SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2113

REALIZACIJA VIDEO PORTAFONA KORIŠTENJEM UGRADBENOG RAČUNALA

Marko Vraničar

Zagreb, lipanj 2011.

SADRŽAJ

1. Uvod

Video portafoni su uređaji koji se najčešće ugrađuju u kućanstvima i stambenim objektima na samom ulazu. Sadrže jedan ili više fotoosjetljivih senzora odnosno kamera koje su usmjerene od samog ulaza u objekt prema potencijalnom korisniku koji ulazi u objekt. Za razliku od video nadzornih sustava kamera je većinu vremena isključena te se uključuje na korisnikov poziv. Korisnik svjesno daje naredbu video portafonu za početak rada kako bi informirao drugu stranu unutar objekta o svom dolasku. Nakon uključenja kamere slika korisnika uhvaćena senzorom prikazuje se drugoj strani na zaslonu.

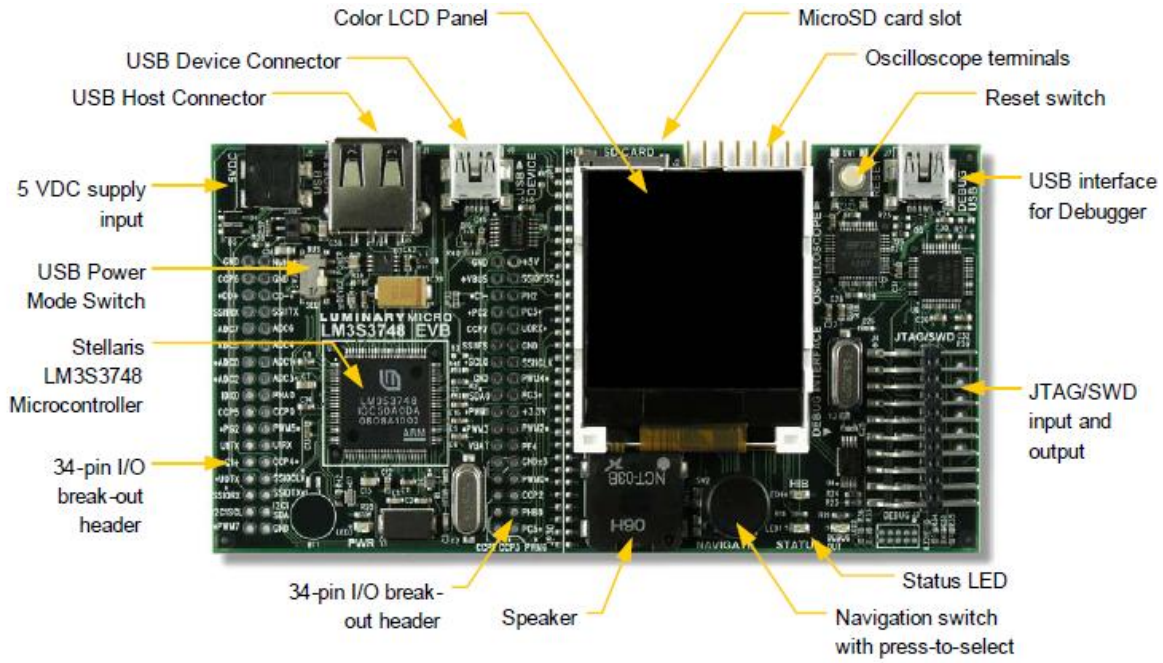
Vide-portafoni se najčešće sastoje od 2 osnovna dijela, unutarnjeg i vanjskog. Vanjski dio se ugrađuje izvan objekta kod ulaza. Ima ugrađenu kameru ili koristi eksternu kameru i njome upravlja.. Na zahtjev korisnika kamera se uključuje i moguće je čitanje slike. Unutarnji dio portafona se ugrađuje unutar objekta i služi za prikaz slike uhvaćene kamerom. Povezivanje dijelova može biti žičano ili bežično. U našem slučaju kao vanjski dio koristimo eksternu kameru i razvojni sustav koji njome upravlja, a računalo ima ulogu unutarnjeg dijela koji se nalazi unutar objekta i prikazuje sliku na zaslonu. Povezivanje se izvodi žičano ili preko drugog razvojnog sustava za bežični prijenos. Najčešće je moguće obostrano upravljanje portafonom, sa korisnikove strane i strane unutar objekta, pošto korisnik treba na neki način informirati drugu stranu o svom dolasku.

Glavna značajka video portafona je prijenos i prikaz slike. Namijenjeni su za rad u stvarnom vremenu, dakle slika se mora u relativno brzom vremenskom intervalu prikazati na zaslonu. Najčešće se radi o sekvenci slika ili video prijenosu koji se prikazuje, što uvjetuje brzim prijenosom slike. Nije naglasak na kvaliteti slike, bitno je samo da se može prepoznati osoba. Detalji na slici ne igraju značajnu ulogu. Veće ograničenje predstavlja konačna brzina prijenosa i konačno vrijeme potrebno za dohvat i prilagodbu slike.

2. Razvojni sustav sa LM3S3748 mikrokontrolerom

Razvojni sustavi služe za simulaciju rada mikrokontrolera i razvoj aplikacija. Koriste se kako bi se u praksi ustanovilo da li sklop sa mikrokontrolerom zadovoljava očekivanja korisnika, te da se otkriju eventualne greške koje nisu bile otkrivene tijekom simuliranja programa na računalu.

Stellaris LM3S3748 razvojni sustav (EVB) je kompaktna i multifunkcionalna platforma za testiranje Stellaris LM3S3748 ARM mikrokontrolera Cortex-M3 porodice. Omogućava lakši razvoj aplikacija, mogu se testirati ključne značajke LM3S3748 mikrokontrolera, uključujući USB 2.0 kontroler koji podržava brzinu prijenosa od 12 Mbps, analogno-digitalne pretvarače (ADC), serijsko sučelje, podršku za SD karticu. Programska podrška uključuje i mogućnost za rad sustava kao osciloskop, što u ovom radu nije razmatrano.

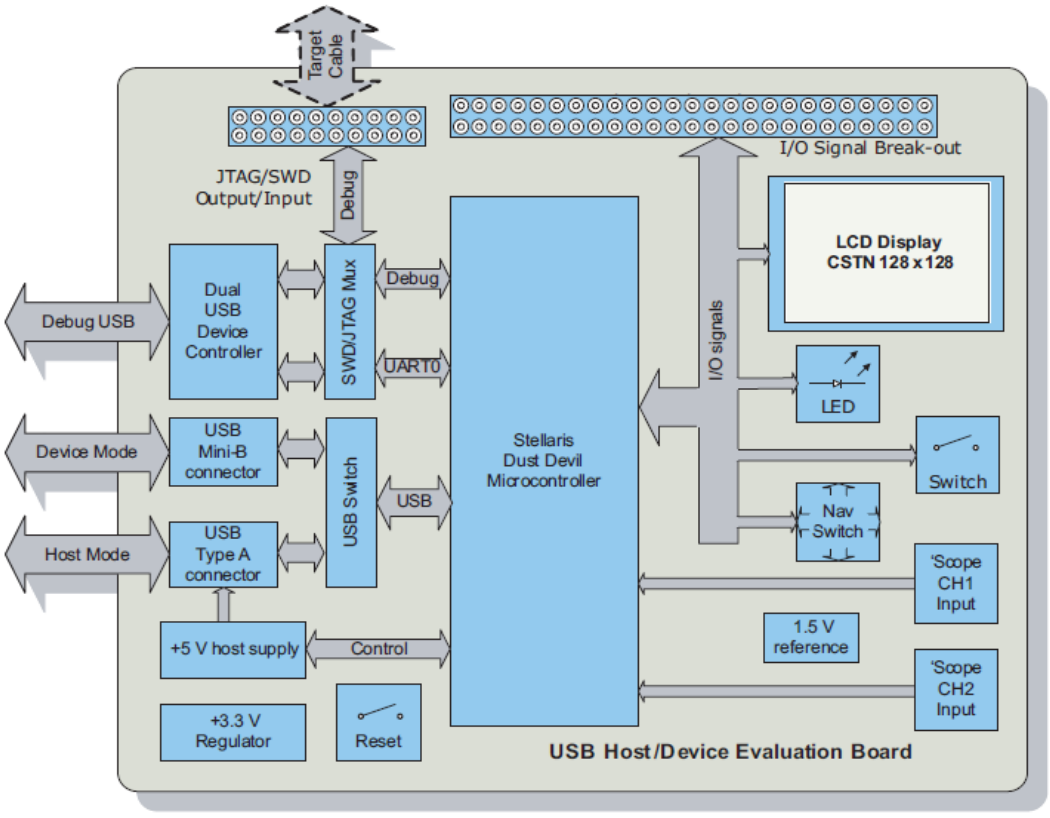


Slika 1. *Razvojni sustav Stellaris LM3S3748*

(LM3S3748 Evaluation Kit User's Manual, 2010)

LM3S3748 EVB sadrži slijedeće značajke:

1. Stellaris LM3S3748 mikrokontroler
2. USB Host i Device priključci
3. BUS-POWERED ili SELF_POWERED način rada
4. USB-om je osigurano napajanje, omogućava serijsku komunikaciju i debugiranje
5. LCD display u boji razlučivosti 128 x 128 piksela
6. LED diode i navigacijski prekidač
7. 8Ω speaker sa pojačalom
8. Slot za microSD karticu
9. USB sučelje za debugiranje i napajanje
10. DC priključak za 5 V napajanje
11. Standardni ARM 20-pinski JTAG priključak za debugiranje
12. I/O priključci



Slika 2. *Blok shema razvojnog sustava Stellaris LM3S3748*

(LM3S3748 Evaluation Kit User's Manual, 2010)

LM3S3748 EVB podržava oba USB načina rada: USB Host i USB Device, što omogućava široku primjenu i komunikaciju sustava sa raznim vanjskim jedinicama preko USB sučelja (npr. miš, tipkovnica, USB memory stick, web-kamera itd.).

Značajke mikrokontrolera LM3S3748:

- 32-bitni RISC procesor ARM Cortex-M3 v7M arhitekture
- 50 MHz takt
- Hardware-sko dijeljenje i jednociklusno množenje
- Integrirani Nested Vectored Interrupt Controller (NVIC)
- 37 izvora prekida sa 8 razina prioriteta
- 128 KB single-cycle flash
- 64 KB single-cycle SRAM
- Pre-programibilni ROM
- DMA kontroler
- 2 SSI modula
- USB Host kontroler
- 4 općenamjenska 32-bitna brojila
- 2 potpuno programibilna UART-a 16C550 tipa
- Osam 10-bitna ADC
- 2 integrirana analogna komparatora
- Dva I2C modula
- 4 PWM generatora
- QEI modul
- 3 do 61 GPIO, ovisno o konfiguraciji
- On-chip low drop-out (LDO) regulator napona
- Hibernation modul

U daljnjem tekstu će biti razmatrani samo dijelovi sustava koji se koriste u sklopu završnog rada.

2.1. USB

LM3S3748 sadrži full-speed USB kontroler koji podržava Host i Device način rada. U Host modu, sustav predstavlja host kontroler (master) kod komunikacije sa spojenim USB uređajima. To znači da on pokreće i vodi sve transakcije, odnosno prijenos podataka. U Device modu, sustav djeluje kao uređaj (slave) koji može biti povezan s drugim USB Host uređajem, kao što je računalo. Kod USB komunikacije uvijek postoji samo jedan Host i on pokreće i vodi komunikaciju. Device uređaja može biti više. Kod odabira načina rada koristi se multipleksor upravljan preko GPIO pina (PH2/PB0).

Tablica 1. *Raspored USB signala po pinovima*

(LM3S3748 Evaluation Kit User's Manual, 2010)

Microcontroller Pin	EVB Function	To isolate, remove...
Pin 70 USBDM	USB Data-	-
Pin 71 USBDP	USB Data+	-
Pin 73 USBRBIAS	USB bias resistor	-
Pin 66 PB0	Input (see Rev A0 errata)	-
Pin 67 PB1	Input (see Rev A0 errata)	-
Pin 84 PH2	USB Host/Device mux control	JP33
Pin 83 PH3/USB0EPEN	Host power enable (active high)	-
Pin 76 PH4/$\overline{\text{USB0PFLT}}$	Host power fault (active low)	-

Ako priključeni USB uređaj povlači struju veću od 1A ili ako je prekoračena prekidačeva termička granica uslijed priključenja uređaja koji vuče struju veću od 500 mA, prekidač prekida struju prema dotičnom uređaju. USB kontroler može biti konfiguriran za generiranje prekida ako je USB0PFLT_ aktivan. Kod USB Device načina rada, malim prekidačem se odabire između BUS-POWERED i SELF_POWERED načina napajanja, što znači da je sustav moguće napajati preko USB porta ili preko posebnog priključka za napajanje.

2.2. LCD

Razvojni sustav sadrži modul sa LCD display-em (*liquid crystal graphics display*) veličine 128 x 128 piksela. Moguće je na LCD-u prikazati maksimalno 65536 različitih boja (16-bita po boji). Modul sa LCD zaslonom u boji ima ugrađen kontroler sa 8-bitnim paralelnim sučeljem. GPIO port G se koristi za prijenos podataka prema i sa LCD modula. Pri tome se koriste 3 kontrolna signala (A0, WR_, RD_) kojima se šalju upravljačke komande za čitanje/pisanje te registar/data kontrolni signali, preko GPIO pinova.

Tablica 2. *Raspored LCD signala po pinovima*

(LM3S3748 Evaluation Kit User's Manual, 2010)

Microcontroller Pin	EVB Function	To isolate, remove...
Pin 19 PG0	LCD Data 0	JP16
Pin 18 PG1	LCD Data 1	JP14
Pin 19 PG2	LCD Data 2	JP17
Pin 16 PG3	LCD Data 3	JP18
Pin 41 PG4	LCD Data 4	JP26
Pin 40 PG5	LCD Data 5	JP27
Pin 37 PG6	LCD Data 6	JP28
Pin 36 PG7	LCD Data 7	JP29
Pin 25 PC4	LCD Write Enable (active low)	JP30
Pin 24 PC5	LCD Read Enable (active low)	JP21
Pin 72 PB2	LCD Register / Data select	JP11
Pin 61 PF1/PWM1	Backlight control	JP19

Kao pozadinsko osvjetljenje koristi se bijeli LED indikator. Da bi osvjetljenje bilo uključeno potrebno je postaviti pin PF1/PWM1 u aktivno stanje. Za regulaciju osvijetljenosti koristite se PWM funkcije. LCD modul sadrži unutarnji generator napona i zahtijeva samo jedno istosmjerno napajanje od 3V.

2.3. Navigacijski prekidač

EVB sadrži navigacijski prekidač sa 4 moguća stanja (smjera) i funkcijom odabira kod pritiska (znači 5 moguća stanja sklopke). Svaki od signala spojen je na GPIO pinove LM3S3748 mikrokontrolera. Prije čitanja stanja prekidača, programski treba postaviti interne pull-up otpornike od 200 kΩ pojedinih pinova.

Tablica 3. *Raspored signala navigacijskog prekidača po pinovima*

(LM3S3748 Evaluation Kit User's Manual, 2010)

Microcontroller Pin	EVB Function	To isolate, remove...
Pin 65 PB3	Up Switch	JP13
Pin 92 PB4	Down Switch	JP10
Pin 91 PB5	Left Switch	JP9
Pin 90 PB6	Right Switch	JP23
Pin 89 PB7	Select Switch	D1

2.4. Virtualni COM Port

Virtualni COM Port (VCP) omogućuje Windows aplikacijama (kao što je HyperTerminal) komunikaciju sa razvojnim sustavom preko USB sučelja. Komunikacija se odvija preko UART0 (*Universal asynchronous receiver/transmitter*) porta na LM3S3748. UART pretvara prijenos podataka iz paralelnog i serijski oblik i obrnotu, te najčešće podržava komunikacijske standarde kao što su RS-232, RS-422 ili RS-485. Riječ *Universal* u nazivu označava da se format podataka i brzine prijenosa mogu programski konfigurirati, a da se stvarna električka svojstva signala, razina i način prijenosa (kao što je diferencijalni prijenos i sl.) rješavaju vanjskim sklopovljem (*driver*-ima) spojenim na UART. Nakon instalacije FT2232 VCP drivera, Windows dodjeljuje VCP određen broj COM port-a. U sklopu završnog rada se koriste UART0 i UART1 portovi.

Tablica 4. *Raspored UART signala po pinovima*

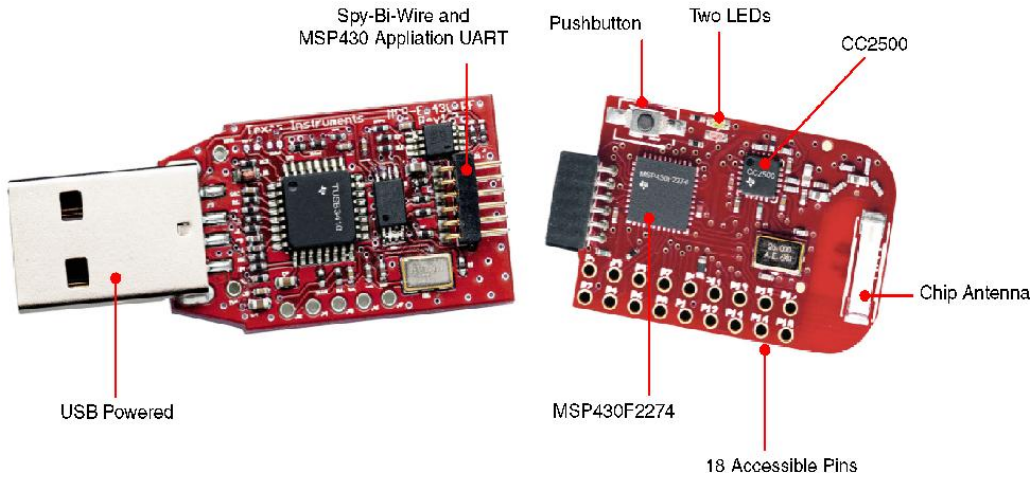
(LM3S3748 Evaluation Kit User's Manual, 2010)

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]
UART	U0Rx	26	I	TTL
	U0Tx	27	O	TTL
	U1Rx	23	I	TTL
	U1Tx	22	O	TTL

Električke sheme spajanja pojedinih komponenti dane su na kraju dokumentacije.

3. EZ430-RF2500 razvojni sustav

EZ430-RF2500 je kompletan razvojni sustav temeljen na MSP430 mikrokontroleru namijenjen za razvoj aplikacija za bežični (*wireless*) prijenos podataka između dva nezavisna primopredajnika. Paket uključuje 2 nezavisna uređaja sa MSP430F2274 mikrokontrolerom i CC2500 2.4-GHz wireless primopredajnikom (*transceiver*), eZ430-RF USB sučelje za debugiranje, te modul za baterijsko napajanje.



Slika 3. *eZ430-RF USB sučelje za debugiranje (gore lijevo)*

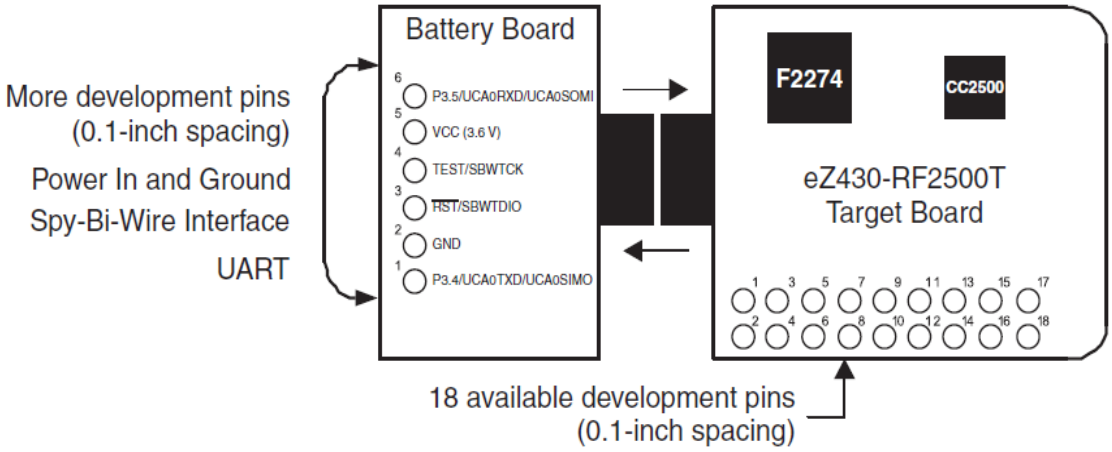
eZ430-RF2500T modul sa MSP430 mikrokontrolerom (gore desno)

(eZ430-RF2500 Development Tool User's Guide, 2009)

Slika 4. *Modul za baterijsko napajanje*

(eZ430-RF2500 Development Tool User's Guide, 2009)

EZ430-RF2500 se može koristiti kao samostalni razvojni sustav. Modul sa MSP430F2274 mikrokontrolerom moguće je odvojiti od sučelja za debugiranje te ga integrirati u vlastiti dizajn. Pojedinim izlaznim pinovima se lako pristupa nakon skidanja plastične zaštite. Modul za baterijsko napajanje služi isključivo kao izvor napajanja odvojenog modula.



