RT-RK

TEME BSC/MSC RADOVA

Za studente Elektrotehničkog
i Matematičkog fakulteta
u Beogradu

Dragi studenti,

U saradnji sa Elektrotehničkim i Matematičkim fakultetom u Beogradu nudimo mogućnost stipendirane izrade MSc ili BSc rada u trajanju od 3 meseca. U nastavku ove brošure nalaze se spisak tema koje biste mogli da radite u okviru Instituta RT-RK.

Pre upoznavanja sa potencijalnim temama, želeli bismo da vam predstavimo prednosti našeg stipendijskog programa, kao i iskustva kolega koji su ga uspešno završili, a trenutno su zaposleni kod nas.

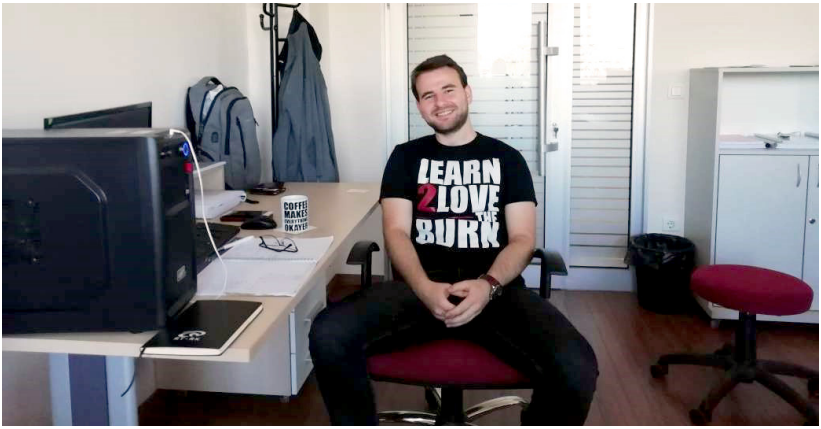
Benefiti RT-RK stipendijskog programa:

- Izrada diplomskog rada uz podršku inženjera mentora
- Mogućnost korišćenja savremene opreme i alata
- Mogućnost profesionalnog i akademskog razvoja
- Moderno opremljeno radno okruženje sa prostorijama za odmor i zabavu
- Organizovana druženja i sportske aktivnosti
- Fleksibilno radno vreme i neobavezan stil oblačenja

Sve zainteresovane pozivamo da pošalju svoje prijave ili potencijalne nedoumice na adresu: jobs@rt-rk.com

Institut RT-RK

Program stipendiranja iz ugla studenata i mentora



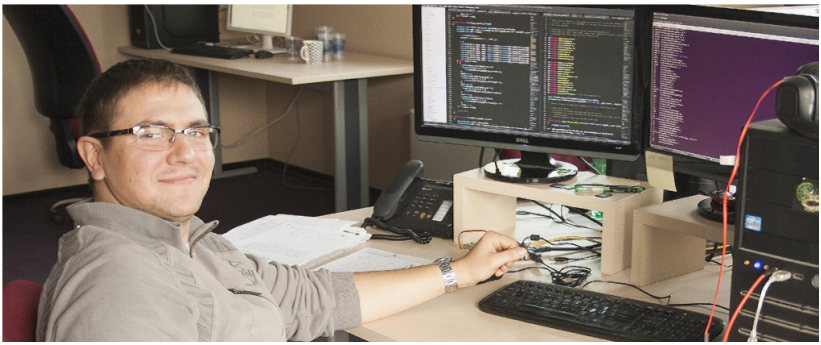
Nikola Milutinović

Kao neko ko je završavao četvrtu godinu Elektrotehničkog fakulteta, na prvom mestu mi je bilo da napišem kvalitetan diplomski rad i uz to naučim nešto novo. Ne postoji puno firmi koje nude tako nešto, pri čemu se ispostavilo da RT-RK nudi baš ono što mi je potrebno. Program stipendiranja je takav da ovde radimo svoj diplomski rad sa temama koje su zavisile od tima u kojem bismo radili. Uz to smo još i dobijali stipendiju što je dodatna motivacija da naučite više i odradite vaš diplomski rad. Od kada sam došao, radio sam 8 umesto predviđenih 4 sata, jer mi se veoma svidela atmosfera na poslu i želeo sam da naučim više. Kolege su bile prijatne i mentor mi je izlazio u susret kad god mi je trebala pomoć. Naravno, isto tako se očekivalo od mene da sam savladam puno toga, što mi je pomoglo da se osamostalim i poradim na samopouzdanju. Tema je bila ozbiljna i iziskivala je od mene spremnost da se susretnem sa stvarnim problemima koje jedan ozbiljan softver nosi, da budem spreman da naučim puno toga novog, kao i da primenim naučeno na fakultetu. Posle puno truda, uspona i padova, težih nedelja, uspeo sam da rešim dosta problema i puno toga naučim. Nakon diplomiranja na fakultetu, profesori su bili prezadovoljni i izrazili su želju za još ovakvih ozbiljnih tema.

