



**УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**



Немања Поповић

Имплементација чувања ЕИТ података ослањајући се на SQLite базу података

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2014.

	УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Датум:
	ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА	Лист/Листова: 2/48

(Податке уноси предметни наставник - ментор)

Врста студија:	<input checked="" type="checkbox"/> Основне академске студије <input type="checkbox"/> Основне струковне студије
Студијски програм:	Рачунарство и аутоматика
Руководилац студијског програма:	проф. др Никола Јорговановић

Студент:	Немања Поповић	Број индекса:	E13533
Област:	Рачунарска техника и рачунарске комуникације		
Ментор:	проф. др Миодраг Темеринац		
НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА: <ul style="list-style-type: none"> - проблем – тема рада; - начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; - литература 			

НАСЛОВ ДИПЛОМСКОГ (BACHELOR) РАДА:

ИМПЛЕМЕНТАЦИЈА ЧУВАЊА ЕИТ ПОДАТАКА ОСЛАЊАЈУЋИ СЕ НА SQLITE БАЗУ ПОДАТАКА

ТЕКСТ ЗАДАТКА:

<p>Циљ рада јесте извршити миграцију тренутног решења чувања ЕИТ података са тренутног нестандардног решења на SQL базу. SQLite и база ће обезбедити бржи, модуларнији и стабилнији приступ ЕИТ подацима од тренутног нестандардног решења. У оквиру реализације задатка се покрива неколико целина: интеграција SQLite базе података у решење програмске подршке за дигиталну телевизију, дизајн базе података према потребним/доступним DVB подацима и захтевима за филтрирање и сортирање и интеграција SQLite базе података на тренутно решење за руковање ЕИТ подацима. Као циљ рада потребно је постићи једнаку функционалност као и тренутно решење и упоредити резултате перформанси између ових решења. Метрика: поређење са тренутним решењем (перформансе и меморијски отисак).</p>
--

Руководилац студијског програма:	Ментор рада:

Примерак за: <input type="checkbox"/> - Студента; <input type="checkbox"/> - Ментора
--



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:		
Идентификациони број, ИБР:		
Тип документације, ТД:	Монографска документација	
Тип записа, ТЗ:	Текстуални штампани материјал	
Врста рада, ВР:	Завршни (Bachelor) рад	
Аутор, АУ:	Немања Поповић	
Ментор, МН:	проф. др Миодраг Темеринац	
Наслов рада, НР:	Имплементација чувања EIT података ослањајући се на SQLite базу података	
Језик публикације, ЈП:	Српски / латиница	
Језик извода, ЈИ:	Српски	
Земља публикавања, ЗП:	Република Србија	
Уже географско подручје, УГП:	Војводина	
Година, ГО:	2014	
Издавач, ИЗ:	Ауторски репринт	
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, ФО: <small>(поглавља/страни/ цитата/табела/слика/графика/прилога)</small>	5/38/0/11/9/0/0	
Научна област, НО:	Електротехника и рачунарство	
Научна дисциплина, НД:	Рачунарска техника	
Предметна одредница/Кључне речи, ПО:	EIT подаци, SQLite база података	
УДК		
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, ВН:		
Извод, ИЗ:	У раду је реализована интеграција SQLite базе података у Comedia 2.0 програмско решење, као и прелазак чувања EIT података са нестандардног начина на SQLite базу података	
Датум прихватања теме, ДП:		
Датум одбране, ДО:		
Чланови комисије, КО:	Председник: доц. др Јелена Ковачевић	Потпис ментора
	Члан: доц. др Милош Сланкаменац	
	Члан, ментор: проф. др Миодраг Темеринац	



UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, ANO :		
Identification number, INO :		
Document type, DT :	Monographic publication	
Type of record, TR :	Textual printed material	
Contents code, CC :	Bachelor Thesis	
Author, AU :	Nemanja Popović	
Mentor, MN :	PhD Miodrag Temerinac	
Title, TI :	Implementation of the EIT data storage relying on SQLite database	
Language of text, LT :	Serbian	
Language of abstract, LA :	Serbian	
Country of publication, CP :	Republic of Serbia	
Locality of publication, LP :	Vojvodina	
Publication year, PY :	2014	
Publisher, PB :	Author's reprint	
Publication place, PP :	Novi Sad, Dositeja Obradovica sq. 6	
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)	5/38/0/11/9/0/0	
Scientific field, SF :	Electrical Engineering	
Scientific discipline, SD :	Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, S/KW :	EIT data, SQLite database	
UC		
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, N :		
Abstract, AB :	The work was carried integration SQLite database Comedia 2.0 software solution, and moving storage EIT data with a non-standard way in SQLite database	
Accepted by the Scientific Board on, ASB :		
Defended on, DE :		
Defended Board, DB :	President: Jelena Kovačević, PhD	Mentor's sign
	Member: Miloš Slankamenac, PhD	
	Member, Mentor: Miodrag Temerinac, PhD	

Zahvalnost

Zahvaljujem se svom mentoru prof. dr Miodragu Temerincu na stručnoj pomoći tokom izrade završnog (*bachelor*) rada.

Posebno se zahvaljujem Miletu Davidoviću, Tatjani Samardžić, Tijani Jević, Đorđu Glišić i Dušanu Živkov na stručnoj pomoći, savetima i utrošenom vremenu.

Zahvaljujem se rukovodstvu RT-RK na ukazanoj prilici da se bolje upoznam sa načinom rada u inženjerskom okruženju i budem uključen u proces razvoja novih programskih rešenja.

Na kraju se zahvaljujem svima onima koji su na bilo koji način doprineli izradi ovog završnog rada.



SADRŽAJ

1. Uvod	1
2. Teorijske osnove.....	3
2.1 Digitalna televizija.....	3
2.2 SQLite	4
2.3 Comedia	4
2.3.1 Upravljač informacionim događajima.....	7
2.3.2 Sistem datoteka za upravljanje podacima sa mogućnošću samooporavka po nasilnom isključivanju.....	8
3. Koncept rešenja	9
3.1 Integracija SQLite baze podataka u Comedia 2.0 programsko rešenje	9
3.2 Prelazak sa trenutnog načina čuvanje EIT podataka na SQLite bazu podataka	11
4. Programsko rešenje.....	20
4.1 Integracija SQLite baze podataka u Comedia 2.0 programsko rešenje	20
4.2 Prelazak sa nestandardnog načina čuvanja EIT podataka na SQLite bazu podataka	24
5. Rezultati.....	29
5.1 Testiranje integracije SQLite baze podataka u Comedia programsko rešenje .	32
5.2 Testiranje prelaska čuvanja EIT podataka sa statičkih lista na SQLite bazu podataka	33
5.2.1 Provera da li su svi podaci prethodno čuvani u dinamičkoj listi sačuvani i u SQLite bazu podataka.....	33
5.2.2 Zauzetost radne memorije samo za integraciju SQLite baze podataka u EIM modul i poređenje zauzete radne memorije na pozivu funkcija za čitanje podataka iz dinamičke liste i iz SQLite baze podataka.....	34
6. Zaključak	36
7. Literatura	38

SPISAK SLIKA

Slika 2.1 Prikaz Comedia 2.0 arhitekture	5
Slika 2.2 Tok kreiranja i prikazivanja binarne datoteke	6
Slika 2.3 Prikaz uloge <i>Maestro player</i> -a u Comedia programskom rešenju	7
Slika 3.1 Prikaz Comedia arhitekture sa integrisanom SQLite baza podataka.....	10
Slika 3.2 Prikaz adaptacionog sloja između SQLite-a i Comedia jezgra	11
Slika 5.1 MStar platforma na kojoj je testiran rad	30
Slika 5.2 Platforma koja je korišćena za ispise.....	30
Slika 5.3 LAUTERBACH uređaj za otklanjanje grešaka.....	31
Slika 5.4 MStar STB sa MStar platformom za hvatanje ispisa i LAUTERBACH uređajem za otklanjanje grešaka u toku rada	31

SPISAK TABELA

Tabela 3.1 EIT_EVENT Tabela	14
Tabela 3.2 EIM_SHORT_EVENT Tabela	15
Tabela 3.3 EIM_TABLE_EXTENDED_EVENT Tabela	15
Tabela 3.4 EIM_PARENTAL_RATING Tabela	16
Tabela 3.5 EIM_COMPONENT Tabela	17
Tabela 3.6 EIM_MULTI_COMPONENT Tabela.....	17
Tabela 3.7 EIM_CONTENT_IDENTIFIER Tabela.....	18
Tabela 3.8 EIM_CRID.....	19
Tabela 5.1 Utrošena memorija prilikom uključivanja <i>sqlite</i> modula u <i>Comedia 2.0</i>	32
Tabela 5.2 Zauzetost memorije posle testnih funkcija	33
Tabela 5.3 Prikaz stanja memorije nakon pozivanja funkcije za dobijanje liste EIT događaja	35

SKRAĆENICE

- SQL** - *Structured Query Language*, relacioni upitni jezik
- eCos** - *Embedded Configurable Operating System*, ugrađeni podesivi operativni sistem
- EIM** - *Event Information Manager*, upravljač informacionim događajima
- EIT** - *Event Information Table*, tabele informacionih događaja
- DMJFS** - *Data Manager Journalized Files System*, sistem datoteka za upravljanje podacima sa mogućnošću samooporavka po nasilnom isključivanju
- DVB-T** - *Digital Video Broadcasting-Terrestrial*, emitovanje digitalnog zemaljskog video signala
- DVB-C** - *Digital Video Broadcasting-Cable*, emitovanje digitalnog kablovskog video signala
- DVB-S** - *Digital Video Broadcasting-Satellite*, emitovanje digitalnog satelitskog video signala
- ATSC** - *Advanced Television Systems Committee*, napredni televizijski sistemski odbor
- ISDB-T** - *Integrated Services Digital Broadcasting*, integrisane usluge digitalnog emitovanja
- DTMB** - *Digital Terrestrial Multimedia Broadcast*, digitalno zemaljsko multimedijalno emitovanje
- ACID** - *Analysis Console for Intrusion Database*, analiza konzole za neovlašćeni pristup u bazu
- RAM** - *Random Access Memory*, radna memorija
- SDK** - *Software Development Kit*, programski komplet za razvoj

ELG - *European Launching Group*, evropska grupa za nadgledanje razvoja digitalne televizije

HDTV - *High Definition Television*, televizija visoke rezolucije

WYSIWYG - *What You See Is What You Get*, šta vidiš to ćeš i da dobiješ

1. Uvod

U ovom radu je prikazano prilagođavanje SQLite baze podataka za rad na STB (*Set Top Box*) uređaju, kao i integracija SQLite baze podataka u okruženje programske podrške za čuvanje EIT(*Event Information Table*) podataka dostupnih po DVB(*Digital Video Broadcasting*) standardu. Ovaj standard definiše digitalni prenos slike i zvuka zastupljen u Evropi.

