

SADRŽAJ

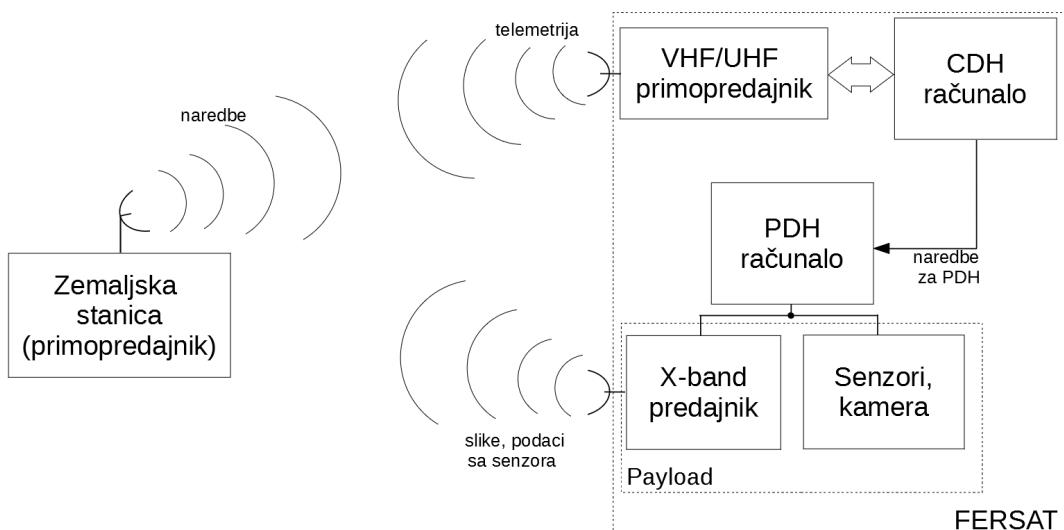
1. Uvod	1
2. Komponente korisnog tereta (<i>payload</i>) satelita i PDH računalo	3
3. Kamera	6
3.1. Pregled tehnologije senzora slike	6
3.2. Načelo rada CMOS senzora	6
3.3. Upravljanje vremenom ekspozicije kod CMOS senzora	9
3.4. Kriterij odabira kamere za satelit	12
3.5. Arducam 5MP Mini Plus	13
3.5.1. Pregled kamere i senzora	13
3.5.2. Ugrađene mogućnosti obrade slike	14
3.5.3. SCCB sučelje	15
4. Vanjska memorija	18
4.1. Načela rada Flash memorije	18
4.1.1. MOS tranzistor s plutajućom upravljačkom elektrodom	18
4.1.2. NOR i NAND Flash memorije	21
4.2. Greške u radu Flash memorije	22
4.2.1. Trajne greške	22
4.2.2. Privremene greške	23
4.2.3. Zadržavanje (retencija) podataka	25
4.3. Winbond W25N01GV NAND Flash memorija	26
5. Razvijena programska potpora	29
5.1. Razvojni sustav, korišteni programski paketi i biblioteke	29
5.2. Kamera	30
5.3. Flash memorija	33
5.3.1. SPI sučelje između Flash memorije i mikrokontrolera	33

5.3.2. Upravljački program (<i>driver</i>) za memoriju	36
5.3.3. Datotečni sustav	37
5.4. Modul s fotosenzorima	49
5.5. X-band odašiljač	51
5.6. Testiranje modula razvijene programske potpore	53
5.7. Integracija operacijskog sustava FreeRTOS	55
6. Zaključak	61
Literatura	62

1. Uvod

FERSAT: projekt izrade satelita

Cilj projekta FERSAT, koji se u trenutku pisanja ovog teksta provodi na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu, je izrada satelita, njegovo lansiranje u Zemljinu orbitu i komunikacija s istim za vrijeme trajanja misije [4]. FERSAT je ujedno i naziv za navedeni satelit. Cilj misije je prikupljanje fotografija Zemlje iz svemira, analiza svjetlosnog onečišćenja i mjerjenje debljine ozonskog sloja Zemlje. FERSAT pripada skupini minijaturnih satelita, preciznije, radi se o nanosatelitu formata 1U (dimenzije 10 cm x 10 cm x 10 cm). Velik broj takvih minijaturnih satelita moguće je postaviti u raketu zajedno s „velikim“ satelitom radi kojeg se raketa lansira. Stoga je cijena slanja minijaturnog satelita u svemir relativno niska. FERSAT bi orbitirao na visini od oko 500 km iznad Zemljine površine (niska Zemljina orbita) u ukupnom trajanju misije od oko 3 godine, prije nego li ga Zemljina atmosfera uspori te padne na površinu¹. Način komunikacije i upravljanja satelitom prikazan je slikom 1.1.



Slika 1.1: Upravljanje i komunikacija s FERSAT satelitom.

¹Satelit tijekom pada izgori i dezintegrira se u bezopasne komadiće.

Putem VHF/UHF (*Very High Frequency/Ultra High Frequency*) radioveze satelitu se izdaju naredbe, a satelit šalje podatke o telemetriji (svom statusu). Za upravljanje satelitom zaduženo je CDH (*Command and Data Handling*) računalo, koje prvenstveno osigurava uključivanje, odnosno isključivanje pojedinih komponenti satelita u zadanim vremenskim trenutcima. Također, na temelju naredbi primljenih sa Zemlje, CDH računalo šalje odgovarajuće instrukcije PDH računalu (*Payload Data Handling*), koje je zaduženo za rad s komponentama korisnog tereta. U korisni teret (engl. *payload*) spadaju senzori i kamere. Iako bi se prikupljeni podaci sa senzora i kamere mogli slati VHF/UHF vezom, odlučeno je da će se za FERSAT razviti predajnik za odašiljanje u X-pojasu (kao jedna od komponenti korisnog tereta), koji će omogućiti puno veću brzinu prijenosa podataka.

Za ispravno funkcioniranje satelita potreban je i niz drugih podsustava. Potrebno je osigurati opskrbu energijom iz baterijskog sustava koji se puni iz solarnih panela. Potreban je sustav za osiguravanje ispravne orientacije satelita (senzori i kamera usmjereni prema Zemlji). Nužna je i mehanička konstrukcija koja će izdržati vibracije prilikom lansiranja, kao i pomno proračunati toplinski dizajn, budući da je satelit u svemiru izložen temperaturnim ekstremima. Neke od komponenti satelita biraju se od komercijalno dostupnih (npr. CDH računalo, sustav za osiguravanje orientacije), dok se druge razvijaju na Fakultetu (komponente korisnog tereta, zemaljska stanica, itd.).

Struktura rada

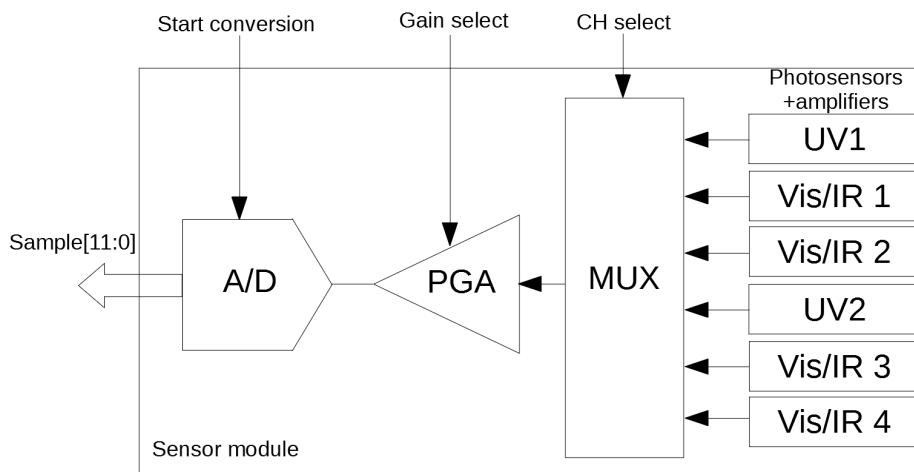
Ovaj rad bavi se razvojem programske potpore za računalo za upravljanje korisnim teretom satelita (PDH računalo). Najprije će se pobliže razmotriti PDH računalo i komponente korisnog tereta (poglavlje 2). Zatim će biti dan dublji teoretski pregled tehnologije senzora slike (kamera) i tehnologije Flash memorije koja je dio PDH računala (poglavlja 3 i 4), kako bi se dalo opravdanje za odabir konkretnih modela navedenih uređaja. Također, za razvoj adekvatne programske potpore nužno je razumijevanje principa rada tih komponenti. Nadalje, bit će opisana razvijena programska potpora po pojedinim modulima (poglavlje 5, odjeljci 5.2-5.5), gdje je poseban naglasak dan na razvijeni datotečni sustav. Na kraju, bit će prikazan način integracije programskih komponenti u više zadatačni sustav korištenjem operacijskog sustava za rad u stvarnom vremenu (poglavlje 5, odjeljak 5.7).

2. Komponente korisnog tereta (payload) satelita i PDH računalo

Komponente korisnog tereta su kamera, modul s fotosenzorima i X-band odašiljač. Slijedi kratak opis njihove uloge unutar misije satelita.

Kamera. Svrha kamere je prikupljanje fotografija Zemlje. Kamera će biti kalibrirana za vjernu reprodukciju boja. Detalji o kameri nalaze se u zasebnom poglavlju.

Modul s fotosenzorima. Modul s fotosenzorima ima dvije zadaće. Prva je analiza svjetlosnog onečišćenja; u tu svrhu na modulu se nalaze četiri fotodiode osjetljive u vidljivom i infracrvenom području, svaka s različitim pojasmnim filtrima (staklo ispred diode). Metodologijom spektralnog razlučenja razvijen je algoritam, u sklopu FER-SAT projekta [19], kojim je moguće procijeniti udio pristiglog zračenja iz svjetlosnih dioda za rasvjetu, naspram konvencionalnih rasvjetnih tijela. Druga zadaća modula je utvrđivanje debljine ozonskog sloja, za to su korištene dvije fotodiode zadužene za mjerjenje reflektiranog UV zračenja. Kompletan modul čine diode zajedno s lancem za akviziciju podataka; prikaz na slici 2.1.

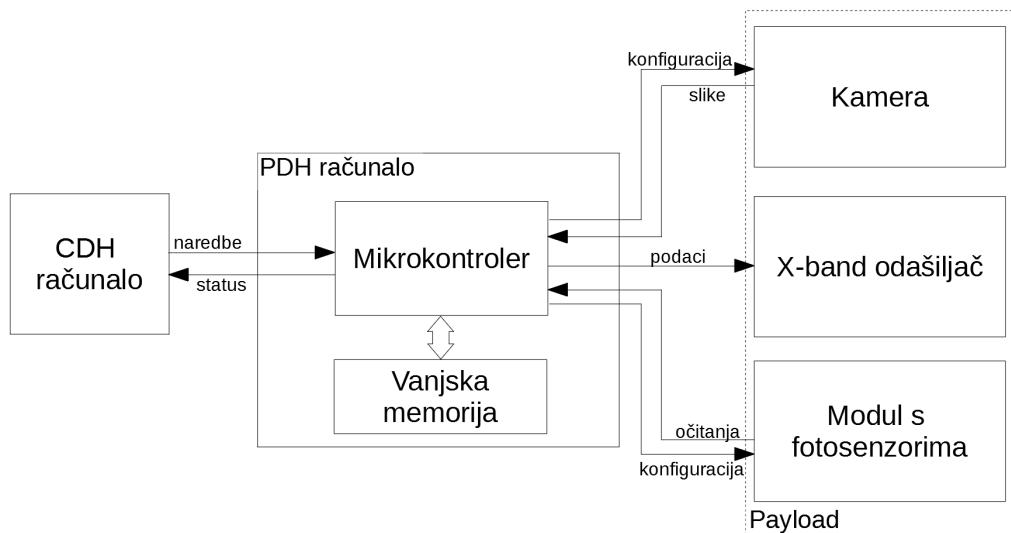


Slika 2.1: Funkcionalna shema modula s fotosenzorima.

X-band odašiljač. Služi za slanje prikupljenih podataka na Zemlju. Brzina prijenosa podataka iznosi između 0.5 i 16 Mbit/s. X-band odašiljač koristi frekvenciju nosioca 10.45 GHz i BPSK ili QPSK modulaciju.

PDH računalo upravlja korisnim teretom. Povezivanje PDH računala i komponenti korisnog tereta prikazano je slikom 2.2. Dužnosti PDH računala su:

- konfiguracija kamere, davanje naredbe za fotografiranje i pohrana fotografija
- konfiguracija modula s fotosenzorima, pokretanje i upravljanje uzorkovanjem, pohrana uzorka
- slanje pohranjenih fotografija i podataka senzora na X-band odašiljač.



Slika 2.2: Povezivanje PDH računala i ostalih komponenti u satelitu.

PDH računalo izvršava naredbe koje mu daje CDH računalo. Primjerice, naredba može biti: „*Pošalji prethodno pohranjenu fotografiju na X-band odašiljač.*“ PDH računalo sastoji se od mikrokontrolera i vanjske memorije. Mikrokontroler koji je odabran za korištenje u FERSAT satelitu je STM32F407 proizvođača STMicroelectronics, temeljen na ARM Cortex-M4 jezgri. Uvidom u druge, već realizirane projekte izrade minijaturnih satelita, ocijenjeno je kako mikrokontroler s dodatnim mehanizmima za poboljšanje pouzdanosti nije potreban (primjerice *lockstep* procesor, kao kod Cortex-R porodice). Odabrani mikrokontroler pokazuje zadovoljavajuće specifikacije vezano za performanse (takt rada jezgre 168 MHz, preko 100 kB radne memorije, DMA jedinice), ta nudi podršku za pregršt komunikacijskih sučelja: SPI, I2C, USART, CAN; nudeći pritom više kontrolera za pojedina sučelja.¹ Tu je i podrška za paralelno su-

¹Dodatna prednost je i dostupnost velikog broja razvojnih sustava na Fakultetu.

čelje za povezivanje kamere (DCMI). Vanjska memorija služi za pohranu prikupljenih podataka te se o njoj detaljno raspravlja u zasebnom poglavlju.

PDH računalo i X-band odašiljač povezuju se putem paralelnog sinkronog komunikacijskog sučelja (detaljnije u poglavlju 5, odjeljak 5.5). Pri tome PDH računalo samo treba slati podatke, dok se modulacija i ostali postupci potrebni za odašiljanje provode autonomno unutar X-band modula. Na modulu s fotosenzorima moguće je odabratи kanal i pojačanje programabilnog pojačala (slika 2.1). Prikupljeni uzorak je 12-bitni. Povezivanje se vrši kombinacijom SPI sučelja i direktnog upravljanja putem GPIO pinova. Sučelje PDH računala prema kameri, kao i sučelje mikrokontrolera i vanjske memorije nije unaprijed zadano te će biti obznanjeno kasnije u tekstu, nakon što se definira konkretan model kamere i memorije. PDH i CDH računalo komunicirat će putem CAN sučelja, ali točan protokol komunikacije nije poznat budući da u trenutku pisanja ovog teksta odabir modela CDH računala još nije dovršen.

3. Kamera

3.1. Pregled tehnologije senzora slike

Senzor slike je elektronički uređaj sačinjen od fotoosjetljivih elemenata posloženih u dvodimenzionalnu matricu koji služi za pretvaranje (vidljive) svjetlosti u digitalnu sliku. Dva najzastupljenija tipa senzora slike danas u uporabi su CCD (*Charge Coupled Device*) i CMOS (*Complementary Metal Oxide Semiconductor*) senzor [17]. Dok CCD senzori, zahvaljujući ponajprije svojoj većoj osjetljivosti, pronalaze uporabu u posebnim profesionalnim aplikacijama, CMOS senzori su se nametnuli kao dominantni u uređajima opće namjene i potrošačkoj elektronici. Razlog tome prvenstveno je proizvodni proces koji omogućuje jednostavnu integraciju senzora i popratnog sklopolja (A/D pretvornici, pojačala, upravljačko sklopolje, procesor za obradu slike itd.), što smanjuje veličinu uređaja, njegovu potrošnju i cijenu. Budući da je senzor u kameri odabran za satelit CMOS tipa, slijedi podrobniji opis načina rada CMOS senzora.

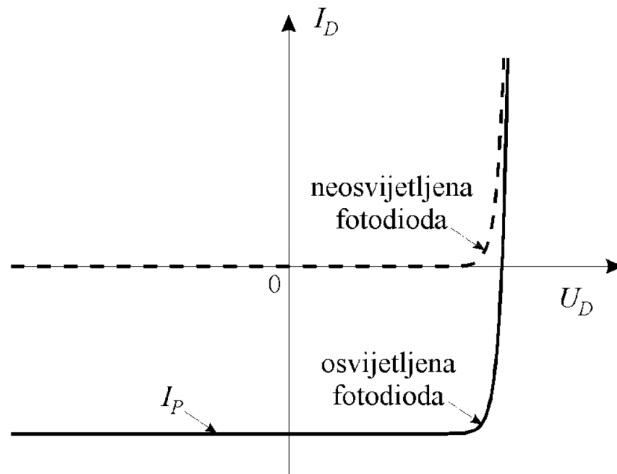
3.2. Načelo rada CMOS senzora

Fotoosjetljivi element koji se koristi kod CMOS senzora je fotodioda (klasični PN spoj ili P+/N/P *pinned* dioda). Ako se fotodioda obasja svjetлом, uslijed fotoelektričnog efekta dolazi do nastanka slobodnih nosioca te, ako je zatvoren strujni krug, počinje teći fotostruja. Ovo se vidi kao pomak krivulje prema dolje na I/U karakteristici diode (slika 3.1). Kako su fotostruje dioda u senzorima slike vrlo malene, jednostavnije je pustiti struju da teče izvjesno vrijeme i integrirati je, tj. izmjeriti ukupno generirani naboј. Taj princip ilustriran je pomoću jednostavnog sklopa sa slike 3.2. Integracija traje toliko dugo koliko je otvorena sklopka. Zatvaranjem sklopke, kondenzator se isprazni te je sklop spreman za novu integraciju. Naboј na kondenzatoru proporcionalan je broju pristiglih fotona, sve do točke zasićenja [9]. Dodavanjem nabojskog pojačala fotodiodi dobiva se pasivni piksel senzor (engl. *Passive Pixel Sensor*, PPS), prikazan

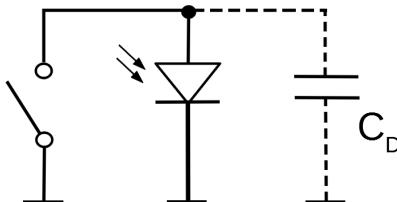
slikom 3.3. Napon na izlazu nabojskog pojačala dan je formulom:

$$U_{IZ} = -\frac{q_{prikljeno}}{C_F} [V] \quad (3.1)$$

Pri tome $q_{prikljeno}$ predstavlja ukupno prikupljeni naboj prilikom integracije, a C_F je iznos kapaciteta kondenzatora u povratnoj vezi operacijskog pojačala. Kod PPS CMOS senzora, diode se slažu u strukturu prikazanu slikom 3.4a. Vrijednosti pojedinog piksela očitavaju se selekcijom pojedinog retka nakon čega analogno-digitalni pretvornik vrši kvantizaciju vrijednosti napona pojedinog stupca (prikaz stupca na slici 3.4b). Pokazuje se kako je pasivni piksel senzor susceptibilan određenim parazitnim efektima¹ koji se rješavaju prelaskom na tzv. aktivni piksel senzor (APS), čija je struktura prikazana slikom 3.5. Njegova glavna odlika je korištenje odvojnog pojačala (naponsko sljedilo) za spoj na sabirnicu stupca.



Slika 3.1: I-U karakteristika fotodiode. I_P označava fotostruju. Preuzeto iz [3].

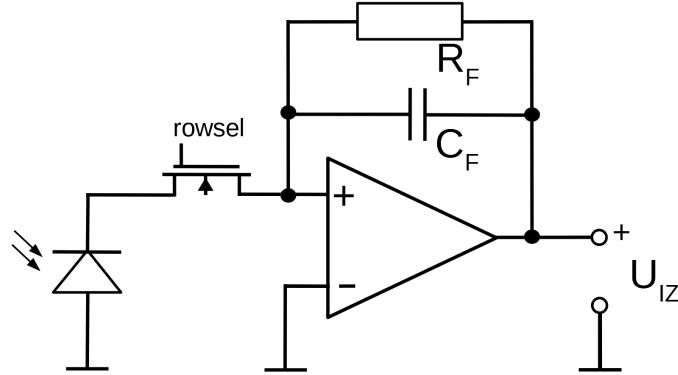


Slika 3.2: Jednostavan sklop za integriranje fotostruje. C_D je parazitni kapacitet diode.

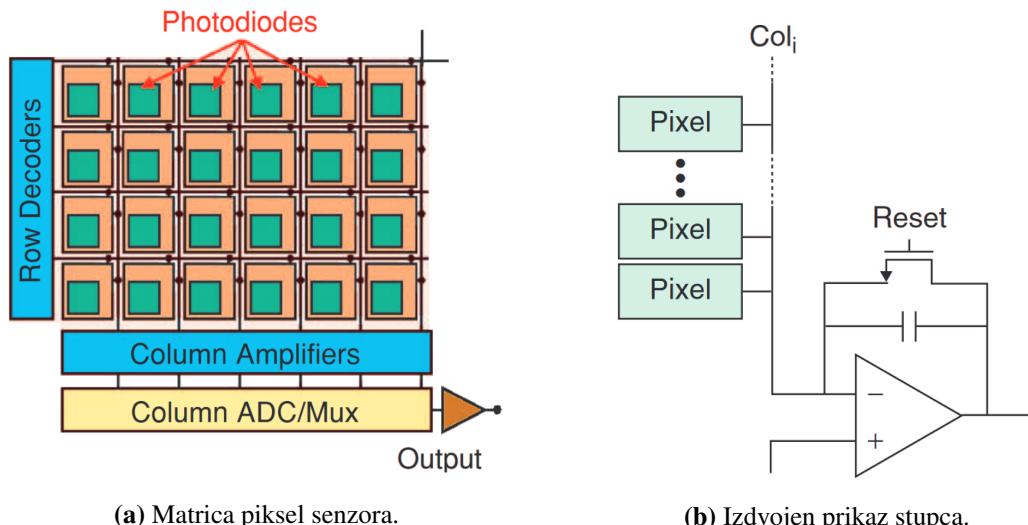
Budući da se u senzorima slike koriste fotodiode koje su podjednako osjetljive za cijeli raspon valnih duljina vidljive svjetlosti, postavlja se pitanje kako nastaje slika u

¹Za detalje pogledati u [18].

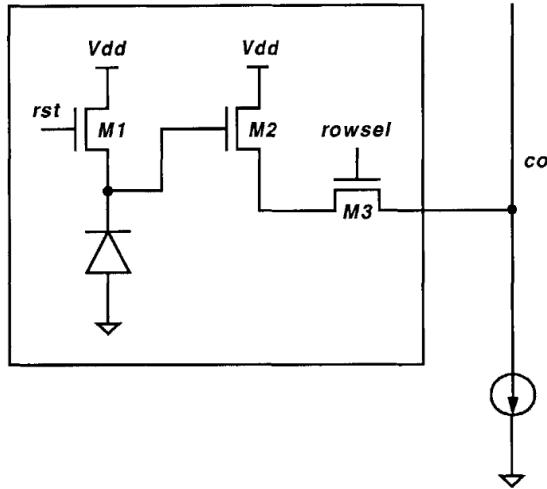
boji. Na odgovor ukazuje slika 3.6: na senzor se preko piksela nanose filtri crvene, zelene i plave boje posloženi u ponavljajući uzorak. Kako bi za svaki piksel postojala vrijednost za sva tri kanala boje, vrijednost nedostajućih kanala dobiva se od susjednih piksela interpolacijom. Navedeni postupak u literaturi se naziva *demosaicing* (raspreplitanje mozaika).



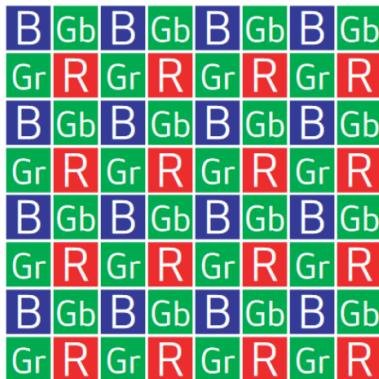
Slika 3.3: Nabojsko pojačalo u sklopu pasivnog piksel senzora (PPS). Tranzistor *rowsel* služi za spajanje diode: u senzoru slike se veći broj dioda spaja na isto pojačalo. Dioda je okrenuta suprotno u odnosu na sliku 3.2 kako bi napon na izlazu imao pozitivan predznak. Otpornik R_F ima vrlo veliku vrijednost otpora i služi osiguravanju statičke struje ulaznom stupnju operacijskog pojačala. Napomena: sklopovlje za reset nije prikazano.