Diplomski rad iz ugla studenta i mentora

Jovana Simić

Ćao, ja sam Jovana, student master studija Softverskog inženjerstva na Elektrotehničkom fakultetu u Beogradu. Tokom programa stipendiranja radila sam na razvoju aplikacije koja služi za upravljanje STB uređajem putem glasa, što je i bilo tema mog diplomskog rada. Prilikom izrade rada u svakom trenutku sam mogla da računam na pomoć, kako od mentora, koji je svojim savetima i stručnošću doprineo da lakše savladam probleme sa kojima sam se susrela tokom izrade aplikacije, tako i kompletnog tima koji je uvek na raspolaganju za pomoć i saradnju. Takođe bih istakla i važnost kurseva koje sam pohađala tokom programa stipendiranja, koji su mi pružili znanja o novim tehnologijama koje se koriste u svakodnevnom radu na profesionalnim projektima. Nakon diplomiranja sam se zaposlila i nastavila svoje usavršavanje u oblasti digitalne televizije.



Miodrag Dinić, Senior inženjer - Mentor

Smatram da je stipendijski program veoma dobro osmišljen i da je pre svega odlična prilika za mlade ljude željne praktičnog znanja i usavršavanja. Naši timovi su veoma plodno tlo za definisanje interesantnih i produktivnih tema za BSc i MSc radove studenata, gde se tokom izrade istih stiču stručna znanja iz oblasti operativnih sistema, programskih prevodilaca i sistemskog programiranja uopšte. Kao mentor tokom izrade BSc i MSc radova, mogu se pohvaliti samo pozitivnim iskustvima u radu sa našim stipendistima. Kroz mentorski rad trudim se da razvijam inženjerski pristup rešavanju problema kod mladih kolega kao i kritičku misao koja je neizostavan deo tog procesa. Radimo zajedno na usavršavanju poslovne komunikacije i učimo kako funkcioniše proces razvoja softvera na velikim projektima i u velikim timovima kao što je naš. Kao rezultat ove saradnje stipendisti brane svoj BSc ili MSc rad, a mi dobijamo kvalitetne potencijalne kolege koje će biti spremne da postanu deo našeg tima kao uspešni inženjeri.

1. AUTONOMNA VOŽNJA

Autonomna vozila postaju stvarnost i postaju u sve većoj meri dostupna sa sve više funkcionalnosti (pomoć pri parkiranju, preticanju, vožnja u gužvi, konvoj na autoputu, itd.). Svake godine postepeno prelazi se na sledeći tehnološki nivo, sve do postizanja potpune autonomije (nivo 5). Da bi se omogućila bezbedna vožnja bez apsolutne intervencije vozača (nivo 5) potrebno je na inteligentan način obrađivati i ukrštati ulaze sa kamera visoke rezolucije, LIDAR-a (eng. LIgth raDAR), infracrvenih senzora, i ostalih inteligentnih senzora. Postojeći i novi algoritmi bazirani na mašinskom učenju, konvolutivnim neuralnim mrežama (eng. Convolutional Neural Networks (CNN)) i računarskoj viziji (eng. Computer Vision (CV)) treba da pomognu da se postigne cilj potpune autonomne vožnje a zadatak diplomskog je da se pokaže kako se mogu koristiti u realnom vremenu da bi se omogućila primena u savremenim vozilima.

Computer Vision (CV)

Algoritmi

1. Semantic Segmentation
2. Optical Flow
3. Stereo Block Matching
4. Classifier: Dollar Toolbox
5. Camera Mirror Replacement
6. Object Detection:
    a. Pedestrian
    b. Road Sign
    c. Vehicle
    d. Truck
    e. Cyclist
7. Pose Estimation
8. Panorama Stitching
9. Surround View
10. ISP preprocessing

Koraci u izradi zadatka

- Upoznati se sa ADAS (eng. Advanced Driver Assistance Systems) tehnologijom

- Upoznati se sa odabranim algoritmom baziranom na računarskoj viziji
- Upoznati se sa namenskom ADAS platformom (arhitektura, jezgra, memorija, okruženje)
- Upoznati se sa procesom prilagođenja algoritama za namensku platformu za rad u realnom vremenu
- Upoznati se sa načinom paralelizacije algoritama i efikasnog korišćenja memorije
- Implementirati optimizovanu verziju algoritma tako da se postigne ubrzanje i zadrži apsolutno poklapanje rezultata (eng. bit-exact)
- Rešenje testirati pomoću jediničnih testova

Rešenje realizovati upotrebom programskog jezika C++/C/[ASM] za namensku ADAS platformu.

