturiran je na sljedeći način. U poglavlju 2 opisano je komunikacijsko sučelje između mikrokontrolera STM32L4P5VET6 i senzorskog podsustava (SPI - *Serial Peripheral Interface*). Detaljan opis sklopovlja senzorskog podsustava dan je u poglavlju 3. U poglavlju 4 opisani su razvijeni upravljački programi za pojedine sklopovske komponente, cjelokupna programska potpora za senzorski podsustav, i integracija s ostalim dijelovima programske potpore PDH računala korištenjem operacijskog sustava za rad u stvarnom vremenu FreeRTOS.

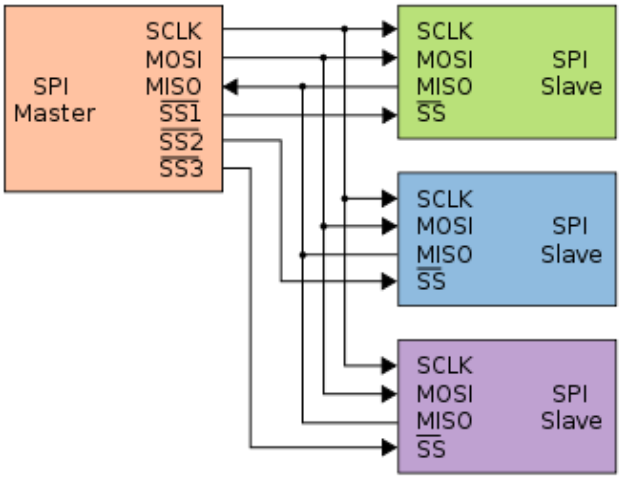
2. Sučelje za komunikaciju sa senzorskim podsustavom

Komponente senzorske pločice FERSAT-a, odnosno analogno-digitalni pretvornik (ADC) ADS131M08 i temperaturni senzori ADT7301 komuniciraju s PDH računalom putem sučelja SPI. S obzirom da se pri razvoju programske potpore PDH računala koriste *Low-Level* biblioteke, za ispravnu implementaciju upravljačkih programa nužno je razumijevanje strukture i načina rada SPI periferije izabranog mikrokontrolera. U nastavku ovog poglavlja dan je općeniti opis SPI protokola i opis SPI periferije porodice mikrokontrolera STM32L4.

2.1. SPI protokol

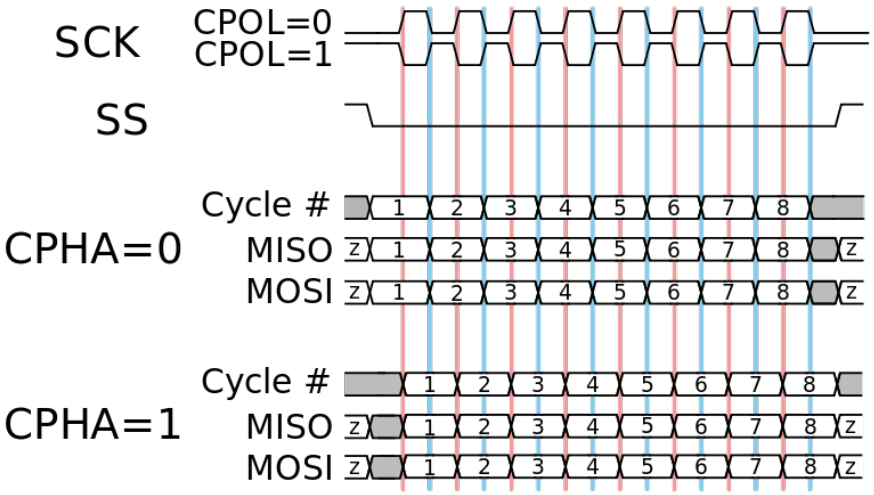
Serial Peripheral Interface (SPI) sinkrono je serijsko komunikacijsko sučelje, razvijeno u tvrtki Motorola. S obzirom da SPI sučelje omogućava brzinu prijenosa do nekoliko desetaka Mbit/s, obično se koristi za prijenos signala visokih frekvencija između računala ili mikrokontrolera i periferijskih sklopova. Podaci se prenose između jedne upravljačke jedinice (engl. *master*) i više upravljanih jedinica (engl. *slave*) korištenjem četiri prijenosne linije: SCLK (*Serial Clock*), MISO (*Master Input Slave Output*), MOSI (*Master Output Slave Input*) i CS (*Chip Select*, ponekad se naziva i *Slave Select*). Signal takta pogoni *master* uređaj, a pomoću linije CS *master* uređaj odabire koji *slave* uređaj smije komunicirati preko linija MISO i MOSI. Slika 2.1 prikazuje tipičan način spajanja uređaja SPI sučeljem, u konfiguraciji jednog *master* uređaja i tri *slave* uređaja.

Postoje 4 temeljna načina rada (engl. *modes*) SPI sučelja, a razlikuju se po polaritetu signala takta (engl. *Clock Polarity, CPOL*) i načinu čitanja podatka sa strane master uređaja (engl. *Clock Phase, CPHA*). CPOL određuje logičku razinu u koju će signal takta poprimiti u neaktivnom stanju (engl. *idle state*). CPOL=0 će postaviti nisku razinu, a CPOL=1 visoku. CPHA određuje na koji brid signala takta će *master*



Slika 2.1: Povezivanje uređaja SPI sučeljem [12]

