Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Multimedijske arhitekture i sustavi

Laboratorij računalnog inženjerstva

Projektna dokumentacija

**Grupa 4**

**Članovi:**

**0036477215 Luka Macan**

**0036484456 Tomislav Bićanić**

**0036479176 Marko Filipović**

**0036480644 Jurica Martinčević**

**0036475593 Jure Knezović**

**Sadržaj**

[1 Uvod 3](#_Toc504622040)

[2 Opis sustava 4](#_Toc504622041)

[3 Shema 6](#_Toc504622042)

[3.1 Konceptualna shema (kamera, PYNQ , ethernet i osobno računalo) 6](#_Toc504622043)

[3.2 Blok dizajn 7](#_Toc504622044)

[3.3 Zynq konfiguracija 8](#_Toc504622045)

[4 Korišteni algoritmi 9](#_Toc504622046)

[5 Upute za pokretanje aplikacije 12](#_Toc504622047)

[6 Literatura 13](#_Toc504622048)

# Uvod

Zadatak našeg projekta bio je osmisliti i implementirati jednostavni ugradbeni računalni sustav za snimanje fotografija.

Sustav je trebao na zahtjev aplikacije s računala primiti sliku putem OV7670 kamere, obraditi je na razvojnoj pločici PYNQ-Z1, proslijediti komprimiranu i kriptiranu sliku putem protokola Ethernet na računalo. Računalo treba putem aplikacije dekriptirati sliku te je prikazati u pripadnoj aplikaciji koja je razvijena zajedno sa ugradbenim računalnim sustavom.

Ovakav sustav bi se mogao koristiti za jednostavne aplikacije u kojima je potrebno primiti sliku samo na zahtjev korisnika, primjerice za korištenje kamere na portafonu, gdje bi se pritiskom tipke dobila slika osobe pred portafonom te se na osnovu slike može odlučiti hoće li se osobi omogućiti ulazak u zgradu ili ne.

# Opis sustava

Zadaća sustava je da na zahtjev korisnika dohvati sliku s kamere, u prihvatljivom je vremenu pošalje i prikaže u korisničkoj aplikaciji. Kako je slanje sirove slike (engl. *RAW image)* presporo, implementiran je JPEG koder koji prije slanja komprimira i kodira sliku korištenjem odgovarajućih algoritama. Format JPEG je odabran jer je to najčešće korišteni format za kompresiju slika koji nudi visoki stupanj kompresije uz dobro očuvanje kvalitete. Osim toga postoji i velik broj aplikacija koje mogu čitati slike kodirane JPEG standardom te je implementirano mnogo biblioteka u različitim programskim jezicima koje mogu dekodirati i prikazati JPEG slike.

Slanje slike odvija se TCP/IP protokolom implementiranim pomoću lwIP biblioteke. Korišten je RAW API poziv radi veće brzine. Prilikom odabira između TCP i UDP slanja paketa, odabran je TCP budući da prilikom definiranja problema nisu definirani zahtjevi na brzinu slanja, te pošto je TCP sigurniji prijenos, u vidu retransmisije i ostalih alata koji osiguravaju primitak paketa, kao i sigurniji prijenos budući da se mogu slati korisnički podatci u kodiranom zaglavlju.

Konfiguracija kamere se vrši preko SCCB sučelja koje je kompatibilno s I2C standardom. Prilikom pokretanja sustava internim registrima kamere se šalju unaprijed pripremljene optimalne vrijednosti te se nakon toga kamera postavlja da šalje piksele u YUV422 formatu.

Pikseli se s kamere dohvaćaju preko ZV porta koji je spojen na 11-bitni GPIO, a sinkronizacija sa signalima vsync, href i pclk je izvedena programski.