Convolutional Neural Network (CNN)

Algoritmi

1. Classification:
    a. AlexNet
    b. VGG16
2. Object Detection:
    a. Yolo-Tiny
    b. R-CNN
3. Generative adversarial network (GAN)

Custom Layers

- Razvoj i testiranje proizvoljnih slojeva za neuralne mreže

Benchmark/Testing/Automatization

- Automatizacija procesa prevođenja istreniranog CNN modela u mašinski kod za ciljanu namensku platformu uz mogućnost dinamičkog prilagođenja parametara
- Razvoj koda za automatsko treniranje mreža i merenje performansi u Caffe okruženju
- Razvoj koda za automatsko pre-procesiranje različitih otvorenih skupova

podataka i priprema za integraciju sa Caffe okruženje
- Razvoj koda za automatsko izvršenje aplikacija na namenskoj ploči (idealno obezbediti podršku za različite generacije uređaja), skupljanje rezultata, kao i prikaz rezultata merenja (npr. statistike i metrike relevantne za obuku, slika na kojima se najviše gresi, itd.) Dodatno obezbediti prilagođenje izvorne aplikacija da obezbede sličnu podršku

Koraci u izradi zadatka
- Upoznati se sa ADAS (eng. Advanced Driver Assistance Systems) tehnologijom
- Upoznati se sa odabranim algoritmom baziranom na neuralnim mrežama
- Upoznati se sa namenskom ADAS platformom (arhitektura, jezgra, memorija, okruženje)
- Upoznati se sa procesom prilagođenja algoritama za namensku platformu za rad u realnom vremenu
- Upoznati se sa procesom obučavanja neuralnih mreža (odabir podataka za obuku, automatizovanje okruženja za treniranje, treniranje modela, prilagođenja modela, provera tačnosti)
- Rešenje testirati pomoću jediničnih testova

Rešenje realizovati upotrebom programskog jezika C++/Python za PC i namensku ADAS platformu kao i Caffe CNN okruženja.

CARLA - open-source simulator for autonomous driving research

Traffic scenarios simulation
- Postavka CARLA simulatora
- Razvoj i postavka scenarija vožnje i okolnog saobraćaja
- Razvoj i postavka simuliranog okruženja (vremenske prilike, topologija puteva, saobraćajni znakovi)
- Razvoj i postavka heterogenih i komplementarnih senzora za autonomnu vožnju u simuliranom okruženju (Lidar, kamere, senzori rastojanja, GPS, itd.)

- Prikupljanje i vizualizacija podataka sa senzora

Koraci u izradi zadatka

- Upoznati se sa ADAS (eng. Advanced Driver Assistance Systems) tehnologijom
- Upoznati se sa namenskom ADAS platformom (arhitektura, jezgra, memorija, okruženje)
- Upoznati se sa CARLA simulatorom i mogućnostima za podešavanje i proširenje
- Kreirati početnu postavku za simulaciju i postepeno proširivati sa neophodnim elementima
- Omogućiti spregu izmedju simulatora i namenske platforme za razmenu podataka

Rešenje realizovati upotrebom programskog jezika Python/C/C++ za PC i namensku ADAS platformu.

Programski prevodioci

LLVM predstavlja potporu za razvoj raznih sistemskih alata, izmedju ostalog i programskog prevodioca.
Takođe, LLVM spada u jedan od najatraktivnijih projekata današnjice na kome rade najveće kompanije našeg doba.

Proširenje LLVM infrastrukture prevodioca za praćenje stek argumenata funkcije

Korisničke verzije softvera se prevode sa optimizacijama nivoa -02 ili više. U takvim programima može doći do neočekivanog prekida i u tom slučaju nastaju datoteke jezgra (eng. core fil) koje čuvaju sliku memorije programa u trenutku prekida. Datoteke jezgra služe za analizu okolnosti neočekivanog prekid. Prva stvar na koju nailazimo tokom te analize jeste trag pozivanja funkcija (eng. call-trace) koje su prethodile prekidu. U takvom tragu većini parametara funkcije ne znamo vrednost jer se one smatraju optimizovanim u tom trenutku. Neki parametri su stvarno optimizovani, ali vrednosti nekih parametara funkcije se

mogu potražiti u roditeljskom funkcijskom okviru (eng. function frame). Analizom asemblerskog koda u roditeljskom funkcijskom okviru na mestu pozivanja posmatrane funkcije iskusni programeri mogu dedukovati koju vrednost je parametra imao na početku funkcije. Automatizacija takve tehnike opisana je u DWARF 5 standardu i već je implementirana u programskom prevodiocu GCC i alatu GDB još 2011, ali delimicno i u kompajlerskoj infrastrukturi LLVM za arhitekturu X86 od 2019. Ono sto nedostaje u oba ova kompajlera jeste pracenje ulaznih vrednosti argumenata funkcije koji se prenose preko steka. Cilj ovog rada je unapređenje kompajlerske infrastrukture LLVM dodavanjem podrške za praćenje ulaznih vrednosti takvih parametara, sa naglaskom na arhitekturu X86.

U okviru zadatka potrebno je:

- Upoznati se sa LLVM ekosistemom
- Upoznati se sa konvencijom pozivanja funkcije (eng. calling convention) za arhitekturu X86
- Prikupljanje informacija o nacinu prenosenja argumenata preko steka prilikom poziva funkcije za arhitekturu X86
- Upoznati se sa delom instrukcijskog skupa arhitekture X86
- Unapredjenje algoritma za prepoznavanje vrednosti prosledjenih kao argument funkcije dodavanjem podrske za argumente koji se prosledjuju preko steka
- Testiranje resenja koriscenjem alata kao sto su llvm-dwarfdump, llvm-locstats

Rešenje implementirati korišćenjem programskog jezika C++ prateći savremene standarde.
Student će se realizacijom ovog zadatka upoznati dublje sa principima rada programskog prevodioca i proći će kroz proces rada i doprinosa na projektu otvorenog koda.