Slika 3.4: Struktura CMOS senzora slike. Preuzeto iz [17].



Slika 3.5: Aktivni piksel senzor s tri tranzistora (3T APS). Strujni izvor na sabirnici stupca služi za postavljanje statičke radne točke odvojnog pojačala (označeno s M2). Preuzeto iz [18].



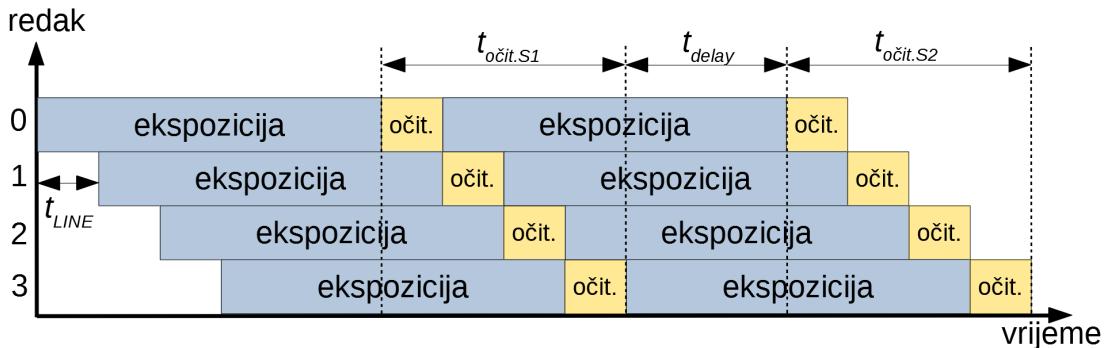
Slika 3.6: Bayerova BG/GR matrica. Preuzeto iz [27].

3.3. Upravljanje vremenom ekspozicije kod CMOS senzora

U kamerama bez fizičke blende (otvora promjenjive veličine između leće i senzora) jedini način upravljanja svjetlinom slike, izuzev promjene pojačanja pojačala, je promjena duljine trajanja integracije svjetla, tj. promjena vremena ekspozicije. Jedna od važnijih razlika CCD i CMOS senzora je u načinu na koji se senzor (elektronički upravljanje) izlaže svjetlu. Kod CCD senzora cijela se matrica izlaže svjetlu istovremeno (*global shutter*), dok se kod CMOS senzora izlaganje vrši redak po redak² (*rolling*

²Postoje i *global shutter* CMOS senzori, no oni plaćaju cijenu manje osjetljivosti zbog dodatnih tranzistora potrebnih po svakom pikselu senzoru, čime se smanjuje udio površine dostupan za fotoosjetljivi element.

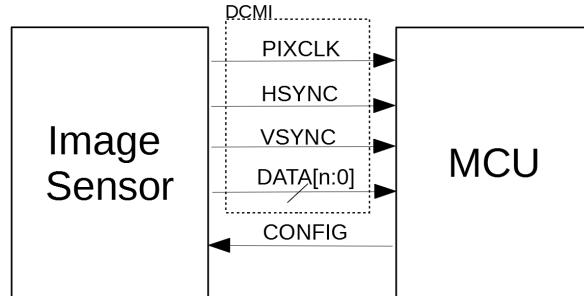
shutter) [29], kako prikazuje slika 3.7. *Rolling shutter* omogućuje uzimanje većeg broja sličica po sekundi (engl. *framerate*) za istu rezoluciju, uz nedostatak da se pojavi izobličenje slike kod fotografiranja objekata koji brzo mijenjanju svoju poziciju unutar kadera.



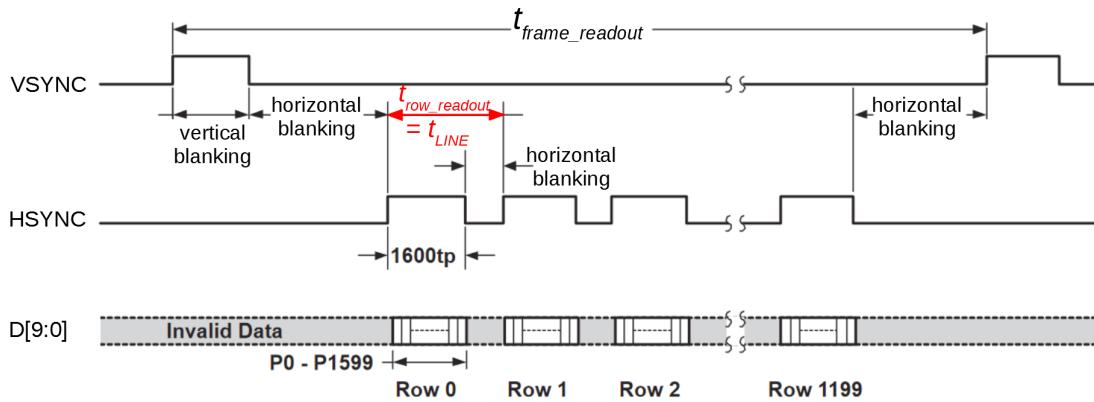
Slika 3.7: Vremenski dijagram eksponiranja redaka na *rolling shutter* senzoru. Kod uzas-topnog fotografiranja, ukoliko je kombinirano vrijeme eksponicije i očitavanja retka dulje od vremena očitavanja slike, između očitavanja slika prisutna je pauza (označena s t_{delay}). Radi kompaktnosti prikaza nacrtan je dijagram za senzor sa samo četiri retka.

Na slici 3.7 vidi se kako nakon završetka eksponiranja retka dolazi do njegovog očitavanja. Očitavanje vrši mikrokontroler preko odgovarajućeg paralelnog komunikacijskog sučelja, poput sučelja DCMI (*Digital Camera Interface*) [32]. DCMI sučelje čini jedna paralelna podatkovna linija i tri sinkronizacijske linije usmjerene od senzora prema mikrokontroleru (slika 3.8). PIXCLK predstavlja tzv. frekvenciju piksela, odnosno radi se o taktu sinkronizacije prijenosa podataka: na svaki padajući (ili rastući) brid, mikrokontroler s paralelnih podatkovnih linija uzorkuje vrijednost jednog piksela. Aktivnost linije VSYNC označava granicu između dviju sličica (engl. *frames*), a aktivnost linije HSYNC granicu između dvaju redaka (slika 3.9). Aktivnost bilo koje od SYNC linija označava da se vrijednost s podatkovnih linija ne uzorkuje. Pri konfiguraciji parametara CMOS senzora vrijeme eksponicije tipično se specificira kao broj jedinica koje imaju trajanje vremena linije t_{LINE} (*line time*). Sa slike 3.7 se vidi kako t_{LINE} predstavlja vrijeme od početka eksponicije jednog retka do početka eksponicije sljedećeg retka. Minimalno vrijeme eksponicije, dakle, iznosi 1 t_{LINE} . Neki senzori nude mogućnost postavljanja vremena eksponicije da iznimno bude manje od 1 t_{LINE} ; tada se između eksponiranja i očitavanja retka umeće mrtvo vrijeme. Maksimalno vrijeme eksponicije u teoriji može biti proizvoljno dugo, samo valja imati na umu da za održavanje maksimalnog broja sličica u sekundi kombinirano vrijeme eks-

pozicije i očitavanja retka mora biti manje od vremena očitavanja jedne sličice (slika 3.7). Promjena vrijednosti t_{LINE} za efekt ima i produljenje aktivnosti signala HSYNC (*horizontal blanking*) ili VSYNC (*vertical blanking*).



Slika 3.8: DCMI sučelje između senzora slike i mikrokontrolera. Sučelje za konfiguraciju senzora slike nije unutar DCMI specifikacije.



Slika 3.9: Vremenski dijagram prijenosa 10-bitne slike rezolucije 1600x1200 putem DCMI sučelja. VSYNC je aktivan visoko, a HSYNC je aktivan nisko. Uočiti kako je t_{LINE} jednak vremenu očitavanja retka. Prerađeno iz [26].

Kako onda odrediti vrijeme ekspozicije u sekundama? Za to je potrebno poznavati iznos t_{LINE} . Kod senzora s DCMI sučeljem vrijeme linije može se dobiti formulom:

$$t_{LINE} = \frac{1}{f_{pix}}(n_{pix,hor} + n_{blank,hor}) \quad (3.2)$$

pri čemu je f_{pix} frekvencija takta za piksele (PIXCLK), $n_{pix,hor}$ broj piksela u jednom retku, a $n_{blank,hor}$ broj umetnutih *blanking* perioda tijekom očitavanja retka. Navedene parametre definira proizvođač ili se mogu konfigurirati u postavkama senzora.

3.4. Kriterij odabira kamere za satelit

Kriteriji odabira kamere za satelit mogu se podijeliti na dva potkriterija: prvi je vezan prvenstveno uz željene karakteristike senzora slike, drugi je vezan uz način povezivanja kamere s mikrokontrolerom. Kriteriji glase:

(a) Rezolucija senzora mora biti minimalno 5 MP (megapiksela). Slike moraju biti u boji. Mora postojati mogućnost produciranja slika bez kompresije (RAW format). Vrijeme ekspozicije mora se moći podešavati ručno. Leća kamere mora biti izmjenjiva, koristeći pritom neki od standardnih načina montiranja.

(b) Kamera se s mikrokontrolerom mora moći povezati nekim od standardnih se-rijskih komunikacijski sučelja (SPI, USART, I2C) ili putem paralelnog DCMI sučelja. Povezivanje putem USB sučelja nije prihvatljivo. Način korištenja kamera (konfiguracija parametara senzora, protokol prijenosa slike na mikrokontroler) mora biti precizno dokumentiran.

Prvi kriterij definirao je projektni tim zadužen za obradu slike sa satelita, na temelju namjene kamere kao komponente korisnog tereta, navedene u poglavlju 2. Zahtjev za korištenjem navedenih komunikacijskih sučelja proizlazi iz činjenice da odabrani mikrokontroler sadrži kontrolere za ta sučelja, u vidu svojih perifernih jedinica. Korištenje bilo kakvog ne-standardiziranog (*proprietary*) ili od strane mikrokontrolera nepodržanog sučelja i pripadajućeg komunikacijskog protokola uvodi komplikacije u vidu potrebne dodatne programske (možda i sklopovske) potpore. Proizvođači kamera u tom slučaju tipično daju na raspolaganje programsku potporu koju su oni razvili³, no ona ne mora biti otvorenog koda i upitne je portabilnosti. USB sučelje, pak, poprilično je složeno i korištenje same periferije bez biblioteka s visokom razinom apstrakcije, što je bio jedan od ciljeva u ovom radu, iziskivalo bi značajne napore. Ovi zahtjevi na sučelje, nažalost, eliminiraju veliki broj kamera na tržištu koje se deklariraju kao industrijske ili kao kamere za znanstvenu uporabu.

Kao jedan od kandidata razmotren je UCAM-III proizvođača 4D Systems [8]. Kamera koristi UART sučelje, dobro je dokumentirana, ima senzor u boji s mogućnošću produciranja slika u RAW formatu. Nažalost, senzor zakazuje po pitanju rezolucije koja iznosi samo 640 s 480 točaka (VGA rezolucija).

Sljedeći razmotreni kandidat je 5MP OV5642 proizvođača Arducam [11]. Kako i ime govori, kamera koristi 5 megapikselski (2592 s 1944 točaka) senzor OV5642 proizvođača Omnipixel, koji ispunjava sve navedene kriterije. Za konfiguraciju senzora koristi se I2C, a za prijenos slike DCMI sučelje. Arducam u svojoj ponudi ima još

³Primjer takvog sučelja je BCON proizvođača Basler [15].

jednu kameru s istim senzorom koja ne koristi DCMI, već SPI sučelje; radi se o modelu 5MP Mini Plus. Dodatno, taj model ima i međuspremnik za privremenu pohranu slike, što je velika prednost. Tu prednost oslikava primjer koji slijedi. Neka je na raspolaganju mikrokontroler sa 100 kB RAM memorije, DCMI sučeljem povezan na kameru, te SPI sučeljem spojen na vanjsku Flash memoriju. Ako se slika veličine 5 MB želi pohraniti na Flash, RAM memorija mora se koristiti kao međuspremnik za pohranu fragmenata slike. Mora se osigurati da je mikrokontroler u stanju dovoljno brzo očitati podatke s DCMI sučelja i poslati ih na vanjsku memoriju, inače će doći do prepisivanja podataka (*data overrun*) i gubitka (kvarenja) slike. Kako bi se ovo osiguralo, tipično je nužno koristiti DMA jedinice prilikom prijenosa podataka. S druge strane, Arducam Mini 5MP Plus cijelu sliku najprije spremi u vlastiti međuspremnik, a mikrokontroler je zatim očitava proizvoljno sporo, bez opasnosti kvarenja podataka. Dodatna prednost korištenja SPI sučelja nad DCMI sučeljem je ušteda u broju korištenih GPIO pinova mikrokontrolera. Zbog svega navedenog, za kameru je odabran upravo Arducam Mini 5MP Plus.

3.5. Arducam 5MP Mini Plus

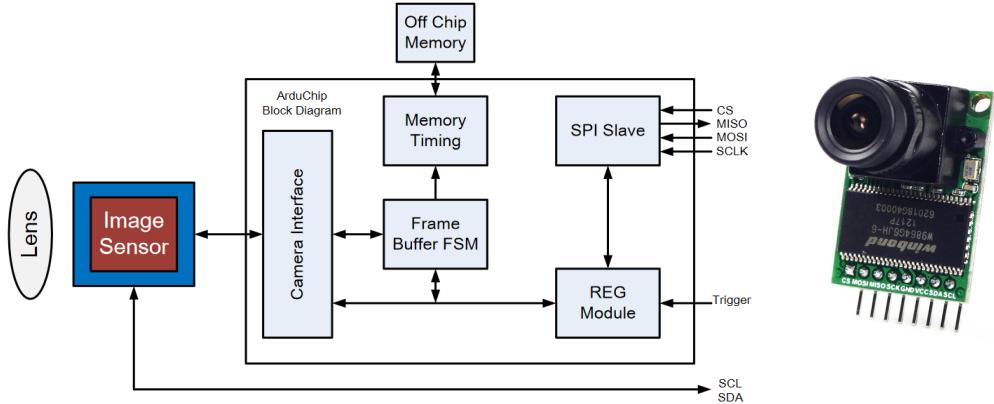
3.5.1. Pregled kamere i senzora

Blok dijagram modula kamere prikazan je slikom 3.10a, a fizički izgled uređaja slikom 3.10b. S blok dijagrama vidljivo je kako su tri glavna integrirana kruga na kameri senzor slike, memorija za privremenu pohranu slike te Arduchip sklop. Senzor slike je Omnivision OV5642 te je detaljnije opisan niže u tekstu. Memorija je kapaciteta 8 MB te je kao takva dostašna za pohranu 8-bitnih RAW slika u punoj rezoluciji. Arduchip je zadužen za generiranje upravljačkih signala, realizaciju SPI sučelja i povezivanje senzora slike i memorije za privremenu pohranu. Na uređaju je Arduchip realiziran korištenjem CPLD-a.

Kamera prima napajanje od 3.3 ili 5 V, koristi I2C sučelje za direktnu konfiguraciju parametara senzora te SPI sučelje za prijenos slike, kao i za primanje nekih naredbi, poput naredbe za početak fotografiranja. Istu naredbu moguće je izdati i preko zasebnog *trigger* priključka.

Blog dijagram integriranog kruga na kojem se nalazi senzor prikazan je slikom 3.11, te se sastoji od sljedećih funkcionalnih cjelina: senzora slike u užem smislu, procesora za digitalnu obradbu slike, sklopolja za generiranje signala vremenskog vođenja i sklopolja za ostvarivanje sučelja prema drugim komponentama. Senzor

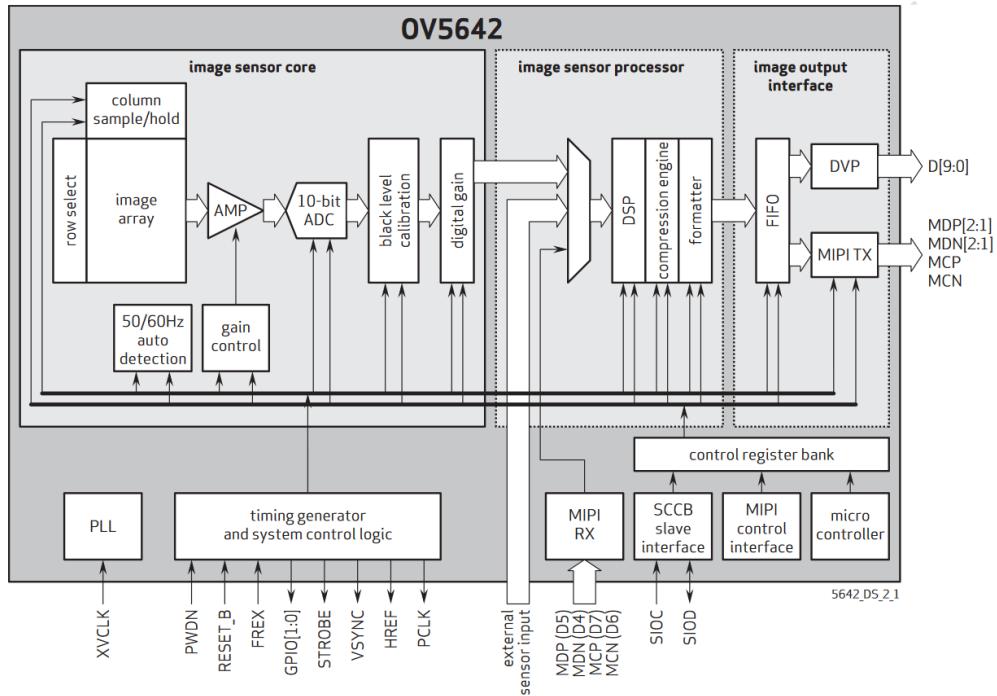
koristi BG/GR Bayerovu matricu (slika 3.6), te može producirati slike bez kompresije s pikselima kvantiziranim na maksimalno 10 bita. Ugrađeni procesor omogućuje i stvaranje komprimiranih slika u JPEG formatu.



(a) Blok dijagram.

(b) Fizički izgled modula.

Slika 3.10: Arducam 5MP Mini Plus kamera. Preuzeto iz [11].



Slika 3.11: Blok dijagram OV5642 senzora slike. Preuzeto iz [27].

3.5.2. Ugrađene mogućnosti obrade slike

Zahvaljujući integriranom procesoru, nad slikom je moguće provesti niz različitih postupaka obrade prije negoli se ona pošalje mikrokontroleru. Svrha tih postupaka je

poboljšanje percipirane kvalitete slike. Dok je to korisno u uređajima potrošačke elektronike (primjerice fotoaparat na mobitelu) kojima je ovaj senzor primarno namijenjen, navedene obrade mogле bi utjecati na rezultate mjerena koja se žele provesti pomoću satelita (utjecaj na točnost kalibracije boje). U nastavku teksta kratko se diskutiraju važnije metode digitalne obrade slike koje omogućuje senzor te se ocjenjuje je li pojedinu funkciju obrade nužno deaktivirati ili ne.

(1) Kalibracija razine crne boje. Uslijed termičkih procesa na fotodiodama dolazi do generiranja naboja čak i kad one nisu izložene svjetlu. Ovo se manifestira kao pomak nule vrijednosti piksela, odnosno crna boja na slici blijedi. Zbog toga senzor ima određeni broj prekrivenih piksel senzora, koji nisu izloženi svjetlu, pa izmjerene vrijednosti tih piksela može koristiti za kompenzaciju pomaka nule. Kalibraciju razine crne boje valja uključiti.

(2) Automatski balans bijele boje. Balans bijele boje podrazumijeva množenje vrijednosti piksela sva tri kanala boje (kod slike u RGB formatu) različitim faktorima za svaki kanal, kako bi se postigao vjerniji prikaz boja, odnosno kako bi bijela boja izgledala kao bijela boja. Balans bijele boje potrebno je isključiti.

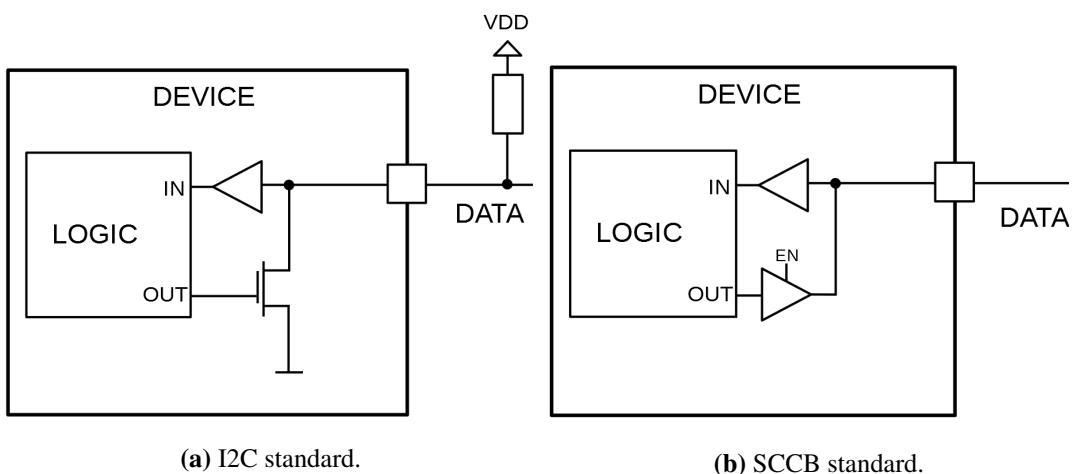
(3) Gama korekcija. Vrijednost piksela na slici proporcionalna je intenzitetu svjetla kojem je fotodioda bila izložena prilikom fotografiranja. No, ljudi svjetlinu ne percipiraju linearно. Svrha gama korekcije je promjena svjetline slike kako bi kontrast bio vjerniji stvarnom. Za svaki piksel moguće je izračunati tzv. kanal svjetline, čija se vrijednost potencira tzv. gama faktorom, te je ovu promjenu moguće vratiti natrag u RGB prostor [35]. Za mjerena na FERSAT-u nužno je da se sačuvaju izvorni linearni odnosi, pa gama korekcija mora biti isključena.

(4) Uklanjanje neispravnih piksela. Zbog pogrešaka prilikom proizvodnje senzora slike, moguće je da neki pikseli stalno poprimaju minimalnu ili maksimalnu moguću vrijednost. Piksel se može detektirati kao neispravan ako njegova vrijednost uveleike odudara od njemu susjednih. Postoji više metoda različitih stupnjeva složenosti kako se vrijednost neispravnog piksela može pretpostaviti [2]. Dok korisnička dokumentacija ne navodi tip metode koju koristi ovaj senzor, patentne prijave sugeriraju da bi se moglo raditi o vrlo jednostavnom kopiranju vrijednosti susjednog, ispravnog piksela [20]. Uklanjanje neispravnih piksela može se uključiti.

