УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Јоаким Јањатовић

Израда виртуелизованог окружења за извршавање апликација у возилу са приказом на више екрана

МАСТЕР РАД

Нови Сад, 2017

	УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА МАСТЕР РАД	Датум:

(Податке уноси предметни наставник - ментор)

СТУДИЈСКИ ПРОГРАМ:	Рачунарство и аутоматика
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	Проф. Др Мирослав Поповић

Студент:	Јоаким Јањатовић	Број индекса:	Е2 39/2016
Област:	Рачунарске мреже, магистрале и протоколи у аутомобилу		
Ментор:	Доц. др Милан Бјелица		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ – МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ МАСТЕР РАДА:

Израда виртуелизованог окружења за извршавање апликација у возилу са приказом на више екрана

ТЕКСТ ЗАДАТКА:

Израдити виртуелизовано окружење за извршавање инфозабавних апликација различите безбедносне критичности, унутар возила. Апликацијама је потребно омогућити приказ на више екрана, на циљној платформи са једним систем на чипу.

Руководилац студијског програма:	Ментор рада:

Примерак за: О - Студента; О - 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**:		Дипломски – мастер рад	
Аутор, **АУ**:		**Јоаким Јањатовић**	
Ментор, **МН**:		**Доц. др Милан Бјелица**	
Наслов рада, **НР**:		**Израда виртуелизованог окружења за извршавање апликација у возилу са приказом на више екрана**	
Језик публикације, **ЈП**:		Српски	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		**2017**	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		**7/53/16/12/27/0/0**	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кqучне речи, **ПО**:		**Окружење за апликације у возилу, Инфозабавни систем, Више екрана, Виртуелизација, Хипервизор, QNX, AGL, ASIL B**	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		**Овај рад представља виртуелизовано окружење за извршавање апликација са приказом на више екрана унутар возила. Безбедносно критични и безбедносно некритични софтвер се извршавају изоловано у различитим оперативним системима на истом систему на чипу. Истраживање показује да QNX са хипервизорским проширењем и Линуксом у виртуелној машини уз средњи слој софтвера за комуникацију представља погодно окружење за извршавање графичких апликација у возилима.**	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	**Доц. др Богдан Павковић**	
	Члан:	**Доц. др Јелена Митровић-Симић**	Потпис ментора
	Члан, ментор:	**Доц. др Милан Бјелица**	

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**:		Master Thesis	
Author, **AU**:		**Joakim Janjatovic**	
Mentor, **MN**:		**Milan Bjelica, Ph.D**	
Title, **TI**:		**Virtualization environment for in-vehicle applications with multiple displays**	
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**:		**2017**	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		**7/53/16/12/27/0/0**	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		**Environment for in-vehicle applications, In-vehicle Infotainment, Multiple displays, Virtualization, Hypervisor, QNX, AGL, ASIL B**	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		**This paper presents virtualization environment for in-vehicle applications with multiple displays. Safety critical and non safety critical software run isolated inside different operating systems on the same SoC. The research shows that QNX with hypervisor extension and Linux inside a virtual machine with communication middleware is suitable environment for graphical in-vehicle applications.**	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	**Bogdan Pavković, Ph.D**	
	Member:	**Jelena Mitrović-Simić, Ph.D**	Menthor's sign
	Member, Mentor:	**Milan Bjelica, Ph.D**	

Захвалност

Захваљујем се доц. др Милану Бјелици и Тихомиру Анђелићу на стручним саветима за израду овог рада.

Посебно се захваљујем својим родитељима Снежани и Александру на пруженој подршци у току читавог школовања.

На крају захваљујем се свима онима који су на било који начин допринели изради мастер рада.

САДРЖАЈ

Списак слика

Списак табела

Скраћенице

ABS - систем против блокаде кочница (енг. ABS - Anti-lock Braking System)

ADAS - напредни системи за помоћ возачу (енг. ADAS - Advanced Driving Assistance Systems)

ASIL - ниво интегритета аутомобилске безбедности (енг. ASIL - Automotive Safety Integrity Level)

ESP - електронска контрола стабилности (енг. ESP - Electronic Stability Program)

IVI - инфозабавни системи у аутомобилима (енг. IVI – In-Vehicle Infotainment)

ISO - међународна организација за стандардизацију (енг. ISO - International Organization for Standardization)

IPC - комуникација између процеса (енг. IPC - Inter-process Communication)

IVC - комуникација између процеса у различитим оперативним системима односно виртуелним машинама (енг. IVC - Inter-virtual-machine communication)

1. Увод

Досадашњи начин развоја рачунарских система за аутомобилску индустрију производи веома сложене системе са пуно малих зависних делова. За развој, а касније и одржавање таквих система потребно је пуно знања, времена и средстава. Једна од главних тема у аутомобилској индустрији је проширење рачунарских система у аутомобилима уз смањење сложености.

Модерни инфозабавни системи у аутомобилима (енг. IVI – In-Vehicle Infotainment) најчешће поседују више различитих вишејезгарних процесора на једној или више умрежених платформи. Један од највећих изазова је направити визуелно напредан систем са богатим садржајима који је у исто време довољно јефтин за прилагођавање различитим моделима аутомобила и масовну производњу. Тренутно аутомобилска индустрија тежи уочавању компоненти заједничких за све системе и њиховом поједностављењу као и вишекратној употреби. Преостале компоненте специфичне за сваки модел аутомобила пожељно је развијати на што већем нивоу апстракције.

Задатак рада је да се изради виртуелизовано окружење за извршавање инфозабавних апликација различите безбедносне критичности, унутар возила. Апликацијама је потребно омогућити приказ на више екрана, на циљној платформи са једним систем на чипу (енг. SOC - System on chip). У оквиру задатка биће одабрана виртуелизациона архитектура оперативних система и израђен софтвер за дељење ресурса и комуникацију између апликација, написан програмским језиком Це++ (енг. C++).

Овај рад је сачињен од седам поглавља.

Друго поглавље описује шта су инфозабавни системи у аутомобилима, које су најчешће примене, изазови при пројектовању и даје њихову архитектуру.

У трећем поглављу дат је опис архитектуре целог система, средњег слоја софтвера и комуникације са графичким апликацијама.

Четврто поглавље садржи опис израђених програмских модула

Пето поглавље описује начин испитивања софтвера.

Шесто поглавље садржи кратак опис шта је урађено у овом раду и који су правци даљег развоја.

2. Теоријске основе

Рачунарске системе у аутомобилима у безбедносном смислу можемо поделити на критичне и некритичне. Системи критични по безбедност као што су ABS (енг. ABS - Anti-lock Braking System), ESP (енг. ESP - Electronic Stability Program), Електро-серво волан у савременим возилима чине основу за Напредне системе за помоћ возачу ADAS (енг. ADAS - Advanced Driving Assistance Systems) попут аутоматског кочења пре судара, одржавања брзине, одстојања и траке на путу, аутоматског паркирања итд. Овакви системи оптимизују се за хардверску платформу на којој се извршавају како би се што боље искористили расположиви ресурси [1]. Информације о тренутном стању возила као што су брзина кретања, број обртаја мотора и сличне такође спадају у безбедносно критичне. Међу некритичне системе спадају инфозабавни, комуникацијски и други системи чијим отказом у току вожње безбедност људи не може бити угрожена.

Уочљива је тежња аутомобилске индустрије ка развоју самосталних возила без возача. Системи комуникације између возила и инфраструктуре са циљем повећања ефикасности и уштеде енергије развијају се упоредо са телекомуникационим мрежама пете генерације. У таквим возилима путницима је потребно пренети информације о вожњи и омогућити им да на безбедан начин прекрате време.

2.1 Проблеми система у возилима

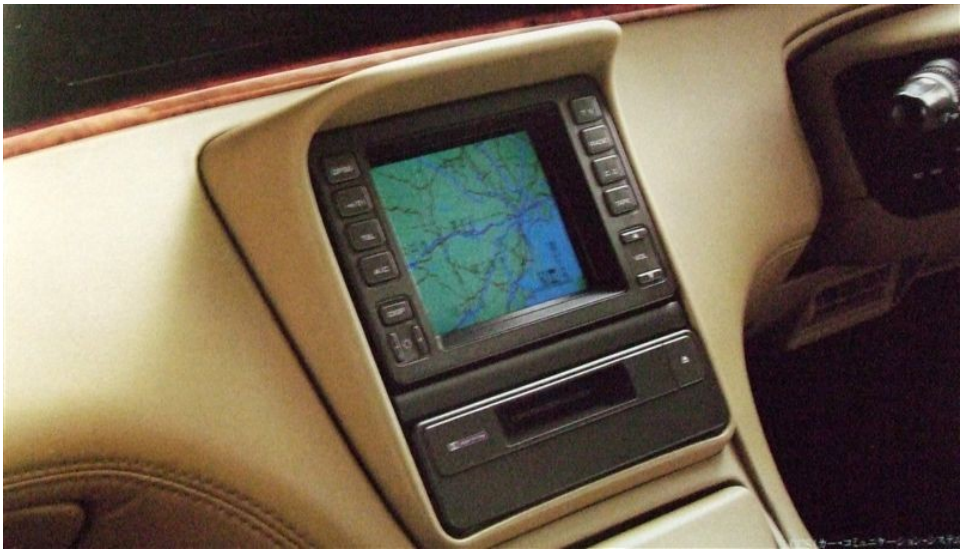
Развој нових технологија захтева сагледавање проблема из више углова. Из угла продаје, аутомобил треба да поседује што више привлачних могућности на које су купци већ навикнути у другим индустријама попут потрошачке електронике. Главне разлике уређаја потрошачке електронике попут мобилних телефона и сет-топ боксова у односу на аутомобиле су безбедност и цена. Ако се мобилни телефон угаси и поново покрене сам од себе корисник ће у најгорем случају изгубити неколико минута пре него што настави да га

користи. Кашњење од неколико секунди, док се пребацују телевизијски канали, такође је прихватљиво. Са друге стране возачу у аутомобилу треба омогућити све ове додатне садржаје без ометања критичних система и угрожавања безбедности људи.

2.1.1 Потреба за додатним садржајем

Аутомобили су у механичком смислу достигли врх лествице и ту је смањен простор за значајан напредак који се углавном своди на повећање ефикасности постојећих система [2]. Приметне су тежње смањењу запремине мотора, повећању ефикасности додавањем турбо пуњача, и смањењу расипања енергије у системима као што су расхладни, издувни и разни електрични системи. Са друге стране хибридна и електрична возила, иако још увек заузимају мали део тржишта, успешно су задобила поверење купаца те се очекује развој технологија у том правцу.

У погледу информација и додатних садржаја доступних возачу аутомобили су значајно напредовали. Произвођачи аутомобила возачима поред слушања музике у току вожње нудили су и сателитску навигацију са скромним приказом информација на малим екранима. Први аутомобил са сателитском навигацијом на екрану са катодном цеви у боји осетљивим на додир била је Мазда Еунос Космо из 1990-те године (Слика 2.1) [3]. Овакви системи били су прескупи за масовну производњу а тек популарност ДВД уређаја за пуштање филмова и концерата код куће створила је потребу и прилику за увођења истог доживљаја и у аутомобиле.



Слика 2.1 Први аутомобил са сателитском навигацијом

Промена трендова у потрошачкој електроници пресликава се и на аутомобилску индустрију па је тако могућност преузимања садржаја са интернета постала још једна од ставки на списку додатне опреме приликом куповине аутомобила. Данашњи инфозабавни системи нуде могућност приказа разних садржаја па чак и апликација са мобилних

телефона. Примери таквих сервиса су Епл Карплеј (енг. Apple CarPlay) и Андроид Ауто (енг. Android Auto) (Слика 2.2).

Слика 2.2 Апликације са мобилних телефона на инфозабавном систему

Инфозабавни системи често на сликовит начин приказују стање како спољашњости тако и унутрашњости аутомобила, а могућности има толико да човек може данима да га подешава и опет да не испроба све. На Слици 2.3 приказана су два савремена инфозабавна система који се уграђују у Мерцедес и Ауди.

Слика 2.3 Савремени инфозабавни системи

У складу са класом и на основу планиране цене аутомобила одређују се и расположива средства за развој. Тако аутомобили више класе по правилу имају више додатне опреме и моћнији инфозабавни систем. Добар пример за то су Тесла Модел С и Модел три приказана на Слици 2.4, где је у јефтинијем моделу постављен само један централни екран док скупљи поседује још један испред возача.

