МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.

«___» _____2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТР"

Тема: Система контролю версій автомобільних мультимедійних систем

Виконавець: Гуляєв А.О.

Керівник: Артамонов Є.Б.

Нормоконтролер: Тупота Є.В.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Освітнього ступеня магістр
Спеціальність 123 "Комп'ютерна інженерія"
(шифр, найменування)

Спеціалізація 123.02 "Системне програмування"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Литвиненко О. Є.

«____»_____2020 р.

ЗАВДАННЯ
на виконання дипломної роботи (проекту)

Гуляєва Антона Олександровича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема роботи: "Система контролю версій автомобільних мультимедійних систем"
затверджена наказом ректора від " 27 " серпня 2020 року № 1203 /ст.
2. Термін виконання роботи: з 05.10.2020 до 31.12.2020
3. Вихідні дані до роботи: автомобільне програмне забезпечення
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) дослідження стану проблеми;
2) структура репозиторіїв програмного забезпечення для автомобільних операційних систем;
3) розробка програмного забезпечення для контролю версій програмного забезпечення автомобільних мільтимедійних систем
5. Перелік обов'язкового графічного матеріалу:
1) архітектура системи платформи *IVI*;
2) взаємодія компонентів мультимедійної системи;
3) зміст компонентів мультимедійної системи;
4) зміни регіональних налаштувань у файлі *eeprom*;
5) схема алгоритму фільтрації за назвою ПЗ;
6) схема алгоритму обробки запиту на пошук версій ПЗ

6. Календарний план

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1	Ознайомитись з постановкою задачі дипломного проектування	05.10.2020 – 06.10.2020	
2	Вивчити спеціальну літературу і технічну документацію автомобільних операційгих систем	07.10.2020 – 10.11.2020	
3	Провести аналіз методів побудови і модифікації автомобільних програмних систем	11.11.2020 – 14.11.2020	
4	Написати і налагодити програму	15.11.2020 – 19.11.2020	
5	Написати пояснювальну записку	29.11.2020 – 13.12.2020	
6	Підготовити графічний і демонстраційний матеріали	21.12.2020 – 25.12.2020	

7. Дата видачі завдання 05.10.2020

Керівник Артамонов Є.Б.
(підпис)

Завдання прийняв до виконання Гуляєв А.О.
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Система контролю версій автомобільних мультимедійних систем”: 80 с., 12 рис., 23 літературних джерел, 1 додаток.

ОПЕРАЦІЙНА СИСТЕМА, ІНФОРМАЦІЙНО-РОЗВАЖАЛЬНИЙ КОМПЛЕКС АВТОМОБІЛЯ, ПРОШИВКА, ПРОГРАМНИЙ РЕПОЗИТОРІЙ

Мета дослідження – розробити систему контролю версій автомобільних мультимедійних систем.

Об’єкт дослідження – автомобільне програмне забезпечення.

Предмет дослідження – систему контролю версій автомобільних мультимедійних систем.

Встановлено, що на основі результатів запропонованого методу аналізу версій встановленого автомобільного ПЗ є можливість проводити пошук оновлень у хмарній базі репозиторію ПЗ автомобільних мультимедійних систем.

Результати дипломної роботи рекомендується використовувати при розробці систем контролю і оновлення версій вбудованого програмного забезпечення. Розвиток проекту передбачається за рахунок введення модуля віддаленого управління оновленням версій ПЗ.

Результати дипломної роботи було представлено на науково-практичній конференції та опубліковані у збірнику тез доповідей:

Гуляєв А.О. Система контролю версій автомобільних мультимедійних систем / Гуляєв А.О. // Тези доповідей наук.-практ. конф. “Сучасні тенденції розвитку системного програмування” (25-26 листопада 2020 р.). – К.: НАУ, 2020. – С. 7.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

IVI – *In-Vehicle Infotainment* (Інформаційно-розважальний комплекс автомобіля)

ГП – граничний пристрій

РЛ – реверсивний лічильник;

ЦЛЗ – цифрова лінія затримки

ВСТУП

Вже майже настав час коли електронні системи автомобілів будуть через вбудовану камеру визначати водіїв та пасажирів, сидіння автоматично регулюватися, щоб максимально зручно вписатися в пасажирів, навігатор та інформаційно-розважальною системою буде перемикатися на програми відповідно до уподобань пасажирів та синхронізуватись з смартфонами та планшетами, завантажуючи музику чи телевізійні шоу через портативні пристрої, і все це не буде вимагати торкатися моніторів на портативних пристроях.

Автомобільне обладнання вже може керувати програмами на смарт-телефонах, такими як вибір дзвінків, перегляд відео чи прослуховування музики. Перебуваючи в дорозі, навігатор надає звіт про стан дорожнього руху в режимі реального часу та інструкції та контролює стан транспортного засобу для управління ризиками. Автомобіль шукає найближчу (чи навіть найдешевшу) заправку, коли бензин або акумулятор загинуть. На сьогодні впроваджені практично всі функції автомобіля, які раніше були лише у проектах, тепер час за навчанням розумних систем і налагодженням програмного забезпечення автомобілів.

Першою технологічною революцією в автомобілебудуванні став інтерес автомобільних компаній до електричних стартерів - їх вперше встановили в 1911 році. Потім нововведення стали стосуватися зручності водія і навіть його розваг за кермом: в 1925 році з'явився прикурювач, в 1930 - радіо, в 1956 - підсилювач керма, в 1970 - касета в 1984 - надувні подушки безпеки. Роком пізніше - програвачі компакт-дисків, в 1994 році - панель приладів комп'ютерної діагностики автомобіля, в 1995 - *GPS*, в 2000 - *USB* і *Bluetooth*, перші ластівки «підключеного» до всього автомобіля.

Перший досвід створення розумної машини стався в середині XX століття. *General Motors Firebird II* - чотиримісний автомобіль 1956 року зі незалежною підвіскою. Під титановим корпусом переховувався газотурбінний двигун *Whirlfire GT*-304 на 200 к.с., електропакет і інтегрована система кондиціонування

повітря рівня не гірше, ніж на початку *XXI* століття. *Firebird II* в плані дизайну і ергономіки продовжив версію автомобіля 1953, який був названий «реактивним літаком на колесах» (розробники та інженери, дійсно, надихалися концептами винищувачів того часу). Однак в *Firebird II* вперше була застосована структура для поїздок по шосе майбутнього - складна система управління, яка повинна була взаємодіяти з електричним дротом, вбудованим в проїжджу частину, щоб посилати сигнали і служити орієнтиром для новітніх автомобілів. Передбачалося, що електромагнітне поле мінімізує небезпечні ситуації на дорозі, скоротивши людський фактор. На ті часи це була дуже сміліва модель, яка викликала фурор на виставках, але так і не потрапила в серійне виробництво.

Шосе майбутнього будувалися в Європі і США. Першим серійним автомобілем, який став реально з ними взаємодіяти, був *Citroen DS* - легендарна легковик, яка посіла третє місце в рейтингу автомобілів століття. Малопотужний двигун 75 к.с. нічим не виділявся в ті часи, але зате автомобіль відрізняла передова трансмісія, об'єднана з рульовим керуванням, гальмами і гідропневматичною підвіскою. Така конструкція випередила розвиток автомобілебудування на багато років вперед. *Citroen DS* вмів взаємодіяти з шосе за допомогою електричного сигналу, проте ні про яке самостійному автопілоті зовсім не йшлося - це була більше забава. До речі, саме неймовірна популярність, передові технології і нехай і щодо ілюзорний, але автопілот зробили цей *Citroen* літаючим автомобілем Фантомаса.

Експерименти з бортовими комп'ютерами в 60-70 рр. проводились, але так і не увійшли в серію. Варто згадати експериментальний *Chrysler Plymouth*, який оснастили бортовим комп'ютером (ну, наскільки можна назвати бортовим комп'ютер, який займав половину заднього сидіння) і генератором для живлення системи, виведеним на дах автомобіля. Лабораторні випробування проводилися протягом 10 років, але ні про яку серійності виробництва не могло бути й мови.

Перші електронні системи з'явилися в автомобілях ще в 60-х роках, і завдяки цьому галузь серйозно змінилася - сьогодні електроніка, і особливо програмне забезпечення, є основними джерелами інновацій. Програмне забезпечення підвищує надійність за допомогою систем активної і пасивної

безпеки, таких як антиблокувальна гальмівна система і електронна система курсової стійкості (*ESC*). Крім того, сьогодні відбувається поступова інтеграція побутової електроніки в автомобілі.

Програмне забезпечення для автомобілів дуже надійно - рівень відмов становить не більше одного збою на мільйон операцій в рік. Більшість людей навіть не уявляють, наскільки багато автомобільних функцій управляються сьогодні програмно, проте навряд чи вам доводилося коли-небудь чути про блакитному екрані в автомобілі, хоча для ПК це звичайна справа.

Інформаційно-розважальна програма у традиційному розумінні – це засоби масової інформації, які передають споживачам інформативний та розважальний контент. Побутова електроніка, така як планшети та смартфони, змінила ландшафт інформаційно-розважального комплексу, пропонуючи платформу, яка забезпечує величезний обсяг персоніфікованих даних, будь то ціни на акції, інформація про погоду або останні меми. Автомобільна промисловість швидко рухається до адаптації цієї концепції до сучасних транспортних засобів. Інформаційно-розважальна система в автомобілі означає сучасну пропозицію автомобілів (зробіть глибокий вдих): підключені пристосовані до власного пристрою сумісні настроювані бездротові найсучасніші голосово-активовані сенсорні екрани з голосовою активацією, інтегровані мультимедійні системи,

Сьогодні на частку електроніки припадає близько 80% функціональних інновацій автомобільної галузі, і програмне забезпечення - це ключ до більшості з них. У міру того як ПЗ стає все більш істотною частиною вартості обладнання, в бізнес-моделях починають враховувати необхідність повторного використання та обміну програмним забезпеченням.

Високошвидкісні шини, такі як *Ethernet,* все ширше використовуються сьогодні в автомобілебудуванні для підтримки взаємодії між *ECU* і розробки нових функцій, особливо в області безпеки. Інформація з різних джерел аналізується і консолідується для формування повної моделі середовища, дозволяючи розробляти нові функції, які підтримують водія в критичних ситуаціях. Наприклад, якщо увагу водія відволікає пасажир, то програма може визначити, що їде попереду гальмує, і попередити про це водія або ж автономно

включити гальмування. Водій ніколи не здогадається про існування такого програмного забезпечення, поки не виникне небезпечна ситуація.

В автомобілебудуванні сьогодні назріла чергова програмна революція - все ширше починають застосовуватися засоби мультимедіа та побутової електроніки. Автомобілі будуть підключатися до Інтернету і до всіх видів мобільних і встановлених будинку пристроїв, причому неухильно зростатиме частка рішень на базі вільного програмного забезпечення. Кожного дня з'являються нові версії ПЗ для автомобільних систем, а фахівці, що працюють з налагодженням електроніки змушені ставати фахівцями з встановлення і налаштування ПЗ. Саме тому сьогодні проблема відстеження версій і оновлень автомобільного ПЗ є доволі актуальною задачею.

Мета дослідження – розробити систему контролю версій автомобільних мультимедійних систем.

Об'єкт дослідження – автомобільне програмне забезпечення.

Предмет дослідження – систему контролю версій автомобільних мультимедійних систем.

Встановлено, що на основі результатів запропонованого методу аналізу версій встановленого автомобільного ПЗ є можливість проводити пошук оновлень у хмарній базі репозиторію ПЗ автомобільних мультимедійних систем.

Результати дипломної роботи рекомендується використовувати при розробці систем контролю і оновлення версій вбудованого програмного забезпечення. Розвиток проекту передбачається за рахунок введення модуля віддаленого управління оновленням версій ПЗ.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ СТАНУ ПРОБЛЕМИ

1.1. Аналіз принципів використання сучасних автомобільних мультимедійних систем

Автомобільна мультимедійна система або система *IVI* (інформаційно-розважальний комплекс автомобіля (англ. *In-Vehicle Infotainment*)), як є частіше називають у закордонних публікаціяї, очевидно, є абсолютно новою практикою використання автомобільних комп'ютерів, завдяки чому перевезення не просто пересуваються з одного місця в інше. Він несе різноманітну нероздільну інформацію.

Багато відомих автомобільних компаній не тільки усвідомлюють, що система *IVI* стане ключовим моментом у майбутньому, але також розуміють особливості спеціальностей, а дружнє використання може бути привабливістю для диференціації від конкурентів (рис. 1.1).

Category	Car Audio	Car Navigation	Infotainment
Photo			
Broadcasting	FM/AM	FM/AM/HD Radio	FM/AM/HD Radio
Music	Tape/CD	CD/DVD/External Media Player	DVD/External Media Player/On-Line Service
Navigation	No	Yes (Off-line)	Yes (On-line)
Touch Support	No	Yes	Yes
Communication	No	No	Yes
Mobile Support	No	No	Yes

Рис. 1.1. Історія системи *IVI*

На відміну від минулого, коли водії могли слухати лише радіо, музику та дивитися фільми, нові креативні інновації змусять смартфони та інші мобільні пристрої підключатися до автомобілів, а це означає, що інтеграція технологій є

великим викликом. За даними *Juniper Research,* у 2025 році буде 90 мільйонів автомобілів, які зможуть виходити до Інтернет та зможуть працювати з програмами на смартфонах.

Згідно з даними *IHS Automotive Research,* опублікованим у березні 2018 р., Думка користувача щодо систем *IVI* переключиться з несуттєвих на кашові. За підрахунками, до 2025 року дохід напівпровідника для *IVI* сягне понад 8500 мільйонів доларів США. Китай, один з найважливіших ринків для системи *IVI,* займе більше половини світового доходу, близько 5300 мільйонів доларів США самостійно. У звіті *IHS* у квітні 2018 року передбачається, що дохід від дротового та бездротового Інтернету та технологій підключення, пов'язаних з напівпровідником для використання автомобілів, зросте з 430 до 540 мільйонів доларів США. Більше того, завдяки зростанню IBC суміжні галузі очікують від нього опосередкованої вигоди. Наприклад, у 2017 році було продано лише 2% автомобілів, які були обладнані головним дисплеєм (*HUD*), але це, ймовірно, зросте до 9%, що було б близько 9 мільйонів,

Є дві основні причини, за якими набирається надлишок систем *IVI*:

1. Обсяг продажів автомобілів, обладнаних системами *IVI,* до 2025 року, як очікується, буде вдвічі більшим, ніж зараз.

2. У 2025 році середній дохід однієї системи *IVI* становитиме більше 200 доларів США. Зіткнувшись з цим гігантським ринком, пов'язані компанії повинні оновити новітні технології та підготувати бездоганні товари, щоб конкурувати з конкурентами.

Коли автомобіль пропонує не просто транспортні послуги, а через *IVI* як носій для спілкування із змінними електронними пристроями як продукт штучного інтелекту нової ери, важливішим є сертифікація його тесту на сумісність та сумісність (*IOP*) відповідного обладнання та програмне забезпечення, щоб переконатися, що функціональність, якість та ефективність достатньо хороші, щоб вирушити в дорогу.

Для того, щоб усі ці програми, згадані вище, мали найкращу якість, цей стандарт слід застосовувати суворо, щоб переконатися, що технологічний протокол між смартфонами, програмами, інформаційним вмістом, системами *IVI*

та підключенням до Інтернету може працювати в цілому. Таким чином, користувачі можуть ефективно спілкуватися з автомобілями за допомогою смартфонів.

Інформаційно-розважальна система в машині (*IVI*) – це термін з автомобільної промисловості, який відноситься до транспортних систем, що поєднують доставку інформації з доставкою розваг пасажирам та водіям. Для надання та управління такими послугами системи *IVI* обладнані інтерфейсом людини-машини (*HMI*), що складається з аудіо / відео інтерфейсів, клавіатур, сенсорних екранів тощо.

Додаткові інструменти та функції, характерні для цих систем, включають відтворення аудіо / відео та двосторонні засоби зв'язку (*CD* / радіо, навігація, голосові команди, розваги на задніх сидіннях тощо). Крім того, зв’язок мобільних пристроїв в останні роки став головним компонентом систем *IVI*, намагаючись задовольнити зростаючі потреби у доступі до Інтернету та загального вмісту смартфонів, таких як потокове передавання музики, інформація про дорожній рух та прогнози погоди.

Система *IVI* складається з безлічі взаємопов’язаних апаратних та програмних компонентів, а загальну архітектуру або структуру довільної системи можна описати набором шарів, починаючи від апаратного рівня внизу і закінчуючи *HMI* вгорі. Між кожним рівнем чітко визначені інтерфейси, і кожен шар включає ряд ключових технологічних блоків у програмному та / або апаратному забезпеченні для забезпечення бажаних функціональних можливостей системи (рис. 1.2).

