SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2021-72

Programska potpora za upravljanje kamerom na CubeSat nanosatelitu

Nikola Gudan

Zagreb, lipanj 2022.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Hannon le.

SADRŽAJ

1. Uvod

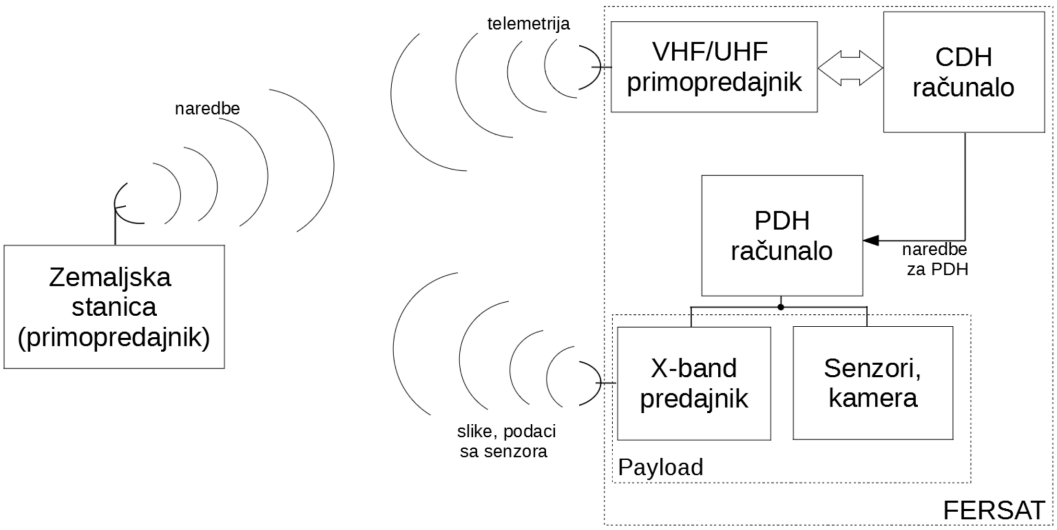
Ovaj završni projekt se izvodi u sklopu projekta FERSAT, koji se od 2018. godine provodi na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu [9]. Cilj projekta je izrada, lansiranje i korištenje jednog nanosatelita u CubeSat formatu, dimenzija 10 cm x 10 cm x 10 cm, volumena jedne litre i težine ne veće od 4/3 kg. Navedene dimenzije satelita odgovaraju formatu CubeSat 1U. Planirana visina orbite satelita je između 500 i 600 km, a očekivano trajanje misije je 3 godine. Korisni teret satelita (engl. *payload*) se sastoji od 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.

Kako bi se moglo upravljati radom korisnog tereta, na satelit će biti ugrađeno PDH (engl. *Payload Data Handler*) računalo, čija će zadaća biti prikupljanje podataka s kamere i senzorskog podsustava, pohranjivanje prikupljenih podataka u trajnu memoriju (engl. *non-volatile memory*), te slanje podataka na Zemlju pomoću komunikacijskog sustava. Izabrani mikrokontroler za ulogu PDH računala je STM32L471VGT6 proizvođača ST Microelectronics.

Ostalim podsustavima, koji nisu direktno vezani uz koristan teret, upravlja CDH (engl. *Command and Data Handler*) računalo. CDH računalo može upravljati položajem i orijentacijom satelita, slanjem telemetrijskih podataka na Zemlju, a također upravlja i napajanjem korisnog tereta i šalje naredbe PDH računalu preko CAN (engl. *Controller Area Network*) sučelja. U trenutku pisanja ove dokumentacije, konkretno CDH računalo još nije odabrano.

Slika 1.1 prikazuje blok dijagram cijelog sustava. U okviru ovog projekta razvijena je programska potpora PDH računala za upravljanje kamerom i *flash* memorijom.



Slika 1.1: Blok dijagram FERSAT-a i komunikacija sa zemaljskom postajom [3]

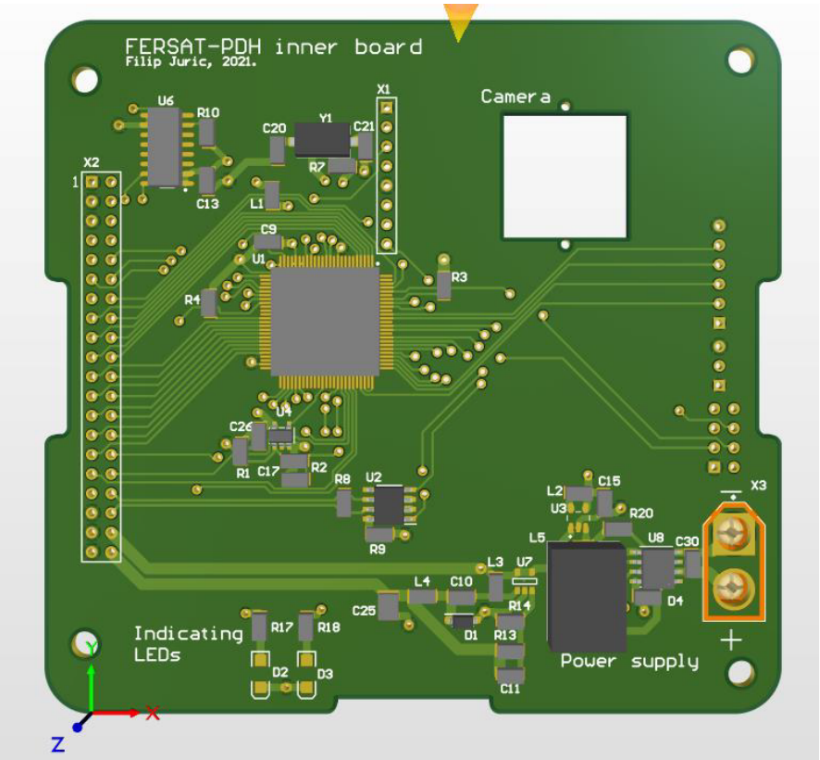
Sustav za upravljanje kamerom se sastoji od Arducam Mini 5MP Plus kamere. Upravljanje kamerom se sastoji od konfiguracije kamere i samog korištenja kamere, odnosno slikanja i spremanja slike. Konfiguracija kamere je nužna kako bi se ispravno podesili parametri trajanja ekspozicije, pojačanje i formata u kojem se slika želi spremiti.

2. Arhitektura sustava

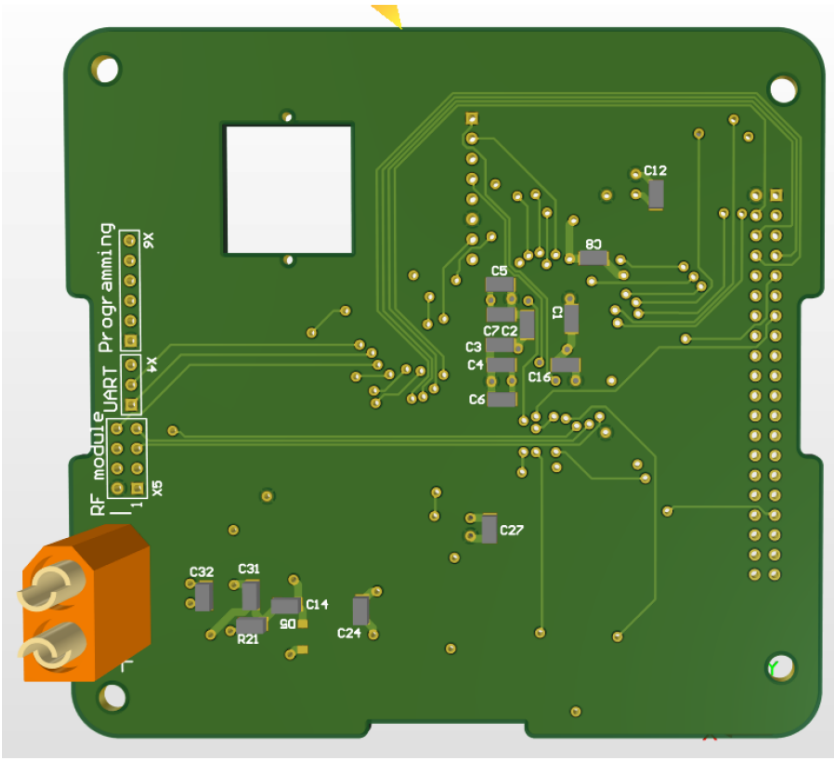
Slanjem određenog signala na sklop za kameru može se uslikati slika, a nakon slikanja slika se spremi na vlastiti međuspremnik kamere. Cilj je spremljenu kameru pročitati iz međuspremnika kamere i spremiti ju na *flash* memoriju koja se nalazi na pločici PDH-a, gdje može biti spremljena dok se ne zatraži slanje slike preko X-band predajnika na Zemlju.