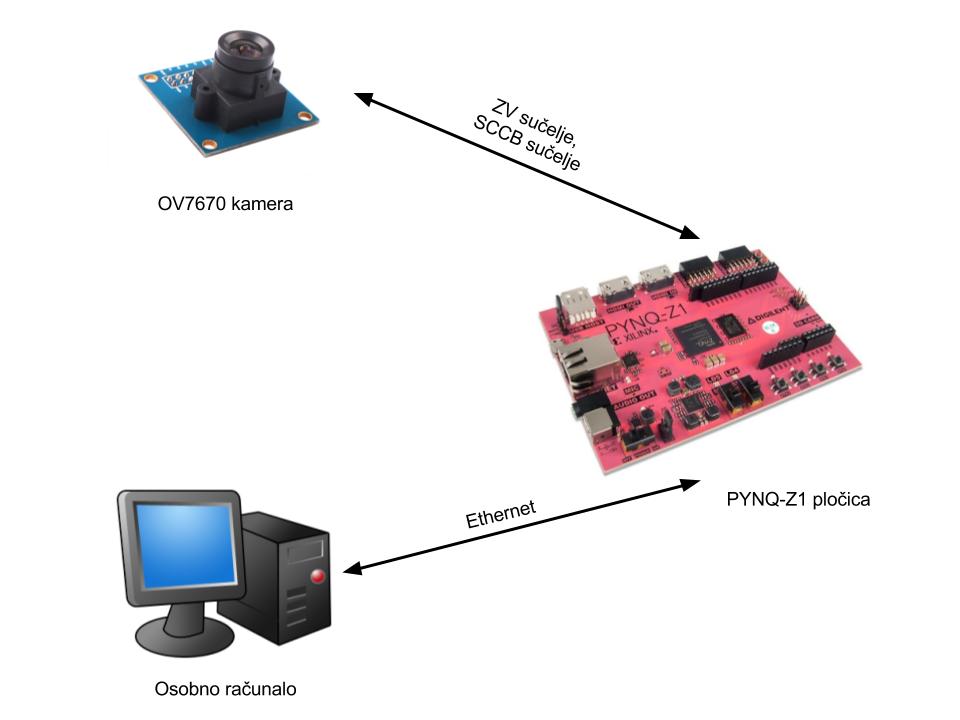
Za sigurnost podataka između računala i PYNQ pločice koristimo XOR svakog poslanog bajta sa tajnim ključem koji je poznat samo korisniku i samoj pločici.

Za izradu aplikacije za računalo je odabran programski jezik Java, koji koristi Java FX za izradu korisničkog sučelja. Programski jezik Java je odabran zbog portabilnosti koda, tj. jednom napisani kod se može izvršavati pod najzastupljenijim operacijskim sustavima kao što su macOS, Windows i Linux.

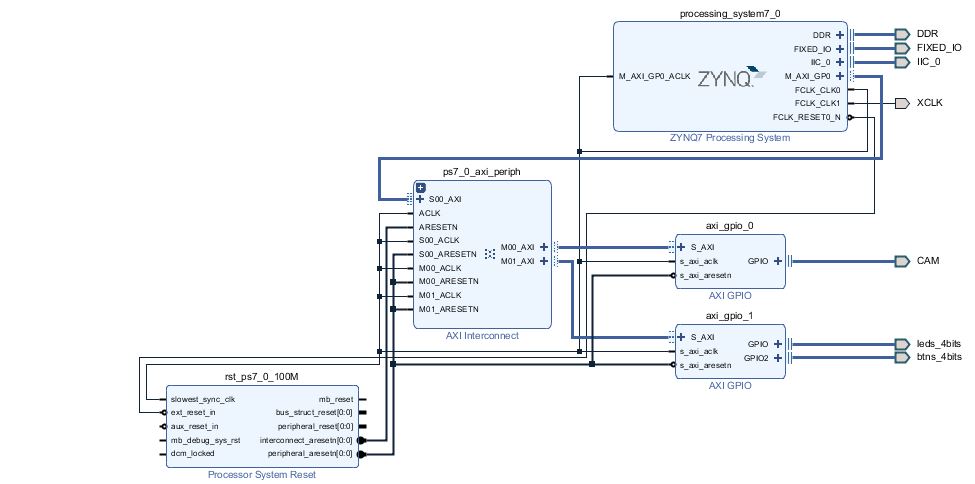
Sustav se sastoji od Pynq pločice, kamere i aplikacije. Pynq plocica pokrece lwIP server koji na zahtjev sa aplikacije pokreće dohvat slike sa kamere. Nakon što se slika dohvati s kamere, pokreće se jpeg koder, koji dobivene YUV pixele kodira u JPG format. Dobivena slika u JPG formatu se šalje aplikaciji koja je zatim prikazuje na zaslonu korištenog uređaja, te omogućuje pohranu na sami uređaj, kao i manipulaciju nad slikom.