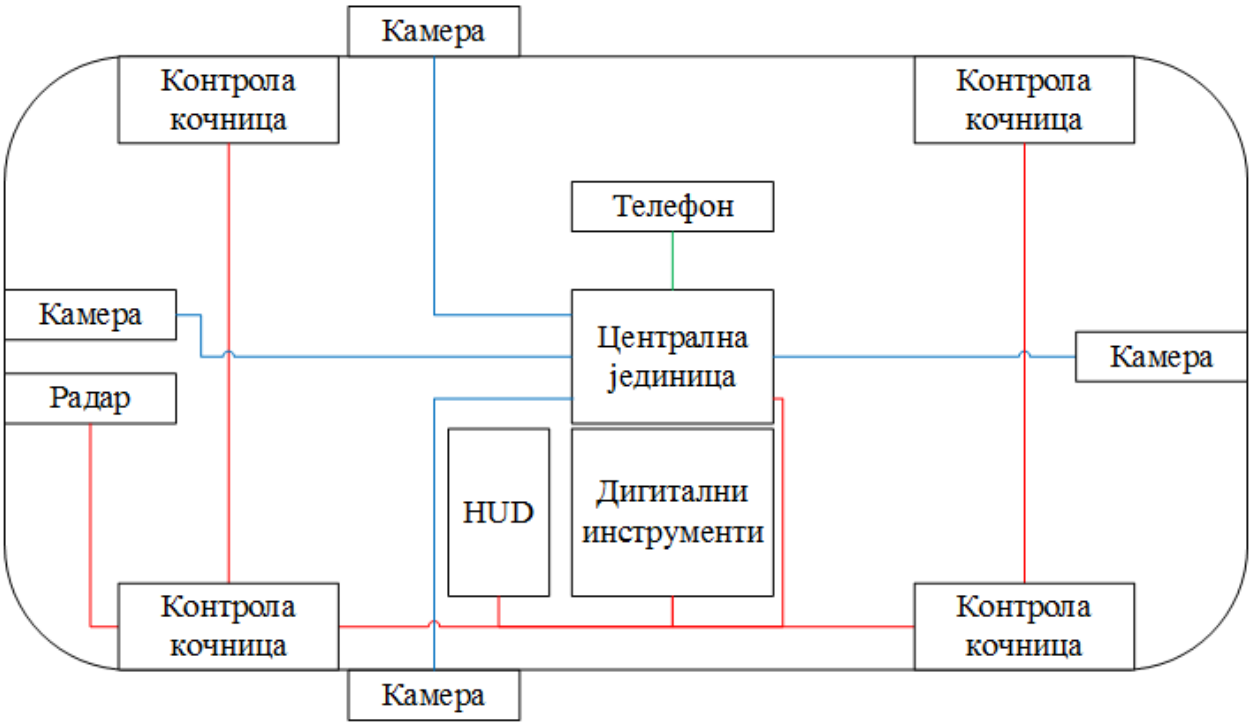
Слика 2.4 Инфозабавни системи у аутомобилима различите класе

Предвиђа се значајан напредак у аутомобилској индустрији и то на два поља [4]. Концепт умрежених аутомобила (енг. Connected Car) условљен је развојем телекомуникационх мрежа, док ће самовозећи аутомобили бити могући захваљујући напретку софтвера (енг. Software-Defined Car). Крајњи исход биће могућност масовног коришћења аутомобила као услуге превоза (енг. Car-as-a-Service).

2.1.2 Пројектовање софтвера у аутомобилској индустрији

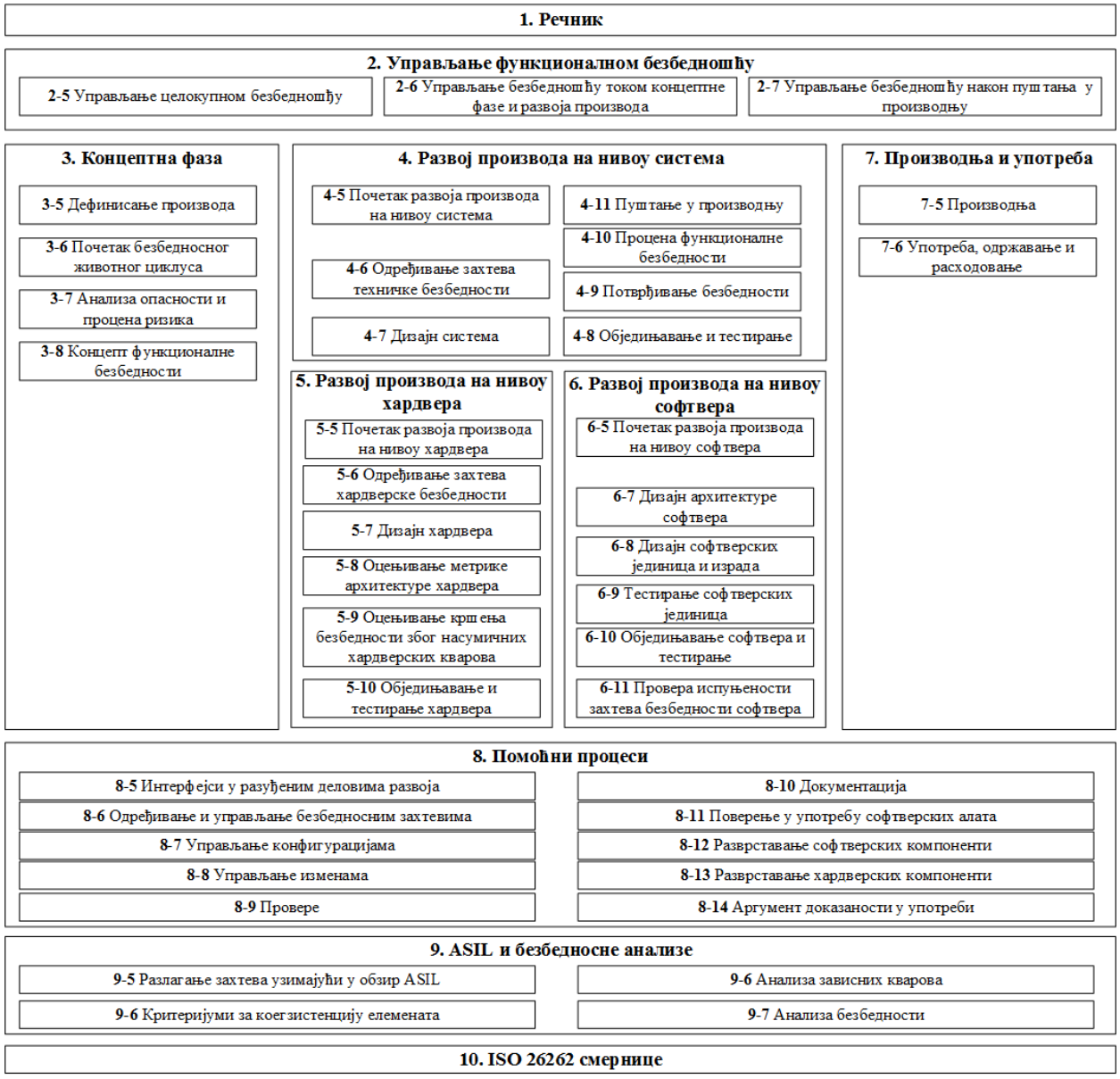
Различити произвођачи аутомобила развили су сопствене начине организације софтверских система [5]. Међутим, сви то раде на сличан начин. Електрични и софтверски системи подељени су у домене, као што су инфозабавни и погонски (енг. powertrain). Неки домени су безбедносно критични, неки нису, неки су кориснички оријентисани а неки су уграђени (енг. embedded) тако да корисник нема директног додира са њима. Домени су подељени на системе који у комбинацији остварују неку функционалност као што су упозорење на напуштање саобраћане траке, или сателитско навођење.

Системи се састоје од компоненти које остварују део функционалности, нпр. прослеђивање порука унутар инфозабавног система. Компоненте су организоване у софтверске модуле који су сачињени од датотека софтверског кода са класама, методама и функцијама. Системи из истог домена могу бити разуђени по аутомобилу тако да их је у циљу заједничког функционисања неопходно умрежити неком од аутомобилских магистрала (Слика 2.5). Инфозабавни систем или централна јединица (енг. Head unit), дигитални инструменти на табли (енг. Instrument cluster) и екран за смањено ометање возача (енг. HUD - Heads up display) најчешће су одвојени хардверски системи повезани на CAN (енг. CAN - Controller Area Network) магистралу преко које комуницирају међусобно и са осталим системима у аутомобилу. Безбедносно критичне магистрале на слици су представљене црвеном, некритичне зеленом а директне видео везе са камерама плавом бојом.



Слика 2.5 Умрежени системи у аутомобилу

Напредни системи за помоћ возачу попут аутоматског кочења, одржавања брзине и одстојања, приказа брзине и основних информација о возилу, критични су за безбедност и њиховим отказом угрожавају се животи људи. Један од стандарда аутомобилске индустрије је ISO 26262 (енг. ISO 26262) о функционалној безбедности друмских возила. Функционална безбедност је одсуство неразумног ризика од опасности настале услед неисправности електричних односно електронских система. То значи да не сме доћи до повреде људи услед квара на електроници. Овај стандард прописује процедуре које је потребно спровести од планирања развоја преко испоруке безбедног аутомобила све до краја његовог века трајања. Првенствено се односи на електричне и електронске системе али и на аутомобил у целини. На Слици 2.6 дат је преглед процеса ISO 26262 стандарда. Састоји се из десет делова: 1. Речник, 2. Управљање функционалном безбедношћу, 3. Концептна фаза, 4. Развој производа на нивоу система, 5. Развој производа на нивоу хардвера, 6. Развој производа на нивоу софтвера, 7. Производња и употреба, 8. Помоћни процеси, 9. ASIL (енг. ASIL - Automotive Safety Integrity Level) и безбедносне анализе, и 10. ISO 26262 смернице. Процеси трећег, четвртог, петог, шестог и седмог дела спадају у безбедносни животни циклус који прати развој производа од препознавања ризика до потврде да су безбедносни захтеви испоштовани у испорученом производу.



Слика 2.6 ISO 26262

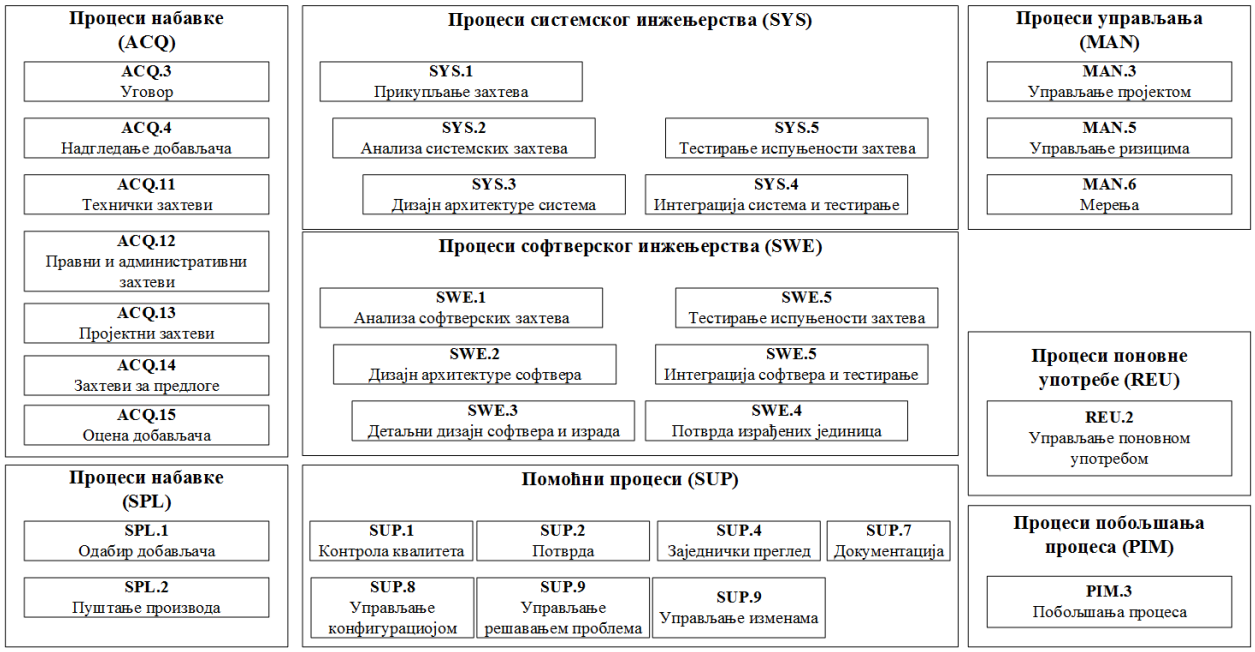
Према ASIL класификацији ризика у аутомобилској индустрији постоје четири нивоа ризика приказана у Табели 2.1 [6]. Ако коришћење неког система не носи никакав ризик, над њим се врши само контрола квалитета (енг. QM – Quality Management). У разматрање догађаја као што би био отказ инструмент табле узимају се вероватноћа изложености догађају, могућност контроле и последице. У овом примеру када возач између осталог губи информацију о тренутној брзини, вероватноћа изложености је велика. Задржава се пуна контрола над возилом али, на пример, услед кретања превеликом брзином могуће су тешке повреде тако да овај догађај и системи везани за њега спадају под степен ризика Б.