Flash memorija, osim što služi za pohranu slike, služi i za pohranu podataka s drugih senzora. Ona prima i šalje podatke ovisno o poslanoj naredbi putem SPI komunikacije s mikrokontrolerom.

Upravljačko sklopovlje PDH računala se sastoji od STM32F471VGT6 mikrokontrolera, spomenute vanjske *flash* memorije, konektora za povezivanje s ostalim dijelovima sustava (uključujući i konektor za povezivanje s kamerom), sustava za napajanje, upravljačkog sklopovlja za CAN komunikaciju i sklopa za kontrolu izvođenja programa (engl. *watchdog*) [1]. Izgled tiskane pločice upravljačkog sklopovlja PDH računala prikazan je na slikama 2.1 i 2.2. Konektor X1 služi za povezivanje sustava s kamerom.



Slika 2.1: Prikaz gornje strane tiskane pločice upravljačkog sklopovlja PDH računala [1]



Slika 2.2: Prikaz donje strane tiskane pločice upravljačkog sklopovlja PDH računala [1]

Programska podrška za PDH računalo već je razvijena [3]. Međutim, u međuvre-

menu je došlo do promjene izbora mikrokontrolera PDH računala, te je stoga postojeću programsku podršku bilo potrebno prilagoditi trenutačnom sklopovlju.

S obzirom na prirodu ovog završnog projekta, gdje je naglasak bio na prilagođavanju postojeće programske podrške, u ovom radu će biti raspravljeni izazovi i izmjene do kojih je došlo tijekom prilagođavanja programske podrške. U poglavlju 2 dan je detaljan opis I^2C (engl. *Inter-Integrated Circuit*) komunikacije, te su istaknute razlike između starog i novog sklopovlja koje su bile ključne za prilagođavanje programske podrške. Na isti način opisani su SPI (engl. *Serial Peripheral Interface*) komunikacija i DMA (engl. *Direct Memory Access*) prijenos u poglavlju x. Detaljan pregled razvijene programske podrške dan je u poglavlju y, gdje je opisana integracija programske podrške za kameru i *flash* memorija u FreeRTOS operacijski sustav za rad u stvarnom vremenu.

3. Sučelja za komunikaciju

3.1. I^2C sučelje

Za konfiguraciju kamere Arducam 5MP Mini Plus PDH računalo koristi I^2C komunikaciju. U nastavku slijedi općeniti opis I^2C protokola kao i razlike između I^2C periferijski sklopova na prethodno korištenom (STM32F407VGT6) i trenutačnom (STM32L471VGT6) mikrokontroleru.

3.1.1. I^2C protokol

I^2C je jednostavna dvosmjerna sinkrona serijska sabirnica razvijena od strane *Philips Semiconductors* (sada *NXP Semiconductors*) 1982. godine [7]. Koristi dvije linije:

- serijska podatkovna linija (SDA, *Serial Data Line*),
- serijska taktna linija (SCL, *Serial Clock Line*).

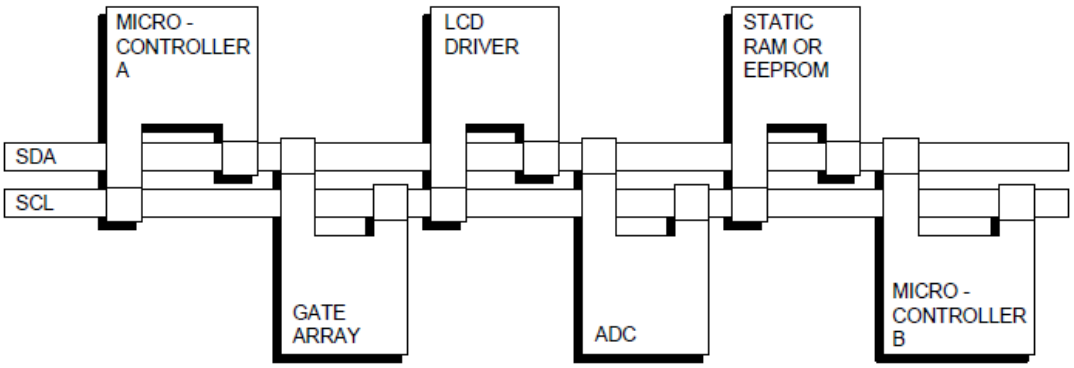
Obje linije su pritegnute na visoku logičku razinu preko *pull-up* otpornika. Moguće brzine prijenosa su:

- do 100 kbit/s u *Standard-mode* načinu rada,
- do 400 kbit/s u *Fast-mode* načinu rada,
- do 1 Mbit/s u *Fast-mode Plus* načinu rada,
- do 3.4 Mbit/s u *High-speed* načinu rada.

Navedene brzine se koriste kod dvosmjernog prijenosa, a moguća je i brzina do 5 Mbit/s u jednosmjernom prijenosu. Više uređaja se može spojiti na jednu sabirnicu, a svaki uređaj je prepoznatljiv po svojoj jedinstvenoj adresi i može se ponašati kao prijamnik ili odašiljač, ovisno o funkciji uređaja [2]. Protokol najčešće, a tako i u ovom slučaju, koristi 7-bitno adresiranje, a moguće je i korištenje 10-bitnog adresiranja. Osim konfiguracije prijamnika i odašiljača, uređaj također može biti *master* uređaj ili *slave* uređaj tijekom prijenosa podataka. *Master* uređaj je uređaj koji inicijalizira

prijenos podataka na sabirnici i generira signal takta kako bi omogućio prijenos. U tom trenutku, bilo koji uređaj koji je adresiran smatra se *slave* uređajem.

Na I^2C sabirnicu se također može spojiti više *master* uređaja, a primjer jednog takvog spoja sa dva mikrokontrolera dan je na slici 3.1. Prijenos podataka bi možda



Slika 3.1: Primjer I^2C sabirnice sa spojena dva mikrokontrolera [2]

mogao izgledati ovako:

1. Mikrokontroler A želi poslati podatke mikrokontroleru B:
    - mikrokontroler A (*master* uređaj) adresira mikrokontroler B (*slave* uređaj)
    - mikrokontroler A (*master*-odašiljač) šalje podatke mikrokontroleru B (*slave* uređaj-prijamnik)
    - mikrokontroler A prekida prijenos
2. Mikrokontroler A želi primiti podatke sa mikrokontrolera B:
    - mikrokontroler A (*master* uređaj) adresira mikrokontroler B (*slave*)
    - mikrokontroler A (*master*-prijamnik) prima podtke sa mikrokontrolera B (*slave*-odašiljač)
    - mikroknotroler A prekida prijenos.

U svakom od navedenih slučajeva mikrokontroler A je generirao takt i prekidao prijenos. Kod prijenosa podataka na I^2C sabirnici, *master* uređaj uvijek generira signal takta. U ovom radu korišten je samo jedan mikrokontroler, odnosno *master* uređaj, pa će se u daljnjem tekstu podrazumijevati samo taj slučaj.

Opis komunikacije i vremenski dijagram

I^2C komunikacija započinje sa *start* simbolom i završava sa *stop* simbolom. Komunikacijom se može čitati ili pisati ovisno o R/W bitu u adresi. Struktura adresiranja kod

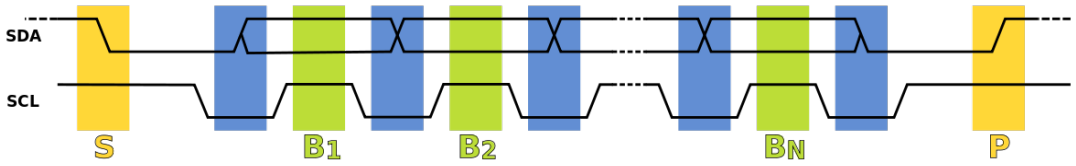
7-bitne adrese je prikazana u tablici 3.1.

Tablica 3.1: Struktura adresiranja kod 7-bitne adrese [7]

	Adresno polje							R\W
Pozicija bita u bajtu	7	6	5	4	3	2	1	0
Značenje	MSB						LSB	1=READ, 0=WRITE

Iz tablice je vidljivo da najmanje značajan bit označava želi li se podatak čitati ili pisati.

Imajući na umu izgled adresnog bajta, vremenski dijagram tipčne I^2C komunikacije prikazan je na slici 3.2.