3.5.3. SCCB sučelje

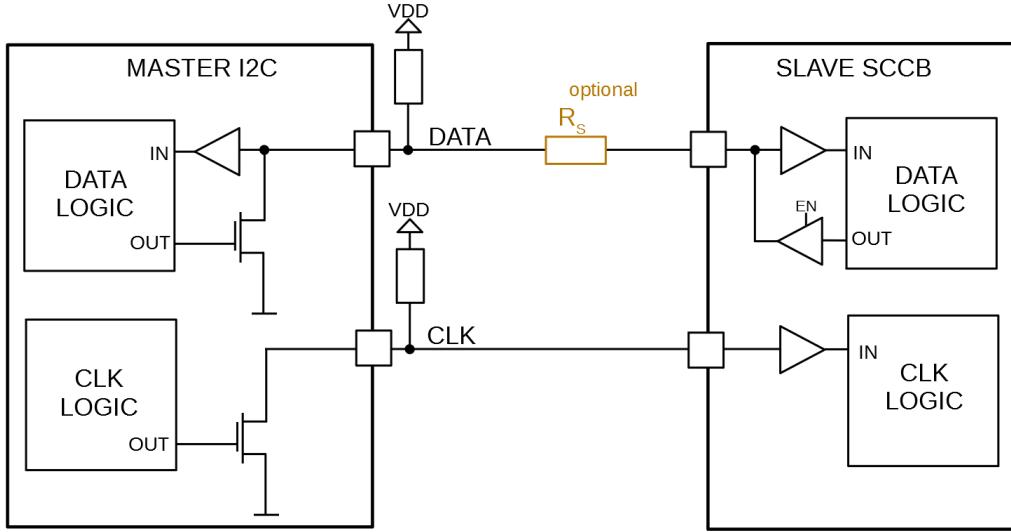
Pozorniji promatrač uočit će kako na blok dijagramu senzora (slika 3.11) nigdje nije naznačeno I2C sučelje, ali postoji jedinica nazvana „*SCCB slave interface*“. SCCB

(*Serial Camera Control Bus*) serijsko je sinkrono komunikacijsko sučelje čiju specifikaciju daje tvrtka Omnivision [25]. Iako između I2C i SCCB sučelja postoje razlike, povezivanje jednog I2C upravljača (*master*, u ovom slučaju mikrokontrolera) i jednog SCCB izvršioca (*slave*, senzora) moguće je bez poteškoća. I jedno i drugo sučelje koriste jednu podatkovnu (*data*) liniju i jednu liniju za takt (*clock*). Kod I2C sučelja, podatkovna linija i linija za takt povezane su preko *pull-up* otpornika na visoku naponsku razinu [24]. Uređaj koji želi postaviti podatak na sabirnicu čini to preko tranzistora u spoju otvorenog kolektora (ili odvoda), spajajući pritom sabirnicu na nisku naponsku razinu (slika 3.12a). Kod SCCB sučelja, koristi se izlazni pogonski sklop u *push-pull* konfiguraciji, s mogućnošću postavljanja stanja visoke impedancije (slika 3.12b). Kako bi se izbjegao potencijalni nastanak konfliktog stanja na sabirnici, kod SCCB sučelja preporuka je postaviti otpornik serijski na podatkovnu liniju. No kako u našem slučaju nema više od jednog I2C upravljača na sabirnici, ne dolazi do procesa arbitracije pa konfliktna stanja ne bi smjela nastati. Način povezivanja I2C upravljača i SCCB izvršioca prikazan je slikom 3.13.



Slika 3.12: Spoj uređaja na podatkovnu liniju kod I2C i SCCB standarda. Spoj na liniju takta vrši se na isti način.

Nadalje, potrebno je provjeriti postoje li kakve razlike u komunikacijskom protokolu. Generiranje *start* i *stop* uvjeta jednako je u oba sučelja. Oba sučelja također prenose 8-bitne podatke. Prilikom adresiranja izvršioca, I2C koristi 7-bitnu adresu nakon koje slijedi R/W bit, ovisno o tome želi li se od izvršioca čitati ili u njega upisivati. SCCB uređaji pak imaju specificiranu jednu 8-bitnu adresu prilikom čitanja, a drugu 8-bitnu adresu prilikom pisanja, s razlikom u posljednjem, najmanje značajnom bitu. Jasno je da su ova dva formata adresiranja jednaka, ali ovaj detalj valja imati na umu prilikom



Slika 3.13: Spoj I2C upravljača i SCCB izvršioca.

programiranja I2C kontrolera na *master* uređaju. Druga razlika po pitanju protokola odnosi se na *Acknowledge* bit (bit potvrde o prijemu), koji slijedi nakon svakog poslanog 8-bitnog podatka. Kod I2C standarda, niska razina prilikom devetog *clock* pulsa označava *Acknowledge*, a visoka razina *Not Acknowledge*. Ako izvršioc kojem je poslan podatak ne potvrdi prijem, upravljač treba ili prekinuti komunikaciju ili generirati ponovljeni *start* uvjet. Prema SCCB specifikaciji, izvršioc za *Acknowledge* bit može postaviti bilo koju vrijednost (visoko, nisko, visoka impedancija). Kako provjeriti o kojoj se vrijednosti zaista radi? Najjednostavnije je povezati mikrokontroler i senzor i isprobati. Pokazuje se da I2C kontroler na mikrokontroleru ulazi u takva stanja (unutar svog automata stanja) koja potvrđuju da senzor zaista generira *Acknowledge* uvjet. SCCB standard nadalje specificira da upravljač, nakon što mu je izvršioc poslao prvi i, prema SCCB specifikaciji, jedini 8-bitni podatak, mora generirati *Not Acknowledge* uvjet, što konformira s I2C standardom.⁴

⁴Iz navedenog slijedi kako je senzor u svojstvu SCCB izvršioca u potpunosti kompatibilan s I2C upravljačem. O razlozima zašto senzor onda ne koristi I2C sučelje može se samo nagađati.

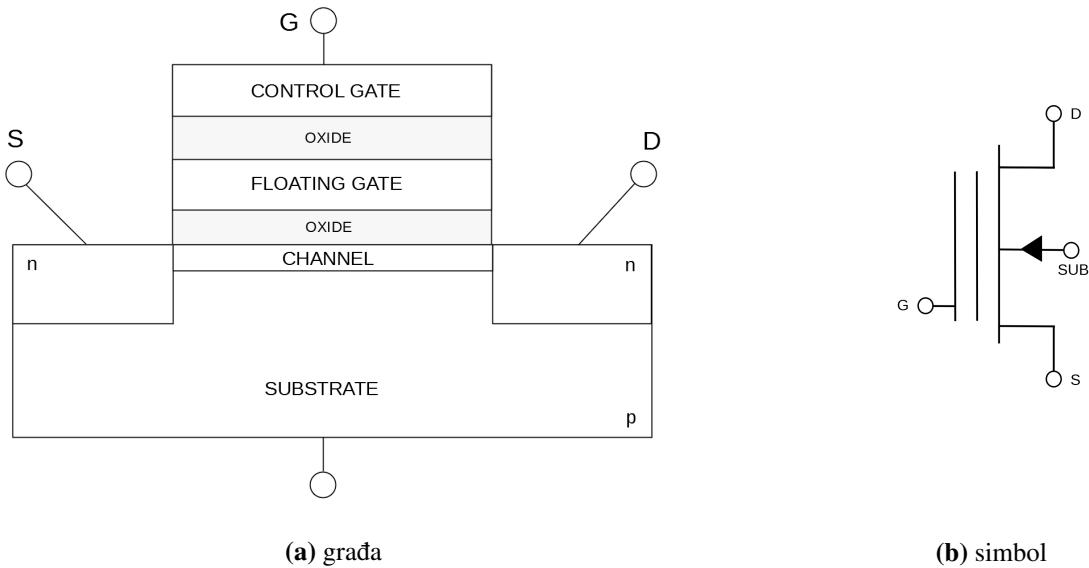
4. Vanjska memorija

4.1. Načela rada Flash memorije

Flash memorija pripada skupini memorija koje zadržavaju podatke nakon isključenja napajanja (engl. *non-volatile storage*) [37]. Prve Flash memorije javljaju u periodu kada je tehnologija EEPROM memorije (engl. *Electiracally Erasable Programmable Read Only Memory*) već bila etabrirana. Osnovna prednost Flash memorije nad EEPROM memorijom je manje zauzeće površine čipa po bitu, odnosno posljedično, manja cijena za isti kapacitet. EEPROM memoriji tipično su potrebna dva tranzistora po memorijskoj ćeliji, dok je Flash memoriji dovoljan jedan. Kod EEPROM memorije, pisanje i brisanje moguće je na razini individualnog okteta. Flash memorija također omogućuje pisanje samo jednog okteta, no brisanje je moguće samo u većim jedinicama, tzv. blokovima (tipično par kB i više). Osnovni gradivni element memorijske ćelije i EEPROM i Flash memorije je MOSFET s plutajućom upravljačkom elektrodom, čiji je princip rada opisan u sljedećem odjeljku.

4.1.1. MOS tranzistor s plutajućom upravljačkom elektrodom

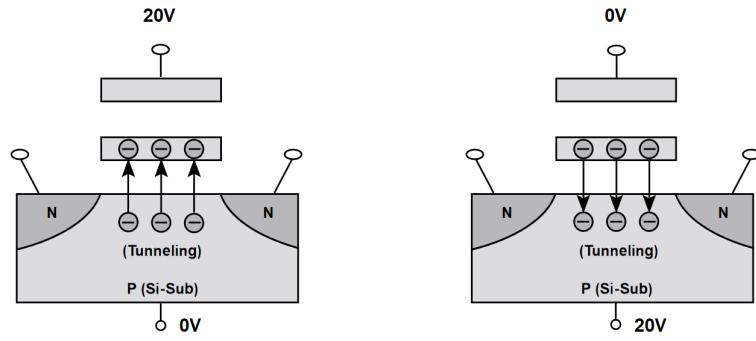
Građa MOSFET-a s plutajućom upravljačkom elektrodom prikazana je na slici 4.1a, a simbol komponente na slici 4.1b. Sa slike 4.1a se vidi kako je između upravljačke elektrode (*control gate*) i kanala deponirana nakupina vodljivog materijala koja je oksidom električki izolirana od ostalih elemenata u tranzistoru. U literaturi na engleskom jeziku ta se nakupina naziva *floating gate* [5]. Metode kojima se elektroni mogu dovesti i odvesti s plutajuće upravljačke elektrode oslanjaju se na kvantomehaničke fenomene [7]. Jedna od metoda je Fowler-Nordheimovo tuneliranje, koje podrazumijeva primjenu visokog napona (oko 20 V) između upravljačke elektrode i supstrata. Ovisno o polaritetu, elektroni se dovode ili odvode, kako prikazuje slika 4.2. Alternativna metoda dovođenja elektrona je tzv. injekcija vrućih elektrona (*hot electron injection*), tj. primjena visokonaponskog pulsa između upravljačke elektrode i elektrode



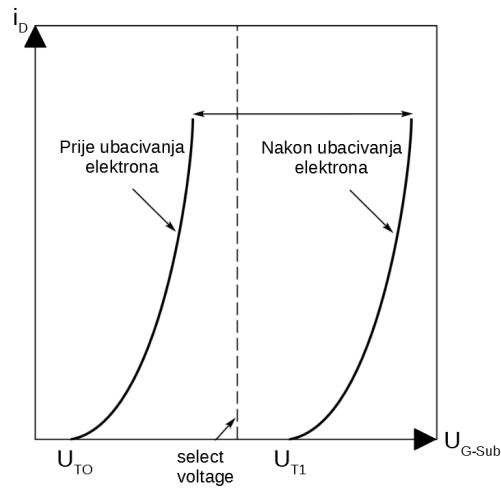
Slika 4.1: N-kanalni MOS tranzistor s plutajućim *gateom*.

dovoda (*source*). Odvođenje elektrona ovom metodom nije moguće. Prisustvo elektrona na plutajućoj upravljačkoj elektrodi za posljedicu ima promjenu napona praga tranzistora, kako prikazuje slika 4.3. Ovo svojstvo može se iskoristiti kako bi se na tranzistoru pohranila vrijednost jednog bita. Princip pohrane ilustrira sklop prikazan slikom 4.4. Neka u izbrisanim stanju (bez prisustva elektrona na plutajućoj upravljačkoj elektrodi) tranzistor ima napon praga U_{T0} , a u programiranom stanju U_{T1} , pri čemu je $U_{T1} > U_{T0} > 0 \text{ V}$. Dok se ne čita vrijednost pohranjena na tranzistoru (tranzistor nije selektiran), na upravljačku elektrodu primjenjuje se napon $U_G = 0 \text{ V}$. Bez obzira na napon praga, tranzistor je u stanju zapiranja. Ako se pohranjena vrijednost želi očitati, primjenjuje se napon upravljačke elektrode U_G takav da je $U_{T0} < U_G < U_{T1}$. Ovaj napon U_G označen je kao *select voltage* na slici 4.3. U slučaju da je tranzistor u izbrisanim stanju, on će voditi te će se na izlazu očitavati 0 V. U programiranom pak stanju tranzistor ne vodi, te se na izlazu očitava VDD. Kod Flash memorije konvencija je da se u izbrisanim stanju (niska naponska razina na izlazu) vrijednost bita čita kao „1“, a u programiranim stanju (visoka razina napona na izlazu) vrijednost bita čita kao „0“.

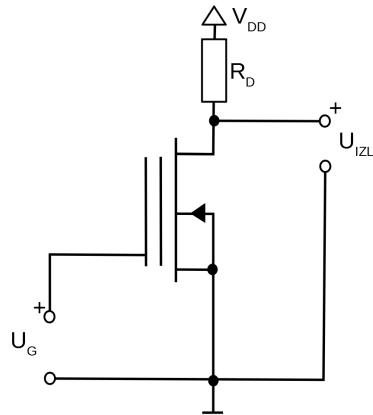
Valja još istaknuti da je u jednom tranzistoru moguće pohraniti i više od dva logička stanja; moduliranjem količine elektrona koji se ubacuju na plutajuću upravljačku elektrodu moguće je postići veći broj diskretnih stanja napona praga. Tada se govori o višerazinskoj memorijskoj ćeliji (MLC, engl. *Multi-Level Cell*). Standardna ćelija s dva stanja naziva se SLC (*Single-Level Cell*). U dalnjim opisima u ovom tekstu podrazumijevat će se da memorije koriste SLC ćelije.



Slika 4.2: Fowler-Nordheimoveo tuneliranje: dovođenje elektrona (lijevo) i odvođenje elektrona (desno) s plutajuće upravljačke elektrode. Preuzeto iz [37].



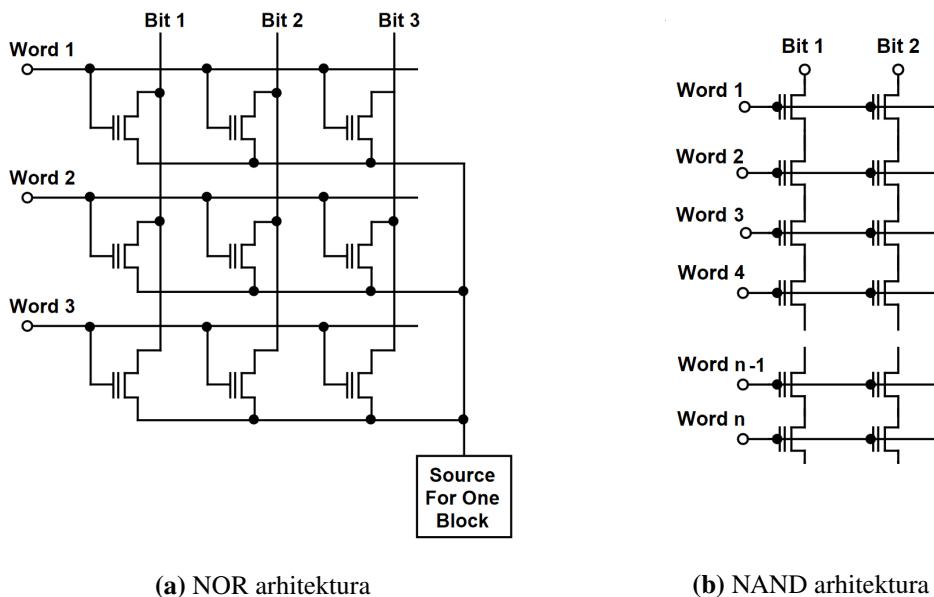
Slika 4.3: Pomak napona praga tranzistora uslijed ubacivanja elektrona na plutajuću upravljačku elektrodu. Prilagođeno iz [5].



Slika 4.4: Električna shema jednostavne memorijske ćelije. Napomena: u stvarnim memorijama se ne koristi otpornik na elektrodi odvoda (drain) zbog velike statičke disipacije, ovdje je stavljen zbog zornijeg prikaza.

4.1.2. NOR i NAND Flash memorije

Opisane MOS tranzistore s plutajućom upravljačkom elektrodom potrebno je nekako složiti u strukturu kako bismo dobili funkcionalnu memoriju. Kod Flash memorija najčešće je korišteno slaganje u NOR (slika 4.5a) i NAND (slika 4.5b) strukturu. I u jednom i u drugom slučaju čitanje iz memorije radi se selektiranjem željene *word* linije i subsekventnim očitavanjem vrijednosti na svim *bit* linijama. Kod NOR arhitekture ponašanje pojedine celije jednako je onom opisanom u primjeru sa slike 4.4 iz prethodnog odjeljka. Valja uočiti da se prilikom čitanja elektroda dovoda i supstrat nalaze na istom potencijalu. Kod NAND arhitekture tranzistori za pojedini bit se spajaju serijski. Kako bi se mogla očitati vrijednost za željenu riječ, napon koji se primjenjuje na *word* linije koje nisu selektirane mora biti takav da tranzistor uvijek vodi, neovisno o svome stanju programiranja. To se postiže primjenom napona koji je veći od napona praga tranzistora i u izbrisanim i u programiranim stanju. Također, budući da se tijekom čitanja tranzistori koriste kao sklopke, supstrat i elektroda dovoda ne smiju biti međusobno spojeni. U tom slučaju, kada se govori o naponu upravljačke elektrode, podrazumijeva se razlika protencijala upravljačke elektrode i supstrata.



Slika 4.5: Arhitekture Flash memorije. Napomena: spoj elektrode substrata nije prikazan; za detalje vidjeti objašnjenje u tekstu. Preuzeto iz [7].

S korisničke perspektive, temeljna razlika NOR i NAND Flash memorije je u sučelju za pristup uređaju. NOR memorija koristi odvojenu podatkovnu i adresnu sabirnicu što omogućuje izravan nasumični pristup (engl. *random access*) bilo kojoj memorijskoj

lokaciji. Ovo NOR Flash memorije čini idealnim za pohranu i izvođenje programskog koda. S druge strane, pristup memorijskim lokacijama kod NAND memorije izvodi se neizravno, preko međuspremnika. Primjerice, prilikom čitanja, najprije je potrebno dati zahtjev za prebacivanje sadržaja nekog raspona memorijskih lokacija u međuspremnik, a zatim je moguće izravno čitanje iz međuspremnika. No, NAND memorije imaju prednost u većoj brzini pisanja. To je posljedica korištenja Fowler-Nordheimovog tuneliranja za programiranje tranzistora, pri čemu je moguće veliki broj tranzistora programirati istovremeno. NOR memorije za programiranje koriste injekciju vrućih elektrona pri čemu je programiranje moguće samo oktet po oktet. Iz navedenog slijedi kako je NAND Flash memorija pogodna za primjenu kao uređaj za pohranu opće namjene, primjerice za pohranu slika, videozapisa itd.

4.2. Greške u radu Flash memorije

Greške u radu Flash memorije dijele se na trajne i privremene [37]. Trajna greška je takvo neispravno stanje bita za koje ne postoji mehanizam oporavka. Privremene greške podrazumijevaju neispravna stanja bita koja je moguće ispraviti, tipično brisanjem bloka.

4.2.1. Trajne greške

Svako programiranje i brisanje memorijske ćelije izlaže oksidni sloj između kanala tranzistora i plutajuće upravljačke elektrode izvjesnom naprezanju materijala. Kako broj ciklusa pisanja i brisanja raste, u oksidnom sloju ostaje zarobljen sve veći broj elektrona. To za posljedicu ima trajni pomak razine napona praga prema programiranom stanju. U jednom trenutku napon praga brisanjem više nije moguće spustiti dovoljno nisko te se ćelija trajno nalazi u programiranom stanju. Prilikom pokušaja brisanja bloka, interno sklopolje memorije detektirat će da postoji bit koji nije izbrisani te će signalizirati grešku. Navedeni blok tada valja proglašiti neispravnim (engl. *bad block*) te ga se više ne smije koristiti [37]. Vrlo rijetko, može se dogoditi da bit trajno ostane u izbrisanim stanju. Memorija će tada javiti grešku prilikom programiranja te je opet blok potrebno proglašiti neispravnim.¹ Proizvođači memorija

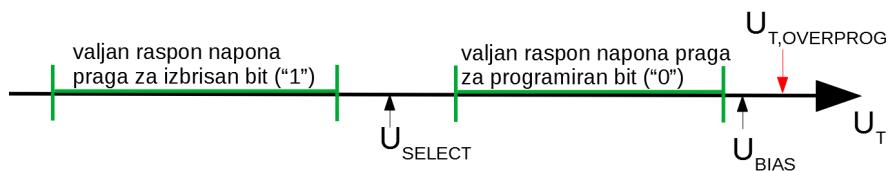
¹Možda je zbunjujuće da se zbog greške u samo jednom bitu cijeli blok od nekoliko desetaka kB treba proglašiti neispravnim, no takve su specifikacije proizvođača. U suprotnom, trebao bi postojati mehanizam koji može pratiti koji su točno bitovi neispravni. Nakon pojave jednog neispravnog bita u bloku za očekivati je i pojavu drugih, budući da su sve ćelije bile izložene sličnom naprezanju.

specificiraju izdržljivost (engl. *endurance*) memorije kao broj ciklusa pisanja i brisanja određenog bloka do kojeg garantiraju njegov ispravan rad.² Za NOR memorije izdržljivost iznosi do 100 000 ciklusa, a za NAND memorije do 1 000 000 ciklusa. NOR Flash memorije manje su izdržljive zbog toga što je oksidni sloj izložen većem naprezanju uslijed različitih postupaka za dovođenje i odvođenje elektrona s plutajuće upravljačke elektrode (vidi prethodni odjeljak). Uslijed ograničenja proizvodnog procesa, proizvođačima je dozvoljena isporuka memorija s već postojećim neispravnim blokovima (do 2% ukupnog broja blokova). Takvi blokovi tada su označni posebnim vrijednostima na pojedinim memorijskim lokacijama.

4.2.2. Privremene greške

Slijedi opis privremenih grešaka koje se ponajprije odnose na NAND Flash memoriju.

Preprogramiranje (engl. *over programming*) je greška do koje dolazi zbog ubacivanja prevelike količine elektrona na plutajući *gate*. Posljedično, napon praga pomiče se na razinu veću od razine napona koja se postavlja na neselektiranu *word* liniju (slika 4.6). Prilikom očitavanja vrijednosti nekog drugog tranzistora koji je u seriji s preprogramiranim tranzistorom uvijek će se čitati „0“, bez obzira na stvarno upisanu vrijednost ciljnog tranzistora, budući da preprogramirani tranzistor stvara prekid u strujnom krugu (slika 4.7).



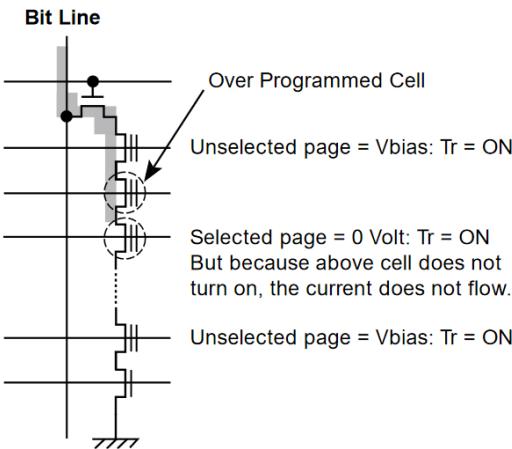
Slika 4.6: Napon praga preprogramiranog tranzistora u odnosu na valjane vrijednosti. Pri vrijednosti upravljačkog napona U_{BIAS} tranzistor bi uvijek trebao biti u stanju vođenja, ali kod pogreške preprogramiranja nije.

Inverzija logičke vrijednosti uslijed pisanja (engl. *write disturb*) je greška kod koje prilikom programiranja ćelije dolazi do nehotičnog programiranja susjednih ćelija, kako ilustrira slika 4.8. Naponski odnosi u susjednim ćelijama mogu biti takvi da se mala količina elektrona ubaci na plutajući *gate* te ponavljanjem ove akcije može doći do dovoljnog pomaka napona praga za promjenu logičke vrijednosti.