ASIL ниво	Последице	Могућност контроле	Вероватноћа изложености	Примери догађаја
QM	Нема последица	Нормална	Мала вероватноћа	Отказ радија
A	Лаке повреде	Нормална	Велика вероватноћа	Кашњење слике са задње камере
B	Тешке повреде	Нормална	Велика вероватноћа	Изостанак звучног упозорења на опасност од судара, отказ инструмент табле
C	Кобне, неизвесно преживљавање	Тешко за контролу	Средња вероватноћа	Блокада точкова, отказ аутоматског мењача
D	Кобне, неизвесно преживљавање	Тешко за контролу	Велика врватноћа	Блокада волана, активирање ваздушног јастука у току вожње

Табела 2.1 ASIL нивои ризика

Управљање пројектом подлеже ASPICE (енг. ASPICE – Automotive Software Process Improvement and Capability Determination) стандарду односно верзији ISO 33001 (енг. ISO 33001) стандарда прилагођеној аутомобилској индустрији [7]. Овај стандард дефинише процену спровођења процеса приликом развоја софтвера. Није довољно да производ буде добар, битан је процес доласка од идеје до производа. У зависности од тога колико добро су успостављени процеси и колико се поштују, пројекат добија оцену од нула до пет. Нула значи да процес није успостављен а петица да се процес поштује, да предвиђа потешкоће и чак да сам себе допуњава и побољшава. То би значило да је процесом предвиђено како се поступа у случају да се примети могућност побољшања које би се применило на следећи циклус.

На Слици 2.7 дат је графички приказ ASPICE модела за процену процеса. Од осам група процеса, SWE (енг. SWE - Software Engineering Process Group) се односи на софтверско инжењерство и спада у примарни животни. Поред примарног постоји и организациони и животни циклус процеса подршке. Обимност овог модела огледа се у тома што се од тридесетдва процеса само део процеса SWE.3 детаљни дизајн софтвера и израда (енг. Software Detailed Design and Unit Construction) односи на писање кода.



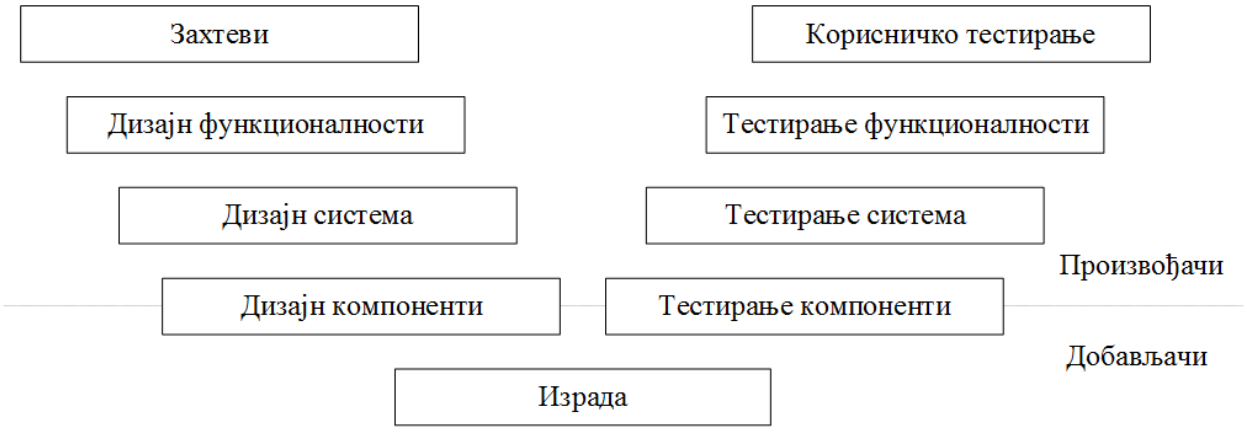
Слика 2.7 ASPICE модел за процену процеса

Увидевши сложеност ASPICE-а HIS група (нем. HIS - Herstellerinitiative Software) чији чланови су Ауди, BMW, Даимлер , Порше и Фолксваген направили су сужено издање прилагођено тренутном стању у индустрији. Задржали су само шеснаест неопходних процеса чиме је процена значајно поједностављена.

Иако ISO 26262 и ASPICE на вођење пројекта гледају из различитих углова, они се допуњују (Слика 2.8). ISO 26262 из угла безбедности а ASPICE из угла управљања пројектом заједнички описују управљање функционалном берbедношћу. На пример, развој производа на нивоу софтвера тј. шести део ISO 26262 стандарда одговара SWE групи ASPICE процеса. Вођење пројекта у складу са ASPICE-ом доноси признање консултантске куће да производи настају на јасно дефинисан начин односно у складу са ISO 26262 стандардом. То значи да су безбедносни ризици препознати, прорачунати и да је производ пројектован узимајући их у обзир.

Слика 2.8 Управљање функционалном безбедношћу возила

Најчешће коришћен развојни процес је V-model. Користи се за планирање развоја хардвера, софтвера као и целог система. Уже дефинише редослед и везу поступака од постављања захтева преко развоја до провере исправности и провере да ли су захтеви испуњени. Графички се представља у облику латиничког слова в, а пример софтверског V-model-а дат је на Слици 2.9. Чест је случај да добављачи на основу дизајна система праве дизајн компоненти које израђују и затим их тестирају. Коначно тестирање врши онај ко је поставио захтеве.



Слика 2.9 Софтверски V-model

Архитектуре аутомобилских система развијају се користећи AUTOSAR (енг. Automotive Open System Architecture) стандард. Његове улоге су стандардизација архитектуре микроконтролерских електронских контролних јединица (енг. ECU – Electronic Control Unit), методологије развоја, језика за опис архитектуре као и

архитектуре и функционалности средњег слоја софтвера [5]. Дефинише три софтверска слоја: апликативни софтвер, радно окружење (енг. RTE - Run-time environment), и основни софтвер (енг. BSW - Basic software). AUTOSAR мета-модел дефинише језик за описивање софтверске архитектуре. Архитектура се црта као UML дијаграм а затим серијализује у XML формат. Идеја је да се повећа преносивост апликативног софтвера, да се стандардизују интерфејси између компоненти и што већи део кода који не зависи од хардверске платформе да се напише аутоматски. Поред са класичног AUTOSAR-а који се извршава на микроконтролерима постоји и AUTOSAR Adaptive који се извршава у хипервизорским виртуелним машинама на микропроцесорима и представља основу будућих система аутономне вожње и умрежених возила.

Мноштво процедура и обимна документација који прате развој софтвера у аутомобилској индустрији значајно поскупљују коначан производ. Цена додатног хардвера само је мали део у односу на време и рад које је потребно утрошити да сви делови задовоље стандарде аутомобилске индустрије. Један од изазова је интеграција више система чак и из различитих домена, различите безбедносне критичности односно њихово изоловано извршавање коришћењем виртуелизационих технологија на истом микропроцесору. Смањење броја хардверских компоненти уз задржавање постојећих функционалности и додавање нових омогућило би бржи и јефтинији развој нових електронских система у аутомобилској индустији.

2.1.3 Оперативни системи

Један од два најчешће коришћена оперативна система у потрошачкој електроници је Андроид (енг. Android) и то из више разлога. Изворни код Андроида доступан је јавности, апликативне програмске спреге (енг. API – Application Programming Interface) добро су документоване, и што је можда и најбитније, апликације је могуће програмирати у Јави (енг. Java), једном од најзаступљенијих програмских језика на свету. Ово га чини веома погодним за примену код инфозабавних система у возилима. Међутим са аспекта безбедности, овај систем не задовољава све критеријуме за употребу и у инструмент табли.

GENIVI алијанса обезбеђује стандардизацију архитектуре инфозабавних система. Главни производ је софтверска развојна платформа прилагођена већини хардверских платформи за аутомобилску индустрију али и јефтиним плочама као што је Распбери Пи. Референтна аритектура GENIVI Baseline доступна је као отворени код и служи за проверу да ли је нека софтверска компонента у складу са овом архитектуром. Произвођачи

софтвера, чланови ове алијансе могу да добију потврду да је њихов инфозабавни систем у складу са GENIVI архитектуром.

AGL (енг. AGL – Automotive Grade Linux) је пројекат подржан од стране Линукс фондације, многих произвођача аутомобила и добављача инфозабавних система. Још увек је у фази развоја а за циљ има да подржи употребу не само у инфозабавном систему већ и у инструмент табли. Ови системи омогућавају брз развој софтвера али још увек не задовољавају безбедносне стандарде како би били самостално коришћени у аутомобилској индустрији.

Примери оперативних система који задовољавају критеријуме аутомобилске индустрије по питању безбедности су Блекбери QNX Неутрино (енг. Blackberry QNX Neutrino) и Винд Ривер VxWorks (енг. Wind River VxWorks) [8]. QNX је већ дуго година на тржишту, проверен и доказано поуздан у разним индустријама. Заснован је на микрокернелу односно само минималан скуп функција се извршава унутар језгра а све остало укључујући руковаоце извршава се у корисничком простору. Делови језгра комуницирају путем порука које се распоређују на основу приоритета нити. Нити вишег проритета добијаће, на пример, улазно-излазне поруке пре нити нижег приоритета што овом систему омогућава рад у реалном времену.

VxWorks је такође веома заступљен систем за рад у реалном времену. Подржава мноштво различитих платформи а користи се у авионима, свемирским летелицама, аутомобилима, индустријским роботима, мрежним уређајима и медицинским апаратима за операције и терапије. Такође је заснован на микрокернелу за рад у реалном времену а одликују га модуларна архитектура, подршка за већину модерних протокола и добро развојно окружење са емулатором. Омогућава изолацију апликација и заштиту система ако нека од њих пукне, као и контролу комуникације између њих.

Приликом пројектовања архитектуре система потребно је размотрити из којих целина ће се систем састојати и како ће међусобно комуницирати. Потребно је предвидети на ком оперативном систему ће се извршавати који део софтвера али и узети у обзир да је обавезна изолација различитих нивоа наспрам безбедносних захтева. Такође унутар једног безбедносног нивоа апликације се морају заштитити једне од других тако да се не дозволи директно писање једне апликације по меморији друге.

2.2 Виртуелизација

Безбедни оперативни системи за рад у реалном времену нису погодни за брз развој графичких мултимедијалних апликација док оперативни системи из домена потрошачке електронике нису сертификовани по стандардима аутомобилске индустрије. То значи да

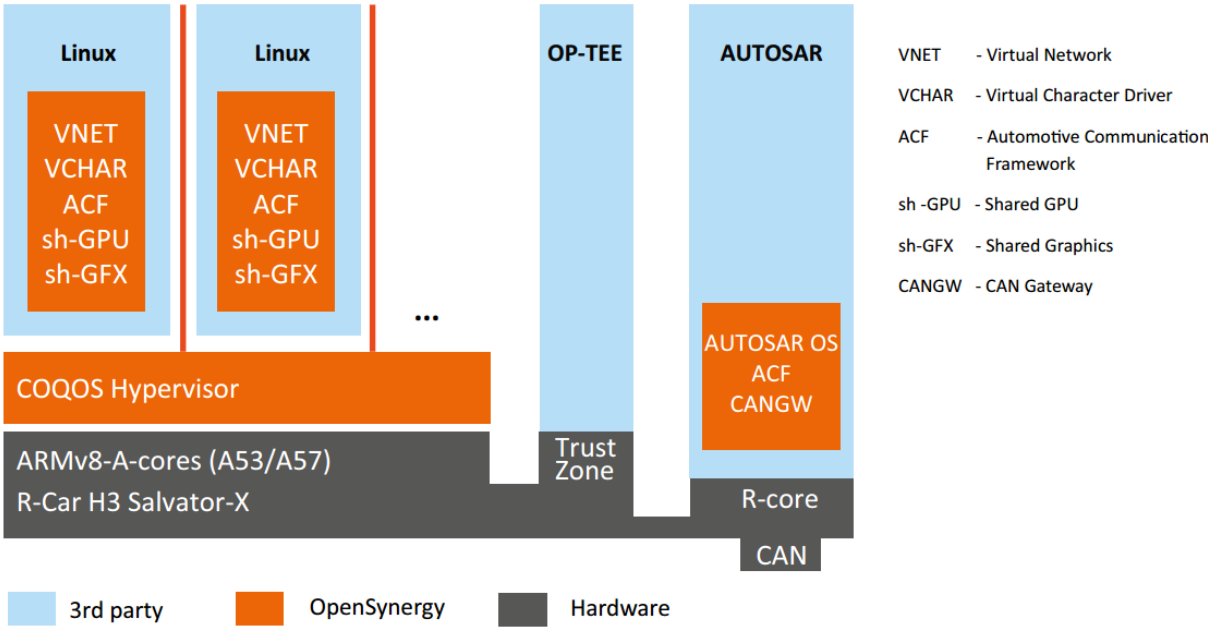
тренутно није могуће одабрати један оперативни систем за једну хардверску платформу који ће испунити све захтеве. Тако долазимо до потребе за постојањем више оперативних система на једној хардверској платформи а решење за то је хардверска виртуелизација. Од свих врста виртуелизације за решавање проблема овог рада избор се своди на два типа хипервизора. Код хипервизора првог типа гостујући оперативни системи покрећу се из виртуелизационог слоја софтвера. Хипервизор другог типа гостујући оперативни систем покреће као процес унутар другог оперативног система. Хипервизор распоређује процесорско време, приступ меморији, графичкој јединици и другим уређајима.