Нижче наводиться короткий опис кожного шару та їх основних будівельних блоків [16]:

– рівень інтерфейсу людина-машина (*HMI*): *HMI* – це інтерфейс для користувача системи *IVI* і керує відображенням головного блоку системи *IVI*. Він відповідає за обробку та реагування на введення користувачами, такі як введення на сенсорному екрані, вхід розпізнавання мови та вхід на основі ручки / кнопки [16];

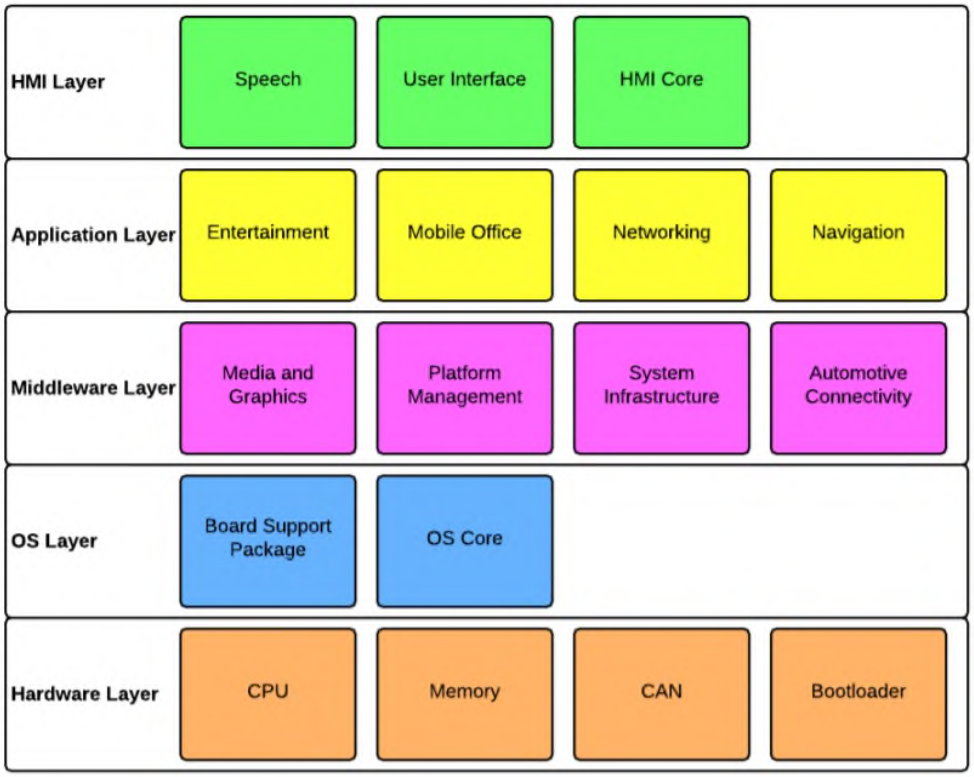

Рис. 1.2. Аналіз архітектури систем платформи *IVI*

– рівень додатків: Рівень додатків містить сукупність програм, усі призначені для забезпечення певної функції на користь користувача. Додатки залежать від іншого програмного забезпечення, яке називається системним програмним забезпеченням, щоб його можна було виконувати. Системне програмне забезпечення відрізняється від прикладного програмного забезпечення в тому сенсі, що воно обслуговує останнє (яке, у свою чергу, обслуговує користувача) [17];

– рівень проміжного програмного забезпечення: Рівень проміжного програмного забезпечення складається з компонентів та інтерфейсів у програмному забезпеченні, яке забезпечує додатки послугами, недоступними з рівня операційної системи, щоб можна було реалізувати функціональні області прикладного рівня. Отже, рівень проміжного програмного забезпечення спрощує зв’язок та введення / виведення даних між рівнем програми та рівнем операційної системи, в результаті чого розробники програм можуть зосередитися на конкретній меті та функціональності своєї програми [18];

– рівень операційної системи (ОС): Рівень операційної системи, як правило, становить операційну систему разом із пакетом підтримки плат (*BSP*) та драйверами. Сама операційна система, як правило, управляє апаратними та програмними ресурсами та забезпечує додатки загальними послугами [12]. *BSP* забезпечує основний код підтримки, який полегшує перенесення ОС на нове апаратне середовище [11]. Драйвери працюють і керують підключеними апаратними пристроями, надаючи програмні інтерфейси апаратним пристроям. Таким чином, операційна система може отримати доступ до апаратних функцій, не знаючи жодної детальної інформації про апаратне обладнання, яке використовується в даний час [12].

– апаратний рівень: Апаратний рівень складається з процесора з додатковим необхідним обладнанням та прошивкою для завантаження ОС. Крім того, цей рівень часто оснащений набором автомобільних пристроїв вводу-виводу, таких як інтерфейси *CAN* / *MOST* [16].

Останніми роками кількість програмного забезпечення, знайденого в сучасних транспортних засобах, різко зросла, і типовий автомобіль може мати до 100 мільйонів рядків коду в своїх системах [12]. Оскільки системи *IVI* перейшли від розкоші до товару, важливість конкурентної переваги над своїми суперниками на цьому кордоні є життєво важливою. У традиційному підході до розробки системи *IVI* у виробника автомобілів, як правило, дуже мало інженерів програмного забезпечення, які беруть участь у реальній розробці. Натомість постачальники першого рівня розробляють все програмне забезпечення відповідно до набору вимог, передбачених виробником [13]. Цей підхід може запропонувати постачання дуже дорослого продукту (наприклад, вбудованого *Windows*), але тримає виробника в повній залежності від інструментів та функцій, що надаються цим рівнем.

Рішення, засноване на відкритому вихідному коді, з іншого боку, дозволяє розробникам *OEM* та першого рівня безпосередньо записувати функції у це рішення. Крім того, у відкритому програмному забезпеченні можуть бути тисячі постачальників та вкладників, що пом'якшує ризик, пов'язаний із покладанням на одного постачальника системи [24].

Багато споживачів також звикли до швидкого технологічного обороту, який виявляється в багатьох сучасних споживчих товарах (наприклад, мобільних телефонах), і очікують того ж від апаратного та програмного забезпечення, що знаходиться в їхніх автомобілях. Оскільки цикл розвитку / доставки автомобілів для автомобілів майже втричі перевищує цикл типових споживчих електронних пристроїв, це є основною проблемою для виробників. Інша річ, яку вимагає рішення з відкритим кодом, полягає в тому, що функції та програми, пов'язані з системами *IVI*, рідко є унікальними для автомобільної галузі. Навпаки, більшість зазнає сильного впливу споживчого сектору, зв’язок, який модель розробки програмного забезпечення з відкритим кодом може використати для забезпечення передачі інновацій між двома галузями [3].

В таблиці 1.1 міститься перелік основних складових автомобільних мільтимедійних систем

Таблиця 1.1

Основні компоненти автомобільних мільтимедійних систем

Категорія	Особливості	Опис
1	2	3
Інтерфейс людина-машина (*HMI*)	Кнопки керма для управління без очей Сенсорний екран Голосова активація	Все, що пов’язано з тим, що водій може бачити і торкатися, а також як водій взаємодіє з автомобілем.
Підключення	Принесіть свій власний пристрій (*BYOD*) *USB* *Bluetooth* *Wi-Fi*	Як смарт-пристрій підключається до автомобіля, чи автомобіль до хмари, чи автомобіль до інших автомобілів. Національне управління безпеки дорожнього руху Міністерства транспорту США (*DOT*) (*NHTSA*) оголосило про потенційне регулювання, яке найближчим часом передбачає встановлення зв'язку між транспортними засобами (*V2V*).
Телематика	Карти навігації та дорожнього руху Статус транспортного засобу Попередження про погоду	"Інформація" в ін9формаційно-розважальному секторі. Навігація та карти, попередження про погоду, стан автомобіля (температура масла, тиск масла, напруга акумулятора, тощо).

Закінчення табл. 1.1

1	2	3
Радіо / Аудіо / Відео	Супутникове радіо *DVD*-програвачі Інтернет-трансляція	Розваги, включаючи такі засоби масової інформації, як відео, музика та подкасти. Не включає можливості запису.
Розширені системи допомоги водію (*ADAS*)	Камери (спереду, ззаду) Чорна скринька Попередження про зіткнення Виявлення сліпих плям Радар Попередження про виїзд із смуги руху	Відволікання водія не можна запобігти, але цим можна керувати. *ADAS* включає ЕБУ (електронний блок управління), призначений виключно для реєстрації даних (чорний ящик), який можна переглянути в разі аварії, передні / задні камери для усунення сліпих зон, радар на приблизну відстань до інших транспортних засобів та система попередження про виїзд із смуги руху.

*Texas Instruments,*Аналогові пристрої,*Freescale Semiconductor*, іАльтерає провідними постачальниками у пропонуванні інноваційних продуктів, що відповідають стандартам *AEC-Q*, для автомобільних інформаційно-розважальних систем.

Для комплексного рішення автомобільної інформаційно-розважальної програми *Freescale* випустила*i.MX* 6 Серіяпроцесорів. Поєднуючи енергоефективні можливості обробки архітектури *ARM Cortex-A*9 з передовою 2*D* та 3*D* графікою, а також відео високої чіткості, одно-, дво- та чотириядерні процесори *i.MX* 6 забезпечують новий рівень продуктивності мультимедіа. *Freescale* також співпрацює з *QNX*, компанією з програмним забезпеченням операційної системи в режимі реального часу, яку нещодавно придбала компанія *Blackberry*, щоб забезпечити повне апаратне та програмне рішення на ринку інформаційно-розважальних систем.

Польові програмовані ворота (*FPGA*) унікальні тим, що вони дозволяють інженерам забезпечити майбутню роботу своєї системи за допомогою пост-виробничих модернізацій, що позбавляє потреби виготовляти нову друковану плату для кожної ітерації плати. АльтераПЛІС циклон *V*вирішити технічні

проблеми, пов’язані із збільшенням інтеграції та продуктивності, меншою потужністю та швидшим виведенням на ринок, одночасно задовольняючи вимоги до витрат. Для інформаційно-розважального програмного забезпечення *Cyclone V* підтримує безліч входів камери та інтерфейси дисплея, включає вбудований радіопроцесор *DSP* і має жорстку процесорну систему з центром на двоядерному процесорі *ARM® Cortex-A9 MPCore* ™. Завдяки багатому набору периферійних блоків, ПЛІС *Cyclone V* зменшують потужність системи, вартість та розмір плати.

Інформаційно-розважальна система, природно, містить безліч периферійних технологій, що забезпечують життєдіяльність у транспорті, з метою управління тепловою енергією, управління живленням, шиною транспортування орієнтованих на медіасистему (*MOST*), захисту від електростатичного розряду та управління вібраціями.*Bourns 'CGA-MLC* серії *ESD* протектори, наприклад, розроблені спеціально для використання в автомобільних схемах, що вимагають захисту від електростатичного розряду.

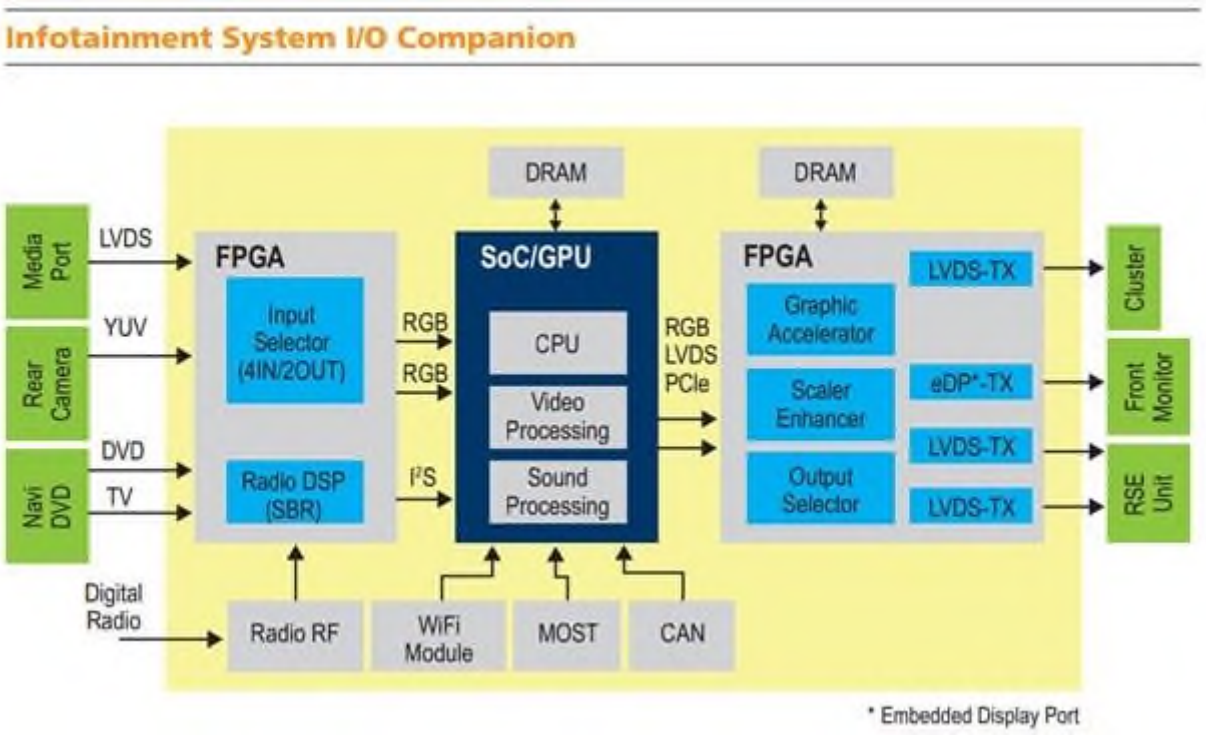

Рис. 1.3. Ведення інноваційних автомобільних рішень

1.2. Аналіз типів сертифікатів систем *IVI*

Коли система *IVI* стає все більш досконалою, охоплення її послугами також ускладнюється та розширюється. Окрім базової навігаційної системи, потокового відео, додатків та технології зберігання даних, автомобільний комп'ютер є ключовим фактором, що впливає на всю продуктивність системи *IVI*, що також є головним фактором для покупців. Нещодавно Алліон контактувала з багатьма запитами на тестування системи *IVI*, щоб переконатися, що система *IVI* здатна вільно співпрацювати з програмними та апаратними компонентами різних компаній. Тому в цій статті ми представимо три основні технології, що з'єднують різні портативні пристрої з автомобільним комп'ютером. Більше того, оскільки стандарт безпеки та запит на сертифікацію для специфікацій автомобілів набагато жорсткіші, ніж звичайні електронні вироби, наприклад, для продуктів, що зберігаються в машині, необхідно пройти механічні та фізичні випробування, щоб перевірити її стійкість на різний тиск, температуру та вологість. Тому ми також запровадимо екологічні тести для специфікацій автомобілів.

1.2.1. Стандарт сертифікації *Wi-Fi Miracast* ™

На основі техніки *Wi-Fi Display*, *Wi-Fi Alliance* встановлює технологію *Wi-Fi Miracast* ™ та запускає програму сертифікації *Wi-Fi CERTIFIED Miracast* ™. Пристрої, які мають сертифікат *Wi-Fi* СЕРТИФІКОВАНИЙ *Miracast* ™, в основному забезпечують сумісність, сумісність та безпеку продукту *Wi-Fi CERTIFIED Miracast* ™. За допомогою системи *IVI* користувач може переглядати кліпи або слухати музику зі смартфонів, насолоджуючись вмістом аудіо / відео в режимі реального часу. Як і на (рис. 1.4), пристрої, які використовують цю техніку, можуть з'єднуватися між собою без необхідності додаткового *Wi-Fi* або роз'ємів. Іншими словами, пристрій, який сертифікований *Wi-Fi CERTIFIED Miracast* ™, має можливість *Wi-Fi* для прямого підключення і не потрібно використовувати існуючу точку доступу.

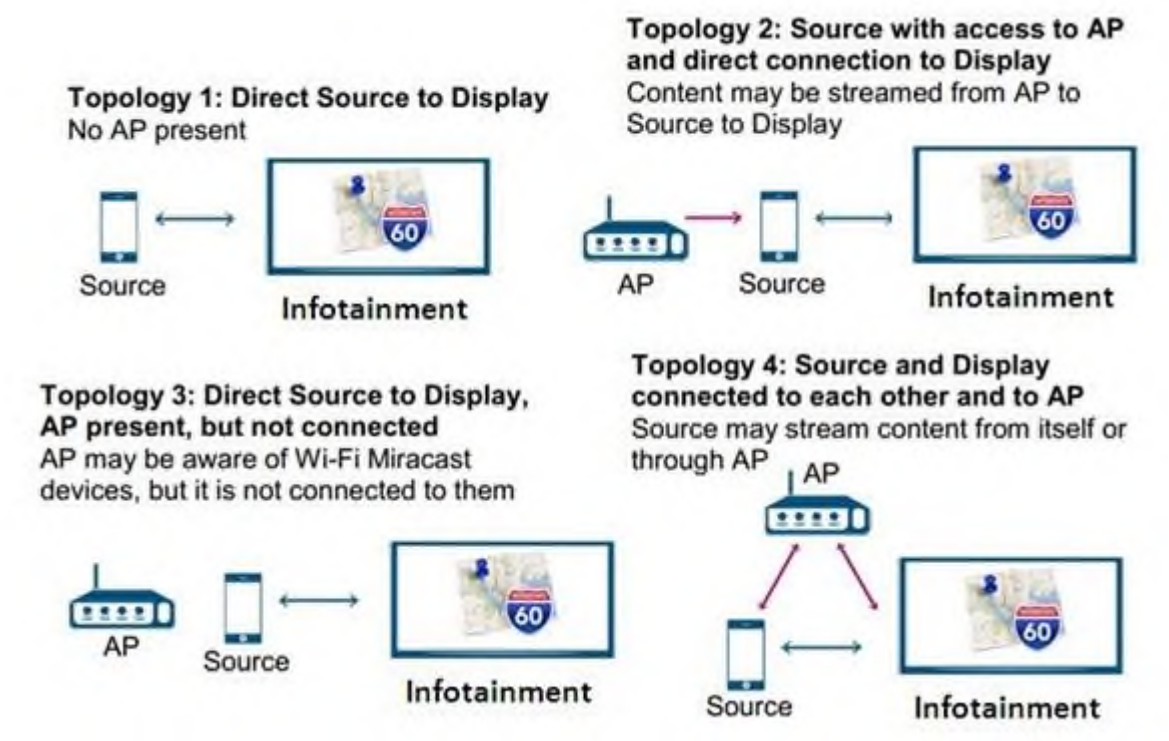

Рис. 1.4. Топологія *Miracast* ™

За даною сертифікацією користувач може вільно відображати фотографії, фільми або послугу потокового передавання аудіо-візуальних медіа через смартфон на моніторі автомобіля.

Згідно з *Wi-Fi Alliance®*, "*Wi-Fi CERTIFIED Miracast* ™ використовує безліч відомих та надійних методів від *Wi-Fi*, включаючи *Wi-Fi CERTIFIED n* (покращена пропускна здатність та покриття), *Wi-Fi Direct* ™ (пристрій до пристрою підключення), *Wi-Fi Protected Access*®2 (*WPA*2 ™, захист / захист), *Wi-Fi Multimedia* ™ (*WMM®*, управління дорожнім рухом)) та *Protected Setup* ™ тощо. " В результаті, оскільки програма сертифікації *Wi-Fi CERTIFIED Miracast* ™ базується на технічних стандартах, як зазначено вище, коли компанії хочуть, щоб їхні продукти *Wi-Fi* отримали сертифікат *Wi-Fi CERTIFIED Miracast* ™, їх продукція повинна пройти *Wi-Fi* СЕРТИФІКОВАНО *n*, сертифікати *Wi-Fi Protected Access*®2 та *Wi-Fi Direct* ™.