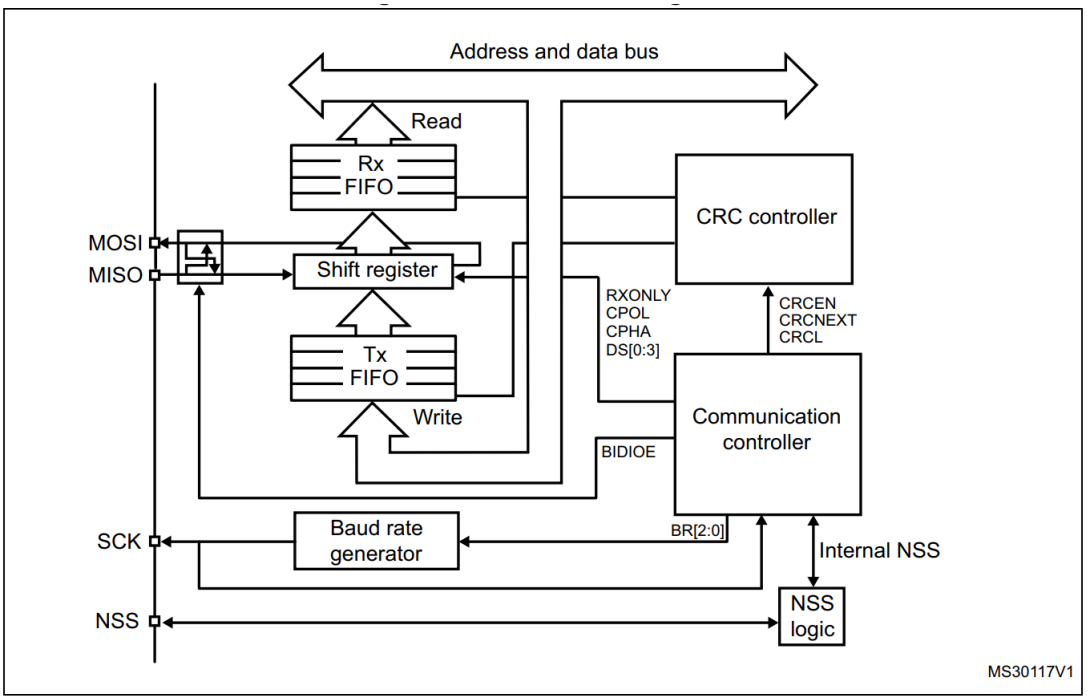
uređaj čitati podatak koji se nalazi na liniji. CPHA=0 obično znači da će to biti prvi brid, a CPHA=1 drugi. Ovisno o postavci CPOL, taj brid može biti rastući ili padajući. Ako je CPOL=0 prvi brid će biti rastući, a ako je CPOL=1, prvi brid će biti padajući. Slika 2.2 prikazuje vremenske odnose za različite postavke CPOL i CPHA.



Slika 2.2: Vremenski dijagram SPI prijenosa s različitim postavkama CPOL i CPHA. Crvene linije označavaju prvi, a plave drugi brid signala takta. [12]

2.2. Struktura SPI periferije STM32L4

Slika 2.3 prikazuje blok dijagram SPI periferije porodice mikrokontrolera STM32L4 [7].



Slika 2.3: Blok dijagram SPI periferije STM32L4 [7, str. 1451]

Primanje podataka odvija se na način da riječ koja se prima po MISO liniji prvo ulazi u posmačni registar (engl. *shift register*), pri čemu se na svaki period SPI takta posmiče za jedno mjesto. Kada je primljena cijela riječ, ona se na sljedeći brid takta prebacuje na kraj reda Rx FIFO *First In First Out*, pri čemu se postavlja zastavica RXNE (*Receiver Buffer Not Empty*) u registru stanja SPI periferije(SPIx_SR). Prvom podatku u redu programski se može pristupiti preko SPI podatkovnog registra (SPIx_DR). Čitanje ovog registra automatski čisti zastavicu RXNE ukoliko je popunjeno manje od četvrtine maksimalnog kapaciteta reda.

Slanje podataka odvija se na sličan način. Riječ upisana u SPIx_DR sprema se na kraj reda Tx FIFO. Prva riječ u redu prebacuje se u posmačni registar, te se izlazni bit pri svakom posmaku šalje po liniji MOSI. Ako Tx FIFO sadrži manje podataka od pola svog kapaciteta, postavlja se zastavica TXE (*Transmitter Buffer Empty*). Postavljanje te zastavice signalizira programu da se sljedeća riječ može upisati u red.

Važno je naglasiti da SPIx_DR nije fizički registar, već se radi o virtualnom registru koji služi za pristup redovima Rx FIFO i Tx FIFO. Pisanje u ovaj registar umeće podatak na kraj reda Tx FIFO, a čitanje sadržaja registra vraća prvi podatak u redu

Rx FIFO. Oba reda su veličine 32 bita, odnosno mogu primiti 4 8-bitne riječi. Nivo popunjenosti, tj. broj 8-bitnih riječi u redu može se dobiti čitanjem bitova FTLVL[1:0] za Tx FIFO, odnosno FRLVL[1:0] za Rx FIFO u SPIx_SR.

SPI kontroler može raditi s duljinama riječi od 4 do 16 bita. U izradi ovog rada korištena je duljina riječi 8 bita.

Kontroler omogućuje hardversko izračunavanje CRC zaštitnog koda. Ova je mogućnost nakon reseta isključena, no može se omogućiti postavljanjem bita CRCEN u registru SPIx_CR1. Tada će pogreška u prijenosu koju otkrije CRC biti signalizirana postavljanjem zastavice CRCERR u registru SPIx_SR.