Ispravnost softvera u automobilskoj industriji

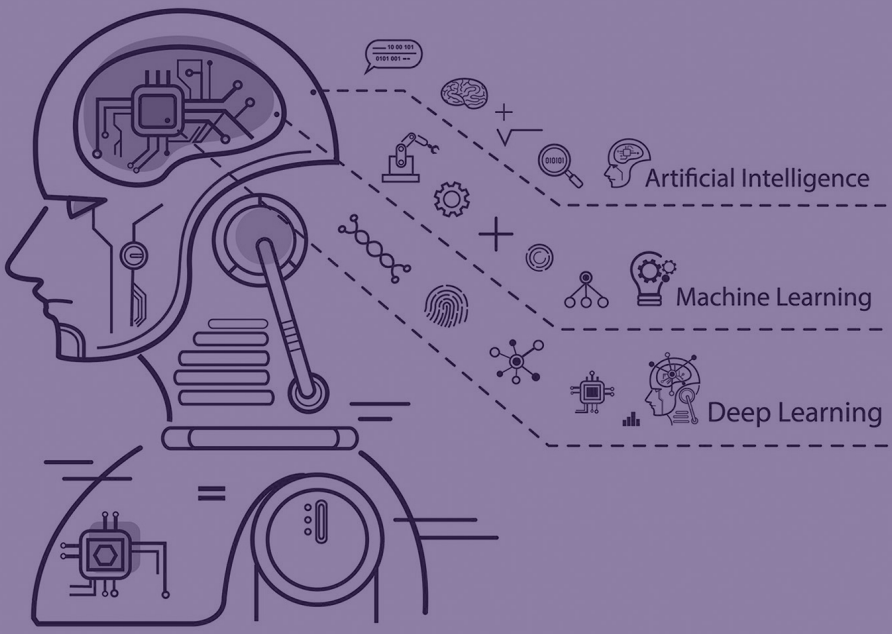
Greške u softveru automobila mogu imati fatalne posledice po vozača i njegovu okolinu. Zbog toga je ispravnost softvera koji se razvija za automobilsku industriju od izuzetne važnosti i veoma je aktuelan razvoj preporuka, standarda i automatizovanih alata koji imaju za cilj da olakšaju i ubrzaju izgradnju ispravnog softvera. Najznačajnije preporuke koje se prate u razvoju softvera za automobilsku industriju su definisane od strane organizacija MISRA i AUTOSAR. One obezbeđuju visok kvalitet softvera i smanjuju verovatnoću za pojavljivanje grešaka u kodu.

U okviru zadatka potrebno je:

- Upoznati se sa zahtevima koji su propisani MISRA/AUTOSAR preporukama za korišćenje jezika C/C++ u sistemima sa kritičnom bezbednošću
- U okviru LLVM infrastrukture, implementirati dodatak za prevodilac Clang koji omogućava automatsku proveru ispunjavanja odabranih preporuka
- Rešenje validirati sopstvenim testovima kao i testovima koji prate primere date u MISRA/AUTOSAR preporukama
- Rešenje validirati u okviru projekta Automotive Grade Linux

Rešenje realizovati u programskom jeziku C++, u okviru Linux operativnog sistema. Student će se realizacijom ovog zadatka upoznati dublje sa principima rada programskog prevodioca i tehnikama statičke analize softvera. Proći će kroz proces rada i doprinosa na projektu otvorenog koda.

2. MAŠINSKO UČENJE



Proširenje TensorFlow biblioteke za treniranje veštačke inteligencije

TensorFlow je biblioteka otvorenog koda za numeričko izračunavanje. Koristi se kao okvir za mašinsko učenje koji korisniku značajno olakšava proces prikupljanja podataka, treniranje modela i predviđanje budućih rezultata. TensorFlow obezbeđuje Python front-end API zgodan za korišćenje u aplikacijama za mašinsko učenje, dok se sama numerička izračunavanja obavljaju u C++ kodu visokih performansi oprimizovanom za procesore i Nvidia grafičke karte. Permisivno je licenciran, razvijen i održavan od strane kompanije Google.

U okviru zadatka potrebno je:

- Upoznati se sa strukturom TensorFlow biblioteke
- Upoznati se sa reprezentacijom brojeva u pokretnom zarezu korišćenjem aritmetike brojeva u nepokretnom zarezu (fixed point arithmetic)
- Upoznati se sa CUDA programiranjem
- Implementirati neke od složenih numeričkih operacija koje će koristiti dinamičku artimetiku brojeva u nepokretnom zarezu i čije jezgro će se izvršavati u paraleli na Nvidia grafičkoj karti
- Rešenje validirati sopstvenim testovima

Rešenje realizovati u programskom jeziku C++ i Python, u okviru Linux operativnog sistema. Student će se realizacijom ovog zadatka dublje upoznati sa funkcionisanjem TensorFlow numeričkih operatora koji predstavljaju jezgro ove biblioteke, kao i principima optimizacije koda pisanog za CUDA. Proći će kroz proces rada i doprinosa na projektu otvorenog koda.