Slika 5. *Pristup pinovima na modulu sa MSP430F2274 mikrokontrolerom*

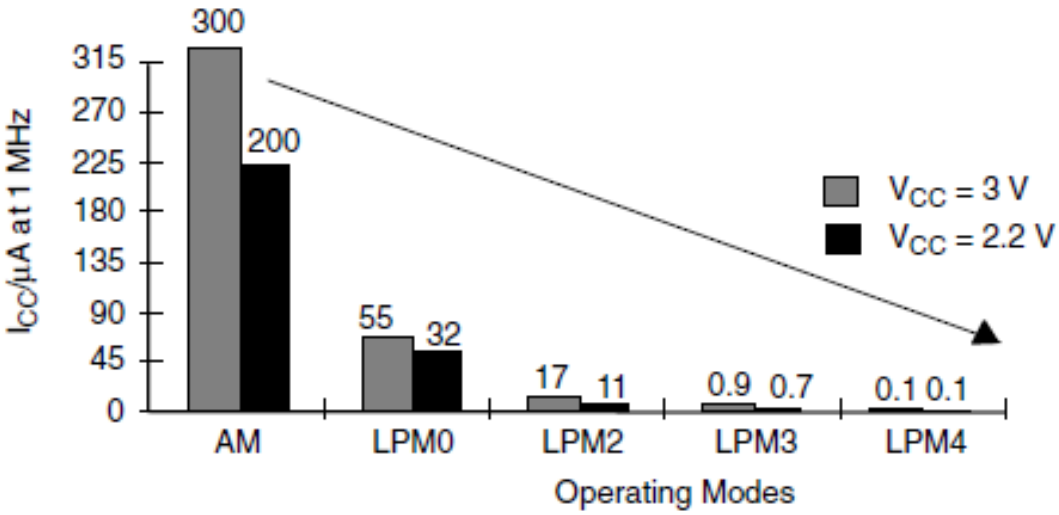
(eZ430-RF2500 Development Tool User's Guide, 2009)

EZ430-RF2500 sadrži slijedeće značajke:

1. USB sučelje za programiranje i debugiranje
2. 21 lako dostupni izlazni pin
3. Visoko integrirani MSP430 MCU male snage (ultra-low-power) s maksimalnim taktom od 16 MHz
4. CC2500 RF primopredajnik
5. 2 GPIO pina spojena na zelenu i crvenu LED diodu radi vizualne povratne informacije
6. Prekidni taster za povratnu informaciju od korisnika

3.1. MSP430 porodica

MSP430 je mixed-signal mikrokontroler, pripada 16-bitnoj porodici RISC mikrokontrolera tvrtke Texas Instruments koja je specifična po vrlo maloj potrošnji, te stoga namijenjena prijenosnim uređajima koji koriste baterijsko napajanje. Organizacija mikrokontrolera je podređena niskoj potrošnji pa tako podržava 4 načina rada smanjene



Slika 6. *Tipična potrošnja struje kod 1MHz takta*

(MSP430x2xx Family User's Guide, 2008)

potrošnje (low-power) i brzi Wake-Up u trajanju od 6µs. MSP430 troši manje od 300µA u aktivnom načinu rada pri taktu od 1MHz i naponu napajanja od 3V. Potrošnja u načinu rada smanjene potrošnje s aktivnom detekcijom vanjskog prekida i aktivnim *WatchDog* timerom iznosi 1µA.

Značajke mikrokontrolera MSP430F2274:

- 16-bitni RISC arhitektura
- Niski napon napajanja 1.8 V do 3.6 V
- Smanjena potrošnja:
    - Aktivni način rada: 270 µA na 1 MHz, 2,2 V

    - Standby Mod: 0.7 μA
    - Off Mode: 0.1 μA
- Brzo vraćanje iz Standby Moda (Wake-Up), manje od 1μs
- Von Neumann organizacija memorije
- PLL i digitalno upravljivi oscilator do 16 MHz
- 32 KB + 256 B Flash Memorija
- 1 KB RAM
- Dva 16-bitna brojila sa sklopovljem za hvatanje i usporedbu (Capture/Compare)
- Univerzalno serijsko komunikacijsko sučelje:
    - UART
    - IrDA
    - SPI
    - I2C
- 10-bitni ADC
- 2 operacijska pojačala

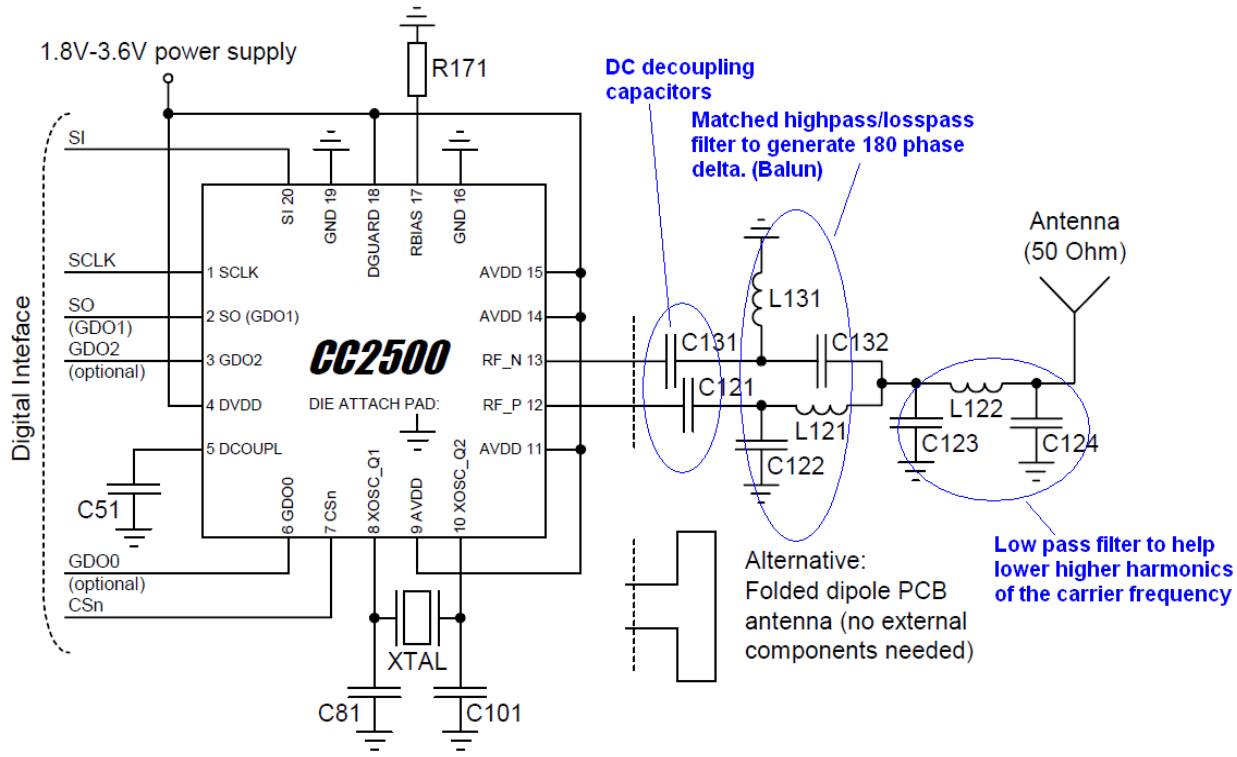
3.2. CC2500 RF primopredajnik

CC2500 je nisko-budžetni bežični (wireless) primopredajnik koji radi u području od 2.4 GHz. Koristi se kod wireless aplikacija koje zahtijevaju jako malu potrošnju. Namijenjen je za rad u frekvencijskom području od 2400 do 2483.5 MHz (ISM - *Industrial, Scientific and Medical*) i kratkog je dometa (SRD - *Short Range Device*). Radio-frekvencijski primopredajnik (*RF transceiver*) sadrži integrirani konfigurabilni modem osnovnog pojasa. Modem podržava različite modulacijske postupke te se protok podataka (*data rate*) može podešavati do 500 kBaud (*data rate* u kbps je polovica *baud rate*-a). Podržani modulacijski postupci su OOK (ASK), 2-FSK, GFSK i MSK.

CC2500 pruža opsežnu hardversku podršku za rukovanje paketima, međupohranu podataka (*buffering*), burst prijenos, procjenu dostupnosti kanala, indikaciju kvalitete veze i *Wake-on-Radio* podršku. *Wake-on-Radio* podrška je hardware-ski implementirana verzija radio-komunikacije sa niskom potrošnjom. Omogućuje da mikrokontroler u potpunosti uđe u

'*sleep*' mod, a da se takt dovodi samo na unutarnje sklopovlje radio-primopredajnika. Time se znatno smanjuje potrošnja.

Osnovnim operacijskim parametrima i 64-bajtnim transmit/receive FIFO moguće je upravljati preko SPI sučelja. Uobičajeno se CC2500 koristi zajedno s mikrokontrolerom (u našem slučaju MSP430) koji njime upravlja i nekoliko dodatnih pasivnih komponenti.



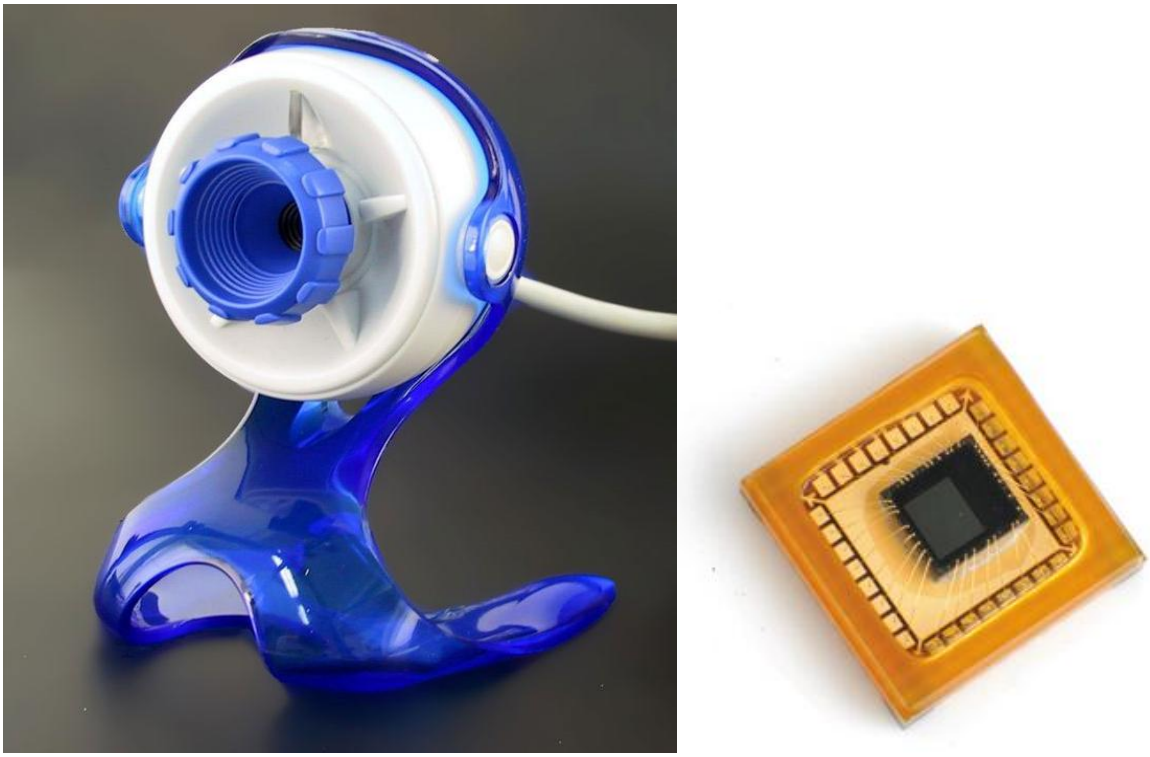
Slika 7. *Tipična shema spajanja CC2500 RF primopredajnika*

(CC2500 DATA SHEET, 2009)

Detaljnije elektroničke sheme spajanja pojedinih komponenti dane su na kraju dokumentacije.

4. Kamera

Koristimo web-kameru tvrtke *Chicony* model DC-4110. Zbog svojeg jednostavnog dizajna njome je moguće lako upravljati preko USB sučelja. Osnovni dio kamere čini optički CMOS senzor tvrtke *PixArt Imaging* model PAC207B. Senzor ima ugrađeno USB sučelje i podržava kompresiju slike. Pogodan je za realizaciju niskobudžetnih kamera najčešće korištenih kod raznih aplikacija koje distribuiraju sliku putem interneta.



Slika 8. *Web-kamera Chicony DC-4110 (lijevo) i optički senzor PAC207B (desno)*

(preuzeto sa: http://cokupic.pl/produkt/Chicony-iCam-140-DC-4110)

(preuzeto sa: http://ab-log.ru/smart-house/video_camera_security/i-look_111_linux)

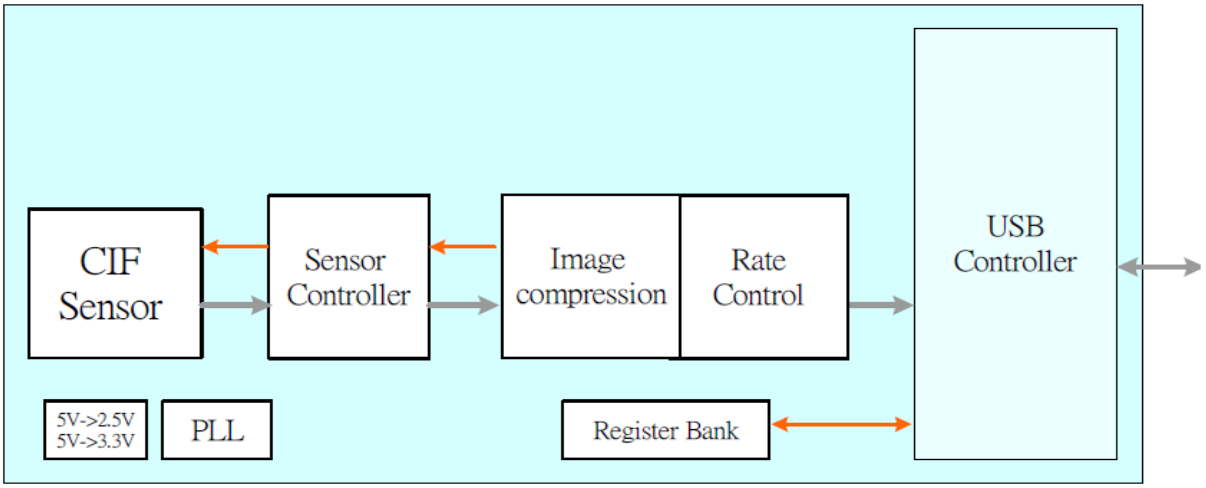
Osnovne značajke PAC207B senzora:

- 5.6µm x 5.6µm CIF CMOS senzor, realiziran 0.25µm-skom tehnologijom
- Potrebno dodati kristal 12MHz ili 6MHz za generiranje interne frekvencije takta od 48MHz

- Napajanje: +5V
- Potrošnja: < 25mA
- *Optical format*: 1/7“
- S/N (signal/šum) odnos: 35dB
- Maksimalna razlučivost slike 352 x 288 piksela
- Slika u RGB Bayer piksel rasteru
- Maksimalna brzina (*frame rate*): 30fps
- Minimalno trajanje ekspozicije (*frame time*) ~ 1/160000 sec
- Programiranje senzora se vrši preko USB *control pipe*-a
- Transfer slike preko USB *isochronous pipe*-a ili USB *bulk pipe*-a
- Za prekidni način rada korišten USB *interrupt pipe*
- Pakiranje: 32-pinski LCC

Kamera podržava 4 osnovna načina rada:

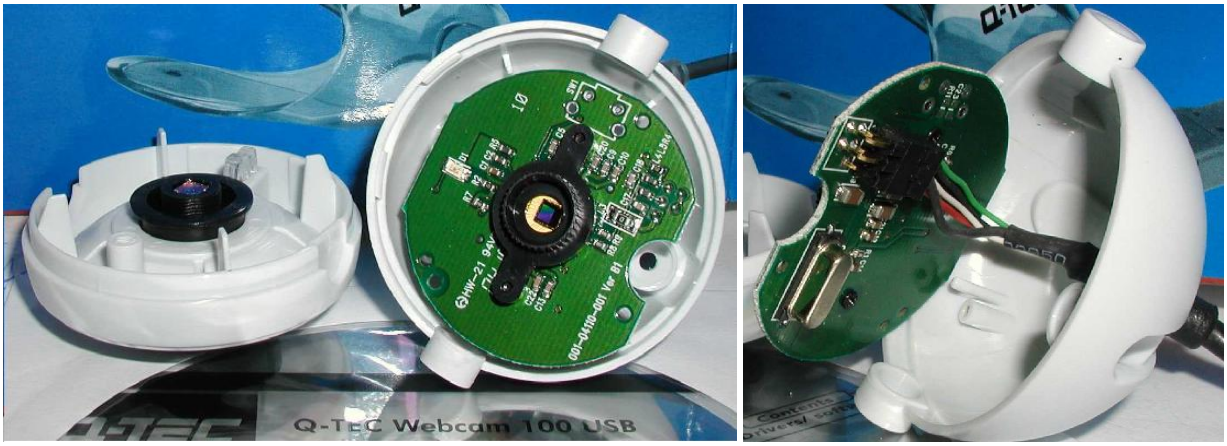
- do 24fps, CIF format (352x288) sa kompresijom, 8 Mbps USB propusnost
- do 24fps, QVGA format (320x240) sa kompresijom, 8 Mbps USB propusnost
- do 30fps, QCIF format (176x144) bez kompresije, 5.6 ~ 6 Mbps USB propusnost
- do 30fps, QQVGA format (160x120) bez kompresije, 5.6 ~ 6 Mbps USB propusnost (*bandwith*)



Slika 9. *Blok dijagram optičkog senzora PAC207B*

(PAC207B DATA SHEET, 2003)

PAC207B CIF senzor je temeljen na Pixart PAS106 senzoru s poboljšanom kvalitetom slike i osjetljivošću. Sadrži unutarnje regulatore, podržava kompresiju i internu obradu slike. Parametri se podešavaju upisom u kontrolne registre senzora. Ima ugrađen SRAM koji služi kao međuspremnik podataka i ugrađen USB kontroler. Svi registri se postavljaju putem USB sučelja. Podržava spajanje vanjskog serijskog EEPROM (93C46A 16bit) za pohranu podataka poput *Vendor ID*, *Product ID* i parametara senzora.



Slika 10. *Unutrašnjost web-kamera Chicony DC-4110, lijevo prednja strana (front), desno stražnja strana (back)*

(preuzeto sa: http://gkall.hobby.nl/pac20x.html)

Pošto kod video portafona nije potrebna velika razlučivost i kvaliteta slike parametre senzora podešavamo na razlučivost slike 176x144 (RGB Bayer piksel raster) bez kompresije. Shema spajanja senzora i realizacija kamere su dane na kraju dokumentacije.

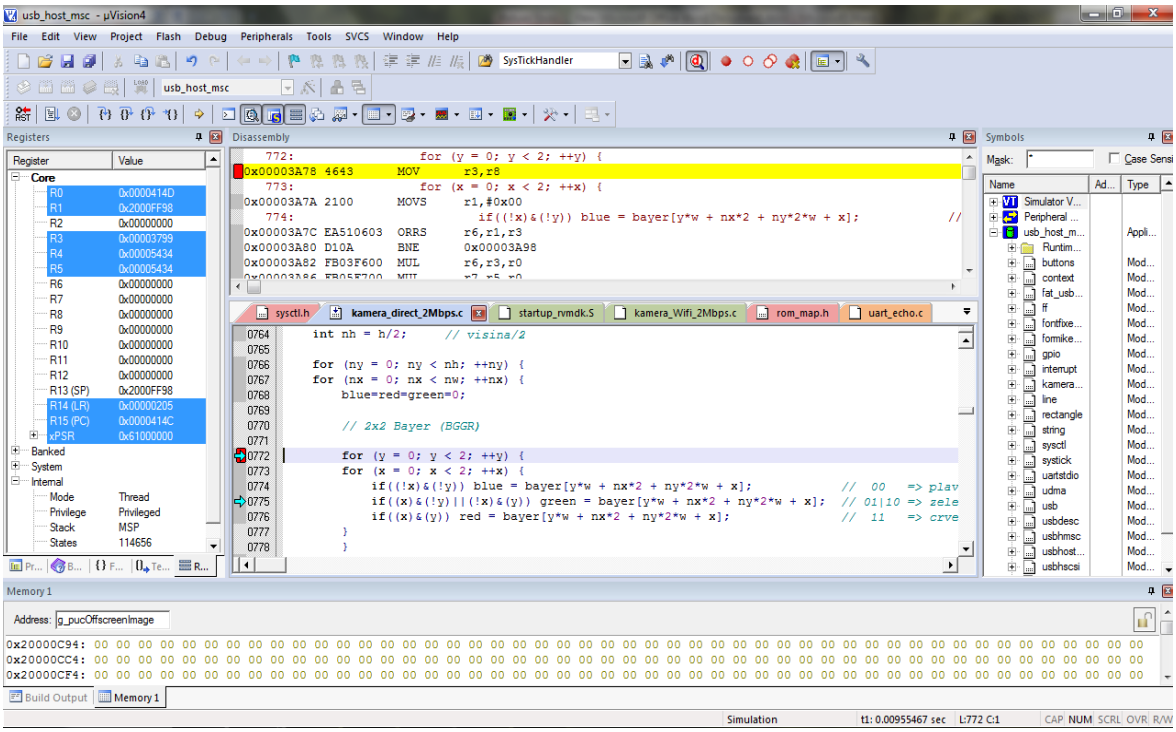
5. Programska podrška

U okviru završnog rada potrebno je odvojeno programirati 2 razvojna sustava LM3S3748 EVB i eZ430-RF2500. Koristimo 2 različita razvojna okruženja pošto pojedine aplikacije ne podržavaju različite arhitekture i porodice mikrokontrolera. Korištena razvojna okruženja su Keil µVision za programiranje LM3S3748 ARM mikrokontrolera i IAR Embedded Workbench za programiranje MSP430F2274 mikrokontrolera.