1.2.2. Стандарт сертифікації *Bluetooth*

Згідно зі статистичними даними *ABI Research*, до 2025 року було б 60 мільйонів автомобілів, оснащених технологією *Bluetooth*, що має темпи зростання ринку близько 47% порівняно з 2018 роком. Технологія *Bluetooth* впроваджується *Bluetooth SIG* для цілей рук -безкоштовне спілкування, а також це одна з найпопулярніших технологій стандарту бездротового зв'язку на короткий діапазон. Через *Bluetooth* контактна інформація мобільного телефону синхронізується з системою *IVI*, що дозволяє користувачам проводити телефонні дзвінки та текстові повідомлення в режимі "вільні руки". Користувачі також можуть користуватися Інтернетом на більшому моніторі *IVI* завдяки можливості підключення до Інтернету з мобільного телефону. Для енергійного ринку смарт-телефонів, компанії запускають нові продукти кожні шість місяців або щороку та оновлюють операційну систему (ОС) кожні три-шість місяців, тоді як на розробку системи *IVI* потрібно близько трьох років, а її життєвий цикл на ринку триває десять років. Таким чином, компанії повинні переконатися у сумісності між портативними пристроями та системами *IVI*, особливо у взаємозв'язку між різними версіями у випадку виникнення проблем сумісності, які можуть спричинити більш високу плату за відшкодування витрат та скарги клієнтів. Наприклад, загальні проблеми сумісності між мобільними телефонами та автомобілями наведені нижче: тристоронній зв’язок, голосове управління, контактна інформація, повторний набір з історії тощо. Вони можуть виникати внаслідок різних версій *Bluetooth* та різних підтримуючих профілів, які можуть вплинути на намір покупця. Таким чином, компанії повинні переконатися у сумісності між портативними пристроями та системами *IVI*, особливо у взаємозв'язку між різними версіями у випадку виникнення проблем сумісності, які можуть спричинити більш високу плату за відшкодування витрат та скарги споживачів. Наприклад, загальні проблеми сумісності між мобільними телефонами та автомобілями такі: тристоронній зв’язок, голосовий контроль, контактна інформація, повторний набір з історії тощо. Вони можуть виникати внаслідок різних версій *Bluetooth* та різних підтримуючих профілів, що може

вплинути на намір покупця. Отже, компаніям необхідно переконатися у сумісності між портативними пристроями та системами *IVI*, особливо у взаємозв'язку між різними версіями на випадок проблем із сумісністю, які можуть спричинити більш високу плату за відшкодування витрат та скарги споживачів. Наприклад, загальні проблеми сумісності між мобільними телефонами та автомобілями такі: тристоронній зв’язок, голосове управління, контактна інформація, повторний набір з історії тощо. Вони можуть виникати внаслідок різних версій *Bluetooth* та різних підтримуючих профілів, що може вплинути на намір покупця. Поширені проблеми сумісності між мобільними телефонами та автомобілями, як показано нижче: тристоронній зв’язок, голосовий контроль, контактна інформація, повторний набір з історії тощо. Вони можуть виникати внаслідок різних версій *Bluetooth* та різних підтримуючих профілів, що може вплинути на намір покупця. Загальні проблеми сумісності між мобільними телефонами та автомобілями: Вони можуть виникати внаслідок різних версій *Bluetooth* та різних підтримуючих профілів, що може вплинути на намір покупця.

Коли виробники розробляють продукцію з технологією *Bluetooth*, окрім тесту на відповідність кваліфікації *Bluetooth*, перевірка сумісності з іншими пристроями є одним з найважливіших факторів, що породжує технологію *Bluetooth*. Тому необхідно пройти ціле тестове рішення, таке як тест аналізу з камерами, ноутбуками та смартфонами. *Allion* має найновіші пристрої та оснащений найсучаснішими інструментами для проведення професійного тесту *Bluetooth*, допомагаючи продуктам компаній безперебійно функціонувати із програмним або апаратним забезпеченням різних марок та забезпечуючи цілісність технології *Bluetooth*.

1.2.3. Стандарт сертифікації *MirrorLink* ™

Консорціум підключення до автомобілів (*CCC*) був заснований різними компаніями різних галузей, до яких належать відомі виробники мобільних телефонів *Nokia*, *Samsung* та *LG Electronics*; виробники транспортних засобів *Daimler*, *General Motors*, *Honda*, *Hyundai Motor Company*, *Toyota* та *Volkswagen*;

та постачальники систем *Alpine* та *Panasonic*. *MirrorLink* ™ – це стандарт логотипу, який *CCC* запровадив для передачі даних в автомобілі. *MirrorLink* ™ – це відкритий і загальноприйнятий технічний стандарт для підключення до системи *IVI* та відповідних мобільних пристроїв, що пропонує користувачам найпростіший і прямий спосіб створити безшовне середовище спілкування в машині.

MirrorLink ™ – це зростаючий галузевий стандарт, що дозволяє користувачам встановлювати мережеве з'єднання між смартфонами та системою *IVI* через *USB, Wi-Fi, Bluetooth, UPnP* (*Universal Plug and Play*), *VNC* (Віртуальні мережеві обчислення) та *RTP* (у реальному часі Протокольні) технології. Деякі автомобільні виробники вже прийняли саморозвиваючіся технології, такі як *ConnectedDrive* від *BMW*, *SYNC* від *Ford* та *Entune* від *Toyota*. Найближчим часом те, що з'являється на маленьких моніторах на смартфонах, буде безперешкодно відображатися на автомобільних моніторах, що дозволить користувачам використовувати кнопки на центральній консолі і навіть кнопки на кермі для управління своїми функціями смарт-телефону, наприклад, руками безкоштовний набір, відтворення аудіо та відео, навігація, мобільні програми тощо

MirrorLink ™ не тільки співпрацює з розробниками смартфонів чи автомобільними компаніями, щоб створити союз, але також тісно співпрацює з розробниками програм для смартфонів. Через важливість безпечного водіння стандарт *APP* є відносно жорсткішим. Для специфікації вимог передбачено два фокуси: (*I*) Сумісність дисплея, яка визначає розмір піктограми, роздільну здатність та орієнтацію. (*II*) Управління користувальницьким інтерфейсом, що включає в себе натискання на один дотик та поворотну ручку / ручку перемикання.

Крім того, система не повинна подавати інформацію, яка може призвести до потенційно небезпечної поведінки водія або інших учасників дорожнього руху. Матриці символів 5×7 (від ширини до висоти) повинні бути мінімальним використанням буквено-цифрових символів у разі невдалої ідентифікації. Забороняється як автоматична прокрутка, так і контрольована користувачем ручна прокрутка. Більше того, драйвери повинні мати можливість відновити

перервану послідовність взаємодій із системою в точці переривання або в іншій логічній точці. Тому недоцільно пересаджувати змагання зі смартфонів при розробці програм для *MirrorLink* ™. Він повинен слідувати нормам ЄС, щоб як покращити охоплення послугами *MirrorLink* ™, так і полегшити їхнім додаткам для розваг, аудіо / відео та безпеці їзди задовольнити необхідні вимоги до підключеного керування користувачем.

1.3. Висновки до розділу

Швидкий цикл розвитку побутової електронної промисловості спонукав споживачів очікувати нових, кращих та швидших функціональних продуктів протягом декількох місяців. На відміну від циклу розробки автомобільної промисловості, де найшвидший час, коли новий автомобіль переходить від дизайну до виробництва, становить близько двох років – і це його штовхає.

Для того, щоб залишатися конкурентоспроможними, *OEM*-виробники швидко працюють над впровадженням підключених функцій у нові транспортні засоби. Деякі виробники, такі як *Ford* та *Mercedes Benz*, зазнали критики за впровадження інформаційно-розважальних систем. *Mercedes Benz* критикували за інтеграцію *Facebook* та *Twitter* у їхню інформаційно-розважальну систему. *Ford* виявив недолік за програмні помилки та заплутаний користувальницький інтерфейс. Поспішаючи бути першим на ринку досвіду автомобілів *XXI* століття, виробники автомобілів ризикували зіткнутися з нестандартними, випадковими та непродуктивними.

На даний момент *In-Vehicle Infotainment* (*IVI*) представляється Диким Заходом автопрому. Різні партнерські відносини між виробниками обладнання та постачальниками програмного забезпечення, а також нестандартні користувальницькі інтерфейси (різні піктограми та жести для сенсорного екрану) відрізняються від фактичного стандарту, який *iOS* та *Android* створили у галузі смартфонів та планшетів.

РОЗДІЛ 2
СТРУКТУРА РЕПОЗИТОРІЇВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМОБІЛЬНИХ ОПЕРАЦІЙНИХ СИСТЕМ

Зараз кожен автомобіль має декілька електронних блоків управління (*electronic control unit, ECU*), пов'язаних між собою внутримашинной мережею. Ці блоки взаємодіють через стандартні шинні архітектури, такі як мережа контролерів (*controller area network, CAN*), мережа передачі даних мультимедійних систем (*media-oriented systems transport, MOST*), *FlexRay* і локальний інтерконект (*local interconnect network, LIN*). У порівнянні з *Ethernet*, широко використовуваним для зв'язку ПК, перераховані шини працюють повільніше - в автомобілях обсяг інформації, що пересилається невеликий, але її необхідно обробити за кілька мілісекунд. Збільшення числа пов'язують *ECU* призводить до необхідності створення більш складних структур Внутримашинних мереж, які потребують особливої електричної та електронної архітектури.

– надійність: автомобільні програмні системи повинні працювати виключно надійно в складній мережі *ECU* протягом всього терміну експлуатації автомобіля;

– функціональна безпека: такі функції, як антиблокувальна гальмівна система і *ESC*, вимагають безвідмовної роботи, що визначає високі вимоги до процесів розробки програмного забезпечення і до самих програм;

– робота в режимі реального часу: швидка реакція (від мікросекунд до мілісекунд) на зовнішні події вимагає оптимізованих операційних систем і особливої програмної архітектури;

– мінімальне споживання ресурсів: будь-яке доповнення обчислювальних ресурсів або пам'яті збільшує вартість продуктів, що при мільйонних тиражах виливається в чималі гроші;

– надійна архітектура: автомобільне програмне забезпечення повинне витримувати спотворення сигналів і підтримувати електромагнітну сумісність;

– електронно-механічне управління замкнутого циклу.

2.1. Операційні системи автомобілів

У сучасних автомобілях все частіше і частіше можна зустріти операційні системи, на відміну від мультимедійних систем вони можуть підлаштовуватися під користувача і управляти безліччю функцій. Провідні розробники програмного забезпечення як *Apple, Google, Microsoft* і *Linux* планують масове впровадження своїх систем на серійні автомобілі.

Як подейкують експерти автомобільної промисловості, до 2030 року більше 75% машин будуть мати на борту одну або більше операційних автомобільних систем. Як такої звичної операційною системою, автомобільна система не є. Швидше за все це програма, яка забезпечує водієві доступ до встановленими програмами вашого смартфона.

У такому випадку головним пристроєм системи буде смартфон, а дисплей на центрально панелі буде тільки його додатковим екраном смартфона. Можна сказати, що без смартфона така операційна система майже марна, тому заздалегідь варто задуматися про сумісність вашого смартфона з автомобільної операційною системою.

2.1.1. *Android Auto* від *Google*

Найбільш затребуваною і найпоширенішою автомобільної операційною системою є *Android Aut*o від компанії *Google*. За попередніми даними близько двадцяти восьми компаній планують впровадження даної операційної системи на свої серійні автомобілі.

Як правило, смартфон підключається до цієї операційної системи за допомогою *USB* роз'єму. Крім управління і підключення, система підзарядить ваш смартфон. Не виключається варіант підключення смартфона до системи за допомогою *Bluetooth* технології. Управління додатками буде здійсняться за допомогою сенсорного екрану на центральній панелі автомобіля, голосом (при

наявності в автомобілі функції голосового управління) або кнопками на керма управління.

Завдяки цій операційній системі можна управляти:

– дзвінками зі смартфона;

– читання і озвучення смс повідомлень, пошти та іншого;

– використання вбудованої навігації в смартфон;

– відтворення музики на смартфоні;

– отримання інформації про стан автомобіля;

– використання інших функцій смартфона.

Не можна говорити, що це повний список функцій, він постійно розширюється відповідно з прогресом смартфонів. Система передбачає збереження даних навіть при включеному запалюванні.

2.1.2. Операційна система *CarPlay* від *Apple*

Паралельно з попередньою операційною системою, компанія *Apple* веде розробки своєї системи *CarPlay*. В основному цій системі віддають перевагу дорогі марки автомобілів, наприклад це *Ferrari, Lexus, BMW, Mercedes-Benz, Mini, Toyota* і *Citroen*. Хоча ці виробники не виключають і хочуть встановити дві системи паралельно *CarPlay* і *Android Auto*.

Як уже зрозуміло, система розроблена для інтеграції з широко відомим смартфоном *Apple*. До системи смартфон підключається за допомогою кабелю з *Lighting* роз'ємом. Крім того, на *Apple* смартфоні повинна бути встановлена система *iOS*7 і вище. Управління, функціонал і інші дрібниці в системі *CarPlay* такі ж, як і в *Android Auto*.

2.1.3. *Windows* в автомобілі

Трохи відстаючи від конкурентів, йде зі своїми розробками компанія *Windows*. В основу цієї автомобільної ОС покладена вже готова структура, яку застосовують на ПК і смартфонах цієї компанії. Як і в попередніх системах, основою управління служить сенсорний дисплей на передній панелі. У планах запровадити власну систему управління голосом під назвою *Cortana*.

Функції як і раніше стандартні: повідомлення, дзвінки, радіо, навігація, інформація про автомобіль і мультимедіа смартфона. Не дивлячись на вогкість системи і недосконалий код, бажання впровадити в автомобілі висловили компанії *Ford, Nissan, Fiat* і *Kia,* в якості другої операційної системи.

2.1.4. *Linux Automotive Grade*

Не так відома нині операційна система *Automotive Grade* від *Linux,* але в ній зацікавилися багато авто виробники. Плюсом є відкритий код операційної системи. Це дозволяє коригувати *AGL* систему під потрібні виробника. Можна міняти графіку, додавати безліч нових функцій і просто зробити таким, як його хоче бачити покупець. Роботою цієї операційної системи зацікавилися авто компанії *Toyota, Land Rover, Jaguar, Nissan* і *Hyundai*.

Як висновок хочеться сказати, що в майбутньому авто виробники хоча впроваджувати кілька операційних систем паралельно на один автомобіль. Таким чином купуючи автомобіль не потрібно буде підбирати під нього смартфон або ж підбирати автомобіль під ваш смартфон.

2.2. Аналіз розширених можливостей *Android Automotive OS*

Android Automotive OS - це найсильніше втручання *Google* уавтомобільний домен. Протягом тривалого часу автовиробники насторожено ставилися до того, щоб надати найбільшій пошуковій системі доступ до своїх інформаційно-розважальних систем та приватних даних драйверів. Вони змирилися з необхідністю капітального ремонту застарілих інформаційно-розважальних систем, які вийшли на поверхню додатку *Android Auto*.

Технологія дзеркального відображення даних смартфона на приладовій панелі автомобіля зробила такі *IVI*-системи такого автомобіля більш дружніми та функціональними. Водії по-справжньому оцінили можливість легко отримати доступ до своїх часто використовуваних *Google Maps, Waze, YouTube Music, Spotify* та безлічі програм у соціальних мережах на більших екранах *HMI,* що відволікають увагу. *Android Auto* поставляється з ще більшою кількістю

оновлень, таких як додана програма запуску програм, що означає, що для відкриття програми потрібно менше натискань, покращений та простіший інтерфейс, більші піктограми та темна тема, схожа на салон автомобіля.

У 2018 році дзеркальні рішення (*Android Auto* та *Apple CarPlay*) вже були стандартними або додатковими функціями для всіх нових автомобілів, придбаних в.

2.2.1. Спадкові вбудовані інформаційно-розважальні системи проти *Powered by Android Automotive OS*

Високий попит на дзеркальні рішення є чітким знаком того, що застарілі вбудовані інформаційно-розважальні системи (*IVI*) більше не можуть задовольнити потреби користувачів і йти в ногу знові тенденції цифрової кабіни. Дуже часто мляві навігаційні системи в транспортному засобі додають обмежені можливості для розваг (радіо, відтворення аудіо). Однак головним питанням усіх застарілих систем *IVI* є їх старомодний користувальницький інтерфейс, який значно просунувся на платформах *Android* завдяки великим інвестиціям *Google* та десятиліттям у розробку програмного забезпечення. Серед інших критичних підводних каменів класичних інформаційно-розважальних систем є їх складні та складні варіанти пошуку. Їх поетапний запис пункту призначення може зайняти більше десятка натискань, щоб остаточно ввести адресу. Змінити його під час руху здається майже неможливим подвигом.

Крім того, постійно зростаюча складність цифрової кабіни вимагає більших вкладень виробників обладнання та постачальників рівня 1 у впровадження функцій, пов’язаних із безпекою та розвагами. Отже, нестандартні рішення, такі як *Android Automotive OS,* можуть значно скоротити витрати на додаткові дослідження та розробку повністю життєздатного рішення *HMI*.

Затратна і дорога розробка власного програмного забезпечення *IVI* та загальноприйняте впровадження *Android Auto* (вже) вплинули на виробників продукції на користь нових можливостей співпраці. Спадкові автовиробники, такі як *GM, Volvo* та *Renault-Nissan-Mitsubishi Alliance* першими стали

партнерами *Google,* щоб створити нову різновид інформаційно-розважальних систем на базі *Android Automotive Operating System*.

Volvo була однією з перших, хто продемонстрував ранні версії автомобілів (і) з ОС *Android Automotive* в основі. *GM, Audi* та *Fiat Chrysler* також приєднуються до перегонів.

2.2.2. Переваги автомобільної ОС *Android* для виробників, постачальників першого рівня та кінцевих користувачів

Android Automotive OS - це повністю настроювана вбудована інформаційно-розважальна система, розроблена для плавного підключення до масиву служб *Google* та сторонніх додатків (наприклад, *Spotify, Amazon Music*), знайомих мільйонам користувачів. За даними *Frost* і *Sullivan*, все більше автовиробників підтримують використання ОС *Android*, що дозволяє їм використовувати технології побутової електроніки. Вони також прогнозують, що до 2025 року ОС *Android Automotive* займе перші позиції у вбудованій інформаційно-розважальній системі.