Slika 3.2: Vremenski dijagram I^2C komunikacije [7]

- Prijenos podataka se inicijalizira *start* uvjetom (S) tako da SDA linija prijeđe u nisku logičku razinu dok SCL linija ostaje u visokoj logičkoj razini.
- (Plavo područje) SCL prelazi u nisku logičku razinu i SDA postavlja prvi podatkovni bit dok je SCL u niskoj logičkoj razini.
- (Zeleno područje) Podaci se primaju dok SCL poraste za prvi bit (B_1). Kako bi podaci bili valjani, SDA se ne smije promijeniti između rastućeg brida SCL-a i sljedećeg padajućeg brida.
- Postupak se ponavlja, SDA se postavlja dok je SCL u niskoj razini, a podaci se čitaju dok je SCL u visokoj razini (B_2 do B_n).
- Nakon posljednjeg bita slijedi taktni impuls, tijekom kojeg SDA prelazi u nisku razinu pripremajući se za *stop* uvjet.
- Signalizira se *stop* uvjet kada SCL prijeđe u visoku logički razinu, nakon čega slijedi prelazak u visoku logičku razinu SDA signala.
- (Plavo područje) SCL prelazi u nisku logičku razinu i SDA postavlja

Start i *stop* uvjete uvijek generira *master* uređaj. Nakon svakog bajta prijamnik šalje odašiljaču ACK bit kojim se signalizira uspješno primanje podatka, odnosno NACK bit kojim se signalizira neuspješno primanje podatka. ACK i NACK bitovi se nazivaju signalom potvrde i definiraju se na sljedeći način: odašiljač otpušta SDA liniju tijekom potvrdnog takta kako bi prijamnik mogao spustiti SDA na nisku razinu na kojoj i ostaje tijekom visoke razine takta. Ako SDA ostaje u visokoj razini tijekom devete periode takta, to predstavlja NACK (engl. *Not Acknowledge*) signal, a suprotan slučaj predstavlja ACK (engl. *Acknowledge*) signal. Ako je došlo do NACK signala, *master* uređaj može generirati *stop* uvjet kako bi prekinuo prijenos ili može ponovno generirati *start* uvjet kako bi započeo novi prijenos. Vremenski dijagram cijele komunikacije s potvrdnim signalima prikazan je na slici 3.3.

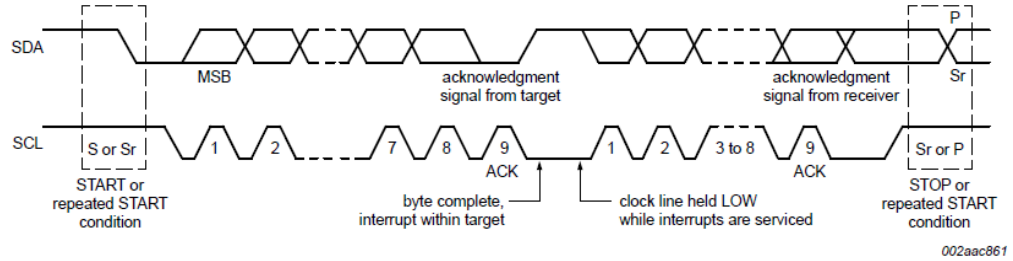


Figure 6. Data transfer on the I²C-bus

Slika 3.3: Prijenos podataka na I^2C sabirnici [2]

3.1.2. Razlika I^2C periferije na STM32L471VGT6 i STM32F407VGT6 mikrokontrolerima

Tijekom prijenosa koda sa starog mikrokontrolera na novi, primjećeno je da postoji razlika između struktura I^2C periferija. Točnije, postoji razlika između registarskih mapa na dvama periferijama, koje su vidljive usporedbom tablica 3.2 i 3.3 (*napomena*: u tablicama nisu prikazani svi registri, jer se neki registri niti ne koriste, ili se koriste samo tijekom konfiguracije periferija. Za puni prikaz tablica treba provijeriti dokumentacije mikrokontrolera [4], [5]).

Tablica 3.2: Registarska mapa I^2C periferije STM32L471VGT6 mikrokontrolera [5]

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	I2C_CR1	Res.								PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value									0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x4	I2C_CR2	Res.					PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.								ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	I2C_ICR	Res.																		ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.		STOPCF	NACKCF	ADDRCF	Res.		
	Reset value																			0	0	0	0	0	0			0	0	0			
0x24	I2C_RXDR	Res.																								RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	I2C_TXDR	Res.																								TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Tablica 3.3: Registarska mapa I^2C periferije STM32F407VGT6 mikrokontrolera [4]

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	I2C_CR1	Res.																SWRST	Res.	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Res.	SMBUS	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x4	I2C_CR2	Res.																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Res.		FREQ[5:0]					
	Reset value																				0	0	0	0	0			0	0	0	0	0	0
0x10	I2C_DR	Res.																								DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	I2C_SR1	Res.																SMBALERT	TIMEOUT	Res.	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	I2C_SR2	Res.																PEC[7:0]								DUALF	SMBHOST	SMBDEFAUL	GENCALL	Res.	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0

Vidljiva je razlika između količine registara, raspodjele i značenja njihovih bitova, kao i njihovih imena, što implicira različite funkcionalnosti pojedinih registara. Tako, npr. I^2C periferija kod STM32F407VGT6 sadržava 2 status registra: I2C_SR1 i I2C_SR2, dok kod STM32L471VGT6 postoji samo jedan status registar I2C_ISR. Ta razlika je bitna zato što se tijekom prijenosa podataka na I^2C sabirnici trebaju

provjeravati razne zastavice koje se mijenjaju tijekom komunikacije, kao što je npr. zastavica za prazni odašiljački registar (STM32F407VGT6: registar I2C_SR1 bit 7, STM32L471VGT6: bit 0), zastavica za puni prijamnički registar (STM32F407VGT6: registar I2C_SR1, bit 6, STM32L471VGT6: bit 2), zastavica za završetak prijenosa (STM32F407VGT6: ne postoji, STM32L471VGT6: bit 6) itd.

Vidljivo je također da kod STM32L471VGT6 postoji zastavica ADDR, koja inače kod STM32F407VGT6 signalizira uspješan primitak adrese uređaja mete, a kod STM32L471VGT6 ta zastavica se koristi isključivo u *slave* načinu rada, tako da ta zastavica nije bitna za ovaj projekt. Kako onda mikrokontroler zna da je poslana adresa točna? Naime, STM32L471VGT6 ima poseban registar za pohranu adrese uređaja mete, pa kada mikrokontroler pošalje *start* uvjet on automatski nakon završetka *start* uvjeta pošalje i adresu uređaja mete, a uspješan primitak adrese signalizira zastavica I2C_ISR_TXIS kod slanja podataka, odnosno I2C_ISR_RXNE zastavica kod primitka podataka.

Vidljive su i razlike u raspodjeli zastavica u registrima, kao i razlike u funkcijama koje zastavice signaliziraju. Inače bi te razlike stvarale probleme kod konfiguracije I^2C periferije, no, kako je tu brigu riješio kod generator ugrađen u STM32CubeIDE razvojno okruženje, nije bila posvećena pažnja tim razlikama. Način implementacije spomenutih razlika u programsku podršku opisan je u poglavlju z.

3.2. SPI sučelje

Protokol SPI se koristi za prijenos podataka između *flash* memorije i mikrokontrolera, odnosno međuspremnika kamere. Kako bi se oslobodili resursi na mikrokontroleru, za prijenos podataka između kamere i *flash* memorije se, u kombinaciji sa SPI protokolom, koristi i DMA prijenos. U ovom poglavlju bit će opisan SPI protokol i DMA prijenos i bit će istaknute razlike i problemi kod prilagođavanja programske podrške za STM32L471VGT6 mikrokontroler.

3.2.1. SPI protokol

SPI je sinkrono serijsko komunikacijsko sučelje koje se koristi za komunikaciju na kratkim udaljenostima, pretežito u ugradbenim računalnim sustavima [8].

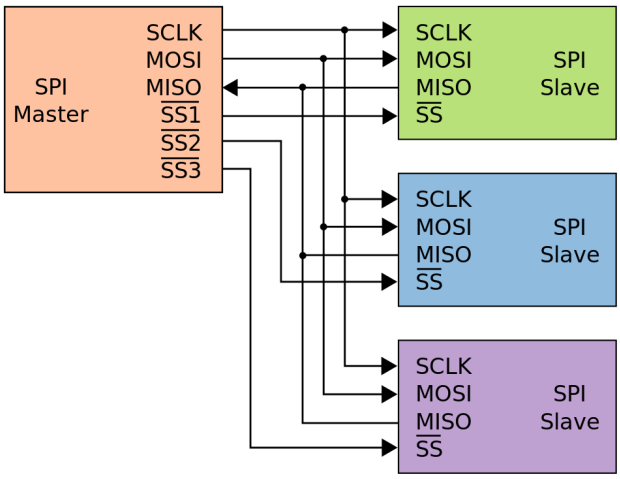
SPI uređaji komuniciraju u *full-duplex* načinu rada koristeći *master-slave* arhitekturu, obično sa jednim *master* uređajem. Više *slave* uređaja može biti spojeno na jedan upravljač tako da se aktivira određeni *chip select* signal za pojedini uređaj.