# Shema

## Konceptualna shema (kamera, PYNQ , ethernet i osobno računalo)



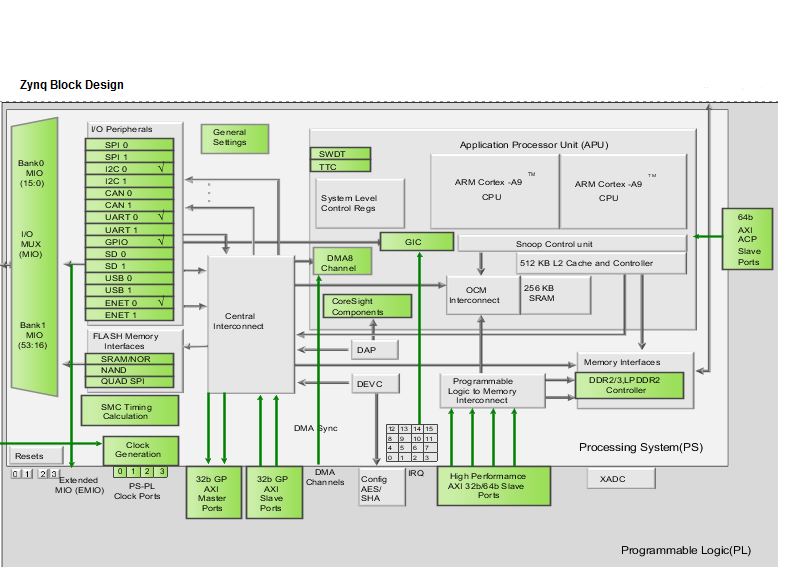
## Blok dizajn



Slika 3.1 Blok dizajn implementiranog sustava

Na slici 3.1 prikazan je blok dizajn implementiranog sustava koji se sastoji od nekoliko blokova. Najbitniji je processing\_system7 koji predstavlja dvojezgreni ARM procesor (PS) na kojemu se izvodi aplikacijski kod. Na njegovu AXI master sabirnicu preko interconnecta spojena su dva AXI GPIO slave bloka: na jedan su dovedeni signali s kamere (vsync, href, pclk i d[0-7], svi su ulazni), a na drugi 4 gumba (ulazni) te 4 LED diode (izlazni) koje služe za pronalaženje grešaka (engl. *debugging*)u računalnom kodu. Blok rst\_ps7 se brine o postavljanju reset signala prilikom dovoda napajanja ili resetiranja sustava.

## Zynq konfiguracija



Slika 3.2 Zynq konfiguracija

Na slici 3.2 prikazane su postavke PS-a. Od periferije su uključeni I2C kontroler koji se koristi za konfiguriranje internih registara kamere preko SCCB sučelja, Ethernet kontroler za slanje slike na zahtjev do računala klijenta te UART kontroler koji se koristio za pronalaženje grešaka prilikom izrade sustava.

Postavljeno je da procesorske jezgre rade na taktu od 650 MHz, dok se na programibilnu logiku šalje takt od 125 MHz, što je po dokumentaciji najviši takt na kojemu AXI GPIO blok može raditi na uređajima brzinskog razreda -1 kojemu pripada i PYNQ-Z1 pločica.

# Korišteni algoritmi

Izrada JPEG formata slike se sastoji od dva osnovna koraka:

* Diskretna kosinusna transformacija – DCT
* Huffmanovo kodiranje

Nakon završetka kodiranja potrebno je dodati markere i zaglavlja koje zahtjeva JPEG JFIF standard.

Nakon što se podaci dohvate iz kamere i spreme u memoriju PYNQ-a nad njima se pokreće DCT koji radi nad blokovima od 64 podatka, a nad svakim blokom se izvode sljedeće funkcije:

Funkcija *read\_ycbcr\_block* čita jedan blok veličine 8x8 pixela te ga sprema u varijablu *ycbcr\_block.*

*read\_ycbcr\_block(raw\_ycbcr\_image, x, y, &ycbcr\_block);*

Funkcija *shift\_ycbcr\_block* vrijednostima komponenata u bloku oduzima vrijednost 128 da bi se vrijednosti pomakle iz raspona [0, 255] u raspon [-128, 127]. Ova operacija je potrebna zbog toga što je DCT definiran nad simetričnim signalom.