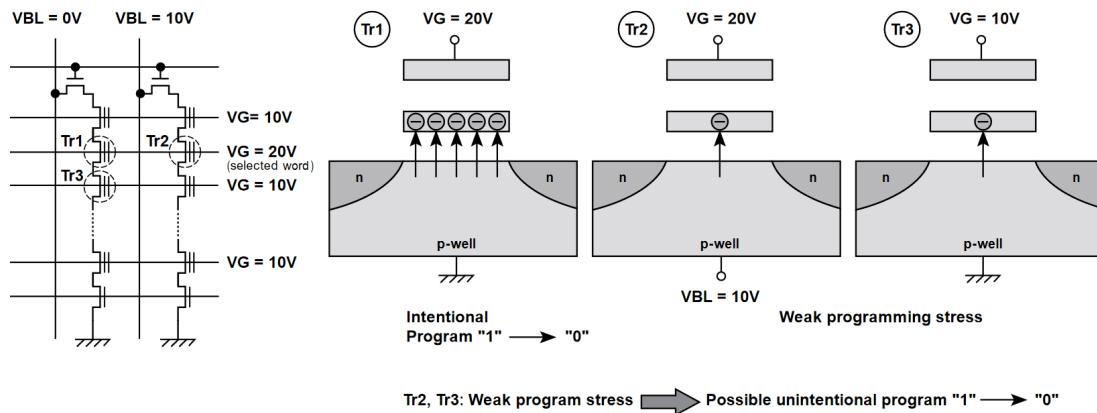
²Dozvoljena je pojava određenog broja neispravnih blokova pri čemu se to neće računati kao greška u izdržljivosti [22].

Inverzija logičke vrijednosti uslijed čitanja (engl. *read disturb*) je greška koja zahvaća ćelije koje se nalaze u istoj *bit* liniji kao i ćelija koja se očitava (slika 4.9). Zbog izloženosti naponu za neselektiranu liniju, dolazi do skakanja manje količine elektrona na plutajući *gate*, pa nakon velikog broja ciklusa čitanja može doći do promjene vrijednosti bita iz „1“ u „0“.

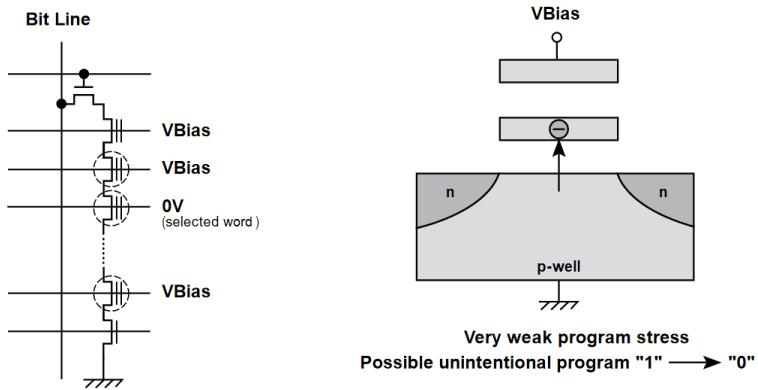
Privremene greške javljaju se s poprilično malenom učestalošću od otprilike 1 naprema 10^{10} programiranih bitova. Za osiguravanje ispravnog rada mogu se koristiti kodovi za ispravljanje pogrešaka (ECC - engl. *Error Correcting Code*).



Slika 4.7: Greška prilikom čitanja *bit* linije zbog preprogramirane ćelije. Preuzeto iz [37].



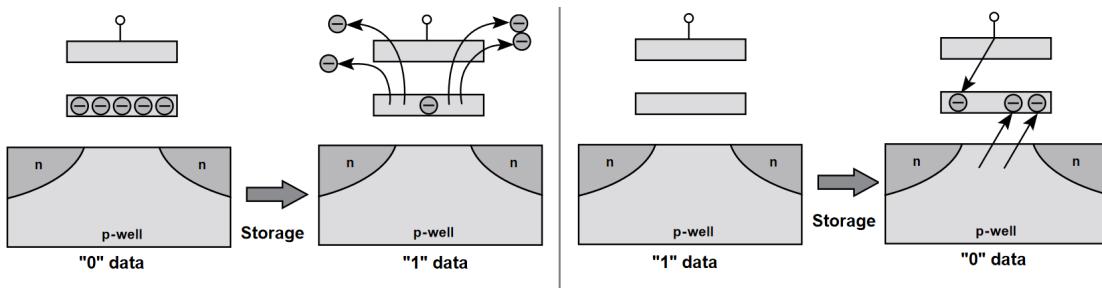
Slika 4.8: Inverzija logičke vrijednosti uslijed pisanja. Supstrat tranzistora nalazi se na potencijalu VBL. VG je potencijal na upravljačkoj elektrodi. Ukoliko se tranzistor želi programirati, postavlja se VG=20V i VBL=0V. Pojedine kombinacije napona mogu rezultirati neželjenim „slabim“ programiranjem. Preuzeto iz [37].



Slika 4.9: Inverzija logičke vrijednosti uslijed čitanja. Do inverzije može doći tek nakon velikog broja čitanja bez da je blok bio izbrisani. Preuzeto iz [37].

4.2.3. Zadržavanje (retencija) podataka

Pojam retencije podataka odnosi se vremenski period tijekom kojeg memorija može zadržati pohranjene podatke, prije nego što se oni nepovratno izgube. Mehanizam gubitka vrijednosti bita u memorijskoj ćeliji prikazan je slikom 4.10: moguće je ili gubitak elektrona „curenjem“ iz plutajuće upravljačke elektrode ili njihovo ubacivanje iz susjednih vodljivih struktura. Pokazuje se da je zadržavanje vrijednosti bita to kraće što je ćelija izložena većem broju ciklusa pisanja i brisanja. Tako primjerice jedan od JEDEC³ standarda propisuje trajanje retencije od minimalno 10 godina nakon izlaganja memorije broju ciklusa pisanja i brisanja u iznosu od 10% od deklariranog (tj. od specificirane izdržljivosti), ali samo godinu dana nakon izlaganja punom broju deklariranih ciklusa pisanja i brisanja [38]. Valja napomenuti da se retencija podrazumijeva uz isključeno napajanje memorije, budući da neki memorijski uređaji imaju implementirano periodičko osvježavanje vrijednosti bitova.

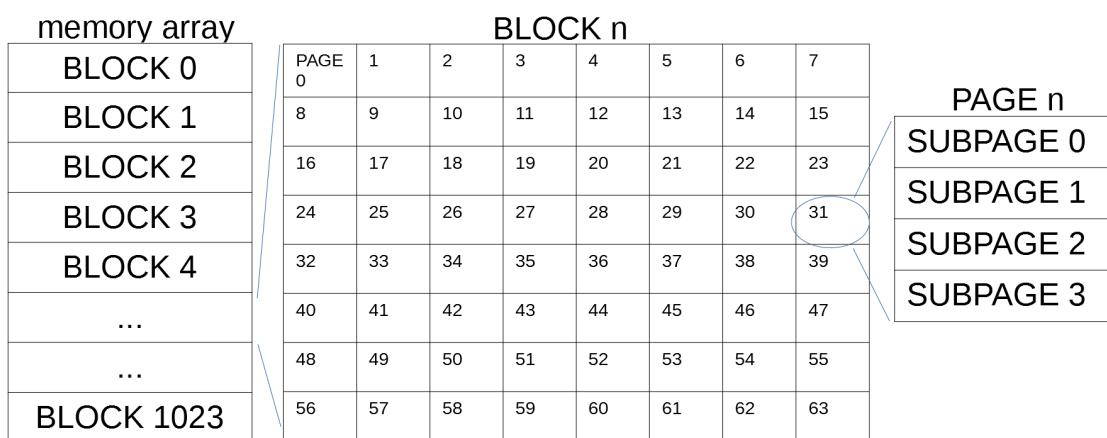


Slika 4.10: Gubitak pohranjene vrijednosti zbog gubitka (lijevo) ili ubacivanja (desno) elektrona. Preuzeto iz [37].

³Joint Electron Device Engineering Council, organizacija koja se bavi razvojem standarda u poluvodičkoj industriji.

4.3. Winbond W25N01GV NAND Flash memorija

S obzirom na zahtjeve unutar misije satelita (pohrana fotografija i podataka sa senzora), odabrana je Flash memorija NAND arhitekture. Radi se o modelu W25N01GVxxIG proizvođača Winbond, ukupnog kapaciteta 128 MB [39]. Memorija može komunicirati s mikrokontrolerom korištenjem *quad*, *dual* ili običnog SPI sučelja pri brzini SPI takta do 104 MHz, što omogućuje najveću teoretsku brzinu prijenosa podataka od 416 Mbit/s. Zbog ograničenja od strane odabranog mikrokontrolera, korištena brzina prijenosa u ovom projektu manja je za red veličine (detaljnije opisano u poglavlju 5). Struktura memorije prikazna je slikom 4.11.

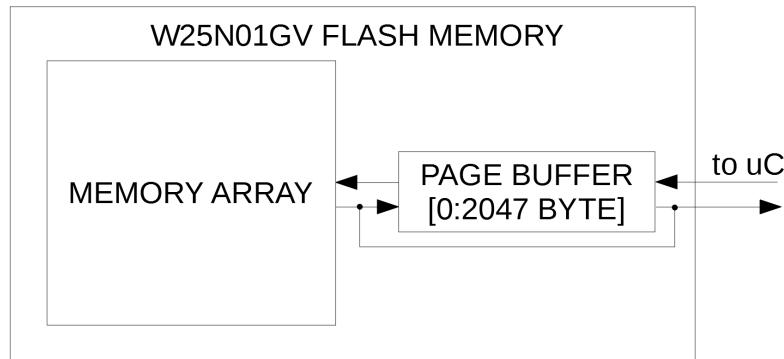


Slika 4.11: Struktura W25N01GV memorije.

Memorija se sastoji od ukupno 1024 bloka, svaki kapaciteta 128 kB. Valja se prisjetiti kako je blok najmanja jedinica koju je moguće izbrisati u Flash memoriji. Drugim riječima, jednom upisane oktete, ma koliko ih malo, nije moguće prepisati dok se ne izbriše cijeli blok. Svaki blok sastoji se od manjih jedinica, tzv. stranica (*pages*), njih ukupno 64. Stranica je veličine 2048 okteta plus dodatnih 64 (tzv. *spare area*). Ukoliko se ne koristi kod za ispravljanje pogrešaka (ECC), na raspolaganju je svih 2112 okteta. Ukoliko se koristi ECC, korisniku je na raspolaganju samo prvih 2048 okteta⁴. Kako je ECC nužan za ispravljanje privremenih pogrešaka, on se u ovom projektu koristi te se od sada nadalje u tekstu stranica smatra velikom točno 2048 B. Čitanje i pisanje u memoriju (programiranje) odvija se putem međuspremnika koji je velik koliko i jedna stranica (slika 4.12). Iz toga slijedi kako je odjedanput moguće uprogramirati najviše 2048 okteta. Čitanje se iznimno može vršiti mimo međuspremnika, pri čemu

⁴Čak i u tom slučaju korisnik može koristiti pojedine dodatne oktete. Za detalje pogledati u [39] i [40]. Zbog izvjesnih komplikacija koje korištenje tih okteta izaziva, ta mogućnost ne koristi se u ovom radu.

memorija na SPI sučelje kontinuirano izbacuje oktete krenuvši od neke lokacije; pri-tom je potrebno uključiti poseban mod rada memorije. Zbog načina na koji se računaju i pohranjuju kodovi za ispravljanje grešaka, najmanja količina podataka koja se može programirati odjedanput je 512 B, odnosno jedna podstranica⁵ [40]. Činjenica da postoji minimalna veličina upisa veća od jednog okteta stvara određene probleme koji su zajedno s načinom njihovog rješavanja opisani u poglavlju 5. Osim navedenog glavnog memorijskog prostora, ova Flash memorija sadrži i deset dodatnih stranica čiji je sadržaj nakon programiranja moguće zaključati, odnosno zaštititi od brisanja.

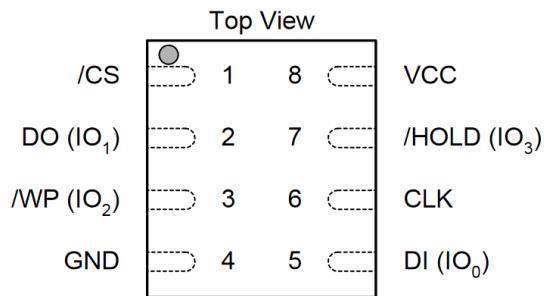


Slika 4.12: Upis i čitanje iz memorije vrši se putem međuspremnika veličine jedne stranice. U posebnom modu rada moguće je direktno čitanje, mimo međuspremnika.

Cjelokupna komunikacija s mikrokontrolerom odvija se preko SPI sučelja, odnosno preko istog se osim podataka prenose i naredbe prema memoriji. Popis svih naredbi i njihov detaljan opis moguće je pronaći u [39]. Tipično se svaka SPI transakcija između memorije i mikrokontrolera sastoji od dvije faze: izdavanja naredbe (mikrokontroler memoriji) i prijenosa podataka (upis ili čitanje, bilo same memorije, bilo statusnih registara).

Pogledom na priključke memorijskog modula (slika 4.13) osim standardnih priključaka za SPI i napajanje uočavaju se dva priključka za koje na prvi pogled nije jasno čemu služe: /HOLD i /WP. Priključak /HOLD koristi se ako se memorija nalazi u *daisy chain* SPI konfiguraciji s drugim uređajima. U tom slučaju, kada je /HOLD linija aktivna svi podaci na podatkovnoj SPI liniji se zanemaruju i samo propuštaju kroz interni posmačni registar. Priključak /WP služi kao sigurnosni mehanizam za zaštitu od nehotičnog upisa u memoriju: ukoliko je /WP aktivan, memorija prelazi u *read only* način rada. /WP linija ne mora se koristiti.

⁵Za ovu jedinku proizvođač u svojoj dokumentaciji spominje pojam *sektor*. Zbog intuitivnosti autor ovog teksta koristi pojam *podstranica* (engl. *subpage*).



Slika 4.13: Nacrt komponente W25N01GV memorije u WSON 8 kućištu. DO označava *Data Out*, a DI *Data In*. Neki od priključaka imaju različitu funkciju s obzirom na to koristi li se standardno, *dual* ili *quad* SPI sučelje. Preuzeto iz [39].

5. Razvijena programska potpora

5.1. Razvojni sustav, korišteni programski paketi i biblioteke

Prilikom razvoja programske podrške korištena je razvojna pločica STM32F4 Discovery koja sadrži STM32F407 mikrokontroler. Pločica se putem USB sučelja povezuje s računalom i time omogućuje učitavanje programskog koda na memoriju mikrokontrolera, kao i otklanjanje pogrešaka (*debugging*) putem SWD sučelja. Za povezivanje mikrokontrolera s Flash memorijom korištena je Mikroe Flash 5 Click pločica koja na sebi ima Windbond W25N01GV čip. Za ispravno funkcioniranje memorije bilo je potrebno spojiti vanjski *pull-up* otpornik između /CS linije i napajanja memorije, zbog zahtjeva da /CS linija prati napon napajanja prilikom uključenja istog. Flash memorija i kamera (koja troši struju iznosa do 600 mA) napajani su izvorom napajanja „YwRobot“, reguliranim naponom iznosa 3.3 V.

Programska potpora razvijena je u programskom jeziku C. Pri razvoju je korišteno integrirano razvojno okruženje STM32CubeIDE, uz uporabu GCC prevodioca (kao dio *GNU Arm Embedded Toolchain* paketa). Navedeni IDE omogućuje, između ostalog, jednostavno generiranje *startup* koda i jednostavnu konfiguraciju RCC (*Reset and Clock Control*) sklopljiva putem grafičkog korisničkog sučelja. Putem korisničkog sučelja moguće je i automatski generirati kod za inicijalizaciju željenih perifernih jedinica, no ova mogućnost, osim u jednom navratu, nije korištena budući da generirani kod ne slijedi uvijek proceduru inicijalizacije opisanu u korisničkom priručniku mikrokontrolera. Također, korisnik svoj vlastiti kod mora upisivati na točno predviđena mjesta, inače će ga alat prilikom novog ciklusa automatske generacije izbrisati, što se pokazuje veoma frustrirajućim za korisnika.

Proizvođači mikrokontrolera sa svojim proizvodima tipično isporučuju programske biblioteke koje olakšavaju rad s perifernim jedinicama mikrokontrolera. STMicroelectronics isporučuje dva paketa biblioteka: HAL (*Hardware Abstraction Layer*)

i LL (*Low Layer*) biblioteke [33]. HAL biblioteke nude visoku razinu apstrakcije, te su zbog jednostavnosti korištenja iste na prvi pogled veoma primamljive. Primjerice, neka korisnik želi poslati određeni broj okteta preko SPI sučelja. Tada samo treba pozvati HAL funkciju kojoj će predati pokazivač na memoriju lokaciju podataka i broj podataka koji želi poslati. No, korištenje HAL funkcija sa sobom nosi niz nedostataka. Prvo, korištenje funkcije po principu „crne kutije“ znači da korisnik ne razumije što ta funkcija u sebi radi. U slučaju bilo kakvog problema s periferijom, korisnik će ga teže otkloniti. Čak i kada korisnik pokuša uvidjeti u izvorni kod funkcije, vjerojatno će mu djelovati poprilično nerazumljivim pa će ionako morati posegnuti za korisničkim priručnikom mikrokontrolera. Razumijevanju sadržaja funkcije ne pomaže ni činjenica da su HAL funkcije često ispunjene provjerama vrijednosti i modifikacijama internih HAL struktura. Navedene HAL strukture dodatno zauzimaju radnu memoriju i procesorske resurse (npr. HAL *timeout timer*), čime se ne osiguravaju optimalne performanse. HAL funkcije najkorisnije su kada se nešto želi brzo osposobiti, a o optimizaciji se može brinuti kasnije. LL funkcije, pak, predstavljaju način pristupa registrima periferije koji je gotovo u potpunosti izravan. Primjera radi, dano je tijelo funkcije kojom se osposobljuje prekid na svaki *update event* vremenskog sklopa TIM:

```
static inline void LL_TIM_EnableIT_UPDATE(TIM_TypeDef *TIMx) {
    TIMx->DIER |= TIM_DIER_UIE;
}
```

Možda najveća prednost LL funkcija je u tome što je kod čitljiv bez dodatnih komentara. Također, određene LL funkcije za inicijalizaciju imaju određenu „pamet“, primjerice kod inicijalizacije UART periferije korisnik samo specificira željenu brzinu prijenosa (engl. *baud rate*), a funkcija će proračunati i podesiti vrijednost svih registara koji su relevantni za određivanje brzine prijenosa. No, LL funkcije također imaju bitan nedostatak. Naime, *inline* oznaka je preporuka, ali ne i garancija da će se funkcija direktno ugraditi u kod [6]. Pri isključenoj optimizaciji i pod nekim uvjetima pri ostalim razinama optimizacije ove *inline* funkcije će se pozivati, što povećava vrijeme njihovog izvođenja.

U ovom radu koristi se kombinacija LL funkcija i direktnog pristupa registrima. LL funkcije se koriste u općenitom slučaju, a direktni pristup registrima u segmentima koda koji su vremenski kritični. HAL funkcije se ne koriste.

5.2. Kamera

Za rad s kamerom razvijene su sljedeće korisničke funkcije (isječak 5.1).

Isječak 5.1: Korisničke funkcije za Arducam 5MP Mini Plus.

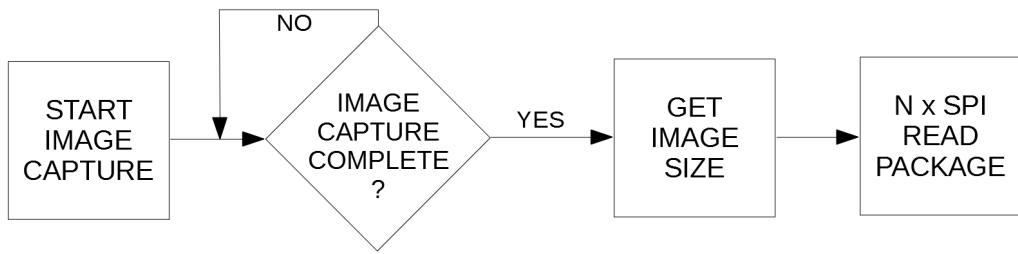
```
void ACAM_init(void);
uint8_t ACAM_test_comms(void);
void ACAM_reset(void);
void ACAM_select_JPEG(void);
void ACAM_select_RAW(uint8_t resolution);
void ACAM_exp_gain_auto(void);
void ACAM_exp_gain_manual(void);
void ACAM_set_exposure(uint16_t nr_lines, uint8_t nr_lines_frac);
void ACAM_set_gain(uint8_t gain);
void ACAM_start_capture(void);
uint8_t ACAM_is_cap_complete(void);
uint32_t ACAM_get_image_size();
void ACAM_spi_read_package(uint8_t* buff, uint16_t size);
```

Prva funkcija služi inicijalizaciji svih potrebnih perifernih jedinica na mikrokontroleru za rad s kamerom (GPIO priključci, SPI i I2C sučelja, DMA jedinica, vremenski sklopovi). Funkcija `ACAM_test_comms()` testira ispravnost SPI i I2C komunikacije čitanjem *ID* registara na kameri za navedena sučelja. Funkcija `ACAM_reset()` resetira senzor slike i CPLD na kameri. Odabir formata slike vrši se funkcijama `ACAM_select_JPEG()` i `ACAM_select_RAW()`. Prednost JPEG formata je u manjoj veličini slike (par stotina kB), ali nije namijenjen mjeriteljskim aplikacijama zbog kompresije i uključenih funkcionalnosti obrade slike (primjerice automatski balans bijele boje). Slike u RAW formatu imaju isključene sve mogućnosti obrade slike koje mogu interferirati s mjeranjima, kako je diskutirano u poglavlju 3. Iako senzor podržava kvantizaciju do 10 bita, čini se kako Arducam kamera interno ima mogućnost pohrane najviše 8 bita po pikselu RAW slike. Operacija raspreplitanja mozaika ne provodi se na kameri, pa RAW slika pri punoj rezoluciji od 2592x1944 točaka zauzima oko 5.04 MB¹. Način upravljanja ekspozicijom odabire se funkcijama `ACAM_exp_gain_auto()` i `ACAM_exp_gain_manual()`. Ukoliko se odabere ručno upravljanje ekspozicijom, potrebno je postaviti parametre vremena ekspozicije i pojačanja pojačala² funkcijama `ACAM_set_exposure()` i `ACAM_set_gain()`.

¹Funkcija `ACAM_select_RAW()` omogućuje odabir jedne od četiri unaprijed odabrane rezolucije. No uporaba svih ostalih rezolucija osim maksimalne predviđena je isključivo u svrhu otklanjanja pogrešaka (*debugging*), radi kraćeg vremena potrebnog za prijenos manje slike na računalo. Naime, manje slike ne predstavljaju skaliranu verziju slike u punoj rezoluciji, već odrezak (*crop*).

²Pojačalo označeno kao AMP unutar *image sensor core* na slici 3.11. Vrijednosti pojačanja su od 1 do 32. Preferirani način upravljanja svjetlinom i dalje je promjena trajanja ekspozicije, budući da pojačalo pojačava i šum.

Vrijeme ekspozicije podešava se po cijelobrojnom i frakcionalnom dijelu trajanja vremena linije (t_{LINE}).³ Korištenje preostalih funkcija ilustrira dijagram toka na slici 5.1. Nakon izdavanja komande za početak slikanja pomoću `ACAM_start_capture()` valja pričekati dok kamera ne javi da je fotografiranje dovršeno (provjerom povratne vrijednosti funkcije `ACAM_is_cap_complete()`). Korisnik će sliku tada htjeti, putem međuspremnika određene veličine u radnoj memoriji, prebaciti negdje drugdje, primjerice na vanjsku memoriju. Stoga je potrebno odrediti veličinu slike u oktetima (veličina JPEG slika je varijabilna), pomoću funkcije `ACAM_get_image_size()`, kako bi bilo poznato koliko ciklusa korištenja RAM međuspremnika je potrebno (označeno s N na slici 5.1). Za prijenos podataka slike s kamere u radnu memoriju mikrokontrolera koristi se funkcija `ACAM_spi_read_package()`, uz specifikaciju lokacije i veličine međuspremnika u memoriji.



Slika 5.1: Sekvenca fotografiranja i prijenosa slike. Za detalje pogledati u tekstu.

Osim navedenih korisničkih funkcija, razvijene su i funkcije za komunikaciju s kamerom putem I2C i SPI sučelja, točnije za čitanje i postavljanje vrijednosti registara kamere (isječak 5.2).