Opis sučelja

SPI sabirnica se sastoji od četiri signala:

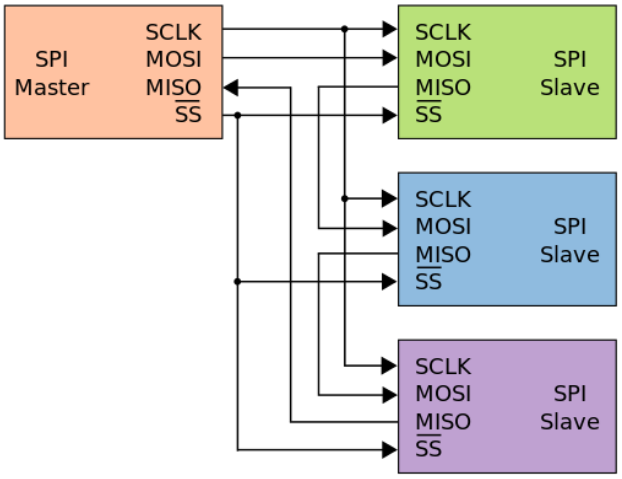
- SCLK: Serijski takt (izvor je *master* uređaj),
- MOSI: *Master Output Slave Input* (izvor podataka iz *master* uređaja),
- MISO: *Master Input Slave Output* (izvor podataka iz *slave* uređaja),
- CS/SS: *Chip/Slave Select* (aktivan nisko, signal iz *master* uređaja, označava da se prenose podaci).

MOSI na *master* uređaju se spaja na MOSI na *slave* uređaju, dok se MISO na *master* uređaju se spaja na MISO na *slave* uređaju. CS/SS se koristi za pokretanje komunikacije između *slave* i *master* uređaja. Za svaki *slave* uređaj postoji zaseban CS/SS priključak na *master* uređaju. Takav način spajanja se naziva neovisni *slave* uređaj. Primjer spajanja tri *slave* uređaja na jedan *master* uređaj u konfiguraciji neovisnog *slave* uređaja prikazan je na slici 3.4.



Slika 3.4: Spoj tri *slave* uređaja na jedan *master* uređaj u konfiguraciji neovisnog *slave* uređaja. Vidljivo je da *master* uređaj ima tri SS priključka, a svaki odgovara jednom *slave* uređaju, dok se SCLK, MOSI i MISO linije međusobno dijele između *slave* uređaja [8]

Moguće je još spojiti uređaje u konfiguraciju ulančavanog *slave* uređaja. U toj konfiguraciji *slave* uređaji dijele isti CS/SS, a ulančavanjem preko MISO/MOSI linija podaci se prenose prema načelu posmačnog registra, koji je objašnjen u sljedećem potpoglavlju. Prikaz spajanja tri *slave* uređaja se nalazi na slici 3.5.



Slika 3.5: Spoj tri *slave* uređaja na jedan *master* uređaj u konfiguraciji ulančavanog *slave* uređaja [8]

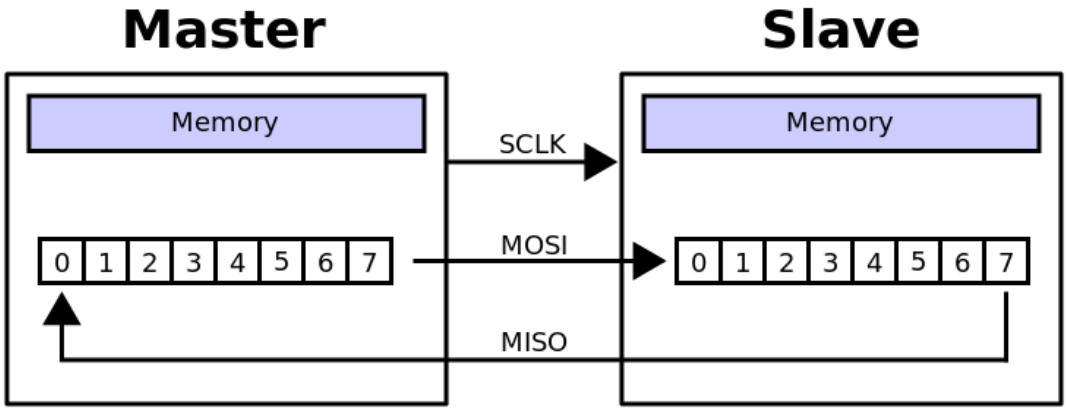
Način rada

SPI sabirnica radi s jednim *master* uređajem i jednim ili više *slave* uređaja. Ako se koristi jedan *slave* uređaj, onda CS signal može biti postavljen u nisku logičku razinu, ako *slave* uređaj to dopušta. Neki *slave* uređaji zahtijevaju padajući brid CS signala kako bi započela komunikacija. Ako se koristi više *slave* uređaja potreban je zaseban CS signal *master* uređaja za svaki *slave* uređaj.

Prijenos podataka Za početak komunikacije *master* uređaj konfigurira takt koristeći frekvenciju koju podržava *slave* uređaj, obično do nekoliko MHz. *Master* uređaj zatim odabire *slave* uređaj postavljanjem CS linije u nisko logičko stanje. Ako je potreban period čekanja, npr. za AD (analogno-digitalnu) pretvorbu, *master* uređaj mora pričekati minimalno taj period vremena prije puštanja takta.

Tijekom svakog perioda takta obavlja se prijenos podataka u *full-duplex* načinu rada. To znači da *master* uređaj pošalje jedan bit na MOSI liniju, koji *slave* uređaj pročita, dok u isto vrijeme *slave* uređaj šalje jedan bit na MISO liniju, koji *master* uređaj pročita. Takva sekvenca se održava čak i kada se izvodi jednosmjerni prijenos podataka.

Prijenosi podataka uključuju dva posmačna registra zadane veličine, npr. 8 bitova, jedan u *master* i drugi u *slave* uređaju. Registri su spojeni u topologiji virtualnog prstena (slika 3.6).

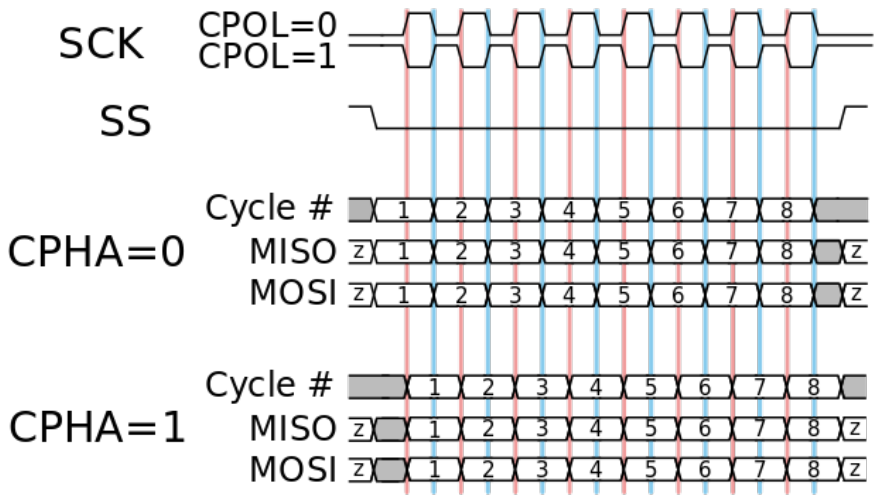


Slika 3.6: Tipičan spoj dvaju posmačna registra koji formiraju kružni međuspremnik [8]

Podaci se obično pomiču tako da se prvo pomakne najznačajniji bit. Na brid takta, *master* i *slave* uređaj pomaknu bit i pošalju ga na prijenosnu liniju. Na sljedeći brid takta, na svakom prijamniku bit se uzorkuje s prijenosne linije i postavlja se kao novi najmanje značajni bit u posmačnom registru. *Master* i *slave* uređaji u potpunosti razmjene podatke u registrima nakon što se svi bitovi u registrima prebace. Ako je potrebno razmijeniti još podataka, posmačni registri se ponovno napune te se postupak ponavlja, a prijenos se može obavljati za bilo koji broj perioda takta. Kada je prijenos dovršen, *master* uređaj prestaje davati takt i obično isključi CS signal, odnosno postavi ga na visoku razinu.

Prijenos se obično obavlja u riječima širine 8 bitova, no moguća je i širina riječi od 16 bita, ili čak 12 bitova, koji se koristi za digitalno-analogne i analogno-digitalne pretvornike.

Polaritet takta i faza Osim što mora podesiti frekvenciju takta, *master* uređaj mora isto tako podesiti polaritet takta (CPOL, engl. *Clock Polarity*) i fazu (CPHA, engl. *Clock Phase*) ovisno o podacima. Vremenski dijagram je prikazan na slici 3.7.



Slika 3.7: Vremenski dijagram koji pokazuje polaritet takta i fazu. Crvene linije označuju vodeće bridove, a plave linije označavaju prateće bridove [8].

CPOL određuje polaritet kanala. Polaritet može biti invertiran jednostavnim inverterom.

- Ako je CPOL = 0, onda takt miruje u niskom logičkom stanju, a svaki period se sastoji od impulsa visokog logičkog stanja. To znači da je vodeći brid rastući brid, a prateći brid padajući brid.
- Ako je CPOL = 1, onda takt miruje u visokom logičkom stanju, a svaki period se sastoji od impulsa niskog logičkog stanja. To znači da je vodeći brid padajući brid, a prateći brid rastući brid.