SQLite je baza podataka namenjena za rad sa ugrađenim aplikacijama koje rade u realnom vremenu. Veoma je konfigurabilna pa je zbog toga i odabrana za integraciju u Comedia 2.0 programsko rešenje. Ono predstavlja osnovu za programski razvoj aplikacija koje su usko povezane sa Comedia jezgrom i TV dekomerom. Sa ubrzanim razvojem digitalne televizije, sa jedne strane dolazi do sve strožijih zahteva po pitanju performansi i pouzdanosti rada sistema, dok se sa druge strane javlja sve veća potreba za fleksibilnost zbog velikog broja različitih servisa, a samim tim i velike količine EIT podataka koje je potrebno čuvati. U njima se čuvaju sve dodatne informacije koje se emituju na nekom servisu kao što su: emisija koja se trenutno prikazuje, njen kratak opis, vremenski interval u kojem se emituje, šta je sledeće što će da se prikazuje i slično. EIT podaci čuvani u dinamičkim listama preusmereni su na čuvanje u SQLite bazu podataka. Prilagođavanje SQLite baze podataka za Comedia 2.0 programsko rešenje, kao i prelazak čuvanja EIT podataka iz dinamičkih lista u SQLite bazu podataka je urađeno u cilju bržeg, modularnijeg i stabilnijeg pristupa EIT podacima. Rešenje je realizovano na MStar platformi sa MIPS arhitekturom i eCos (*Embedded Configurable Operating System*) operativnim sistemom. Ovaj ugrađeni podesivi operativni sistem omogućava platformi da radi sa veoma malo resursa, što je i karakteristika ugrađenih sistema.

Ovaj rad je opisan u pet poglavlja. Prvo poglavlje predstavljaju teorijske osnove neophodne za razumevanje programske podrške u okviru koje je implementirano

prilagođavanje SQLite baze podataka na pomenutu platformu i čuvanje EIT podataka u bazu, umesto u dinamičku listu gde je do sada čuvano. U drugom poglavlju predstavljen je koncept rešenja koji predstavlja opis realizacije prilagođavanja SQLite baze, projektovanje baze podataka i čuvanje dostupnih EIT podataka u projektovanu bazu. Treće poglavlje predstavlja programsko rešenje sa opisom modula korišćenih tokom rada. Četvrto poglavlje čine opis i rezultati testiranja. U petom poglavlju je kratko predstavljeno sve što je urađeno sa eventualnim mogućnostima za dalji razvoj i dodatno usavršavanje trenutnog rešenja.

2. Teorijske osnove

2.1 Digitalna televizija

Digitalna televizija je prenos audio i video signala digitalno obrađenog od strane multipleksiranih signala, za razliku od potpuno analognih kanala i razdvojenih signala korišćenih u analognoj televiziji. Digitalna televizija može da podrži više od jednog kanala u jednom propusnom opsegu. To je inovativni servis koji predstavlja prvu značajnu evoluciju u televizijskoj tehnologiji još od televizora u boji iz 1950.

Tokom 1991, proizvođači opreme za emitovanje i prijem TV programa dogovaraju se kako da oforme zajedničku evropsku platformu za razvoj zemaljske digitalne televizije. Krajem iste godine emiteri programa, proizvođači opreme i organizacije za definisanje standarda dogovaraju formiranje grupe koja će nadgledati razvoj digitalne televizije u Evropi – ELG(*European Launching Group*). ELG se proširuje i okuplja vodeće evropske medijske interesne grupe, kako privatne tako i javne, proizvođače potrošačke elektronike, distributere i regulatorna tela. Septembra 1993. članovi ELG potpisuju dokument *Memorandum of Understanding* koji utvrđuje pravila rada, a ELG postaje DVB. Razvoj digitalne televizije, koji je u to vreme u Evropi već bio otpočeo, nastavlja se ubrzanim tempom. Priprema se studija o mogućnostima uvođenja zemaljske digitalne televizije u Evropi. Uvode se novi koncepti koji vode računa o razlikama potencijalnih krajnjih korisnika kao što su prenosna televizija i HDTV (*High Definition Television*). Evropska industrija za proizvodnju satelitske opreme za TV emisiju prateći nova zbivanja u potpunosti prelazi na digitalnu TV tehnologiju. DVB organizuje forume koji okupljaju sve vodeće evropske televizijske interesne grupe, što pruža šansu da se kompletan evropski digitalni televizijski sistem zasnuje na zajedničkom pristupu. Postaje jasno da će satelitska i kablovska televizija početi prva sa emitovanjem DTV programa. Manje tehničkih problema, jednostavniji pravni propisi i prioriteta tržišta uslovlili su

da se ova dva sistema prenosa razvijaju brže od zemaljskih DTV sistema. Do 1997 razvoj DVB projekta je uspešno pratio početne planove i projekat je ušao u sledeću fazu, javno objavljivanje standarda koji će biti raspoloživi svima i pretvaranjem ideje o digitalnoj televiziji u stvarnost.

Mnoge zemlje su zamenile emitovanje analogne televizije digitalnom televizijom i dozvoljavaju upotrebu drugih televizijskih radio opsega. Nekoliko regiona u svetu su u različitim fazama adaptacije i sprovode različite standarde emitovanja. Najbitniji načini prenosa digitalnog televizijskog signala su:

- ❖ Zemaljski DVB-T, ATSC, ISDB-T, DTMB
- ❖ Kablovski DVB-C
- ❖ Satelitski DVB-S

2.2 SQLite

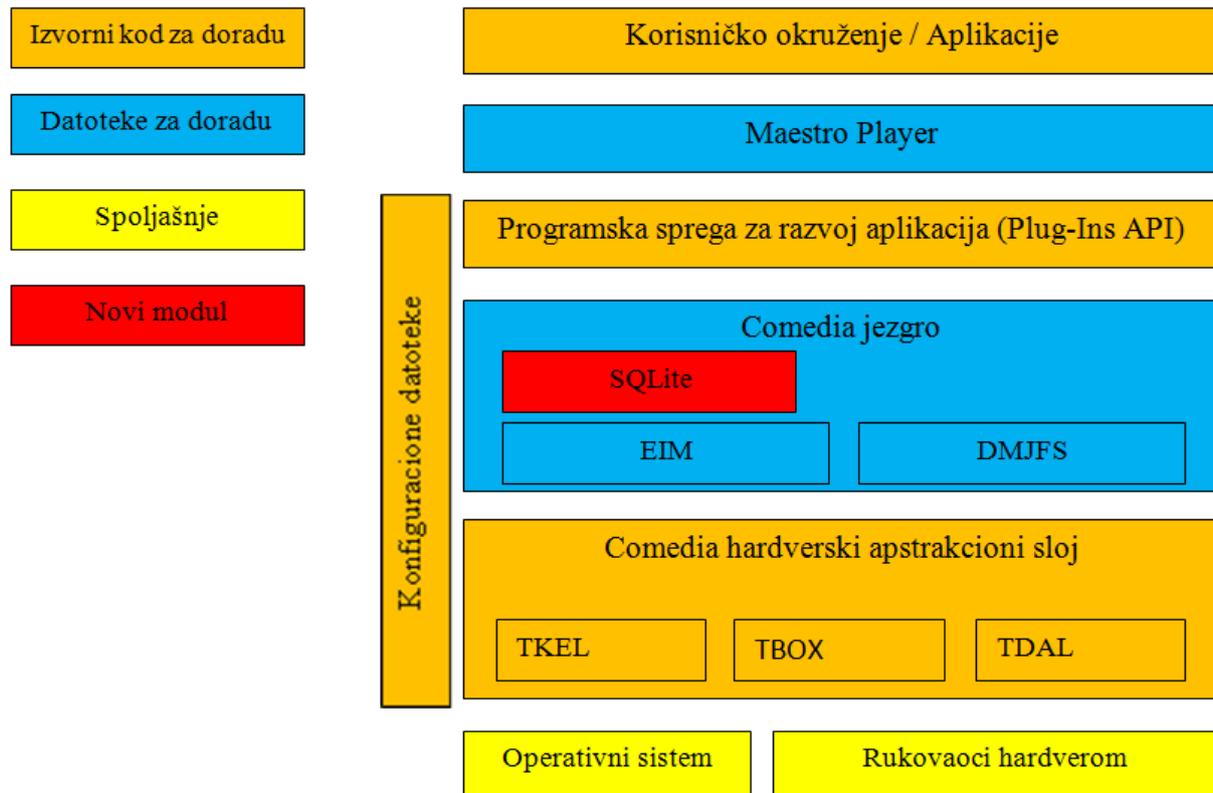
SQLite je ACID-kompatibilan (ACID - *Analysis Console for Intrusion Databases*) ugrađen sistem za upravljanje bazama podataka sadržan u relativno maloj (~225 KB) C programskoj biblioteci. Izvorni kod za SQLite je u javnom vlasništvu.

Za razliku od klijent-server sistema za upravljanje bazama podataka, jezgro SQLite-a nije samostalan proces sa kojim aplikacija komunicira. Umesto toga, SQLite baza podataka je uvezana i postaje sastavni deo aplikacije. Baza podataka može biti dinamično pozivana. Aplikacija koristi SQLite funkcionalnost kroz proste funkcionalne pozive, koji smanjuju latenciju u pristupu bazama podataka kako su funkcionalni pozivi unutar jednog procesa efikasniji od međuprocenke komunikacije. Kompletna baza podataka (definisane, table, indeksi i sami podaci) je kao jedan međuplatformna datoteka na mašini domaćinu. Ovako prost dizajn je postignut zaključavanjem kompletnog fajla baze podataka na početku transakcije.

SQLite je ugrađen u sve veći broj popularnih programa. Na primer, Mozila Firefox pohranjuje mnoštvo konfiguracionih podataka, (bukmarkovi, kukiji, itd.), u interno upravljano SQLite bazu podataka. Kao drugi primer, Guglov Android operativni sistem za mobilne telefone i druge male uređaje sadrži SQLite.

2.3 Comedia

Comedia je posrednik za STB i TV. Comedia 2.0 ima za cilj da dekoduje DVB format. Ovo je osnova za programski razvoj aplikacija koje su usko povezane sa Comedia jezgrom i TV dekomerom. Arhitektura programske podrške je prikazana na sledećoj slici:



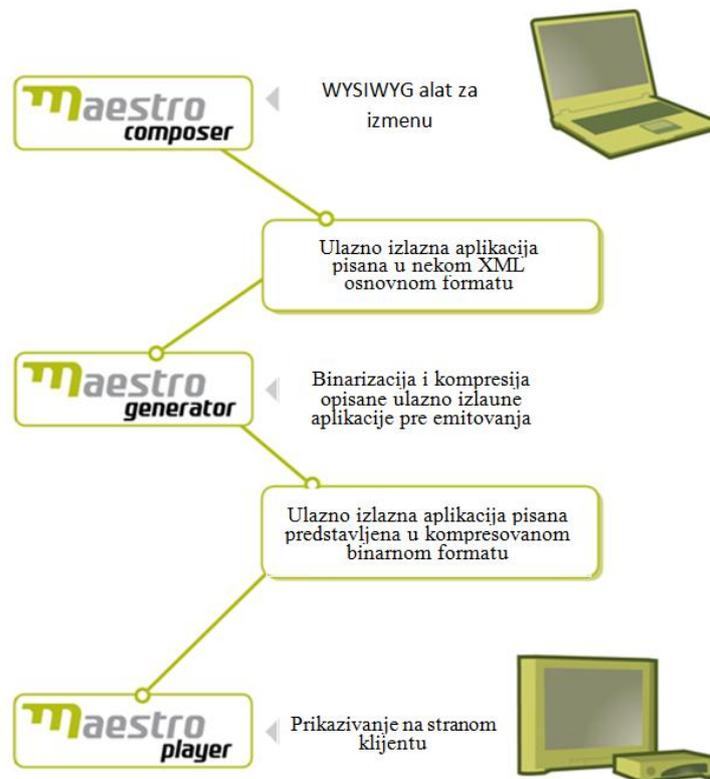
Slika 2.1 Prikaz Comedia 2.0 arhitekture

Comedia hardverski apstrakcioni sloj služi za prilagođavanje Comedia programskog rešenja operativnom sistemu na koji naleže. Ta prilagođavanja se vrše kroz TKEL, TBOX i TDAL module. TKEL je sloj za apstrakciju funkcionalnosti operativnog sistema realnog vremena. TBOX je modul koji deklariše pomoćne funkcije i makroe i služi u svrhu otklanjanja grešaka programske podrške. TDAL je sloj za apstrakciju rukovalaca (*drivers*) i sastoji se od mnoštva modula. Svaki blok fizičke arhitekture DTV prijemnika poseduje odgovarajući TDAL rukovalac.