5.1. Keil µVision for ARM

Keil µVision ARM je razvojno okruženje koje sadrži skup alata za razvoj aplikacija konkretno za ARM mikrokontrolere. Keil je jedna od vodećih svjetskih tvrtki u području programiranja mikrokontrolera pa je tako programski paket µVision moguće naći u različitim inačicama za različite tipove mikrokontrolera. ARM procesori su danas dominantni te se ARM arhitektura koristi u približno 75% svih 32 bitnih RISC procesora, što je svrstava u najrašireniju vrstu 32 bitnih arhitektura.

µVision razvojna platforma je vrlo jednostavna za korištenje što nam omogućuje vrlo brzu izradu programa. Sastoji se od nekoliko važnijih dijelova kao što su: editor izvornog koda (*source code editor*), program za ispravljanje pogrešaka (*debugger*) te simulatora u vrlo moćnom okruženju. Učinkovit prevodilac (*compiler*) podržava programske jezike C /C++ te se prevođenjem postiže efikasnost i brzina izvršavanja programa kao kod izvorno pisanog koda u assembleru.

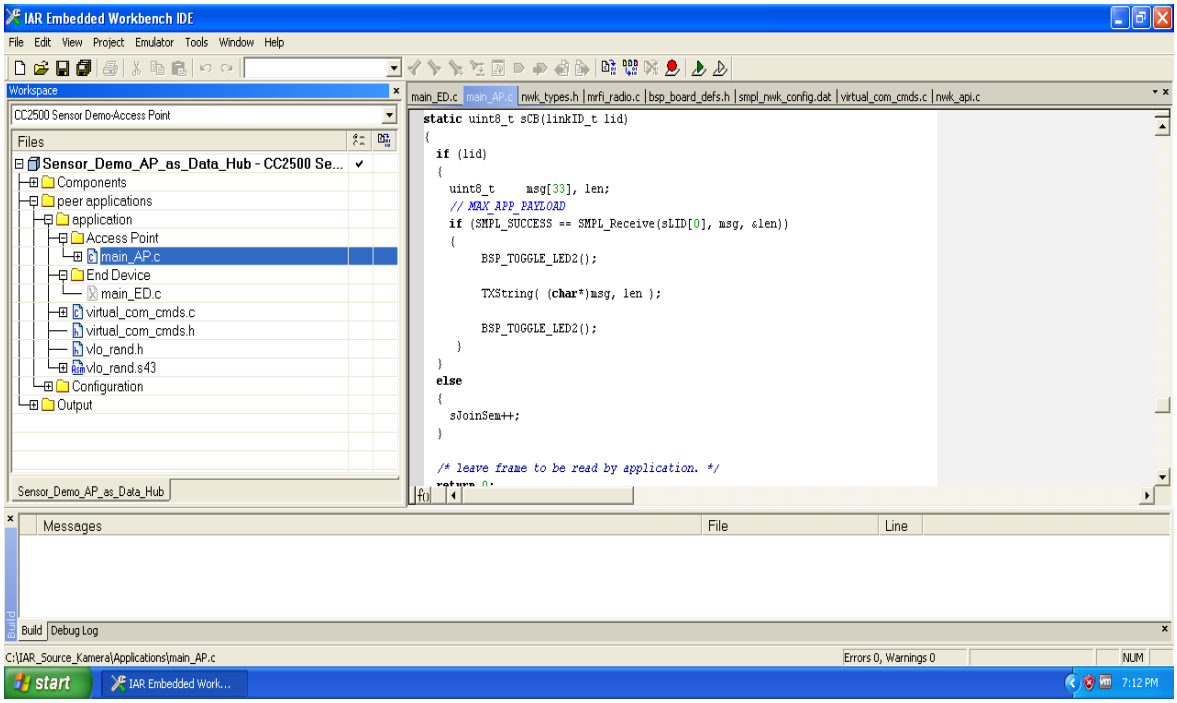


Slika 11. *Keil µVision korisničko sučelje*

Cortex-R4, ARM7 i ARM9 arhitekture. LM3S3748 spada u Cortex-M3 porodicu te je podržan od strane razvojnog okruženja.

5.2. IAR Embedded Workbench for MSP430

Od programske podrške u sklopu završnog rada se koristi Programski paket IAR Embedded Workbench za MSP430 mikrokontrolere. IAR je snažno i učinkovito okruženje za razvoj aplikacija za MSP430 mikrokontrolere u jeziku C /C++ i assembleru. Razvojno okruženje uključuje prevodilac, linker i debugger (simulator). Dodatni alati i dobar ugrađeni sustav pomoći (help) znatno olakšavaju pisanje programa. Podržava široku paletu MSP430 uređaja te generira kompaktan i učinkovit kod.



Slika 12. *IAR Embedded Workbench korisničko sučelje*

Podržane porodice MSP430 procesora:

- CC430(CC430F5133, CC430F5135…)
- G2xx (MSP430G2001, MSP430G2101…)
- 1xx (MSP430C111, MSP430C112…)
- 2xx (MSP430AFE253, MSP430F2274…)
- 3xx (MSP430C311S, MSP430C312…)
- 4xx (MSP430C412, MSP430C413…)
- 5xx (MSP430BT5190, MSP430F5304…)
- 6xx (MSP430F6630, MSP430F6631…)

Nas zanima samo 2xx porodica pošto razvojni sustav koristi navedeni tip procesora MSP430F2274.

6. Rad sa razvojnim sustavima

LM3S3748 razvojni sustav je odabran iz razloga što podržava USB Host način rada. To nam odgovara jer je time moguće čitati podatke sa različitih uređaja namijenjenih za rad u USB Device načinu rada, kao što je u našem slučaju USB web-kamera. Potrebno je napisati program koji preko USB Host sučelja učitava sliku sa web-kamere i prikazuje sliku na LCD display-u. Nakon toga učitanu sliku ovisno o odabranom načinu prijenosa šaljemo direktno ili bežično (*wireless*) na računalo koje preko odgovarajuće aplikacije prikazuje sliku na zaslonu.

6.1. Inicijalizacija periferija LM3S3748 mikrokontrolera

Prije svega potrebno je inicijalizirati odgovarajuće periferije LM3S3748 mikrokontrolera koje se koriste. U slučaju da se slika šalje direktno potrebno je inicijalizirati UART0 koji predstavlja virtualni COM port. Učitana slika je malih dimenzija i veličine tako da serijska RS-232 komunikacija predstavlja dovoljno brz način prijenosa. Brzina kojom se podaci prenose je neuobičajena za RS-232 komunikaciju i iznosi 2 Mbps. Odabrana brzina je dovoljno velika da se slika veličine 19 KB prenese na računalo u 77.8 ms. Mikrokontroler LM3S3748 i vanjsko sklopovlje na razvojnom sustavu podržavaju proizvoljne brzine rada serije pa tako i 2 Mbps. Ako se slika šalje bežično, vezu između LM3S3748 razvojnog sustava i eZ430-RF2500T modula za bežični prijenos ostvarujemo preko UART1 porta, također pri brzini od 2 Mbps. Mikrokontroler MSP430F2274 koji se nalazi na modulu podržava proizvoljne brzine prijenosa serijske RS-232 veze.

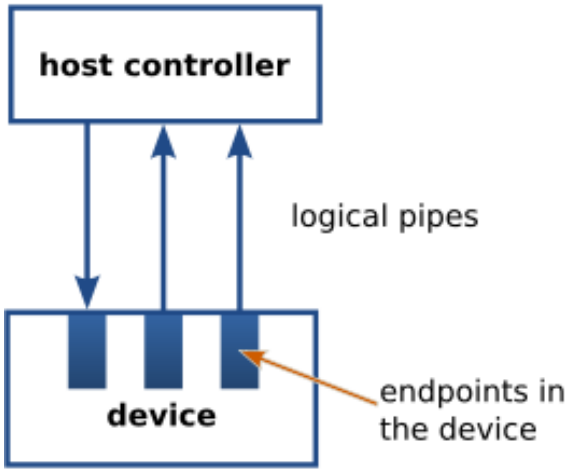
Nakon inicijalizacije serije podešavamo LM3S3748 za rad u USB Host načinu rada. Osim konfiguracije USB Host moda potrebno je inicijalizirati mikroDMA kontroler za brz prijenos podataka preko USB porta. Isto tako je potrebno inicijalizirati LCD display za prikaz podataka i konfigurirati tipke čije stanje se provjerava u prekidnoj rutini. Prekid izaziva brojilo svakih 10 ms.

6.2. Inicijalizacija kamere

Kamera se inicijalizira upisom odgovarajućih predefiniranih vrijednosti u niz registara PAC207B optičkog senzora. Komunikacija između senzora i razvojnog sustava ostvarena je preko USB Host sučelja. Pretpostavljamo da se na sustav uvijek spoja navedeni tip kamere pa nije potrebno dodatno provjeravati radi li se o pravome tipu.

6.2.1. USB komunikacija

USB komunikacija se temelji na *pipe*-ovima koji predstavljaju logičke kanale. *Pipe* čini vezu između Host uređaja i logičkog 'subjekta' (*entity*) nađenog na Device uređaju. Subjekt se još naziva *endpoint*. Uređaj koji radi u USB Device modu može imati do 32 *endpoint*-a (16 ulaznih i 16 izlaznih). *Endpoint*-ovi su implementirani od strane proizvođača uređaja te ih stoga nije moguće izbrisati već su konstantno prisutni. Postoje dvije vrste *pipe*-ova. To su protočni (*stream*) *pipe*-ovi i *pipe*-ovi za slanje kratkih poruka (*message*). *Message pipe* služi isključivo za prijenos kontrolnih podataka te podržava dvosmjerni protok informacija od i prema Host uređaju. Još se naziva *Control transfer pipe.*



Slika 13. *USB komunikacija, pipe-ovi i endpoint-ovi*

(Preuzeto sa: http://en.wikipedia.org/wiki/File:USB_pipes_and_endpoints_%28en%29.svg)

Control transfer pipe – koristi se kod slanja kratkih jednostavnih naredbi prema Device uređaju te je ovisno o naredbi moguć odgovor odnosno odziv Device uređaja u obliku vraćenog podatka ili skupa podataka. Nalazi se na adresi 0, svaki USB Device uređaj ga ima i označava se kao '*endpoint 0*'.

Stream pipe podržava jednosmjerni protok podataka od odnosno prema Host uređaju. U *Stream pipe*-ove ubrajamo:

Isochronous transfer pipe – jamči brz protok podataka, najčešće najbrži mogući, ali uz mogućnost gubitka podataka. Najčešće korišten za prijenos zvuka i slike u stvarnom vremenu.

Interrupt transfer pipe – najčešće korišten kod uređaj kod kojih je potreban brz odziv.

Bulk transfer pipe – korišten kod prijenosa velike količine podataka, no brzina slanja nije zajamčena. Moguća su kašnjenja. Najčešće korišten kod prijenosa datoteka (npr. sa USB memory stick-a).

Osim *Control transfer endpoint-a* PAC207B ima još 6 *endpoint*-ova:

- 2 *Bulk* tipa (ulazni i izlazni)
- 2 *Interrupt* tipa (ulazni i izlazni)
- 2 *Isochronous* tipa (ulazni i izlazni)

Zbog nedostupnosti potrebne dokumentacije optičkog senzora PAC207B te zbog nedostatka primjera izvedbe programske podrške na računalu, za prijenos slike koristimo prvi ulazni (ulazni u odnosu na Host uređaj) *endpoint Bulk* tipa. U *Bulk* transferu Device uređaj šalje podatke u paketima od maksimalno 64 bajta, dok je u *Isochronous* transferu moguće slanje paketa veličine do 1024 bajta. Brzina *Bulk* prijenosa zadovoljava potrebe video portafona tako da nema potrebe za *Isochronous* prijenosom. *Control transfer endpoint-u* (*ednpoint 0*) se pristupa i upravljan je od strane USB drivera koji je već implementiran te ga je moguće referencirati kroz pripadajuće biblioteke Keil-a.

Kod priključenja kamere program prvo traži *Bulk* IN *enpoint* i alocira potreban *pipe* za komunikaciju. Radi lakše kontrole toka programa implementiran je izbornik koji se prikazuje na display-u i bira se način rada. Isječak programskog koda koji alocira potreban *pipe* je dan u nastavku.

```
  for(iIdx = 0; iIdx < 6; iIdx++)   // prolazimo kroz svih 6 endpointo-va
   {
      pEndpointDescriptor = USBDescGetInterfaceEndpoint(pInterface, iIdx, pDevice-
>ulConfigDescriptorSize);

      if(pEndpointDescriptor == 0) break;      // ako više nema endpoint-ova izađi
      //
      // provjeri radi li se o Bulk endpoint-u
      //
          if((pEndpointDescriptor->bmAttributes & USB_EP_ATTR_TYPE_M) ==
          USB_EP_ATTR_BULK)
      {
         if(pEndpointDescriptor->bEndpointAddress & USB_EP_DESC_IN)
         {   // ako je Bulk IN
            //
            // alociraj USB Pipe za Bulk IN endpoint.
            //
            ulBulkInPipe = USBHCDPipeAlloc(0,
                           USBHCD_PIPE_BULK_IN,
                           pDevice->ulAddress,
                           0);
            //
            // konfiguriraj USB pipe kao Bulk IN endpoint.
            //
            USBHCDPipeConfig(ulBulkInPipe,
                            pEndpointDescriptor->wMaxPacketSize,
                            pEndpointDescriptor->bInterval,
                            pEndpointDescriptor->bEndpointAddress &
                            USB_EP_DESC_NUM_M);

          g_eState = STATE_CAMERA_MENU_INIT; // sljedeće stanje, inicijaliziraj izbornik
         }
      }
   }
```

U izborniku se odabire jedan od načina rada, a to su:

- Photo 128 x 76 (slika veće razlučivosti)
- Photo 88 x 72 (slika manje razlučivosti)
- Video 88 x 72 (video manje razlučivosti)

Ovisno o odabranom modu inicijaliziramo PAC207B senzor i čitamo sliku odgovarajuće veličine. Inicijalizacija senzora se sastoji od upisivanja predefiniranih vrijednosti u njegove registre. Popis važnijih registara je dan u nastavku.

Tablica 4. *Važniji registri PAC207B optičkog senzora*

Adresa Registra	**Opis**
0x00	*Chip ID* Identifikacijski broj senzora
0x01	*Chip ID* Identifikacijski broj senzora
0x02	*PXCK* Brzina rada, faktor kojim se dijeli frekvencija takta, ekspozicija 4-26, inicijalno 4, 12MHz / n EXPOSURE_KNEE = 15
0x08	Brightness Razina osvjetljenja
0x0B	*Rgain* Razina komponente crvene boje, inicijalno 0x64
0x0C	*Ggain* Razina komponente zelene boje, inicijalno 0x64
0x0D	*Bgain* Razina komponente plave boje,

	inicijalno 0x64
0x0E	*PGA global gain* Razina kontrasta
0x13	Upis registara u senzor 0x01 => upis
0x1C	Upis registara u senzor 0x01 => upis
0x40	*Pipe start / stop* Omogućuje slanje slike 0x00 => stop 0x01 => start
0x41	*Image format / LED* Odabir veličine slike i kontrola led diode Bit_0 = Image Format 0x1 => 176x144 Bayer 0x0 => 352x288 Bayer Bit_1 = LED 0x1 => OFF 0x0 => ON
0x4A	*Compression balance size* Kompresija slike 0xFF => bez kompresija 0x88 => kompresija

Kad sustav čita sliku veće razlučivosti 128 x 76 piksela, pretpostavlja se da se slika ne šalje na računalo, već je to samo pokusni način rada. Slika se prikazuje na LCD display-u i u slučaju da se ipak pokušava poslati, neće biti poslana u cijelosti. Serijska komunikacija pretpostavlja fiksnu veličinu slike koja je u ovom slučaju veličina slike manje razlučivosti. Kod čitanja slike manje razlučivosti 88 x 72 piksela slika se šalje na računalo. Potprogrami za inicijalizaciju i pokretanje kamere su dani u nastavku.

```
////////////////////////////////////////////////////////////////////////////////
// Inicijalizacija kamere ////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int camera_init(unsigned char camera_mode)
{
        unsigned char CamSpeed = 26;

        if(camera_mode==2)
        {
                camera_mode--;
                CamSpeed = 5;
        }

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,    // Bit_0=Image Format
                             0x00, 0x41, 0, NULL, 0 )!=0 ) {  // Bit_1=LED
                return 0;                                     // Bit_2=Compression test
        }                                                     // mode enable

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,    // Power Control
                             0x00, 0x0f, 0, NULL, 0 )!=0 )  return 0;

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,    // Analogni prednapon
                             0x30, 0x11, 0, NULL, 0 )!=0 )  return 0; // (Analog Bias)
        //
        // Niz kontrolnih upisa za inicijalizaciju
        //
        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,
                             0x10, 0x0f, 0, NULL, 0 )!=0 )  return 0;
        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x01,
                             0, 0x0002, 8, &pac207_sensor_init[0], 8 )!=8 )  return 0;
        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x01,
                             0, 0x000a, 8, &pac207_sensor_init[1], 8 )!=8 )  return 0;
        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x01,
                             0, 0x0012, 8, &pac207_sensor_init[2], 8 )!=8 )  return 0;
        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x01,
                             0, 0x0040, 8, &pac207_sensor_init[3], 8 )!=8 )  return 0;
        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x01,
                             0, 0x0042, 8, &pac207_sensor_init[4], 8 )!=8 )  return 0;
        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x01,
                             0, 0x0048, 4, &PacReg72, 4 )!=4 )  return 0;

        ////////////////////////////////////////////////////////////////////////////
```

```
        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,          // Compression Balance
                             0xff, 0x4a, 0, NULL, 0 )!=0 ) {   // 0xff - bez kompresije
                return 0;                                      // 0x88 - kompresija
        }

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,          // Sram test value
                             0x00, 0x4b, 0, NULL, 0 )!=0 )  return 0;

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,          // Bit_0=Image Format
                             camera_mode, 0x41, 0, NULL, 0 )!=0 ) {
        return 0;                                              // 0x1 - 176x144 Bayer
        }                                                      // 0x0 - 352x288 Bayer

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,          // PXCK,  default: 4
                             CamSpeed, 0x02, 0, NULL, 0 )!=0 ) {  //  EXPOSURE 4-26,
        return 0;                                                 // 12MHz /n = 12MHz/26
        }                                                         // EXPOSURE_KNEE 15
        return 1;
}

////////////////////////////////////////////////////////////////////////////////////////
// Završna inicijalizacija i pokretanje slanja slike ///////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
int camera_start(unsigned long ulBulkInPipe )
{
        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,          // Kontrast = 2
                             2, 0x0e, 0, NULL, 0 )!=0 )  return 0;

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,          // Razina osvijetljenosti
                             220, 0x08, 0, NULL, 0 )!=0 )  return 0;   // Brightness = 220

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,        // Učitaj registre u senzor
                             0x01, 0x13, 0, NULL, 0 )!=0 )  return 0;

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,
                             0x01, 0x1c, 0, NULL, 0 )!=0 )  return 0;

        SysCtlDelay(15*SysCtlClockGet()/30);
```

```
        //
        // Kamera start
        //
         if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,                    // Start pipe
                              0x01, 0x0040, 0, NULL, 0 )!=0 )  return 0;   // Početak
                                                                           // prijenosa
        return 1;
}
```

Osnovna funkcija koja se višestruko ponavlja kod inicijalizacije, odnosno kod upisivanja u registre kamere *usb_control_msg* opisana je u nastavku. Funkcija vraća broj uspješno upisanih, odnosno pročitanih bajtova. Opis argumenata funkcije:

dir – smjer komunikacije (pisanje ili čitanje registara)