Ця надійна операційна система є гарним знаком для розробки *IVI*, що надає додаткові можливості розробникам:

– сприяє розробці, тестуванню та розгортанню процесів на рівні додатків, включивши стек технологій *Android* з відкритим кодом з однією з найбільших баз розробників. На даний момент перші користувачі ОС можуть використовувати і підтримку майстра, додану до *Android Studio* для розробки медіа-додатків, офіційно випущеної на щорічній конференції розробників «*Google I / O* 2019»

– прискорити час виведення на ринок прототипу автомобіля. Маючи лідера програмного забезпечення на базі автовиробника, розробники інформаційно-розважальних програм отримують безліч нових функціональних можливостей, які можуть спростити їх життя та скоротити час, витрачений на розробку, налаштування та тестування медіа-програм. На прикладі цього, *Google* також випустив найновішу версію, яка вже включає *Google Play Store* (програми для мультимедіа та обміну повідомленнями, автентифікація, навігація та *POI* тощо). Це дозволяєавтомобільний розробникдля емуляції керування, специфічного для

автомобіля, без реального автомобіля під рукою. Це не тільки допомагає розробникам швидше приступати до створення додатків на ОС, але також забезпечує повномасштабну платформу для тестування як нових, так і існуючих медіа-програм.

– Поліпшенню оновлення, щоб забезпечити водіїв останніми оновленнями всіх функцій інформаційно-розважальної програми.

– вирішувати численні проблеми технічного обслуговування та безпеки за допомогою найновіших версій програмного забезпечення.

– покращує досвід клієнтів, забезпечуючи високий рівень мобільності та зв'язку. Ви вже можете почати інтегрувати сторонні платформи (наприклад, *Amazon Music* та *Audioburst*) у свою інформаційно-розважальну систему з надходженням емулятора. Або ви можете розробляти власні програми, щоб, наприклад,оплата в машині.

– застосовувати добре розбірливу функцію розпізнавання голосу від *Google* до функцій управління автомобілем (кондиціонування повітря, опалення тощо).

– побудувати на вершині ОС свої специфічні для бренду програми та використовуйте настроюваний користувальницький інтерфейс.

2.2.3. Впровадження функцій та програм ОС *Android Automotive OS*

Завдяки нещодавно доданим сервісам *Google Play* та нестандартним функціям *Android Automotive,* виробники вже можуть переробити свої застарілі інформаційно-розважальні системи. Однак їм слід враховувати той факт, що ОС *Android Automotive* не має спеціальних для автомобілів функціональних можливостей. Численні *API* недоступні або обмежені, як і багато служб.

Оскільки *Android Automotive* набирає лише пари і не може задовольнити всі потреби виробників обладнання та постачальників рівня 1, вам доведеться заповнити прогалини спеціальними функціями. Щоб охопити всі рівні вбудованої інформаційно-розважальної системи, виробникам доведеться інтегрувати своє обладнання, алгоритми обробки та власні програмні рішення в ОС *Android Automotive*. Хороша новина полягає в тому, що її гнучка архітектура,

в основі якої лежить ядро *Linux*, забезпечує різноманітні інтеграції, а головне, реалізацію власних функцій.

Давайте детальніше розглянемо архітектуру ОС *Android Automotive* на цьому етапі:

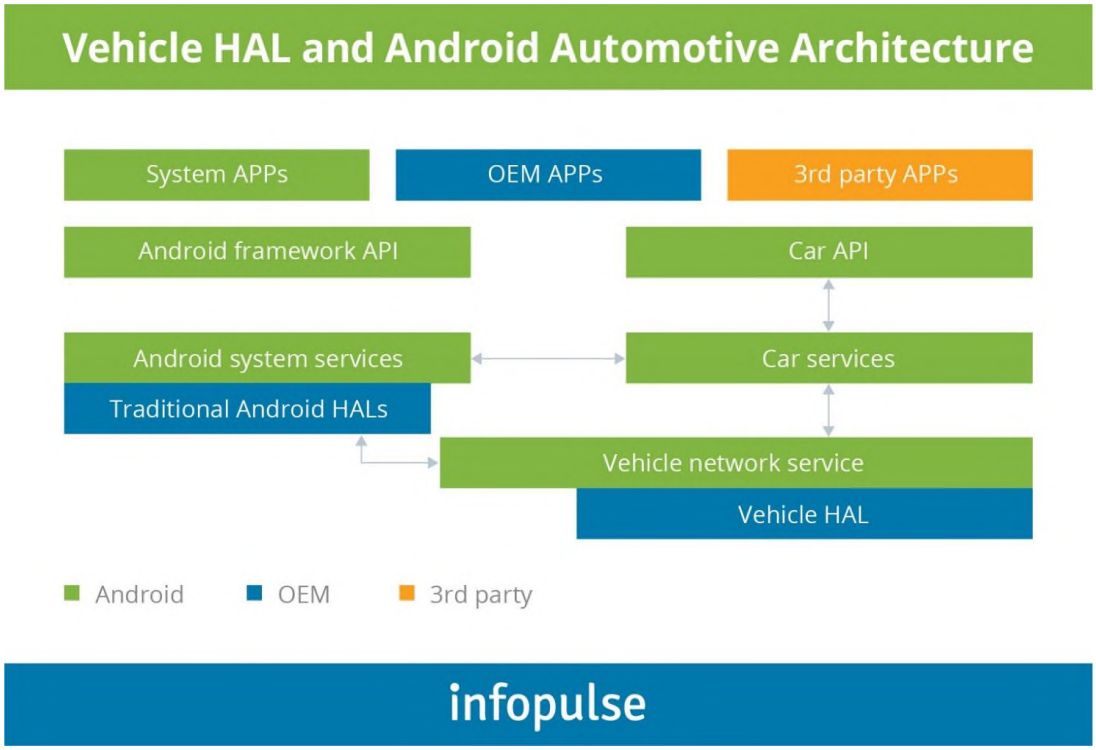

Рис. 2.1. Взаємодія компонентів мультимедійної системи

На перший погляд ви можете сказати, що інтегрована архітектура готової до роботи інформаційно-розважальної системи передбачає необхідне включення та модифікації, необхідні від виробників обладнання, постачальників рівня 1 та сторонніх постачальників додатків для *Android*.

2.2.4. Увімкнення багатокористувацької підтримки

Щоб створити власне рішення, слід впровадити низку модифікацій компонентів ОС *Android Automotive*. Припустимо, вам потрібно увімкнути підтримку для кількох користувачів за допомогою спеціальних роздільних звукових зон (*SSZ*). Останнє додає елемент персоналізації до вашої інформаційно-розважальної системи, якого всі споживачі цифрового контенту очікують сьогодні.

Очікується, що сучасна інформаційно-розважальна система забезпечить інтуїтивне керівництво, дані про дороги в реальному часі, взаємодію та розваги не тільки для водія, але й для пасажирів на передніх та / або задніх сидіннях. Це означає, що кожній людині потрібно більше дисплеїв для доступу до окремого пристрою та передачі свого аудіовмісту, не втручаючись в роботу інших та не спілкуючись з одним інтерфейсом *Android.*

Ви можете вирішити цю складність двома можливими варіантами. Ви можете встановити окремий *SoC* для кожного екрану, але це економічне рішення. Або ви можете застосувати один потужний *SoC* з *Android Automotive OS* за ним. Проблема полягає в тому, що нам потрібно ізолювати середовище всередині *Android* для кожного користувача, щоб забезпечити його індивідуальний зв'язок з аудіосистемою автомобіля.

На цьому етапі ми дійшли до ряду модифікацій в ОС *Android* для реалізації рішення. Для цього знадобиться перевиконання різних рівнів, проілюстрованих на схемі ключових компонентів платформи *Android*:

Ось компоненти, які потрібно змінити:

– ОС *Android Automotive* повинна бути перенесена на *SoC,* вибраний *OEM.* Адаптація буде здійснена на рівні ядра, модифікація драйвера та впровадження нового тощо. Для конкретної автомобільної аудіосистеми потрібен власний драйвер ядра *Linux*;

– слід також впровадити *HAL* (апаратний рівень абстракції). На рівні *HAL* вам доведеться встановити функціонал зв'язку з автомобільною аудіосистемою;

– потрапляючи до рідних бібліотек *C / C ++,* слід додати підтримку автомобільної аудіосистеми та функціональність для окремих звукових зон;

– також підтримку слід додати до *Java API Framework*;

– після додавання підтримки в *Java API Framework* ви зможете розробити програму для відтворення мультимедійного вмісту в певній медіазоні;

– для реалізації ізольованого середовища потрібна купа модифікацій, які можуть вплинути на всі рівні архітектури ОС *Android Automotive* - від рівня ядра до *Java API Framework.*

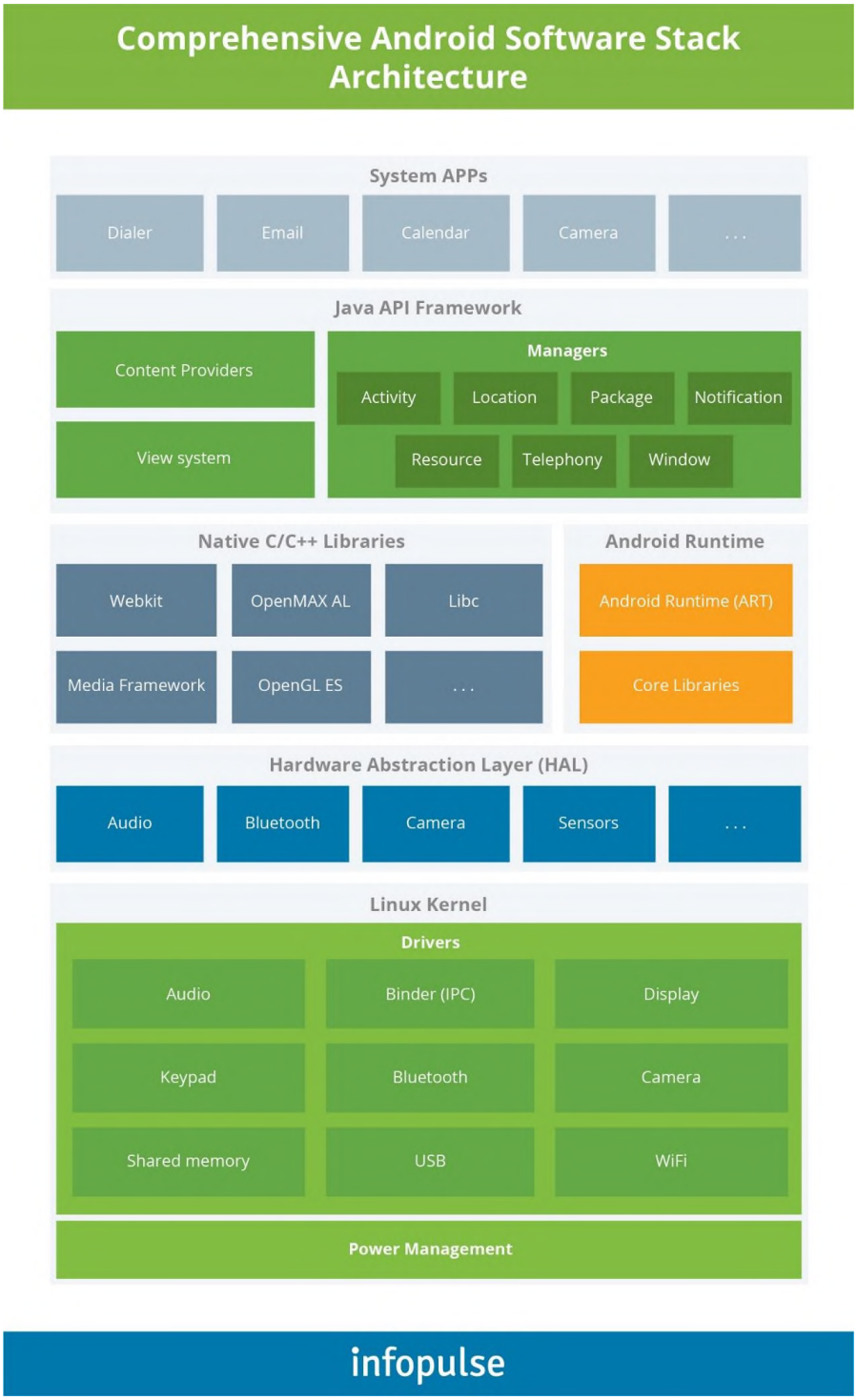

Рис. 2.2. Структура компонентів мультимедійної системи

За допомогою ізольованого середовища ви зможете запустити кілька екземплярів програми (наприклад, *Facebook*) в одній ОС *Android*, кожна з яких працює у власному ізольованому середовищі з різними обліковими даними користувачів. Впровадження ізольованого середовища передбачає модифікації компонентів *Android* на різних рівнях, що дозволяє їм працювати одночасно. Наприклад, програмний компонент *Media Framework* повинен мати можливість

спілкуватися з аудіосистемою автомобіля, а також забезпечувати окремі звукові зони для різних користувачів. Більше того, він повинен відтворювати різний мультимедійний вміст для кількох користувачів одночасно.

Інформаційно-розважальні системи поєднують в одному модулі функції розваг, мультимедіа та інформації про водія. Вони пропонують *AM* / *FM* або супутникове радіо, програвач постійного струму / *DVD* для музики та відео, навігаційну систему, порти даних та мультимедіа (*USB*, синій зуб, лінійний вихід, лінійний вихід, відеовхід), а також загальну інформацію та інформацію про стан автомобіля.

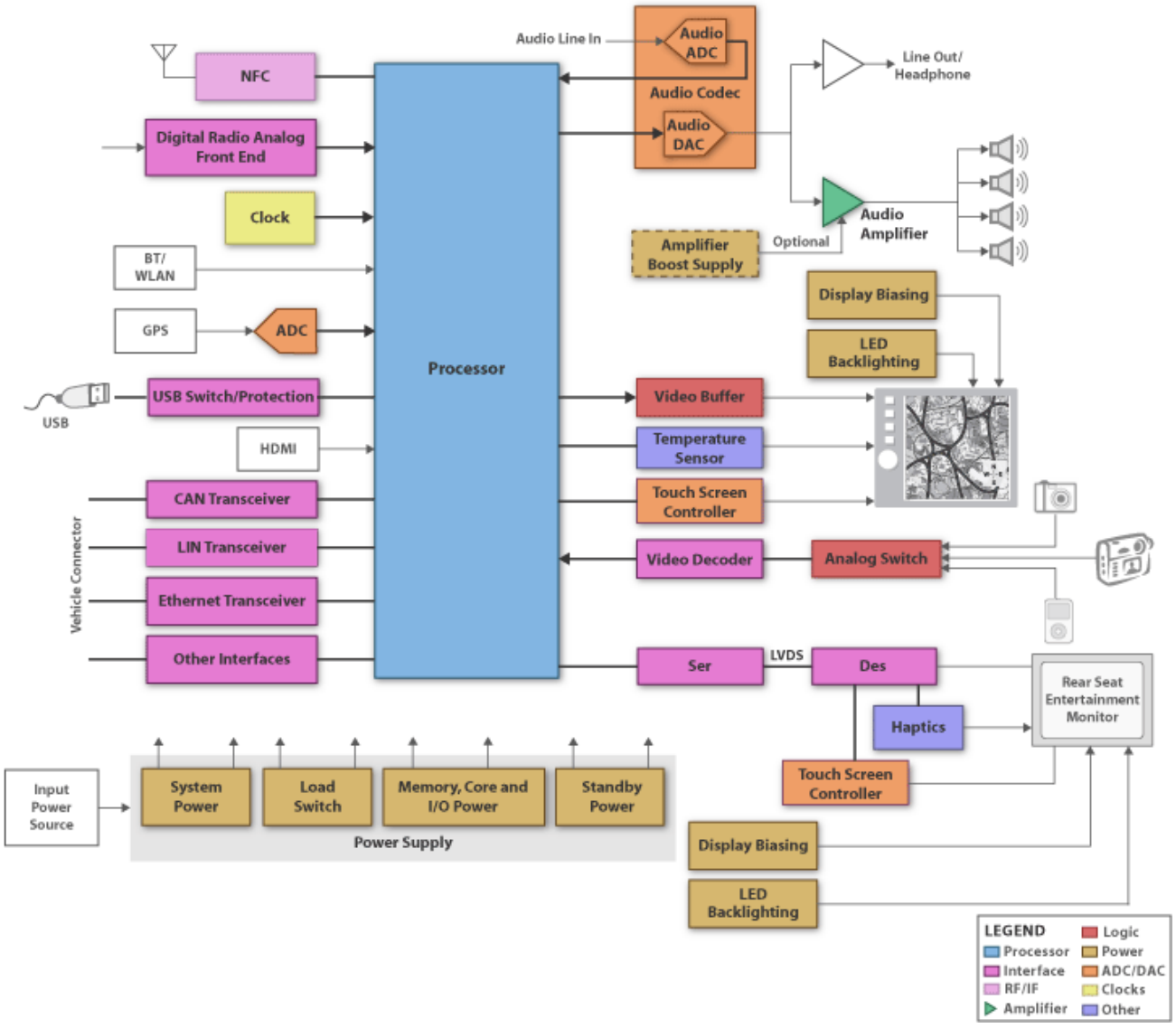

Рис. 2.3. Дизайн Інформаційно-розважальної системи

Управління живленням: Блок живлення підключений до мережевої плати 12 *В* або 24 *В* і регулює зниження / зменшення напруги ЦСП, *UC*, пам'яті та ІС та функцій в інформаційно-розважальній системі. У деяких випадках може бути 10 і більше різних силових рейок, що робить конструкцію джерела живлення критичним завданням при спробі проектувати за розміром, вартістю та ефективністю.

Лінійні регулятори з низьким струмом спокою допомагають зменшити струм витоку акумулятора під час роботи в режимі очікування (вимкнення запалювання), є стійким до напруги скидання навантаження для пристроїв, безпосередньо підключених до акумулятора, і потребують низького відключення та відстеження при низькому рівні роботи кривошипа.