Comedia jezgro realizuje najvažnije operacije (rašćlanjivanje DVB podataka, kontrola pristupa i organizacija servisa, prikupljanje EIT podataka, podrška za snimanje, kontrola podsetnika, dekodovanje multimedijalnih sadržaja). Komunikacijom sa nižim slojevima programske podrške se kontroliše fizička arhitektura. Obezbeđuje potrebne programske sprege ka višim programskim slojevima (grafičkoj korisničkoj sprezi). Većina proizvođača DTV softvera koji se koristi u DTV prijemnicima ne ograničava svoje usluge samo na osnovne OS funkcije već se OS prodaje kao deo kompletnog paketa koji uključuje i tzv. *middleware* sloj. *Middleware* obezbeđuje odgovarajući API koji apstrakuje funkcionalnost TV uređaja, HW platforme, kao i OS funkcija, čime je omogućeno da proizvođači aplikativnog DTV softvera ne moraju da poznaju hardverske specifičnosti DTV prijemnika. Ovim je omogućeno da se aplikacije koje koriste isti *middleware* API mogu izvršavati na

različitim hardverskim platformama, tj. *middleware* formira virtuelnu mašinu. Osnovni problem kod *middleware*-a je nepostojanje jedinstvenog standarda. Mnogi proizvođači softvera nude svoja rešenja koja su međusobno nekompatibilna.

Programska sprega za razvoj aplikacija sadrži funkcije koje koristi *Maestro Player*. On izvršava binarnu datoteku koju generiše *Maestro Composer*. Na sledećoj slici je prikazan tok od kreiranja da prikaza binarne datoteke:



Slika 2.2 Tok kreiranja i prikazivanja binarne datoteke

Maestro player pored prikaza binarne datoteke kontroliše pametnu karticu, spoljne događaje i reakcije na daljinski upravljač, nakon čega se na kraju sve prikazuje na ekran preko korisničkog okruženja. Na sledećoj slici je prikazano kako funkcioniše *Maestro player*.



Slika 2.3 Prikaz uloge *Maestro player*-a u Comedia programskom rešenju

2.3.1 Upravljač informacionim događajima

EIM (*Event Information Manager*) je jedan od integralnih modula Comedia programskog rešenja. Uloga EIM modula je da obezbedi detaljne informacije o događajima koji se emituju na određenim servisima. EIM modul koristi Table modul da pristupi EIT sekcijama koje nose informacije o događajima i da parsira njihov sadržaj. Ovaj modul obavlja lokalno snimanje podataka za zadati maksimalni vremenski interval. Jedinstven način za dobavljanje događaja od strane EIM modula je korišćenjem *Window* zahteva. Klijent može da preuzme listu događaja korišćenjem funkcije *EIM_WindowEventGet*. Prilikom poziva ove funkcije moraju se zadati sledeći argumenti:

- ❖ Režim zahteva (EIT *prisutan/sledeći*, vreme početka emitovanja je unutar zadatog intervala, emisija je potpuno ili delimično unutar zadatog intervala ili za emisije koje se potpuno ili delimično emituju u intervalu sa zadatim trajanjem, računato od ponoći)
- ❖ Vremenski interval za koji se emituju događaji (sa početno vreme i vreme trajanja događaja)
- ❖ Lista servisa za koje treba dobiti događaje

2.3.2 Sistem datoteka za upravljanje podacima sa mogućnošću samooporavka po nasilnom isključivanju

Platforma na kojoj je realizovan ovaj rad koristi samo fleš memoriju. DMJFS (*Data Manager Journalized Files System*) je modul koji omogućava rad sa fleš memorijom u okviru Comedia programskog rešenja. Funkcije koje on podržava su namenjene za rad sa 3 vrste sistemskih objekata:

- ❖ Particije: postavljanje, uklanjanje, reset i status
- ❖ Direktorijume: kreiranje, brisanje, otvaranje, zatvaranje i status
- ❖ Datoteke: otvaranje/kreiranje, čitanje, pisanje, pozicioniranje, pražnjenje, zatvaranje i status

DMJFS radi na nivou blokova. Blokovi nisu prepisivi, što znači da blok koji treba da bude prepisan prvo se obriše pa zatim nastavlja sa upisom. Upis se uvek vrši posle poslednjeg upisanog bajta, a kada je blok pun, ciklično se briše da bi sledeći blok nastavilo sa upisom. Ova karakteristika omogućava da se zadrže stare verzije podataka i rukovanje oporavkom ako je više blokova konfigurisano u particiju.

3. Koncept rešenja

Realizacija ovog rada prvenstveno se oslanja na korišćenje Comedia 2.0, već postojećih raščlanjivača EIT podataka, EIM, TKEL i TDAL modula, kao i modula za rukovanjem sa fleš memorijom (DMJFS). Comedia programsko rešenje na kojem je realizovan ovaj rad ne podržava hard disk i standardni sistem datoteka koji koriste funkcije za rad sa datotekama. Da bi se to omogućilo SQLite bazi podataka, tokom njene integracije u EIM modul neophodno je bilo izvršiti i njeno prilagođavanje sistemu datoteka koji koristi ova platforma, odnosno fleš memoriji i DMJFS sistemu datoteka. Integracija SQLite baze podataka u Comedia programsko rešenje je realizovana kao dodavanje novog modula u Comedia jezgro, tako da je omogućeno njeno korišćenje i u druge svrhe a ne samo u EIM modulu. Rešenje je realizovano iz sledećih celina:

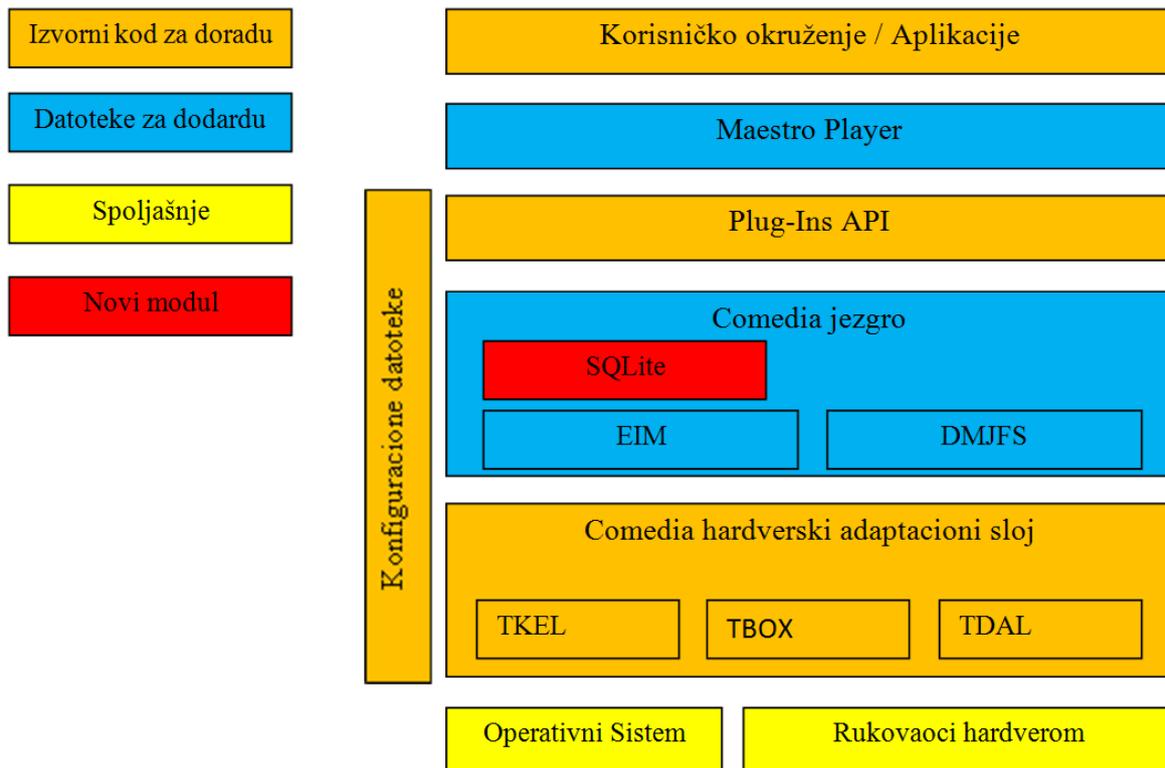
- ❖ Integracija SQLite baze podataka unutar Comedia programskog rešenja
- ❖ Prelazak sa trenutnog načina čuvanja EIT podataka na čuvanje korišćenjem SQLite baze podataka

3.1 Integracija SQLite baze podataka u Comedia 2.0 programsko rešenje

SQLite je pisan tako da može da radi na *windows* i *linux* platformama sa standardnim sistemom datoteka. Platforma koja je korišćena tokom dodavanja SQLite baze podataka kao novog modula, ne koristi standardni sistem datoteka već fleš memoriju namenjenu za trajno čuvanje podataka.

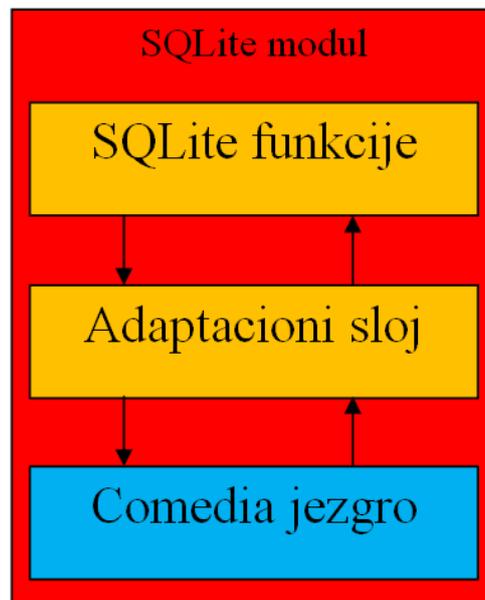
Integracija SQLite baze podataka obuhvata dodavanje novog modula u Comedia programsko rešenje u Comedia jezgro. Ovaj novi modul se oslanja na TKEL, TDAL i DMJFS module. Na sledećoj slici je prikazano kako je organizovana Comedia i gde je integrisan novi modul (*SQLite*).

Na sledećoj slici je prikazana *Comedia 2.0*, tačnije njena arhitektura i organizacija modula koje ona koristi.



Slika 3.1 Prikaz Comedia arhitekture sa integrisanom SQLite baza podataka

U zavisnosti od ciljne platforme i operativnog sistema koji se koristi, često se razlikuje rad sa sistemskim pozivima zaduženim za rukovanje memorijom. Da bi se ovaj problem lakše rešavao sa promenom platforme, prelaskom na drugi operativni sistem ili integracija ovog modla za neke druge svrhe, napravljen je jedan adaptacioni sloj preko kojeg funkcionišu pozivi ka fleš memoriji i sistemski pozivi za funkcije koje su specifične za ciljnu platformu na koju se vrši integracija SQLite baze podataka.