Primena višeslojnih impulsnih neuronskih mreža u analizi i odlučivanju kod vremenskih signala

Impulsne neuronske mreže (Spiking Neural Network - SNN) predstavljaju treća generacija neuronskih mreža. Karakteristično za njih je što su sastavljene od veštačkih neurona koji sadrže vremensku komponentu u načinu funkcionisanja. Za razliku od modela prethodne dve generacije neuronskih mreža (jednosmeri perceptroni i dvosmerne višeslojne neuronske mreže), model impulsnog neu-

rona je bliži biološkom neuronu i sam mehanizam obrade signala je bliži obradi koja se odvija u biološkim neuronskim sistemima.

Za treću generaciju neuronskih mreža predvidja se masovna eksploatacija zbog mogućnosti njihove jednostavne implementacije na hardverskom nivou. Veštački impulsni neuroni se uspešno implementiraju uz pomoć postojeće tehonologije u takozvanim neuromorfnim čipovima. Neuromorfni čipovi su danas dosupni u domenu istraživanja.

Glavni problem kod impulsnih neuronskih mreža je njihovo obučavanje. S obzirom na način rada, kod mreže impulsnih neurona se ne mogu primeniti poznate metode mašinskog učenja (na osnovu gradijenta greške, propagacijom unazad i slične metode zasnovane na mehanizmima učenja za obradu vremenski nezavisnih signala).

U okviru zadatka potrebno je:

- Upoznati se sa strukturom i načinom funkcionisanja impulsnih neuralnih mreža.
- Upoznati se sa načinima obučavanja impulsnih neuralnih mreža (Hebbian learning, Spike-timing-dependent plasticity - STDP, Reservoir computing)
- Modelovati veštački neuron uz pomoć električnih elemenata (R, L, C i nelinarnih elemenata).
- Povezati neurone u mrežu i uspostaviti funkcije za odredjivanje greške u odlučivanju.
- Osmisliti način za obučavanje mreže.
- Obučiti mrežu na skupu dostupnih podataka. U toj svrsi se može koristiti, po dogovoru, neki skup vremenskih podataka (npr. zvučni zapis govora ili muzike) sa spoljašnjeg izvora javno dostupnih baza za mašinsko učenje.
- Testirati kvalitet odlučivanja neuronske mreže za zadati cilj (npr. prepoznavanje reči, prepoznavanje govornika, klasifikacija signala i.t.d.).

Rešenje realizovati u programskom jeziku C++ i Python, uz mogućnost upotrebe CUDA ili OpenCL. Prilikom realizacije ovog zadatka, student će se upoznati sa načinom funkcionisanja neuronskih mreža do nivoa modelovanja samih neurona. Zadatak je istraživačkog karaktera, jer za opisan problem ne postoji

ustanovljeno opšte rešenje koje treba implementirati. Tokom rešavanja izabranog praktičnog i aktuelnog problema, student će proći kroz proces istraživanja i ovladaće elementarnim principima na kojima se zasniva rad neuronskih mreža. Poželjno je predznanje iz obrade signala i modelovanja električnih kola.

Optimizacija TensorFlow Lite numeričkih operacija korišćenjem OpenMP biblioteke

TensorFlow Lite je okvir za mašinsko učenje otvorenog koda. Koristi se za izvršavanje pretreniranih TensorFlow modela na mobilnim i ugrađenim uređajima ograničenih hardverskih resursa. Podržan je za Android i iOS operativne sisteme preko C++ API-ja, a takođe nudi i Java omotač za Android. Permisivno je licenciran, razvijen i održavan od strane kompanije Google.

OpenMP je biblioteka koja nudi API za optimizaciju programa korišćenjem paralelnog izvršavanja koda u više niti. Pojedine numeričke operacije u okviru TensorFlow Lite alata koriste mogućnosti OpenMP biblioteke zarad postizanja boljih performansi izvršavanja.

U okviru zadatka potrebno je:

- Upoznati se sa strukturom TensorFlow Lite alata
- Upoznati se sa OpenMP bibliotekom i mogućnostima paralilizacije koda korišćenjem iste
- Dodati podršku za optimizaciju nekih od TensorFlow Lite operacija koristeći OpenMP API
- Rešenje validirati sopstvenim testovima
- Izmeriti uticaj implementirane izmene na performanse i veličinu binarnog paketa

Rešenje realizovati u programskom jeziku C++ a kao razvojno okruženje koristiti Linux operativni sistem. Student će se realizacijom ovog zadatka dublje upoznati sa funkcionisanjem TensorFlow Lite numeričkih operatora, kao i principima optimizacije koda korišćenjem OpenMP alata. Proći ce kroz proces rada i doprinosa na projektu otvorenog koda.