Хипервизор треба да обезбеди изолацију слојева и компоненти у складу са ISO 26262 стандардом. Пример може да буде хипервизор првог типа са два оперативна система. У једном се извршавају апликације са безбедносно критичним захтевима а у другом све остало попут мултимедије или навигације. Дакле могуће је постићи уштеду без губитка безбедности или успоравања развоја.

Приликом одабира хипервизора треба размотрити више аспеката [9]. Добра хардверска подршка је основа за добру виртуелизацију. Данашње јединице за управљање меморијом веома су развијене, међутим хардверска подршка за виртуелизацију још увек је у развоју и сви пропусти у развоју хардвера морају бити софтверски решени. Виртуелизација додаје још један слој софтвера који распоређује ресурсе а један од кључних делова је графичка јединица (енг. GPU - Graphical Processing Unit). У случају када имамо два оперативна система где на пример један приказује инфозабавни систем а други приказује упозорење на пешаке испред возила, виртуелизациони слој мора паметно да распореди ресурсе како би упозорење на пешаке било приказано без кашњења и застајкивања. Такође потребно је изоловати апликације једне од других унутар једног оперативног система. Ако безбедносно критични систем покреће дигиталну инструмент таблу и други дисплеј за смањено ометање возача те две апликације приказиваће неке заједничке информације у исто време али независно једна од друге тако да ако једна откаже друга наставља неометано да ради. Главни разлог увођења хипервизора је појефтињење система, међутим, треба имати на уму да одржавање система мора да се настави и након изласка возила из производње тако да у цену треба урачунати и одржавање софтвера за два различита оперативна система.

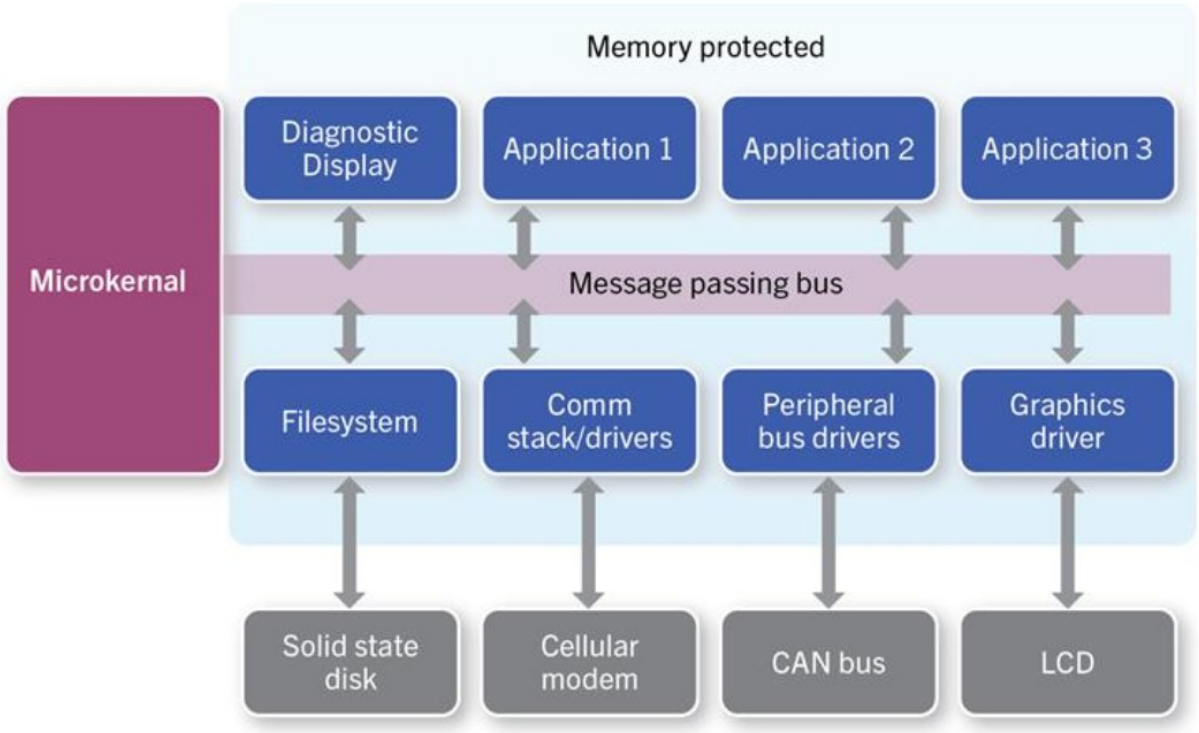
Пројекат xLuna Automotive настао је по узору на xLuna Space, и за циљ је имао да омогући безбедносно критичне позиве у помоћ на платформи Freescale i.MX 6 Quad намењеној инфозабавном систему [10]. Основа за развој хипервизорског слоја били су FreeRTOS и ARM TrustZone технологија за изолацију, док је за инфозабавни систем коришћен Андроид.

Опенсинерџи COQOS (енг. OpenSynergy COQOS) је модуларан пакет софтвера заснован на хипервизорској архитектури који користе многи у аутомобилској индустрији. Општи приказ овог система дат је на Слици 2.10 [11]. Подржава рад са више оперативних система одједном на апликативним процесорским језгрима. Интегрише се и са АУТОСАР окружењем на посебном језгру за рад у реалном времену и омогућава безбедну комуникацију између компоненти. Прављен је да користи све погодности модерних чипова па се користи и за напредне системе за помоћ возачу.



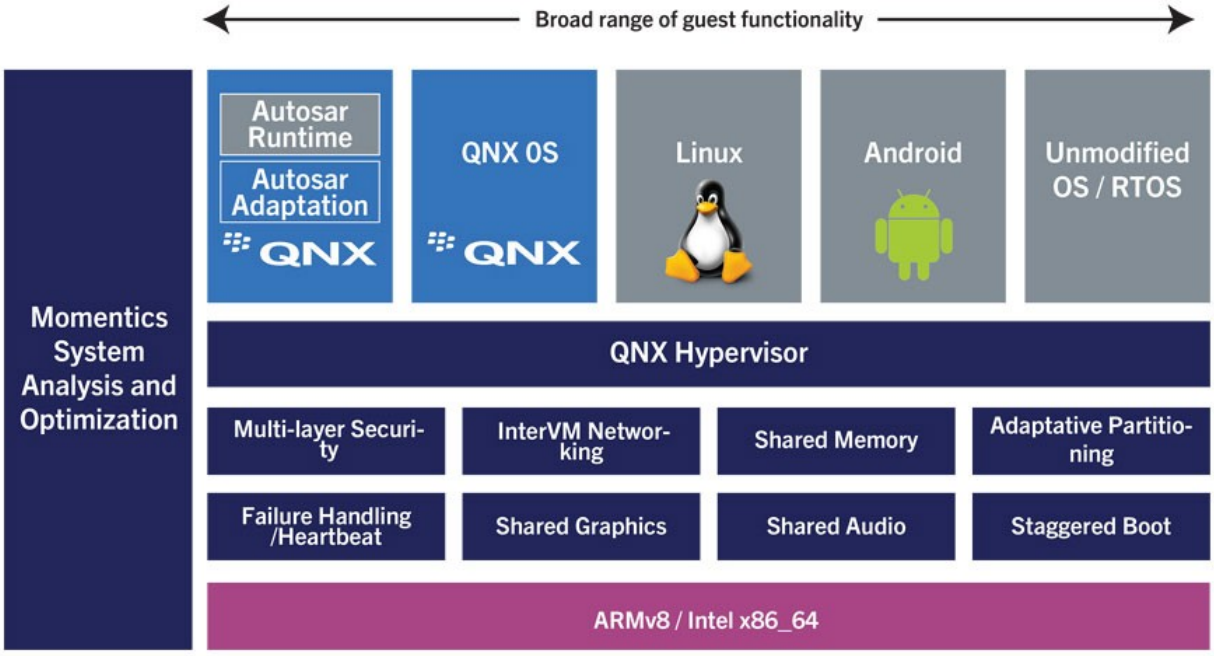
Слика 2.10 Пример Опенсинерџи хипервизорског система

QNX је развио хипервизорско проширење за добро познати Неутрино оперативни систем за рад у реалном времену. Неутрино ради на принципу размене порука путем магистрале између микрокернела и осталих делова система (Слика 2.11) [12]. На тај начин добили су решење сертификовано за коришћење у индустријском, медицинском и аутомобилском софтверу.



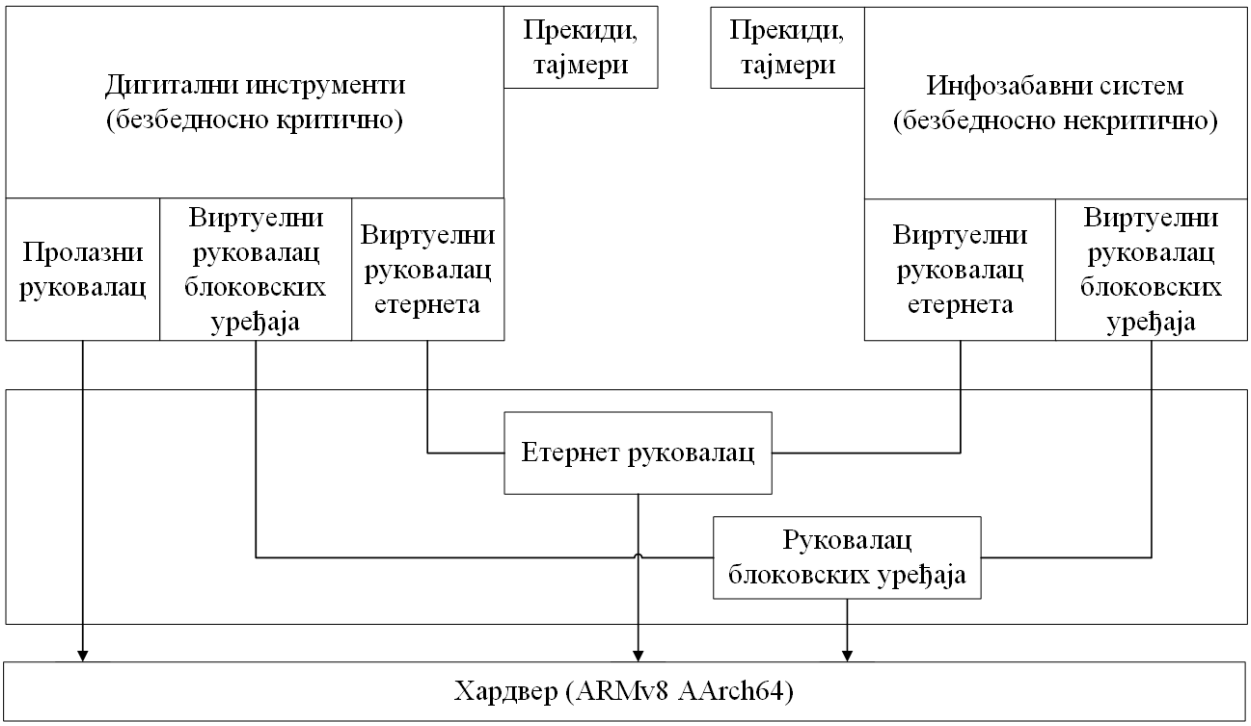
Слика 2.11 Магистрала за поруке микрокернелског QNX оперативног система

Хипервизор у најновијој верзији 2.0 заснован је на Неутрину 7.0 и то је први систем који, поред Интел (енг. Intel) x86-64 процесора, подржава и АРМв8 (енг. ARMv8) шездесетчетворобитне архитектуре. Подржане су хардверске оптимизације, и направљени су развојни пакети за све модерне развојне платформе познатих произвођача Интел, Ренесас (енг. Renesas), NXP и Куалком (енг. Qualcomm). Виртуелне процесоре који користе распоређивање задатака на основу приоритета могуће је распоредити на физичке процесоре тако да неки од њих добију искључиво право коришћења одређених језагара. Тиме се максимално искоришћава расположиво процесорско време уз гарантовање извршавања приоритетних задатака. На Слици 2.12 дат је општи приказ система заснованом на QNX Хипервизору са више оперативних система.



Слика 2.12 Пример QNX хипервизорског система

Виртуелне машине уређајима могу да приступе на више начина. Постоје емулирани уређаји попут тајмера, пролазни руковаоци који би се користили код CAN магистрале као и виртуелни руковаоци коришћени за блоковске уређаје система датотека. На Слици 2.13 приказан је сценарио са три врсте дељених уређаја.



Слика 2.13 Различити начини дељења уређаја

Графичку јединицу може да контролише безбедносно критични оперативни систем који прима команде од осталих система и исцртава слику на једном или више екрана.