Крім забезпечення підвищеної ефективності перетворення, імпульсні джерела живлення забезпечують поліпшення *EMI* за допомогою контролю швидкості зменшення імпульсного транзистора, перескаку частоти, розширеного спектру або методу тріангуляції для ослаблення пікової спектральної енергії, низького *Iq*, плавного пуску для послідовності потужності та обмеження струму піку, Поетапне перемикання для декількох регуляторів *SMPS* для мінімізації струму пульсацій та нижчої вхідної ємності, вищої частоти перемикання для менших компонентів (*L* та *C*) та функцій *SVS* для індикацій коричневого кольору

Інтерфейси зв'язку дозволяють обмінюватися даними між незалежними електронними модулями автомобіля, віддаленими підмодулями інформаційно-розважальної системи, а також зовнішніми пристроями, такими як пам'ять *USB* або джерела відео.

Високошвидкісна *CAN* (до 1 *Мбіт/с*, *ISO* 119898) - це двопровідна, відмовостійка диференціальна шина. Завдяки широкому вхідному діапазону загального режиму та технології диференціального сигналу, він служить основним типом шини автомобіля для підключення різних електронних модулів у машині між собою. *LIN* підтримує низькошвидкісні (до 20 *кбіт/с*) однопровідні дротові мережі, які в основному використовуються для зв'язку з віддаленими підфункціями інформаційно-розважальної системи. Інтерфейси *LVDS* використовуються для передачі великих обсягів даних (наприклад, *HD*

відеоданих) за допомогою високошвидкісного послідовного з'єднання до зовнішнього місця, такого як відеоекран.

Фронтальний аудіовхід і аудіовихід часто об'єднуються в єдиний кодек. Вхідний рівень аудіосигналу від джерела перетворюється в цифрові зразки АЦП і надходить у ЦСП системи. На вихідній стороні АЦП перетворюють цифровий вихід на аналоговий сигнал, який посилюється до рівнів, необхідних колонкам або навушникам, що використовуються в системі. Використовуючи підсилювачі класу *D*, енергоефективність системи може перевищувати 90% при збереженні низького коефіцієнта *THD*. Ця покращена ефективність призводить до значного зменшення розмірів, ваги та тепла. Автомобільні аудіорішення класу *D* класу *D* демонструють надзвичайно низький рівень EMI і використовуються в системах *OEM* із суворими вимогами щодо електромагнітної сумісності.

Аудіо *DSP* виконує демодуляцію *I* / *Q* і виводить цифровий звук і дані. Це включає такі функції, як гучність, високі, низькі та низькі звукові ефекти, а також більш складні функції, такі як змішування вхідних каналів та обробка кількох каналів цифровим способом, виконуючи обробку звукових ефектів, такі як *Dolby® Pro Logic® II, SRS® Circle Surround ™ II, TruSound* та інші аудіо алгоритми. *UC + DSP* контролює користувальницький інтерфейс, інтерфейс шини та мережевий інтерфейс, а також *GPS*-навігацію та управління сенсорним екраном. Він також використовується для обробки та виведення відеоданих із декількох джерел.

2.3. Архітектура інформаційно-розважального програмного забезпечення в автомобілі

Оскільки машини все частіше стають мережевими IT-об'єктами, загрози безпеці також зростають. Найпомітнішим шлюзом для спроб злому автомобілів є інформаційно-розважальний домен. Таким чином, архітектури інформаційно-розважальних систем наступного покоління повинні вирішувати ці проблеми з нуля. У статті розглядаються архітектурні підходи, які гарантують необхідний рівень безпеки.

Однією з перших комп'ютерних систем у машині був бортовий комп'ютер *Cadillac Seville* 1978 року, керований мікропроцесором *Motorola* 6802 зі 128 байтами оперативної пам'яті та двома кілобайтами ПЗУ. Друкований вихідний код міг займати не більше декількох сторінок.

На відміну від цього, навіть найнижчий кінцевий автомобіль сьогодні містить щонайменше десяток мікропроцесорів; автомобілі найвищого класу оснащені понад 100 мікропроцесорами.

Завдяки інформаційно-розважальним системам, що використовують складні операційні системи, такі як *Microsoft Windows* та різні дистрибутиви *Linux*, загальний вміст вбудованого програмного забезпечення може легко перевищувати 100 мільйонів рядків коду.

Складність зумовлена невблаганним попитом на кращі можливості, оцифровкою ручних та механічних функцій та взаємозв’язком нашого світу. Незважаючи на те, що це зростання електронного контенту було корисним для суспільства, це зростання також є ключовим джерелом нашої надійності, безпеки, вартості та проблем, пов'язаних із часом випуску на ринок. Архітектура інформаційно-розважальної системи наступного покоління повинна допомогти розробникам управляти цією складністю.

2.3.1. Консолідація автомобільної електроніки

Інший важливий автомобільний тренд - це консолідація *ECU*. Оскільки автомобіль продовжує трансформуватися в електронну систему систем, кількість електронних компонентів та пов'язаний з ними вміст проводки стрімко зросли. Це зростання електроніки створює значні виробничі витрати, фізичний слід і час для виклику на ринку для виробників автомобілів. Відповідь полягає в тому, щоб змінити тенденцію зростання і замість цього об’єднати різні функції в меншу кількість електронних компонентів.

Консолідація процесорів тісно узгоджується з тенденцією до змішаних систем критичності, в яких безпека, безпека або критичні компоненти в режимі реального часу повинні співіснувати з менш критичними компонентами. Наприклад, консолідація головного пристрою інформаційно-розважальної

системи з критично важливими для безпеки камерами заднього огляду та / або компонентами інформаційного кластера в режимі реального часу призводить до створення системи змішаної критичності, як показано на рисунку 2.4.

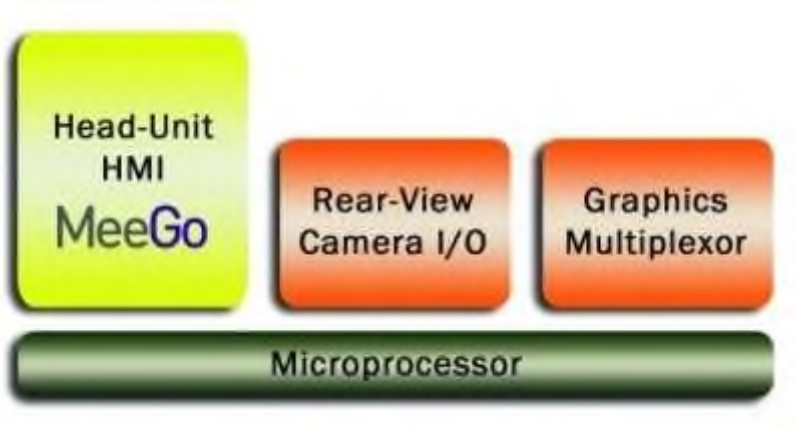

Рис. 2.4. Змішана критичність інформаційно-розважальної системи нового покоління

Архітектура інформаційно-розважальної системи наступного покоління повинна гарантувати, що консолідовані компоненти не взаємодіють непередбачуваними способами, створюючи ризик надійності для критично важливих систем.

У 2010 році дослідники з двох університетів, які досліджували безпеку сучасних автомобілів, відключили двигун і гальма рухомого автомобіля, зламавши вбудований діагностичний порт. Це протвережуючий приклад того, як забезпечення безпеки автомобілів стикається з грізним набором викликів, включаючи застарілі компоненти, не призначені для безпеки, та підхід до ланцюга поставок, який, можливо, досяг меж масштабованості.

Також у 2010 році американські автовиробники ввели функцію, яка дозволяє власникам автомобілів маніпулювати замками та запускати двигун з будь-якої точки планети за допомогою смартфона. Цей зв’язок передбачає віддалену телематичну систему автомобіля, яка стала стандартною у багатьох моделях.

Підключення автомобіля до широкосмугових мереж є пусковим механізмом, який створює загрозу для досвідчених зловмисників. Один недолік може дозволити віддаленому зловмисникові заподіяти шкоду цілому парку транспортних засобів. Комунікації можуть включати автомобіль до сервісного

центру або іншої інфраструктури *OEM*, автомобіль до постачальника мультимедійних послуг, автомобіль до автомобіля, автомобіль до електромережі (електромобілі), автомобіль до смартфона або навіть автомобіль до банку.

На відміну від високоякісних центрів обробки даних, автомобіль навряд чи буде оснащений повним комплектом *IDS*, *IPS*, брандмауерами та *UTM*. Незважаючи на це, нещодавні вторгнення в *Sony*, *Citigroup*, *Amazon*, *Google*, *Sony* та *RSA* яскраво демонструють, що ці захисні механізми є швейцарським сиром проти витончених зловмисників.

Коли атака *Stuxnet* виявилася в 2010 році, керівник Міністерства оборони США, генерал Кіт Олександр, припустив, що критичну інфраструктуру США слід ізолювати у власній захищеній мережі, відмінній від Інтернету. Незважаючи на те, що це може здатися важким рухом, саме таке мислення потрібно. Критично важливі системи автомобіля повинні бути сильно ізольовані від ЕБУ та мереж, не критичних для безпечної експлуатації.

Хоча фізична ізоляція мережі бажана, точки дотику неминуче існуватимуть. Наприклад, навігаційна система автомобіля на деяких ринках повинна бути відключена під час руху автомобіля, що передбачає зв'язок між системами, що мають різну критичність безпеки. Ці підключення збільшують ризик програмних загроз, таких як ескалація привілеїв через вразливості операційної системи, атаки бічних каналів на криптографію та відмови в обслуговуванні.

Архітектура інформаційно-розважальної системи наступного покоління повинна вирішувати ці важливі нові загрози безпеці з нуля. Взаємодія між критичними та некритичними системами та мережами повинна бути обґрунтована на найвищих рівнях управління, суворо контролюватися під час виконання, а також аналізуватися та сертифікуватися без уразливостей на найвищих рівнях забезпечення.

Операційні системи з відкритим кодом, такі як *MeeGo* або *Ubuntu*, добре поважають за дотримання останніх і найкращих мультимедійних стандартів та доступність програм сторонніх розробників. Однак ми не можемо обов'язково

залежати від мультимедійної ОС, щоб контролювати всі аспекти консолідованих інформаційно-розважальних систем наступного покоління.

Загальні операційні системи не можуть завантажуватися досить швидко, не можуть гарантувати реагування в режимі реального часу для таких протоколів, як *CAN*, а також недостатньо надійні та захищені для таких важливих для безпеки функцій, як камера заднього виду та інтегровані кластери. Тому нам потрібна системна архітектура, в якій мультимедійні операційні системи та їх додатки можуть мирно співіснувати з програмами високого рівня захисту в режимі реального часу, розміщеними в операційній системі з рейтингом безпеки в режимі реального часу.

Одним з можливих рішень є наявність декількох процесорів, присвячених різним завданням. Однак це призводить до тих самих проблем, що і сьогоднішні кілька ЕБУ, знижуючи швидкість зв'язку між додатками та збільшуючи складність меж.

Потреба утримувати ці функції на одному процесорі цілком реальна, що дозволяє оптимізувати вартість, цикли обробки, доступність даних та складність. Нам потрібні пісочниці для цих навантажень із різною критичністю. Можна вважати кожну пісочницю ізольованою персоною.

Наступні чотири підходи до декількох персон були комерціалізовані в тій чи іншій формі:

– багатозавантажувальний;

– *Webtop*;

– гіпервізор типу 2;

– гіпервізор типу 1.

2.3.2. Багатозавантажувальний інтерфейс

Протягом останніх кількох років концепція багатозавантаженості була спробована на декількох ноутбуках та нетбуках. У сценарії подвійного завантаження ноутбука замість основної операційної системи на платформі може бути запущена вторинна операційна система, як правило, масштабована *Linux*. Зменшена система, як правило, використовується лише для перегляду веб-

сторінок, і основною метою є надання користувачеві можливості перегляду за кілька секунд після холодного завантаження.

Вторинна операційна система знаходиться в окремому сховищі і ніколи не працює одночасно з основною операційною системою платформи. У деяких випадках полегшене середовище виконується на вторинному мікропроцесорі (наприклад, *ARM SoC,* незалежний від основного процесора *Intel* нетбука).

Вторинна операційна система має хорошу ізоляцію з точки зору безпеки; однак незручність при перезавантаженні та неможливість плавного переключення між особами сильно обмежили прийняття. Варіант багатозавантаженості також непрактичний в консолідованому інформаційно-розважальному середовищі, яке вимагає одночасного виконання персон, наприклад мультимедійної інформаційно-розважальної підсистеми разом із кластером та мережевими комунікаціями в реальному часі.

2.3.3. Концепція веб-версії *Webtop*

Концепція веб-версії забезпечує обмежене середовище перегляду (веб-сторінка), незалежне від основного середовища операційної системи. Однак замість подвійного завантаження веб-поверх працює як додаток поверх основної операційної системи. До тих пір, поки веб-сторінка запускає критично важливі частини системи, а основна операційна система здатна розміщувати критично важливі програми, ізольовані від веб-сторінки, тоді такий підхід може забезпечити необхідну пісочницю. Однак середовище на основі браузера може не відповідати багатим функціональним вимогам сучасних інформаційно-розважальних систем.

Пов’язаний підхід - це запуск веб-сторінки або реплікаційного дисплея зі закріпленого смартфона. Це теж забезпечило б необхідну пісочницю (процесор смартфона використовується для мультимедійних додатків замість вбудованого інформаційно-розважального комп’ютера). Ця техніка дистанційного керування смартфоном була протестована на деяких автомобілях і залишається цікавим варіантом. Однією з його проблем є забезпечення того, щоб смартфон міг забезпечити автомобільний дизайн.

2.3.4. Гіпервізор типу 2

Гіпервізори типу 2 схожі на веб-версії тим, що вторинна персона працює як додаток поверх основної операційної системи. Однак замість розміщення лише браузера вторинною персоною є повноцінна гостьова операційна система, що працює у віртуальній машині, створеній додатком гіпервізор (рис. 2.5).

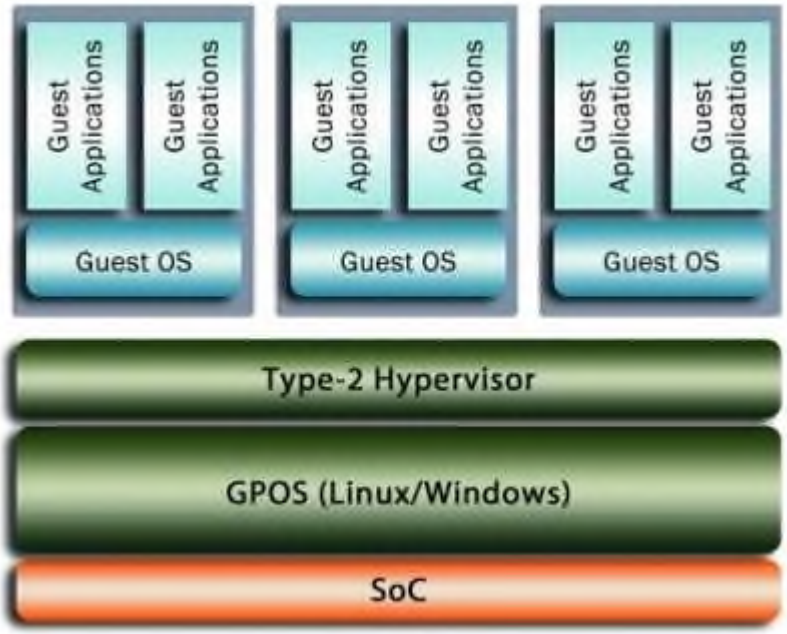

Рис. 2.5. Гіпервізор типу 2: додаток поверх операційної системи загального призначення

Ця гостьова операційна система може бути операційною системою в режимі реального часу, придатною для розміщення таких важливих програм, як камера заднього виду або кластер. Гіпервізор використовує основну операційну систему для обробки вводу-виводу. Підхід віртуалізації відповідає вимогам забезпечення повних функціональних середовищ як для критичних, так і для некритичних персон.

Однак модель *Type*-2 не забезпечує сильної ізоляції. Помилки або вразливості безпеки в первинній операційній системі загального призначення вплинуть на критичні функції, що працюють у віртуальній машині в режимі реального часу. Крім того, встановлено, що додатки гіпервізора типу 2, розгорнуті в корпоративному просторі, містять уразливості, які порушують пісочницю.

Гіпервізори типу 1 також забезпечують функціональну повноту та одночасне виконання декількох персон. Однак, оскільки гіпервізор працює на чистому металі, ізоляція персони не може бути порушена через слабкі місця в операційних системах *persona*. Таким чином, гіпервізор типу 1 представляє перспективний підхід як з точки зору функціональності, так і з точки зору безпеки. Однак загроза вразливості гіпервізора все ще існує, і не всі гіпервізори типу 1 розроблені для забезпечення високого рівня безпеки.

Окремий варіант, гіпервізор типу 1 на основі мікроядер, спеціально розроблений для задоволення вимог сучасного інформаційно-розважального середовища в режимі реального часу, швидкого завантаження та безпеки. Мікроядра забезпечують вищу архітектуру безпеки та безпеки, ніж великі операційні системи загального призначення, такі як *Linux, MeeGo, Android* та *Windows*.

Мікроядро виконує лише мінімальний набір критично важливих системних послуг, таких як управління процесами, обробка винятків та міжпроцесовий зв'язок, у режимі супервізора та забезпечує архітектуру, яка дозволяє виконувати складне програмне забезпечення системи в користувацькому режимі, де їм дозволено доступ лише ресурси, які розробник системи визнав належними. Уразливість або несправність одного компонента не може завдати шкоди критичному компоненту, оскільки заражена підсистема просто не має доступу до цього ресурсу. Оскільки мікроядро відносно просте, його можна офіційно перевірити та сертифікувати незалежними регуляторами на найвищий рівень безпеки.

У мікроядрі гіпервізора Тип-1 віртуалізація системи будується як послуга на мікроядрі. Таким чином, на додаток до ізольованих віртуальних машин, мікроядро забезпечує відкритий стандартний інтерфейс для важких критичних програм, таких як інформаційні кластери драйверів, криптографічні підсистеми та драйвери шини *CAN*, які не можуть бути довірені гостю загального призначення. Архітектура мікроядра типу-1 показана на рисунку 2.6.