3.
DIGITALNA TELEVIZIJA

Napredna provera autentičnosti Android STB uređaja upotrebom blockchain tehnologije

Jedan od gorućih problema u savremenom svetu digitalne televizije jeste validacija priključenih STB uređaja. Blockchain je rastuća lista zapisa koji su povezani upotrebom kriptografskih metoda. Provera da li je priključeni uređaj validan predstavlja veliki izazov u savremenim TV mrežama. Upotrebom blockchain tehnologije i pametnih ugovora istražiti mogućnost njihove upotrebe u svetu digitalne televizije a zatim realizovati sistem proveru autentičnosti Android STB uređaja.

U okviru zadatka potrebno je:

- Upoznati se sa blockchain tehnologijom
- Upoznati se sa tehnologijom pisanja pametnih ugovora
- Upoznati se sa procesom pravljenja privatne blockchain mreže
- Implementirati sistem za validaciju STB uređaja upotrebom blockchain tehnologije
- Rešenje testirati pomoću JUnit testova

Rešenje realizovati upotrebom programskog jezika Java, Android STB uređaj, Android SDK i open source java blockchain biblioteka.

Siguran prenos digitalnih TV podataka

Blockchain je rastuća lista zapisa koji su povezani upotrebom kriptografskih metoda. Za siguran prenos TV podataka u svetu postoje brojna komercijalna i open source rešenja i implementacije, ali ni jedna nije zasnovana na novoj kriptografskoj tehnologiji. Upotrebom blockchain tehnologije i pametnih ugovora najpre istražiti mogućnost njihove upotrebe u svetu digitalne televizije, a zatim realizovati sistem za siguran prenos TV sadržaj.

U okviru zadatka potrebno je:

- Upoznati se sa blockchain tehnologijom
- Upoznati se sa tehnologijom pisanja pametnih ugovora

- Upoznati se sa procesom pravljenja privatne blockchain mreže
- Upoznati se sa Node.js sistemom
- Upoznati se sa osnovama kriptografije i TV tokom podataka
- Implementirati sistem za siguran prenos digitalnih TV podatak upotrebom blockchain tehnologije. Realizovati sistem koji će: praviti ključeve za zaštitu multimedijalnog toka audio-video podataka, prenos ključeva putem blockchain tehnologije i reprodukciju na STB uređaju
- Rešenje testirati pomoću JUnit testova

Rešenje realizovati upotrebom programskih jezika Java i Java Script, Android STB uređaj, Android SDK, Node.js i open source java blockchain biblioteka.

Umreženi Android uređaji

Chromecast - slanje audio-video podataka od skora više nije ograničenje samo za popularne Android aplikacije. Predstavljanjem Cast aplikacione programske podrške omogućeno je pisanje privatnih cast aplikacija koje međusobno mogu da dele audio i video podatke.

U okviru zadatka potrebno je:

- Upoznati se sa Android SDK i napisati jednostavnu aplikaciju za Android mobilni i STB uređaj
- Upoznati se sa Cast aplikacionom programsko podrškom
- Napisati Android mobilnu aplikaciju koja će pomoću Cast aplikacione programske podrške povezati na Android STB aplikaciju i na taj način omogućiti reprodukciju audio i video podataka
- Proširite Android aplikacije i dodati mogućnost za pauziranje i premotavanje audio i video podataka
- Rešenje testirati pomoću JUnit testova

Rešenje realizovati upotrebom programskog jezika Java i Java Script, Android STB uređaja, Android mobilnog uređaja, Android SDK i CAF SDK.

Vulkan na televiziji

Sa rastom performansi Android uređaja i pojavom Vulkan programske podrške otvara se mogućnost za obradu i detekciju kvaliteta digitalnog TV signala na STB uređaju.

U okviru zadatka potrebno je:

- Upoznati sa se Android SDK i napisati jednostavnu Android korisničku aplikaciju
- Upoznati sa se Android NDK i napisati jednostavnu Android native aplikaciju
- Upoznati se sa Vulkan programskom podrškom
- Napisati Android Vulkan aplikaciju koja će izvršavati algoritam za detekciju kvaliteta video podataka
- Rešenje testirati pomoću JUnit testova

Rešenje realizovati upotrebom programskog jezika Java, C++, Android STB uređaja, Android SDK, NDK i Android Vulkan programske podrške.

Televizija na sunce

Solarna energija je sve jeftinija i pristupačnija. STB uređaji su postali podrazumevani uređaji u svakom domaćinstvu. Zašto ne bi spojili ekonimično i zanimljivo.

U okviru zadatka potrebno je:

- Povezati STB uređaj na solarni panel tako da STB uređaj dobija napajanje i informaciju koliko struje proizvodi solarni panel
- Razviti Android TV aplikacija koja će informacije o proizvodnji potrošnji energije čuvati na cloud bazi podataka, na primer firebase
- Razviti Android mobilnu aplikaciju koja će informacija sa cloud baze prikazati korisniku na mobilnom uređaju
- Rešenje testirati i verifikovati Junit testovima

Rešenje realizovati upotrebom programskog jezika Java i C, Android STB uređaj, Android SDK, Firebase SDK i potrebnih dodatnih open source biblioteka.