Slika 3.2 Prikaz adaptacionog sloja između SQLite-a i Comedia jezgra

Posao ovog adaptacionog sloja je da pozive upućene ka memoriji prilagodi obliku koji zahteva platforma, a povratne vrednosti funkcija platforme prilagodi zahtevima SQLite-a.

Zbog trenutne konfiguracije operativnog sistema na kojem radi ovaj uređaj za reprodukciju digitalnog televizijskog signala i njegove maksimalne optimizacije po pitanju performansi, neke systemske funkcije koje koristi SQLite nedostaju. Pored tih funkcija koje nedostaju postoje i neke funkcije koje su sa istim nazivom ali im se razlikuju neki ili svi ulazno-izlazni argumenti pa čak i povratne vrednosti tih funkcija. Iz tog razloga je morao da se napravi jedan adaptacioni sloj koji će da razdvoji te funkcije koje su specifične za ovaj operativni sistem a koristi ih SQLite.

Sa korišćenjem ovog modula na nekoj drugoj platformi koja koristi nekakav drugačiji način čuvanja podataka na masovnu ili fleš memoriju, potrebne su samo eventualne izmene tela funkcija koje se nalaze u pomenutom adaptacionom sloju.

3.2 Prelazak sa trenutnog načina čuvanje EIT podataka na SQLite bazu podataka

Trenutni način čuvanja EIT podataka je vlasnički definisano rešenje. To rešenje se sastoji od C-ovskih lista. Liste su napravljene od struktura koje ujedno predstavljaju i element te liste. Te strukture kao jedno polje sadrže novu strukturu koja se odnosi na jedan EIT događaj. Liste koje nose EIT događaje se razlikuju u zavisnosti od mesta njihovog korišćenja

u EIM-u, ali podatak koji one nose na kraju uvek završi u istoj strukturi koja predstavlja jedan EIT događaj.

Sa integracijom SQLite baze podataka u EIM, podaci čuvani u listama se u funkcijama za rščlanjivanje EIT događaja čuvaju u SQLite bazu podataka umesto u listu koja čuva EIT podatke. SQLite baza EIT podatke čuva u nekoliko tabela. Baza je organizovana tako da postoji jedna tabela koja je glavna, dok se ostale povezuju na nju pomoću stranog ključa. U glavnoj tabeli se čuvaju podaci koji se direktno upisuju u strukturu koja predstavlja EIT događaj, dok se u pomoćnim tabelama čuvaju podaci koji predstavljaju liste u strukturi koja je EIT događaj. Pomoćne tabelle su povezane sa glavnom tabelom pomoću stranog ključa zbog kaskadnog brisanja elemenata iz baze.

Kaskadno brisanje elemenata je korisno radi bržeg brisanja EIT događaja iz baze nakon isteka njegovog vremena trajanja. Ono nam omogućava da se na brisanju elemenata iz glavne tabelle obrišu i elementi iz svih ostalih tabela i pomoćnih tabela koji su pomoću stranog ključa međusobno povezani. Ovim se postiže da se samo jednom komandom za brisanje elementa iz glavne tabelle obrišu i elementi iz svih ostalih tabela koji su stranim ključem povezani sa zadatim elementom za brisanje. To znači da ako se obriše element iz glavne tabelle, biće obrisani i elementi iz svih ostalih tabela koji imaju isti strani ključ, tj ključ elementa iz glavne tabelle. Sledećim tabelama su prikazane tabelle i organizacija baze u kojoj se čuvaju EIT podaci:

EIT_EVENT Tabela	
US_EVENT_ID	PRIMARNI KLJUČ
UI_TRANSPORT_STREAM_ID	PRIMARNI KLJUČ
UI_ORIGINAL_NETWORK_ID	PRIMARNI KLJUČ
UI_SERVICE_ID	PRIMARNI KLJUČ
UC_DEMUX_ID	PRIMARNI KLJUČ
UC_EIT_PF_FLAG	PRIMARNI KLJUČ
UC_EIT_SCHEDULE_FLAG	PRIMARNI KLJUČ
UL_COMPRESSED_START_TIME	
UL_COMPRESSED_STOP_TIME	
UC_RELATE_WINDOWS_COUNT	
AB_WIN_ID_LIST	
US_REFERENCE_COUNT	
ST_START_TIME	

ST_DURATION_TIME	
UC_VERSION_NUMBER	
E_RUNNING_STATUS	
B_FREE_CA_MODE	
E_TYPE	
UC_NB_PARENTAL_RATING	
UC_CONTENT1	
UC_CONTENT2	
UC_LINKAGE_TYPE	
LINKAGE_PUC_PRIVATE_DATA_BYTE	
LINKAGE_UI_TRANSPORT_STREAM_ID	
LINKAGE_UI_ORIGINAL_NETWORK_ID	
LINKAGE_UI_SERVICE_ID	
LINKAGE_UC_DEMUX_ID	
LINKAGE_UC_EIT_PF_FLAG	
LINKAGE_UC_EIT_SCHEDULE_FLAG	
BROADCAST_ID	
BROADCAST_UC_LANGUAGE	
BROADCAST_PUC_SELECTOR_BYTE	
BROADCAST_PUC_TEXT	
TELEPHONE_B_FOREIGN_AVAILABILITY	
TELEPHONE_UC_CONNECTION_TYPE	
TELEPHONE_PUC_COUNTRY_PREFX	
TELEPHONE_PUC_INTERNATIONAL_AREA_CODE	
TELEPHONE_PUC_OPERATOR_CODE	
TELEPHONE_PUC_NATIONAL_AREA_CODE	
TELEPHONE_PUC_CORE_NUMBER	
TIME_SHIFTED_US_REFERENCE_SERVICE_ID	
TIME_SHIFTED_US_REFERENCE_EVENT_ID	
PREF_NAME_IDENTIFIER_UC_ID	

PRODUCT_INFO_PRICE	
PRODUCT_INFO_ID	
DVB_ICON_EXTENSION	
DVB_ICON_DATA	
DVB_ICON_ICON_SIZE	
MEM_ALLOC	
MEM_FREE	
STX_ALLOC	
STX_FREE	
EXT_ALLOC	
EXT_FREE	

Tabela 3.1 EIT_EVENT Tabela

Prethodna tabela predstavlja glavnu tabelu na koju su sve ostale tabele povezane pomoću stranog ključa. Obeležja koja se nalaze u ovoj tabeli se odnose na polja strukture koja predstavlja jedan EIT događaj. Ova obeležja ne predstavljaju polja koja su liste definisane kao polje strukture koja predstavlja EIT događaj. Pošto se liste u C-u definišu pomoću struktura, svaka struktura koja predstavlja element liste pored pokazivača na sledeći element ima i dodatna polja koja nose informacije o elementu te liste. Zbog toga je za svaku listu koja je predstavljena kao polje glavne strukture definisana nova pomoćna tabela. Sledeće tabele prikazuju šta je njihov primarni ključ, šta je strani ključ i šta su obeležja svake od tabela.

EIM_SHORT_EVENT Tabela	
AUTO_INCREMENT	PRIMARNI KLJUČ
US_EVENT_ID	STRANI KLJUČ
UI_TRANSPORT_STREAM_ID	STRANI KLJUČ
UI_ORIGINAL_NETWORK_ID	STRANI KLJUČ
UI_SERVICE_ID	STRANI KLJUČ
UC_DEMUX_ID	STRANI KLJUČ
UC_EIT_PF_FLAG	STRANI KLJUČ
UC_EIT_SCHEDULE_FLAG	STRANI KLJUČ
UC_NB_SHORT_EVENT	
UC_LANGUAGE_CODE	

PUC_EVENT_NAME	
PUC_EVENT_DESCRIPTION	
STRANI KLJKUČ povezan sa EIT_EVENT tabelom	

Tabela 3.2 EIM_SHORT_EVENT Tabela

U jednom događaju kratki opisi za njega mogu da stižu na više jezika i za svaki jezik opis na tom jeziku, ovi podaci se prihvataju u obliku liste kao jedno polje glavne strukture koja čuva EIT podatke. Da bi se ti podaci sačuvali u bazu neophodno je bilo napraviti novu tabelu u kojoj ključ neće biti isti što i u glavnoj tabeli, jer kada bi bio isti ključ bi se ponavljao onoliko puta koliko ima elemenata pomoćne liste i upis u bazu ne bi bio moguć. Iz tog razloga je ključ EIM_SHORT_EVENT pomoćne tabele auto inkrement. On se automatski uvećava na svakom upisu u ovu tabelu pa samim tim ne postoji mogućnost pojavljivanja istog ključa više puta. Na pretrazi ove tabele se zadaje strani ključ, na taj način će se dobiti opisi na svim jezicima za određeni događaj, a ako je potreban opis za određeni jezik to se može navesti kao uslov pretrage u *select* naredbi.

EIM_TABLE_EXTENDED_EVENT Tabela	
AUTO_INKREMENT	PRIMARNI KLJKUČ
US_EVENT_ID	STRANI KLJKUČ
UI_TRANSPORT_STREAM_ID	STRANI KLJKUČ
UI_ORIGINAL_NETWORK_ID	STRANI KLJKUČ
UI_SERVICE_ID	STRANI KLJKUČ
UC_DEMUX_ID	STRANI KLJKUČ
UC_EIT_PF_FLAG	STRANI KLJKUČ
UC_EIT_SCHEDULE_FLAG	STRANI KLJKUČ
UC_NB_EXTENDED_EVENT	
UC_LANGUAGE_CODE	
UC_DESC_NUMBER	
PUC_ITEM_DESCRIPTION	
PUC_ITEM	
PUC_TEXT	
STRANI KLJKUČ povezan sa EIT_EVENT tabelom	

Tabela 3.3 EIM_TABLE_EXTENDED_EVENT Tabela

Kao i u tabeli EIM_SHORT_EVENT, i u EIM_TABLE_EXTENDED_EVENT tabeli može da dođe do istog problema sa jezicima. Da bi se to rešilo iskorišćen je isti način čuvanja podataka i u ovoj tabeli.

EIM_PARENTAL_RATING Tabela	
AUTO_INCREMENT	PRIMARNI KLJUČ
US_EVENT_ID	STRANI KLJUČ
UI_TRANSPORT_STREAM_ID	STRANI KLJUČ
UI_ORIGINAL_NETWORK_ID	STRANI KLJUČ
UI_SERVICE_ID	STRANI KLJUČ
UC_DEMUX_ID	STRANI KLJUČ
UC_EIT_PF_FLAG	STRANI KLJUČ
UC_EIT_SCHEDULE_FLAG	STRANI KLJUČ
UC_COUNTRY_CODE	
UC_RATING	
STRANI KLJKUČ povezan sa EIT_EVENT tabelom	

Tabela 3.4 EIM_PARENTAL_RATING Tabela

Za različite zemlje emitovanje određenih sadržaja na nekom od servisa može da bude predviđeno za različite uzraste. I u ovoj tabeli je taj problem rešen kao i u prethodnim tabelama.

EIM_COMPONENT Tabela	
AUTO_INCREMENT	PRIMARNI KLJUČ
US_EVENT_ID	STRANI KLJUČ
UI_TRANSPORT_STREAM_ID	STRANI KLJUČ
UI_ORIGINAL_NETWORK_ID	STRANI KLJUČ
UI_SERVICE_ID	STRANI KLJUČ
UC_DEMUX_ID	STRANI KLJUČ
UC_EIT_PF_FLAG	STRANI KLJUČ
UC_EIT_SCHEDULE_FLAG	STRANI KLJUČ
UC_STREAM_CONTETN	

UC_COMPONENT_TYPE	
UC_COMPONENT_TAG	
UC_LANGUAGE_CODE	
PUC_TEXT	
STRANI KLJKUČ povezan sa EIT_EVENT tabelom	

Tabela 3.5 EIM_COMPONENT Tabela

Tekst koji se emituje za različite jezike opet predstavlja isti problem kao i u prve dve pomoćne tabele. Način na koji je ovo rešeno je isti kao i u njima.