Isječak 5.2: SPI i I2C funkcije za komunikaciju s Arducam 5MP Mini Plus kamerom.

```

uint8_t ACAM_spi_read(uint8_t reg);
void ACAM_spi_write(uint8_t reg, uint8_t val);

uint8_t ACAM_i2c_read(uint16_t reg);
void ACAM_i2c_write(uint16_t reg, uint8_t command);
void ACAM_i2c_write_seq(const struct acam_i2c_reg command_seq[]);
  
```

Za rad s I2C periferijom koriste se prekidne rutine, a za rad sa SPI periferijom prozivanje statusnih zastavica (*status flag polling*). Iznimno, `ACAM_spi_read_package()` (opisana kod korisničkih funkcija) koristi DMA prijenos.

³Podešavanje trajanje ekspozicije u milisekundama nije moguće zbog toga što proizvođač nije specificirao sve parametre potrebne za izračun.

Tijekom rada na razvoju programske podrške za kameru pokazalo se kako dokumentacija kamere i senzora ima značajne nedostatke. U dokumentaciji senzora [27] puno je toga nedorečeno (primjerice izostanak opisa funkcionalnosti pojedinih regis-tara) te je česta napomena kako je za detalje potrebno „obratiti se Omnivision FAE-u (*Field Application Engineeru*)“.⁴ No, aplikacijski inženjer nije dostupan ukoliko se ne napravi veća narudžba senzora i ne potpiše ugovor o povjerljivosti podataka. Stoga, jasno je kako namjera Omnivisiona nije da krajnji korisnik ulazi u konfiguraciju sen-zora, već bi to trebali raditi OEM (*Original Equipment Manufacturer*) proizvođači, poput Arducama. Arducam na korištenje zaista daje kod koji konfigurira preko 50 registara odjedanput bez ikakvih komentara [28], no problemi nastaju kada se nave-denii kod želi modificirati, budući da u njemu postoje registri čija funkcionalnost nije naznačena u dostupnoj dokumentaciji. Jedan od primjera takve poteškoće je naizgled jednostavna promjena rezolucije slike, za što je u stvarnosti potrebno modificirati ve-liki broj različitih parametara. Drugi problem je nekvalitetna dokumentacija sa strane Arducama [12], kojoj nedostaju, između ostalog, vremenski dijagrami za SPI komu-nikaciju; štoviše, bilo kakvi podaci o vremenskim ograničenjima (vrijeme potrebno za reset, vrijeme potrebno za stabilizaciju postavljenih parametara). Do željenih vremen-skih parametara bilo je potrebno doći ili uvidom u umetnuta čekanja u Arducamovim oglednim primjerima [1], ili metodom pokušaja i pogreške.

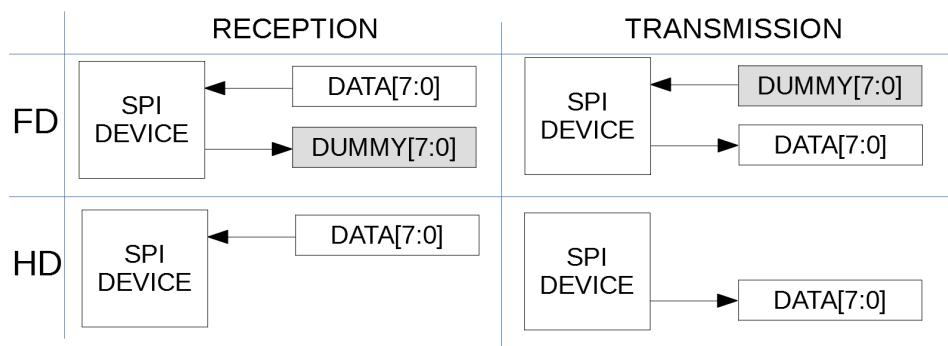
5.3. Flash memorija

5.3.1. SPI sučelje između Flash memorije i mikrokontrolera

U poglavlju 4 već je spomenuto kako odabrani mikrokontroler ne može iskoristiti punu brzinu prijenosa koju nudi W25N01GV Flash memorija. Prvi razlog je taj što STM32F407 mikrokontroler ne podržava niti *quad* niti *dual* SPI sučelje [31]. Drugo, umjesto frekvencije SPI takta od 104 MHz koliko podržava memorija, mikrokontroler može ostvariti najviše 42 MHz. Ovo je i dalje poprilično visokih 42 Mbit/s maksimalne teoretske brzine prijenosa (valja se prisjetiti kako X-band odašiljač specificira maksimalnu brzinu prijenosa od 16 Mbit/s). Prilikom ostvarenja programske podrške za SPI sučelje treba uzeti u obzir da sama jezgra mikrokontrolera radi na 168 MHz što nije značajno više od brzine SPI takta. Posljedično je utvrđeno kako je optimalno rješenje koristiti SPI periferiju u *full-duplex* načinu rada uz prozivanje statusnih zas-

⁴Dokumentaciju na raspolaganje daje Arducam, a na dokumentaciji je vidljiv pečat „*Confidential for Omnivision Technologies Internal Only*“.

tavica. Slijedi detaljno pojašnjenje ovih pojmoveva. Prilikom komunikacije putem SPI sučelja, mikrokontroler je upravljač (*master*), odnosno onaj koji generira takt, a memorija izvršioc (*slave*). Već je spomenuto kako se komunikacija između mikrokontrolera i memorije odvija u dvije faze. Primjera radi, neka mikrokontroler pokušava pročitati 1 kB podataka s memorije. U prvoj fazi, mikrokontroler šalje zahtjev za čitanjem memoriji. U drugoj fazi, memorija šalje podatke mikrokontroleru. Ukoliko SPI periferija na mikrokontroleru radi u *full-duplex* načinu rada, za svaki odaslanji oktet jedan se mora primiti. To znači da u prvoj fazi mikrokontroler, osim što šalje oktete koji sadrže naredbu, mora i primati *dummy* oktete (slika 5.2). U drugoj fazi pak *dummy* okteti se šalju. Alternativa *full-duplex* načinu rada je *half-duplex* način rada, gdje se primaju ili šalju okteti samo u smjeru komunikacije koji je relevantan, dakle nema *dummy* okteta.

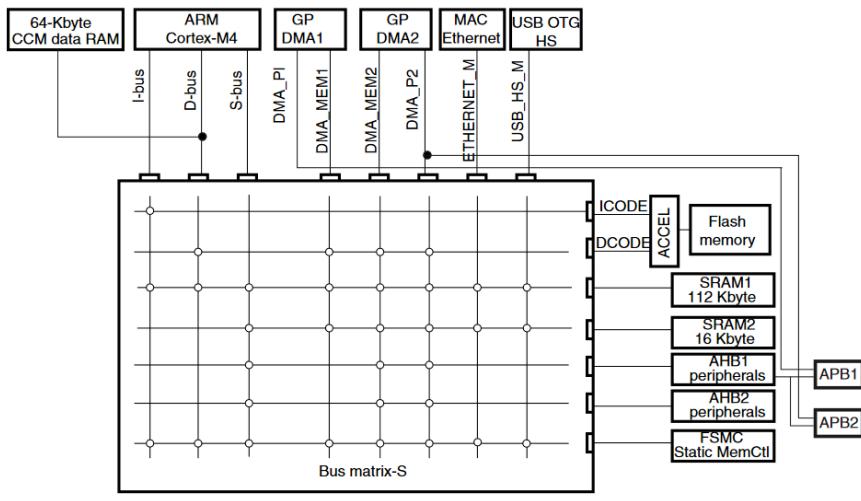


Slika 5.2: Primanje (lijevo) i odašiljanje (desno) okteta kod SPI uređaja, u *full-duplex* načinu rada (gore) i *half-duplex* načinu rada (dolje).

Iako je logično da bi se za komunikaciju s memorijom trebao koristiti *half-duplex* mod, to nažalost nije moguće. Naime, prilikom rekonfiguracije smjera prijenosa na konkretnom mikrokontroleru, potrebno je isključiti SPI periferiju. Tada svi GPIO pinovi odlaze u stanje visoke impedancije [34], memorija na CLK liniji detektira nedozvoljeno stanje te se komunikacija zablokira. Moguće je pritezni otpornikom definirati potencijal CLK linije dok je periferni kontroler isključen, no ovo radi samo za manje frekvencije (do 1 MHz), budući da pritezni otpornik kvari vremensku konstantu signala.

Drugo pitanje koje se postavlja je kao detektirati RXNE (*Receiver Buffer Not Empty*) i TXE (*Transmitter Buffer Empty*) događaje na perifernom kontroleru. Na svaki RXNE događaj potrebno je prebaciti oktet iz RX međuspremnika u memoriju (ili ako se radi o *dummy* oktetu samo pročitati RX međuspremnik), a na svaki TXE događaj potrebno je postaviti oktet iz memorije (ili *dummy* oktet) u TX međuspremnik. Jedna mogućnost, ujedno i ona koja je korištena u ovom projektu, je ispitivanje vrijednosti

RXNE i TXE zastavica u petlji iz glavnog programa (*status flag polling*), te prebacivanje okteta nakon što se detektira događaj. Druga mogućnost je izazivanje prekida na RXNE/TXE događaj i prebacivanje oktetra unutar prekidne rutine. No, kako procesor radi na 168, a SPI periferija na 42 MHz, kod SPI prijenosa će se na svaka 32 strojna ciklusa procesora pojaviti dva prekida (TXE i RXNE događaj). Latencija ulaska u prekidnu rutinu minimalno je 12 ciklusa, potrebno je još vrijeme za obradu događaja u prekidnoj rutini (prebacivanje oktetra) i povratak iz prekidne rutine. Jasno je kako procesor osim obrade prekida ne bi mogao obavljati nikakav drugi koristan posao, što više veoma je izvjesno da bi se brzina SPI komunikacije morala usporiti zbog nemogućnosti obrade prekida na vrijeme. Treća mogućnost, koja se doima najlegantnijom, je korištenje DMA kontrolera za autonomno prebacivanje oktetova između RX i TX međuspremnika i memorije. No, u nekim veoma rijetkim situacijama može se dogoditi da DMA kontroler na prospojnoj matrici predugo čeka na dostupnost AHB sabirnice (slika 5.3), toliko dugo da dođe do pojave *read overrun* okolnosti, odnosno da se neki oktet ne stigne pročitati iz RX međuspremnika prije nego li ga prebriše sljedeći [34].⁵ Ovo bi se moglo pokazati kobnim u slučaju da do greške dođe prilikom prijenosa podataka koji su bitni za funkcioniranje datotečnog sustava. Iako je ovo stanje greške moguće detektirati i u tom slučaju, primjerice, pokrenuti ponovljeni prijenos, u svrhu izbjegavanja pretjerane složenosti odlučeno je ići na pristup s prozivanjem statusnih zastavica.



Slika 5.3: Prospojna matrica AHB sabirnice kod STM32F407 mikrokontrolera. Pristup sa- birnici se dodjeljuje *round robin* algoritmom. U pojedinim slučajevima DMA kontroler može čekati dulje od 32 strojna ciklusa na pristup. Preuzeto iz [31].

⁵Periferni kontroleri na recentnijim STM mikrokontrolerima imaju mogućnost pauziranja prijenosa u slučaju opasnosti od *read overrun* okolnosti.

5.3.2. Upravljački program (*driver*) za memoriju

Za rad s memorijom razvijen je veći broj korisničkih funkcija. Većina ih se odnosi na pristup i modifikaciju vrijednosti pojedinih bita unutar statusnih registara memorije. Primjerice, funkcija:

```
uint8_t W25N_check_ECC_on(void);
```

provjerit će je li aktivirano korištenje kodova za ispravljanje pogrešaka (ECC). Sada će biti nabrojane i ukratko opisane najvažnije funkcije za rad s memorijom (isječak 5.3).

Isječak 5.3: Važnije funkcije za rad s W25N01GV Flash memorijom.

```
uint8_t W25N_init(void);
uint8_t W25N_block_erase(uint16_t block_nr);
uint8_t W25N_page_data_read(uint16_t page_addr);
uint8_t W25N_read_data(uint16_t col_addr,
                      uint8_t* data, uint16_t nr_bytes);
uint8_t W25N_prog_data_load(uint16_t col_addr,
                            uint8_t* data, uint16_t nr_bytes);
uint8_t W25N_rand_prog_data_load(uint16_t col_addr,
                                  uint8_t* data, uint16_t nr_bytes);
uint8_t W25N_prog_execute(uint16_t page_addr);
```

Funkcija `W25N_init()` izvršit će inicijalizaciju svih perifernih jedinica na mikrokontroleru potrebnih za rad s memorijom. Također, provjerit će je li komunikacija preko SPI sučelja uspostavljena (u slučaju da nije, preko povratne vrijednosti javit će grešku) i isključiti zaštitu od programiranja i brisanja koja je automatski aktivna po uključenju napajanja memorije. Funkcijom `W25N_block_erase()` izvodi se brijanje bloka. Nakon što se odabrana stranica prebaci u međuspremnik Flash memorije pomoću funkcije `W25N_page_data_read()`, funkcijom `W25N_read_data()` moguće je pročitati željeni niz okteta. Adresa, odnosno redni broj prvog okteta za čitanje zadaje se putem parametra `col_addr`. Upis u međuspremnik izvodi se pomoću funkcija `W25N_prog_data_load()` i `W25N_rand_prog_data_load()`. U slučaju prve funkcije, svi oktetni u međuspremniku koji nisu zadani bit će postavljeni u FFh (vrijednost u izbrisanim stanju), u slučaju druge funkcije okteti koji nisu zadani zadržavaju prethodnu vrijednost. Druga funkcija mora se koristiti prilikom višestrukog programiranja iste stranice. Konačno, upis iz međuspremnika u stranicu u memoriji vrši se funkcijom `W25N_prog_execute()`. Sve ove funkcije, izuzev `W25N_init()`, prije izvršavanja provjerit će nalazi li se memorija u *busy* (zaposlenom) stanju i to javiti putem povratne vrijednosti. Funkcije je moguće pozivati u petlji dok se ne ostvari uvjet pristupa (izlazak memorije iz *busy* stanja).

Implementirane su i funkcije za upravljanje neispravnim blokovima (isječak 5.4).

Isječak 5.4: Funkcije za upravljanje neispravnim blokovima.

```
void W25N_scan_factory_BB(uint32_t* BBBT);  
  
void W25N_bb_manage(uint16_t lba, uint16_t pba);  
void W25N_read_bbm_lut(uint8_t* container);
```

Funkcija `W25N_scan_factory_BB()` će skenirati sva 1024 bloka i u tablici bitova, danoj preko pokazivača `BBBT`, označiti one koji su tvornički označeni kao neispravni. Valja imati na umu da se tvorničke oznake neispravnog bloka gube brisanjem bloka. Funkcija `W25N_bb_manage()` omogućuje specificiranje zamjenskog bloka (adresa `pba`) za blok koji je uočen kao neispravan (adresa `lba`). Pri tome se koristi tvornička mogućnost memorije da u prozivnu tablicu (*LUT*) pohrani do 20 parova adresa neispravnih i zamjenskih blokova. Prozivnu tablicu moguće je iščitati funkcijom `W25N_read_bbm_lut()`. No, dokumentacija memorije [39] nedorečena je glede funkcije ove prozivne tablice. Nije jasno služi li ta tablica samo kao registar parova adresa koji se stavlja na raspolaganje korisniku, ili par adresa jednom označen u tablici rezultira automatskom supstitucijom adrese, interno u memoriji, prilikom adresiranja neispravnog bloka. Zbog te nedorečenosti, funkcije koje koriste prozivnu tablicu neispravnih blokova nisu korištene u ovom radu.

5.3.3. Datotečni sustav

S obzirom na namjenu, na datotečni sustav dane su sljedeće specifikacije:

- (s.1) Mora se omogućiti pohrana podataka prikupljenih sa senzora i slika u datoteke. Veličina slike je par stotina kB (JPEG format) ili 5 MB (RAW format), dok će podaci sa senzora zauzimati do par desetaka kB.
- (s.2) Datoteke ostaju u memoriji sve do slanja na Zemlju. Nakon toga se u pravilu brišu.
- (s.3) Sve datoteke ne moraju biti poslane u prvom navratu komunikacije prema Zemlji; nove datoteke mogu nastati između perioda komunikacije.
- (s.4) Naredbe za fotografiranje/akviziciju i slanje na odašiljač izdaje upravljačko računalo.
- (s.5) Datotečni sustav mora zadržati konzistenciju u slučaju iznenadnog gubitka napajanja PDH računala.

(s.6) Memorija mora ostati upotrebljiva tijekom trogodišnjeg trajanja misije.

S obzirom na vrstu i model memorije koja se koristi za pohranu (NAND Flash memorija, model W25N01GV), postoje sljedeća ograničenja:

- (o.1)** Prije prepisivanja vrijednosti okteta, potrebno je izbrisati cijeli blok.
- (o.2)** Programiranje bloka izvodi se po stranicama. Stranice se unutar bloka moraju programirati slijedno.
- (o.3)** Najmanja jedinica upisa je podstranica (veličine 512 B).
- (o.4)** Specificirana izdržljivost bloka memorije iznosi 100 000 W/E ciklusa.
- (o.5)** Ograničeni broj blokova može zakazati prije dosezanja specificirane izdržljivosti.

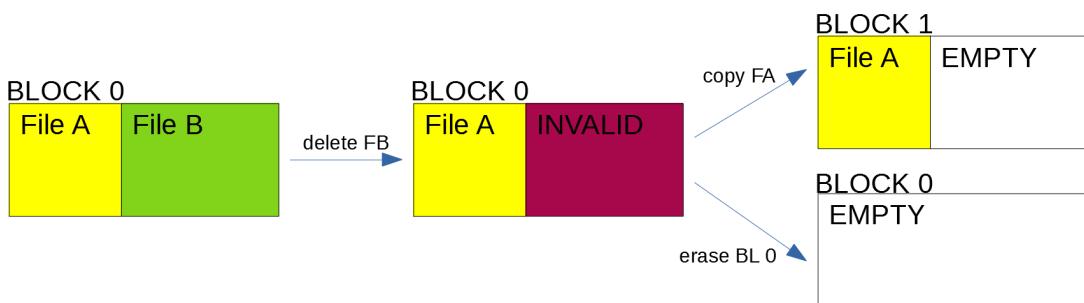
Na temelju specifikacija i ograničenja, donesene su sljedeće odluke o dizajnu:

- Za alokaciju memorije koristit će se pristup temeljen na FAT tablici (*File Allocation Table*).⁶ (**s.2, s.3**)
- Nema potrebe za mogućnošću modifikacije sadržaja datoteke. Nadopunjavanje sadržaja datoteke (*append*) treba omogućiti. (**s.1, o.1, o.2**)
- Nije potrebna podrška za rad s direktorijima, niti napredne mogućnosti imenovanja datoteka. (**s.4**)
- Koristit će se algoritam dinamičkog ujednačenja trošenja blokova (*Dynamic Wear Leveling*). (**s.6, o.4, s.3**)
- Koristit će se sustav upravljanja neispravnim blokovima (*Bad Block Management*). (**o.5**)
- Koristit će se pristup za osiguravanje konzistentnosti u slučaju pada sustava (*Journaling*). (**s.5**)
- Dvije datoteke ne mogu dijeliti isti blok. Posljedično, maksimalni broj datoteka u sustavu sa svim ispravnim blokovima je 1024. (**o.1, s.2**)
- Datoteka se može čitati samo slijedno. Proizvoljno pozicioniranje pokazivača unutar datoteke nije moguće. (**o.3**)

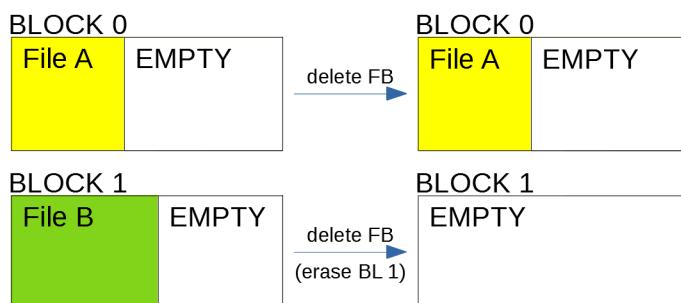
Sada će detaljnije biti pojašnjeni razlozi za posljednje dvije odluke o dizajnu. Odluka da dvije datoteke ne mogu dijeliti isti blok donesena je kako bi se izbjegle komplikacije vezane za tzv. skupljanje otpada (*garbage collection*). To se može razmotriti

⁶U suprotnosti s *log-structured* datotečnim sustavima.

na primjeru koji prikazuje slika 5.4. Ukoliko se želi izbrisati jedna od dviju datoteka na nekom bloku (datoteka B), to se može napraviti tako da se prostor na bloku koji je zauzimala ta datoteka proglaši nevažećim. No, taj prostor sada je neiskoristiv; ako se želi vratiti natrag u uporabu, datoteka A najprije se mora prekopirati na slobodan prostor na nekom drugom bloku, a zatim je potrebno izbrisati blok na kojem su se izvorno nalazile datoteke A i B. Ako pak određeni blok može pripadati samo jednoj datoteci, brisanje datoteke direktno je vezano uz brisanje blokova; nema problema sa skupljanjem otpada (slika 5.5). Osim jednostavnosti, ovaj pristup ima prednost i u manjem trošku procesorskog vremena za rad datotečnog sustava. Nedostatak je povećana interna fragmentacija blokova (neiskoristivi prostori na kraju bloka ako ga datoteka ne zauzima u cijelosti).



Slika 5.4: Skupljanje otpada (*garbage collection*) kod datotečnog sustava za Flash memoriju.



Slika 5.5: Brisanje datoteke kod datotečnog sustava koji ne dozvoljava dijeljenje blokova među datotekama.

Sada će se pojasniti i posljedice ograničenja o minimalnom upisu po podstranici. Što ako bi korisnik u datoteku htio zapisati manje od 512 B? Naravno, moguće je zapisati korisničke oktete, a ostatak, do kraja podstranice, napuniti FFh vrijednostima. Tada je na početak svake podstranice potrebno zapisati njezinu stvarnu veličinu, kako to pokazuje slika 5.6. Ukoliko bi se htjela pročitati proizvoljna lokacija unutar datoteke,

tada bi bilo potrebno pročitati sve indikatore veličine podstranice do te lokacije. Nаравно, ovo je sporo, budуći da zahtjeva niz čitanja iz memorije. Pohrana indikatora veličine podstranice u zasebnu strukturu bila bi nepraktična zbog varijabilne veličine datoteke. S obzirom način upotrebe unutar misije satelita (slanje cjelovitih datoteka na odašiljač), zapravo i nema potrebe za čitanjem priozvoljnih lokacija unutar datoteke, pa zbog svega navedenog ova mogućnost nije implementirana.

col. addr:	00	01	02	03	04	05	06	...	511
subp. 0	4		ABh	1Eh	27h	5Dh	FFh	FFh	FFh
subp. 1	2		32h	2Ah	FFh	FFh	FFh	FFh	FFh
subp. 2	FFh								
subp. 3	FFh								

Slika 5.6: Prikaz stranice kod koje je korisnik najprije pohranio četiri okteta, a zatim još dva. Prva dva okteta unutar podstranice koriste se za pohranu indikatora veličine podstranice. Okteti označeni s FFh unutar nulte i prve podstranice više se ne mogu koristiti za upis, budуći da je ECC kod za te podstranice već izračunat i pohranjen.

Ovakav način upisa podataka po podstranicama za neizbjježnu posljedicu ima i gubitak memorijskog prostora. U idealnom slučaju, količina prostora raspoloživog za pohranu smanjuje se tek neznatno, za dva okteta po podstranici, koji su potrebni za pohranu indikatora veličine podstranice. Upisom broja okteta koji nije višekratnik od 510 dolazi do pojave interne fragmentacije podstranice i gubitka iskoristivog prostora.

FAT tablica, datoteke i opisnici datoteka

FAT tablica (*File Allocation Table*) opisuje koji blokovi pripadaju kojoj datoteci [21]. Funkcionalnost je ilustrirana na primjeru sa slike 5.7: neka su u sustavu prisutne dvije datoteke, datoteka A veličine 3 bloka (označena žutom bojom) i datoteka B veličine 6 blokova (označena zelenom bojom). Kod FAT tablice, svaki blok pokazuje na sljedeći blok te datoteke, sve do posljednjeg bloka koji ima oznaku kraja datoteke (EOF). Valja uočiti kako datoteke ne moraju zauzimati kontinuirani memorijski prostor (primjerice datoteka B s primjera), što omogućuje jednostavnu ekspanziju datoteke bez potrebe za predalociranjem prostora prilikom njezinog stvaranja. U ovdje implementiranom rješenju, prilikom alokacije bloka na FAT tablici traži se prvi po redu slobodan blok.