USB_RTYPE_DIR_IN	0x80
USB_RTYPE_DIR_OUT	0x00

cmd – kod čitanja ili pisanja jednog bajta je 0x00, a kod čitanja ili pisanja više bajtova je 0x01.

value – vrijednost koja se upisuje. Kod čitanja je 0x00.

index – indeks registra kamere u koji se upisuje ili čija vrijednost se čita.

tlen, rlen – broj bajtova koji se upisuju ili čitaju. Koristi se kod višestrukog upisivanja.

rbuf – spremnik (*buffer*) sa kojega se čitaju podaci za upisivanje u registre kamere ili u koji se upisuju podaci pročitani sa kamere. Ako je *NULL* pretpostavlja se da se podatak upisuje u kameru, da je dan preko argumenta *value*, te da je veličina poruke 1 bajt.

```
////////////////////////////////////////////////////////////////////////////////
// Funkcija za slanje i primanje poruka sa kamere ////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
unsigned long
usb_control_msg ( unsigned char dir, unsigned char cmd,
        unsigned short value, unsigned short index, unsigned short tlen, void *rbuf,
        unsigned short rlen )
{
   tUSBRequest SetupPacket;
   SetupPacket.bmRequestType = dir | USB_RTYPE_VENDOR | USB_RTYPE_INTERFACE;
   SetupPacket.bRequest = cmd;
   SetupPacket.wValue = value;
   SetupPacket.wIndex = index;
   SetupPacket.wLength = tlen;

   return USBHCDControlTransfer (0,
                    &SetupPacket,
                    pDevice->ulAddress,
                    (unsigned char *)rbuf,
                    rlen,
                     pDevice->DeviceDescriptor.bMaxPacketSize0);
}
```

6.2.2. Podešavanje osvjetljenja slike (*Brightness*)

Osim inicijalizacije kamere, implementiran je dio programa kojim se vrši podešavanje jačine osvjetljenja na samoj slici. Potprogram upisuje željenu vrijednost jačine osvjetljenja u odgovarajući 8-bitni registar kamere na adresi 0x08 (*Brightness* registar). Bitno je napomenuti da vrijednost upisana u registar ne utječe na sam proces detekcije slike kod senzora, već se samo na osnovu nje kodiraju dobiveni pikseli slike sa vrijednostima većeg intenziteta svijetlosti. To znači da se na ovakav način može slika samo programski osvijetliti te takvo osvjetljenje ne utječe na kvalitetu slike ukoliko je slika već kod detekcije bila loše kvalitete. Implementirano je da se potprogram pokreće kod pritiska navigacijskog prekidača u smjeru *sjever/jug* (*North/South*). Kao argument potprograma se navodi 8-bitna vrijednost koja se upisuje u registar. Potprogram se nalazi u nastavku.

```
////////////////////////////////////////////////////////////////////////////////
// Podešavanje osvijetljenosti (BRIGHTNESS) ////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int camera_brightness(unsigned char brightness)
{
        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,     // BRIGHTNESS 0-255
                        brightness, 0x08, 0, NULL, 0 )!=0 ) {    // default: 4
                return 0;
        }

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,     // Učitaj registar u senzor
                                0x01, 0x13, 0, NULL, 0 )!=0 ) {  // (Load register to sensor)
                return 0;
        }

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,
                                0x01, 0x1c, 0, NULL, 0 )!=0 ) {
                return 0;
        }
        return 1;
}
```

6.2.3. Podešavanje kontrasta slike

Slično kao kod potprograma za podešavanja osvjetljenja implementiran je potprogram za podešavanje kontrasta na slici. Željeni intenzitet kontrasta se upisuje u 8-bitni registar kamere na adresi 0x0E (*PGA global gain* registar). Ni kod podešavanje kontrasta slike, nema utjecaja na ukupnu kvalitetu slike. Vrijednost upisana ne utječe na detekciju slike kod senzora, već se na osnovu nje kodiraju pojedini pikseli slike sa vrijednostima većeg intenziteta kontrasta. To znači da ukoliko je kvaliteta slike bila loša već kod detekcije, programsko podešavanje kontrasta je neće poboljšati. Implementirano je da se potprogram pokreće kod pritiska navigacijskog prekidača u smjeru *istok/zapad* (*East/West*). Kao argument potprograma se navodi 8-bitna vrijednost koja se upisuje u registar. Kod potprograma je u nastavku.

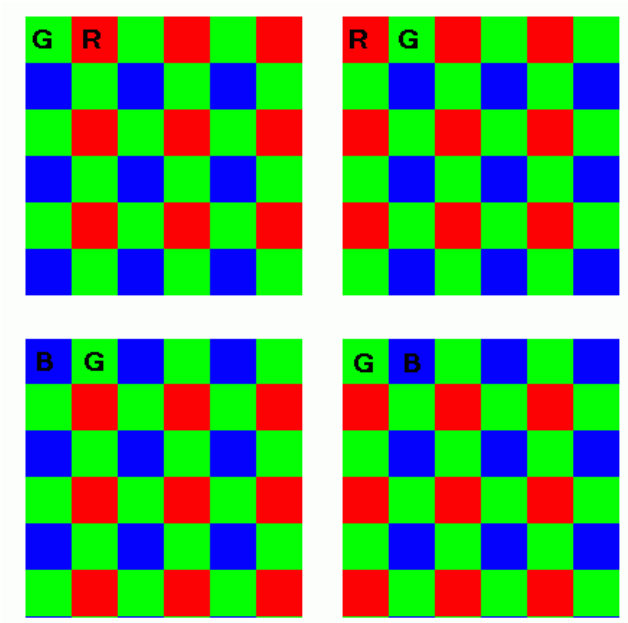
```
////////////////////////////////////////////////////////////////////////////////
// Podešavanje kontrasta (GAIN) ////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int camera_contrast(unsigned char contrast)
{
        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,          // CONTRAST 0-31
                             contrast, 0x0e, 0, NULL, 0 )!=0 ) {   // default: 9
        return 0;                                              // GAIN_KNEE   20
        }

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,     // Učitaj registar u senzor
                             0x01, 0x13, 0, NULL, 0 )!=0 ) {  // (Load register to sensor)
                return 0;
        }

        if( usb_control_msg( USB_RTYPE_DIR_OUT, 0x00,
                             0x01, 0x1c, 0, NULL, 0 )!=0 ) {
                return 0;
        }
        return 1;
}
```

6.3. Bayer piksel raster

Bez obzira o kojoj veličini slike se radi, slika se uvijek čita sa pikselima raspoređenim u tzv. *Bayer* matricu. Svaki pojedini piksel može biti crvene (R), zelene (G) ili plave (B) boje.



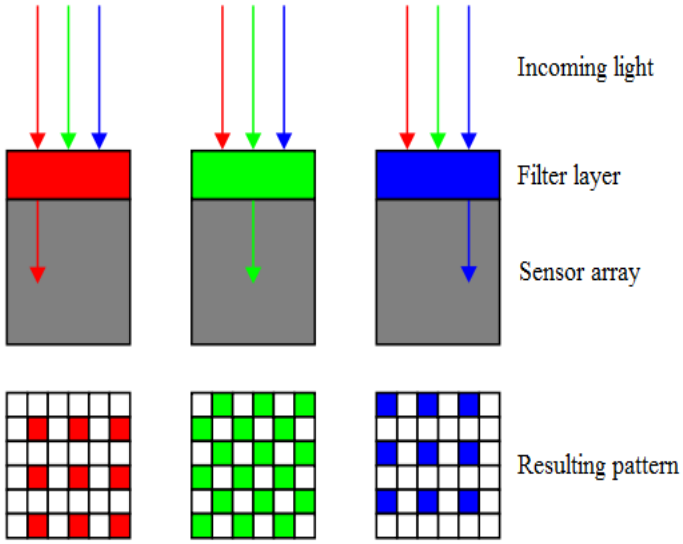
Slika 14. *Moguće kombinacije Bayer uzorka*

(Preuzeto sa: http://arnholm.org/astro/software/aviraw/)

Slika se sastoji od niza manjih matrica veličine 2x2 piksela koje predstavljaju uzorke odnosno piksele konačne slike u 24-bitnom bitmap formatu. Moguće kombinacije boja unutar 2x2 matrica su GRBG, RGGB, BGGR i GBRG.

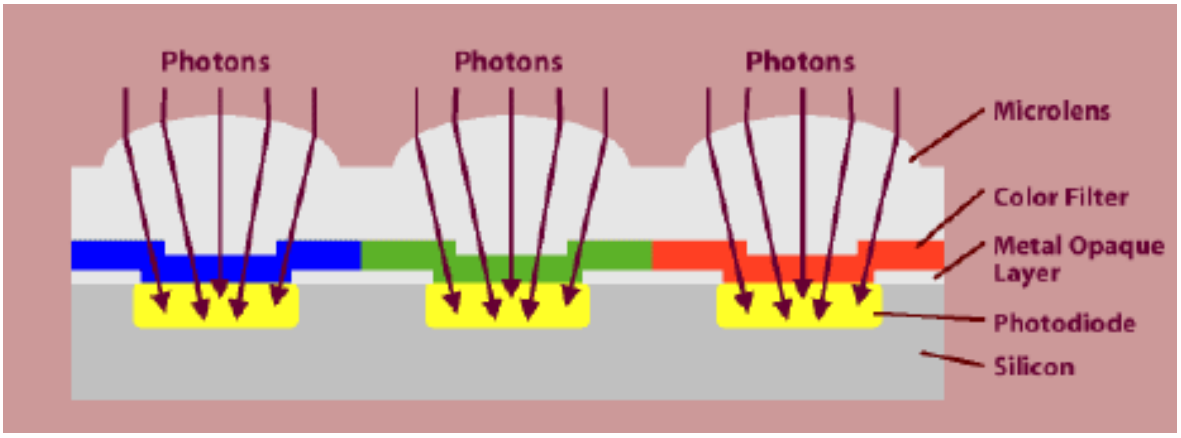
Takav raspored piksela je posljedica filtra kroz koji senzor detektira svijetlost. Većina optičkih senzora koristi ovakav tip detekcije. Svijetlost prolazi kroz matricu filtara gdje svako pojedino polje matrice služi za detekciju boje određene valne duljine. U ovoj fazi električni signali svake ćelije razmjerni su jačini upadne svjetlosti. Na kraju se na osnovu dobivenih uzoraka formira *Bayer* matrica. Svaki uzorak ima 50% zelene, 25% crvene i 25% plave boje. Dvostruko veći broj zelenih piksela se uzima zato što je ljudsko oko najosjetljivije na zelenu svjetlost. Nakon detekcije boja, na osnovu različite jakosti signala iz pojedinačnih ćelija dobiju se intenziteti pojedinih boja koji se kodiraju i slika je spremna za slanje. Svaki piksel se kodira sa 8-bitnim binarnim brojem gdje 0x00 predstavlja najmanji intenzitet, a 0xFF najjači intenzitet.

Kod digitalnih fotoaparata se svaki piksel konačne slike izračunava na temelju podataka o intenzitetima susjednih boja. O takvoj interpolaciji boja ovisi konačna kakvoća snimljene fotografije. Poboljšanje kakvoće digitalne fotografije uz poboljšanja fotoosjetljivih senzora zasluga su usavršavanja interpolacijskih algoritama. Detekcija slike senzorima najčešće je relativno brz postupak, ali zato obrada i pohrana traju dosta dugu. Kako bi se ubrzao taj dio procesa, mnogi kvalitetniji fotoaparati imaju više radne memorije za privremenu pohranu fotografija kako obrada snimaka ne bi ometala snimanje.



Slika 15. *Detekcija pojedinih boja*

(Preuzeto sa: http://en.wikipedia.org/wiki/File:Bayer_pattern_on_sensor_profile.svg)



Slika 3. *Građa filtara optičkih senzora*

(Preuzeto sa: http://www.siliconimaging.com/RGB%20Bayer.htm)

Slika koju tako učitamo je veličine 176 x 144 piksela u *Bayer* rasteru kod manje razlučivosti odnosno 352 x 288 kod veće razlučivosti. Svaki piksel je spremljeni kao 8-bitna vrijednost. Dakle slika koju čitamo je veličine:

$176 \cdot 144 = 25\,344B$ => manja razlučivost

$352 \cdot 288 = 101\,376B$ => veća razlučivost

Na raspolaganju na stoji 64 KB SRAM memorije. Sliku veće razlučivosti nije moguće učitati jednim prolazom u potpunosti, stoga se ne učitava cijela već jedan dio slike veličine $352 \cdot 152 = 53\,504B$. Budući da je svaka slika spremljena u *Bayer* piksel rasteru sastavljena od manjih 2x2 matrica piksela koji predstavljaju konačni piksel snimljene slike, konačna slika je dvostruko manjih dimenzija bez obzira na razlučivost. Tako je slika manje razlučivosti na kraju veličine:

$$\frac{176}{2} \text{ x } \frac{144}{2} \Rightarrow 88 \text{ x } 72$$

Isto tako i slika veće razlučivosti ima konačne dimenzije dvostruko manje:

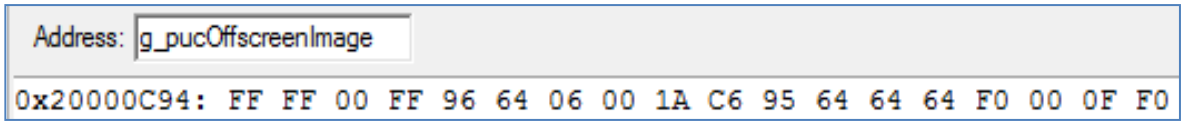
$$\frac{352}{2} \text{ x } \frac{152}{2} \Rightarrow 176 \text{ x } 76$$

Pod *"konačna"* slika se podrazumijeva slika u 24-bitnim bitmap formatu. Kod navedenog formata se svaki piksel slike kodira sa 3 bajta, što u konačnici daje 24 bita. Svaki bajt predstavlja intenzitet plave, zelene i crvene boje. Sa 24-bitnim formatom je ukupno moguće kodirati $2^{24} = 16\,777\,216$ boja, što se još naziva *true color* formatom. Na LCD display-u kojeg koristimo je moguće prikazati slike u 16-bitnim bitmap formatu pa je moguće prikazati $2^{16} = 65\,536$ boja. Bez obzira na ograničenje LCD display-a slike spremamo u 24-bitnim formatu pošto na računalu nismo ograničeni kvalitetom prikazane slike te je stoga moguće prikazati slike sa preko 16M boja.

Kod veće kvalitete slike problem predstavlja zauzeće memorije pa sukladno s tim i dulje vrijeme potrebno za slanje slike na računalo. Unatoč navedenim nedostacima sliku spremamo u formatu veće kvalitete pošto kod brzine serijskog prijenosa od 2 Mbps razlika u brzini osvježavanja prikaza na LCD-u i na računalu nije toliko osjetna.

6.4. Čitanje slike sa kamere i prikaz

Ovisno o odabranom načinu rada čitamo sliku odgovarajuće veličine sa kamere. Čitanje se vrši u potprogramu preko *BULK IN endpoint*-a. Svaka slika se sastoji od zaglavlja (*start of frame - SOF*) i *Bayer* matrice piksela. Nakon svakoga retka učitane slike dolazi oznaka kraja retka (*end of line - EOL*). Oznaka kraja retka sadrži 2 bajta i u slučaju bez kompresije glasi 0x0F i 0xF0, a u slučaju komprimirane slike 0x1E i 0xE1. U programu je implementirano da se uvijek čita slika bez kompresije. Zaglavlje se učitava prije svake slike. Kod prvog čitanja SOF se sastoji od 18 bajta, a kod svakog slijedećeg od 15 bajta.



Slika 16. *SOF kod prvog čitanja*

Prva 3 bajta (0xFF, 0xFF i 0x00) se pojavljuju samo kod prvog čitanja i nema ih kod svakog slijedećeg. Sljedeća 3 bajta (0xFF, 0x96 i 0x64) se ponavljaju kod svakog čitanja i uvijek su ista. Nadalje slijede redom bajtovi:

<*SEQ*> –Broj između 0x00 i 0x07.

0x00 – Konstantan bajt. Ponavlja se kod svakog čitanja.

<*PXCLK*> – Vrijednost upisana u *PXCK* registar kamere.

<*BR1*>, <*BR2*> – Nepoznati brojevi.

<*Rgain*>, <*Ggain*>, <*Bgain*> – Vrijednosti upisane u *Rgain, Ggain* i *Bgain* registre kamere.

0xF0, 0x00, 0x0F i 0xF0 – Konstantan niz bajtova. Ponavlja se kod svakog čitanja.

Nakon svakog čitanja potrebno je naći i odrediti SOF slike. To međutim nije jednostavno pošto se sinkronizacija izgubi nakon svakog čitanja, te nije zajamčeno da će se zaglavlje nalaziti uvijek na početku pročitanog niza podataka. Moguće je također da zaglavlja nema kod novo pročitane slike. Nestanak zaglavlja se često javlja kod veće razlučivosti pošto zbog ograničene memorije nije moguće u cijelosti učitati sliku. Iz navedenih razloga se mora nakon svakog čitanja iznova tražiti SOF koji se sada može nalaziti bilo gdje u pročitanom nizu. Jedino što je uvijek isto i zbog čega se i traži zaglavlje je činjenica da nakon zadnjeg bajta zaglavlja počinje nova slika. Podaci učitani prije zaglavlja nemaju nikakvo značenje, niti se mogu iskoristiti.

Pošto se zaglavlje ne nalazi uvijek na početku pročitanog niza mora se uzeti veći spremnik u koji se spremaju pročitani podaci. Uzima se veličina spremnika od 59 000 bajta

Veličina spremnika => BUFFER_SIZE 59000

Pretpostavlja se da će se slika manje razlučivosti u cijelosti učitati u prvih 45 000 bajtova spremnika. Efektivna veličina slike u *Bayer* rasteru je 25 344 bajta. Dakle ostaje još $45\,000 - 25\,344 = 19\,656$ bajta. Svaki redak slike završava sa 2 EOL bajta. Pošto slika ima 144 redaka ostaje još $19\,656 - 2 \cdot 144 = 19\,368$ bajta. Potrebno je definirati interval unutar kojeg se traži zaglavlje. Kao interval se uzima prvih 18 000 bajta učitane slike. Ukoliko program ne nađe zaglavlje u tom intervalu čita se nova slika. Ako bi interval bio veći od 19 368 bajta postoji mogućnost da se SOF nađe prekasno, odnosno da jedan dio slike neće biti učitan u cijelosti. Nakon zaglavlja slijedi 25 632 bajta slike i ako taj dio prelazi 45 000 bajt spremnika slika nije u cijelosti učitana.

Slika manje razlučivosti => BUFFER_VIDEO 45000

SOF interval 176 x 144 => SOF_RANGE_SMALL 18000

Problem se javlja kod čitanja slike veće razlučivosti. Kod takvog čitanja punimo cijeli raspoloživi spremnik od 59 000 bajta. Naravno, slika nije učitana u cijelosti, već se pretpostavlja da je učitano prvih 53 504 bajta slike, dakle slika veličine 352 x 152 piksela u *Bayer* rasteru. Imamo 152 retka pa $2 \cdot 152 = 304$ bajta otpada na EOL. Ostane još $59\,000 - 53\,808 = 5192$ bajta koji bi se mogli iskoristiti kao interval za traženje zaglavlja.

SOF interval 352 x 152 => SOF_RANGE_BIG 5192

Kod detekcije zaglavlja pokazalo se dovoljno naći samo 4., 5. i 6. bajt (0xFF, 0x96 i 0x64). Prva 3 bajta se ne ponavljaju te stoga njih ne tražimo. Ostali bajtovi se ponavljaju ovisno o konfiguraciji pa se ne preporuča koristiti njih kod usporedbe. Potprogram kojim se vrši detekcija je dan u nastavku. Kao argument se zadaje veličina SOF intervala u kojem se traži zaglavlje. Potprogram vraća poziciju početka zaglavlja u *bufferu*.