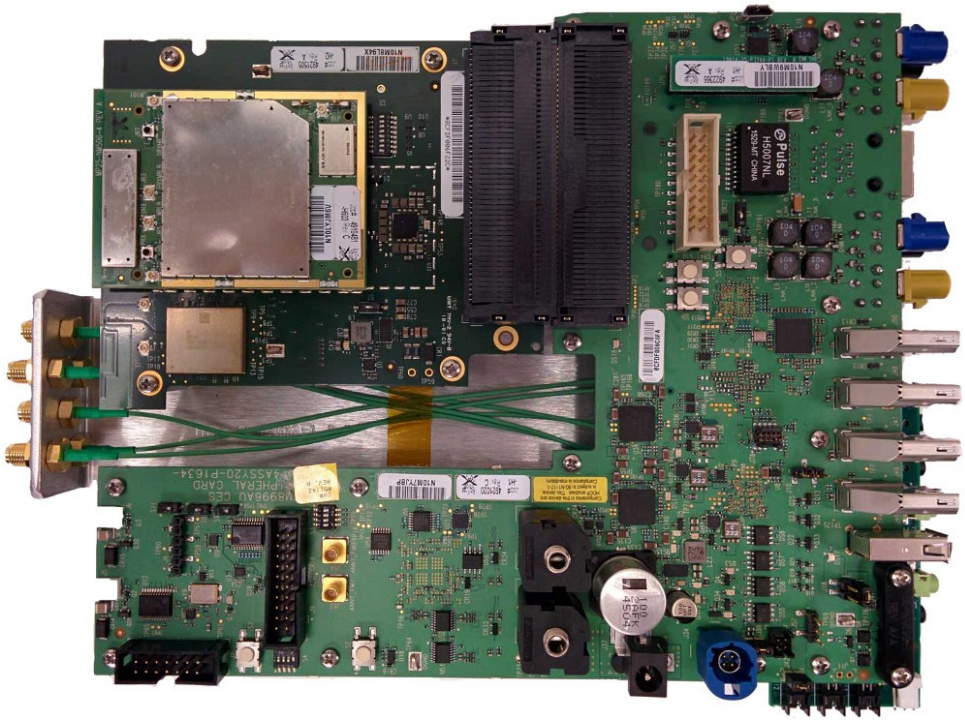
Други начин је коришћење виртуелних руковаоца и у том случају сви оперативни системи могу да користе графичку једниницу у исто време. Улога хипервизора је да посредује односно контролише приступ виртуелних руковалаца графичкој јединици. На Слици 2.14 приказана су два оперативна система где је сваком додељен по један екран а све то се исцртава у исто време на једној графичкој јединици посредством хипервизора. Први оперативни систем покреће апликацију са инструментима док је други намењен инфозабавном систему са навигацијом. У овом случају виртуелни руковаоци омогућавају комбиновање видео слојева два система односно приказ навигације са другог система на првом без нарушавања безбедности.

Слика 2.14 Хипервизор са виртуелним графичким руковаоцима

2.3 Избор технологија

Када се у обзир узму тренутно расположиве технологије и стандарди које је потребно испоштовати, избор се значајно сужава. Развој нових система често се одвија упоредо на више нивоа. Развој софтвера за нови хардвер креће и пре него што он постане доступан. Виртуелне машине и пробне верзије оперативног система на развојним плочама користе се за развој у раним фазама. Када се донесе коначна одлука о томе који хардвер ће се уграђивати у серијској производњи, стабилна верзија оперативног система и добар део софтвера већ су развијени. Даљи развој тече у смеру мањих проширења функционалности, прилагођења корисничког интерфејса и тестирања система.

Према потребама тржишта и циљној категорији возила одређује се шта све нови систем мора да подржи. Возачи малих градских аутомобила, луксузних лимузина, камиона, багера и аутобуса имају различите потребе и резличите проблеме са којима се суочавају током вожње. Међутим заједничко им је да би им свима помогле камере које би покриле непрегледне углове, користио би им прегледан екран са графичким информацијама о стању возила, мултимедијални садржаји како за возаче тако и за путнике као и умреженост са осталим возилима у циљу скраћења времена путовања и повећања безбедности. Изабрани хардвер осим што мора да задовољи техничке услове пројекта, мора и да се уклопи у планирану цену. Ту улогу не игра само количина већ и претходна сарадња са произвођачима, као и могућност проширења производње. За потребе овог рада одабрана је Куалком Снепдрегон (енг. Snapdragon) C820A развојна платформа за аутомобилску индустрију (Слика 2.15).



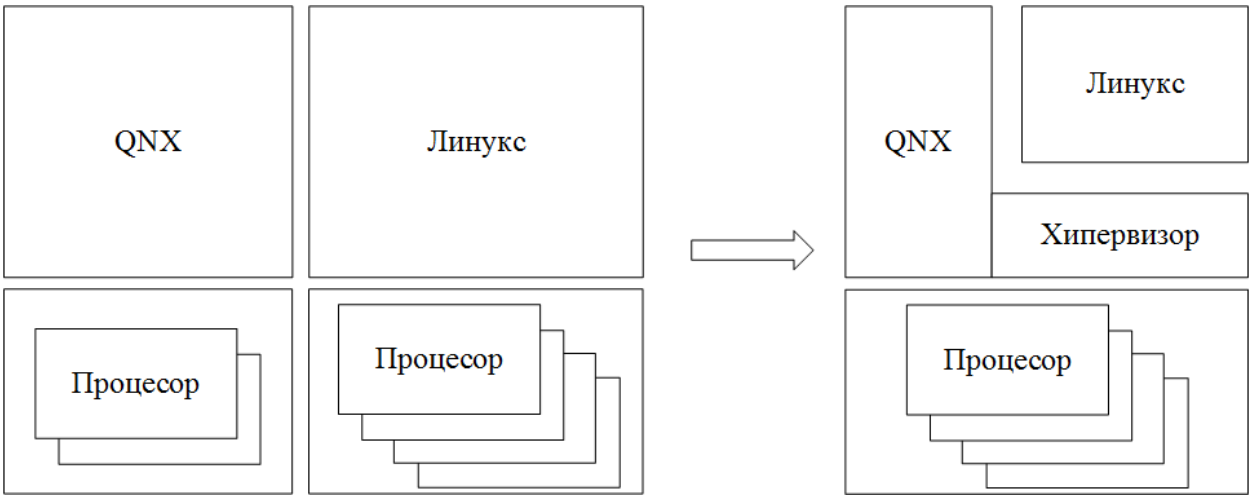
Слика 2.15 Куалком Снепдрегон C820A развојна платформа

Главни разлози за одабир баш ове платформе су што је систем на чипу C820 доказан у потрошачкој индустрији у мобилним телефонима високе класе, активно прилагођавање најновијих оперативних система за рад на овом чипу, као и постојећа сарадња са произвођачима хардвера и оперативних система.

За инфозабавну јединицу изабран је AGL као опште решење за безбедносно некритичан део софтвера. Ослања се на заједницу отвореног кода уз учешће и усмеравање од стране аутомобилске индустрије. Идеја овог система је да се делови софтвера који су заједнички за све пројекте пруже као отворени код и тиме избегне вишеструко трошење

времена и новца. Модуларан приступ дозвољава комбиновање са власничким софтвером који се често развија у уско специјализованим фирмама. Једино што преостаје је кориснички интерфејс који се прави посебно за сваки нови модел аутомобила користећи више нивое апстракције [13][14][15].

QNX Неутрино оперативни систем је често употребљавано стабилно решење за критичан софтвер, са добром документацијом и подршком. Са хипервизорским проширењем чини јединствену комбинацију која задовољава више потреба. Један оперативни систем може у исто време да извршава безбедносно критичне задатке, и да служи као домаћин другом виртуелном оперативном систему. У овом случају постиже се вишеструка уштеда. Уместо два одвојена хардвера све ће се извршавати на једном систему на чипу, и то на два оперативна система. Један је QNX Неутрино са хипервизором за безбедносно критични софтвер и други је виртуелни линукс за инфозабавни систем као на Слици 2.16.



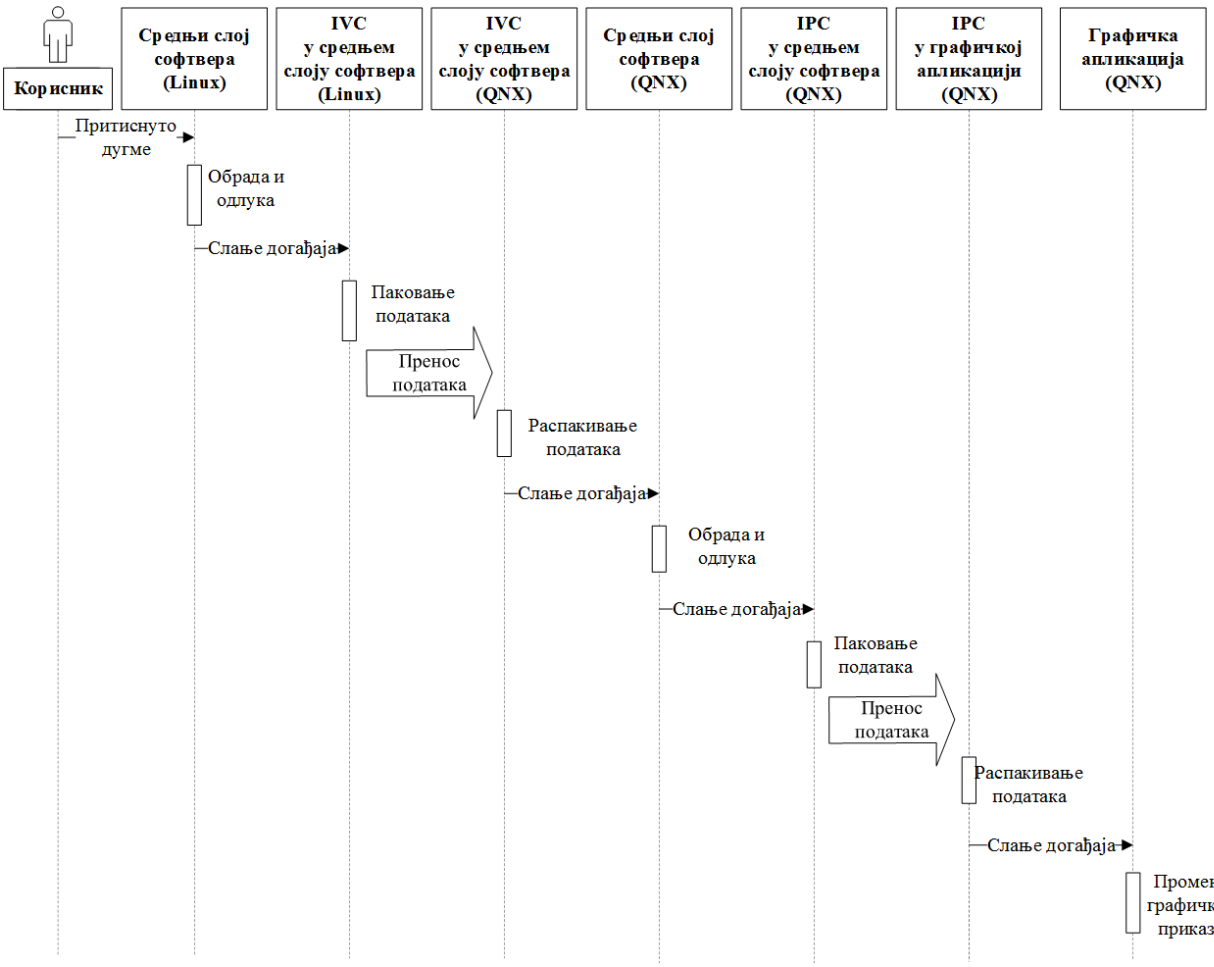
Слика 2.16 Замена два хардвера једним

2.4 Комуникација у оквиру виртуелизованог система

За комуникацију ка сензорским и другим јединицама у аутомобилу могу да се користе неке од аутомобилских магистрала. У зависности од врсте и учестаности података може се предвидети потребан проток и одабрати најјефтиније решење које задовољава потребе. Одабрана развојна платформа са системом на чипу C820A подржава комуникацију путем ЛИН (енг. LIN - Local Interconnect Network), CAN, и Етернет (енг. Ethernet) магистрала.

Да би систем радио као једна целина, графичке апликације и помоћни софтвер морају да комуницирају међусобно. У погледу безбедносне изолације система, ISO 26262 ASIL B условљава једносмерну комуникацију од безбедносно критичног ка безбедносно некритичном систему. Једна од предности коришћења хипервизора у односу на потпуно

независне системе је и могућност оптимизоване размене графике између оперативних система посредством виртуелних графичких руковалаца. Приказ графике са безбедносно некритичног система на делу екрана безбедносно критичног система не нарушава стандард у случају да ово понашање контролише безбедносно критични систем. Комуникација између процеса (енг. IPC - Inter-process Communication) унутар једног оперативног система вршиће се путем утичница (енг. socket), док ће се комуникација између процеса у различитим оперативним системима односно виртуелним машинама (енг. IVC - Inter-virtual-machine communication) вршити путем мрежних утичница (енг. network socket). Разлика је у томе што се у првом случају комуникација обавља унутар језгра оперативног система, док у другом случају иде преко мрежног интерфејса. Пример тока комуникације од тренутка када корисник притисне неко дугме до промене графичког приказа у апликацији дат је на Слици 2.17. Комуникација која се одвија између два оперативна система врши се путем IVC механизма док се комуникација унутар једног оперативног система врши путем IPC механизма. Графичка апликација по пријему догађаја, омогућава приказ графичког садржаја са другог система.