2.3. Postupak slanja i primanja podataka

SPI sučelje može funkcionirati u nekoliko načina rada s obzirom na smjer komunikacije. Ti načini rada su: *Full Duplex Master, Full Duplex Slave, Half Duplex Master, Half Duplex Slave, Simplex Receive Only* i *Simplex Transmit Only*. U upravljačkim programima izrađenim u sklopu ovog rada korišten je način rada *Full Duplex Master*, pa će zato u nastavku ovog potpoglavlja biti opisan postupak kojeg upravljački program mora izvršiti za ispravnu komunikaciju u tom načinu rada.

Važno je naglasiti da je u *Full Duplex Master* načinu rada signal SPI takta određen slanjem podataka. SPI kontroler počinje generirati signal takta upisom prve riječi podatka u podatkovni registar, te nastavlja generirati signal takta dok sve riječi nisu poslane. Ako je zadnja riječ poslana i nema novih riječi u redu Tx FIFO, kontroler prestaje s generiranjem takta do sljedećeg upisa u podatkovni registar.

Upravljački program mora slijediti sljedeću proceduru[1]:

1. Omogućiti SPI postavljanjem bita SPE u registru SPIx_CR1.
2. Upisati prvu riječ za slanje u podatkovni registar.
3. Čekati dok se ne postavi zastavica TXE i zatim upisati sljedeću riječ u podatkovni registar. Čekati dok se ne postavi zastavica RXNE i zatim pročitati riječ iz podatkovnog registra. Ponavljati ovaj korak do (uključivo) predzadnje pročitane riječi.

[1]Ova procedura je prilagođena verzija procedure opisane u priručniku mikrokontrolerske porodice STM32F4 [6, str. 887].

4. Čekati dok se ne postavi zastavica RXNE i pročitati zadnju riječ iz podatkovnog registra.

Čekanje zastavice TXE prilikom upisa druge riječi nije obavezno jer će ona sigurno biti postavljena, odnosno Tx FIFO sigurno može primiti dvije 8-bitne riječi. Međutim, čekanje je potrebno za svaki sljedeći upis.

Gornja procedura implementirana je funkcijom `SPI_TransmitReceive()`, koja pruža praktično sučelje za SPI prijenos po uzoru na HAL biblioteke.

```
1   void SPI_TransmitReceive(SPI_TypeDef *SPIx, uint8_t len,
        uint8_t *tx_buffer, uint8_t *rx_buffer)
2   {
3     LL_SPI_Enable(SPIx);
4
5     LL_SPI_TransmitData8(SPIx, tx_buffer[0]);
6     for (int i = 0; i < len - 1; i++) {
7       while ( !LL_SPI_IsActiveFlag_TXE(SPIx) );
8       LL_SPI_TransmitData8(SPIx, tx_buffer[i + 1]);
9       while ( !LL_SPI_IsActiveFlag_RXNE(SPIx) );
10      rx_buffer[i] = LL_SPI_ReceiveData8(SPIx);
11    }
12
13    while ( !LL_SPI_IsActiveFlag_RXNE(SPIx) );
14    rx_buffer[len - 1] = LL_SPI_ReceiveData8(SPIx);
15  }
```

Ako je nakon dovršenog SPI prijenosa potrebno staviti mikrokontroler u način rada male potrošnje, tada treba na ispravan način onemogućiti SPI. U nekim drugim SPI načinima rada potrebno je onemogućiti SPI nakon svakog prijenosa, npr. ako u *Receive Only Master* načinu rada mikrokontroler komunicira sa sklopom koji kontinuirano šalje podatke, kako bi se spriječilo slanje dodatnih neželjenih podataka. U *Full Duplex* načinu rada to obično nije potrebno, međutim dobra je praksa slijediti pravilnu proceduru za onemogućavanje SPI sučelja nakon završetka prijenosa kako bi se spriječila korupcija zadnje poslane riječi. Procedura za onemogućavanje je:

1. Čekati dok se ne isprazni odlazni red (TX FIFO) provjerom bitova FTLVL[1:0] u SPI statusnom registru.
2. Čekati dok se ne spusti zastavica BSY u SPI statusnom registru.
3. Onemogućiti SPI čišćenjem bita SPE u registru CR1.

Ova je procedura implementirana funkcijom `SPI_Disable()`:

```
1    void SPI_Disable(SPI_TypeDef *SPIx) {
2      while (LL_SPI_GetTxFIFOLevel(SPIx) != LL_SPI_TX_FIFO_EMPTY)
           ;
3      while (LL_SPI_IsActiveFlag_BSY(SPIx));
4      LL_SPI_Disable(SPIx);
5    }
```

Tijekom uhodavanja SPI komunikacije sa sklopom ADT7301 uočena je greška u implementaciji HAL biblioteke za rad s SPI periferijom kada je SPI u *Receive Only* načinu rada. Naime, u navedenom načinu rada potrebno je očistiti SPE bit točno jedan ciklus SPI takta nakon primitka predzadnje riječi kako bi se spriječilo da uređaj koji šalje podatak inicira prijenos nove riječi [6, str. 894]. HAL funkcija `HAL_SPI_TransmitReceive()` čisti SPE bit tek nakon primitka zadnje riječi, zbog čega senzor šalje još jednu riječ, pa ukupna duljina prijenosa iznosi 24 bita umjesto 16. Također, funkcija čeka da se sklopovski očisti zastavica RXNE prije nego što završi s izvođenjem, a s obzirom da se zadnja riječ nikad ne pročita, zastavica uvijek ostaje postavljena. Zbog toga funkcija uvijek čeka do isteka *timeout* intervala, što nije poželjno ponašanje. No, to nije predstavljalo problem u daljnem tijeku izrade ovog rada, jer su za razvoj korištene LL biblioteke i *Full Duplex* način rada.