Alexa on TV

Far-field uređaji se nalaze na tržištu već neko vreme. Omogućavaju korisnicima da putem glasovnih komandi izvršavaju akcije na pametnim uređajima. Već je uveliko moćuće puštati muzuku, uključivati svetlo, kontrolisati grejanje itd. Hajde zajedno da razvijemo aplikacije za kontrolu TV uređaja.

U okviru zadatka potrebno je:

- Razviti aplikaciju na Alexa uređaju koja će prepoznavati komande korisnika kao što su pusti kanal, pusti film i sl
- Istražiti i razviti cloud protokol koji je povezati Alexa aplikaciju sa Android TV STB uređajem
- Napisati Android TV aplikaciju koja će prihvatati komande iz cloud-a i izvršavati na STB uređaju
- Rešenje testirati preko unit testova

Rešenje realizovati upotrebom programskog jezika Java, Alexa SDK, Android STB uređaj, Android SDK, Firebase SDK, Google cloud engine, AWS i potrebnih dodatnih open source biblioteka.

Offline entertainment

OTT uređaji imaju malo ili nimalo funkcionalnosti bez mreže. Vozila se često nalaze u okruženjima u kojima je mreža nedostupna a očekujemo rast zastupljenosti OTT uređaji u automotive industriji. Napraviti rešenje koje podržava prezentaciju sadržaja u uslovima bez mreže.

U okviru zadatka potrebno je:

- Napraviti Android aplikaciju koja pušta multimedijalni sadržaj
- Prepoznaje odsustvo mreže
- Pušta uskladišten sadržaj kada je mreža nedostupna i pamti statistike ovakvog ponašanja
- Kad dobije pristup mreži na cloud bazu (Firebase npr) šalje informacije o vremenu provedenom bez mreže i puštenom sadrzaju

Rešenje testirati JUnit testovima, uključujući i potrebno vreme da aplikacija odreaguje na isključivanje mreže.

HEVC bitstream Analizator

High Efficiency Video Coding (HEVC) je najnoviji standard za video kompresiju.

Podržava 8K rezoluciju i u poređenju sa AVC kodekom ima bolje performanse, pružajući zančajan kvalitet slike čak i pri niskom bitrate-u.

Pri implementaciji bilo kog streaming-a na savremenim Set Top Box (STB) uređajima javlja se potreba za analizom signalnih informacija a vrlo često i za analizom meta podataka vezanih za konkretne audio i/ili video frame-ove.

S tim u vezi, doprinos koji bi diplomski rad na ovu temu doneo, u praktičnom delu, treba da obuhvata:

- Kreiranje desktop aplikacije HEVC Analyzer (Java ili C++)
- Omogućiti importovanje HEVC bitstream-a
- Parsirati HEVC bitstream
- Na grafičkom interfejsu smisleno organizovati prikaz meta podataka frame-ova
- Obezbediti navigaciju po frame podacima

4.
ALATI U MREŽNOM PROGRAMIRANJU

Unapređenja jezika P4 za programiranje mrežnih uređaja

P4 je programski jezik dizajniran da omogući programiranje mrežnih uređaja i logike prosleđivanja paketa. Za razliku od jezika opšte namene kao što je C ili Python, P4 je jezik specifičan za domen sa nizom konstrukcija optimizovanih oko prosleđivanja mrežnih podataka. P4 je jezik otvorenog koda, permisivno licenciran i održava ga neprofitna organizacija zvana P4 jezički konzorcijum.

U okviru zadatka potrebno je:

- Upoznati se sa definicijom standarda za verziju R4-16 jezika
- Dodati novi konstrukt u gramatiku jezika za računanje veličine zaglavlja paketa.
- Doprineti razvoju jezika rešavanjem nekog od javno prijavljenih problema.
- Rešenja validirati sopstvenim testovima

Rešenje realizovati u programskom jeziku S++ a kao razvojno okruženje koristiti Linux operativni sistem. Student će se realizacijom zadatka upoznati sa paradigmama iz oblasti mrežnih komunikacija čija rešenja nudi novi jezik R4. Proći će kroz proces rada i doprinosa na projektu otvorenog koda.

Implementacija razvojnog okruženja za P4 jezik nadogradnjom VSCODE alata

Visual Studio Code (VSCODE) je editor otvorenog koda razvijan pod okriljem Microsofta koji ima podršku za mnoštvo programskih jezika. Osnovna prednost VSCODE ogleda se u tome što je program otvorenog koda, ne zahteva mnogo hardverskih resursa, radi na svim zastupljenim operativnim sistemima (Windows, Linux, Mac), jednostavan je za korišćenje i moguće ga je nadograditi podrškom za nove programske jezike na načine koji su opisani detaljnom dokumentacijom.