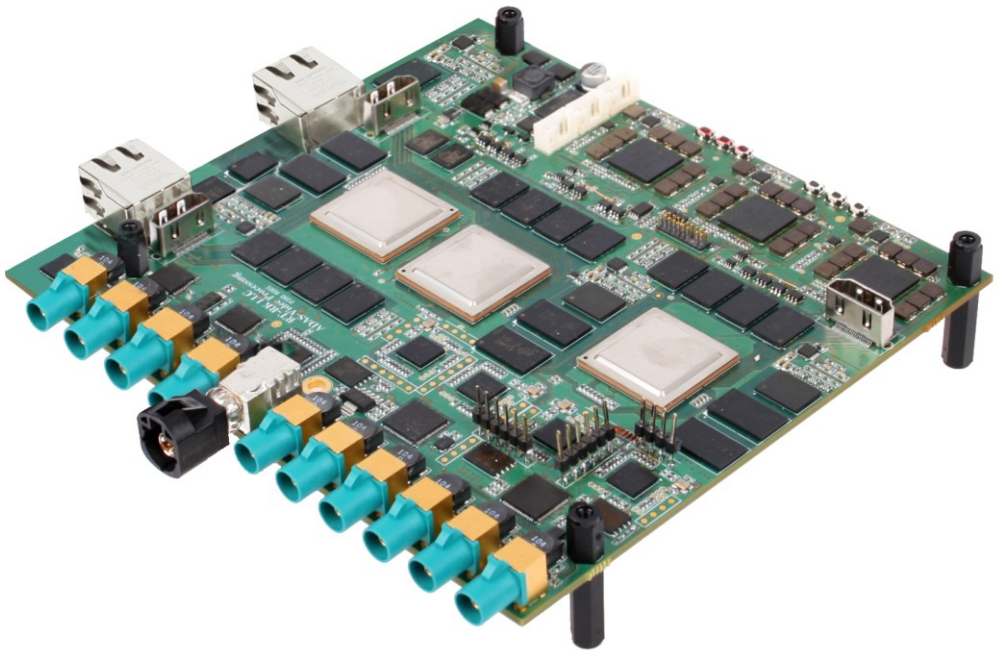
Слика 2.17 UML дијаграм тока комуникације

3. Концепт решења

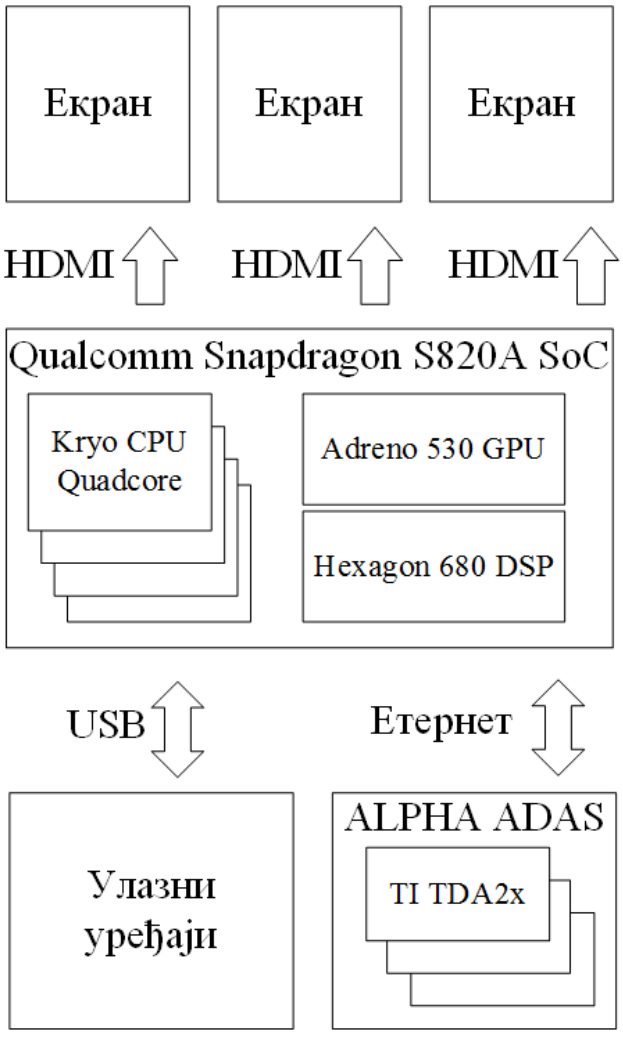
У овом поглављу приказан је концепт виртуелизованог система са предложеним комуникационим проширењима. Софтвер је осмишљен са циљем да корисник стекне утисак јединствености система и поред тога што се његови делови извршавају независно и то на више екрана. Такође приказани су и примери графичких апликација које се извршавају у виртуелизованом окружењу и користе предложени комуникациони софтвер. С обзиром на циљану примену у аутомобилској индустрији, нагласак је стављен на поделу система на безбедносно критични и безбедносно некритични део. С обзиром на уочену потребу за умрежавањем са другим системима у возилу [16], као додатак омогућено је повезивање са ADAS системима.

3.1 Архитектура система

На изабрану платформу са C820A системом на чипу повезана су три екрана. Први замењује аналогне инструменате испред возача (енг. instrument cluster), други помаже возачу приказујући само најбитније информације тако да возач не мора да спушта поглед са ветробранског стакла (енг. HUD - Head-up Display) а трећи је део инфозабавног система и осетљив је на додир. У сврху симулације утиска у возилу на систем су повезани волан са тастерима, педале и џогшатл (енг. jog shuttle), посебан уређај за управљање инфозабавним системом. За приказ информација односно пренос видео садржаја (енг. streaming) са ADAS система коришћена је АЛФА развојна платформа са три Тексас Инструментс (енг. Texas Instruments) TDA2x интегрисана кола повезана преко етернет мреже (Слика 3.1). Уређаји и њихове везе представљени су на Слици 3.2.



Слика 3.1 АЛФА развојна плоча



Слика 3.2 Блок-дијаграм хардвера са мониторима и периферијама

Коришћена су два оперативна система QNX са хипервизором и AGL. На првом се извршавају безбедносно критичне апликације а на другом безбедносно некритичне. QNX има контролу над ресурсима као што су процесорско време, радна меморија, графичка једниница и дели их са виртуелним AGL-ом. Користећи посебан руковалац виртуелна машина приступа виртуелној графичкој јединици. На тај начин постиже се максимално искоришћење ресурса уз изолацију два система једног од другог. Овакав распоред система омогућава истовремено извршавање процеса различитог приоритета и критичности без нарушавања безбедности.

Развијен је средњи слој софтвера (енг. Middleware) као подршка комуникацији између графичких апликација. С обзиром да се комуникација одвија како између апликација на једном оперативном систему тако и између два оперативна система, потребно је покрити два случаја. Три графичке апликације ослониће се на графичке библиотеке прилагођене оперативном систему на којем ће се извршавати као и на средњи слој софтвера. На Слици 3.3 дат је општи приказ средњег слоја софтвера односно његовог распореда по оперативним системима као и предвиђених места за графичке апликације.

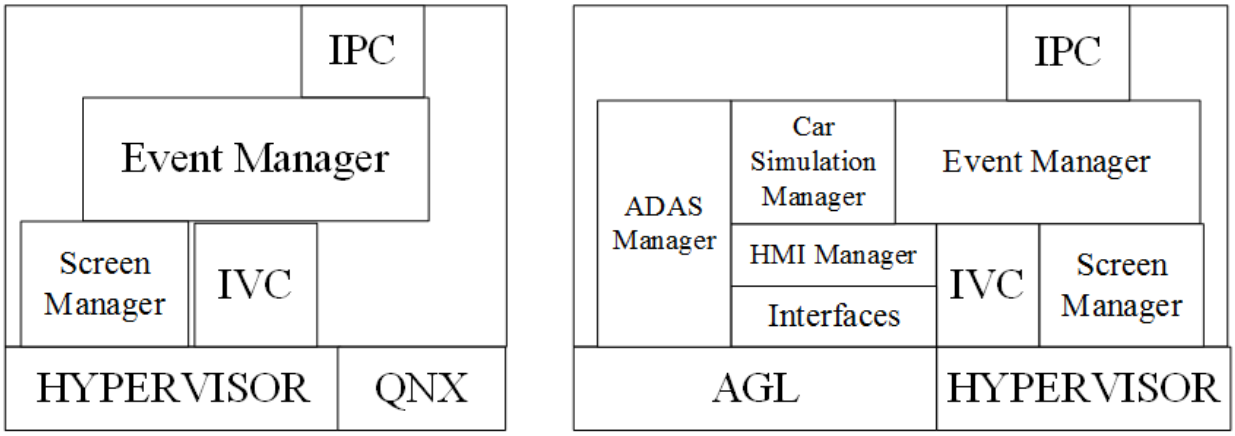
Слика 3.3 Распоред софтвера по оперативним системима

3.2 Средњи слој софтвера

Замисао је да у коначној верзији логика буде на страни безбеднијег оперативног система односно QNX-а и да се обавештење о његовом стању шаље другом систему. На тај начин безбедносно критични део система био би у сагласности са ISO 26262 ASIL B стандардом. Због посебности услова рада на изради решења у овом раду, систем је направљен тако да је логика на страни Линукса али ће бити могуће пребацити је на страну QNX-а. Средњи слој састоји се од следећих модула:

- Event Manager је контролни модул који на основу улаза од различитих модула распоређује задатке за апликације.
- Screen Manager је модул за дељење графичких елемената између графичких апликација. Ослања се на могућности хипервизора.
- HMI Manager је веза система са корисничким командама.
- Car Simulation Manager преузима симулационе податке из базе података и прослеђује их у Event Manager.
- Shared Element прати стање односно позицију дељених елемената на екрану.
- ADAS Manager омогућава пријем обрађеног видео тока са посебне ADAS платформе.
- IPC је група модула који врше трансформацију података, паковање у низ бајтова и распакивање из низа бајтова у одговарајуће структуре података, и пренос података између процеса.
- IVC слично као IPC али намењено комуникацији између процеса у различитим оперативним системима односно виртуелним машинама.

Механизам функционише тако што Event Manager контролише све модуле путем одговарајућих редова (енг. queue). Модули информације намњене за Event Manager стављају у редове а Event Manager их узима из редова. Овакав дизајн захтева коришћење једне или више нити које у петљи читају све редове. Средњи слој софтвера за AGL и QNX разликују се у неким деловима и приказани су на Слици 3.4.



Слика 3.4 Средњи слој софтвера на два оперативна система

Event Manager као главни део средњег слоја софтвера има непосредан приступ већини информација али неке попут оних о додиру екрана достављају му се посредно. Овај модул управља системом на основу порука које добија. Поруке скупљају Manager модули намењени појединим деловима система и стављају их у редове из којих их Event Manager преузима. На графику тока података (Слика 3.5) може се видети пример слања информације за Event Manager о додиру екрана.

Слика 3.5 Ток података у систему

Када корисник додирне екран, апликација инфозабавног система добија догађај од оперативног система. Апликација прослеђује информацију у Event Manager користећи IPC. На другој страни IPC ставља информацију у ред из којег је преузима Event Manager.

Screen Manager служи за дељење видеа и друге графике између различитих дисплеја. Користи се да покрене или заустави видео односно прикаже или склони слику на одређеној позицији.