2.4. Prijenos podataka korištenjem DMA sklopa

Prijenos podataka SPI sučeljem između PDH računala i AD pretvornika podrazumijeva slanje relativno velike količine podataka, što može prilično dugo trajati, ovisno o brzini prijenosa. Zato je korisno koristiti DMA (*Direct Memory Access*) sklop za prijenos podataka između SPI periferije i memorije. Za vrijeme DMA prijenosa procesor je rasterećen pa može obavljati druge korisne zadatke, odnosno moguće je ostvarenje neblokirajućih funkcija za SPI prijenos. Dodatna prednost je i smanjenje ukupnog trajanja komunikacije. Naime, kada se SPI komunikacija obavlja programski, program mora prvo upisati odlaznu riječ u podatkovni registar, zatim čekati primitak cijele dolazne riječi, te pročitati riječ prije upisa sljedeće riječi. To ponekad može uzrokovati diskontinuirani prijenos, odnosno periode čekanja između slanja dvije riječi. [6, str. 890] Korištenjem DMA slanje i primanje može se obavljati istovremeno, što eliminira ovaj problem.

Odabrani mikrokontroler sadrži dva DMA sklopa, a svaki od njih ima 7 prijenosnih

kanala. Periferija SPI3 spojena je na sklop DMA2, i to tako da je linija zahtjeva za dolazni prijenos (RX) spojena na kanal 1, a odlazni prijenos (TX) na kanal 2.

Prije korištenja DMA sklopa potrebno je obaviti određenu inicijalizaciju. Većina te inicijalizacije, kao što je omogućavanje prekida, postavljanje smjera prijenosa i slično, može se obaviti u alatu CubeMX, pa će ovdje biti opisani samo oni koraci koji se moraju obaviti nakon CubeMX inicijalizacije, s obzirom da su oni posebno važni za razvoj programske potpore. Ne smije se zaboraviti ni inicijalizacija SPI periferije, kako bi ona mogla slati zahtjeve za DMA prijenos. U nastavku je opisana potrebna procedura i programski kod kojim se ta procedura implementira, te je uz svaki korak procedure naveden broj linije koja implementira taj korak.

1. Postaviti adresu periferije za svaki kanal upisom u registar CPAR. Ova se adresa neće inkrementirati nakon svakog prijenosa. (2)

2. Postaviti adresu memorije za svaki kanal upisom u registar CMAR. Ova će adresa biti inkrementirana nakon svakog prijenosa, a iznos inkrementa ovisi o veličini riječi. (3)

3. Omogućiti *Transfer Complete* (TC) prekide za svaki kanal upisom bita TCIE u registru CCR. (4)

4. Postaviti duljinu prijenosa, odnosno broj riječi koje se trebaju prenijeti za svaki kanal upisom u registar CNDTR. (8)

5. Omogućiti SPI RX zahtjeve za DMA prijenos upisom bita RXDMAEN u SPI registru CR2. (12)

6. Omogućiti odgovarajuće DMA kanale upisom bita EN u registru CCR za svaki kanal. (16, 17)

7. Omogućiti SPI TX zahtjeve za DMA prijenos upisom bita TXDMAEN u SPI registru CR2. (21)

8. Omogućiti prekide koje okidaju SPI zastavice RXNE i TXE upisom bitova RXNEIE odnosno TXEIE u SPI registru CR2. (25, 26)

```
1    void DMA_Channel_Init(DMA_TypeDef *DMAx, uint32_t channel,
        uint32_t periph_addr, uint8_t *mem_addr) {
2      LL_DMA_SetPeriphAddress(DMAx, channel, periph_addr);
3      LL_DMA_SetMemoryAddress(DMAx, channel, (uint32_t)mem_addr);
```

```
4       LL_DMA_EnableIT_TC(DMAx, channel);
5     }
6
7     void DMA_Set_Channel_Data_Length(DMA_TypeDef *DMAx, uint32_t
          channel, uint32_t length) {
8       LL_DMA_SetDataLength(DMAx, channel, length);
9     }
10
11    void SPI_Enable_DMA_Rx_Request(SPI_TypeDef *SPIx) {
12      LL_SPI_EnableDMAReq_RX(SPIx);
13    }
14
15    void DMA_Enable_CH1_CH2(DMA_TypeDef *DMAx) {
16      LL_DMA_EnableChannel(DMAx, LL_DMA_CHANNEL_1);
17      LL_DMA_EnableChannel(DMAx, LL_DMA_CHANNEL_2);
18    }
19
20    void SPI_Enable_DMA_Tx_Request(SPI_TypeDef *SPIx) {
21      LL_SPI_EnableDMAReq_TX(SPIx);
22    }
23
24    void SPI_Enable_RXNE_TXE_Interrupts(SPI_TypeDef *SPIx) {
25      LL_SPI_EnableIT_RXNE(SPIx);
26      LL_SPI_EnableIT_TXE(SPIx);
27    }
```

Po završetku DMA prijenosa za svaki kanal, bit će generiran *Transfer Complete* (TC) prekid. Bitno je primijetiti da se prekid generira za svaki kanal posebno. U prekidnoj rutini tih prekida potrebno je očistiti zastavicu CTCFIFx (za kanal x) u DMA registru IFCR i onemogućiti odgovarajući kanal kako bi se spriječio nastavak prijenosa na sljedeći SPI zahtjev kada se postavi zastavica TXE ili RXNE. Kada završe oba prekida, potrebno je onemogućiti SPI RX i TX DMA zahtjeve. Programski kod koji obavlja ovu zadaću dan je u nastavku. S obzirom da prekidne rutine TC prekida izgledaju gotovo identično za oba kanala, navedena je samo jedna od njih.

```
1     void DMA_Transfer_Complete_RX_interrupt_handler() {
2       LL_DMA_ClearFlag_TC1(DMA2);
3       LL_DMA_DisableChannel(DMA2, LL_DMA_CHANNEL_1);
4       if (tc == 1) {
5         DMA_Disable(SB_SPIx);
6         tc = 0;
```

```
7       } else {
8         tc = 1;
9       }
10    }
11
12    void DMA_Disable(SPI_TypeDef *SPIx) {
13      SPI_Disable(SPIx);
14      SPI_Disable_DMA_Requests(SPIx);
15      // Postavi CS prikljucak u visoku razinu
16      LL_GPIO_SetOutputPin(ADC_CS_GPIOx, ADC_CS_PIN);
17      // Omoguci DRDY prekide ADC-a
18      NVIC_EnableIRQ(EXTI4_IRQn);
19    }
```