Nakon detekcije SOF u drugom potprogramu se sa slike uklanjaju svi završeci redaka (EOL - 0x0F, 0xF0). Kao argumenti potprograma se zadaju veličina slike u *Bayer* rasteru (176 x 144 ili 352 x 152) i pozicija prvog piksela slike u *bufferu*. Potprogram je dan u nastavku.

```
////////////////////////////////////////////////////////////////////////////////
// Nađi Start_of_frame (SOF) u prvih SOF_RANGE znakova ////////////////////////
// Prvi SOF = 0xFF, 0xFF, 0x00, 0xFF, 0x96, 0x64 //////////////////////////////
// Tražimo 4. znak ----------^  jer svi ostali frame-ovi počinju od njega /////
////////////////////////////////////////////////////////////////////////////////
unsigned int find_SOF(int sof_range)
{
        unsigned int i;
        int x, nasli_pocetak = 0;
        for(i=0; i<sof_range; i++)      // traži SOF unutar intervala
        {                               // pozicioniraj se na prvi element
              for(x=0; x<12; x++)       // potrebno naći 12 uzastopnih znakova SOF
              {                         // x je brojač (pointer na znak)
                    if((x>2)&&(x<9))            // znakovi između 2. i 9. se mogu razlikovati
                          continue;             // zanemari ih
                    if(g_pucOffscreenImage[i+x] != (unsigned char)sof[x])  // usporedba
                          break;    // ako se ne podudara pozicioniraj se ne slijedeći i
                                    // ponovi postupak (reset x)
                    else if(x==2)   // ako se podudaraju prva 3 znaka 0xFF, 0x96, 0x64
                          nasli_pocetak = 1;    // pretpostavlja se da smo nasli SOF
                }
                if(nasli_pocetak)   // ako smo našli izađi
                        break;
        }
        return i;           // vrati početnu poziciju SOF u bufferu
}
```

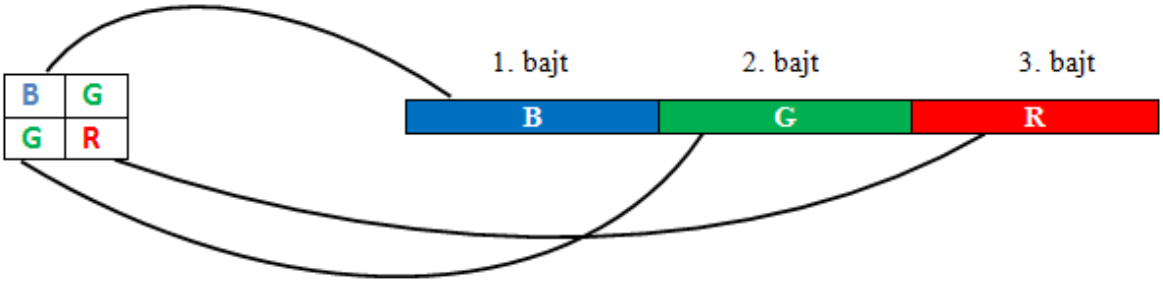
```
////////////////////////////////////////////////////////////////////////////////
// Pronađi svaki End_of_line (EOL) i iznova spremi sliku u buffer bez (EOL) ////////////
////////////////////////////////////////////////////////////////////////////////
void remove_EOL(int row_length, int max_row, unsigned int start)
{
        int row, j=0;

        for (row = 0; row < max_row; row++) {          // prolazak kroz sve retke
           for(; start<=BUFFER_SIZE ; start++, j++)    // prolazak kroz sve piksele
           {                                           // j je brojač piksela u retku
                if((g_pucOffscreenImage[start]==0x0F) &&
                (g_pucOffscreenImage[start+1]==0xF0))
               {        // ako smo došli do kraja retka
                        j -= row_length;  // resetiraj j
                        start +=2;        // preskoči EOL oznaku
                        break;            // slijedeći redak
               }
               g_pucOffscreenImage[row*row_length + j]=g_pucOffscreenImage[start];
           }            // ako nismo na kraju retka posmakni cijeli redak (prebriši EOL)
        }
}
```

Konačno imamo sliku u jednoj cjelini bez oznaka kraja redaka EOL. Pikseli slike su i dalje spremljeni u *Bayer*-ovoj matrici. Kombinacija boja je dana u nastavku:

Slika 17. *Korištena Bayer matrica piksela sastavljena od manjih 2x2 matrica*

Svaka manja 2x2 matrica predstavlja jedan piksel konačne slike. Dakle potrebno je na osnovu BGGR kombinacije boja dobiti kodiran konačni piksel u 24-bitnom bitmap formatu. Radi jednostavnosti i zbog brzine izvođenja programa ne koriste se algoritmi za interpolaciju piksela i boja radi postizanja bolje kvalitete. Za potrebe video portafona vrhunska kvaliteta slike ne igra značajnu ulogu. Budući da je kod 24-bitnog bitmap formata intenzitet plave, zelene i crvene boje kodiran sa 8 bita te pošto je kod danog *Bayer* rastera svaki piksel kodiran sa 8 bita, vrijednosti boja se samo kopiraju u odgovarajuća RGB polja u zapisu bitmap formata. Radi jednostavnosti se ne vrši interpolacija pa se od dane dvije zelene boje u 2x2 matrici uzima se samo jedna. Konačna slika se sprema u memoriju kako je prikazano u nastavku:



Slika 18. *Prikaz spremanja slike iz Bayer rastera u jedan piksel 24-bitnog bitmap*

Potprogram koji vrši pretvorbu Bayer rastera u 24-bitni bitmap format je dan u nastavku. Kao argumenti potprograma se zadaju veličina slike u *Bayer* rasteru (176 x 144 ili 352 x 152) i pozicija prvog piksela slike u *bufferu* koji nakon pretvorbe postaje početak odnosno prvi bajt konačne slike. Slika u *Bayer* rasteru se prvo podjeljuje na manje 2x2 matrice. U petlji se prolazi kroz svaku matricu i intenziteti boja se kopiraju u RGB polja 24-bitnog formata. Varijable *x* i *y* predstavljaju koordinate svake matrice i preko njih se vrši prepoznavanje boja.

Tablica 5. *Prepoznavanje boja prema koordinatama (x,y)*

(0,0) => plavi piksel
(0,1)|(1,0) => zeleni piksel
(1,1) => crveni piksel

x/y	0	1
0	B	G
1	G	R

```
////////////////////////////////////////////////////////////////////////////////
// Bayer BGGR u 24bpp bitmat pretvorba ////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void bayer_to_24bpp_save(int w, int h, unsigned char *bayer)
{
        int x, y, nx, ny;
        unsigned long blue, green, red;

        // Piksel u 24bpp bitmap-u je 2x2 piksela u Bayer-u (BGGR)
        // Raspodjeli na manje 2x2 matrice
        int nw = w/2;           // širina/2
        int nh = h/2;           // visina/2

        for (ny = 0; ny < nh; ++ny) {     // Prolaz kroz sve 2x2 matrice
          for (nx = 0; nx < nw; ++nx) {
            blue=red=green=0;

            // 2x2 Bayer (BGGR)
            for (y = 0; y < 2; ++y) {           // y koordinata 2x2 matrice
              for (x = 0; x < 2; ++x) {         // x koordinata 2x2 matrice
                if((!x)&(!y)) blue = bayer[y*w + nx*2 + ny*2*w + x];     // 00  => plavi piksel
                if((x)&(!y)||(!x)&(y)) green = bayer[y*w + nx*2 + ny*2*w + x]; // 01|10 =>
                                                                        // zeleni piksel
                if((x)&(y)) red = bayer[y*w + nx*2 + ny*2*w + x];       // 11  => crveni piksel
              }
            }
            bayer[ny*nw*3 + nx*3] = blue;           // Bit_7-0 Plava (Blue)
            bayer[ny*nw*3 + nx*3+1] = green;        // Bit_15-8 Zelena (Green)
            bayer[ny*nw*3 + nx*3+2] = red;          // Bit_23-16 Crvena (Red)
          }
        }
}
```

Nakon pretvorbe slike u bitmap format, moguće je sliku prikazati na LCD display-u. LCD display podržava slike u 16-bitnom formatu, te je stoga potrebno prilagoditi sliku za prikaz. Slika se iscrtava piksel po piksel. Problem nastaje kod prikaza slike veće razlučivosti budući da dimenzije prelaze veličinu LCD display-a. Dimenzije display-a su 128 x 128

piksela. Slika koja se treba prikazati je veličine 176 x 76 piksela. Radi jednostavnosti prikazuje se samo središnji dio slike veličine 128 x 76 piksela. Ostatak slike se zanemaruje i ne prikazuje se na display-u. Kod manje razlučivosti prikaz ne predstavlja problem jer dimenzije slike ne prelaze veličinu display-a. Prikaz slike se vrši u potprogramu. Kao argumenti potprograma se zadaju dimenzije slike u 24-bitnom bitmap formatu.

```
////////////////////////////////////////////////////////////////////////////////
// Prikaz 24bpp bitmap slike na display-u ///////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
void slika_24bpp(long lWidth, long lHeight)
{
        unsigned long x, y, boja;

        for(y = 0; y < lHeight; y++)
        {
          for(x = 0; x < lWidth; x++)
          {
            // Citamo redom plavu, zelenu i crvenu 8-bitnu boju (3*8=24bpp)
            boja =0;
            // Plava (Blue)
            boja |= (unsigned long)g_pucOffscreenImage[y*lWidth*3 + x*3];
            // Zelena (Green)
            boja |= ((unsigned long)g_pucOffscreenImage[y*lWidth*3 + x*3 + 1]<<8);
            // Crvena (Red)
            boja |= ((unsigned long)g_pucOffscreenImage[y*lWidth*3 + x*3 + 2]<<16);
            // Pretvorba 24bpp u 16bpp
            GrContextForegroundSet(&g_sContext, boja);
            // Prikaži piksel na sredini display-a
            GrPixelDraw(&g_sContext, x+(128-lWidth)/2, y+33);
          }
        }
}
```

6.5. Slanje slike na računalo

Nakon izvršene prilagodbe slike pročitane sa kamere potrebno je sliku poslati na računalo. Implementirana su 2 načina slanja na računalo.

6.5.1. Serijska komunikacija sa računalom

Prvi način slanja je već prije spomenuti preko serijske komunikacije. Slika manje razlučivosti se šalje preko serijske RS-232 komunikacije na računalo. Slanje slike veće razlučivosti nije implementirano i u slučaju slanja moguć je gubitak ili potpuno krivi prikaz na računalu pošto je program na računalu prilagođen za primanje i prikaz slike manje razlučivosti. U slučaju da koristimo način rada čitanja slike veće razlučivosti pretpostavlja se da se ne šalje na računalo.

Brzina serijske komunikacije može se proizvoljno postavljati. Najveća podržana brzina kod koje nije primijećen gubitak podataka je 2 Mbps. Mikrokontroler LM3S3748 na razvojnom sustavu podržava proizvoljne brzine rada RS-232 komunikacije pa tako i nestandardne brzine. Računalo također podržava navedenu brzinu serijske komunikacije. Slika manje razlučivosti zauzima 19 008 bajta. Brzinom od 2 Mbps slika se prenose na računalo u 76.032 ms. U idealnom slučaju očekivana brzina osvježavanja na računalu uz navedenu brzinu serijske komunikacije je 13 slika u sekundi, što je za potrebe video portafona potpuno zadovoljavajuće. Ako se uzme u obzir i vrijeme čitanja slike sa kamere te vrijeme prilagođavanja slike prije slanja, realno je moguće dobiti brzinu osvježavanja od oko 4 slike po sekundi. Iako je postignuta relativno mala brzina osvježavanja, rezultat je zadovoljavajući pa se stoga još uvijek može govoriti o praćenju slike u realnom vremenu što kod video portafona igra značajnu ulogu.

Veći nedostatak predstavlja veza između razvojnog sustava i računala koja kod ovakvog oblika komunikacije mora biti žičana. Dakle potrebno je kablom spojiti razvojni sustav sa računalom. To je ograničenje jer bi u tom slučaju računalo trebalo biti fizički relativno blizu portafona, odnosno udaljenost računala od portafona ovisi o dužini kabla kojim je spojeno. Međutim ovakav tip komunikacije predstavlja jeftin i relativno jednostavan način povezivanja.

Slika 19. *Prikaz spajanja sustava preko serijske RS-232 veze sa računalom*

Serijska RS-232 komunikacija je ostvarena preko UART0 porta na LM3S3748 mikrokontroleru. UART0 je implementiran na GPIO A portu koji podržava serijsku komunikaciju. Programski odsječak koji inicijalizira seriju je dan u nastavku:

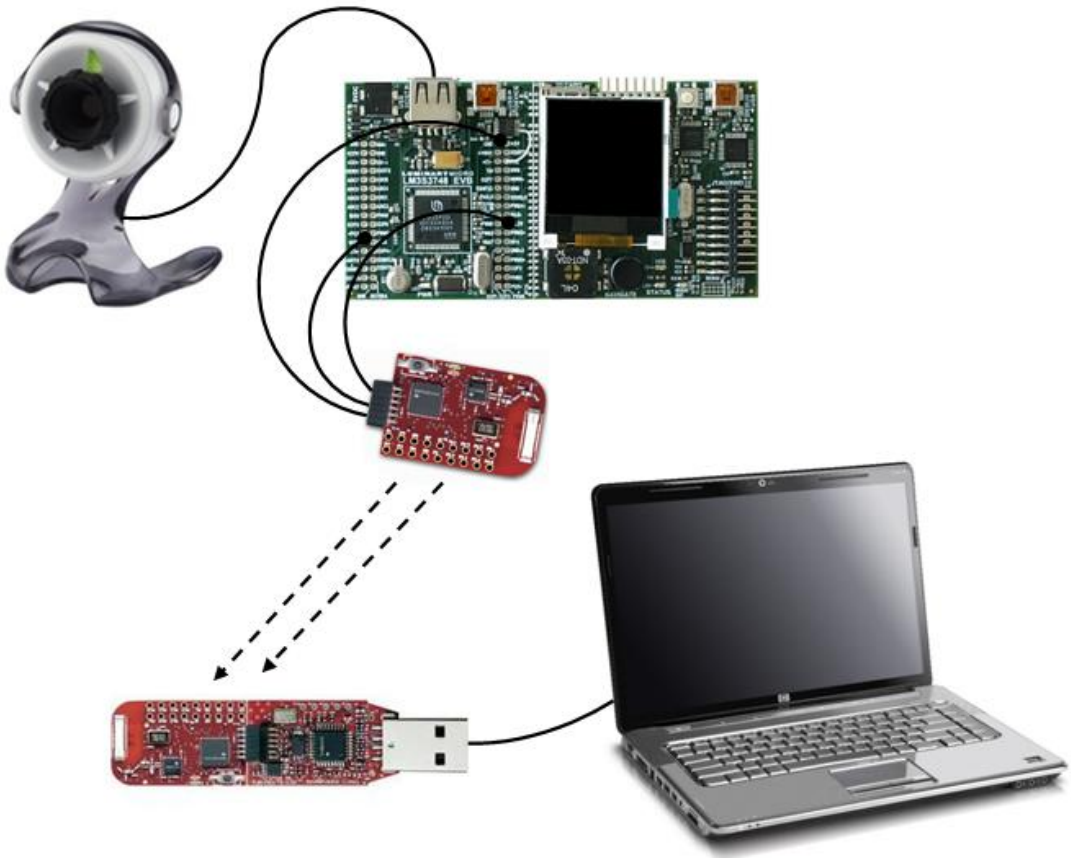
```
//////////////////////////////////////////////////////////////////////////////////
// Inicijalizacija UART0 za seriju i postavljanje brzine ////////////////////////////
//////////////////////////////////////////////////////////////////////////////////
ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);   // UART0 za seriju
ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);   // GPIOA podržava UART
//
// postavi GPIO A0 i A1 kao UART.
//
ROM_GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
//
// postavi brzinu serije na 2 Mbps
//
UARTStdioInitExpClk(0, 2000000);
```

Radi sinkronizacije slike na računalu u slučaju mogućeg gubitka podataka uslijed pogrešaka u prijenosu, dodaju se 3 dodatna sinkronizacijska bajta. Zbog jednostavnosti svi bajtovi su jednaki i kodirani sa 0x12. Sinkronizacijski bajtovi se šalju umjesto zadnjeg piksela slike koji u 24-bitnom bitmap formatu zauzima također 3 bajta, tako da se i dalje šalje ukupno 19 008 bajta. Programski odsječak za slanje slike je dan u nastavku:

```
////////////////////////////////////////////////////////////////////////////////
// Prijenos slike na računalo preko serije ////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
for(start=0; start<3; start++)  // dodavanje 3 sinkronizacijska bajta
{                               // umjesto zadnjeg piksela slike
        g_pucOffscreenImage[88*72*3-3+start]=0x12;
}

for(start=0;  start<88*72*3;  start++)          // slanje slike
        UARTprintf("%c", g_pucOffscreenImage[start]);
```

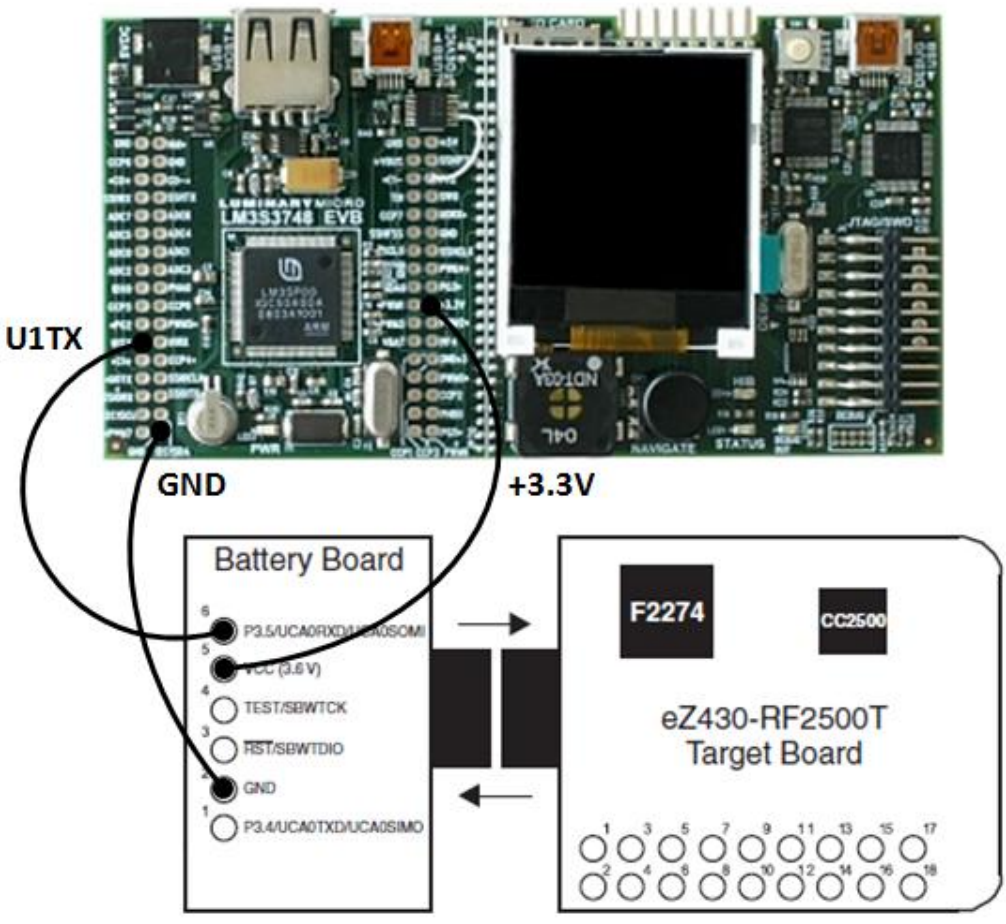
6.5.2. Bežična komunikacija sa računalom



Slika 20. *Prikaz spajanja modula za odašiljanje na razvojni sustav sa LM3S3748 mikrokontrolerom i modula za primanje podataka na računalo*

Drugi način vezanja je preko bežične komunikacija. Za takve potrebe se koristi drugi razvojni sustav EZ430-RF2500 sa MSP430 mikrokontrolerom. Prednost ovakve komunikacije je što računalo može biti udaljeno od portafona do nekoliko desetaka metara. Pošto je komunikacija bežična nije potreban kabel za spajanje. Računalo fizički ne mora biti u neposrednoj blizini portafona.

Nedostatak je što se razvojnim sustavom bežično šalju paketi maksimalne veličine do 33 bajta. To je veliko ograničenje jer je potrebno proći određeno vrijeme između slanja paketa. Pošto su paketi veličine 33 bajta, ukupno se šalje $19\,008/33 = 576$ paketa. Problem je što slanje vremenski traje puno duže nego kod serijske komunikacije. Postiže se brzina osvježavanja od oko 0.35 slika po sekundi. Dakle svake 2 do 3 sekunde prikaže se nova slika na računalu. Brzina osvježavanja je 10 puta lošija u odnosu na serijsku žičanu komunikaciju.