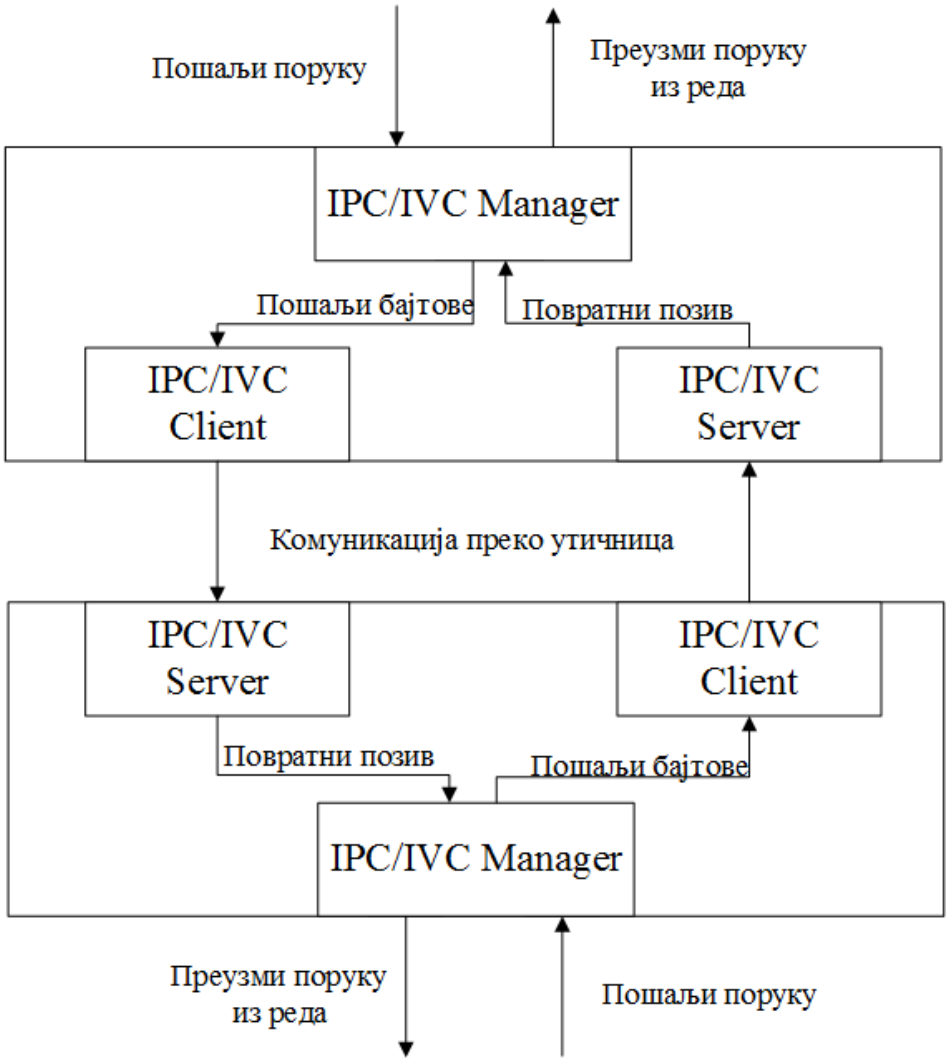
HMI Manager преузима корисникове акције кроз улазне уређаје. Садржи низ интерфејса који догађаје односно поруке стављају у један ред. Ово се врши путем функције повратног позива (енг. callback) који позивају сви интерфејси. Поруке из реда преузима Event Manager и обрађује их.

Car Simulation Manager намењен је за симулацију стања аутомобила као што су количина горива у резервоару, брзина кретања итд. Из базе података чита податке о стању и преко реда их прослеђује у Event Manager на обраду.

Shared Element служи за праћење стања дељених елемената на екранима. Стање означава на ком екрану се приказује елемент а могуће је да исти елемент буде приказан на више екрана.

ADAS Manager са пратећим модулима служи за пријем MJPEG видео тока са додатне умрежене ADAS платформе. Коришћена је АЛФА развојна платформа. На ADAS платформи могуће је покренути више различитих алгоритама за помоћ возачу. Осим пријема тока потребно је обезбедити препознавање садржаја односно слика и сместити их на одговарајуће место у меморији са којег апликације могу да их преузму путем повратног позива.

IPC Manager је модул који управља комуникацијом између процеса. Када преузме објекат поруке, претвара је у низ бајтова који затим преко IPC Client шаље. IPC Client служи да успостави конекцију са другом страном за комуникацију и да пошаље низ бајтова. IPC Server прихвата конекцију од друге стране и прихвата низ бајтова. Примљен низ бајтова IPC Server предаје у IPC Manager путем повратног позива. Резултат обраде низа бајтова су објекти који представљају поруке. IPC Manager нове поруке складишти у реду из којег их преузима Event Manager. IVC Manager, IVC Client и IVC Server функционишу на истом принципу као и IPC али су намењени за комуникацију процеса на различитим оперативним системима односно виртуелним машинама. Ток комуникације и између IPC односно IVC Manager, Client и Server модула графички је приказан на Слици 3.6.



Слика 3.6 IPC и IVC Client, Server и Manager

Модули система комуницирају путем порука односно посебних врста објеката. Свака порука има тип и одредиште, а неке врсте имају и додатна поља.

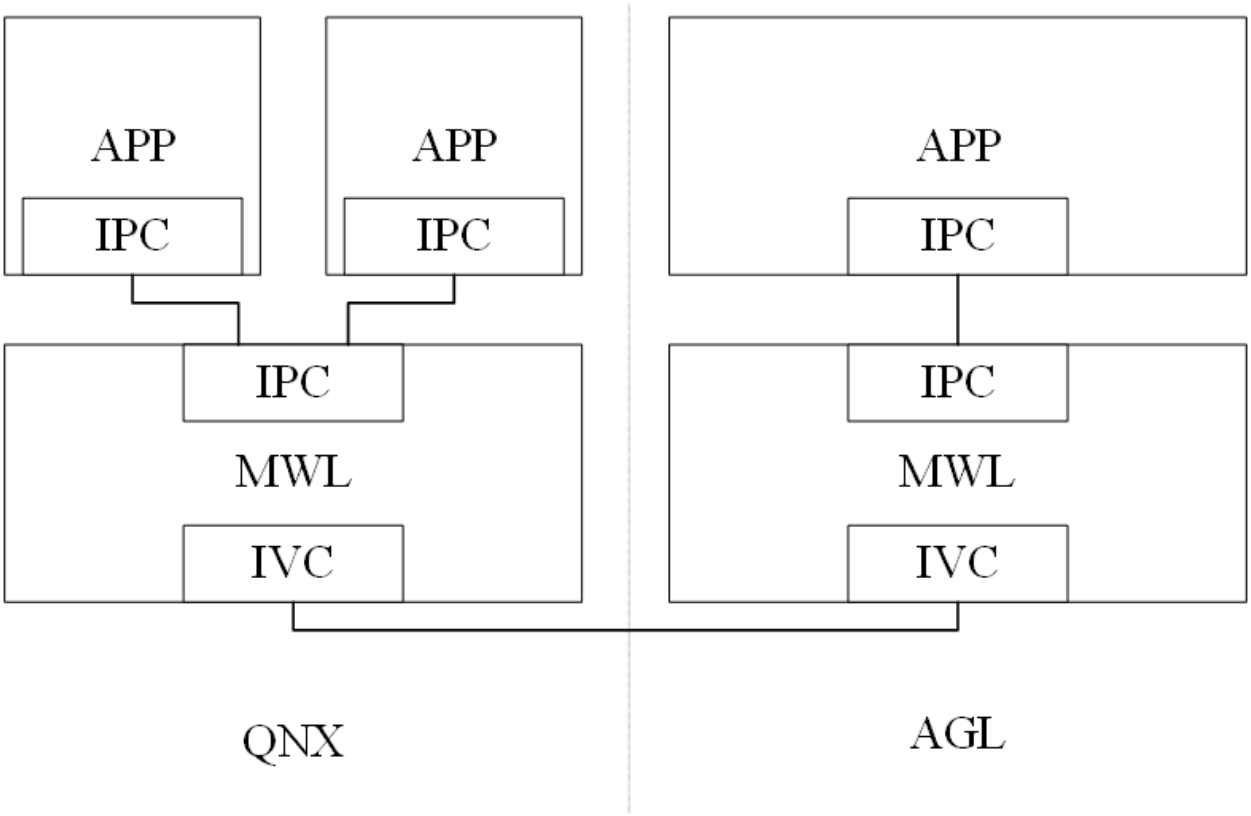
HMIInputMessage представља корисничку акцију насталу употребом неког од улазних уређаја. Додатно поље је идентификација уређаја, а сваки уређај имаће своју подврсту поруке са специфичним информацијама. Улазни уређаји су тастатура, волан, додири екрана.

SharedElementMessage је тип поруке који преноси информацију о дељеним елементима. Садржи идентификатор врсте садржаја нпр. видео или графички елемент. Свака врста дељеног елемента имаће додатна поља попут путање до видео датотеке и управљања пуштањем видеа.

HMIFeedbackMessage користи се када апликација захтева акцију од средњег слоја софтвера односно интерфејса које контролише. Садржи и идентификатор интерфејса и методу за преузимање информације. Подврсте имају и додатна поља за врсту акције попут додира или пуштања екрана.

3.3 Графички приказ

Израда графичких апликација није тема овог рада али са циљем појашњења сврхе средњег слоја софтвера биће укратко објашњена идеја односно њихова планирана намена. Предвиђено је постојање три графичке апликације, на сваком екрану по једна. Комуникација са средњим слојем обавља се IPC механизмом (Слика 3.7). Средњи слој не обавезује на коришћење одређене технологије односно алата за прављење графичких апликација. С обзиром на доступност библиотека и алата за овај систем коришћено је Куте (енг. Qt) окружење.



Слика 3.7 Веза апликација и средњег слоја софтвера

Два екрана испред возача замишљена су са инфорамционом наменом. Возачу приказују информације о стању возила као што су тренутна брзина, број обртаја мотора. С обзиром да могу да комуницирају са инфозабавним системом могу и да прикажу информације о њему, наравно у облику прилагођеном за што мање одвраћање пажње возача. Неке од њих су упрошћена сателитска навигација или који је тренутно активан аудио или видео садржај. На Слици 3.8 виде се две апликације намењене за рад на безбедносно критичној страни система.

Слика 3.8 Две апликације на безбедносно критичној страни система

Централни екран више је намењен мултимедијалним садржајима али такође постоји могућност коришћења повезаности са остатком система односно може да приказује информације о стању возила (Слика 3.9). Уобичајени примери употребе су приказ сателитске навигације, контрола и приказ видео и аудио садржаја, приказ информација са ADAS система, приступ садржајима са мобилног телефона, приступ садржајима са интернета, контрола грејања, вентилације и климе (енг. HVAC – Heating Ventilation and Air Conditioning) итд.

Слика 3.9 Инфозабавни систем

4. Програмско решење

У овом поглављу описани су израђени програмски модули средњег слоја софтвера и дат је преглед функција апликативне програмске спреге.

4.1 Преглед програмских модула

Средњи слој софтвера HMI MWL (енг. Human Machine Interface Middleware) направљен је из два слична дела који се извршавају на различитим оперативним системима. Обухваћене целине приказане су у Табели 4.1.

Оперативни систем	Програмски модули
QNX	Event Manager IPC Manager Cluster IPC Manager HUD Shared Element IVC Manager Screen Manager
Linux	Event Manager IPC Manager Shared Element IVC Manager Screen Manager Car Simulation Manager HMI Manager Adas Manager

Табела 4.1 Модули средњег слоја софтвера

4.2 Опис програмских модула

Апликативна спрега EventManager.hpp је једноставна (Табела 4.2). Када се покрене на основу прочитаних порука позива одговарајуће приватне функције.

Функције	Опис
EventManager ()	Подразумевани конструктор.
~EventManager ()	Подразумевани деструктор.
bool run ()	Покреће све компоненте под његовом контролом.
bool isRunning ()	Враћа одговор да ли је покренут.

Табела 4.2 EventManager

У IPCManager.hpp описана је спрега контролног модула за комуникацију између процеса у једном оперативном системиму и ослања се на клијента и сервера (Табела 4.3).

Функције	Опис
IPCManager (int portClient, int portServer, char *address)	Конструктор. Прави IPCClient и IPCServer објекте.
IPCManager (int port, eIVCLinkType type, char *address)	Конструктор који прави или IPCClient или IPCServer.
virtual ~IPCManager ()	Деструктор.
bool startClient ()	Покреће IPCClient.
bool startServer ()	Покреће IPCServer.
bool stopClient ()	Зауставља IPCClient.
bool stopServer ()	Зауставља IPCServer.
bool send (HMIMWLMessage *message)	Шаље поруку.
HMIMWLMessage * getNextMessage ()	Узима следећу поруку из реда.
void callback (unsigned char *packedMessage, int size)	Повратни позив који окида IPCServer када је следећа порука спремна.

Табела 4.3 IPCManager

Аналогно у IVCManager.hpp описана је спрега главног контролног модула за комуникацију између процеса у различитим оперативним системима и ослања се на клијента и сервера (Табела 4.4).

Функције	Опис
IVCManager (int portClient, int portServer, char *address)	Конструктор. Прави IVCClient и IVCServer објекте.
IVCManager (int port, eIVCLinkType type, char *address)	Конструктор који прави или IVCClient или IVCServer.
virtual ~IVCManager ()	Деструктор.
bool startClient ()	Покреће IVCClient.
bool startServer ()	Покреће IVCServer.
bool stopClient ()	Зауставља IVCClient.
bool stopServer ()	Зауставља IVPCServer.
bool send (HMIMWLMessage *message)	Шаље поруку.
HMIMWLMessage * getNextMessage ()	Узима следећу поруку из реда.
void callback (unsigned char *packedMessage, int size)	Повратни позив који окида IVCServer када је следећа порука спремна.

Табела 4.4 IVCManager

Функције за контролу стања дељеног елемента описане су у SharedElement.hpp (Табела 4.5).