CPHA određuje fazu podatkovnih bitova u odnosu na takt.

- Ako je CPHA = 0, strana koja šalje podatke mijenja podatak na prateći brid prethodnog perioda takta, dok strana koja prima podetke prihvaća podatak na (ili ubrzo nakon) vodeći brid perioda takta. Izlazna strana zadržava valjani podatak sve do pojave pratećeg brida trenutnog perioda takta.
- Ako je CPHA = 1, strana koja šalje podatke mijenja podatak na vodeći brid trenutnog perioda takta, dok strana koja prima podatke prihvaća podatak na (ili ubrzo nakon) pratećeg brida perioda takta. Izlazna strana zadržava valjani podatak do pojave vodećeg brida sljedećeg perioda takta. Na zadnji period, *slave* uređaj zadržava valjani podatak na MISO liniji sve dok *slave* uređaj ne bude deselektiran.

MOSI i MISO signali su obično stabilni za vrijeme pola perioda takta, sve do sljedeće promjene takta. SPI *master* i *slave* uređaji mogu uzorkovati podatke u bilo kojem

vremenu unutar te polovice periode takta.

Kombinacije različitih konfiguracija CPOL i CPHA bitova predstavljaju načine rada. Konvecija je da CPOL predstavlja viši bit, dok CPHA predstavlja niži bit. Načini rada kod ARM-ovih mikrokontrolera su prikazani u tablici 3.4.

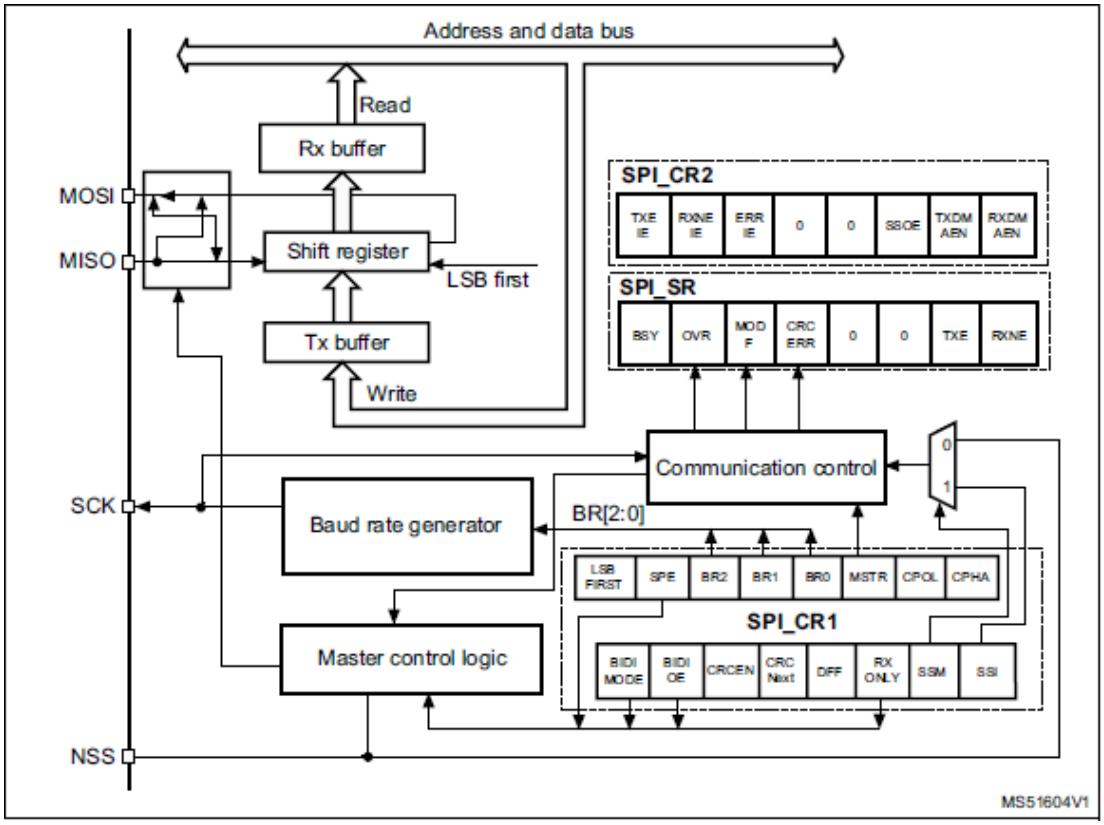
Tablica 3.4: SPI načini rada kod ARM-ovih mikrokontrolera [8]

SPI način rada	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

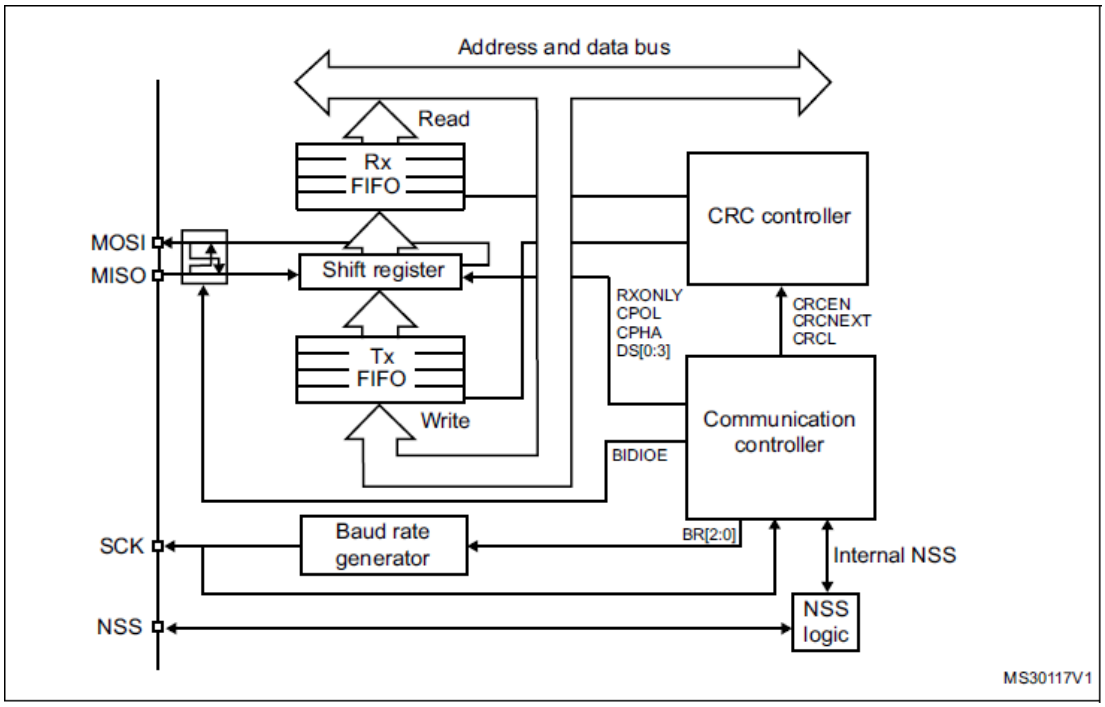
3.2.2. Prilagodba programske podrške sa STM32F407VGT6 mikrokontrolera na STM32L471VGT6 mikrokontroler

Što se tiče programske podrške za SPI komunikaciju između mikrokontrolera, *flash* memorije i kamere, nije došlo do nikakvih poteškoća kod prijenosa programske podrške s prethodno korištenog mikrokontrolera na trenutni. Pogledom na blok dijagrame SPI periferije (slike 3.8 i 3.9) vidljiva je velika sličnost između mikrokontrolera. Razlike između kontrolnih registara nisu problem, s obzirom na to da njih podešava generator koda, dok se razlike u statusnim registrima mogu zanemariti radi korištenja definicija registara i zastavica u programskoj podršci koju je pružao proizvođač mikrokontrolera.

Došlo je, međutim, do poteškoća kod prijenosa programske podrške za DMA prijenos, koje će biti objašnjene u sljedećem poglavlju.