EIM_MULTI_COMPONENT Tabela	
AUTO_INKREMENT	PRIMARNI KLJKUČ
US_EVENT_ID	STRANI KLJKUČ
UI_TRANSPORT_STREAM_ID	STRANI KLJKUČ
UI_ORIGINAL_NETWORK_ID	STRANI KLJKUČ
UI_SERVICE_ID	STRANI KLJKUČ
UC_DEMUX_ID	STRANI KLJKUČ
UC_EIT_PF_FLAG	STRANI KLJKUČ
UC_EIT_SCHEDULE_FLAG	STRANI KLJKUČ
UC_COMPONENT_TAG	
UC_NB_COMPONENT_TEXT	
UC_LANGUAGE_CODE	
PUC_TEXT	
STRANI KLJKUČ povezan sa EIT_EVENT tabelom	

Tabela 3.6 EIM_MULTI_COMPONENT Tabela

Polje glavne strukture koje predstavlja listu i čuva se u ovoj tabeli je pod opcijom prevodioca. Ova tabela je potrebna da bi se to polje moglo čuvati kada je opcija prevodioca postavljena da je aktivno čuvanje ovih podataka. I u ovoj tabeli se pojavljuje problem čuvanja podataka na više jezika koji je identičan prethodnim slučajevima i na isti način je i ovde rešen.

EIM_CONTENT_IDENTIFIER Tabela	
NB_IDENTIFIER	PRIMARNI KLJUČ
US_EVENT_ID	STRANI KLJUČ
US_EVENT_ID	STRANI KLJUČ
UI_TRANSPORT_STREAM_ID	STRANI KLJUČ
UI_ORIGINAL_NETWORK_ID	STRANI KLJUČ
UI_SERVICE_ID	STRANI KLJUČ
UC_DEMUX_ID	STRANI KLJUČ
UC_EIT_PF_FLAG	STRANI KLJUČ
UC_EIT_SCHEDULE_FLAG	STRANI KLJUČ
STRANI KLJKUČ povezan sa EIT_EVENT tabelom	

Tabela 3.7 EIM_CONTENT_IDENTIFIER Tabela

U ovoj tabeli ključ nije auto inkrement kao i u prethodnim slučajevima. Razlog je što u ovom slučaju jedinstvena identifikacija elementa liste je *NB_IDENTIFIER*, a onda dalje po tom polju se vrši pretraga u EIM_CRID tabeli. Ovde nije bilo dovoljno samo uvezati tabele pomoću stranog ključa. Da bi element iz EIM_CRID tabele bio obrisan na brisanju elementa iz tabele EIM_CONTENT_IDENTIFIER tabele, njen strani ključ je uvezan sa primarnim ključem glavne tabele EIT_EVENT, a zatim strani ključ EIM_CRID tabele je uvezan sa primarnim ključem pomoćne tabele EIM_CONTENT_IDENTIFIER. Na ovaj način je postignuto da se brisanjem nekog elementa iz glavne EIM_CRID tabele kaskadno brišu element iz EIM_CONTENT_IDENTIFIER tabele koji sadrže isti strani ključ, a zatim i elementi iz EIM_CRID tabele koji sadrže strani ključ EIM_CONTENT_IDENTIFIER tabele. Drugi rečima, brisanjem elementa iz glavne tabele se briše cela hijerarhija elemenata pomoćnih tabela koji su uvezani sa njim pomoću stranih ključeva.

EIM_CRID Tabela	
AUTO_INKREMENT	PRIMARNI KLJUČ
NB_IDENTIFIER	STRANI KLJUČ
US_EVENT_ID	
US_EVENT_ID	
UI_TRANSPORT_STREAM_ID	
UI_ORIGINAL_NETWORK_ID	
UI_SERVICE_ID	
UC_DEMUX_ID	
UC_EIT_PF_FLAG	
UC_EIT_SCHEDULE_FLAG	
E_TYPE	
E_LOCATION	
UC_LENGTH	
UC_FIRST_BYTE	
STRANI KLJKUČ povezan sa EIM_CONTENT_IDENTIFIER tabelom	

Tabela 3.8 EIM_CRID

U ovoj pomoćnoj tabeli je iskorišćen isti mehanizam povezivanja tabela, samo što u ovom slučaju strani ključ predstavlja polje NB_IDENTIFIER a tabela sa kojom se ova tabela povezuje je pomoćna tabela EIM_CONTENT_IDENTIFIER, pa se tek preko nje povezuje sa glavnom tabelom EIM_EVENT.

4. Programsko rešenje

Programsko rešenje je realizovano korišćenjem Eclipse razvojnog okruženja u *Windows-u*, u C programskom jeziku, dok je prevođenje koda rađeno pomoću Linux virtuelne mašine.

Pošto se ovaj rad sastoji iz dva dela, u ovom poglavlju će biti posebno opisana integracija SQLite baze podataka u Comedia 2.0, a posebno prelazak čuvanja EIT podataka sa trenutnog vlasnički definisanog rešenja na SQL bazu podataka.

4.1 Integracija SQLite baze podataka u Comedia 2.0 programsko rešenje

Da bi se što lakše izvršila integracija i prilagođavanje SQLite baze podataka Comedia 2.0 programskom rešenju, korišćen je SQLite za namenske platforme. Ova verzija SQLite baze podataka omogućava veliku konfigurabilnost, ali pošto to nije bilo dovoljno neophodne su bile određene izmene poziva nekih sistemskih funkcija. Sve izmene rađene u SQLite bazi podataka su opcione. Dovoljno je samo u *make* datoteci obrisati *-DPOSIX_ECOS* opciju prevodioca, ponovo prevesti kod i da SQLite radi kao da nisu vršene nikakve izmene.

Adaptacioni sloj je realizovan u dva fajla: *helper_file.h* i *helper_file.c*. U njima se nalaze deklaracije i implementacije funkcija za rad sa memorijom, kao i funkcije koje su korišćene prilikom testiranja SQLite-a.

SQLite je napisan tako da su sve sistemske funkcije izdvojene u jedan niz (*aSyscall[]*) struktura (*unix_syscall*) sistemskih poziva. Struktura i niz sistemskih poziva su predstavljeni na sledeći način:

```
static struct unix_syscall {  
    const char *zName;           /* Naziv sistemskog poziva*/
```

```

sqlite3_syscall_ptr pCurrent;      /* Trenutna vrednost */
sqlite3_syscall_ptr pDefault;     /* Početna vrednost */
} aSyscall[] = {
#ifdef POSIX_ECOS
  { "open",      (sqlite3_syscall_ptr)ecos_fopen, 0 },
#else
  { "open",      (sqlite3_syscall_ptr)posixOpen, 0 },
#endif
#define osOpen   ((int*)(const char*,int,int))aSyscall[0].pCurrent)

#ifdef POSIX_ECOS
  { "close",     (sqlite3_syscall_ptr)ecos_close, 0 },
#else
  { "close",     (sqlite3_syscall_ptr)close, 0 },
#endif
#define osClose  ((int*)(int))aSyscall[1].pCurrent)
.
.
.
};

```

Svaki element niza *aSyscall[]* predstavlja po jedan sistemski poziv i za svaki sistemski poziv je definisan makro koji se posle svuda kroz SQLite koristi umesto direktnog korišćenja sistemskog poziva. Ovim je omogućena veoma laka izmena korišćenja sistemskih poziva.

Implementirane funkcije za rad sa datotekama iz adaptacionog sloja se oslanjaju na DMJFS. Funkcije koje se koriste u SQLite bazi podataka i njima odgovarajuće funkcije u DMJFS-u, nemaju identične ulazno-izlazne argumente. Takođe, kao i argumenti, razlikuju se i povratne vrednosti funkcija. Taj problem je rešen funkcijama iz gore pomenutog adaptacionog sloja. U funkcijama adaptacionog sloja se prihvataju vrednosti prosleđene iz SQLite-a u obliku sa kojim on radi i prosleđuju odgovarajućim funkcijama iz DMJFS-a u obliku koji on zahteva. Isto tako je urađeno i sa povratnim vrednostima. Prihvata se ono što stigne iz DMJFS-a i SQLite-u se proslede povratne vrednosti u obliku u kojem on njih zahteva.

Pored funkcija za rad sa datotekama i sa masovnom memorijom, mora da postoji i kontrola potrošene RAM(*Random Access Memory*) memorije. Comedia to obezbeđuje TKEL funkcijama. SQLite za rad sa memorijom koristi *malloc*, *free* i *realloc* definisane na sledeći način:

```

#define SQLITE_MALLOC(x)      malloc(x)
#define SQLITE_FREE(x)       free(x)
#define SQLITE_REALLOC(x,y)   realloc((x),(y))

```

Da bi se lako vratila stara funkcionalnost, i ovaj deo izmena je stavljen pod opciju prevodioca na sledeći način:

```

#ifdef POSIX_ECOS
  #define SQLITE_MALLOC(x)          ecos_malloc(x)
  #define SQLITE_FREE(x)           ecos_free(x)
  #define SQLITE_REALLOC(x,y)      ecos_realloc((x),(y))
#else
  #define SQLITE_MALLOC(x)          malloc(x)
  #define SQLITE_FREE(x)           free(x)
  #define SQLITE_REALLOC(x,y)      realloc((x),(y))
#endif

```

helper_file.c se sastoji iz sledećih funkcija:

- ❖ **void* ecos_malloc(size_t size);**
- ❖ **void ecos_free(void *buffer);**
- ❖ **void* ecos_realloc (void *ptr, size_t newsiz);**
- ❖ **int fchmod(int filedes, int mode);**
- ❖ **int fchown(int filedes, int owner, int group);**
- ❖ **int ftruncate(int fd, off_t length);**
- ❖ **int gettimeofday(struct timeval *tp, struct timezone *tzp);**
- ❖ **int utimes(const char *filename, struct timeval tvp[2]);**
- ❖ **oDMJFS_File get_real_dmjfs_descriptor(int index);**
- ❖ **int ecos_fstat(int, struct stat*);**
- ❖ **int ecos_fopen(const char *zFile, int flags, int mode);**
- ❖ **int ecos_lseek(int, off_t, int);**
- ❖ **ssize_t ecos_write(int filedes, const void *buffer, size_t size);**
- ❖ **ssize_t ecos_read(int filedes, void *buffer, size_t size);**
- ❖ **int ecos_close(int filedes);**
- ❖ **int my_fcntl(int index_descriptor, int command, ...);**
- ❖ **int my_stat(const char *filename, struct stat *buf);**
- ❖ **int ecos_delete(const char *filename);**
- ❖ **int ecos_fdatasync (int fildes);**
- ❖ **void printLineSwitch(int line, int switchCode);**
- ❖ **static int callback(void *NotUsed, int argc, char **argv, char **azColName);**
- ❖ **void sqliteTest();**
- ❖ **void sqliteTestMemory();**

Prve tri funkcije (*ecos_malloc*, *ecos_free* i *ecos_realloc*) su funkcije za rad sa RAM memorijom koje zamenjuju sistemske funkcije *malloc*, *free* i *realloc*. Njihova implementacija se oslanja na TKEL funkcije za rukovanje memorijom.