Globalna varijabla `tc` je zastavica koja signalizira je li se prekid drugog kanala već dogodio. Tek kada oba kanala završe s prijenosom može se onemogućiti SPI i omogućiti nove prekide ADC-a.

Prije pokretanja novog prijenosa potrebno je ponoviti korake 4 do 8 inicijalizacijske procedure.

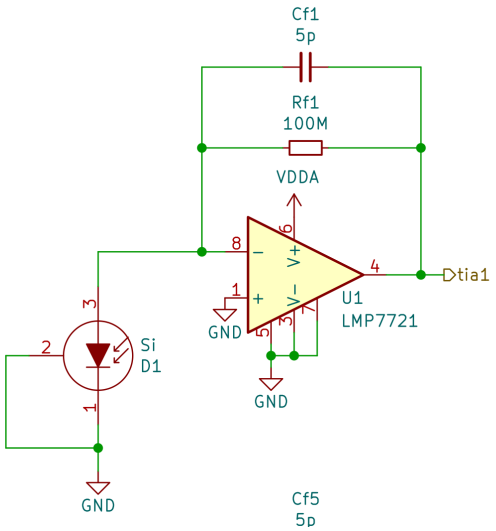
3. Opis korištenog sklopovlja

3.1. Senzorska pločica FERSAT-a

Sve komponente senzorskog podsustava FERSAT-a smještene su na takozvanoj senzorskoj tiskanoj pločici. Komponente koje se na njoj nalaze su:

- 8 fotodetektora (6 detektora vidljive svjetlosti i 2 detektora ultraljubičastog zračenja) s pojačalima LMP7721,
- 8 RC filtara, po jedan za svaki detektor,
- 3 temperaturna senzora ADT7301,
- analogno-digitalni pretvornik ADS131M08.

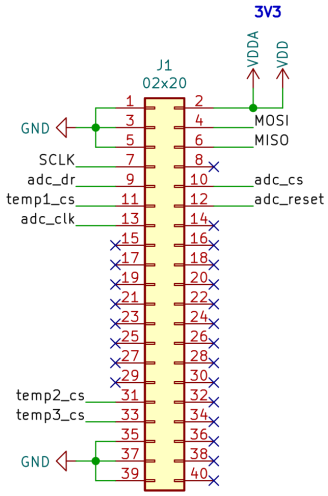
Jedan fotodetektor sastoji se od fotodiode i niskopropusnog filtra izvedenog pojačalom LMP7721. Slika 3.1 prikazuje jedan takav sklop. Na svaku od dioda postavljen je optički filtar (staklo ispred diode) koji propušta samo svjetlost u određenom području valnih duljina. Područje propuštanja razlikuje se za svaku fotodiodu.



Slika 3.1: Električna shema fotodetektora

Izlaz svakog fotodetektora propušta se kroz niskopropusni RC filtar s graničnom frekvencijom 1590 Hz, a zatim se dovodi na ulaz analogno-digitalnog pretvornika ADS131M08. Na pločici se nalaze i tri temperaturna senzora ADT7301 koji služe za kompenzaciju rezultata mjerenja s obzirom na promjenu temperature.

ADC i temperaturni senzori povezani su s PDH računalom jednim SPI sučeljem. Prijenosne linije SPI sučelja, kao i CS linije pojedinih sklopova i drugi upravljački signali, dovedene su do pločice na kojoj se nalazi PDH računalo putem konektora s 40 priključaka, čiji je raspored izvoda prikazan na slici 3.2.



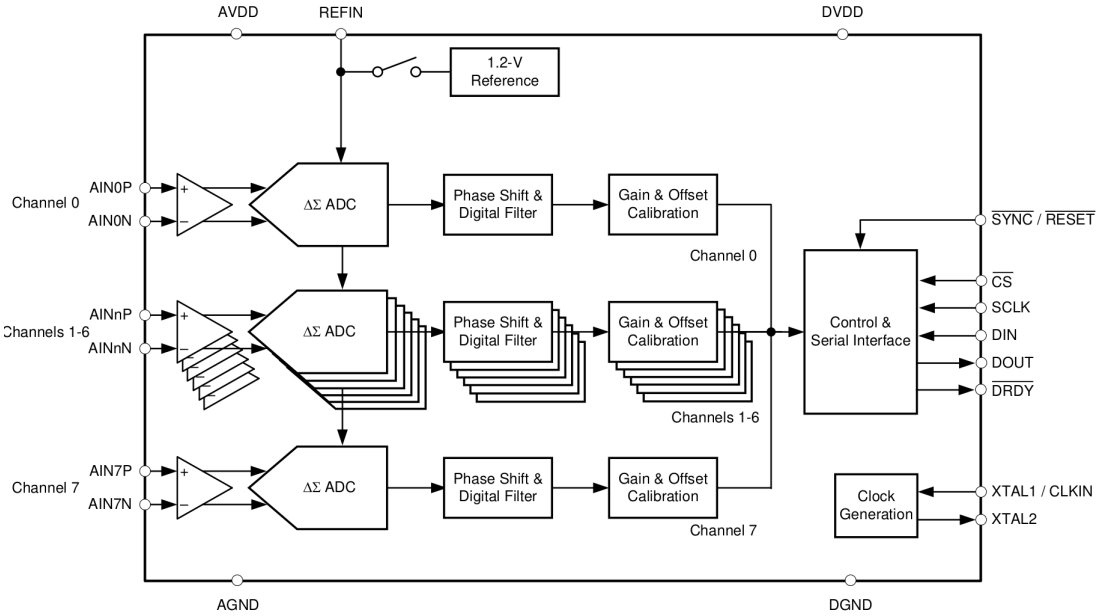
Slika 3.2: Raspored izvoda konektora na senzorskoj pločici