Slika 3.8: SPI blok dijagram mikrokontrolera STM32407VGT6 [4, str. 876]



Slika 3.9: SPI blok dijagram mikrokontrolera STM32L471VGT6 [5, str. 1451]

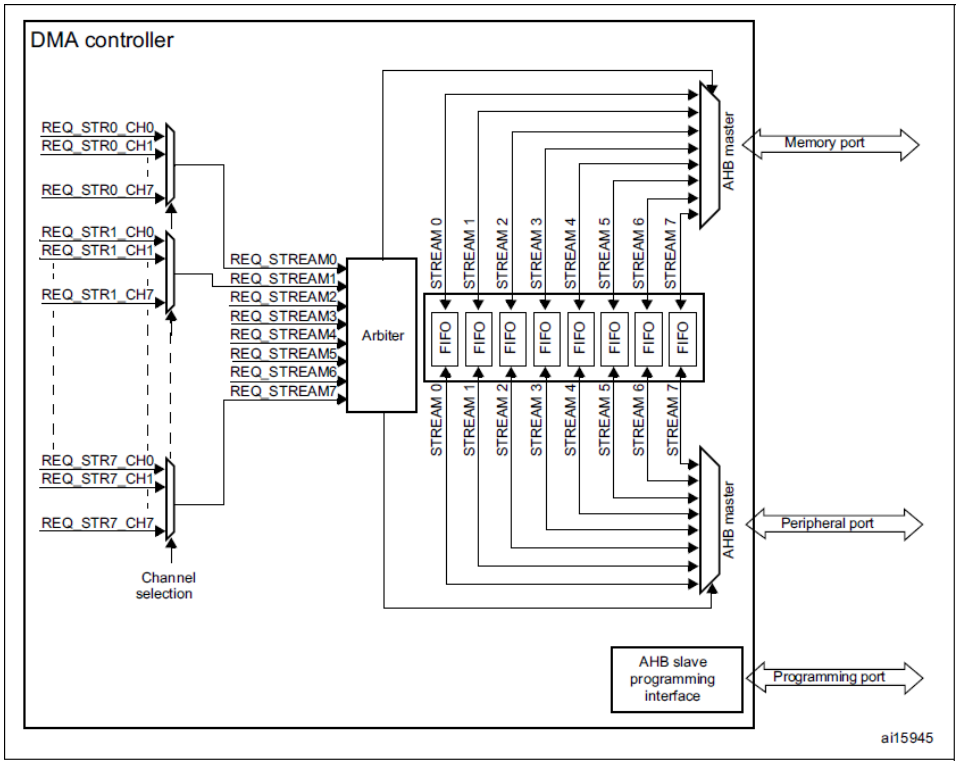
3.2.3. DMA prijenos

DMA se koristi kako bi se omogućio prijenos podataka visokih brzina između periferijskih sklopova i memorije ili između dviju memorijskih jedinica [4]. Podatci se brzo mogu prenijeti bez posredovanja procesora. Na taj se način oslobađaju resursi procesora kako bi se mogle izvoditi druge operacije za vrijeme prijenosa.

U ovom projektu DMA prijenos se koristi između međuspremnika kamere i radne memorije mikrokontrolera.

Razlike DMA periferije na STM32F407VGT6 i STM32L471VGT6 mikrokontrolerima

STM32F407VGT6 mikrokontroler ima dva DMA kontrolera. Blok dijagram jednog DMA kontrolera je prikazan na slici 3.10.



Slika 3.10: Blok dijagram DMA kontrolera na STM32F407VGT6 mikrokontroleru [4]

DMA kontroler obavlja izravan prijenos memorije; kao AHB (engl. *AMBA High-performance Bus*, AMBA na engl. *Advanced Microcontroler Bus Architecture*) *master*, može u bilo kojem trenutku preuzeti kontrolu nad AHB sabirnicom i pokrenuti AHB transakcije. DMA kontroler može pokrenuti sljedeće transakcije:

- prijenos sa periferije na memoriju,

- prijenos sa memorije na periferiju,
- prijenos sa memorije na memoriju.

S obzirom na to da se u ovom radu koriste prijenosi s periferije na memoriju i obrnuto, u daljnjem tekstu će se podrazumijevati samo ti slučajevi.

DMA periferija ima dvije AHB *master* sabirnice, jedna se koristi za pristup memoriji, a druga za pristup periferijama. AHB *slave* sabirnica se koristi za programiranje DMA kontrolera.

Pojedini kontroler ima 8 tokova (engl. *stream*), te za svaki tok postoji 8 kanala (engl. *channel*). Svaki tok je spojen na određeni sklopovski DMA kanal. Tokovi i kanali služe kako bi se ostvarila veza između ostalih periferija i DMA mikokrontrolera, tako da periferije mogu slati zahtjev za DMA prijenos.

Svaki DMA prijenos se sastoji od tri operacije:

- učitavanje sa periferijskog podatkovnog registra ili lokacije u memoriji, čije su adrese zapisane u DMA_SxPAR ili DMA_SxM0AR registru
- spremanje podataka na periferijski podatkovni registar ili lokaciju u memoriji, čije su adrese zapisane u DMA_SxPAR ili DMA_SxM0AR registru
- naknadno dekrementiranje DMA_SxNDTR registra, koji sadržava broj podataka koji se još trebaju prenjeti

Spomenuti registri su prikazani u tablici 3.5. Kada periferija želi pristupiti DMA kontroleru, periferijski sklop šalje zahtjev DMA kontroleru. DMA kontroler poslužuje zahtjev ovisno o prioritetima kanala. Čim DMA kontroler pristupi periferiji, DMA kontroler periferiji šalje signal potvrde. Periferni uređaj ukida svoj zahtjev čim dobije potvrdni signal iz DMA kontrolera. Nakon što periferna jedinica ukine zahtjev, DMA kontroler ukida signal potvrde. Ako ima više zahtjeva, periferna jedinica može pokrenuti sljedeću transakciju.

Tablica 3.5: Registri DMA_SxPAR, DMA_SxM0AR i DMA_SxNDTR kod STM32F407VGT6 mikrokontrolera. *x* označava broj toka [4]

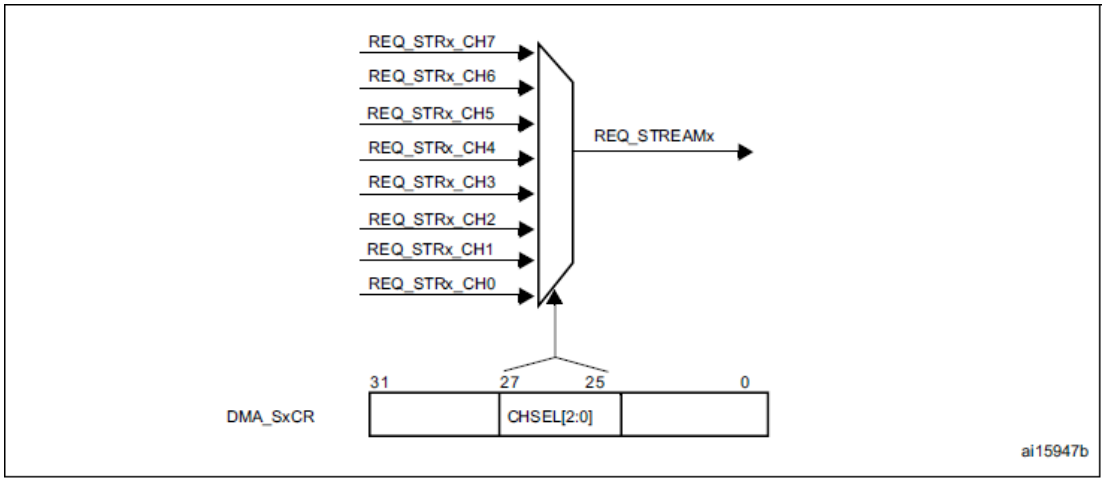
Register name	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DMA_SxPAR	PA[31:0]																															
Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DMA_M0AR	M0A[31:0]																															
Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DMA_SxNDTR	Res.																NDT[15:0]															
Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Svaki tok je povezan sa DMA zahtjevom, koji može biti odabran između 8 mogućih

kanalnih zahtjeva. Odabir kanala određuju bitovi CHSEL[2:0] u DMA_SxCR registru (tablica 3.6). Odabir kanala prikazan je na slici 3.11.

Tablica 3.6: Registar DMA_SxCR kod STM32F407VGT6 mikrokontrolera [4]

Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA_SxPAR	Res.				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Res.	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Slika 3.11: Odabir kanala [4]