*shift\_ycbcr\_block(&ycbcr\_block);*

Sljedeća funkcija je *block\_dct* koja radi DCT nad blokom. Funkcija zbog brzine koristi lookup tablicu u kojoj se nalaze vrijednosti kosinus funkcija korištene kod dvodimenzionalne kosinusne transformacije. Zbog povećanja brzine funkcija ne računa svih 64 koeficijenata, već računa samo prvih 16 koeficijenata koji se nalaze u gornjem lijevom kutu 8x8 bloka.

*block\_dct(&ycbcr\_block, &result\_block);*

Nakon što se izračunaju DCT koeficijenti potrebno ih je kvantizirati. Kvantiziranje se obavlja u funkciji *quantize\_ycbcr\_block* koja koristi dvije kvantizacijske tablice da bi se smanjila količina podatka.

*quantize\_ycbcr\_block(&result\_block);*

Nakon što se blok kvantizira potrebno ga je spremiti u memoriju funkcijom *write\_ycbcr\_block.*

*write\_ycbcr\_block(raw\_ycbcr\_image, x, y, &result\_block);*

Nakon što se nad svim blokovima i komponentama obavi DCT potrebno je te iste podatke kodirati Huffmanovim kodiranjem. Kodiranje počinje tako što se podatci komponente ispišu u privremeni spremnik te komponente u zig-zag redoslijedu. Nakon što se podaci nalaze u privremenim spremnicima potrebno je podatke svakog bloka kodirati Huffmanovim kodovima i spremati te kodove u zajednički spremnik u koji se spremaju podaci pojedinih komponenti.

Blok se kodira Huffmanovim kodom sa funkcijom *code\_block* koja se poziva naizmjenice nad privremenim spremnicima pojedinih komponenata jer JPEG JIFF zahtjeva da ako se u jednom scanu nalazi više komponenti da kodirani blokovi tih komponenti budu isprepleteni.

Kako Huffmanovi kodovi nisu istih duljina, potrebno je popuniti bajtove tog spremnika. Za to se koristi funkcija *put\_bit\_into\_bit\_buffer*.

*void put\_bit\_into\_byte\_buffer(BYTE\_BUFFER\* byte\_buffer, unsigned char bit\_value)*

Nakon što se svi podaci spreme u zajednički spremnik potrebno je nadopuniti zadnji bajt u spremniku sa vrijednostima ‘1’, za to se koristi funkcija round\_byte\_buffer.

*void round\_byte\_buffer(BYTE\_BUFFER\* byte\_buffer)*

Nakon što su svi podaci kodirani huffmanovim kodovima potrebno je generirati markere i zaglavlja koja su potrebna za JPEG JIFF. Za to se koristi funkcija *form\_jpeg* koja prima pokazivač na novi spremnik i u njega sprema sve potrebne markere i zaglavlja, te na kraju dodaje kodirane podatke iz zajedničkog spremnika.