Funkcije *fchmod* i *fchown* su namenjene za dobijanje i menjanje prava pristupa datoteci. Pošto ove funkcionalnosti DMJFS ne omogućava, tela ovih funkcija se sastoje samo iz povratne vrednosti koju SQLite smatra kao validnu.

Funkcija *ftruncate* služi za menjanje veličine fajla. Pošto nju ne podržava DMJFS ni uopšte eCos koji se koristi na ploči, bila je neophodna realizacija ove funkcije oslanjanjem na TKEL i DMJFS.

Funkcije *gettimeofday* i *utimes* takođe ne postoje realizovane u eCos-u, pa je ona realizovana korišćenjem *datettime* paketa koji je već realizovan u komediji.

Fleš memorija na ploči koja je korišćena za izradu ovog rada radi sa 32-bitnim adresama koje na najvišoj poziciji imaju broj 1. Te adrese ako se prenose kao **int** vrednost tretiraju se kao negativne. Pošto eCos funkcije negativne adrese smatraju greškom, one moraju da se prenose u nekom drugom formatu, kako bi samim tim i SQLite te adrese smatrao ispravnim. Taj problem je rešen tako što je napravljen je jedan niz u kojem se čuvaju adrese, a SQLite-u se prosleđuju samo indeksi niza u kojem se čuvaju adrese. Kada SQLite pozove funkcije za rad sa fleš memorijom, on prosledi i indeks niza koji on tretira kao adresu. Funkcija *get_real_dmjfs_descriptor* kao argument prima indeks adrese iz niza adresa i vraća neoznačeni broj koji predstavlja stvarnu adresu. Tek ta adresa se prosleđuje DMJFS-u i onda se pozivaju DMJFS funkcije za rad sa datotekama. Na ovaj način je obezbeđeno da bez ikakvih izmena ovaj modul radi i na nekim drugim pločama ne brinući da li je bit na najvišoj poziciji 0 ili 1.

ecos_fstat je takođe funkcija za rad sa fajlovima namenjena za rad sa hard diskom. Njoj odgovarajuća funkcija iz DMJFS-a je *DMJFS_FileStat* na koju se u realizaciji *ecos_stat* i oslanja.

Funkcije *ecos_fopen*, *ecos_close*, *ecos_read*, *ecos_write*, *ecos_lseek* su funkcije koje se direktno oslanjaju na odgovarajuće funkcije iz DMJFS-a.

fcntl je funkcija koja se u SQLite-u koristi za dobijanje i podešavanje određenih flegova deskriptora. Pošto je u DMJFS-u na drugačiji način odrađeno čitanje, pisanje i pozicioniranje u datoteci, dovoljno je SQLite-u proslediti vrednost koju on smatra ispravnim (vrednost nula) i da on radi kako treba. Sa prelaskom na neku drugu platformu, gde je rad sa memorijom zamišljen na neki drugi način biće potrebno popunjavanje tela *ecos_fcntl* funkcije na mestima naznačenim u komentaru koda.

ecos_delete funkcija se koristi za brisanje fajlova. Njena realizacija se direktno oslanja na *DMJFS_FileDelete* funkciju.

ecos_fdatasync se u SQLite-u poziva pri upisu ili čitanju kada se radi sa više niti. Pošto je u DMJFS-u upis, čitanje i pozicioniranje u datoteci zaštićeno semaforima, ne može doći do

problema pri konkurentnom upisu ili čitanju u fajl pa se ova funkcija sastoji samo od povratne vrednosti nula, koju SQLite smatra kao validnu. Sa promenom ploče gde rad sa fajlovima nije na neki način zaštićen od konkurentnog pristupa, telo ove funkcije se mora popuniti tako da ona bude blokirajuća ako se nad fajlom vrši upis, i da omogućiti nastavak izvršavanja programa kada se upis u fajl završi.

printLineSwitch je funkcija koja je zgodna za korišćenje tokom otklanjanja grešaka SQLite-a za praćenje dokle je on stigao sa izvršavanjem.

callback funkcija je korišćena tokom testiranja funkcionalnosti SQLite-a. Nju poziva SQLite kao povratnu funkciju za ispisivanje rezultata *select* naredbe.

sqliteTest je funkcija u kojoj su definisani testni slučajevi pomoću kojih je testirana funkcionalnost, brzina i memorija koju troši SQLite tokom izvršavanja.

U funkciji *sqliteTestMemory* se nalazi ispis stanja memorije pre i posle poziva testne funkcije (*sqliteTest*).

4.2 Prelazak sa nestandardnog načina čuvanja EIT podataka na SQLite bazu podataka

Pošto su do sada podaci čuvani u dinamičkim listama, svaka od tih lista, kao jedan element liste sadrži EIT podatak, pokazivač na sledeći element i dodatne informacije koje služe drugim funkcijama EIM-a.

EIT podaci u listi se čuvaju u *tEIM_DVB_Definition* strukturi koja je predstavljena na sledeći način:

typedef struct

{

unsigned short usReferenceCount;

unsigned short usEventId;

 tEIM_TripletDVB stTripletDVB;

 tEIM_TimeInterval stTimeInterval;

struct tEIM_DVB_Definition_Flags

 {

unsigned ucVersionNumber :5; /**< Event version number, from the EIT table */

unsigned eRunningStatus :3; /**< Event running status */

unsigned bFreeCAMode :1; /**< Event scrambling, similar to the SDT flag */

unsigned eType :2; /**< Event is present, following or schedule. */

 } flags;

 tEIM_TableShortEvent stShortDescription;

 tEIM_TableExtendedEvent *pstExtendedDescription; /* optional descriptor */

 /* NULL if not defined */

unsigned char ucNbParentalRating; /* optional descriptor */

 /* 0 if not defined */

```

    tEIM_ParentalRating *pstParentalRating;
    unsigned char ucContent[2]; /* optional descriptor */
    /* 0 if not defined */
    /* for item 1 see T. 18 - EN 300 468 */
    /* for item 2 defined by broadcaster */
    tEIM_Component *pstComponent; /* optional descriptor */
    /* NULL if not defined */
#ifdef EIT_ALL_DESCRIPTOR
    tEIM_Linkage *pstLinkage; /* optional descriptor */
    /* NULL if not defined */
    tEIM_DataBroadcast *pstDataBroadcast; /* optional descriptor */
    /* NULL if not defined */
    tEIM_MultiComponent *pstMultiComponent; /* optional descriptor */
    /* NULL if not defined */
    tEIM_Telephone *pstTelephone; /* optional descriptor */
    /* NULL if not defined */
    tEIM_TimeShifted *pstTimeShifted; /* optional descriptor */
    /* NULL if not defined */
    tEIM_PrefNameIdentifier *pstPrefNameIdentifier; /* optional UK descriptor */
    /* NULL if not defined */
    tEIM_ContentIdentifier *pstContentIdentifier; /* optional descriptor */
    /* NULL if not defined */
    /*PPV product data*/
#endif CAK_MERLIN
    tEIM_ProductInfo productInfo;
#endif /*CAK_MERLIN*/
#ifdef PARSE_IMAGE_FROM_EIT
    tEIT_DVB_Icon * eventThumbnail;
#endif /*PARSE_IMAGE_FROM_EIT */
#endif
    EIM_STAT_INF( MEM)
    EIM_STAT_INF( STX) /* Short descriptor text lengths */
    EIM_STAT_INF( ETX) /* Extended descriptor text lengths */
} tEIM_DVB_Definition;

```

Dinamička lista u kojoj se čuvaju EIT podaci, a samim tim i prethodno navedena struktura za čuvanje EIT podataka, je dvostruko spregnuta lista gde poslednji element pokazuje na *NULL*. Element te liste definisan je na sledeći način:

```

/** An event, stored in the service cache.
 */
typedef struct _tEIT_DVB_InternalEvent
{
    tEIT_DVB_Definition *pst_EventDefinition;
    unsigned long ul_CompressedStartTime;
    unsigned long ul_CompressedStopTime;
    unsigned char uc_RelatedWindowsCount;
    BOOL ab_WinIdList[kEIT_DVB_MAX_WINDOW_COUNT];
    struct _tEIT_DVB_InternalEvent *pst_NextEvent;
    struct _tEIT_DVB_InternalEvent *pst_PrevEvent;

```

```
} tEIT_DVB_InternalEvent;
```

U cilju bržeg, stabilnijeg i modularnijeg pristupa podacima koje čuva ova struktura, urađena je integracija SQLite-a u EIM modul.

Da bi se SQLite bazi podataka moglo pristupiti iz različitih mesta EIM-a, bez stalnog otvaranja i zatvaranja baze, definisana je jedna statička lokalna promenljiva koja je pokazivač na bazu. Ovim je omogućeno da se baza otvara samo jednom na uključivanju uređaja i drži sve vreme otvorena dok uređaj radi. Zatvaranje baze se vrši tek na isključivanju uređaja.

Ključna mesta EIM-a za prelazak čuvanja podataka sa EIT statičkih lista na SQLite bazu podataka i mesto izmena prelaska čuvanja i čitanja EIT podataka predstavljaju sledeće funkcije:

- ❖ *EIT_DVB_DecodeEIT*
- ❖ *_SearchTriple*

Funkcija *EIT_DVB_DecodeEIT* se poziva na svakom pristiglom EIT događaju. Ona prihvata sve podatke iz toka podataka i iz njih izvlači EIT podatke. Čuvanje ovih podataka je preusmereno sa statičke liste na SQLite bazu podataka.

Funkcija koja na zahtev korisnika prosleđuje EIT podatke se zove *EIM_WindowEventGet*. Kao argument ove funkcije se prosleđuje lista EIT događaja, ta lista se u malo drugačijem obliku prosleđuje *EIT_DVB_GetEvents* funkciji koja kroz još nekoliko funkcija pomoću *_SearchTriple* funkcije čita sačuvane podatke iz SQLite baze i prosleđuje ih kao argument *EIM_WindowEventGet* funkciji.

Sve izmene prelaska čuvanja EIT podataka sa dinamičkih lista na SQLite bazu podataka su opcione i mogu se isključiti brisanjem opcije prevodioca *-DUSE_SQLITE_FOR_EIT* iz *build/make/generic.mac* datoteke.

Izvršavanje komandi u SQLite-u se radi pomoću funkcije *sqlite3_exec*. Ona prima sledeće argumente:

- ❖ Pokazivač na bazu
- ❖ SQL naredbu
- ❖ Funkciju povratnog poziva (*callback* funkciju)
- ❖ Prvi argument za funkciju povratnog poziva
- ❖ Argument preko kojeg se vraća greška ako dođe do nje tokom izvršavanja SQL naredbe

Funkcija povratnog poziva je od posebnog značaja zato što se preko nje dobijaju podaci koji su rezultat izvršavanja *SELECT* naredbe. U ovom radu čitanje podataka iz baze je urađeno pomoću više funkcija. Svaka od ovih funkcija je namenjena za čitanje podatka samo iz jedne tabele. U svakoj od ovih funkcija se poziva *sqlite3_exec* funkcija, gde se prosleđuje

funkcija povratnog poziva namenjena za parsiranje zadate tabele. Ovim je omogućeno da bude jedna funkcija za čitanje jedne od tabela ujedno i jedan raščlanjivač za tu tabelu, odnosno svaka funkcija povratnog poziva, parsira vrednosti iz njoj odgovarajuće tabele.