Функције	Опис
SharedElement (eSharedElementState state, SharedElementMessage *message)	Конструктор.
SharedElement ()	Подразумевани конструктор.
virtual ~SharedElement ()	Подразумевани деструктор.
eSharedElementState getState ()	Преузима тренутно стање за SharedElement.
void setState (eSharedElementState state)	Поставља ново стање за SharedElement.
SharedElementMessage * getMessage ()	Преузима SharedElement поруку.
bool setMessage (SharedElementMessage *message)	Поставља SharedElement поруку.

Табела 4.5 SharedElement

Screen Manager је модул за контролу приказа графичких елемената на екрану као што су видео и слика и описан је у ScreenManager.hpp (Табела 4.6).

Функције	Опис
ScreenManager()	Подразумевани конструктор.
virtual ~ScreenManager()	Подразумевани деструктор.
bool showVideo(int x, int y, int width, int height, eSourceID source, eDestinationID destination)	Приказује видео из source на destination.
bool showVideo(int x, int y, int width, int height, char *buffer, eDestinationID destination)	Приказује видео из buffer на destination.
bool hideVideo(eDestinationID destination)	Сакрива видео на destination.
bool showImage(int x, int y, int width, int height, char *buffer, eDestinationID destination)	Приказује слику из buffer на destination.
bool hideImage(eDestinationID destination)	Сакрива слику на destination.

Табела 4.6 ScreenManager

Car Simulation Manager је модул који симулира неке параметре вожње читајући их из базе података а спрега је описана у CarSimulationManager.hpp (Табела 4.7).

Функције	Опис
CarSimulationManager ()	Подразумевани конструктор.
~CarSimulationManager ()	Подразумевани деструктор.
bool start ()	Покреће нит која чита из базе података.
bool stop ()	Зауставља нит која чита из базе података.
HMIMWLMessage * getNextMessage ()	Преузима следећу симулациону поруку из реда.

Табела 4.7 CarSimulationManager

HMI Manager је главна контролна јединица за улазне уређаје односно ХМИ интерфејсе и описан је у HMIManager.hpp (Табела 4.8).

Функције	Опис
HMIManager ()	Подразумевани конструктор.
~HMIManager ()	Подразумевани деструктор.
bool start ()	Покреће све интерфејсе који се користе.
bool stop ()	Зауставља све интерфејсе који се користе
HMIMWLMessage * getNextMessage ()	Преузима следећу поруку из реда.
void callback (HMIMWLMessage *message)	Повратни позив који окидају интерфејси када је следећа порука спремна.
void processMessageHMIFeedback (HMIFeedbackMessage *hmiFeedbackMessage)	Обрађује поруке других делова система који захтевају акцију од улазних уређаја односно повратну информацију за корисника.

Табела 4.8 HMIManager

Adas Manager је главниконтролни модул за комуникацију система са ADAS плочом. Контрола се врши преко етернет мреже, исто као и пријем обрађеног видео тока који се складишти у циркуларном баферу. Слике извучене из видео тока једног од доступних ADAS алгоритама достављају се апликацији путем повратног позива која их затим приказује на екрану. Апликативна спрега описана је у ADASManager.hpp (Табела 4.9).

Функције	Опис
ADASManager (image_ready_callback clbk, void *userData)	Конструктор.
~ADASManager ()	Подразумевани деструктор.
bool start ()	Покреће ADASManager.
bool stop ()	Зауставља ADASManager.
bool executeCommand (ADASMessage *adasMessage)	Извршава контролне команде за покретање и заустављање алгоритама.
bool multicastCallback (char *message)	Повратни позив који се окида када стигне нова статусна порука са ADAS плоче.

Табела 4.9 ADASManager

HMIMWLMessage је базна класа за поруке и све врсте порука је наслеђују. Описана је у HMIMWLMessage.hpp (Табела 4.10).

Функције	Опис
HMIMWLMessage (eHMIMWLMessageType type, eHMIMWLMessageDestination destination)	Конструктор.
HMIMWLMessage ()	Подразумевани конструктор.
virtual ~HMIMWLMessage ()	Виртуелни деструктор.
eHMIMWLMessageType getType ()	Преузима тип поруке.
eHMIMWLMessageDestination getDestination ()	Преузима одредиште поруке.
void setDestination (eHMIMWLMessageDestination destination)	Поставља одредиште поруке.
virtual HMIMWLBytes * getBytes ()=0	Виртуелна метода за преузимање садржаја поруке.
static bool unpack (unsigned char *bytes, int size, std::queue< HMIMWLMessage * > *hmimwlMessages)	Виртуелна метода за распакивање садржаја поруке.

Табела 4.10 HMIMWLMessage

5. Испитивање и резултати

За лакше приказивање направљеног виртуелизованог окружења за извршавање апликација у возилу користи се демонстратор посебно израђен за ову намену (Слика 5.1). Уређаји који возачу иначе нису видљиви, сакривени су испод хаубе, а екрани и уређаји за контролу система постављени су на логична места.

Слика 5.1 Демонстратор

5.1 Испитивање система

Испитивања су вршена тако што је систем пуштан да ради око 60 секунди уз мерење заузетости системских ресурса. За то време праћено је процентуално оптерећење централног процесора као и радне меморије. Посматрани су модули средњег слоја софтвера појединачно и систем у целини. У Табели 5.2 приказани су просечни резултати мерења заузетости процесора и радне меморије.

	Централни процесор (QNX) [%]	Радна меморија (QNX) [MB]	Централни процесор (Линукс) [%]	Радна меморија (Линукс) [MB]
QNX без Хипервизора и Линукса	6.75	3306	-	-
Само средњи слој софтвера	9	3309	-	-
QNX са Хипервизором и Линуксом	28	3323	6	80
Цео систем у мировању	28.75	3744	10.9	205
Цео систем оптерећен	29,5	3744	63.1	216

Табела 5.1 Резултати мерења

Систем ради уживо, уз мало зазузеће процесорског времена. Сам QNX хипервизор без осталог софтвера пријављује заузетост процесора од 6,75% и 3306MB радне меморије. Висока заузетост радне меморије у мировању је последица тога што хипервизор резервише меморију за друге системе. Када је средњи слој софтвера покренут, заузето је 9% процесора и 3309MB радне меморије. Након покретања AGL виртуелне машине заузето је 28% процесора и 3323MB радне меморије на QNX-у и 6% процесора и 80MB радне меморије на Линуксу. Цео систем са апликацијама у мировању заузима 28,75% процесора и 3744MB радне меморије на QNX-у и 10,9% процесора и 205MB радне меморије на Линуксу. Цео систем са апликацијама приликом пуштања видеа заузима 29,5% процесора и 3744MB радне меморије на QNX-у и 63,1% процесора и 216MB радне меморије на Линуксу.

Мерено је време графичког одзива на команду отаварања мултимедијалног менија односно пуштања песме. Систем је сниман камером тако да се у истом кадру виде сва три екрана. Током обраде снимка израчунато је време које протекне од додира екрана инфозабавног система до појављивања слике албума на екрану испред возача. У хипервизорском режиму рада односно са инфозабавним системом у виртуелној машини време одзива је око 0,7s док је у случају када инфозабавни систем ради на независној платформи око 0,58s. Обе вредности су задовољавајуће јер се слика албума прикаже пре него што крене песма.

Мерен је број слика које апликација исцрта у секунди (енг. FPS – Frames per second). У хипервизорском режиму рада тај број се креће око 60 док је у случају када инфозабавни систем ради на независној платформи око 90. Обе вредности задовољавају критеријуме течне анимације за људско око.

5.2 Дискусија резултата

Постигнут је очекивани визуелни утисак са три екрана. Постоје ограничења хипервизора односно оперативних система још увек непотпуно прилагођених развојној платформи. Из тог разлога нису биле могуће комплексније операције дељења графичких елемената. С обзиром на моћ коришћеног система на чипу, очекивано је да ће без проблема покретати два оперативна система и графички захтевне апликације са излазом на више екрана. Средњи слој софтвера заузима 3MB радне меморије и 2,25% процесорског времена односно не утиче значајно на перформансе остатка система. Виртуелна машина QNX-у одузме 21,25% процесора и 17MB радне меморије. То значи да графичким апликацијама односно мултимедији остаје више од 70% процесорског времена.

Графичка апликација на Линуксу показала се као најзахтевнија зато што се за пуштање видеа користи централни процесор уместо хардверског декодера. Поређењем резултата добијених када је инфозабавни систем покренут у виртуелној машини и када је на посебној платформи уочавамо да је одзив у хипервизорском режиму око 20% спорији, међутим, разлика од десетог дела секунде за корисника инфозабавног система не игра значајну улогу.

На основу мерења можемо да закључимо да хипервизор не уноси значајно кашњење у односу на систем на две посебне развојне платформе као и да постоји још простора за проширење система. Уз очекиване нове верзије оперативних система који ће омогућити додатно оптимизовано дељење графичких ресурса могућности ће бити додатно проширене.

6. Закључак

Задатак рада је био да се изради виртуелизовано окружење за извршавање апликација са приказом на екранима унутар аутомобила. Циљеви су били и изолација апликација различите безбедносне критичности извршавањем на погодним оперативним системима као и коришћење једне хардверске платформе за излаз на више екрана.

На основу мерења можемо да закључимо да је хардверска платформа довољно брза за предвиђене намене као и да могуће додатно проширење и оптерећење система. QNX са хипервизорским проширењем погодан је за извршавање безбедносно критичних апликација док је виртуелни AGL погодан за брз развој безбедносно некритичних апликација на високом нивоу апстракције.

Даљи развој укључивао би напредније коришћење могућности дељења графичке јединице као и коришћење осталих аутомобилских магистрала. Израда окружења са Андроидом у виртуелној машини убрзала би развој графичких апликација коришћењем добро познатих алата који се користе у потрошачкој електроници и омогућила коришћење већ постојећих. У смислу спровођења процеса развоја софтвера у аутомобилској индустрији могуће је додатно анализирање и тестирање кода односно детаљније вођење документације за безбедносно критични део софтвера. Увођење додатне инфраструктуре за аутоматско превођење кода и спуштање на платформу убрзало би и олакшало развој.

7. Литература

[1] Joakim Janjatovic, Radivoje Ostojic, Gordana Velikic, Tomislav Maruna “Algorithm optimization framework for Advanced Driver Assistance Systems”, Consumer Electronics (ICCE), 2017 IEEE International Conference on, 8-10 Jan. 2017

[2] Three emerging trends in automotive engineering https://www.asme.org/engineering-topics/articles/automotive/3-emerging-trends-automotive-engineering [Доступно 09.11.2016.]

[3] Mazda Cosmo https://en.wikipedia.org/wiki/Mazda_Cosmo [Доступно 09.11.2016.]

[4] Egil Juliussen, “Connected Cars: Perspectives to 2025”, IHS Automotive Technology, April 27, 2016

[5] Miroslaw Staron, “Automotive Software Architectures, An Introduction”, Springer International Publishing, 2017

[6] AUTOSAR and ISO26262: A new approach to vehicle network design and automotive safety http://www.eenewsautomotive.com/content/autosar-and-iso26262-new-approach-vehicle-network-design-and-automotive-safety/page/0/1 [Доступно 09.11.2016.]

[7] VDA QMC Working Group 13 / Automotive SIG, “Automotive SPICE Process Assessment / Reference Model 3.0”, 2015-07-16

[8] Rafael V. Aroca, Glauco Caurin, “A Real Time Operating Systems (RTOS) Comparison”, 2017

[9] Yi Zheng, “Architectures for ISO 26262 systems with multiple ASIL requirements”, QNX, September 2014

[10] Joao Filipe Marques Serra, “Multi-criticality Hypervisor for Automotive Domain”, Master of Science in Electrical and Computer Engineering, Universidade de Coimbra, July 2014

[11] “DATASHEET COQOS SDK v9.0 on Renesas R-Car H3”, OpenSynergy, June, 2017

[12] “QNX Hypervisor, PRODUCT BRIEF”, BlackBerry QNX, 2017

[13] Radivoje Ostojic, Jasmina Pesic, Milan Z. Bjelica, Goran Stupar, “Java-based graphical user interface framework for In-Vehicle Infotainment units with WebGL support”, Consumer Electronics - Berlin (ICCE-Berlin), 2016 IEEE 6th International Conference on, 5-7 Sept. 2016

[14] Jasmina Pešić, Kristina Omerović, Ivana Nikolić, Milan Z. Bjelica, “Automotive cluster graphics: Current approaches and possibilities”, Consumer Electronics - Berlin (ICCE-Berlin), 2016 IEEE 6th International Conference on, 5-7 Sept. 2016

[15] Ivana Nikolic, Milos Ilic, Nikola Popovic, Milena Milosevic, “Application environment for browser-based in-vehicle”, Consumer Electronics (ICCE), 2017 IEEE International Conference on, 8-10 Jan. 2017

[16] Kristina Omerovic, Joakim Janjatovic, Milena Milosevic, Tomislav Maruna, “Supporting sensor fusion in next generation android In-Vehicle Infotainment units”, Consumer Electronics - Berlin (ICCE-Berlin), 2016 IEEE 6th International Conference on, 5-7 Sept. 2016