Svaka datoteka ima svoj jedinstveni identifikacijski broj (interne oznake `file_id`, vrijednosti između 0 i 1023). Taj identifikacijski broj ujedno i odgovara broju prvog

bloka datoteke u FAT tablici. Također, pomoću navedenog broja pristupa se i tablici opisnika datoteke (*File Descriptor Table*, FDT). Među najvažnijim elementima opisnika datoteke su veličina datoteke u oktetima i ukupni broj podstranica koji datoteka zauzima.⁷

FILE A FILE B

BLOCK nr:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	...
FAT entry:	FREE	FREE	3	4	9	FREE	7	8	EOF	10	11	EOF	FREE	FREE	...

Slika 5.7: Primjer FAT tablice s dvije pohranjene datoteke. Za opis pogledati u tekstu.

Flash Translation Layer

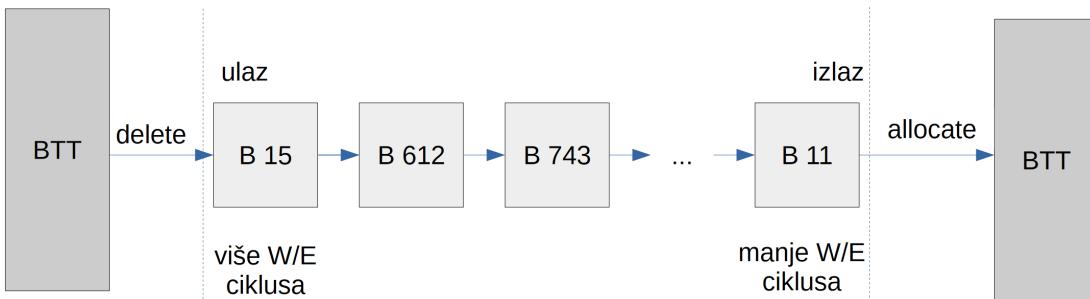
Zadaća sloja za prevođenje kod Flash memorije (*Flash Translation Layer*, FTL) je razdvajanje logičkih i fizičkih blokova [23]. Potreba za FTL-om bit će objašnjena na primjeru. Neka je dan sustav koji u svakom trenutku ima najviše jednu datoteku pohranjenu, te neka je ta datoteka uvijek veličine jednog bloka. Nadalje, neka se ta datoteka periodički briše i ponovno stvara. Prilikom alokacije bloka na FAT tablici, uvijek će se izabrati blok rednog broja 0. Taj blok će na fizičkoj memoriji biti izložen trošenju, dok će ostali blokovi biti netaknuti. Razdvajanjem logičkih i fizičkih blokova, prilikom svake alokacije FAT tablica će i dalje koristiti blok 0, ali se na fizičkoj memoriji svaki put može izabrati neki drugi blok. U tu svrhu treba koristiti tzv. tablicu prevođenja blokova (*Block Translation Table*, BTT) koja povezuje logičke i fizičke adrese blokova (slika 5.8).

LBA	PBA
0	15
1	298
2	FREE
3	77
4	FREE
5	450
...	...

Slika 5.8: Tablica prevođenja blokova (BTT) svakom logičkom bloku (LBA - *Logical Block Address*) koji se koristi dodjeljuje fizički blok s memorije (PBA - *Physical Block Address*).

⁷Primijetiti kako ova dva pokazatelja mogu, ali ne moraju biti ekvivalentna.

Kako odabrati odgovarajući fizički blok? Rješenje koje je korišteno u ovom radu oslanja se na tzv. red slobodnih blokova (*Free Physical Block Queue*). Alokacijom bloka, isti se skida s izlaza reda. Brisanjem datoteke, pripadajući blokovi postavljaju se na ulaz reda (slika 5.9). Iz toga proizlazi kako će blokovi bliže izlazu imati manji broj ciklusa pisanja i brisanja, dok će blokovi bliže ulazu imati veći broj ciklusa pisanja i brisanja. Ovo vrijedi uz pretpostavku da se datoteke koje nastanu, nakon određenog vremena i izbrišu, te da se to radi podjednakom frekvencijom za sve datoteke. Navedenim pristupom izbjegava se pojačano trošenje pojedinih blokova, odnosno ostvaruje se *wear leveling*, dinamičkog tipa. Kada bi u sustavu neka datoteka u odnosu na druge datoteke bila prisutna dugo vremena, blokovi koji je sačinjavaju bili bi izloženi manjem trošenju, na štetu većeg trošenja ostalih blokova. No, kako se s obzirom na specifikacije s početka ovog odjeljka ne predviđa postojanje takvih datoteka tijekom misije satelita, ovaj nedostatak u načinu rada sustava u stvarnosti ne predstavlja problem. Postoje takvi pristupi koji garantiraju podjednako trošenje *svih* blokova (tzv. statički *wear leveling*) [23], no oni uvelike povećavaju složenost datotečnog sustava.



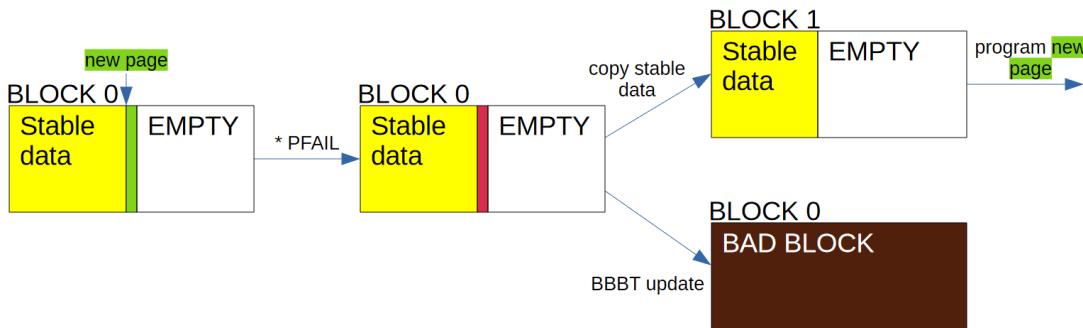
Slika 5.9: Red slobodnih fizičkih blokova.

Upravljanje neispravnim blokovima

Datotečni sustav vodi evidenciju neispravnih blokova (engl. *bad blocks*) putem bit tablice neispravnih blokova (*Bad Block Bit Table*, BBBT). Prilikom prvog korištenja memorije⁸ traže se tvorničke oznake neispravnih blokova te se tako označeni blokovi ne postavljaju u početni red slobodnih blokova. Također, neispravan blok može se pojaviti prilikom pokušaja programiranja ili brisanja bloka. Tada se ažurira BBBT tablica te se blok također više ne vraća u red slobodnih blokova. Postavlja se pitanje što ako blok koji na sebi već ima pohranjene neke podatke postane neispravan. U tom slučaju pristupa se tzv. *copyback* operaciji (slika 5.10), kojom je moguće sačuvati prethodno

⁸Oznaku o prvom korištenju memorije datotečni sustav upisuje na jednu od pomoćnih OTP stranica memorije.

upisane podatke.



Slika 5.10: *Bad block copyback* operacija. Prilikom pokušaja programiranja stranice dolazi do neuspjeha što memorija signalizira porukom o neispravnom bloku. Valjani sadržaj prebacuje se na novi blok, gdje se nastavlja s programiranjem, dok se neispravan blok evidentira u BBT tablici.

Zaglavljek datotečnog sustava

Zaglavljek datotečnog sustava (*filesystem header*) obuhvaća sve do sada opisane strukture (FAT tablica, BTT tablica, red slobodnih blokova, itd.); one su nužne za funkcioniranje datotečnog sustava. Gledajući iz perspektive mikrokontrolera, zaglavljek datotečnog sustava predstavlja skup globalnih varijabli. Ono se može podijeliti na brisljivi (*volatile*) i ne-brisljivi (*non-volatile*) dio. Ne-brisljivom dijelu pripada većina elemenata zaglavljaka datotečnog sustava: prije isključenja napajanja ne-brisljivi dio mora se pohraniti na Flash memoriju. Prilikom ponovnog uključenja napajanja, ne-brisljivi dio zaglavljaka datotečnog sustava potrebno je ponovno učitati u radnu memoriju mikrokontrolera. Brisljivi dio, pak, ne mora biti sačuvan: tu primjerice spadaju neke statusne zastavice iz opisnika datoteke. Zaglavljek datotečnog sustava snima se na Flash memoriju kao zaseban blok te podliježe istim pravilima za *wear leveling* kao i svi ostali blokovi. Također, kako će se pokazati u odjeljku o *journalingu*, ažuriranje pohranjenog zaglavljaka datotečnog sustava ne događa se nesrazmjerno rijetko u odnosu na stvaranje odnosno brisanje drugih datoteka. Kako bi se zaglavljek datotečnog sustava moglo pronaći u memoriji, njegov blok sadrži jedinstvenu oznaku. Također, blok sadrži i broj verzije zaglavljaka, u slučaju da u memoriji bude prisutno više od jednog zaglavljaka (primjerice ako nestane napajanja prije nego li se staro zaglavljek izbriše).

Journaling

Upis podataka u datoteku može se podijeliti na dva koraka: prvi korak je ažuriranje metapodataka (FAT tablica, BTT tablica, itd.), a drugi korak je sam upis podataka u

memoriju. No, što ako tijekom nekog od koraka dođe do iznenadnog pada sustava, uslijed primjerice gubitka napajanja? U slučaju da sustav padne tijekom prvog koraka, neće se dogoditi ništa značajno; tijekom ponovnog uključenja učitat će se zadnja verzija zaglavlja datotečnog sustava koja u sebi još nema reflektirane promjene u metapodacima. Do upisa kao i da nikad nije došlo, a metapodaci i podaci unutar datotečnog sustava su međusobno konzistentni. Ako do pada dođe tijekom drugog koraka, nakon ponovnog uključenja napajanja, metapodaci će također biti oni stari. No, po nekim od blokova možda je pisano, a o tome nigdje ne postoji evidencija. Valja se prisjetiti kako kod Flash memorije to predstavlja problem, budući da blok mora biti u izbrisanim stanju prije nego što se po njemu piše. Jedan od načina kako se može doskočiti ovom problemu je korištenje pristupa poznatog kao *journaling* (engl. *journal* - dnevnik) [14]. Jedan od blokova - tzv. *journal* blok - može se koristiti za bilježenje namjere o pisanju podataka na određeni blok (slika 5.11). U slučaju pada sustava, prilikom ponovnog uključenja slijedno se čita *journal* blok i svi blokovi koji su tamo zabilježeni proglašavaju se neispravnima i brišu. Svi slobodni blokovi tada su ujedno i prazni.

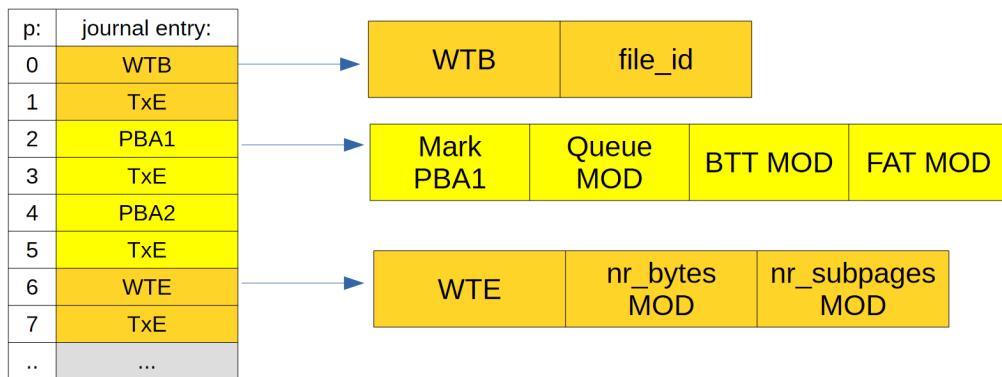
page:	journal content:
0	PBA1
1	TxE
2	PBA2
3	TxE
4	FREE
...	...
63	FREE

Slika 5.11: *Journal* blok. Fizičke adrese blokova po kojima se namjerava pisati bilježe se na zasebne stranice (stranica predstavlja jedan *journal entry* - stavku). Marker TxE garancija je da je upis na prethodnu stanicu u potpunosti dovršen, odnosno da je prethodna stavka u *journalu* ispravna.

No, što ako je upis u datoteku u potpunosti dovršen i tada dođe do pada sustava? Rekonstrukcijom *journal-a* (engl. *journal replay*) izbrisat će se blokovi koji sadrže valjane podatke. U tom slučaju, u *journal* je moguće zapisati oznaku početka (WTB) i oznaku kraja (WTE) pisanja datoteke (slika 5.12). Svi blokovi navedeni u *journalu* koji nisu zaključeni s WTE oznakom bit će izbrisani, dok ostali neće. Podaci datoteke čiji je upis dovršen sada ostaju očuvani, ali metapodaci nisu ažurirani! Ovaj problem moguće je riješiti upisom metapodataka u *journal*, kako prikazuje slika 5.13.

page:	journal content:
0	WTB
1	TxE
2	PBA1
3	TxE
4	PBA2
5	TxE
6	WTE
7	TxE
...	...

Slika 5.12: Oznake početka (*Write Transaction Begin* - WTB) i kraja upisa (*Write Transaction End* - WTE) u datoteku sprječavaju brisanje blokova s valjanim podacima. Vremenski gledano, do upisa u datoteku dolazi nakon postavljanja „PBA2+TxE“ stavki, a prije postavljanja „WTE“ stavke.



Slika 5.13: Proširenje stavke u *journalu* informacijama o ažuriranju metapodataka. Ažuriranje metapodataka prihvatić će se samo ako je prisutan WTE marker. MOD označava modifikaciju.

Budući da se *journal* snima na blok memorije, njegova veličina je ograničena na 32 *journal* stavke. Kada se *journal* blok zapuni, potrebno je redom: pronaći slobodni blok za novi *journal*, snimiti zaglavljne datotečnog sustava na Flash memoriju i izbrisati stari *journal* blok. Prilikom pronalaženja blokova za *journal* vrijede ista pravila za *wear leveling* kao i za ostale blokove. Informacija o tome koji blok se koristi za *journal* zapisana je u zaglavljju datotečnog sustava.

Vrste journalinga (journaling modes)

U specifikaciji zahtjeva s početka odjeljka navedeno je kako se tijekom misije satelita očekuju dvije vrste datoteka. Prva vrsta datoteka su slike, čija veličina doseže do 5 MB. Preračunato, to iznosi oko 40 blokova. Upis slike u memoriju odvijat će se putem međuspremnika u radnoj memoriji mikrokontrolera, u paketima veličine par kB.

U slučaju pada sustava tijekom upisa slike u memoriju, do tada upisane oktete nije potrebno sačuvati: očekuje se da će i kamera izgubiti napajanje pa je ostatak slike izgubljen, a djelomična slika ne smatra se veoma korisnom. Podaci sa senzora, pak, bit će veliki par desetaka kB i zauzimati jedan do dva bloka. Upis u memoriju također će se odvijati u paketima veličine par kB. Vremenski razmak između perioda upisa može biti značajan. U slučaju pada sustava, podaci upisani do tog trenutka smatraju se korisnima i treba ih očuvati. S obzirom na ove dvije vrste datoteka i njihove karakteristike, predviđene su dvije vrste *journalinga*: jaki (*strong*) i slabi (*weak*).

Jaki journaling predviđen je za datoteke koje pohranjuju podatke sa senzora. Ovdje se svaki upis na memoriju bilježi u *journal*, čak i kada nema alokacije novog bloka. Naime, u slučaju da dođe do pada sustava tijekom upisa, blok na koji se upisivalo sadrži potencijalno nevaljane (*corrupt*) podstranice. Kako bi se omogućio kasniji nastavak upisa u datoteku, uz očuvanje dotadašnjih podataka, potrebno je pristupiti *copyback* operaciji, slično onoj za neispravne blokove prikazanoj na slici 5.10: valjni podaci prebacuju se na novi blok gdje se može nastaviti s upisom, a stari blok se briše i vraća u red slobodnih blokova.

Slabi journaling predviđen je za slike. Kod slabog *journalinga*, do upisa na *journal* dolazi samo prilikom stvaranja datoteke i prilikom alokacije novog bloka. Naime, s obzirom na odnos veličine slike i međuspremnika u radnoj memoriji mikrokontrolera, za jednu RAW potrebno je tristotinjak pristupa memoriji radi upisa. Kada bi se svaki pristup vodio kroz *journal*, broj *journal* stavaka bio bi toliki da bi se nekoliko desetaka blokova moralо izložiti ciklusu pisanja i brisanja (za pohranu *journal* bloka i bloka zaglavljа datotečnog sustava). Kako se datoteka u slučaju pada sustava neće nadopunjavati, nema razloga za korištenje jakog *journalinga*. U slučaju pokušaja čitanja datoteke čiji je upis prekinut padom sustava, uspješno će se pročitati svi u potpunosti upisani blokovi.

Nestabilna stanja zaglavljа datotečnog sustava

Bilo koji upis stavke (*journal entryja*) na *journal* blok može rezultirati snimanjem trenutnog stanja zaglavljа datotečnog sustava na Flash memoriju. Ovo se događa ili zbog zapunjavanja *journal* bloka ili u slučaju pojave njegove neispravnosti (*bad block* - potrebno je pronaći novi). Moguće je da tada snimljeno stanje metapodataka u zaglavljу datotečnog sustava nije konzistentno s podacima: primjerice FAT tablica, BTT tablica i red slobodnih blokova su modificirani, ali podaci još nisu zapisani u Flash. Što ako baš u tom trenutku dođe do pada sustava? Tada se radi o tzv. nestabilnom stanju zaglavljа

datotečnog sustava. Stoga je prije upisa stavke na *journal* u takvom slučaju potrebno u zaglavlju datotečnog sustava postaviti posebnu zastavicu o nestabilnosti i naznačiti koja datoteka je nestabilna. Prilikom ponovnog uključenja napajanja, u slučaju izostanka WTE markera potrebno je FAT, BTT tablicu i red slobodnih blokova vratiti na prijašnje stanje.⁹

Dodatne napomene vezane uz *journaling*

Osim podataka o stvaranju datoteke i pisanja u datoteku, u *journal* se naznačuje i podatak o namjeri i dovršetku brisanja datoteke, kao i podatak o pokretanju i dovršetku *copyback* operacije, odnosno, sve što može rezultirati brisanjem ili alokacijom blokova. Dodatno, namjeru o zapisivanju zaglavlja datotečnog sustava na blok u memoriji također treba provesti kroz *journal*. Za to se koristi tzv. sekundarni *journal* blok, koji je neovisan o bloku za *journal* o kojemu se do sada raspravljalo (primarni *journal* blok).

Korisničke funkcije za rad s datotečnim sustavom

Za rad s datotečnim sustavom razvijene su sljedeće korisničke funkcije (isječak 5.5).

Isječak 5.5: Funkcije za rad s datotečnim sustavom.

```
uint16_t open(uint16_t file_name, uint8_t oflag);
uint32_t write(uint16_t filedes, void* buf, uint32_t nbyte);
uint32_t read(uint16_t filedes, void* buf, uint32_t nbyte);
uint8_t close(uint16_t filedes);
uint8_t remove(uint16_t file_name);

uint32_t get_file_size(uint16_t filedes);

uint8_t start_fs(void);
uint8_t suspend_fs(void);
void reinit_fs(void);
```

Prvih pet funkcija napravljeno je po uzoru na POSIX standard¹⁰, ali ne u punoj konformaciji s istim. Funkcije će ovdje biti opisane samo ukratko, dok je za detaljan opis potrebno pogledati u `filesys_api.h` i `filesys_api.c` unutar direktorija s programskim kodom projekta. Pomoću funkcije `open()` otvara se ili stvara datoteka

⁹Ovo se izvodi na temelju podatka o veličini datoteke u podstranicama koji je još onaj stari, budući da se ažurira zajedno s WTE markerom (slika 5.13).

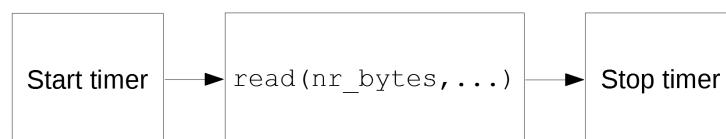
¹⁰Portable Operating System Interface [36].

imena `file_name`, ovisno o postavljenim zastavicama `oflag`. Zastavica `oflag` poprima vrijednost ili je kombinacija nekih od sljedećih vrijednosti: `O_RDONLY`, `O_WRONLY`, `O_CREAT`, `O_JSTRONG` i `O_JWEAK`. Posljednje dvije vrijednosti govore o vrsti *journalinga* koji će se koristiti za tu datoteku. Parametar `file_name` cijeli je broj između 0 i 1023. Funkcija `open()` vratit će identifikator datoteke `filedes` (interni `file_id`). Funkcija `write()` služi za upis, a funkcija `read()` za čitanje iz datoteke. Datoteka se zatvara funkcijom `close()`, a briše funkcijom `remove()`. Sve ove funkcije signalizirat će pogrešku (u slučaju iste) putem povratne vrijednosti. Također, u globalnoj varijabli `errno` bit će vidljiv kod pogreške. Veličinu datoteke moguće je doznati pozivom funkcije `get_file_size()`.

Funkcijom `start_fs()` inicijalizira se datotečni sustav nakon reseta mikrokontrolera. Funkcija `suspend_fs()` služi pohrani trenutnog zaglavlja datotečnog sustava na Flash memoriju: potrebno ju je pozvati prije isključenja napajanja mikrokontrolera. Funkciju `reinit_fs()` potrebno je pozvati ako bilo koja od korisničkih funkcija signalizira kritičnu grešku u datotečnom sustavu.¹¹ U tom slučaju `reinit_fs()` će izbrisati sve blokove na memoriji i ponovno inicijalizirati datotečni sustav.

Brzina čitanja iz datoteke

S obzirom na zahtjeve unutar misije satelita, brzina upisa u datoteku ne predstavlja kritičan čimbenik, stoga se istu nije pokušalo utvrditi. Brzina čitanja, pak, trebala bi biti što veća: cilj je omogućiti odašiljanje X-band predajnika brzinom od 16 Mbit/s. U svrhu utvrđivanja brzine čitanja iz datoteke provedena su mjerena, koristeći postupak prikazan slikom 5.14. Rezultati mjerena prikazani su tablicom 5.1.



Slika 5.14: Postupak utvrđivanja vremena izvođenja funkcije `read()`. Rezolucija vremen-skog sklopa je jedna mikrosekunda. Uvidom u asemblerski kod potvrđeno je da se operacije odvijaju kronološki, kako je prikazano slikom.

Uključenje optimizacije prevodioca povoljno utječe na brzinu čitanja. Korištena je razina optimizacije GCC prevodioca `-O1`; razinom `-O3` moguće je postići još veću brzinu (dodatnih 1.5 Mbit/s u slučaju (3) iz tablice), ali `-O3` u ovoj aplikaciji stvara nestabilnosti prilikom korištenja operacijskog sustava FreeRTOS. Slučajevi (2) i (3)

¹¹Primjer kritične greške je nemogućnost pohrane zaglavlja datotečnog sustava.

odnose se na datoteke kod kojih su svi podatkovni okteti u podstranicama (svih 510) iskorišteni. Na takav način pohranjuju se slike, kao i podaci sa senzorske pločice u slučaju da je veličina paketa u oktetima višekratnik broja 510. Kada bi se, primjerice, pohrana u datoteku obavljala u paketima od 2 okteta (slučaj (4)), osim što bi to rezultiralo vrlo velikom internom fragmentacijom podstranice, značajno bi se usporila brzina čitanja, budući da se u tom slučaju tijekom izvođenja `read()` funkcije većina vremena provodi u manipulaciji s metapodacima. Prilikom čitanja iz Flash memorije i slanja na X-band odašiljač, koristit će se međuspremnik u memoriji. Za najveću brzinu čitanja taj međuspremnik treba biti veličine 2040 okteta, jer u tom slučaju ne dolazi do zamjene stranice u memoriji (za dohvatanje potrebno je neko vrijeme).¹²

Tablica 5.1: Izmjerene brzine čitanja iz memorije prilikom korištenja `read()` funkcije.

	brzina čitanja [Mbit/s]	optimizacija prevodioca	uvjeti
(1)	15.5	isključena	čitanje bez zamjene stranice, svi okteti u podstranicama iskorišteni
(2)	23.9	-o1	čitanje bez zamjene stranice, svi okteti u podstranicama iskorišteni
(3)	21.6	-o1	čitanje sa zamjenama stranice, svi okteti u podstranicama iskorišteni
(4)	0.5	-o1	čitanje sa zamjenama stranice, iskorištena 2 od 510 podatkovnih okteta po podstranici

5.4. Modul s fotosenzorima

Za korištenje modula s fotosenzorima razvijene su sljedeće funkcije (isječak 5.6).