3.2. AD pretvornik ADS131M08

3.2.1. Opis rada sklopa

ADS131M08 je delta-sigma analogno-digitalni pretvornik s 8 kanala i rezolucijom od 24 bita proizvođača Texas Instruments [10]. Za komunikaciju s mikrokontrolerom koristi se SPI sučelje. Svih 8 kanala uzorkuje se istovremeno, a za svaki kanal moguće je postaviti programibilno pojačanje od 1 do 128. Frekvencija uzorkovanja također je programibilna i može iznositi do 32 tisuće uzoraka u sekundi. Slika 3.5 prikazuje blok dijagram sklopa.

Sklop zahtijeva odvojeno napajanje digitalnog i analognog dijela. Analogno napajanje može biti u rasponu 2.7 - 3.6 V, a digitalno napajanje treba biti 1.8 V ili 3.3 V. Referentni napon može se dovesti na vanjski priključak, ili se može koristiti unutarnji



Slika 3.3: Blok dijagram sklopa ADS131M08 [10]

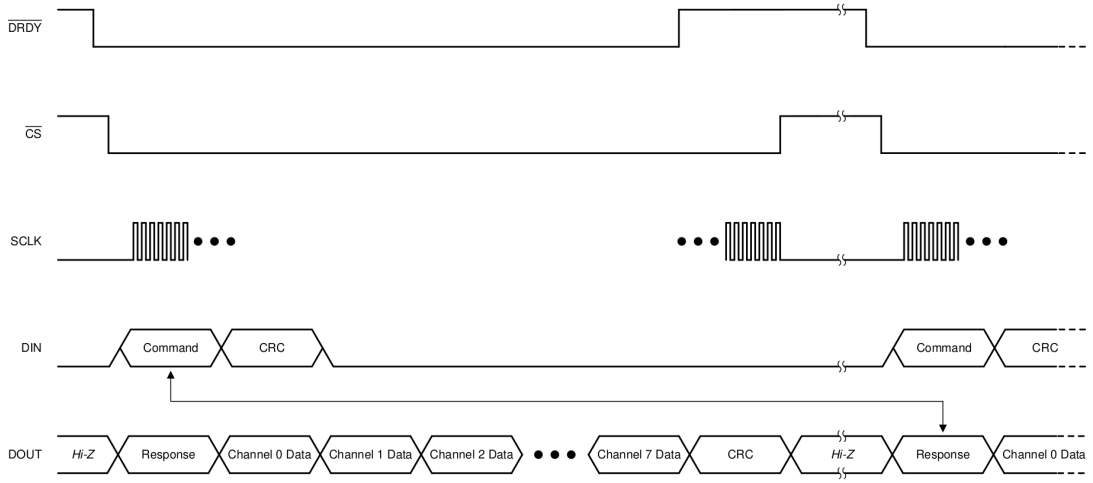
izvor referentnog napona iznosa 1.2 V. Signal takta može se generirati unutar sklopa ili može biti doveden na vanjski priključak.

3.2.2. SPI sučelje

SPI sučelje sklopa ADS131M08 koristi postavke CPOL = 0 i CPHA = 1, što znači da niska logička razina odgovara neaktivnom stanju takta i da se podatak čita na drugi brid takta (padajući). Za komunikaciju se koriste standardni SPI priključci (SCLK, MOSI, MISO i $\overline{\text{CS}}$) i dva dodatna priključka: $\overline{\text{DRDY}}$ (*Data Ready*) i $\overline{\text{SYNC/RESET}}$.

$\overline{\text{DRDY}}$ priključak postaje aktivan u trenutku kada je sklop spreman poslati rezultate konverzije. Aktiviranje priključka može se iskoristiti za okidanje prekida na mikrokontroleru, što omogućuje pravovremeno čitanje. Posebnu pažnju treba obratiti kada se podaci čitaju prvi put nakon uključenja ili kada je prošlo dulje vrijeme od zadnjeg čitanja. Naime, ADC ima unutarnji međuspremnik u kojem se spremaju rezultati konverzije ako nisu pročitani. Međuspremnik ima dovoljno kapaciteta za spremanje posljednje dvije konverzije. Kada je međuspremnik pun, ponašanje priključka $\overline{\text{DRDY}}$ neće biti dosljedno. Proizvođač zato preporuča dva načina za sinkronizaciju: prvi je pročitati dva uzorka zaredom bez čekanja aktivacije $\overline{\text{DRDY}}$ priključka i tako isprazniti međuspremnik [10]. Drugi način sinkronizacije je postavljanje pravokutnog impulsa trajanja duljeg od jednog perioda signala takta ADC-a, ali kraćeg od 2048 perioda na priključak $\overline{\text{SYNC/RESET}}$ [10].