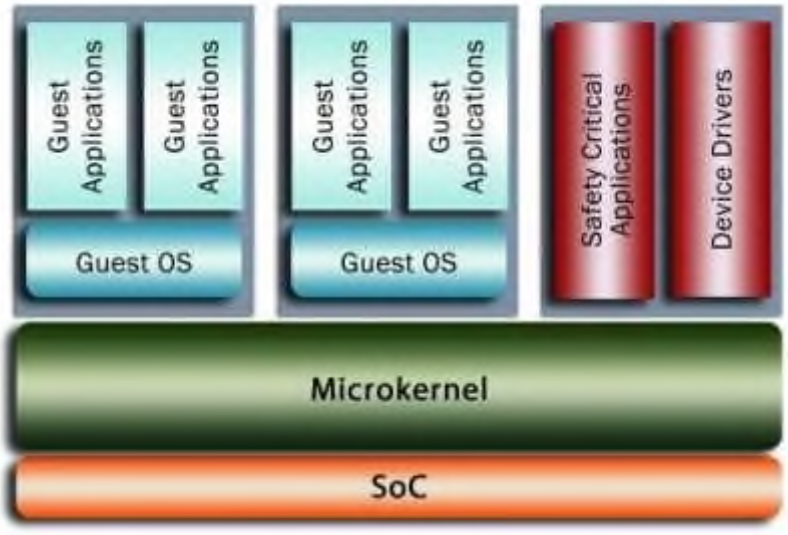

Рис. 2.6. Архітектура гіпервізора мікроядра типу 1

Одним із прикладів гіпервізора мікроядра типу-1 є мультивізор *INTEGRITY Software* від *Green Hills,* побудований на мікроядрі *INTEGRITY,* широко застосовується в автомобільній інформаційно-розважальній системі та інших додатках, що відповідають безпеці в режимі реального часу.

Застосування архітектури гіпервізора мікроядра типу-1 до вищезазначеної інформаційно-розважальної системи змішаної критичності, що складається з основної інформаційно-розважальної ОС та критично важливих для безпеки програм для камери заднього виду та інформаційного кластера драйверів, призводить до архітектури (рис. 2.7).

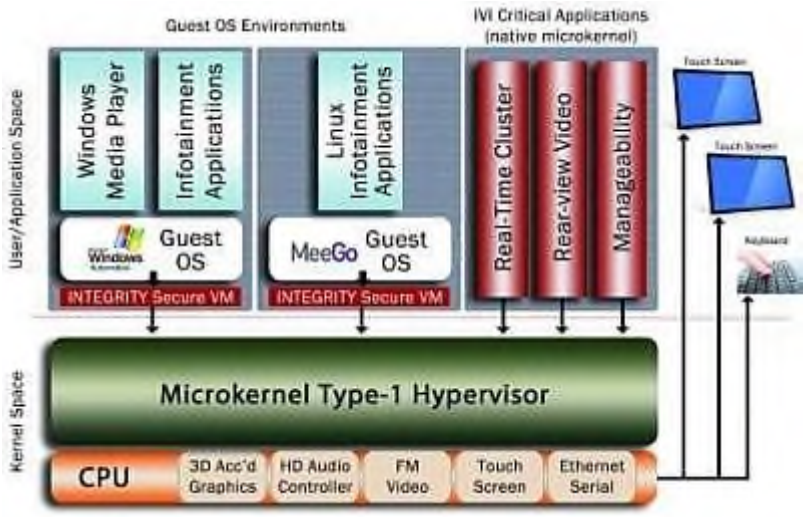

Рис. 2.7. Архітектура мікроядра типу 1 для інформаційно-розважальних систем наступного покоління

2.3.5. Безпечні мережеві транзакції

На рисунку 2.7 також показано додаток для управління. Іншим корисним додатком архітектури мікроядра типу-1 є розміщення захищених підсистем віддаленого зв'язку, що безпосередньо на мікроядрі. Приклади захищених мережевих транзакцій наступного покоління в інформаційно-розважальних системах включають захищений мультимедійний вміст та передачу цифрових прав та віддалене управління системою (наприклад, оновлення мікропрограми та віддалені діагностичні команди) техніками та виробниками.

Ключова ідея тут полягає в тому, що це рішення створює захищене з’єднання, що логічно виходить за межі діапазону від основної системи. Оскільки ключами шифрування, серверними сертифікатами та програмним забезпеченням програмного забезпечення керуються в межах власних легких процесів, гостьова операційна система не може викрасти або пошкодити ці критичні дані, незалежно від проникнення шкідливого програмного забезпечення.

Крім того, власна підсистема безпеки здатна скористатися перевагами *TPM* (або еквівалентних) можливостей, якщо вони є, для апаратного зберігання ключів та атестації платформи. Безпечне з'єднання перемагає атаки "посередині", а також атаки шкідливого програмного забезпечення, які намагатимуться надати криптографічні ключі, що використовуються для безпечного зв'язку.

Genivi - це галузевий альянс, що випускає інформаційно-розважальні платформи для автомобілів з метою зменшення часу виходу на ринок та вартості розробки. Ці довідкові платформи включають попередньо конкурентні функції, які, як вважається, потрібні кожній системі, що дозволяє окремим організаціям сконцентруватися на інноваційних функціях, що забезпечують конкурентні переваги.

Основним принципом досягнення цих цілей є орієнтація на відкриті стандарти та відповідну сертифікацію відповідності. Зважаючи на традиційну роль інформаційно-розважальної системи, яку виконують потужні операційні системи загального призначення, не дивно, що початкові еталонні платформи

Genivi орієнтовані на дистрибутиви *Linux*, які відповідають вимогам заяви про відповідність *Genivi*.

З нетерпінням вперед, зацікавлені сторони автомобільної інформаційно-розважальної системи, включаючи виробників *OEM* та їх постачальників, державні регулятори та пасажирів, повинні звернути увагу за межі мультимедійної системи на новий світ змішаних вимог критичності. Архітектури програмного забезпечення систем наступного покоління потрібні для того, щоб забезпечити майбутні складні інформаційно-розважальні системи, що мають багато можливостей, забезпечувати надійність, безпеку, продуктивність у режимі реального часу та контрольований відбиток, який вимагають автомобільна промисловість та споживачі.

У майбутніх автомобільних системах відбудеться зближення критично важливих функціональних можливостей із традиційними телематичними та цифровими розважальними програмами. Обмеження цих можливостей на одній обчислювальній платформі є критичним для мінімізації розміру, ваги, потужності, виробничої вартості та складності електроніки. Однак безпечне виконання цього вимагає нового системного архітектурного підходу.

Одним з перспективних підходів до пісочниці є віртуалізація системи типу 1, яка може виділити та керувати додатками в режимі реального часу, безпекою та критичними для безпеки нарівні з потужними мультимедійними операційними системами з відкритим кодом. Крім того, доступність технології віртуалізації на широкому діапазоні обчислювальних платформ надає розробникам і технологам остаточну відкриту платформу: можливість запускати будь-який смак операційної системи в будь-якій комбінації, створюючи безпрецедентну гнучкість для розгортання та використання. Ця гнучкість так само вітається в автомобілі, як і на робочих столах та серверах. Ця стаття спочатку з’явилася на *EE Times Europe*.

2.4. Використання *Microsoft Azure* і *Git* для створення репозитарія

2.4.1. *Microsoft Azure* і *Git*

Компанія *Microsoft* останнім часом все частіше звертає свою увагу на *Open Source* технології. Так, наприклад, в останній версії *Visual Studio* 2013 була «з коробки» впроваджена підтримка системи контролю версій *Git*. Хмарна система *Microsoft Azure* також тепер дозволяє працювати з кодом не тільки через *TFS*, а й за допомогою *Git* і інших систем (*SVN, Dropbox, Bitbucket*). У *Microsoft Azure Mobile Services* залишили для роботи тільки *Git*.

Використання *Git* в своєму проекті дає спільну робота над кодом, локальні репозиторії у кожного учасника команди, підтримка версій. Але що ще важливо, так це дотримання принципу *continuous integration* - безперервне розгортання рішення при кожному новому Чекин коду в репозиторій. Кожен раз, коли хтось з учасників команди відправляє свої зміни на сервер, відбувається їх Деплой в хмарі. Все це працює на*Kudu*(До речі, ще один *open source* проект), і досить успішно.

2.4.2. Включення *Git* в сервісі

Для того, щоб почати використовувати *Git* для роботи з *Mobile Services*, необхідно перейти в панель управління цією службою і вибрати відповідний пункт (рис. 2.8).

Після того, як створення сховища завершиться, можна перевірити параметри доступу до нього на сторінці *Configure* в панелі управління службами. У секції *Source Control* буде два параметри: *URL* доступу до репзіторію і *URL*, звернення до якого буде запускати розгортання вашого рішення (рис. 2.9).

Після виконання операції буде створено хмарний *Git*. Давайте отримаємо файли, які зберігаються на сервері, в свій локальний репозиторій. Для цього можна скористатися стандартною консоллю *Git* (наприклад*Git Bash* для *Windows*).

Рис. 2.8. Приклад налаштування репозиторію

Рис. 2.9. Вікно *Source Control*

Клонування серверного сховища (рис. 2.10) локально виконується за допомогою команди:

git clone <*URL*_репозіторія>

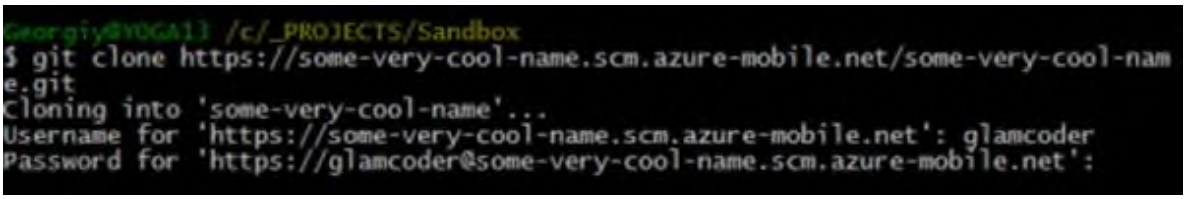

Рис. 2.10. Результат виконання команди

Необхідно вказати *URL*, вказаний в параметрі *GIT URL* в конфіги хмарного сервісу. Також необхідно буде ввести логін і пароль користувача *Git* (якщо у вас

його не було, то при першому створенні *Git* сховища *Microsoft Azure* запропонує придумати цю пару). В результаті має вийти приблизно так, як на картинці нижче (рис. 2.11).

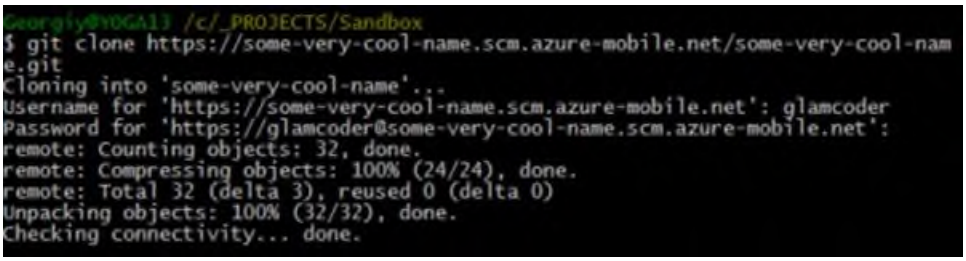

Рис. 2.11. Результат виконання команди

2.4.3. Структура сховища

Як тільки команда завершиться, у вас на жорсткому диску з'явиться директорія з назвою мобільного служби. Всередині неї буде папка*service* і файл *.deployment*. Файл інтересу не представляє, а ось всередину папки *service* варто заглянути.

$ *Ls api extensions package_proj.json scheduler shared table_proj*

Якщо звернути увагу на імена папок, що знаходяться всередині *service*, то можна простежити закономірність, що всі вони називаються так само, як і підсистеми мобільних служб *Microsoft Azure* (*API*, планувальник і таблиці), плюс дві директорії з назвами *extensions* і *shared*.

Директорія *API* відповідає за зберігання файлів, що описують код *Custom API*. Якщо перейти в директорію і подивитися її вміст, то там повинно бути три файли:

$ *Lscoolapi.js coolapi_proj.json readme_proj.md*

Крім файлу *readme.md*, який є в кожній теці і просто описує її вміст, тут зберігаються *js*-скрипти, по одному на кожен *API*, який у вас є. Також, на кожен скрипт з кодом доводиться по одному файлу *.json*, який описує права доступу до конкретного *API*. Давайте подивимося на вміст цих файлів. Нижче - вміст файлу *coolapi.js*:

```
exports.post = function (request, response) {
 // Use "request.service" to access features of your mobile service, eg:
 // var tables = request_proj.service.tables;
 // var push = request_proj.service.push;

 response.send (statusCodes.OK, 'Hello World!');
};

exports_proj.get = function (request_proj, response_proj) {
 var result_proj = {type: 'message', data: 'My project'};
 response_proj.send (statusCodes.OK, result);
};
```

Файл *coolapi_proj.json* містить наступний код:

```
{
 "Routes": {
 "*": {
 "Get": { "permission_proj ": "application"},
 "Post": { "permission_proj ": "application"},
 "Put": { "permission_proj ": "application"},
 "Patch": { "permission_proj ": "application"},
 "Delete": { "permission_proj ": "application"}
 }
 }
}
```

Це опис прав доступу до конкретного *API*, які ставили на самому початку при його створенні. Тут для кожного *HTTP* методу прописані ті чи інші обмеження. У наведеному випадку для всіх методів виставлений тип *application*, що означає, що тільки ті користувачі, у яких є ключ додатку, мають доступ до цих методів. Допустимими значеннями також є:

– *public* - доступ дозволений всім (аналог *Everyone* на порталі);

– *user* - доступ дозволений тільки зареєстрованим користувачам (аналог *Only Authenticated Users*);

– *admin* - доступ дозволений тільки адміністраторам (аналог *Only Administrators*).

Якщо зараз змінити що-небудь в тексті будь-якого з цих файлів, а потім зберегти це в репозиторій, то всі зміни тут же відображаються і в порталі управління (а значить і стануть доступні всім користувачам). Зміна в одному з методів рядку *Hello from Git* і перетворення методу *GET* (доступним для всіх (*public*)) буде відображатись наступним чином.

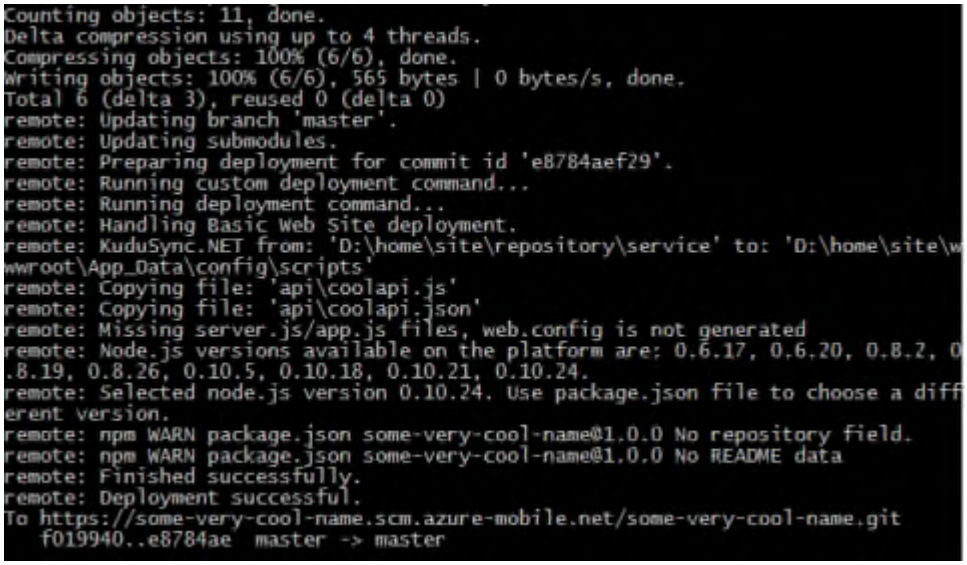

Рис. 2.12. Результат виконання команди

Якщо подивитися в логі операції, то крім простого копіювання файлів на сервер виконуються роботи по розгортанню рішення за допомогою *Kudu*.

Тепер, якщо зайти в портал управління мобільними службами і подивитися код для даного *API*, то можна побачити виконані зміни (рис. 2.13):

Рис. 2.13. Оновлений код *API*

А якщо звернутися до *API* через браузер (що стало можливо після того, як ми послабили обмеження для методу *GET*), можна переконатися, що зміни доступні і ззовні.

Директорія *TABLE* – у цій папці мають зберігатися БД, які є в *Mobile Services*. Але зберігаються тут не дані самих таблиць, а код, який керує подіями вставки / видалення / зміни записів. При створенні таблиці з назвою *My_Table* і коригуванні скрипту, який працює при операції вставки. Такий код вбудовується в результаті в папці *TABLE*:

$ *Ls My_Table.json my_table.insert.js readme_project.md*

За логікою найменування файлів в мобільних службах в файлі *json* зберігається інформація про права доступу до таблиці (і це так, цей файл аналогічний тому, що ми бачили в *API*), а в *js*-файлі зберігається код. Причому назва файлу формується з імені таблиці, до якої він належить, і операції, для якої він визначений. Таким чином, якщо створити подібний файл безпосередньо на локальному комп'ютері (назвати його, наприклад, *my_table.update.js*), написати в ньому код і закоммітіть на сервер, то ці зміни відразу ж будуть доступні в порталі управління, без необхідності ручного створення обробника операції *update*.

Директорія *SHARED* – Тут можуть зберігатися скрипти, які не належать до якоїсь конкретної частини мобільних служб, а навпаки, можуть бути використані повсюдно. Щось на зразок сховища бібліотечного коду в рамках однієї служби. Щоб скористатися цим кодом, треба, по-перше, його написати (і зробити це можна тільки через *Git*, оскільки прямого доступу до цих скриптів з панелі управління немає), а по-друге, підключити новий модуль в існуючому коді.