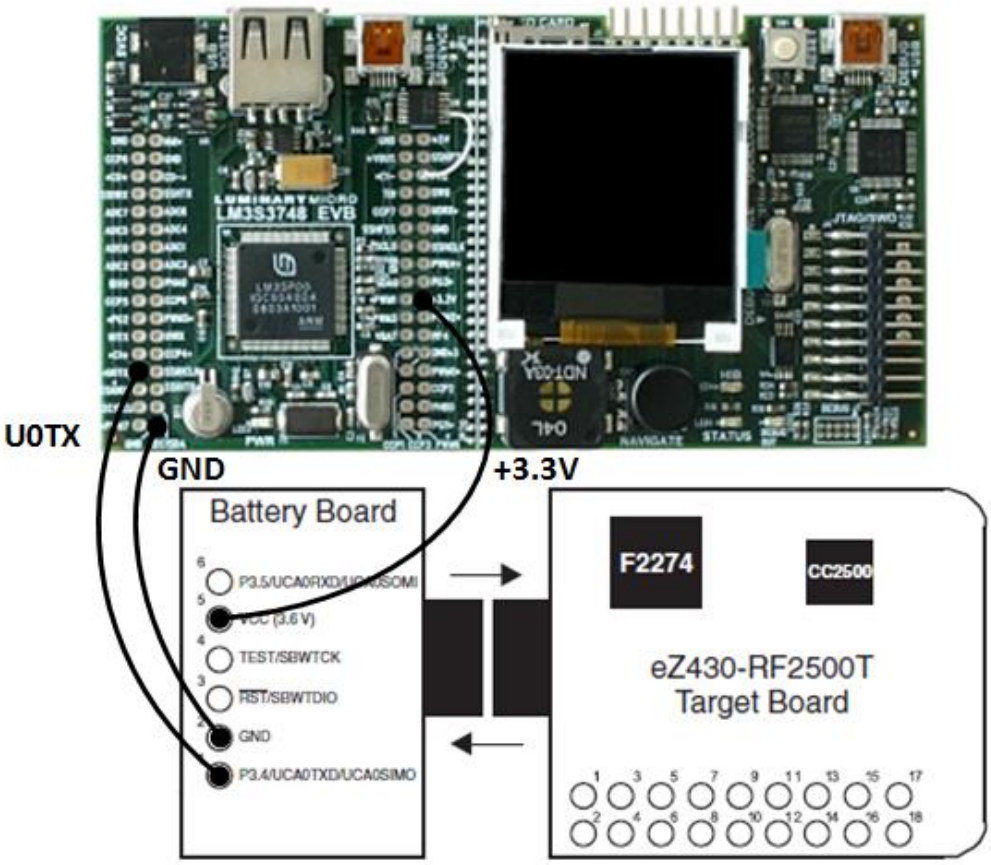
Slika 21. *Prikaz spajanja modula za bežično slanje podataka na razvojni sustav sa LM3S3748 mikrokontrolerom*

Na slici 21 je prikazan način spajanja razvojnog sustava sa LM3S3748 mikrokontrolerom sa modulom za bežičnu komunikaciju. UCA0RXD pin MSP430 mikrokontrolera na modulu za odašiljanje se spaja na U1TX (UART1) pin LM3S3748 mikrokontrolera. Modul i razvojni sustav komuniciraju RS-232 komunikacijom brzinom od 2 Mbps. MSP430 i LM3S3748 mikrokontroler podržavaju proizvoljne brzine serije. Na modul se još dovodi napajanje od 3.3V i GND.

Drugi dio razvojnog sustava EZ430-RF2500 na slici je prikazan da je spojen preko sučelja za *debugiranje* na računalo. Međutim, modul za *debugiranje* podržava maksimalnu brzinu serijske komunikacije od 9600 bps, što je premalo u odnosu na 2 Mbps kojom brzinom bi se primljeni podaci trebali slati na računalo. Zbog toga modul za bežično primanje podataka ne možemo spojiti na računalo preko modula za *debugiranje*. U tu svrhu se demonstracije radi modul za bežično primanje podataka spaja na UART0 port razvojnog sustava sa LM3S3748 mikrokontrolerom. Sklopovlje razvojnog sustava podržava brzine serije do 2 Mbps i uz to računalo prepoznaje modul za bežično primanje podataka kao virtualni COM port. Tako podatke koje šaljemo sa modula računalo prepoznaje isto kao da su podaci koje šalje LM3S3748 mikrokontroler. Iz navedenog razloga se kod serijske veze između sustava i modula za odašiljanje koristi UART1 port jer je UART0 rezervirani za komunikaciju prema računalu.

Navedeno rješenje se ne bi moglo primijeniti ako bi LM3S3748 istovremeno komunicirao sa modulom za odašiljanje preko UART1 i sa računalom preko UART0 porta, budući da sada modul za primanje podataka i LM3S3748 mikrokontroler dijele zajednički UART0 port prema računalu. Način spajanja modula za bežično primanje podataka i razvojnog sustava je prikazan na slici 22.

UCA0TXD pin MSP430 mikrokontrolera na modulu za bežično primanje podataka se spaja na U0TX (UART0) pin LM3S3748 mikrokontrolera na razvojnom sustavu. Sklopovlje razvojnog sustava podržava brzinu serijske komunikacije od 2 Mbps te se MSP430 mikrokontroler može podesiti na navedenu brzinu. Računalo se spaja sa razvojnim sustavom preko USB *debug* sučelja pa prepoznaje modul kao virtualni COM port.



Slika 22. *Prikaz spajanja modula za bežično primanje podataka na razvojni sustav sa LM3S3748 mikrokontrolerom*

6.5.3. Model bežične komunikacije (*peer-to-peer*)

Peer-to-peer je komunikacijski model kod kojeg svaka strana predstavlja inteligentnu radnu stanicu. Stanice imaju iste mogućnosti, pronalaze druge stanice te komuniciraju s njima izravno, bez potrebe autorizacije na nekom centralnom poslužitelju.

Povezivanje ili *linking* je način na koji se uspostavlja *peer-to-peer* komunikacija. Kod povezivanja jedan od uređaja sluša dok drugi šalje *link* poruku (*link message*). *Link* poruka sadrži *link token* (4 bajtni objekt) koji se koristi na strani primatelja za validaciju pošiljatelja i potreban je za pristup mreži.

Pridruživanje ili *joining* je akcija podržana samo kod mreža koje sadrže pristupnu točku (AP, *Access Point*). Predstavlja proces kod kojeg jedna strana (ED, *End Device*) dobije

dopuštenje za pristup mreži koju je uspostavila pristupna točka (AP). Povezivanje se vrši prije bilo koje druge akcije. Strana koja želi pristupiti mreži šalje poruku za pridruživanje (*join message*) koja sadrži *join token* (4 bajtni objekt). Ako se *join token* poklapa sa *token*-om koji AP očekuje, AP odgovara i šalje ED informacije o mreži, a između ostalog i *link token* potreban za pristup mreži. Nakon pristupa mreži svaka stanica se ponaša kao inteligentna radna stanica, jedna drugu ne kontrolira ni ograničava promet. Stoga se nakon povezivanja može govoriti o *peer-to-peer* komunikaciji.

Pridruživanje ili *joining* kod kojeg je nužno da postoji pristupna točka je neophodno samo kod uspostavljanja veze te može predstavljati ograničenje kod "prave" *peer-to-peer* komunikacije.

6.6. Inicijalizacija MSP430F2274 mikrokontrolera

Modul za odašiljanje i modul za primanje podataka su identični, sadrže iste MSP430 mikrokontrolere i CC2500 primopredajnik. Svaki modul se posebno programira. Jedan služi za bežično odašiljanje podatka, a drugi za primanje.

6.6.1. Modul za odašiljanje

Modul za odašiljanje prima podatke sa LM3S3748 mikrokontrolera preko serijske RS-232 komunikacije brzinom od 2 Mbps. Mikrokontroler MSP430F2274 se inicijalizira za rad na taktu od 16 MHz. Brzina serijske RS-232 komunikacije se podešava na 2 Mbps. Omogućuje se prekid na primanje podataka (RX) serijske komunikacije. Programski odsječak početne inicijalizacije je dan u nastavku.

```
////////////////////////////////////////////////////////////////////////////////
// Početna Inicijalizacija MSP430F2274 mikrokontrolera na odašiljačkom modulu //////
////////////////////////////////////////////////////////////////////////////////
BSP_Init();                          // Inicijalizacija modula (CC2500 primopredajnika)
DCOCTL  = CALDCO_16MHZ;              // 16MHz takt
BCSCTL1 = CALBC1_16MHZ;
// Inicijalizacija serije
P3SEL |= 0x30;                       // Pinovi P3.4 i P3.5 = USCI_A0 TXD/RXD
UCA0CTL1 = UCSSEL_2;                 // SMCLK
UCA0BR0 = 0x08;                      // Brzina 2000000 bps
UCA0BR1 = 0x00;
UCA0MCTL = UCBRS_0;
UCA0CTL1 &= ~UCSWRST;                // Inicijaliziraj USCI state machine
IE2 |= UCA0RXIE;                     // Omogući prekid na USCI_A0 RX
__enable_interrupt();                // Globalno omogući prekide
```

Prije uspostavljanja veze sa drugim modulom generira se proizvoljna adresa odašiljačkog uređaja. Kod generiranja je bitno da prvi bajt adrese nije jednak 0xFF ili 0x00 jer se na prijamnoj strani (RX) interpretira kao sveodredišna adresa (*broadcast addresses*). Dio koda za generiranje adrese je dan u nastavku.

```
////////////////////////////////////////////////////////////////////////////////
// Generiranje i postavljanje adrese uređaja ///////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
if (Flash_Addr[0] == 0xFF && Flash_Addr[1] == 0xFF &&    // ako adresa još nije
    Flash_Addr[2] == 0xFF && Flash_Addr[3] == 0xFF )     // postavljena
{
        CreateRandomAddress();              // generiraj novu adresu
}
 lAddr.addr[0] = Flash_Addr[0];
 lAddr.addr[1] = Flash_Addr[1];
 lAddr.addr[2] = Flash_Addr[2];
 lAddr.addr[3] = Flash_Addr[3];

 // postavi novu adresu uređaja
 SMPL_Ioctl (IOCTL_OBJ_ADDR, IOCTL_ACT_SET, &lAddr);
```

Nakon što je adresa postavljena, odašiljački uređaj pokušava uspostaviti vezu sa prijamnim uređajem te šalje poruku za pridruživanje (*join message*). Implementirano je da u slučaju nemogućnosti povezivanja uređaj čeka 500 ms te pokuša ponovo. Za vrijeme čekanja crvena led dioda treperi. Nakon što AP prepozna ED, AP vraća *link token*. Kao indikacija uspješnog povezivanja upale se zelena i crvena LED dioda. Nakon toga ED šalje *link* poruku i u slučaju potvrde od strane AP, pristupa mreži. Programski odsječak povezivanja i pristupanja mreži je dan u nastavku.

```
////////////////////////////////////////////////////////////////////////////////
// Uspostavljanje veze sa prijamnim uređajem ///////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
while (SMPL_SUCCESS != SMPL_Init(0))       // Šalji join message
 {                                         // Ako nije došlo do uspješnog povezivanja
  toggleLED(2);                            // Ugasi/upali crvenu LED diodu
  SPIN_ABOUT_500MS;                        // Pričekaj 500ms
 }

// Upali crvenu i zelenu LED diodu kao indikacija uspješnog povezivanja
if (!BSP_LED2_IS_ON())
  toggleLED(2);
if (!BSP_LED1_IS_ON())
  toggleLED(1);

// Bezuvjetno pristupanje mreži
 linkTo();
```

Kad je ED inicijaliziran, čeka podatke koje mu razvojni sustav šalje preko serijske komunikacije. U prekidnom potprogramu je implementiran brojač koji broji primljene bajtove slike sa razvojnog sustava. Nakon primljena 33 bajta slike ED šalje paket AP. Prekidni potprogram je dan u nastavku.

```
//////////////////////////////////////////////////////////////////////////////
// Prekidni potprogram serijske veze (USCIA) ////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
int brojac=0;   // Resetiraj brojač

#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
        msg[brojac] = UCA0RXBUF;   // Kopiraj primljeni bajt poruke
        brojac++;                  // Uvećaj brojač
        if (33==brojac)            // Ako je primljeno 33 bajta
        {
          SMPL_Send(sLinkID1, msg, sizeof(msg));        // Šalji paket
          brojac=0;                // Resetiraj brojač
        }
}
```

Maksimalna teoretska brzina bežične komunikacije je 250 kbps, no u praksi efektivna brzina slanja podataka je oko 80 kbps jer se uz korisničke podatke šalje perambula i sinkronizacijski okviri. Vrijeme koje prođe od trenutka slanja paketa do trenutka kad podaci budu spremni za korištenje na prijamnoj strani izmjereno je i iznosi oko 3.2 ms. Korisnički podaci u paketu zauzimaju 33 bajta. Efektivna brzina kojom se odašilju podaci slike iznosi $(33 \cdot 8)/0.0032 = 82\,500$ bps. Dakle vrijeme koje je potrebno za slanje jedne slike od 19 008 bajta iznosi $(19\,008 \cdot 8)/82\,500 = 1.8432$ sekunde. U proračunu nije uzeto u obzir vrijeme potrebno da se slika učita sa kamere, prilagodi, pošalje na modul za odašiljanje i sa modula za bežično primanje podataka pošalje na računalo.

6.6.2. Modul za primanje podataka

Isto kao i kod modula za odašiljanje mikrokontroler MSP430F2274 se inicijalizira za rad na taktu od 16 MHz. Brzina serijske RS-232 komunikacije se podešava na 2 Mbps. Programski odsječak početne inicijalizacije je dan u nastavku.

```
////////////////////////////////////////////////////////////////////////////
// Početna Inicijalizacija MSP430F2274 mikrokontrolera na prijamnom modulu ////////
////////////////////////////////////////////////////////////////////////////
BSP_Init();                         // Inicijalizacija modula (CC2500 primopredajnika)
DCOCTL  = CALDCO_16MHZ;             // 16MHz takt
BCSCTL1 = CALBC1_16MHZ;
// Inicijalizacija serije
P3SEL |= 0x30;                      // Pinovi P3.4 i P3.5 = USCI_A0 TXD/RXD
UCA0CTL1 = UCSSEL_2;                // SMCLK
UCA0BR0 = 0x08;                     // Brzina 2000000 bps
UCA0BR1 = 0x00;
UCA0MCTL = UCBRS_0;
UCA0CTL1 &= ~UCSWRST;               // Inicijaliziraj USCI state machine
```

Nakon početne inicijalizacije mikrokontrolera, inicijalizira se i postavlja mreža za bežičnu komunikaciju te se čeka ED da pošalje zahtjev za pridruživanje (*join*). Za vrijeme čekanja zelena i crvena LED dioda su upaljene. Nakon što ED pošalje zahtjev za pridruživanje AP vraća potvrdan odgovor i čeka da ED pristupi mreži. Kao indikacija uspješnog pristupa mreži gasi se crvena LED dioda, a zelena i dalje svijetli. Programski odsječak je dan u nastavku.

```
////////////////////////////////////////////////////////////////////////////
// Postavljanje mreže i čekanje da ED pristupi ////////////////////////////
////////////////////////////////////////////////////////////////////////////
 SMPL_Init(sCB);                    // Postavi mrežu za bežičnu komunikaciju
 BSP_TURN_ON_LED1();                // Upali zelenu LED diodu
 BSP_TURN_ON_LED2();                // Upali crvenu LED diodu

while (1)
 {
       if (sJoinSem && (sNumCurrentPeers < 1))     // Ako je ED poslao zahtjev za
       {                                           // povezivanje i ako još nije povezan
        while (1)       // Čekaj dok ED ne pristupi mreži
        {
              if (SMPL_SUCCESS == SMPL_LinkListen(&sLID[sNumCurrentPeers]))
        break;
        }
        if (BSP_LED2_IS_ON())      toggleLED(2);    // Ugasi crvenu LED diodu
        if (!BSP_LED1_IS_ON())      toggleLED(1);   // Upali zelenu LED diodu

        sNumCurrentPeers++;                         // Označi da je jedan ED povezan
        BSP_ENTER_CRITICAL_SECTION(intState);
        sJoinSem--;                                 // Označi da je ED povezan
        BSP_EXIT_CRITICAL_SECTION(intState);
       }
 }
```

Kad AP primi paket koji mu je poslao ED poziva se potprogram u kojem se primljeni podaci šalju preko serijske komunikacije i zajedničkog U0TX priključka te sklopovlja razvojnog sustavu sa LM3S3748 mikrokontrolerom na računalo. Za vrijeme slanja podataka preko serije svijetli crvena LED dioda. Isti potprogram se poziva i kad AP primi zahtjev za povezivanje. U tom slučaju javlja se glavnom programu da je ED poslao zahtjev za povezivanjem i izlazi iz potprograma. Zahtjev za povezivanje se obrađuje u glavnom programu. Potprogram je dan u nastavku.

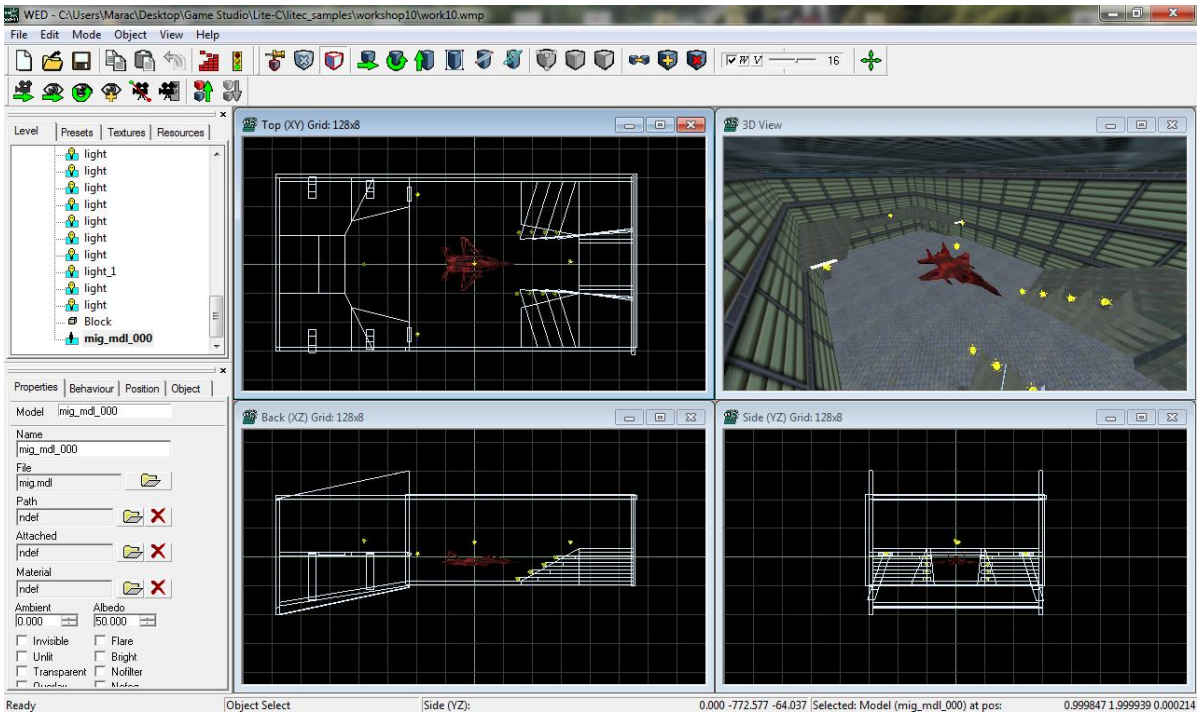
```
////////////////////////////////////////////////////////////////////////////////
// Prekidni potprogram serijske veze (USCIA) ///////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
static uint8_t sCB(linkID_t lid)
{
 if (lid)          // Ako je primljen paket
 {
        uint8_t    msg[33], len;
        if (SMPL_SUCCESS == SMPL_Receive(sLID[0], msg, &len))
        {
                BSP_TOGGLE_LED2();          // Upali crvenu LED diodu
                TXString( (char*)msg, len );    // Pošalji primljene podatke preko serije
                BSP_TOGGLE_LED2();          // Ugasi crvenu LED diodu
        }
 }
 else   sJoinSem++;  // Ako je primljen zahtjev za povezivanjem javi glavnom programu
 return 0;
}
```

7. Programska podrška na računalu

Zadaća software-a na računalu je da pročita sliku koja mu se šalje preko RS-232 serijske komunikacije i prikaže je na svom zaslonu.