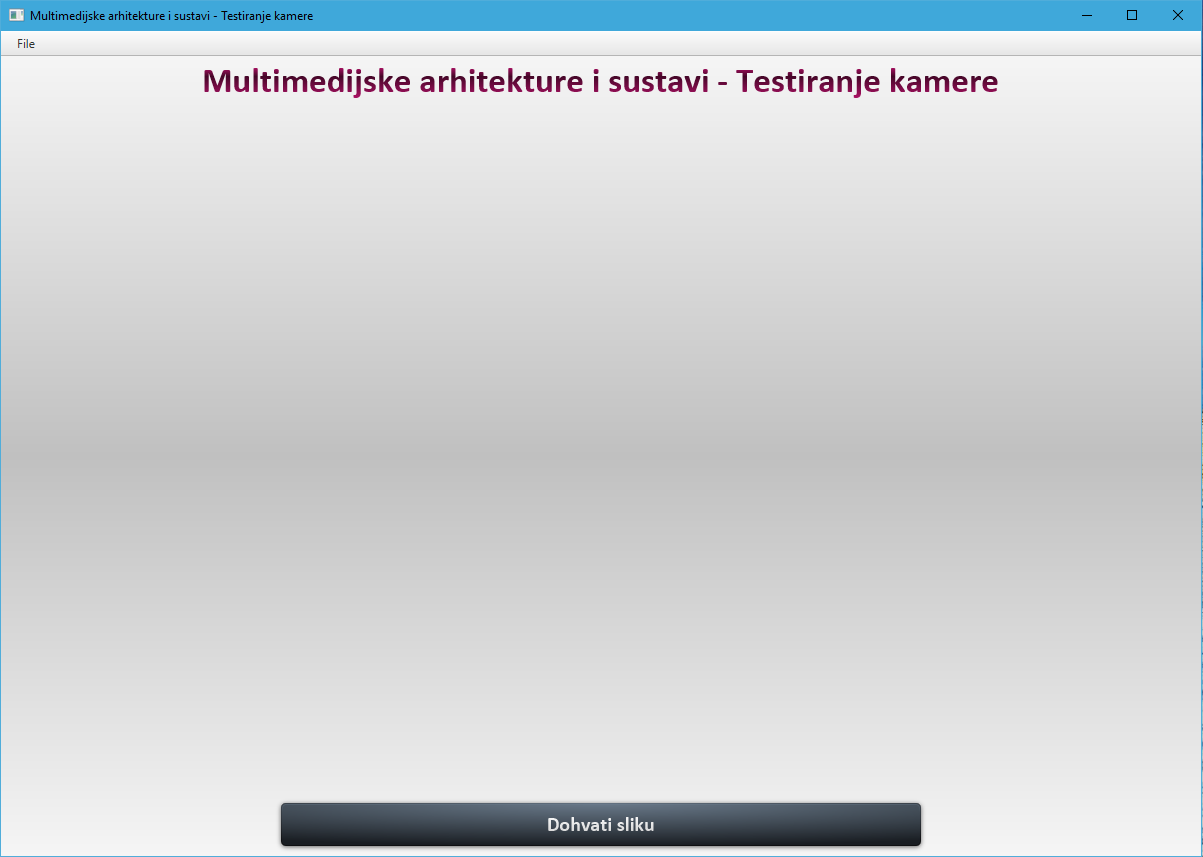
*void form\_jpeg(OUTPUT\_STREAM\* output\_stream, BYTE\_BUFFER\* data)*

Nakon formiranja JPEG-a, prilikom slanja računalu se svaki bit XOR-a sa tajnim podatkom koji je poznat samo korisniku i PYNQ pločici, te se tako postiže sigurniji prijenos putem Etherneta.

# Upute za pokretanje aplikacije

Prije samog pokretanja aplikacije potrebno je priključiti PYNQ pločicu ethernet kabelom na računalo, te na računalu podesiti IP adresu koja odgovara podmreži koju PYNQ ispisuje prilikom pokretanja na svojoj serijskoj vezi. Tijekom korištenja i testiranja aplikacije postavljeno je da je IP adresa PYNQ pločice 192.168.1.10, pa tako treba podesiti IP adresu računala na npr. statičku 192.168.1.2,192.168.1.3…itd, ovisno o odabranoj maski podmreže.

Budući da je aplikacija izvedena u programskom jeziku java, za njeno izvođenje treba imati instaliranu verziju Jave >1.7.0, jer se koristi File.IO paket koji je dostupan od jave 1.7.0. Aplikacija se pokreće pokretanjem .jar datoteke, te se pritiskom na gumb pokreće slanje zahtjeva za dohvat slike s PYNQ pločice. Pretpostavljena IP adresa PYNQ-a je 192.168.1.10, te port 7. Pretpostavljeni format koji aplikacija prihvaća je .jpg. Jednom dohvaćena slika se automatski sprema u direktorij iz kojega je pokrenuta .jar aplikacija. Također nad dohvaćenom slikom se unutar aplikacije mogu izvoditi jednostavne manipulacije, kao npr. rotacija, filtriranje po crvenoj, plavoj ili zelenoj boji, zamagljivanje, ili pak prebacivanje u C/B sliku. Također, slika dobivena manipulacijama se može pohraniti pritiskom na tipku pohrani sliku.



Slika . Izgled sučelja aplikacije

# Literatura

[1] Slajdovi s predavanja – Multimedijske arhitekture i sustavi, FER 2017.

[2] OV7670 spreadsheet – [dokumentacija kamere](https://www.voti.nl/docs/OV7670.pdf)

[3] JavaFX getting started - [JavaFX getting started](https://docs.oracle.com/javase/8/javase-clienttechnologies.htm)

[4] Vivado dokumentacija – [Vivado](https://www.xilinx.com/products/design-tools/vivado.html#documentation)

[5] JPEG JFIF standard specification - [w3 specification](https://www.w3.org/Graphics/JPEG/)