Podaci se prenose preko SPI sučelja u takozvanim okvirima. Svaki okvir sastoji se od 10 riječi (ili manje, ako su neki kanali onemogućeni). Duljina riječi može se postaviti na 16, 24 ili 32 bita, a pretpostavljena duljina je 24 bita. Sučelje radi u *Full-Duplex* načinu rada. Vremenski dijagram jednog okvira prikazan je na slici 3.4.



Slika 3.4: Vremenski dijagram SPI komunikacije. Strelica označava naredbu i pripadajući odgovor. [10]

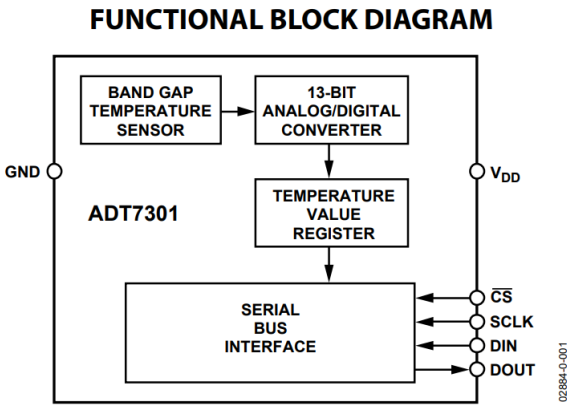
Prva riječ okvira na liniji MOSI, na slici označeno kao DIN, jest naredba koju *master* uređaj šalje ADC-u. Najčešće korištena naredba je *Null* naredba, čiji je instrukcijski kod 0x0000. Kada primi tu naredbu, ADC neće obaviti nikakvu operaciju, već će samo poslati rezultate konverzije. Neke druge naredbe su RREG (čitanje registra), WREG (upis u registar) i RESET. Druga riječ okvira je CRC (*Cyclic Redundancy Check*) zaštitni kod poruke (ili riječ bitova u niskoj razini ako je CRC onemogućen), nakon čega slijedi 8 riječi s bitovima u niskoj logičkoj razini.

Prva riječ okvira na liniji MISO, na slici označeno kao DOUT jest odgovor na naredbu u prethodnom okviru. Ako je naredba bila *Null*, tada je odgovor ispis sadržaja registra stanja ADC-a. U slučaju neke druge naredbe odgovor se može sastojati i od više riječi, primjerice ako se radi o ispisu sadržaja nekoliko registara. Nakon odgovora slijedi 8 riječi, po jedna za svaki kanal, koje sadrže rezultate AD pretvorbe. Moguće je onemogućiti neke kanale korištenjem naredbe WREG, u tom će slučaju riječ koja odgovara tom kanalu biti izostavljena. Posljednja riječ u okviru je CRC zaštitni kod poruke.

3.3. Temperaturni senzor ADT7301

3.3.1. Opis rada sklopa

Sklop ADT7301 proizvođača Analog Devices je temperaturni senzor s integriranim 13-bitnim analogno-digitalnim pretvornikom i serijskim sučeljem SPI. Omogućuje mjerenje temperature u rasponu od -40°C do 150°C, s rezolucijom 0.03125°C i tipičnom preciznošću ± 0.5°C [1]. Blok dijagram sklopa prikazan je na slici 3.5.



Slika 3.5: Blok dijagram sklopa ADT7301 [1]

Senzor uzima mjerenja temperature svakih 1.5 sekundi, što je regulirano internim sklopom za mjerenje vremena (engl. *timer*). Između dva mjerenja, napajanje analognog sklopovlja senzora je ugašeno te ono postaje neaktivno. Digitalno sklopovlje uvijek je aktivno, ali ako se pokuša pročitati vrijednost temperature više puta unutar jednog intervala mjerenja, senzor će uvijek vraćati istu vrijednost (onu koju je izmjerio na početku intervala).

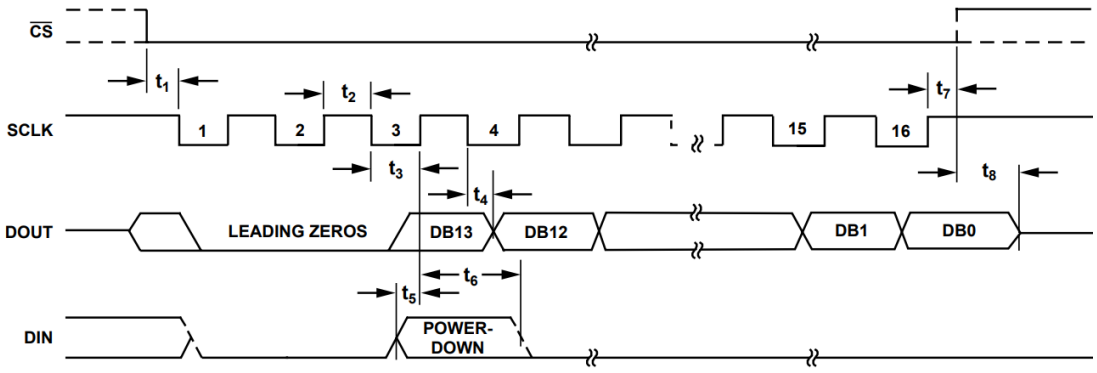
Dodatna mogućnost sklopa je takozvani *shutdown* način rada. U ovom načinu rada sklop troši vrlo malo struje (oko 1 µA), što je korisno ako postoji dulje vremensko razdoblje u kojem se neće uzimati uzorci temperature. *Shutdown* način rada omogućuje se upisom odgovarajućeg bita u kontrolni registar putem serijskog sučelja.

Prilikom ispitivanja senzora u uvjetima sobne temperature, primijećeno je kako nakon nekoliko minuta kontinuiranog rada očitana temperatura počinje rasti, te može pokazivati vrijednosti čak i do 55°C. Zaključeno je da je navedeno posljedica zagrijavanja samog senzora. Zato je odlučeno da će se senzor između mjerenja stavljati u *shutdown* način rada, kako bi se smanjila potrošnja struje, i samim time disipacija toplinske energije.