Isječak 5.6: Funckije za rad s modulom s fotosenzorima.

```
void SB_init(void);
void SB_MUX_channel_select(uint32_t channel);
void SB_PGA_set_gain(pga_gain gain);
void SB_ADC_collect_sample(uint16_t* sample);
void SB_ADC_collect_samples_unmix(uint16_t* samples, uint32_t nr_acq);
void SB_ADC_collect_samples_UV(uint16_t* samples, uint32_t nr_acq);
```

¹²Dodata napomena: testirani slučajevi (1), (2) i (3) odnose se na datoteke veličine jednog bloka. Mjerenja pokazuju kako vrijeme dohvata adrese bloka iz FAT tablice ne predstavlja značajan čimbenik za datoteke veličine 40 blokova (slike u RAW formatu).

Funkcija `SB_init()` inicijalizira sve periferne jedinice potrebne za rad s modulom s fotosenzorima. Funkcijom `SB_MUX_channel_select()` vrši se odabir kanala multipleksora, odnosno odabir jednog od šest fotosenzora na modulu¹³. Pojačanje programabilnog pojačala bira se funkcijom `SB_PGA_set_gain()`. Moguće vrijednosti pojačanja dane su enumeracijskim tipom `pga_gain`:

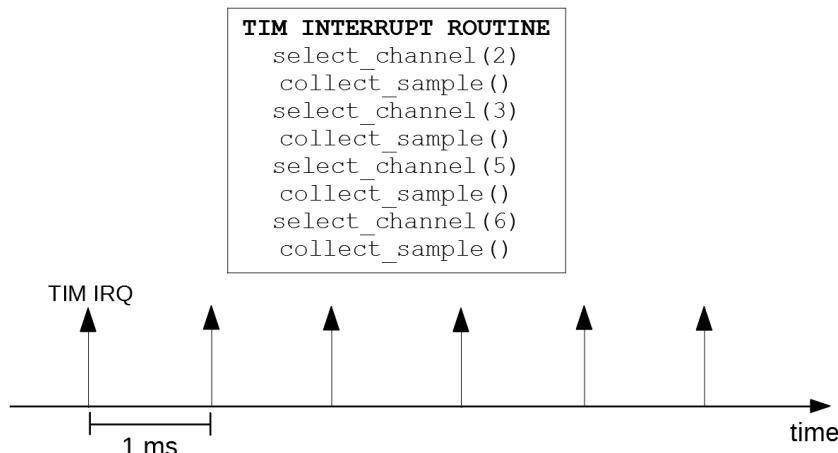
```
typedef enum{
    PGA_GAIN_1,
    PGA_GAIN_2,
    PGA_GAIN_4,
    PGA_GAIN_5,
    PGA_GAIN_8,
    PGA_GAIN_10,
    PGA_GAIN_16,
    PGA_GAIN_32
} pga_gain;
```

Funkcija `SB_ADC_collect_sample()` koristi se za prikupljanje jednog uzorka s prethodno odabranog kanala. Funkcijom `SB_ADC_collect_samples_unmix()` frekvencijom 1 kHz uzorkuju se četiri kanala koja se koriste za utvrđivanje komponenti spektra svjetlosti (svake milisekunde uzorkuju se sva četiri kanala¹⁴). Uzorkovanje se vrši na prekid vremenskog sklopa (slika 5.15), ukupno u `nr_acq` navrata. Funkcija će završiti s izvođenjem tek kada su svi uzorci prikupljeni. Za pohranu uzorka koriste se polurijeći: 12 najmanje značajnih bitova sadrži vrijednost uzorka, dok ostala četiri bita imaju vrijednost nula. Funkcija `SB_ADC_collect_samples_UV()` na isti način prikuplja uzorke s dvaju kanala s UV detektorima, uz iznimku da je frekvencija uzorkovanja 2 Hz.

Unutar funkcija za rad s modulom s fotosenzorima na određena mesta bilo je potrebno umetnuti odgovarajuća čekanja, s obzirom na specificirane vremenske parametre komponenata (npr. vrijeme potrebno za stabilizaciju izlaza programabilnog pojačala nakon promjene vrijednosti pojačanja).

¹³Netko će se možda zapitati zašto funkcija ne prima argument tima `uint8_t`, s obzirom na broj mogućih kanala. Pokazuje se da korištenje prirodnog tipa argumenta, što je za danu arhitekturu riječ (32 bita), ubrzava izvođenje funkcije.

¹⁴Zbog vremena potrebnog za prijenos s A/D pretvornika na mikrokontroler, navedena četiri uzorka vremenski su posmagnuta za otprilike 100 us.



Slika 5.15: Prikupljanje uzoraka s kanala koji se koriste za utvrđivanje komponenti spektra svjetlosti (kanali 2, 3, 5 i 6).

5.5. X-band odašiljač

Za rad s X-band odašiljačem razvijene su sljedeće funkcije (isječak 5.7).

Isječak 5.7: Funckije za rad s X-band odašiljačem.

```

void XBAND_init(void);
void XBAND_config_addr(uint16_t* address);
void XBAND_transmit(uint32_t nr_data);

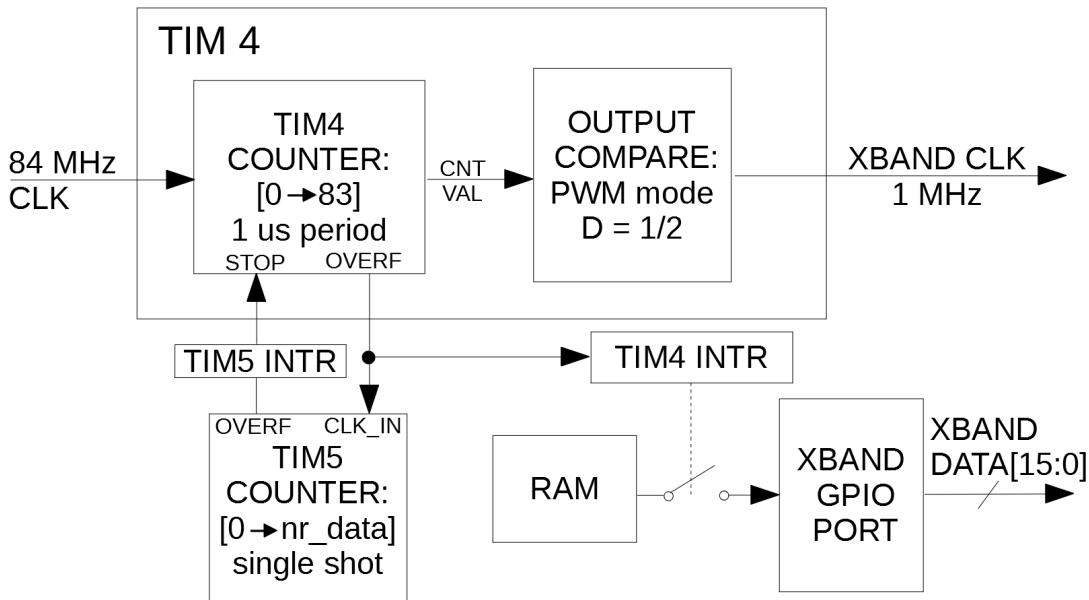
```

Funkcija XBAND_init() vrši inicijalizaciju GPIO pinova potrebnih za sučelje prema X-band odašiljaču i postavlja početnu konfiguraciju vremenskih sklopova koji se koriste za rad. Funkcijom XBAND_config_addr() postavlja se adresa spremnika u radnoj memoriji unutar kojeg su podaci koji se trebaju poslati preko X-band odašiljača. Slanje započinje pozivom funkcije XBAND_transmit(), kojoj je potrebno predati broj poluričeći (16-bitnih podataka) za slanje. Postavljanje vrijednosti GPIO pinova odvija se unutar prekidne rutine. Status operacije slanja moguće je provjeriti putem globalne varijable xband_transmit_ongoing: ukoliko je njezina vrijednost 0, slanje je dovršeno.¹⁵

Princip rada sučelja mikrokontrolera prema X-band odašiljaču prikazan je na slici 5.16. Za rad su potrebna dva vremenska sklopa: TIM4 i TIM5. Brojač vremenskog sklopa TIM4 pokretan taktom od 84 MHz neprekidno broji od 0 do 83 (pa opet od nula). Vrijednost brojača služi kao ulaz u sklop za usporedbu (*output compare*), koji je

¹⁵Za provjeru vrijednosti ove globalne varijable nije potrebno isključiti odgovarajući prekid, budući da je pristup poravnatoj riječi kod Cortex-M procesora atomaran.

konfiguriran tako da za vrijednost brojača koja je manja od 42 na svoj izlaz postavlja nisku naponsku razinu, a za vrijednost veću ili jednaku 42 postavlja visoku naponsku razinu. Ovime se generira pravokutni signal frekvencije 1 MHz, jednakog trajanja niske i visoke razine.



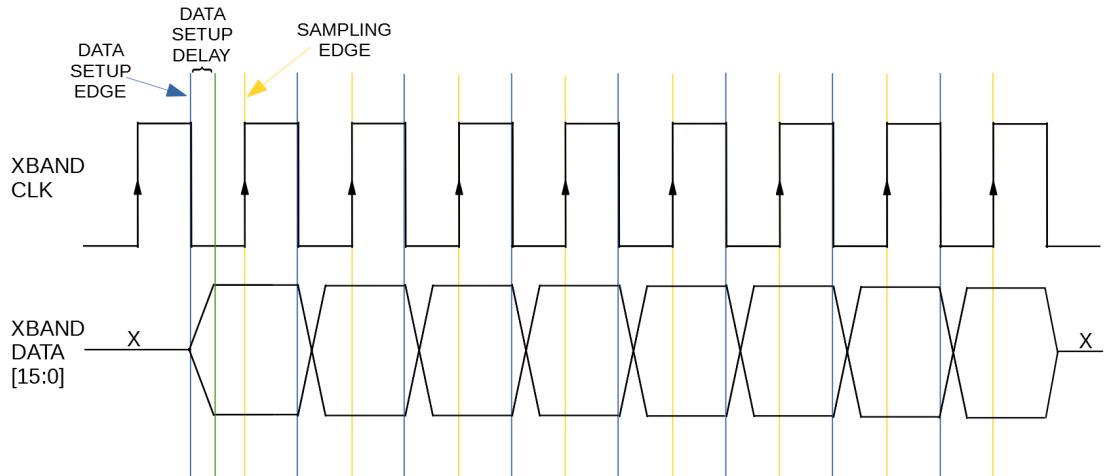
Slika 5.16: Realizacija sučelja prema X-band odašiljaču.

Na svaki preljev (*overflow*) brojača TIM4, dakle svake mikrosekunde, generira se prekid. Unutar prekidne rutine, poluriječ iz radne memorije spremna za slanje postavlja se na odgovarajući GPIO skup priključaka. Također, preljev brojača TIM4 služi kao takt brojaču vremenskog sklopa TIM5. Tako brojač TIM5 povećava svoju vrijednost svake mikrosekunde i broji do vrijednosti `nr_data`, tj. do ukupnog broja 16-bitnih podataka za slanje. Kada dođe do preljeva brojača TIM5, izaziva se prekid unutar kojeg se zaustavlja brojač vremenskog sklopa TIM4. Time je slanje paketa podataka na X-band odašiljač dovršeno. Korištenje jednog vremenskog sklopa za ograničenje vremena aktivnosti drugog skraćuje vrijeme provedeno u prekidnoj rutini (prekidna rutina samostalnog vremenskog sklopa morala bi voditi računa o broju poslanih podataka).

Na slici 5.17 prikazan je izgled signala generiranih za X-band odašiljač na primjeru slanja paketa od osam poluriječi. Podaci se na GPIO priključke postavljaju na padajući brid takta, dok odašiljač mora uzorkovati podatke na rastući brid takta. Podaci su važeći na svaki rastući brid takta osim prvog.¹⁶ Između padajućeg brida takta i trenutka

¹⁶Moguća je izvedba kod koje su podaci važeći na prvi rastući brid takta. No, zbog ograničenja vezanih uz konfiguraciju vremenskog sklopa, tada nije moguće poslati samo jedan podatak (jednu poluriječ), već su minimalno potrebna dva.

kada su na GPIO priključcima važeće vrijednosti postoji vremenska odgoda; radi se o latenciji obrade prekida. Mjerenjima je utvrđeno da ta odgoda iznosi najviše 230 ns (slika 5.18), dakle podatak se postavlja na vrijeme, prije pojave rastućeg brida.¹⁷



Slika 5.17: Slanje paketa od osam poluričeći: signali na X-band CLK i DATA linijama.



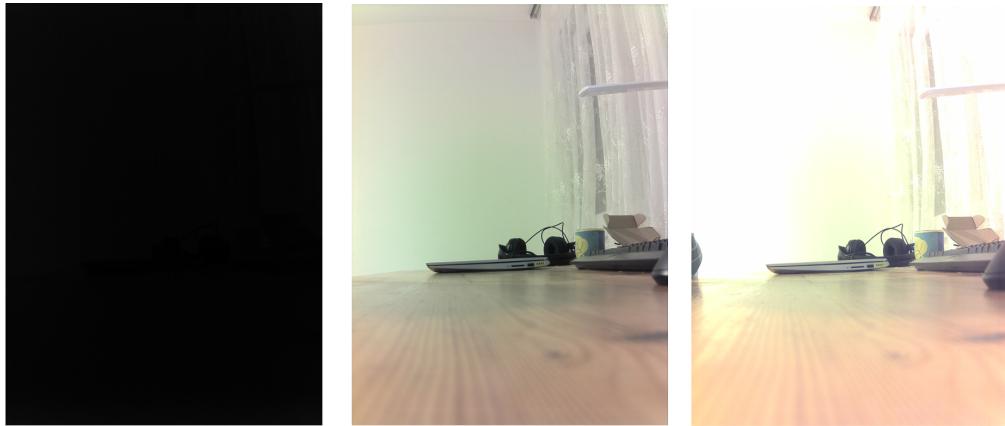
Slika 5.18: Snimljeni signali na X-band CLK (kanal 1, žuto) i DATA[0] liniji (kanal 2, ljučasto) prilikom slanja paketa od osam poluričeći. Bit na DATA[0] liniji neprestano mijenja vrijednost (*toggle*). Latencija postavljanja podataka na GPIO linije iznosi 230 ns.

5.6. Testiranje modula razvijene programske potpore

Programska potpora za sve navedene cjeline (kamera, memorija, X-band odašiljač i modul s fotosenzorima) podvrgнутa je testiranju. Rad svih korisničkih funkcija kamere uspješno je provjeren. Slike u RAW i JPEG formatu na računalo su prebacivane putem UART sučelja (slika 5.19). Rad datotečnog sustava i svih njegovih kom-

¹⁷Latencija iznosi 230 ns za prvi podatak, 190 ns za ostale.

ponenti (*wear leveling*, upravljanje neispravnim blokovima, *journaling*) temeljito je testiran. Pad sustava kod ispitivanja *journalinga* simuliran je umetanjem beskonačne petlje (`while (1)`) na odabrana mjesta unutar programskog koda. Rad sučelja prema X-band odašiljaču testiran je snimanjem signala na osciloskopu (slika 5.18), ili povezivanjem X-band CLK i DATA linija na svjetleće diode razvojnog sustava.

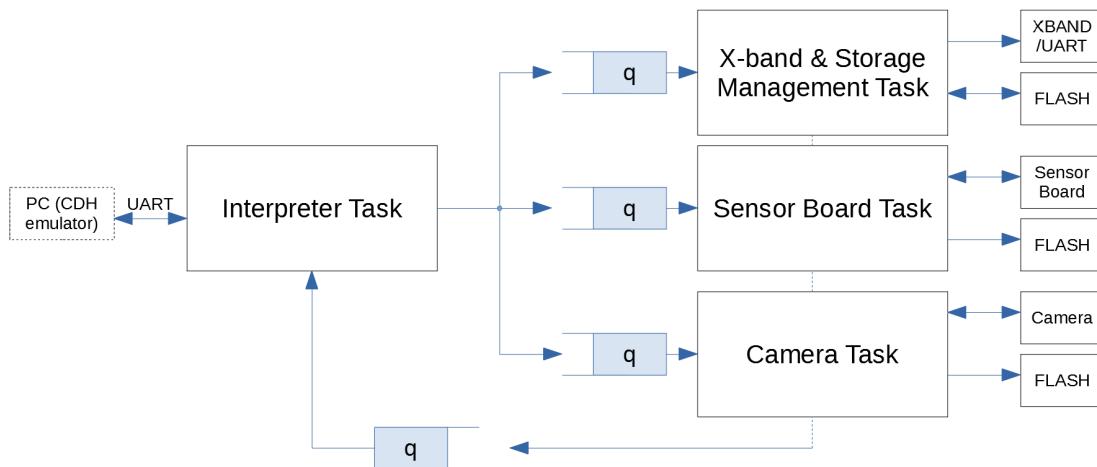


Slika 5.19: Neke od slika (u JPEG formatu) snimljene Arducam 5MP Mini Plus kamerom. Vidljiv je utjecaj promjene trajanja ekspozicije na svjetlinu slike; lijeva slika krajnje je podeksponirana: obrise je jedva moguće razaznati.

Prilikom pokušaja testiranja funkcija za rad s modulom s fotosenzorima došlo je do poteškoća: nekoliko sekundi nakon uključenja napajanja pločice sa senzorima, analogno-digitalni pretvornik postao bi toliko vruć da se više nije mogao dotaknuti prstima. Do premljena senzorska pločica prethodno nije testirana; moguće je da sve komponente nisu ispravno spojene. Druga mogućnost je da do neispravnog rada dolazi zbog ne-kompatibilnih logičkih razina. Naime, mikrokontroler koristi ulazno/izlaznu logiku na 3.3 V, dok komponente senzorskog modula koriste 5 V logiku. Mjerenjem je utvrđeno da GPIO pinovi mikrokontrolera pri visokoj naponskoj razini imaju vrijednost napona od 2.64 V (unutar dozvoljenog raspona), dok je napon praga komparacije za visoku naponsku razinu A/D pretvornika 2.7 V [16]. Postavljanjem /CS signala u visoku razinu A/D pretvornik bi se trebao isključiti; moguće je kako je zbog stvarnog napona na ulazu ostao trajno uključen te se zbog toga pregrijavao. No, spajanjem /CS linije direktno na 5 V i dalje dolazi do pregrijavanja. Zbog poteškoća u pribavljanju druge senzorske pločice u kratkom vremenu, odlučeno je stvari ostaviti kakve jesu. Točnije, prisustvo senzorske pločice u sustavu može se emulirati: spajanjem MISO linije A/D pretvornika priteznim otpornikom na GND, SPI periferija mikrokontrolera uvijek će kao primljeni podatak očitavati nule. Pozivom funkcije `SB_ADC_collect_samples()` dobiva se očekivani broj polurijeći (vrijednosti 0000h).

5.7. Integracija operacijskog sustava FreeRTOS

Razvijene module programske potpore potrebno je ujediniti u cjelinu. Za ostvarenje višezadaćnosti korišten je operacijski sustav za rad u stvarnom vremenu FreeRTOS [10]. Raspodjela poslova na zadatke i način interakcije zadataka prikazan je slikom 5.20.

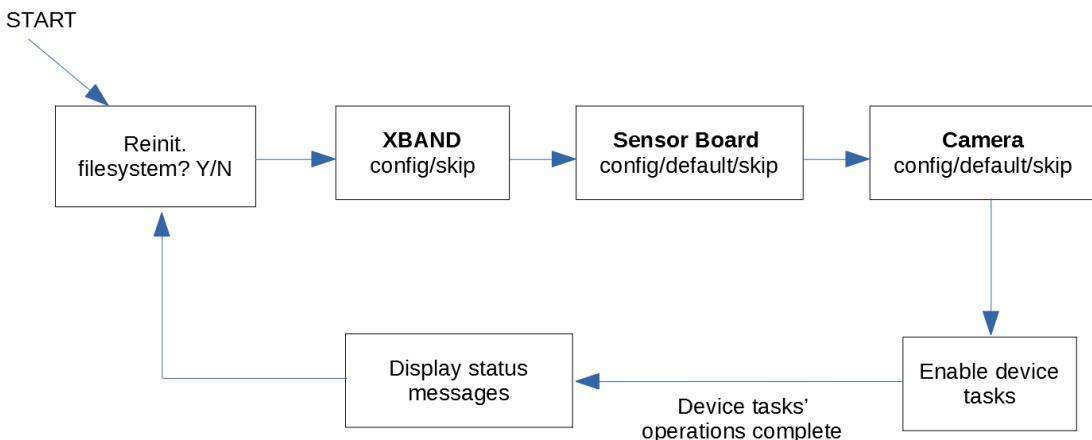


Slika 5.20: Zadaci u operacijskom sustavu PDH računala.

Budući da u trenutku pisanja ovog rada točna specifikacija sučelja (protokola komunikacije) prema CDH računalu nije bila poznata, razvijen je demonstracijski program u kojem se PDH računalo povezuje s osobnim računalom putem UART sučelja. Osobno računalo u tom slučaju služi kao emulator CDH računala, te se putem njega daju naredbe za obavljanje određenih poslova na PDH računalu. Na strani PDH računala, za komunikaciju je zadužen *interpreter* zadatak: na temelju primljenih poruka s osobnog računala, *interpreter* zadatak postavlja parametre za ostale zadatke (*device tasks*), koji su prikazani tablicom 5.2. Automat stanja *interpreter* zadatka prikazan je slikom 5.21: za svaki od uređaja (X-band odašiljač, modul s fotosenzorima, kamera) moguće je odabrati želi li se isti koristiti (u tom slučaju podešavaju se parametri) ili ne. Dodatno, modul s fotosenzorima i kamera imaju mogućnost korištenja unaprijed zadanih vrijednosti (*default*), odnosno istih onih iz prethodnog ciklusa korištenja. U tom slučaju samo je potrebno specificirati ime datoteke za pohranu. Nakon što su svi *device* zadaci obavili svoje zadaće, *interpreter* zadatak će CDH računalu poslati poruku o uspješnom, odnosno neuspješnom izvođenju pojedinih zadataka. U slučaju kritične greške datotečnog sustava, moguće ga je ponovno inicijalizirati.

Tablica 5.2: Konfiguracijski parametri za uređaje PDH računala.

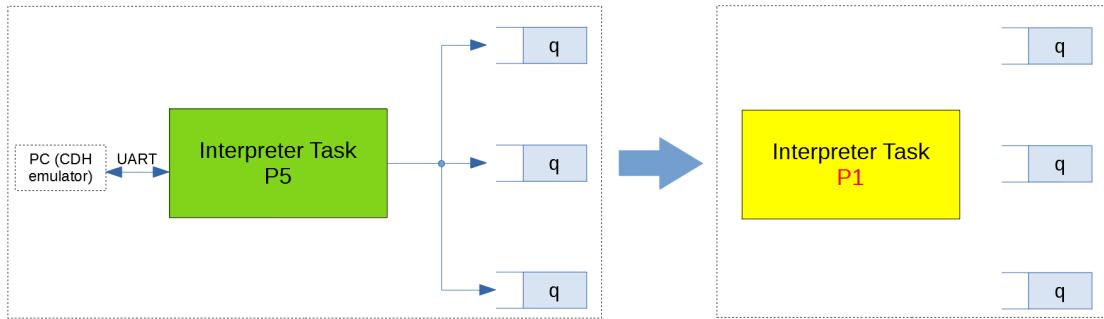
xband/data manag.	sensor board	camera
reinit. filesystem y/n	PGA gain	exposure nr_lines
file to transmit name	mode select	exposure nr_lines_frac
delete after transmission y/n	number of acquisitions	gain
delete some other file y/n	channel select	format (raw/jpg)
file to delete name	file name	file name



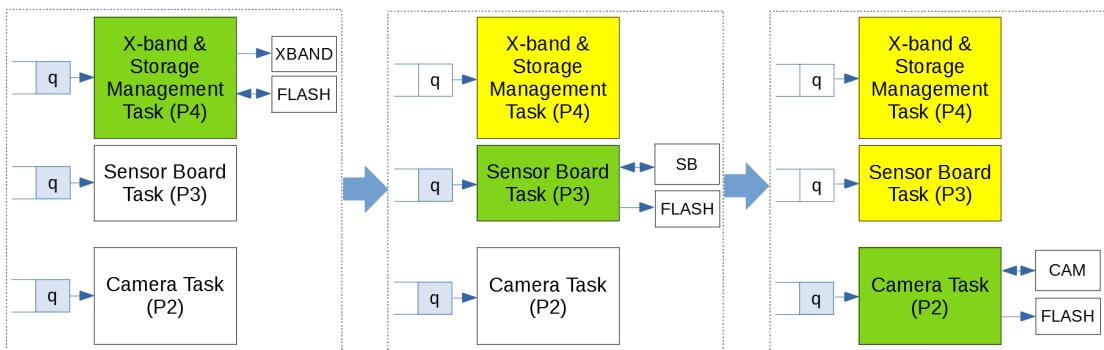
Slika 5.21: *Interpreter* zadatak: automat stanja iz prespektive sučelja prema CDH računalu.