Pomoćne funkcije koje se koriste za upis podataka u SQLite bazu su sledeće:

- ❖ **int command_executioin_insert_or_update(sqlite3 *db, char *sql)**
- ❖ **static int callback_insert_or_update(void *NotUsed, int argc, char **argv, char **azColName)**

command_executioin_insert_or_update funkcija se koristi za upis novih podataka u tabele u bazi. Kao argument prihvata pokazivač na bazu i SQL naredbu. Povratna vrednost ove funkcije je rezultat da li je komanda prosleđena u dobrom formatu i da li je ona izvršena nad zadatom bazom. Za proveru da li je funkcija upisala podatke u baz podataka, ili ako dođe do greške pri izvršavanju, javlja se funkcija povratnog poziva *callback_insert_or_update*.

Funkcije pomoću kojih je vršeno čitanje podataka iz baze su sledeće:

- ❖ **command_select_EIT_EVENT(sqlite3 *db, char *sql)**
- ❖ **command_select_EIM_SHORT_EVENT(sqlite3 *db, char *sql)**
- ❖ **command_select_EIM_TABLE_EXTENDED_EVENT(sqlite3 *db, char *sql)**
- ❖ **command_select_EIM_PARENTAL_RATING(sqlite3 *db, char *sql)**
- ❖ **command_select_EIM_COMPONENT(sqlite3 *db, char *sql)**
- ❖ **command_select_EIM_MULTI_COMPONENT(sqlite3 *db, char *sql)**
- ❖ **command_select_EIM_CONTENT_IDENTIFIER(sqlite3 *db, char *sql)**
- ❖ **command_select_EIM_CRID(sqlite3 *db, char *sql)**

Pretraga glavne tabele (*EIT_EVENT*) se vrši pomoću zadatog tripleta, dok se ostale tabele pretražuju samo ako je uspešno prošla pretraga prve tabele. Kako bi se jedinstveno identifikovao događaj iz pomoćnih tabela dodatni uslov pretrage je identifikator tabele koji je dobavljen kao jedno obeležje glavne tabele. Funkcije u kojima se vrši parsiranje podataka kao rezultat *SELECT* su sledeće navedene funkcije povratnog poziva:

- ❖ **int callback_EIM_TABLE_EXTENDED_EVENT(void *NotUsed, int argc, char **argv, char **azColName)**
- ❖ **int callback_EIM_PARENTAL_RATING(void *NotUsed, int argc, char **argv, char **azColName)**
- ❖ **int callback_EIM_COMPONENT(void *NotUsed, int argc, char **argv, char **azColName)**

- ❖ **int callback_EIM_MULTI_COMPONENT**(void *NotUsed, int argc, char **argv, char **azColName)
- ❖ **int callback_EIM_CONTENT_IDENTIFIER**(void *NotUsed, int argc, char **argv, char **azColName)
- ❖ **int callback_EIM_CRID**(void *NotUsed, int argc, char **argv, char **azColName)

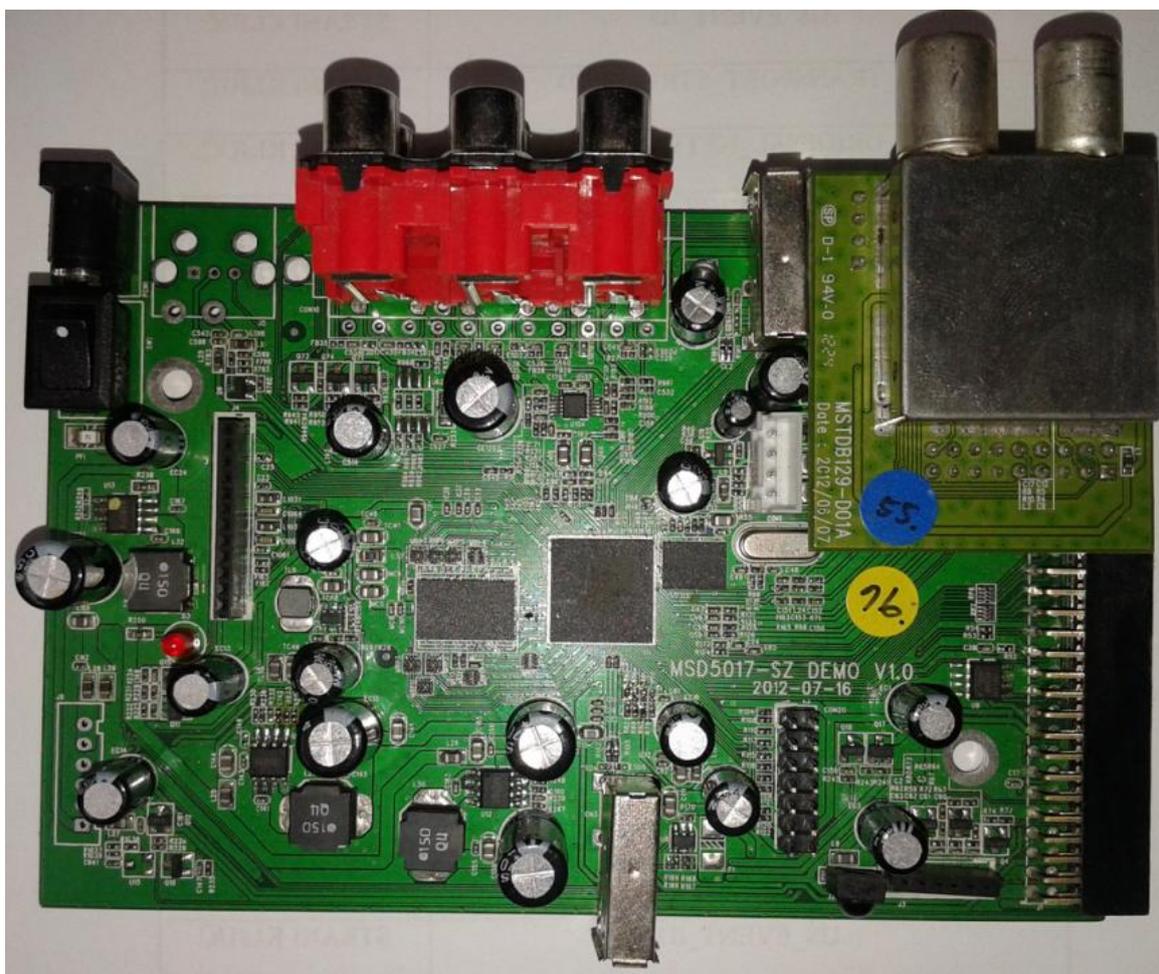
Svaka od navedenih funkcija ima proveru da li podatak već postoji u događaju predstavljenom strukturom *tEIM_DVB_Definition*. Ako ne postoji u funkciji *callback_EIM_TABLE_EXTENDED_EVENT* će se kreirati novi događaj, dok će u ostalim funkcijama biti vraćena greška. U tim funkcijama su takođe rađene provere postojanja događaja, samo što one ne kreiraju novi događaj ako on ne postoji već samo vraćaju kodove greške. U slučaju kada događaj postoji ove funkcije će samo rezultat dobijen iz baze upisati na njima odgovarajuće mesto u strukturi *tEIM_DVB_Definitio*.

5. Rezultati

Testiranje je vršeno na MStar platformi. MStar Semiconductor je kompanija specijalizovana za mešovite tehnologije integriranih kola. MStar pruža punu liniju programskih osnova, koja obuhvata kompletno MStar rešenje, kao i MStar SDK da se ubrza vreme do tržišta za STB uređaje. MStar pruža punu seriju STB proizvoda koji uključuju:

- ❖ Osnovni prebacivač kanala STB
- ❖ Interaktivni prebacivač kanala STB
- ❖ Hibridni IPTV prebacivač kanala STB
- ❖ Napredan kablovski STB
- ❖ Početno automatizovan STB
- ❖ Bežični STB

Platforma koja je korišćena tokom realizacije ovog rada predstavlja napredni kablovski STB. Arhitektura na kojoj on radi je MIPS. Ovaj STB je veoma skromnih kako resursa tako i hardverskih mogućnosti, pa koristi eCos operativni sistem. To je besplatan operativni sistem namenjen za rad sa ugrađenim aplikacijama koje, kao i operativni sistem, rade u realnom vremenu. Visoko konfigurabilna priroda eCos-a omogućava da se operativni sistem prilagodi veoma preciznim zahtevima aplikacije, pružajući najbolje moguće performanse na prilagođenom i optimizovanom hardveru. Realizovan je u C/C++ programskom jeziku i ima kompatibilne slojeve i programske sprege sa POSIX i μ Tron. Ova platforma ne podržava čak ni mrežni priključak, radi sa 32MB radne memorije pa je i operativni sistem podešen da radi sa veoma malo resursa. Na sledećoj slici je prikazano kako ona izgleda:



Slika 5.1 MStar platforma na kojoj je testiran rad

Da bi se mogli hvatati ispisi korišćena je još jedna takođe MStar platforma. Ona je povezana preko serijskog priključka a napajanje koristi sa STB uređaja. Na sledećoj slici je prikazano kako ona izgleda:



Slika 5.2 Platforma koja je korišćena za ispise

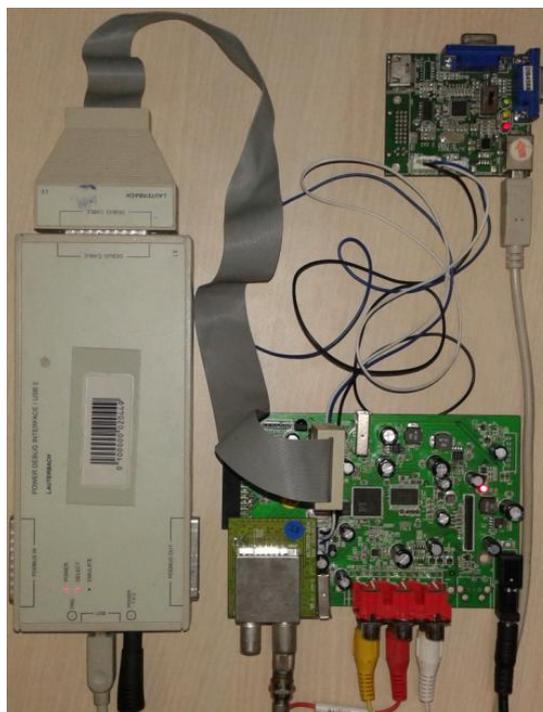
Ispisi koji su dobijani pomoću platforme sa prethodne slike su se ispisivali pomoću programa *Tera Term*.