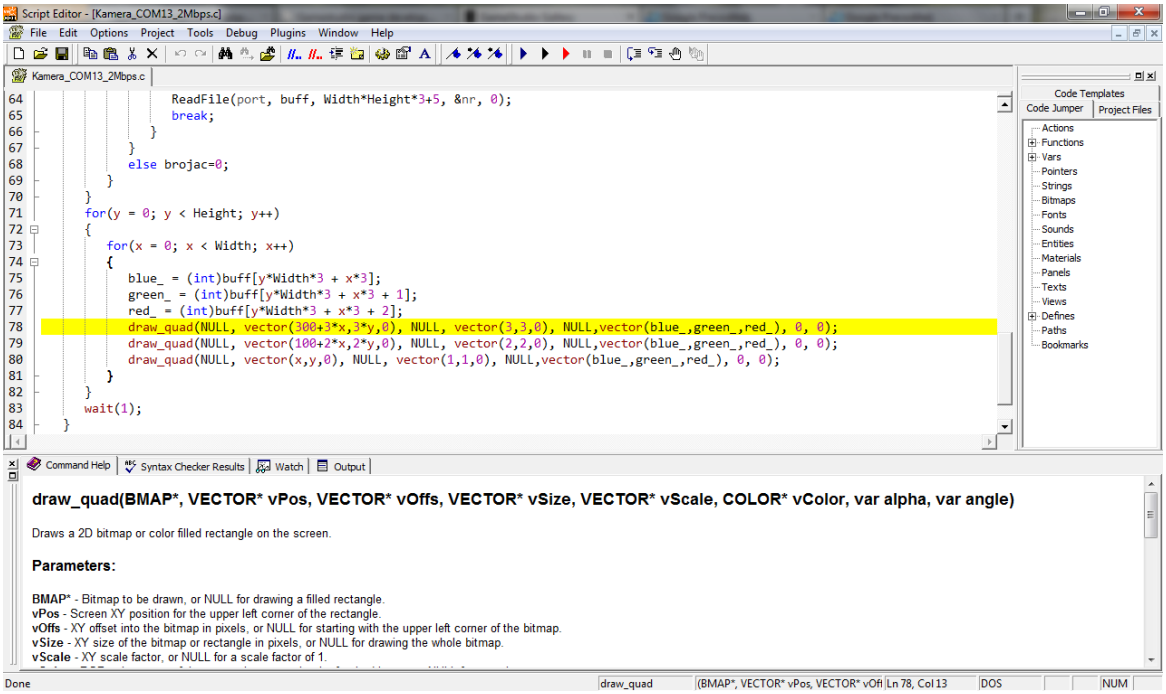
7.1. 3D Gamestudio

Odabrani programski paket za izradu potrebnog sofware-a je 3D Gamestudio (3DGS) zbog svojeg jednostavnog i moćnog grafičkog sučelja. 3DGS je razvojno okruženje koje omogućuje korisnicima dizajniranje i stvaranje 2D i 3D igara, ali i drugih aplikacija sa raznim vizualnim efektima. Uključuje mnoštvo alata i pomoćnih programa za razvoj kao što su *model/terrain* editor, a *level* editor, *script* editor i debugger. Kod kompleksnijih programa može se koristiti integrirani programski jezik nazvan Lite-C po kojem je 3DGS karakterističan ili vanjski jezici za razvoj aplikacija kao što su Visual C++ ili Borland Delphi.



Slika 23. *3D Gamestudio (level editor) korisničko sučelje*

Kod 3DGS-a naglasak je na prikazivanju grafičkih efekata vrlo efikasno uz što manje utrošenih resursa, pa je stoga prikladan za iscrtavanje slike na zaslonu. Program za prikaz slike je pisan u Lite-C jeziku u *script* editoru.



Slika 24. *3D Gamestudio (script editor i debugger) korisničko sučelje*

U programu se prvo inicijalizira virtualni COM port koji je u ovom slučaju označen kao COM13. Postavlja se brzina serijske komunikacije od 2 Mbps. Nakon toga program čeka podatke koji mu se šalju. Nakon što pročita 19 008 bajta sa serije prvo provjerava zadnja 3 koji se šalju radi sinkronizacije i postavljeni su na vrijednost 0x12. Ako se zadnja 3 bajta podudaraju sinkronizacija je uspjela i slijedi prikaz slike. U slučaju da se ne podudaraju došlo je do desinkronizacije pa se dalje čitaju nadolazeći podaci bajt po bajt tako dugo dok se ne pojave uzastopni 3 koji su postavljani na 0x12. Pretpostavlja se da je sinkronizacija uspostavljena i slijedi prikazivanje slike. Usporedbe radi se ista slika prikazuje 3 puta. Prva slika je izvorne veličine 88 x 72 piksela. Druga slika je veličine $2 \cdot 88$ x $2 \cdot 72$ = 176 x 144 piksela kod koje je veličina jednog piksela povećana sa 1x1 na 2x2, odnosno svaki piksel izvorne slike se prikazuje u skupini od 4 istovrsnih piksela. Treća slika je veličine $3 \cdot 88$ x $3 \cdot 72$ = 264 x 216 piksela. Slično kao i kod druge slike veličina jednog piksela povećana je sa 1x1 na 3x3, odnosno svaki piksel izvorne slike se prikazuje u skupini od 9 piksela. Kod programa je dan u nastavku.

```
////////////////////////////////////////////////////////////////////////////////
// Inicijalizacija serijske komunikacije ///////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
long *dcb  = malloc(sizeof(long)*8);          // Alociraj memoriju potrebnu za seriju
        memset(dcb,0,sizeof(long)*8);

port = CreateFile("\\\\.\\COM13",    // COM13 port
            GENERIC_READ | GENERIC_WRITE,
            0,                       // must be opened with exclusive-access
            NULL,                    // no security attributes
            OPEN_EXISTING,           // must use OPEN_EXISTING
            0,                       // not overlapped I/O
            NULL                     // hTemplate must be NULL for comm devices
            );
BuildCommDCB( "2000000,n,8,1", dcb);          // Postavi brzinu serije na 2 Mbps
SetCommState(port, dcb);                      // Inicijaliziraj
```

```
////////////////////////////////////////////////////////////////////////////////
// Čitanje i provjera sinkronizacijskih bajtova ////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
ReadFile(port, buff, Width*Height*3, &nr, 0);        // Čitaj 19008 B sa serije
// Provjera sinkronizacijskih bajtova
if ((buff[19008-1]!=0x12)||(buff[19008-2]!=0x12)||(buff[19008-3]!=0x12))
{                                                    // Ako se ne poklapaju
    while(1)
    {
        ReadFile(port, buff, 1, &nr, 0);             // Čitaj slijedeći bajt sa serije
        if (0x12==buff[0])                           // Provjeri ako je sinkronizacijski
        {
                brojac++;              // Ako se poklapa uvećaj brojač za 1
                if (3==brojac)         // Ako su nađena 3 uzastopna sinkronizacijska bajta
                {
                        // Čitaj  novih 19008 B sa serije
                        ReadFile(port, buff, Width*Height*3, &nr, 0);
                        break;   // Pretpostavka da je nova učitana slika sinkronizirana
                }
        }
        else brojac=0;             // Ako se ne poklapa resetiraj brojač
    }
}
```

```
////////////////////////////////////////////////////////////////////////////////
// Prikaz slike na zaslonu ////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
for(y = 0; y < Height; y++)
{
      for(x = 0; x < Width; x++)
      {
          blue_ = (int)buff[y*Width*3 + x*3];          // Čitaj plavu boju trenutnog piksela
          green_ = (int)buff[y*Width*3 + x*3 + 1];  // Čitaj zelenu boju trenutnog piksela
          red_ = (int)buff[y*Width*3 + x*3 + 2];      // Čitaj crvenu boju trenutnog piksela

         // Crtaj piksel na svakoj od 3 slike (3x3, 2x2, 1x1)
          draw_quad(NULL, vector(300+3*x,3*y,0), NULL, vector(3,3,0),
                          NULL,vector(blue_,green_,red_), 0, 0);
          draw_quad(NULL, vector(100+2*x,2*y,0), NULL, vector(2,2,0),
                          NULL,vector(blue_,green_,red_), 0, 0);
          draw_quad(NULL, vector(x,y,0), NULL, vector(1,1,0),
                          NULL,vector(blue_,green_,red_), 0, 0);
      }
}
wait(1);          // Pričekaj jedan frame
```

8. Zaključak

Iako bežičnom komunikacijom računalo može biti udaljeno od razvojnog sustava za upravljanje kamerom, ipak se pokazalo da je žičana veza puno bolja. Kod bežične glavni nedostatak predstavlja relativno niska brzina prijenosa koja sa korištenim primopredajnikom može doseći maksimum od oko 85 kbps. U našem slučaju bežična komunikacija predstavlja "usko grlo" kod prijenosa, što ima za posljedicu vrlo malu brzinu osvježavanja slike na zaslonu od oko 0.3 slike po sekundi. Rezultat je otežano praćenje slike u stvarnom vremenu.

Kod žičane komunikacije ne postoji taj problem. Maksimalna brzina prijenosa iznosi oko 2 Mbps pa se postiže brzina osvježavanja od nekoliko slika po sekundi. Jedino ograničenje je duljina žice odnosno kabla kojim je razvojni sustav spojen na računalo i preko kojeg se vrši komunikacija.

Rješenje problema "uskog grla" kod bežične komunikacije predstavljala bi kompresija slike. Time bi se znatno ubrzao prijenos, no problematična bi mogla postati brzina rada procesora koji bi komprimirao sliku. Postoji vjerojatnost da bi kompliciranji algoritmi kompresije mogli uzrokovati čak dulje vrijeme obrade slike nego sam prijenos na računalo. Time se efektivno vrijeme osvježavanja slike na zaslonu ne bi promijenilo, no bitno bi smanjilo opterećenje primopredajnika koji bi proporcionalno faktoru kompresije slao manje podataka. Opterećenje primopredajnika nije najveći problem. Veći problem je praćenje slike u stvarnom vremenu, što se ovom metodom može do neke mjere popraviti. Još jedno rješenje bi bilo korištenje primopredajnika sa većom propusnošću (*bandwith*). Navedeno rješenje bi povećalo brzinu osvježavanja slike na zaslonu, no sukladno s tim bi bitno povećalo i cijenu uređaja.

Drugi način na koji bi se brzina osvježavanja mogla povećati je korištenje kamere koja programski ili hardware-ski komprimira sliku i šalje je na razvojni sustav u vrlo kratkom vremenu. Time bi se izbjeglo vrijeme utrošeno na programsko komprimiranje slike u procesoru razvojnog sustava, smanjilo bi se zauzeće memorije te samim tim i vrijeme prijenosa slike na računalo. Glavni nedostatak bi i u ovom slučaju bila povećana cijena uređaja.

Kod video portafona koji se komercijalno proizvode bitna značajka je i prijenos zvuka. Nije implementiran prijenos zvuka budući da korištena kamera nema ugrađen mikrofon.

Navedeni problem bi se mogao riješiti korištenjem dodatnog modula sa mikrofonom koji bi komunicirao sa LM3S3748 mikrokontrolerom na razvojnom sustavu. Slično rješenje dalo bi se implementirati korištenjem mikrofona i pretpojačala čiji izlaz bi se spojio na 10-bitni analogno digitalni pretvornik (ADC) na LM3S3748 mikrokontroleru. Nadalje zvuk bi se zajedno sa slikom slao na računalo. Problem se u tom slučaju opet javlja kod bežične komunikacije zbog male brzine prijenosa. Budući da bi se u navedenom slučaju trebalo slati više podataka, znatno bi se usporila brzina osvježavanja slike na računalo te reprodukcija zvuka. Iz spomenutih razloga prvo bi se trebao riješiti problem "uskog grla" kod bežične komunikacije pa nakon toga implementirati prijenos zvuka.

LITERATURA

[1] Texas Instruments, Stellaris LM3S3748 Evaluation Kit User's Manual, 2010

[2] Texas Instruments, Stellaris LM3S3748 Microcontroller DATA SHEET, 2010

[3] Texas Instruments, Luminary Micro, EK-LM3S3748 Firmware Development Package User's Guide, 2008

[4] Texas Instruments, Luminary Micro, Stellaris ROM User's Guide, 2008

[5] Texas Instruments, Luminary Micro, Stellaris Peripheral Driver Library User's Guide, 2008

[6] Texas Instruments, Luminary Micro, Stellaris Graphics Library User's Guide, 2008

[7] Texas Instruments, Luminary Micro, Stellaris USB Library User's Guide, 2008

[8] Texas Instruments, eZ430-RF2500 Development Tool User's Guide, 2009

[9] Texas Instruments, MSP430x22x2, MSP430x22x4 MIXED SIGNAL MICROCONTROLLER, 2011

[10] Texas Instruments, MSP430x2xx Family User's Guide, 2008

[11] Texas Instruments, CC2500 DATA SHEET, 2009

[12] IAR Embedded Workbench IDE for MSP430 User Guide for Texas Instruments' MSP430 Microcontroller Family, 2008

[13] Texas Instruments, Inc. San Diego, California USA, SimpliciTI Application Programming Interface, 2009

[14] Friedman Larry, SimpliciTI Developers Notes, SimpliciTI: Simple Modular RF, February 1, 2008

[15] Friedman Larry, SimpliciTI Frequency Agility Description, Application Note: SimpliciTI Frequency Agility Description, February 1, 2008

[16] PAC207B SINGLE-CHIP CIF CMOS IMAGE SENSOR WITH USB INTERFACE, 2003

[17] PAC207, 22.07.2007, http://wiki.osdev.info/?PAC207

[18] Hans de Goede, Spca50x-devs, 19.04.2008, *[Spca50x-devs] PATCH: gspcav2-0.0.28 pac207 support*, http://lists-archives.org/spca50x-devs/01638-patch-gspcav2-0-0-28-pac207-support.html

Realizacija video portafona korištenjem ugradbenog računala

Sažetak

U radu je opisano kako je korištenjem razvojnog sustava sa LM3S3748 mikrokontrolerom moguće realizirati video portafon. Razvojni sustav inicijalizira web kameru (model DC-4110) te njome upravlja. Mikrokontroler podržava USB Host način rada pa se slika sa kamere u razvojni sustav učitava USB komunikacijom. Na učitanoj slici su pikseli raspoređeni u tzv. Bayer matricu piksela. Prije nego se može prikazati na LCD display-u potrebno je sliku prilagoditi jer učitani format nije pogodan za prikaz.

Opisana su dva načina povezivanja razvojnog sustava sa računalom. Prvi i jednostavniji način je žičanim putem, a drugi bežičnim putem. Žičani prijenos slike na računalo je realiziran serijskom RS-232 komunikacijom. Za bežičnu komunikaciju se koriste dva identična modula sa MSP430F2274 mikrokontrolerima i CC2500 primopredajnikom. Nakon što se slika prenese na računalo prikazuje se na zaslonu.

Ključne riječi: LM3S3748 mikrokontroler, EZ430-RF2500 razvojni sustav, MSP430F2274 mikrokontroler, PAC207B optički CMOS senzor, USB komunikacija, *Bayer* piksel raster, RS-232 komunikacija, bežična *peer-to-peer* komunikacija.

The realisation of video intercom using embedded computers

Abstract

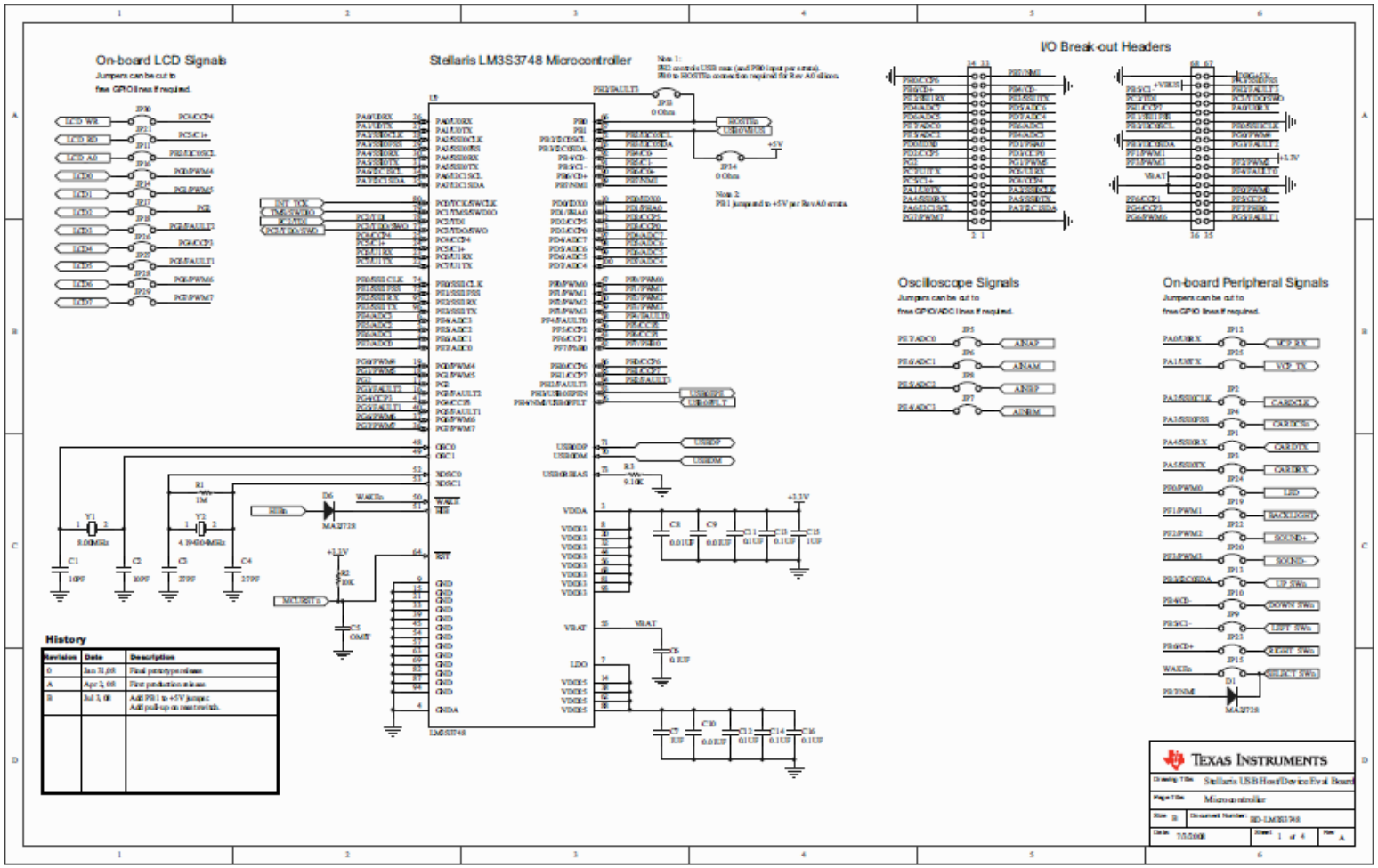
The work describes how to implement a video intercom using a development tool with the LM3S3748 microcontroller. The development tool initializes a webcam (model DC-4110), and manages it. The microcontroller supports USB Host mode, so the picture is read from the camera and stored into the development system via the USB communication. The uploaded image pixels are stored into a so-called Bayer pixels pattern. Before it can be

displayed on the LCD display it is necessary to adjust the picture because the loaded format is not compatible with the display.

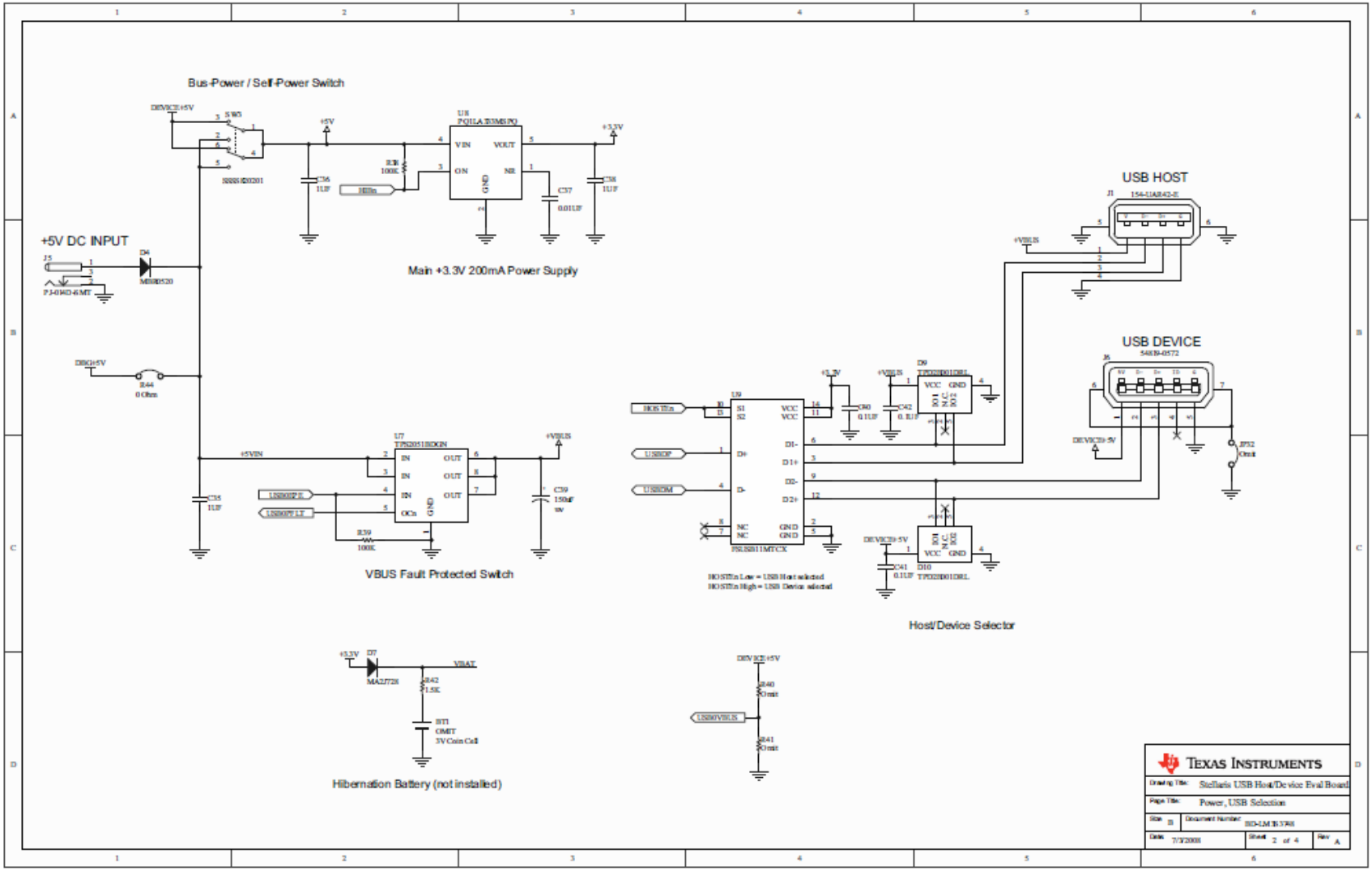
There are two ways of communication described between the development tool and the computer. The first and easier way is by wire, and the other is wireless. Wired image is transferred to the computer over the serial RS-232 communication protocol. For the wireless communication, there are two identical modules used with MSP430F2274 microcontrollers and CC2500 transceivers. After the image is transferred to the computer it is displayed on the computer screen.