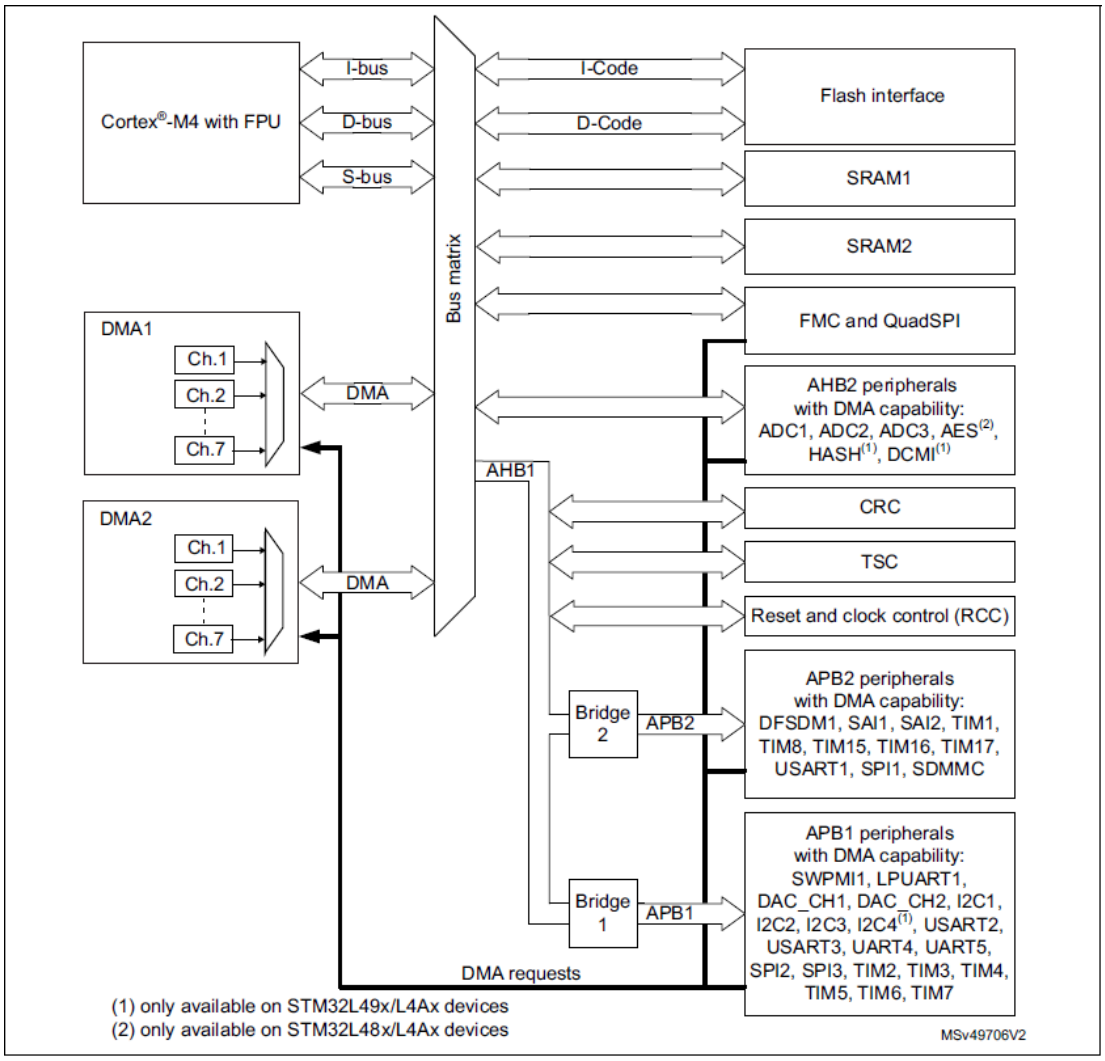
Proučavanjem dokumentacije mikrokontrolera, zaključeno je da je za SPI prijenos putem DMA sklopa potrebno koristiti kanal 0 i tokove 3 (SPI2_RX) i 4 (SPI2_TX) [4, str. 307]. Odabrani tokovi se koriste zato što je kamera spojena na SPI2 periferiju mikrokontrolera.

Blok dijagram DMA periferije na STM32L471VGT6 mikrokontroleru je prikazan na slici 3.12. Vidljivo je da oba mikrokontrolera sadržavaju dvije DMA periferije. Za razliku od SMT32F407VGT6 mikrokontrolera, trenutačni mikrokontroler, STM32L471VGT6, nema dvije AHB *master* sabirnice, već samo jednu AHB *master* sabirnicu.

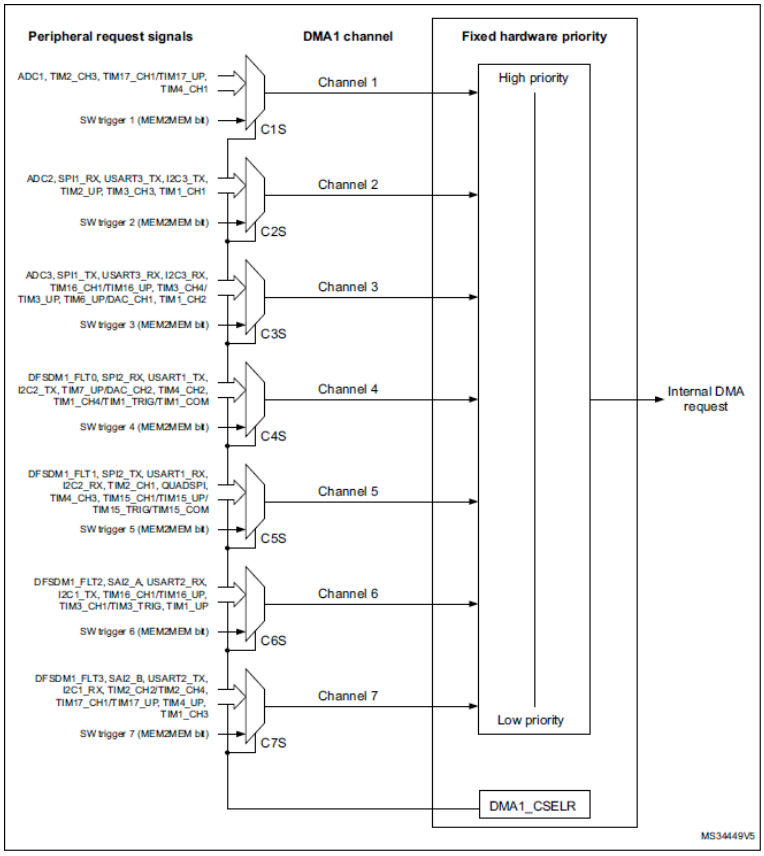
STM32L471VGT6 nema tokove za ostvarivanje veze između periferija i DMA sklopa, već ima samo kanale, te se stoga veza između ostalih periferija i DMA kontrolera ostvaruje na drugačiji način, prikazan na slici 3.13. Iz slike je vidljivo da se na ovom mikrokontroleru trebaju koristiti kanali 4 (SPI2_RX) i 5 (SPI2_TX).

Još jedna razlika između DMA periferija dvaju mikrokontrolera se krije u prekidima koje DMA kontrolera i zastavicama za prekide. STM32F407VGT6 ima 5 razli-

čitih prekida: završetak prijenosa (TC, engl. *Transfer Complete*), obavljena polovica prijenosa (HT, engl. *Half Transfer*), greška u prijenosu (TE, engl. *Transfer Error*), FIFO greška (FE, engl. *FIFO Error*) i greška u direktnom načinu rada (DME, engl. *Direct Mode Error*). STM32L471VGT6 ima 3 različita prekida: završetak prijenosa završetak prijenosa (TC), obavljena polovica prijenosa (HT), greška u prijenosu (TE). Kod STM32L471VGT6 postoji još globalni prekid (GI, engl. *Global Interrupt*) koji se aktivira u svim slučajevima. Ako se, na primjer, želi DMA kontroler namjestiti da zahtijeva prekid kod završetka prijenosa, polovice prijenos i kod greške u prijenosu, to se može podesiti jednostavno aktiviranjem globalnih prekida. S obzirom na to STM32F407VGT6 sadržava više vrsta prekida, on sadržava i 2 prekidna registra, dok STM32L471VGT6 sadržava 1 prekidni registar.



Slika 3.12: Blok dijagram DMA periferije na STM32L471VGT6 mikrokontroleru [5]



Slika 3.13: Mapiranje zahtjeva za DMA1 kontroler kod STM32L471VGT6 mikrokontrolera [5, str. 338]

Što se tiče ostalih dijelova DMA periferija, mikrokontroleri funkcioniraju isto. Postoje, međutim, razlike u načinu konfiguracije DMA kontrolera, međutim, to nije predstavljalo problem, s obzirom na to da je konfiguracija prepuštena generatoru koda. Nazivi registara koji se koriste su također različiti između dva mikrokontrolera, iako su im funkcije iste. To također nije bio problem jer su se koristile definicije registara u programskoj podršci koju je pružao proizvođač mikrokontrolera, pa je bilo potrebno samo promijeniti nazive tih registara.