Створимо в папці *shared* файл з назвою *my_shared.js* і наступним змістом:

```
exports.some_Shared_Method = function () {
 return 'Hello_from_Shared!';
}
```

// Тепер в коді *API* використовуємо цей модуль,

// додавши наступні рядки коду:

exports_proj.get = function (*request, response*)

{

var shared_proj = require ('../*shared*/*shared.js*');

var myshared_proj = require ('../*shared*/*my_shared.js*');

var table_proj = require ('../*table*/*table_shared.js*');

resp_proj.send (*statusCodes.OK, shared_proj.some_Shared_Method*);

response_proj.send (*statusCodes.OK, my_shared.some_Shared_Method*);

};

Це стандартний механізм підключення зовнішніх модулів до додатку. Папка *shared* - це просто директорія на жорсткому диску, де рекомендується зберігати загальні модулі.

В папці *SCHEDULER* будуть лежати скрипти для управління відповідним сервісом, а в папці *EXTENSIONS* зберігається код, який може бути підключений до всього додатку служб і який буде виконуватися рівно один раз при його запуску.

При цьому гарантується, що всі інші операції в додатку будуть припинені до тих пір, поки робота коду розширення не завершиться. Це може бути корисним, якщо ви хочете робити якісь спеціальні настройки в БД при кожному перезапуску мобільних служб.

Правила прошивки мультимедійної системи новим програмним забезпеченням передбачає наступніи порядок дій, який необхідний для прошивки мультимедійної системи новим програмним забезпеенням:

1. Приготувати накопичувач *USB* (флеш-карту). Файлова система накопичувача обов'язково повинна володіти системою *FAT*32. Не будемо вдаватися в необов'язкові подробиці того, для чого ця умова має бути обов'язково виконане.

2. Вставити флеш-карту в комп'ютер і прочитайте її властивості. У разі якщо в накопичувачі, відмінна від система від *FAT*32, зробіть переформатування девайса.

3. Виконання прошивка мультимедійної системи

Необхідно завантажити архів з офіційного сайту, з операційною системою, саме для магнітоли. Після цього скачайте його на ваш накопичувач, тобто на Вашу флеш-карту. Після цих дій, архів повинен виявитися на Вашому накопичувачі. Цей файл представлений на офіційному сайті у вигляді заархивированного архіву. Внутрішній файл, що знаходиться в архіві, відкривати не потрібно, через це дії прошивка виявиться неуспішною. Слідкуйте за тим щоб архів, необхідний для установки, був в «корені» флеш-карти.

Далі, діяти таким чином:

1. Необхідно знайти код автомагнітоли в документах для автомобіля. Якщо він, з якихось причин втрачено, скористайтеся інформацією на офіційному сайті.

2. Увімкніть після цього запалювання автомобіля і запустіть автомагнітолу. Якщо провести ці маніпуляції в іншій послідовності, не включивши запалювання, Ваше пристрій буде вимкнено автоматично, а саме через 10 хвилин. Важливим є те, що якщо це станеться під час прошивки, результати цього можуть бути згубними.

3. Для перевстановлення ПЗ автомагнітоли, вимкніть живлення, і вставте флеш-карту в роз'єм. На індикаторі магнітоли з'явиться інформація з написом про виробленої прошивці і оновленої версії потрібної вам операційної системи. Виберіть потрібну вам систему і натисніть «Так». Після цього підтвердіть Ваш вибір і натисніть кнопку «ОК».

Отже, після виконаних маніпуляцій, активується оновлення Вашого ПО. Після його завершення на Вашій магнітолі має висвітитися «*COMPLETE*». Після цього необхідно витягти флеш-карту. Врахуйте те, що Ваш аудіопрогравач обов'язково повинен відключитися. Якщо Ваша автомагнітола не вимикаючи, то не вимикайте її звичайним способом. У такому варіанті, вимкніть запалювання, і, відкривши капот, на Вашому АКБ зніміть клему від плюса. Через певний час пристрій вимкнеться.

Після цього прошивка, тобто перевстановлення ПЗ завершена. Можна її включати і, потім, вводити потрібний код. Якщо Ви провели всі дії без помилок, то пристрій буде працювати в звичайному режимі.

Прошивка пристрою китайського виробництва передбачає оновлення, а також переустановку ПЗ, в заводських магнітолах від офіційного виробника, які встановлюються в зарубіжних і російських авто, проводиться за бажанням. Вироблені в Піднебесній пристрої, в обов'язковому порядку мають потребу в цій процедурі, за винятком, коли автомобіліст знає досконало китайську мову. При прошивці китайських виробів, теж виникають нюанси. Звернемося до них докладно.

Важливо враховувати те, що офіційної, потрібної саме Вам версії програми, може не бути. Ця проблема буває як з пристроями російських, так і зарубіжних операційних систем. В цьому випадку, вам доведеться шукати і завантажувати кастомний прошивку, і записувати її на флеш-карту.

Важливо враховувати те, що програму можна перекинути нема на *USB*-носій, а також на *SD*-карту. Необхідно щоб місткість Вашого накопичувача-флешки була не менше 2 *ГБ*, а для карти пам'яті, не менше 1 *ГБ*. Ще дуже важливо те, що Ваша файлова система обов'язково повинна бути *FAT* 16 або *FAT* 32. Вкрай важливим є те, що як тільки Ви знайшли установник, відкрити його, і помістити в кореневий каталог Вашого накопичувача.

При прошивці автомагнітоли, візьміть дріт, скріпку або ключ, який спеціально знаходиться в комплекті. Після цього натисніть кнопку «*reset*», підключіть накопичувач, а потім натисніть на кнопку «*power*». Потім автомагнітола включиться і після цього почне встановлюватися ПО. Якщо при перевстановлення ПЗ виникнуть складності, подивіться відео, а також фото інструкцію.

2.6. Висновки до розділу

В данному розділ було розглянуто структуру автомобільних ОС та *IVI* систем в їх складі. Також було оцінено можливості використання стандартних репозиторіїв для оновлення ПЗ автомобільних ОС.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОНТРОЛЮ ВЕРСІЙ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМОБІЛЬНИХ МІЛЬТИМЕДІЙНИХ СИСТЕМ

3.1. Описання модулыв розробленоъ програми

Працюючи спільно з диспетчером автентифікації платформи та панеллю запуску, Navigator управляє запуском, зупинкою, приховуванням та показом програм. Навігатор завжди видно на нижній панелі дисплея.

Область Навігатора включає сім вкладок, де кожна вкладка надає користувачеві доступ до програми або групи програм. Додатки, пов'язані з кожною вкладкою, перелічені нижче:

Натисни і говори - активує автоматичне розпізнавання мови

Домашня сторінка - показує зведений вигляд кількох ключових програм

Навігація - дозволяє користувачеві взаємодіяти з програмою навігації за замовчуванням

Медіа - дозволяє користувачеві відтворювати відео чи музику або слухати радіо

Налаштування - забезпечує доступ до автомобільних налаштувань звуку, клімат-контролю тощо.

Зв'язок - забезпечує доступ до електронної пошти, текстових повідомлень, а також телефонної клавіатури

Програми - користувач може запускати доступні програми

Рисунок 1. Домашній екран

Навігатор реалізований на мові програмування JavaScript. Навігатор залежить від середовища WebWorks та розширень WebWorks для доступу до постійних публікацій / підписок (PPS) та служб екрану. Для отримання додаткової інформації про WebWorks див. Примітки до випуску.

Деякі розширення WebWorks, від яких залежить Navigator, специфічні для Navigator. Ці розширення забезпечують загальнодоступний API, пов’язаний з Navigator, для Navigator та інших компонентів системи. Наприклад, API включає функції для призупинення та відновлення роботи програми. Розширення Navigator WebWorks визначено в:

Щоб зрозуміти реалізацію Навігатора, ви повинні розуміти мову програмування JavaScript та WebWorks Software Development Kit (SDK).

Навігатор реалізований на JavaScript. Поглиблені знання JavaScript потрібні, щоб зрозуміти еталонну реалізацію та розробити нові програми та послуги для платформи додатків QNX CAR. Для отримання додаткової інформації про JavaScript див. Http://www.w3schools.com/js/default.asp.

WebWorks виконує такі функції в програмному забезпеченні платформи:

надає можливість упаковки додатків HTML5

надає WebViews (описано нижче)

забезпечує механізм відокремлення загальнодоступного API JavaScript від приватної реалізації

WebView - це подання, відтворене веб-механізмом, у якому відображається програма. Кожна програма HTML5 має власний WebView. Довірені програми запускаються "в процесі", спільно використовуючи примірник веб-движка. Інші програми працюють "поза процесом", захищені один від одного межами процесу, кожна з яких має свій власний екземпляр веб-механізму.

3.1.1. Керування вікном навігатора

Навігатор використовує WebWorks для управління вікнами.

Додаток WebWorks - це самостійний веб-додаток, який можна розгорнути на платформі додатків QNX CAR. Структура такої програми наведена на малюнку нижче. Розробник програми постачає код JavaScript програми та будь-які необхідні розширення до WebWorks. Прикладне програмне забезпечення може використовувати об'єкти JavaScript WebWorks.

Рисунок 1. Структура програми

За замовчуванням WebWorks налаштовує необхідні WebViews. Роль Навігатора полягає в управлінні цими WebViews. WebView може бути видимим, невидимим або прозорим. Поточна програма відображається у видимому WebView. Прозорий WebView завжди доступний, щоб дозволити сповіщення, такі як діалогові вікна, відображатися поверх поточної програми. Механізм контролера працює одночасно у невидимому WebView. Контролер, перша програма, яка запускається, має доступ до API WebLauncher. Це дозволяє контролеру створювати інші WebViews. Структура інтерфейсу користувача показана на малюнку нижче. Верхня панель стану та нижня панель завжди видно. Кілька WebView можуть бути активними, хоча видно лише по одному.

3.1.2. Отримання інструментів розробки

Щоб отримати та налаштувати інструменти, необхідні для розробки HTML5, виконайте такі дії:

Знайдіть WebWorks для платформи QNX CAR на Ливарному заводі 27.

Завантажте zip-архів та розпакуйте його у свою папку розробки. WebWorks тепер готовий до використання.

За потреби завантажте та встановіть Google Chromium на свій комп’ютер для розробки.

Знайдіть емулятор Ripple для платформи QNX CAR на Ливарному заводі 27.

Завантажте zip-архів та розпакуйте його у свою папку розробки.

У Chromium відкрийте Налаштування> Розширення. Поставте прапорець у полі Режим розробника, а потім натисніть Завантажити розпаковане розширення.

Виберіть вилучену папку Ripple і завантажте її як розширення. Ви повинні побачити кнопку Ripple, додану на панель інструментів вашого браузера. Ripple тепер готовий до використання.

Отримайте веб-сервер за вашим вибором (уникайте файлу: /// протокол).

Виконавши ці завдання, ви отримали всі інструменти, необхідні для розробки HTML5.

Встановлення емулятора Ripple

У вашій системі розробки повинен бути встановлений браузер Google Chromium. Версія Chromium 22. * працює в середовищах Windows, Mac та Linux.

Якщо це повторна інсталяція або оновлення Ripple, обов'язково очистіть кеш-пам'ять у Chromium та видаліть старий плагін, якщо у вас виникли проблеми.

Деякі новіші версії Chromium несумісні з Ripple. Ви можете отримати версію браузера з відкритим кодом Chromium, сумісну з Ripple, завантаживши архівний файл Browsers.zip з проекту QNX CAR платформи Foundry27. Архівний файл знаходиться у файлі Releases> QNX CAR platform 2.0 RR - каталог Ripple.

Ви також повинні мати веб-сервер, який працює у вашій системі. Mac OS постачається з веб-сервером Apache. Ви можете встановити цей же тип веб-сервера в системах Linux. У системах Windows ви можете завантажити інсталятор Windows, Apache, MySql та PHP (WAMP), щоб налаштувати веб-сервер Apache.

Щоб встановити та активувати Ripple:

Завантажте файл пакету Ripple з проекту платформи QNX CAR Foundry27.

Перейдіть на сторінку Випуски файлів.

Подивіться під платформою QNX CAR 2.0 RR - область каталогів Ripple, щоб знайти файл пакету для версії Ripple 1.0.4.

Клацніть на посилання для файлу пакету та дотримуйтесь інструкцій вашого браузера, щоб зберегти файл на своєму комп'ютері.

Розпакуйте вміст пакета. Каталог верхнього рівня пакету містить:

каталог Apps, який містить емуляції Ripple для програм Car Control, Media Player та Navigation; ці емуляції упаковані як розширення WebWorks, що зберігаються в окремих підкаталогах

каталог розширень, що містить файли для розширення браузера Chromium, що реалізує емулятор Ripple

файл README.txt, який містить скорочену версію цих інструкцій.

Розмістіть каталоги, що містять розширення WebWorks для емульованих програм (директори для CarControl, MediaPlayer та програми навігації) та каталог, що містить файли веб-програм, загальні для всіх програм (загальний

каталог), у місці, доступному для вашого веб-сервера. За замовчуванням каталоги знаходяться в <unzip_location> /Ripple.1.0.4/Apps/. Ці каталоги можна розміщувати на локальній машині або в мережі, але вони повинні бути видимими для веб-сервера.

Відкрийте браузер Chromium і перейдіть до діалогового вікна для завантаження розширень, вибравши наступну послідовність параметрів меню: Налаштування> Розширення> Завантажити розпаковані розширення.

Переконайтеся, що параметр Режим розробника, розташований у верхній частині вкладки Розширення, встановлений, натиснувши кнопку Завантажити розпаковане розширення.

У діалоговому вікні завантажувача розширень перейдіть до місця, де ви розпакували каталог chromium, що містить розширення Ripple, а потім виберіть цей каталог. За замовчуванням це <unzip_location> /Ripple.1.0.4/Extention/chromium/. Браузер Chromium відображає попередження про те, що маніфест розширення, який використовує Ripple, застарів. Ця проблема відома команді Ripple і вирішується.

Пізніше в адресному рядку Chromium ви вкажете адресу цільової сторінки власної програми "Hello World".

Спочатку цільова сторінка вашого додатка відображатиметься як звичайна веб-сторінка.

Ми протестуємо одне з включених розширень для Media Player, вказавши адресу для MediaPlayer наступним чином: http: // localhost / <directory_path> /MediaPlayer/index.html

Увімкніть Ripple, клацнувши правою кнопкою миші на сторінці, а потім у спливаючому меню вибравши Емулятор> Увімкнути. Ви також можете ввімкнути Ripple, натиснувши кнопку Ripple, яка відображається у верхньому правому куті вікна браузера Chromium (поруч із меню браузера).

Клацніть Увімкнути в меню Ripple Mission Control.

За замовчуванням ваш веб-сервер може прослуховувати порт 80. Якщо у вас виникають труднощі з дозволами або доступом, перевірте дозволи для

користувача. Крім того, якщо сервер не прив'язується до певної адреси, спробуйте 127.0.0.1, localhost: 9910 або 127.0.0.1:9910.

Якщо ви вперше ввімкнули Ripple для цієї URL-адреси, потрібно вибрати платформу, для якої призначена ваша програма. Емулятор Ripple запам'ятовує цей вибір і завантажує його автоматично під час наступного перезавантаження URL-адреси вашого додатка. Клацніть QNX CAR. Після запуску емулятора ви побачите екран, подібний до наведеного нижче для програми Media Player:

Поточна сторінка тепер показує програму такою, якою вона виглядала б у реальній автомобільній системі (це видно посередині зони перегляду). Елементи керування Ripple для емульованих функцій HMI відображаються з правого боку. Тепер Ripple запущено, і ви можете вибрати активну програму (тобто програму, яку в даний час відображає емульований HMI) і переглянути та змінити її поля за допомогою елементів керування Ripple, показаних праворуч. Додаток працює в центрі вікна браузера, з інформаційними областями емулятора по обидві сторони.

3.1.3. Створення документа конфігурації (config.xml)

У цьому розділі описано, як налаштувати основний файл документа конфігурації для вашої програми "Hello World".

Документ конфігурації - це файл .xml, що містить елементи, що визначають простір імен програми WebWorks, ім’я вашої програми, будь-які дозволи на додаток, стартову сторінку та піктограми, які потрібно використовувати для вашої програми. Цей файл також містить будь-які елементи, необхідні для визначення загальної інформації, яка є специфічною для вашого додатка, наприклад, ім'я автора та опис.

Дійсною назвою файлу конфігураційного документа для програми WebWorks є config.xml, і це не враховує регістр. В архіві програми ваш документ конфігурації повинен використовувати це ім'я файлу, а файл повинен існувати в кореневій папці архіву програми.

Документ конфігурації містить елемент віджета в корені. Елемент віджету забезпечує контейнер для всіх інших елементів. Детальний опис елементів, які

можна використовувати в документі конфігурації, див. На веб-сайті розробника HTML5.

Щоб налаштувати конфігураційний документ для програми "Hello World":

Раніше ви створили каталог під назвою HelloWorld, такий що URL http: // localhost / HelloWorld буде спрямований до цього каталогу. Тепер у своєму каталозі HelloWorld, використовуючи улюблений редактор XML, створіть файл з назвою config.xml у кореневій частині каталогу HelloWorld. Цей файл необхідний для упаковки вашого додатка для платформи додатків QNX CAR.

3.1.4. Створення точки входу для вашого додатка (index.html)

У цьому розділі описано, як створити файл точки входу для програми HTML5 "Hello World".

Точка входу для вашої програми повинна називатися index.html, оскільки цей файл є початковою точкою для проекту та екрану програми на пристрої. . Також переконайтеся, що ви завжди включаєте файл webworks.js у файл index.html. Це повинен бути перший тег сценарію в голівці вашої сторінки, і ми включимо його незабаром.

Щоб створити індексний файл для програми "Hello World":

У каталозі HelloWorld створіть файл із назвою index.html за допомогою вибраного редактора.

Скопіюйте та вставте наступний код у файл index.html.

Запуск "Hello World" у Ripple

У цьому розділі описано, як використовувати Ripple для тестування програми HTML5 в емуляційному середовищі перед запуском на цільовому пристрої.

Щоб протестувати свій додаток WebWorks "Hello World", ви будете використовувати емулятор Ripple, який ви встановили раніше. Ви можете вказати на локальний або віддалений веб-сайт, на якому запущена ваша програма WebWorks, і емулювати все середовище SDK WebWorks без компіляції коду або запуску симулятора. Тестування за допомогою емулятора Ripple забезпечує подібний досвід розробки настільного браузера. Отже, внесення змін настільки ж

просто, як редагування вихідного коду та натискання кнопки Оновити в емуляторі Ripple.