3.3.2. SPI sučelje

Temperaturni senzor ADT7301 koristi SPI postavke CPOL = 1 i CPHA = 1, što znači da visoka logička razina signala takta odgovara neaktivnom stanju i da se podatak čita na drugi brid takta (rastući).

ADT7301 ima mogućnost istovremenog slanja i primanja podataka (engl. *Full Duplex*). Na svojem priključku DOUT, koji je spojen na SPI liniju MOSI, sklop daje 16-bitni izlazni podatak na način da najznačajniji bit podatka izlazi prvi. Bitovi 15 i 14 su u niskoj logičkoj razini, bit 13 je bit predznaka, a ostali bitovi predstavljaju apsolutnu vrijednost očitane temperature. Na priključku DIN, koji je spojen na SPI liniju MISO, sklop prima 16-bitni podatak, gdje svi bitovi osim trećeg najznačajnijeg bita moraju biti u niskoj logičkoj razini. Treći najznačajniji bit označava hoće li sklop prijeći u *shutdown* način rada po završetku ciklusa slanja. Ako je taj bit u visokoj logičkoj razini, sklop će prijeći u *shutdown* način rada, a ako nije, sklop će nastaviti rad u dosadašnjem načinu rada. Slika 3.6 prikazuje jedan SPI ciklus čitanja/pisanja.



Slika 3.6: Vremenski dijagram SPI komunikacije sklopa ADT7301 [1]

4. Razvijena programska potpora

4.1. Korištene biblioteke i alati

Tvrtka ST Microelectronics nudi dva skupa programskih biblioteka za razvoj programske potpore namijenjene njihovim mikrokontrolerima: *Hardware Abstraction Layer* (HAL) i *Low Level* (LL) biblioteke. HAL biblioteke nude višu razinu apstrakcije sklopovlja od LL biblioteka, što omogućuje jednostavniji i brži razvoj, te olakšanu prenosivost između različitih mikrokontrolera istog proizvođača. S druge strane, LL biblioteke nude vrlo nisku razinu apstrakcije, što zahtijeva vrlo dobro poznavanje strukture i načina rada mikrokontrolera od strane programera, ali zato omogućuje pisanje programa koji zauzimaju vrlo malu količinu memorije [8]. Također, korištenje LL biblioteka smanjuje ukupno vrijeme izvođenja jer se eliminiraju česte provjere parametara ugrađene u HAL biblioteke. Upravo zbog vrlo ograničenih resursa dostupnih na odabranom mikrokontroleru, tijekom razvoja programske potpore opisane u radu [5] donesena je odluka o korištenju LL biblioteka. Iz tog razloga LL biblioteke korištene su i u razvoju programske potpore opisane u ovom radu.

Programska potpora razvijena je u programskom jeziku C, uz korištenje prevoditelja GCC u sklopu *GNU ARM Embedded Toolchain* paketa. Za razvoj je korišteno razvojno okruženje STM32CubeIDE, s integriranim alatom CubeMX. Navedeni alat omogućuje automatsko generiranje programskog koda za inicijalizaciju periferija mikrokontrolera i jednostavnu konfiguraciju signala takta kroz grafičko sučelje.

5. Zaključak

Zaključak.

LITERATURA

[1] *ADT7301 Temperature Sensor Datasheet*. Analog Devices, 2011. Rev. B.

[2] Filip Bogdanović. Sklopovi za mjerenje ultra-ljubičastog zračenja na satelitima. Diplomski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2020.

[3] Fabio Falchi, Pierantonio Cinzano, Christopher D. Elvidge, David M. Keith, i Abraham Haim. Limiting the impact of light pollution on human health, environment and stellar visibility. *Journal of Environmental Management*, 92(10), 2011.

[4] Filip Jurić. Upravljačko sklopovlje za prikupljanje i obradu senzorskih signala na CubeSat nanosatelitu. Završni rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2021.

[5] Goran Petrak. Programska potpora ugradbenog računalnog sustava za udaljeno prikupljanje fotografija putem satelita. Diplomski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2021.

[6] *RM0351 Reference manual STM32L47xxx, STM32L48xxx, STM32L49xxx and STM32L4Axxx advanced Arm®-based 32-bit MCUs*. ST Microelectronics, 2021. Rev. 9.

[7] *RM0090 Reference manual: STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs*. ST Microelectronics, 2021. Rev. 19.

[8] *UM7125 Description of STM32F4 HAL and low-layer drivers*. ST Microelectronics, 2021. Rev. 7.

[9] Kristijan Stepančić. Satelitska mjerenja koncentracije ozona. Završni rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2020.

[10] *ADS131M08 8-Channel, Simultaneously-Sampling, 24-Bit, Delta-Sigma ADC*. Texas Instruments, 2021. Rev. B.

[11] Jakov Tutavac. Spectral unmixing of incoherent light for the analysis of light pollution. Diplomski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2020.

[12] Wikipedia. Serial peripheral interface, 2022. URL `https://en.wikipedia.org/wiki/Serial_Peripheral_Interface`. Preuzeto: svibanj 2022.

[13] FER ZKIST. FERSAT - opis projekta, 2022. URL `https://www.fer.unizg.hr/zkist/FERSAT/projekt`. Preuzeto: lipanj 2022.

Programska potpora za prikupljanje i obradu senzorskih podataka na CubeSat nanosatelitu

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Software for CubeSat Nanosatellite Sensor Data Acquisition and Processing

Abstract

Abstract.

Keywords: Keywords.