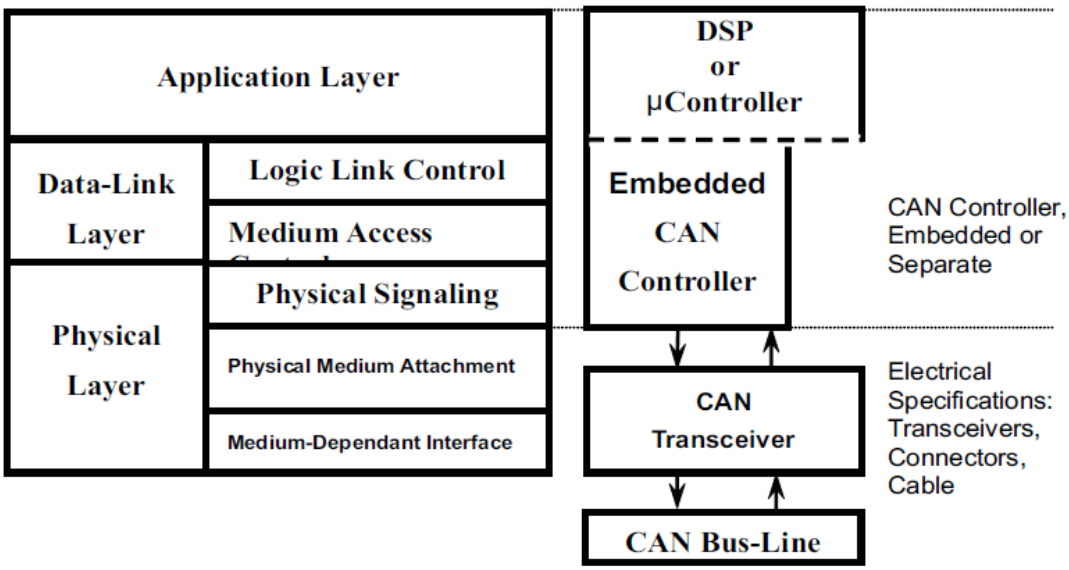
3.3. CAN protokol

S obzirom na nepredvidive poteškoće sa sklopovljem PDH računala, nije ostalo vremena za implementaciju CAN komunikacije. Međutim, u ovom poglavlju dati će se opis protokola, kao i njegova implementacija na mikrokontroleru, te će se izložiti mogućnosti implementacije u ovaj projekt.

3.3.1. Opis protokola

CAN je serijska komunikacijska sabirnica koju je standardizirao ISO (engl. *International Standardization Organization*), a razvijena je od strane BOSCH-a za automobilsku industriju s ciljem da se zamijeni kompliciran žičani kabel s dvožičnom sabirnicom [6]. Specifikacija zahtijeva su visoka otpornost na električne smetnje i sposobnost otkoravanja i ispravljanja greški kod prijenosa podataka.

Komunikacijski protokol CAN opisuje kako se informacija prenosi između uređaja na mreži i kako odgovara OSI (engl. *Open Systems Interconnection*) modelu koji je definiran u slojevima (slika 3.14). Stvarna komunikacija između uređaja spojenih fizičkim medijem je definirana fizičkim slojem modela.



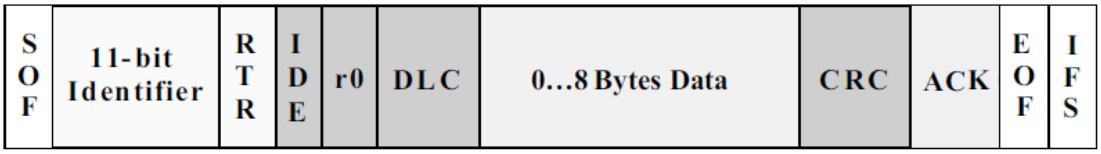
Slika 3.14: OSI model CAN protokola [6, str. 2]

CAN komunikacijski protokol je protokol s višestrukim pristupom, osluškivanjem nosioca, detekcijom sudara i arbitracijom na paritet poruka (CSMA/CD+AMP). CSMA znači da svaki čvor na sabirnici mora čekati određeni period neaktivnosti prije nego što pokuša poslati poruku. CD+AMP znači da se sudari rješavaju bitnom (*bit-wise*) arbitracijom, koja se temelji na prethodno namještenom prioritetu svake poruke u identifikacijskom polju poruke. Viši prioritet uvijek dobiva pristup sabirnici.

Standardni CAN protokol s identifikatorom širine 11 bita omogućava brzine prijenosa od 125 kb/s do 1 Mb/s. Standardni protokol je poslije zamijenjen s proširenim protokolom s identifikatorom širine 29 bita. Standardni 11-bitni identifikator omogućava 2^{11}, ili 2048 različitih identifikatora poruka, dok prošireni 29-bitni identifikator omogućava 2^{29}, ili 536870912 različitih identifikatora.

Standardni CAN protokol

Standardni CAN sa 11-bitnim identifikatorom je prikazan na slici 3.15.



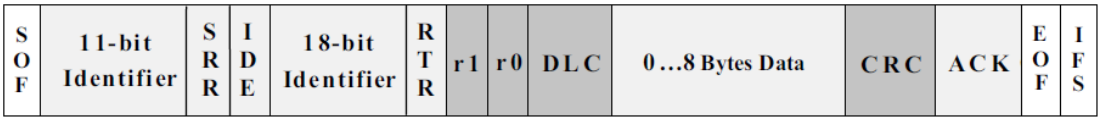
Slika 3.15: Standardna CAN poruka: 11-bitni identifikator [6, str. 3]

Značenje pojedinih bitova na slici 3.15 su:

- SOF - jedan bit, početak okvira (engl. *Start Of Frame*), označava početak poruke i koristi se za sinkronizaciju čvorova na sabirnici nakon mirovanja,
- Identifikator - 11-bitni identifikator standardnog CAN protokola, uspostavlja prioritet poruke. Manja binarna vrijednost znači viši prioritet,
- RTR - jedan bit, zahtjev za udaljenim prijenosom (engl. *Remote Transmission Request*) dominantan je kada se traži informacija s drugog čvora. Svi čvorovi prime zahtjev, ali identifikator određuje traženi čvor. Povratna informacija se također šalje na sve čvorove i svaki čvor ju može iskoristiti ako je potrebno. Na taj su način svi podatci koji se koriste u sustavu uniformni,
- IDE - jedan bit, proširenje identifikatora (engl. *Identifier Extension*), označava da se šalje standardni CAN identifikator bez proširenja,
- r0 - rezervirani bit (za moguću upotrebu kod budućih dopuna standarda),
- DLC - 4-bitna širina podatkovnog koda (engl. *Data Length Code*), sadržava broj bajtova podatka koji se šalje,
- Podatci - može se slati do 64 bitova aplikacijskih podataka,
- CRC - 16-bitna (15 bitova plus granični bit) ciklička provjera redundancije (engl. *Cyclic Redundancy Check*) sadržava kontrolni zbroj (*checksum*, broj poslanih bitova) prethodnih aplikacijskih podataka za detekciju grešaka,
- ACK - svaki čvor koji primi točnu poruku prepisuje ovaj recesivni bit u izvornoj poruci s dominantnim bitom, što znači da je poslana poruka bez greške. Ako prijemni čvor otkrije pogrešku i ostavi ovaj bit recesivnim, on odbacuje poruku i čvor koji šalje poruku ponavlja poruku nakon rearbitraže. Tako svaki čvor priznaje (ACK) integritet svojih podataka. ACK sadrži 2 bita, jedan je bit potvrde, a drugi je graničnik,

- EOF - 7-bitno polje, kraj okvira (engl. *End Of Frame*), označava kraj poruke i onemogućuje trpanje bitova, ukazujući na grešku kod trpanja u slučaju da je dominantan. Kada 5 bitova iste logičke razine nastanu u slijedu kod normalne operacije, bit suprotne logičke razine se *natrpa* u podatke,
- IFS - 7-bitno polje, međuokvirni prostor (engl. *Interframe Space*), sadržava vrijeme potrebno da kontroler pomakne ispravno primljen okvir na njegovu valjanu poziciju u međuspremniku poruka.

Prošireni CAN protokol



Slika 3.16: Proširena CAN poruka: 29-bitni identifikator [6, str. 4]

Na slici 3.16 je vidljivo da je proširena CAN poruka ista kao i standardna, uz dodatak:

- SRR - zamjenski udaljeni pristup (engl. *Substitute Remote Request*), zamjenjuje RTR bit u standardnoj poruci kao rezervirano mjesto u proširenom formatu,
- IDE - recesivni bit u proširenju identifikatora (engl. *Identifier Extension* označava da slijedi još identifikatorskih bitova. 18-bitno proširenje slijedi nakon IDE,
- r1 - nakon RTR i r0 bitova, dodan je još jedan rezervirani bit prije DLC bita.

4. Programska podrška

5. Zaključak

LITERATURA

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

[2] *UM10204 I^2C-bus specification and user manual*. NXP Semiconductors, 2021. Rev. 7.0.

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

[4] *RM0090 Reference manual STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs*. ST Microelectronics, 2021. Rev. 9.

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

[6] *SLOA101B Introduction to the Controller Area Network (CAN)*. Texas Instruments, 2002. Revised May 2016.

[7] Wikipedia. I^2 c, 2022. URL `https://en.wikipedia.org/wiki/I%C2%B2C`. Preuzeto: 30. 05. 2022.

[8] Wikipedia. Serial peripheral interface, 2022. URL `https://en.wikipedia.org/wiki/Serial_Peripheral_Interface`. Preuzeto: 20. 06. 2022.

[9] FER ZKIST. FERSAT - opis projekta, 2022. URL `https://www.fer.unizg.hr/zkist/FERSAT/projekt`. Preuzeto: 18. 06. 2022.

Programska potpora za upravljanje kamerom na CubeSat nanosatelitu

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Software for Camera Control on CubeSat Nanosatellite

Abstract

Abstract.

Keywords: Keywords.