Keywords: LM3S3748 microcontroller, EZ430-RF2500 development tool, MSP430F2274 microcontroller, PAC207B optical CMOS sensor, USB communication, Bayer pixel array, RS-232 communication, wireless peer-to-peer communication.

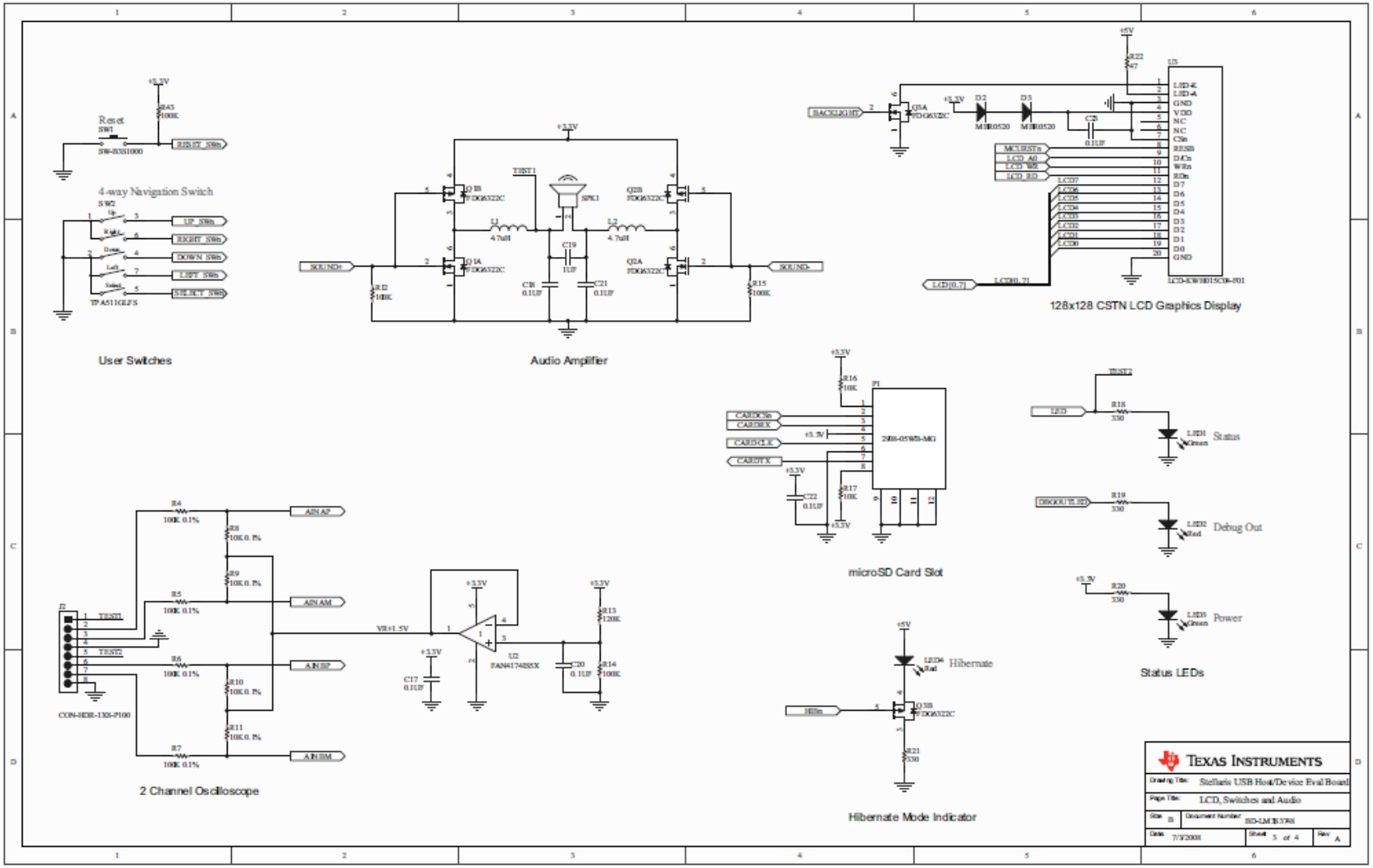
ELEKTRIČKE SHEME



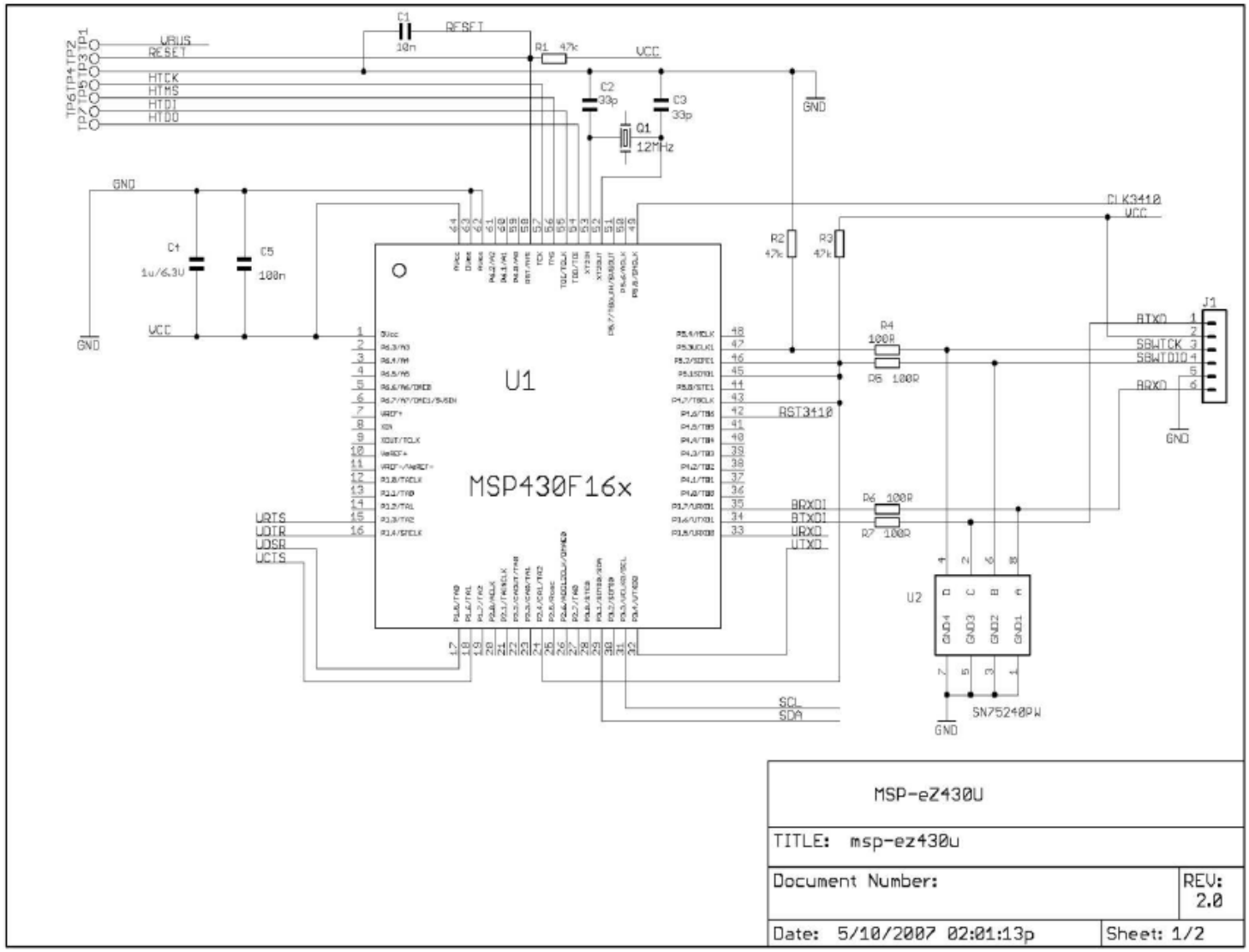
Shema 1. *Razvojni sustav sa LM3S3748 mikrokontrolerom, spajanje mikrokontrolera*



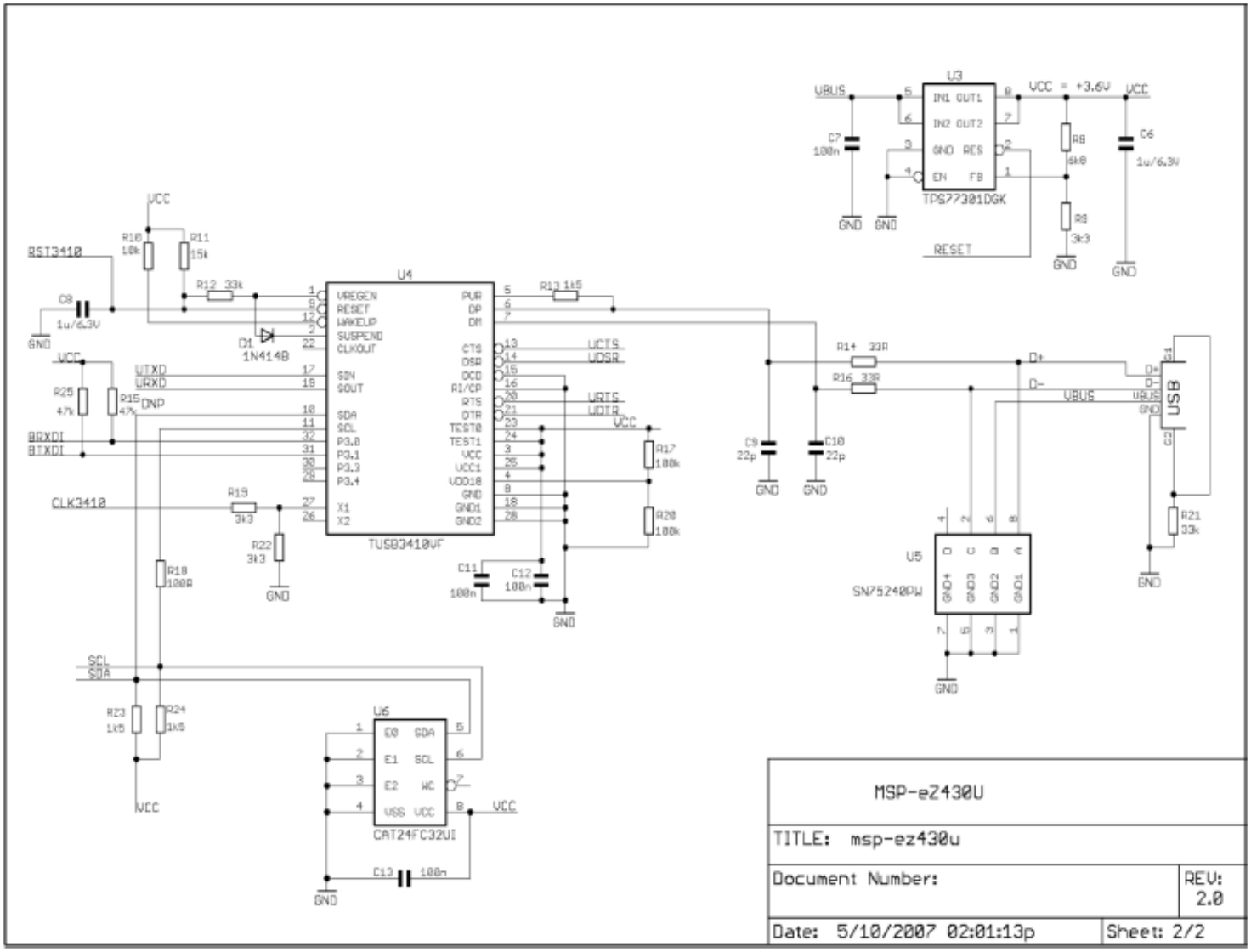
Shema 2. *Razvojni sustav sa LM3S3748 mikrokontrolerom, spajanje regulatora napajanja i selekcija USB načina rada*



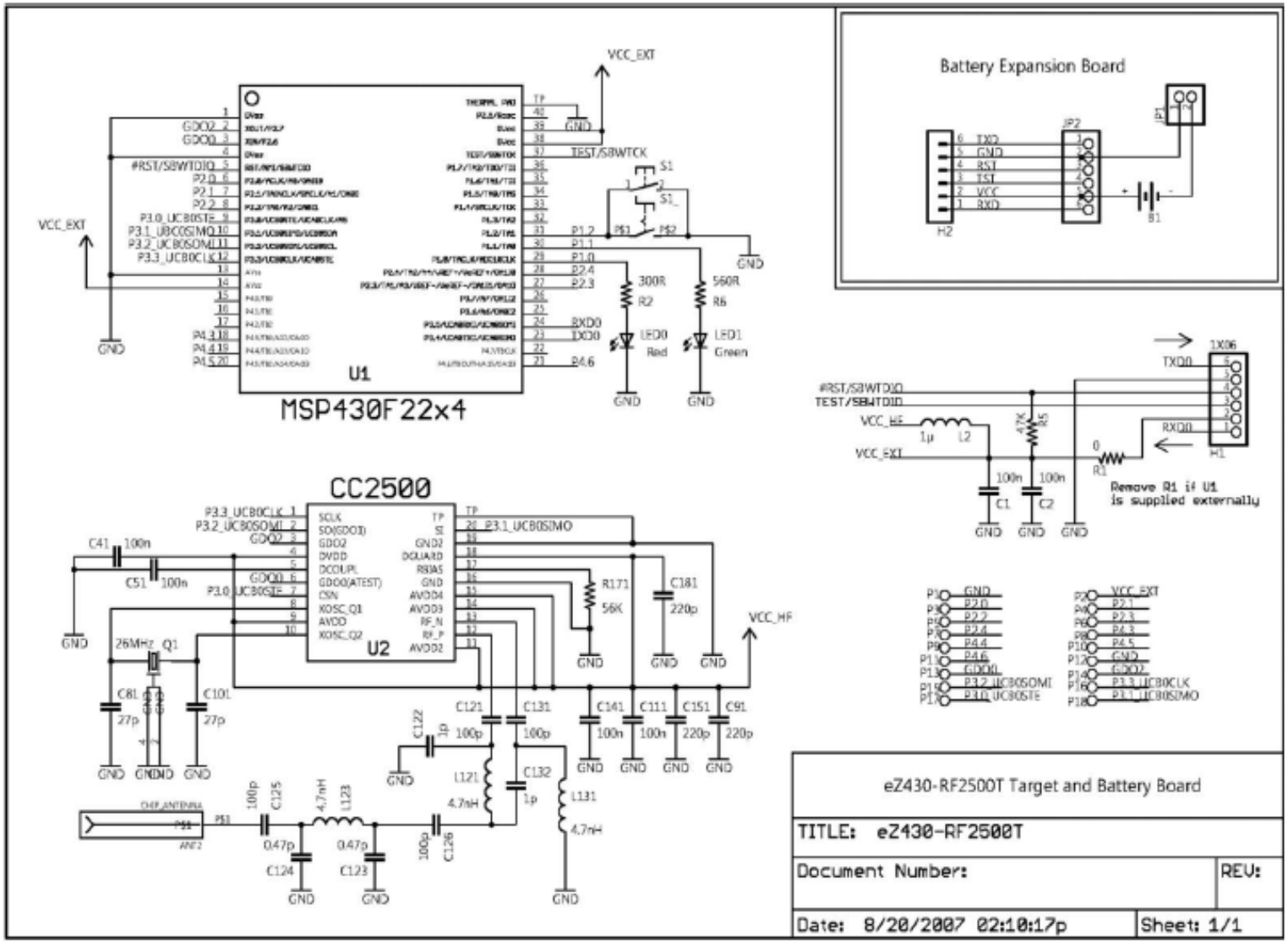
Shema 3. *Razvojni sustav sa LM3S3748 mikrokontrolerom, spajanje LCD-a, prekidača i LED dioda (ostali dijelovi se u projektu ne koriste)*



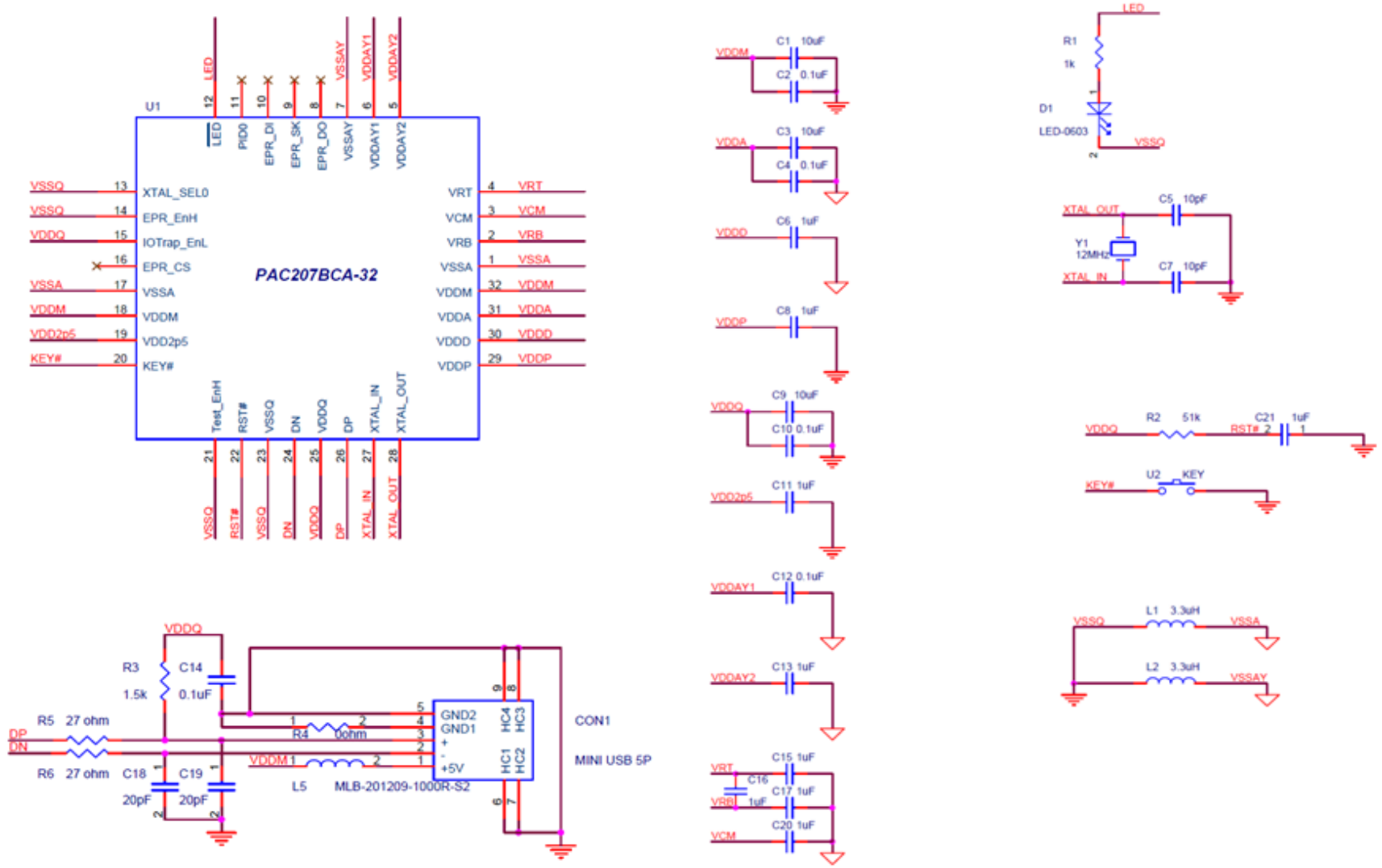
Shema 4. *eZ430-RF2500T modul sa MSP430 mikrokontrolerom, USB sučelje za debugiranje*



Shema 5. *eZ430-RF, USB sučelje za debugiranje*



Shema 5. *eZ430-RF2500T modul sa MSP430 mikrokontrolerom, Spajanje MSP430F22 mikrokontrolera i CC2500 primopredajnika*



Shema 6. *PAC207B optički senzor, preporučena shema spajanja*