Uređaj koji je korišćen za otklanjanje grešaka je *LAUTERBACH*. On je sa STB platformom povezan preko kabla za otklanjanje grešaka. Alat kojim se on pokreće je *TRACE32*. Na sledećoj slici je prikazano kako ovaj uređaj izgleda:



Slika 5.3 LAUTERBACH uređaj za otklanjanje grešaka

Na sledećoj slici je prikazano kako ovo sve izgleda kada je povezano, kako je povezana platforma sa uređajem za otklanjanje grešaka i kako se platforma za ispis napaja sa STB platforme:



Slika 5.4 MStar STB sa MStar platformom za hvatanje ispisa i LAUTERBACH uređajem za otklanjanje grešaka u toku rada

Rezultati ovog rada najbolje se ogledaju u analiziranju utrošene memorije. Testiranje je podeljeno u dva testna slučaja:

- ❖ Integracija SQLite baze podataka u Comedia programskom rešenju
- ❖ Prelazak čuvanja EIT podataka sa statičkih lista na SQLite bazu podataka

5.1 Testiranje integracije SQLite baze podataka u Comedia programsko rešenje

Utrošena memorija samo pri uključivanju ovog modula može se dobiti analiziranjem *comedia.map* fajla. Sekcije koje nas interesuju su:

- ❖ `.text` - izvršni kod u izvršnoj datoteci
- ❖ `.rodata` - konstantni podaci (*read only* promenljive)
- ❖ `.data` - inicijalizovane globalne promenljive
- ❖ `.bss` - ne inicijalizovane globalne promenljive

Ukupna utrošena memorija za uključivanje ovog modula u *Comedia 2.0* iznosi 795kB. Detaljan prikaz koliko se troši na koju sekciju se nalazi u sledećoj tabeli:

Sekcije	<code>.text</code>	<code>.rodata</code>	<code>.data</code>	<code>.bss</code>	Ukupna utrošena memorija
Memorija	760648B	45904B	3144B	5160B	795kB

Tabela 5.1 Utrošena memorija prilikom uključivanja *sqlite* modula u *Comedia 2.0*

Sledeće šta je testirano je koliko ovaj modul troši RAM memorije tokom izvršavanja. Testiranje je vršeno korišćenjem *TKEL_bstats* funkcije. Ova funkcija kao rezultat vraća ukupnu utrošenu RAM memoriju, što u testnim slučajevima nije baš najmerodavnije jer na rezultat merenja utiče kompletno izvršavanje komedije a ne samo ovaj modul. Sva merenja su predstavljena kao zauzetost u odnosu na početno stanje pre pokretanja testa. Testna funkcija pomoću koje je vršeno merenje zauzetosti memorije tokom izvršavanja sadrži merenja posle sledećih testnih slučajeva:

- ❖ Otvaranje baze podataka sa kreiranjem datoteke u fleš memoriji koji se koristi kao baza podataka
- ❖ Kreiranje tabele
- ❖ Upis elementa u tabelu
- ❖ Ažuriranje elementa u tabeli
- ❖ Brisanje elementa iz tabele
- ❖ Zatvaranje baze

Posle svakog testnog slučaja je ispisivan sadržaj kreirane tabele. Test je ponavljen 100 puta, a maksimalne vrednosti zauzete memorije su prikazane sledećom tabelom:

funkcija	Otvaranje	Kreiranje	Upis	Ažuriranje	Brisanje	Zatvaranje
Memorija (u Bajtima)	67240	83512	83564	83664	83656	260

Tabela 5.2 Zauzetost memorije posle testnih funkcija

Iz prethodne tabele se vidi da su ažuriranje i brisanje elemenata iz tabele operacije koje su najzahtevnije pa ovo treba imati u vidu tokom korišćenja baze podataka jer na platformama sa malo resursa kada se radi sa velikom količinom podataka ove funkcije mogu da potroše puno memorije.

Da bi se utvrdilo da li je sva memorija zauzeta tokom izvršavanja testne funkcije na kraju i oslobođena merena je zauzeta memorija pre i posle njenog pokretanja. Ta testna funkcija sadrži sve testne slučajeve iz prethodne tabele. Pošto je jedini način za merenje zauzete memorije bio pomoću *TKEL_bstats* funkcije, i u ovom slučaju na testiranje utiče i kompletno izvršavanje komedije. Zbog toga je test ponavljan više puta i utvrđeno je da se tokom sve većeg broja izvršavanja ukupna zauzeta i oslobođena memorija približava nuli, na osnovu čega se može reći da u ovom modulu ne dolazi do curenja memorije.

5.2 Testiranje prelaska čuvanja EIT podataka sa statičkih lista na SQLite bazu podataka

U ovom delu rada testirani su sledeći slučajevi:

- ❖ Provera da li su svi podaci prethodno čuvani u dinamičkoj listi sačuvani i u SQLite bazu podataka
- ❖ Zauzetost radne memorije samo za integraciju SQLite baze podataka u EIM modul i poređenje zauzete radne memorije na pozivu funkcija za čitanje podataka iz dinamičke liste i iz SQLite baze podataka

5.2.1 Provera da li su svi podaci prethodno čuvani u dinamičkoj listi sačuvani i u SQLite bazu podataka

Podaci upisani u bazu su prvobitno bili duplirani sa podacima iz liste. Upoređivanjem tih podataka je utvrđeno da se svi podaci koji se nalaze u listi, nalaze i u bazi, za šta se može reći da upis u SQLite bazu podataka radi ispravno.

5.2.2 Zauzetost radne memorije samo za integraciju SQLite baze podataka u EIM modul i poređenje zauzete radne memorije na pozivu funkcija za čitanje podataka iz dinamičke liste i iz SQLite baze podataka

Pošto se upis i pretraga EIT događaja u bazi vrši na svakom dobijenom EIT događaju, posle nekog vremena u bazi postaje velika količina podataka što dovodi do sve sporije i zahtevnije pretrage. Ovim dolazi i do prepunjenosti radne memorije što na kraju prouzrokuje blokiranje platforme na kojoj se radi.

Prikaz zauzete i realocirane radne memorije nakon svakog petog poziva funkcije za dobijanje liste EIT podataka za prvih 100 poziva prikazan je sledećom tabelom:

Redni broj poziva	Zauzeta memorija korišćenjem dinamičke liste (u bajtima)	Zauzeta memorija korišćenjem SQLite baze (u bajtima)	Realocirana memorija sa korišćenjem dinamičke liste (u bajtima)	Realocirana memorija sa korišćenjem SQLite baze (u bajtima)
1.	3225752	3419984	3706	5431
5.	3226016	3420212	3780	5504
10.	3277948	3421600	4276	5596
15.	3283668	3427428	4276	5984
20.	3283708	3422740	4348	6090
25.	3284364	3422812	4348	6250
30.	3285020	3423632	4348	6320
35.	3285020	3423980	4348	6390
40.	3284896	3423980	4352	6466
45.	3287524	3423980	4352	6536
50.	3290804	3424440	4352	6606
55.	3290880	3424440	4352	6676
60.	3290864	3424440	4360	6746
65.	3290864	3424440	4360	6844
70.	3290864	3425096	4360	6984
75.	3293488	3427824	4360	7110
80.	3296768	3430456	4360	7439

85.	3296768	3430456	4360	7579
90.	3296664	3439176	4366	7719
95.	3297096	3439176	4366	8913
100.	3297096	3439176	4366	9249

Tabela 5.3 Prikaz stanja memorije nakon pozivanja funkcije za dobijanje liste EIT događaja

Iz prethodne tabele se vidi da je zauzeta memorija na pozivu funkcije za dobijanje liste EIT događaja znatno veća sa njihovim čuvanjem u SQLite bazi podataka nego sa čuvanjem u dinamičkoj listi. Iz ovoga se može zaključiti da sama integracija SQLite baze u EIM modul zahteva više radne memorije.

Takođe iz prethodne tabele se vidi i da je zauzeta radna memorija posle prvog i stotog zahteva za listu EIT događaja manja sa korišćenjem SQLite baze. Razlika u zauzetoj radnoj memoriji sa korišćenjem dinamičkih lista je $3297096 - 3225752 = 71344$ bajta, dok je sa korišćenjem baze zauzeta memorija $3439176 - 3419984 = 19192$ bajta. Sa povećavanjem baze, i mnogo većim brojem EIT događaja u njoj, ova razlika postaje sve bliža razlici dobijenoj korišćenjem dinamičke liste. I ako je tokom rada razlika u zauzetoj radnoj memoriji manja sa korišćenjem SQLite baze, kao što je već rečeno baza ipak zahteva više memorije za samo njeno pokretanje, a pored toga i sa povećavanjem broja EIT događaja pretraga postaje sve zahtevnija. Pored lošijeg odziva baze i sporijeg dobijanja podataka, to se može videti i iz realocirane memorije. Sa povećanjem broja EIT događaja u bazi podataka pretraga zahteva više radne memorije, što posle nekog vremena prouzrokuje blokiranje uređaja.

6. Zaključak

U prvom delu rada opisana je integracija SQLite baze podataka u okruženje programske podrške za digitalnu televiziju u Comedia 2.0 programsko rešenje. Integracija ove baze podataka rađena je na MStar platformi koja radi na eCos operativnom sistemu. Fokus u ovom delu rada je bio prilagođavanje SQLite baze podataka sistemu datoteka koji koristi ova platforma. Integracija je postignuta dodavanjem jednog novog adaptacionog sloja između SQLite baze podataka i Comedia jezgra. Najveći deo ovog adaptacionog sloja čine funkcije za prilagođavanje SQLite baze podataka sistemu datoteka koji koristi ova razvojna platforma.

SQLite baza podataka je integrisana u Comedia programsko rešenje tako da se bez ikakvih izmena može koristiti i u ostalim modulima, ne samo u EIM modulu što je bio jedan od ciljeva ovog rada. Zbog napravljenog adaptacionog sloja olakšan je prelazak na drugi sistem datoteka. Sa prelaskom na neki drugi sistem datoteka dovoljne su samo eventualne izmene tela funkcija adaptacionog sloja koje se koriste za rad sa datotekama.

Drugi deo ovog rada obuhvata prelazak čuvanja EIT podataka sa čuvanja u dinamičku listu na čuvanje u SQLite bazu podataka. U ovom delu je urađeno projektovanje baze podataka prema EIT podacima dostupnim po DVB standardu, kao i integracija SQLite baze u EIM modul.

Načinom na koji funkcioniše upis u bazu i čitanje podataka iz nje, pristup zahtevanim podacima je znatno jednostavniji, ali fleš memoriji se jako često pristupa i u nju upisuje. To pored smanjenja njenog životnog veka znatno povećava vreme odziva ovog modula. Posle nekog vremena, sa sve većom količinom podataka u bazi, pretraga postaje sve zahtevnija pa zbog nedostatka radne memorije dolazi se u situaciju da platforma nema više radne memorije što dovodi do blokiranja platforme.

Ubrzanje i rad ovog modula je moguće postići sa više radne memorije. Na taj način bi se rešio problem pretrage SQLite baze u kojoj se nalazi puno podataka i ne bi dolazilo do blokiranja platforme. Sa povećanjem radne memorije dodatno ubrzanje bi se postiglo i dodavanjem novog modula u Comedia 2.0 programsko rešenje koje će da funkcioniše slično DMJFS modulu koji će omogućiti da se EIT podaci čuvaju u radnu memoriju a ne u fleš memoriju, jer je radna memorija brža od fleš memorije.

7. Literatura

- [1]SQLite: <http://www.sqlite.org/amalgamation.html>, učitano 25.08.2014.
- [2] eCos: <http://ecos.sourceware.org/>, učitano 25.08.2014.
- [3] MStar: <http://www.mstarsemi.com/>, učitano 25.08.2014.
- [4] Comedia:<http://www.iwedia.com/en/software-components/comedia-tv-and-stb-middleware>, učitano 25.08.2014.
- [5]Vijayan Prabhakaran, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, "Analysis and Evolution of Journaling File Systems", Computer Sciences Department, University of Wisconsin, Madiso, dostupno na: https://www.usenix.org/legacy/events/usenix05/tech/general/full_papers/prabhakaran/prabhakaran.pdf
- [6]W. Fischer, "Digital Video and Audio Broadcasting Technology" (2010), A Practical Engineering Guide, Third Edition
- [7]Milan Bjelica, Literatura sa predavanja iz predmeta *PROGRAMSKA PODRŠKA U TELEVIZIJI I OBRADI SLIKE 1*, 2014.
- [8]Milan Bjelica, Literatura sa predavanja iz predmeta *PROJEKTOVANJE NAMENSKIH RAČUNARSKIH STRUKTURA 2*, 2014.
- [9]Maestro: http://www.i-maestro.org/contenuti/contenuto.php?contenuto_id=50, učitano 11.09.2014.