Komunikacija zadataka odvija se putem redova poruka (slika 5.20). *Interpreter* zadatak je zadatak najvišeg prioriteta (P5). Nakon što taj zadatak dovrši komunikaciju s računalom, puni odgovarajuće redove poruka *device* zadataka. Dubina ovih redova poruka je jedan: poruka sadrži strukturu sa svim parametrima (tablica 5.2) potrebnim za taj zadatak. Nakon toga, *interpreter* zadatak spušta svoj prioritet na P1 (slika 5.22); tada se izvršavaju *device* zadaci prioriteta P4-P2. Ti zadaci izvršavaju se slijedno, primajući pritom poruku iz pripadajućeg reda poruka (slika 5.23); na praznom redu se blokiraju. Dijeljene resurse (primjerice Flash memoriju) nije potrebno zaštiti na poseban način budući da ne može doći do istovremenog pristupa više zadataka. Svaki zadatak, po završetku izvođenja, postavlja poruku u posebni statusni red poruka, dubine tri elementa. Svaka poruka u tom redu sastoji se od zastavice koja označava uspješno odnosno neuspješno izvođenje, koda pogreške (ako postoji) i imena datoteke koja je zahvaćena pogreškom (ako postoji). Budući da su svi *device* zadaci sada blokirani na praznom redu, ponovno se pokreće *intepreter* zadatak, koji čita poruke iz statusnog reda i šalje informacije o pročitanim porukama na UART (slika 5.24). Nakon što obradi očekivani broj poruka, podiže svoj prioritet na P5 te se ciklus aktivnosti

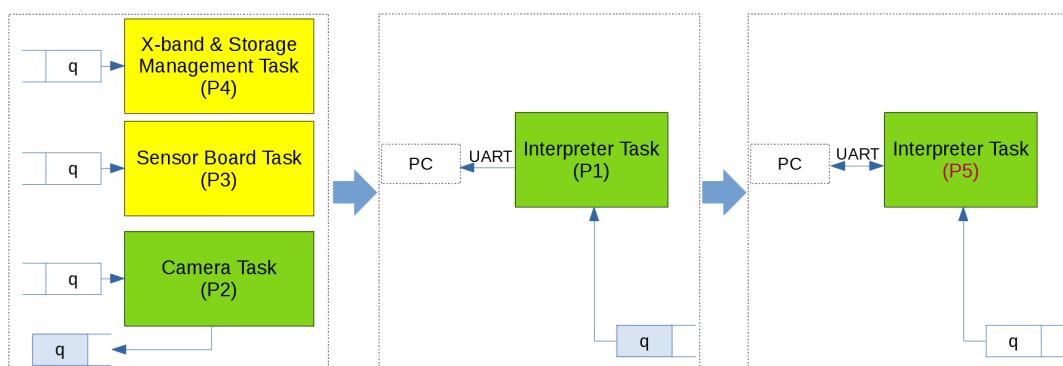
PDH računala ponavlja.¹⁸



Slika 5.22: *Interpreter* zadatak: postavljanje poruka u redove. Nakon što su sve poruke postavljene, *interpreter* zadatak spušta svoj prioritet na najmanji u sustavu (izuzev IDLE zadatka).



Slika 5.23: *Device* zadaci: redoslijed izvršavanja. Zadatak u izvođenju označen je zelenom bojom.



Slika 5.24: Punjenje (prikaz za zadatak kamere) i pražnjenje reda s porukama o statusu *device* zadataka.

¹⁸Alternativno, *interpreter* zadatak mogao bi se blokirati na praznom redu statusnih poruka: tada bi se izvršavao IDLE zadatak (prioritet 0), sve do nekog vanjskog događaja (signalizacija od stane CDH računala). No, kako FreeRTOS port za STM32F4 nema podršku za *tickless* IDLE zadatak (koji omogućuje rad mikrokontrolera u režimu male potrošnje), odnosno isti mora biti korisnički implementiran (netrivijalno), rješenje koje bi obuhvačalo IDLE task nije realizirano.

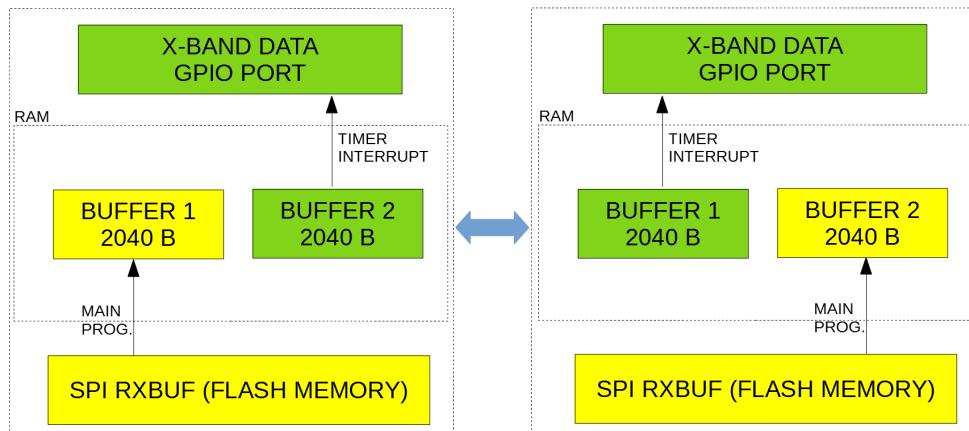
Kako bi se omogućio prijenos slika i podataka sa senzorske pločice na računalo, na temelju konfiguracijske konstante XBAND_SELECT u main.h moguće je odabratи hoće li X-band zadatak zaista koristiti sučelje prema X-band odašiljaču, ili će se podaci preusmjeriti na UART.

Sustav *de facto* koristi kooperativnu višežadačnost: ni u jednom trenutku zadatak višeg prioriteta neće istisnuti zadatak nižeg prioriteta [30]. Iako se konfiguracijska konstanta configUSE_PREEMPTION u teoriji ne mora postaviti na 0, to je dobro učiniti kako bi se skratio vrijeme izvođenja *tick* prekida. Tijekom normalnog rada operacijskog sustava *tick* prekid nije moguće isključiti, a ova aplikacija napravljena je tako da *tick* prekid zapravo nije potreban. Kako bi se izvođenje zadataka što manje prekidalo, uputno je vrijednost konstante configTICK_RATE_HZ postaviti na što manju moguću; za Cortex M4 to je 11 Hz.¹⁹

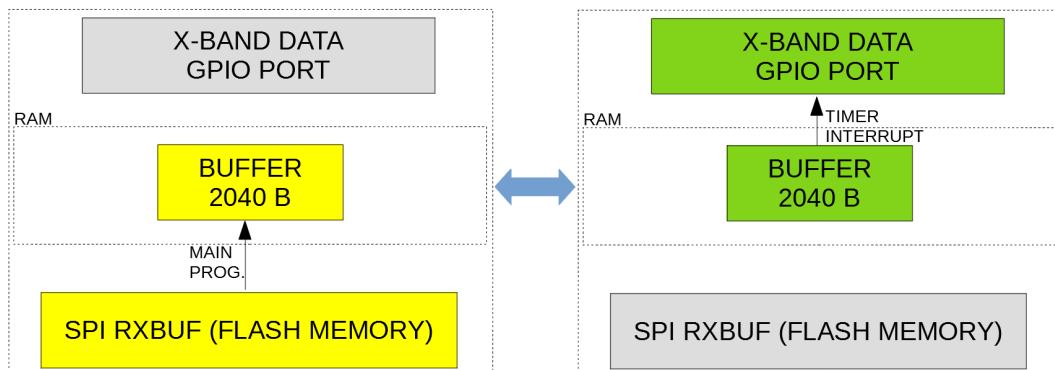
X-band odašiljač radi brzinom odašiljanja do 16 Mbit/s. U idealnom slučaju, PDH računalo šalje podatke X-band odašiljaču dovoljno brzo te odašiljač radi kontinuirano, ostvarujući pritom maksimalnu brzinu. Ukoliko PDH računalo nije dovoljno brzo, odašiljač će raditi s prekidima i efektivna brzina bit će manja. Budući da brzina čitanja podataka iz datotke iznosi 23.9 Mbit/s (odjeljak 5.3), bilo je izgledno da će se maksimalna brzina odašiljanja moći ostvariti. U tu svrhu koristio bi se pristup s dvostrukim međuspremnikom u radnoj memoriji, prikazan slikom 5.25. Nažalost, ovo nije bilo moguće ostvariti budući da prekid vremenskog sklopa u trenutku dok se odvija komunikacija s memorijom dovodi do greške i „smrzavanja“ SPI komunikacije. Vjerovatni razlog je istek nekog internog brojača na strani memorije.²⁰ Komunikacija je zbog navedenih razloga izvedena korištenjem jednog međuspremnika u radnoj memoriji (slika 5.26), što neminovno rezultira prekidima u komunikaciji i manjom efektivnom brzinom (slika 5.27).

¹⁹Minimalna teoretska frekvencija zapravo je nešto manja, no onda se konfiguracija ne može obaviti putem sučelja FreeRTOS-a, već je potreban direktni upis u *reload* registar *systick timera* [13].

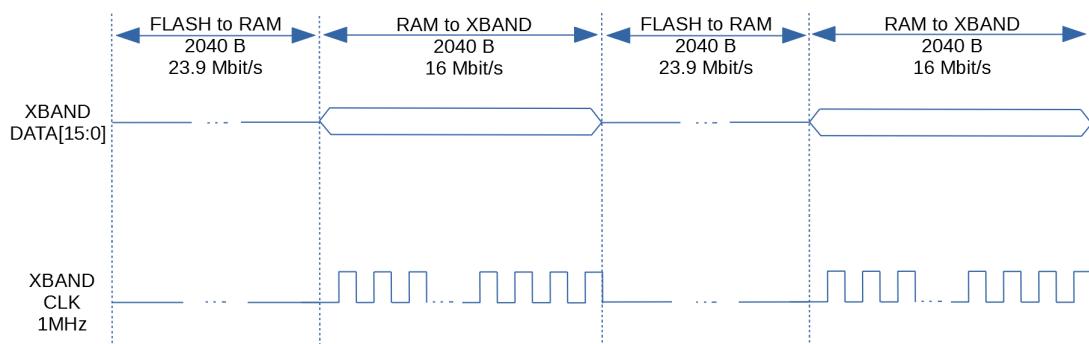
²⁰Razmotrena je i mogućnost korištenja DMA jedinice umjesto prekida, no poziv DMA transakcije na događaj vremenskog sklopa nije moguće realizirati s automatskim inkrementom adrese podatka za prijenos: navedeno bi opet bilo potrebno napraviti u prekidnoj rutini.



Slika 5.25: Postavljanje podataka iz Flash memorije na X-band GPIO pinove koristeći dvostruki međuspremnik u radnoj memoriji. Korišteni međuspremniči su veličine 2040 B kako bi se prilikom prijenosa slika (i općenito datoteka kojima su svi okteti unutar podstranice iskorišteni) izbjegla zamjena stranice unutar `read()` funkcije.



Slika 5.26: Postavljanje podataka iz Flash memorije na X-band GPIO pinove koristeći jedan međuspremnik u radnoj memoriji.



Slika 5.27: Signalni generirani za X-band odašiljač u slučaju korištenja jednog međuspremnika u radnoj memoriji. Efektivna brzina prijenosa iznosi približno 9.6 Mbit/s.

Prekidi prisutni u sustavu prikazani su tablicom 5.3. Konfiguracija prekida je takva da prioritetniji prekid može istisnuti manje prioritetan (za enkodiranje razine prekida koriste se samo *preemption* bitovi). Iako bi se na prvi pogled moglo činiti da nije tako, prekidi vremenski kritičnih aktivnosti ne interferiraju međusobno. Primjerice, prekid vremenskog sklopa TIM7 (akvizicija podataka s modula s fotosenzorima) može prekinuti izvođenje prekida vremenskog sklopa TIM4 (postavljanje podataka na X-band GPIO priključak). No sustav je dizajniran tako da se prikupljanje uzorka s modula s fotosenzorima i slanje na X-band odašiljač ne mogu odvijati istovremeno.

Tablica 5.3: Prikaz prekida i njihovih prioriteta. Manji broj označava veći prioritet (suprotno od prioriteta zadatka). Pozivom funkcije `taskENTER_CRITICAL()` onemogućit će se svi prekidi manje prioritetni od razine dane s `configMAX_SYSCALL_INTERRUPT_PRIORITY`. Napomena: Moguće su i druge kombinacije dodjele prioritetnih razina prekidima.

prekid	opis	prioritet
Kernel Interrupt	svaki <i>tick</i> period	15 ↑
DMA1 Stream 3	ACAM SPI RX TC	6
DMA1 Stream 4	ACAM SPI TX TC	6
USART2	UART RXNE	5
configMAX_SYSCALL_INTERRUPT_PRIORITY	-	4
I2C1	ACAM I2C, various	3
TIM4	XBAND GPIO SET	2
TIM5	XBAND transmission length limiter	1 ↓
TIM7	ADC sample	1 ↓

Kako se SPI komunikacija s Flash memorijom ne smije prekidati, prije njezina početka potrebno je onemogućiti prekide pozivom funkcije `taskENTER_CRITICAL()`.²¹ Radi ostvarenja maksimalne propusnosti, isto je potrebno napraviti i dok traje komunikacija s X-band odašiljačem (prikazana slikom 5.27)

Za sustav poput satelita potreban je i sklop za detekciju neispravnog rada (*Watchdog timer*), odnosno uređaj kojeg je potrebno svako toliko „poslužiti“ kako isti ne bi resetirao mikrokontroler (ako ne dođe do posluživanja, pretpostavka je da je mikrokontroler zapeo s radom).²² Podrška za rad s *Watchdog* sklopom za sada nije implementirana (predviđeno je korištenje vanjskog sklopa, nasuprot onog integriranog u mikrokontroler).

²¹Za razliku od funkcije `taskDISABLE_INTERRUPTS()`, funkcija `taskENTER_CRITICAL()` omogućuje gniažđenje svojih poziva.

²²*Watchdog* skloovi mogu koristiti i intervale do nekoliko desetaka sekundi, pa se atomarnost operacije slanja podataka na X-band odašiljač može održati.

6. Zaključak

U ovom radu dan je pregled funkcije i temeljnih komponenti FERSAT nanosatelita. Opisane su komponente korisnog tereta te su prikazani zahtjevi na funkcionalnost PDH računala. Za mikrokontroler unutar PDH računala odabran je model STM32F407 proizvođača STMicroelectronics. Za kameru je odabran model 5MP Mini Plus prozvođača Arducam, temeljen na OV5642 CMOS senzoru slike. Za kameru je razvijen upravljački program koji omogućuje prikupljanje slika razlučivosti 5 megapiksela u RAW i JPEG formatu, uz mogućnost ručnog upravljanja trajanjem ekspozicije. Za vanjsku memoriju odabran je model W25N01GV proizvođača Winbond, temeljen na NAND Flash tehnologiji. Za navedenu memoriju razvijen je datotečni sustav temeljen na FAT tablici, s podrškom za ujednačenje trošenja blokova (*wear leveling*) i upravljanje neispravnim blokovima (*bad block management*), što će osigurati očuvanje funkcionalnosti memorije tijekom trogodišnjeg trajanja misije satelita. Dodatno, kao zaštita od privremenih grešaka u radu Flash memorije, osposobljeno je korištenje kodova za ispravljanje pogrešaka (ECC), a u svrhu očuvanja konzistencije datotečnog sustava u slučaju nenadanog prekida izvođenja programa (primjerice, zbog propada u naponu napajanja) implementiran je *journaling*. Nadalje, realizirana je programska potpora za interakciju s modulom X-band odašiljača putem paralelnog 16-bitnog sinkronog sučelja, kao i programska potpora za rad s modulom s fotosenzorima. Integracija programskih komponenti u cjelinu ostvarena je pomoću operacijskog sustava FreeRTOS koji radi u modu kooperativne višezadačnosti. Funkcionalnost programske potpore testirana je povezivanjem mikrokontrolera s osobnim računalom koje služi kao emulator upravljačkog CDH računala. Ispravan rad bilo je moguće potvrditi za sve programske komponente izuzev one za rad s modulom s fotosenzorima, čija je ispravnost verificirana samo djelomično, zbog problem sa sklopopovljem dostupnog modula s fotosenzorima.

LITERATURA

- [1] Arducam STM32 Demo Software. URL <https://github.com/ArduCAM/STM32>. Pristupljeno: lipanj 2021.
- [2] Tanbakuchi A. et al. Adaptive pixel defect correction. SPIE and IST: Electronic Imaging Conference, 2003.
- [3] Barić A. et al. Elektronika 1, skripta. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 2016.
- [4] Studentski satelit FERSAT: opis projekta. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu. URL <https://www.fer.unizg.hr/zkist/FERSAT/projekt>. Pristupljeno: lipanj 2021.
- [5] ROM, EPROM, and EEPROM technology. Integrated Circuit Engineering Corporation, 1996.
- [6] Using the Gnu Compiler Collection: An Inline Function is As Fast As a Macro. URL <https://gcc.gnu.org/onlinedocs/gcc/Inline.html>. Pristupljeno: lipanj 2021.
- [7] Flash memory technology. Integrated Circuit Engineering Corporation, 1997.
- [8] *uCAM-II serial camera module datasheet*. 4D Systems, 2021.
- [9] Theuwissen A. CCD or CMOS image sensors for consumer digital still photography? *IEEE International Symposium on VLSI Technology, Systems and Applications*, 2001.
- [10] *The FreeRTOS Reference Manual*. Amazon.com, inc., 2017.
- [11] *Arducam 5MP Mini Plus OV5642 camera module user guide*. Arducam, 2016.
- [12] *Arducam 5MP Mini Plus OV5642 camera module hardware application note*. Arducam, 2016.

- [13] *Cortex-M4 Devices Generic User Guide*. ARM, 2010.
- [14] Arpacı-Dusseau R. Arpacı-Dusseau A. Operating Systems: Three Easy Pieces. Chapter 42: FSCK and Journaling. Arpacı-Dusseau Books , 2018.
- [15] *Application note: Comparison of Basler BCON for LVDS and Camera Link Interfaces*. Basler AG, 2018.
- [16] *ADS1286 12-bit micro power sampling analog-to-digital converter datasheet*. Burr-Brown corporation., 1996.
- [17] Eltoukhy H. El Gamal A. CMOS Image Sensors. *IEEE Circuits and Devices Magazine*, May/June 2005.
- [18] Fujimori I. CMOS passive pixel imager design techniques. Massachusetts Institute of Technology, 1997.
- [19] Tutavac J. Spectral unmixing of incoherent light for the analysis of light pollution. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 2020.
- [20] Dong K. On-chip dead pixel correction in a CMOS imaging sensor. Omniprecision Technologies inc. US7522200B2, 1998.
- [21] Jelenković L. Operacijski sustavi: interni materijal za predavanja. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 2020.
- [22] *Technical note: Program/Erase Cycling Endurance and Data Retention of Macronix SLC NAND Flash Memories*. Macronix International Co. Ltd., 2014.
- [23] *Technical note 29-42: Wear Leveling Techniques in NAND Flash Devices*. Micron Technology inc., 2008.
- [24] *UM10204: I2C-bus specification and user manual*. NXP Semiconductors N.V., 2014.
- [25] *Serial Camera Control Bus functional specification*. Omniprecision Technologies inc., 2002.
- [26] *OV7670/OV7171 CMOS sensor datasheet*. Omniprecision Technologies inc., 2006.
- [27] *OV5642 CMOS QSXGA image sensor datasheet*. Omniprecision Technologies inc., 2009.

- [28] *OV5642 Camera Module Software Application Notes, rev. 1.10.* Omnivision Technologies inc., 2010.
- [29] *Technical note: Rolling Shutter vs. Global Shutter.* QImaging, 2014.
- [30] Barry R. *Mastering the FreeRTOS Real Time Kernel.* Real Time Engineers Ltd., 2016.
- [31] *Reference manual RM0090: STM32F405xx/07xx (...) advanced ARM-based 32-bit MCUs.* STMicroelectronics, 2014.
- [32] *Application note AN5020: Digital Camera Interface (DCMI) for STM32 MCUs.* STMicroelectronics, 2017.
- [33] *User manual UM1725: Description of STM32F4 HAL and low-layer drivers.* STMicroelectronics, 2020.
- [34] *Application note AN5543: Enhanced methods to handle SPI communication on STM32 devices.* STMicroelectronics, 2020.
- [35] Petković T. Upute za treću laboratorijsku vježbu iz Obrade informacija. Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, 2018.
- [36] *Draft Standard for Information Technology - Portable Operating System Interface (POSIX).* The Open Group, 2007.
- [37] *NAND Flash Applications Design Guide.* Toshiba America Electronic Components inc., 2004.
- [38] *Application note AN0011: Flash data retention.* Viking Technology, 2017.
- [39] *Winbond W25N01GVxxIG/IT SLC NAND Flash Memory user guide.* Winbond Electronics Corporation, 2018.
- [40] *Application note AN0000044: How to Program QspiNAND Flash.* Winbond Electronics Corporation, 2020.

Programska potpora ugradbenog računalnog sustava za udaljeno prikupljanje fotografija putem satelita

Sažetak

U ovom radu razvijena je programska potpora za rad s ugradbenim računalnom sustavom na satelitu. Cjelokupni sustav sastoji se od mikrokontrolera, vanjske Flash memorije, kamere, modula s fotosenzorima i odašiljača u X-pojasu. Omogućeno je prikupljanje slika s kamere u RAW i JPEG formatu uz ručno upravljanje trajanjem ekspozicije. Omogućena je akvizicija podataka s više kanala modula s fotosenzorima uz uzorkovanje stabilnom frekvencijom. Omogućena je pohrana prikupljenih podataka na Flash memoriju, za koju je razvijen datotečni sustav s podrškom za ujednačenje trošenja blokova (*wear leveling*), upravljanje neispravnim blokovima (*bad block management*) i sa sustavom za očuvanje konzistencije u slučaju iznenadnog prekida izvođenja (*journaling*). Prema odašiljaču za X-pojas ostvareno je paralelno sinkrono sučelje za prijenos podataka. Programska potpora razvijena je modularno, koristeći operacijski sustav za rad u stvarnom vremenu FreeRTOS. Ostvareno je i povezivanje sustava s osobnim računalom, putem kojeg je moguće upravljati radom sustava i preuzimati prikupljene podatke.

Ključne riječi: ugradbeni računalni sustavi, sustavi za rad u stvarnom vremenu, satelit, FERSAT, upravljanje kamerom, Flash memorija, datotečni sustav, FreeRTOS

Software for Embedded System for Remote Image Acquisition via Satellite

Abstract

In this thesis, software for an embedded system to be used on a satellite is presented. The system consists of the following components: microcontroller, external Flash memory, camera module, photosensor module and X-band transmitter. The following features were developed: image capture via camera, in both RAW and JPEG format, with manual exposure time control; data acquisition from multiple channels on the photosensor board, with a stable sampling frequency; data storage in Flash memory, for which a filesystem, supporting wear leveling, bad block management and journaling, was developed. Additionally, support for X-band module interfacing was provided via a parallel synchronous interface. All software modules were then integrated using an open source real-time operating system FreeRTOS. Finally, interfacing the system with a personal computer was demonstrated. The computer can be used to issue commands and retrieve data from the system.

Keywords: embedded systems, real-time systems, satellite, cubesat, FERSAT, camera control, Flash memory, filesystem, FreeRTOS