U okviru zadatka potrebno je:

- Upoznati se strukturom VSCODE alata i pravilima koje treba ispoštovati

za njegovo proširenje

- Upoznati se sa definicijom standarda za P4-16 verziju programskog jezika
- Upoznati se sa P4 programskim prevodiocem i iskorstiti njegov analizator sintakse prilikom realizacije rešenja
- Implementirati jezički server dodatak (Language Server Extension) za VSCODE alat sa sledećim funkcionalnostima:
  - Semantička provera ispravnosti napisanog P4 programa
  - Iskoristiti API Language Server protokola i dodati podršku za funkcionalnosti kao što su:
    - Prepoznavanje sintakse i ključnih reči jezika
    - Automatsko prepoznavanje i dovršavanje prepoznatih konstrukta tokom samog kucanja
    - Implementacija dodatka za skok na definiciju funkcija/promenljivih
    - Pretraga svih pojavljivanja nekog simbola
    - Ponuda svih mogućnosti iz konteksta nekog simbola
- Rešenje validirati sopstvenim testovima

Rešenje realizovati u programskom jeziku po slobodnom izboru. Tehnologije koje se najčešće koriste su C++, Typescipt, Node.js, Python. Student će realizacijom ovog zadatka doprineti zajednici otvorenog koda okupljenoj oko P4 projekta i pružiti jedan veoma bitan alat P4 programerima za olakšani i brz razvoj aplikacija za mrežne uređaje.

5. QEMU

QEMU je trenutno jedan od najaktivnijih projekata u zajednici otvorenog koda. Bavi se problematikom izvršavanja programa prevedenih za jedan procesor na drugom (različitom) procesoru, kao i virtuelnim uređajima i računarskim sistemima. Iskustvo u radu sa QEMU-om je poželjno i korisno svima koji žele da se bave sistemskim programiranjem.

Sortiranje nizova male dužine za MIPS procesore korišćenjem optimalnih mreža za sortiranje

Sortiranje je jedan od osnovnih elemenata brojnih algoritama i softverskih sistema. Sortiranje nizova male dužine je moguće ubrzati paralelizacijom pomoću optimalnih mreža za sortiranje. Ova tema je dopadljivo predstavljena još u trećem tomu Knut-ove "Umetnosti kompjuterskog programiranja". Tema ovog rada je implementacija takve paralelizacije za MIPS procesore korišćenjem MSA SIMD instrukcijskog seta.

U okviru zadatka potrebno je:

- Upoznati se za MIPS MSA instrukcijskim setom.
- Pored odgovarajućih programa za sortiranje nizova dužine ne veće od 10, potrebno je i uporediti njihove performanse sa gotovim funkcijama za sortiranje std::sort i qsort.
- Poželjna su i eksperimentisanja sa poboljšanim "quick sort" i "merge sort" algoritmima, modifikovanim tako da koriste programe iz ovog rada.
- Obaviti testiranje pomoću emulatora QEMU, kao i na dostupnom MIPS HW-u.

Tehnologije koje će biti korišćene u ovom radu: SIMD, asembler, C, C++, QEMU.

Poređenje SIMD instrukcijskih setova Altivec (PowerPC) i MSA (MIPS)

U poslednjih 30-tak godina je naglo rastao interes za paralelizaciju u računarskim sistemima, i u tom svetlu, razvijena su brojna SIMD proširenja instrukcij-

skih setova. Dva takva proširenja su Altivec (za PowerPC procesore) i MSA (za MIPS procesore).

U okviru zadatka potrebno je:

- Rad treba da obuhvati poredjenje ova dva seta, i to organizovano po grupama instrukcija:
  - instrukcije za učitavanje i skladištenje;
  - instrukcije za manipulaciju podataka u registrima (permu tovanje, pomeranje, brojanje vodećih nula i slično);
  - logičke instrukcije;
  - aritmetičke celobrojne instrukcije;
  - instrukcija za brojeve u pokretnom zarezu.
- Takođe treba predstaviti i uporediti podršku prevodilaca za ova dva seta instrukcija.
- Naposletku, potrebno je izraditi jedan primer programa manje slozenosti koji postize identicnu funkcionalnost na oba procesora, ali pritom koristeci specificne instrukcije za te procesore. Koristiti QEMU prilikom testiranja.

Tehnologije koje će biti korišćene u ovom radu: SIMD, asembler, gcc, QEMU.

Prilog metodologiji merenja performansi emulatora QEMU

Emulator QEMU je veliki softverski sistem koji pokriva emulaciju velikog broja ciljnih procesora, a takodje i uredjaja i računarskih sistema. Ovaj rad treba da se fokusira na samo nekoliko aspekata iz tog mnoštva.

U okviru zadatka potrebno je:

- Izraditi programe za merenje odabranih pojedinačnih instrukcija iz instrukcijskih setova ARM, MIPS i PowerPC platformi (odabrane instrukcije treba da imaju ekvivalentnu ili približno ekvivalentnu funkcionalnost na sve tri platforme).
- Prilagoditi nekoliko bencmark programa iz skupa MiBench svim navede-

nim platformama, i izmeriti odgovarajuće performanse.
- Ispitati uticaj opcija prevodioca za optimizaciju na rezultate.
- Sve dobijene rezultate je potrebno predstaviti graficki, i istražiti koji metod vizualizacije najbolje ilustruje dobijene informacije.
- Uraditi, takodje, kvalitativnu analizu dobijenih rezultata.

www.rt-rk.uns.ac.rs
jobs@rt-rk.com

Milutina Milankovića 19B
11070 Beograd
Srbija