У браузері Google Chromium перейдіть на веб-сервер, ввівши його адресу (наприклад, http: // localhost / CarControl /). Переконайтеся, що ваш веб-сервер працює від імені адміністратора, і що для вашого сервера встановлені відповідні дозволи.

Чудовим інструментом у Google Chromium є веб-інспектор, який дозволяє перевіряти DOM вашого додатка, включаючи стилі CSS та код JavaScript. Щоб розпочати роботу з Web Inspector, просто відкрийте програму «Hello World» у Chromium, клацніть правою кнопкою миші на будь-якому елементі сторінки та виберіть «Перевірити елемент». Це відкриє інше вікно, де ви можете перевірити та змінити стилі DOM та CSS, встановити точки зупинку JavaScript та переглянути трафік мережевих запитів. Ви також можете отримати доступ до консолі JavaScript, щоб перевірити ведення журналу консолі та виконувати довільні команди JavaScript.

Щоб отримати допомогу щодо Web Inspector, відвідайте вікі-сайт Web Inspector або https://developer.blackberry.com/html5/documentation/getting_started_webinsp_microsite_1987477_11.html.

Щоб протестувати програму HTML5 у середовищі, що емулюється Ripple:

Відкрийте браузер Chromium і наведіть вказівник на http: // localhost / <path> /HelloWorld/index.html, де шлях - це місце, де існує каталог HelloWorld, який ви створили раніше (місце, до якого отримує доступ ваш веб-сервер).

Якщо у вас не налаштований веб-сервер, ви можете відкривати файли безпосередньо з локальної файлової системи на своєму комп'ютері, виконавши такі завдання:

Створіть папку з іменем MyApp в одному з таких місць:

C: \ Users \ <Ім'я користувача> \ MyApp (Windows 7)

C: \ Documents and Settings \ <Ім'я користувача> \ MyApp (Windows XP)

/ Користувачі / <Ім'я користувача> / MyApp (Linux)

Скопіюйте папку проекту та весь його вміст у папку MyApp.

В адресному рядку введіть http: // localhost /, а потім папка проекту та цільова сторінка. Наприклад: http: //localhost/MyApp/index.html.

Ваш додаток відкривається у браузері. Ось що ви повинні побачити:

Це виглядає досить добре, але це не робить багато.

Клацніть піктограму Ripple у правій частині панелі інструментів браузера.

Клацніть Увімкнути.

Це повинно перезавантажити сторінку в емуляторі Ripple.

Якщо ви вперше запускаєте емулятор Ripple, перегляньте ліцензійну угоду, а потім прийміть або відхиліть умови.

За замовчуванням ваш веб-сервер може прослуховувати порт 80. Якщо у вас виникають труднощі з дозволами або доступом, перевірте дозволи для користувача. Крім того, якщо сервер не прив'язується до певної адреси, спробуйте 127.0.0.1, localhost: 9910 або 127.0.0.1:9910.

Оскільки ви вперше ввімкнули Ripple для цієї конкретної URL-адреси, ви повинні вибрати платформу, для якої призначена ваша програма. Емулятор Ripple запам'ятає цей вибір та завантажить його автоматично під час наступного перезавантаження URL-адреси "Hello World".

У полі Платформи зліва переконайтеся, що для параметра Платформа встановлено значення QNX CAR. Якщо його наразі не встановлено, клацніть QNX CAR.

Після запуску емулятора та перезавантаження сторінки ви побачите консоль HMI в центрі екрана з вашим поточним додатком "Hello World" в центрі, подібним до такого:

Ваш додаток "Hello World" працює в центрі вікна браузера, з інформаційними областями емулятора по обидві сторони. Ознайомтеся з іншими панелями емулятора Ripple, щоб побачити додаткові налаштування, які можна використовувати для тестування програми "Hello World".

Тепер ви можете отримати доступ до всіх API на HMI.

Праворуч на екрані ви знайдете розширення Ripple, що дозволяють імітувати інформацію про систему HMI. Наприклад, якщо ви змінили гучність із Ripple, ваш додаток отримає ту саму подію, що відбудеться в транспортному

засобі. Ви можете використовувати користувальницький інтерфейс для моделювання системних подій та доступу до усіх API з вашого додатка. У більшості випадків ви отримаєте жорстко закодовані відповіді на свої команди, але поведінка буде такою ж, як і в машині.

Щоб упакувати та скомпілювати програму, потрібно запустити програму BlackBerry WebWorks Packager (BBWP), яка входить до комплекту BlackBerry 10 WebWorks SDK. Пакувальник BlackBerry WebWorks приймає файл архіву програми або розташування папки, компілює його, упаковує, а потім створює файл, необхідний для розгортання.

Коли ви компілюєте програму за допомогою BlackBerry WebWorks SDK, усі її ресурси, включаючи стартову сторінку (index.html), файл документа конфігурації (config.xml), піктограму запуску (icon.png) та інші ресурси , використовуються для створення файлу .bar. Ви можете встановити отриманий файл .bar на пристрій, наприклад, на пристрій Panda для програми "Hello World".

Під час компіляції програми BlackBerry WebWorks Packager виконує такі дії:

Перевіряє вміст файлу архіву програми (.zip) або розташування папки програми.

Створює вихідну цільову папку та очищає всі старі файли (за необхідності).

Створює вихідну цільову папку (якщо вказана) та очищає всі старі файли (за потреби).

Якщо у вас операційна система Windows 7 або Windows Vista, можливо, вам доведеться переконатися, що bbwp та Java працюють із доступом адміністратора. В іншому випадку ці виконувані файли не матимуть доступу до запуску, і ви будете стикатися з помилками при спробі побудувати свою програму.

Якщо ви використовуєте контроль версій або Mac OS, будь-які приховані файли (ім’я файлу, яке починається з крапки) призведе до того, що інструмент упаковки створить файл .bar, який, швидше за все, не зможе (безшумно) під час запуску.

Переконайтеся, що ваш файл документа конфігурації (config.xml) файл точки входу (index.html) та файл JavaScript для WebWorks (webworks.js) знаходяться в корені проекту.

Щоб упакувати програму "Hello World" за допомогою WebWorks:

Стиснути всі файли у вашому додатку в zip-архів, який називається HelloWorld.zip, з index.html, config.xml та значком запуску (icon.png), які знаходяться в корені архіву (не в підкаталозі) .

Вміст каталогу HelloWorld копіюється в архів .zip.

Не стискайте папку каталогів HelloWorld; відкрийте папку та заархівуйте вміст.

Створюючи zip-архів, переконайтесь, що ви не включаєте жодних прихованих файлів, які могли бути створені, таких як .git, .svn та .DS_Store. У Linux ви можете запустити zip із опцією -x для виключення прихованих файлів. Наприклад, наступний приклад виключить усі файли .svn із архіву ZIP:

компакт-диск HelloWorld

zip -r HelloWorld.zip * -x '* .svn *'

У командному терміналі ви виконаєте один із таких інструментів:

bbwp (Linux)

bbwp.bat (Windows)

Ці файли знаходяться в корені архіву WebWorks, який ви завантажили раніше.

Наприклад:

~ / path / BBWP- <версія> / bbwp -d HelloWorld.zip (Linux)

C: \ path \ BBWP- <версія> \ bbwp.bat -d HelloWorld.zip (Windows)

Параметр -d необов’язковий, і він включає режим налагодження, який включає веб-інспектор. Наприклад, наступний результат є типовим результатом цього процесу:

~ / dev / BBWP-1.0.2.9 / bbwp -d HelloWorld.zip

[INFO] Заповнення джерела програми

[INFO] Розбір config.xml

[ПОПЕРЕДЖЕННЯ] Ідентифікатор збірки встановлений у config.xml [версія], але файл ключа підписання не знайдено: author.p12

[ПОПЕРЕДЖЕННЯ] Ідентифікатор збірки встановлений у config.xml [версія], але не вказано пароль для підписання [-g]. Бар не буде підписаний

[INFO] Створення вихідних файлів

[ПОПЕРЕДЖЕННЯ] Не вдалося знайти маркер налагодження

[ІНФОРМАЦІЯ] Інформація: Створений пакет: /home/mylocation/simulator/HelloWorld/HelloWorld.bar

[ПОПЕРЕДЖЕННЯ] Не вдалося знайти маркер налагодження

[ІНФОРМАЦІЯ] Інформація: Створений пакет: /home/mylocation/device/HelloWorld/HelloWorld.bar

[INFO] Упаковка BAR завершена

У командному рядку перейдіть до папки встановлення для BlackBerry 10 WebWorks SDK. Для Windows: cd C: \ Program Files \ Research In Motion \ BlackBerry 10 WebWorks SDK <version>, а для Mac OS: cd "/ Developer / SDKs / Research In Motion / BlackBerry 10 WebWorks SDK <version>".

Встановлення на пристрій

У цій темі описано, як встановити пакет для вашої програми HTML5 на систему QNX CAR.

Платформа для розробки QNX CAR пропонує інструменти для забезпечення належної інсталяції програм.

Для того, щоб використовувати ціль, ви повинні підготувати SD-карту з відповідним зображенням. Картка SD містить завантажувач, IFS і цільовий образ.

Щоб встановити програму "Hello World" на цільовий пристрій:

Скориставшись таким інструментом, як WinSCP, скопіюйте файл, створений після упаковки програми "Hello World" (HelloWorld.bar), і скопіюйте його на цільовий пристрій. Наприклад, ви можете розмістити його в каталозі / tmp.

У вікні терміналу виконайте скрипт bar-install, який знаходиться в папці скриптів (у корені архіву WebWorks, який ви завантажили раніше), таким чином:

/ scripts / bar-install HelloWorld.bar

Змінити каталоги на:

/ apps / app-name / native

Зверніть увагу, що назва програми матиме таку форму:

your-app-name.testDev_HelloWorld ___ d49afbe9

Якщо програма встановлена, але не запускається належним чином (тобто якщо ви торкаєтеся значка і залишаєтесь на панелі прогресу обертання), програма, ймовірно, не має дозволу на виконання; можливо, вам доведеться вручну змінити дозволи наступним чином:

chmod + x / apps / app-name / native / wwe

На сенсорному екрані перейдіть на вкладку Програми (остання вкладка праворуч на нижній панелі Навігатора). Ви повинні побачити свій додаток у поданні ALL. Щоб запустити програму, торкніться її піктограми.

Підвищення продуктивності

Наступні найкращі практики допоможуть вам розробити більш ефективні програми.

Уникайте полотна.

Елементи Canvas та SVG не є оптимальними для використання на мобільних та вбудованих платформах, оскільки належне апаратне прискорення для цих елементів ще не завершено.

Уникайте 2D-перетворень.

Замість цього використовуйте 3D. Наприклад, замість translateX (x) використовуйте translate3d (x, y, z). Це змусить апаратне прискорення перекладу. Ви можете використовувати подібні методи для більшості інших перетворень. Уникайте анімації за допомогою бібліотек JavaScript!

Уникайте непрозорості, закруглених кутів та градієнтів.

Це значне зменшення продуктивності, коли їх часто перемальовують. Якщо їх використовувати економно і переважно на статичних об’єктах, вони не повинні перешкоджати продуктивності, але якщо ви змішуєте ці елементи з анімацією, кнопками або будь-чим, що часто перемальовується, продуктивність страждає. Подумайте про використання зображень замість важкого CSS.

Видаліть елементи з DOM при їх зміні.

Цей прийом особливо корисний, коли ви оновлюєте кілька полів DOM одночасно. Наприклад, якщо ви прокручуєте список зі 100 контактів і хочете їх оновити, оновлення по одному призведе до перемальовування списку 100 разів. Але якщо ви видалите весь список, оновите контакти в пам'яті, а потім знову додасте список, це зробить лише два перемальовування.

Сховати непотрібні вам елементи.

Додавання дисплея: жоден до елементів, які не потрібно відображати, не заважатиме їм відображатись.

Уникайте бібліотек, призначених для використання на робочому столі.

Деякі бібліотеки JavaScript розроблені для використання у настільному браузері з потужним процесором. Спробуйте обмежити кількість сторонніх бібліотек JavaScript, включених у ваш додаток, або спробуйте знайти версії, оптимізовані для мобільного використання.

3.1.5. Використання спрайта

Вони корисні для попереднього завантаження активних станів елементів (наприклад, кнопок із "натиснутим" станом).

Пріоритет маршрутизації аудіоменеджера

Менеджер аудіо дозволяє клієнтам вручну контролювати маршрутизацію своїх потоків.

Основний аудіопристрій підтримує лише основний мікс, тому обраний маршрут передачі аудіопотоку з найвищим пріоритетом. Якщо клієнт не вказує шлях маршрутизації вручну, шлях маршрутизації автоматично обирається для нього на основі пріоритету підключених аудіопристроїв у цьому режимі.

У наступній таблиці перераховані типи аудіо в порядку зменшення пріоритету (тобто, чим вищий тип аудіо в таблиці, тим вищий його пріоритет). Наприклад, якщо AUDIO_TYPE_DEFAULT є типом активного потоку, і починається другий потік типу AUDIO_TYPE_VIDEO_CHAT, політика маршрутизації, настройки та загасання потоку AUDIO_TYPE_VIDEO_CHAT набуває чинності, оскільки він має вищий пріоритет.

Для кожного типу аудіо вказано вплив на звукові потоки з нижчим пріоритетом:

приглушено - потоки з нижчим пріоритетом приглушені для неперехідної події, наприклад телефонного дзвінка. У цьому випадку може бути доречним для програм, що генерують аудіопотоки з нижчим пріоритетом, зробити паузу на час події.

повністю ослаблені - потоки з нижчим пріоритетом приглушуються для перехідних подій, таких як клацання камери. У цьому випадку додаткам може не знадобитися реакція.

Ім'я Опис Правило паралельності

AUDIO_TYPE_SOUND_EFFECT Звукові ефекти, які ніколи не можна послабити, такі як клацання камери. Повністю ослаблює всі потоки звуку нижчого пріоритету.

AUDIO_TYPE_RINGTONE Використовується для відтворення мелодій дзвінка під час вхідного телефонного дзвінка. Повністю ослаблює всі потоки звуку нижчого пріоритету.

AUIDO_TYPE_VOICE_TONES DTMF та сигнали прогресу виклику. Однак також може використовуватися для відтворення нетонального звуку під час телефонного дзвінка. Повністю ослаблює всі потоки звуку нижчого пріоритету.

AUDIO_TYPE_VOICE Потоки, пов’язані з діапазоном голосу, та конкретні речі, пов’язані з телефонією (стільниковий або VOIP). Приглушує всі потоки звуку нижчого пріоритету.

AUDIO_TYPE_VIDEO_CHAT Використовується клієнтом відеочат. AUDIO_TYPE_VOICE не охоплює цей тип через різницю в політиці автоматичного маршрутизації. Приглушує всі потоки звуку нижчого пріоритету.

AUDIO_TYPE_PUSH_TO_TALK Використовується для позначення потоків, пов’язаних із випадками використання “Натисни і говори”. Повністю ослаблює всі потоки звуку нижчого пріоритету.

AUDIO_TYPE_VOICE_RECOGNITION Послуги розпізнавання голосу, такі як VAD. Приглушує всі потоки звуку нижчого пріоритету.

AUDIO_TYPE_TEXT_TO_SPEECH Текстові мовні послуги. Зменшує звукові потоки нижчого пріоритету на 40%.

AUDIO_TYPE_ALERT Сповіщувачі, наприклад про події календаря, електронну пошту, SMS, обмін миттєвими повідомленнями тощо. Зменшує звукові потоки нижчого пріоритету на 40%.

AUDIO_TYPE_VOICE_RECORDING Послуги записування голосу, такі як фокусні точки. Ніяких змін до інших потоків.

AUDIO_TYPE_MULTIMEDIA Використовується програмами медіаплеєра. Ніяких змін до інших потоків.

AUDIO_TYPE_DEFAULT Будь-який некласифікований аудіопотік має типовий тип. Ніяких змін до інших потоків.

Навігація

Платформа QNX CAR 2.0 RR дозволяє використовувати один із декількох підтримуваних навігаційних механізмів.

Ви можете встановити будь-який з цих навігаційних двигунів на платформі:

Системи телекомунікації (TCS)

Розвідник Теленав

Директор вулиці Електробіт (ЕБ)

За замовчуванням вкладка навігації в HMI запускає демонстрацію HTMLGears, яка є лише посиланням на програмування і не має жодних навігаційних можливостей. Щоб замінити HTMLGears, вам потрібно завантажити та розпакувати справжній навігаційний механізм та змінити деякі ключові системні файли, щоб вкладка запустила цей навігаційний механізм.

Інструкції з налаштування кожного з підтримуваних сторонніх навігаційних двигунів ви знайдете на вікі-платформі QNX CAR:

Інструкції з встановлення навігаційних двигунів

Системи телекомунікації (TCS) за бортом навігаційної системи

Навігаційна система TCS - це програма, яка демонструє покрокові навігаційні послуги, маршрутизацію та відображення. TCS включає сенсорні та голосові команди для пошуку та навігації, включаючи запити на вказівки та розташування.

Довідковий додаток являє собою імітацію (не справжню систему за бортом), яка використовує дані GPS для місць у США, налаштованих у налаштуваннях. Після першого сеансу навігації ви можете вибрати пункт призначення - або місце, перелічене в розділі Недавні, або пункт призначення, вибраний із категорії «Близько мене», щоб імітувати покрокову навігацію. На навігаційному екрані програма відображає інформацію про поворот та відстань, а також 3D-карту. На головному екрані HMI відображається інформація про поворот та відстань, коли активна навігація TCS.

Щоб запустити покрокове моделювання:

Торкніться налаштувань, щоб відкрити екран налаштувань.

На екрані налаштувань переконайтеся, що вибрано «Створити доріжку для маршруту» та скасовано вибір гібридного режиму.

Використовуйте доріжку за замовчуванням (Детройт) або виберіть доріжку (файл даних GPS), наприклад Wynn.gps.

Скиньте дані GPS, натиснувши елементи керування на екрані налаштувань у такому порядку: скидання> встановити> запустити> зупинити> виконано

Переконайтеся, що екран розпізнає кожне натискання кнопки.

На головному екрані навігації торкніться "Поруч зі мною" та виберіть пункт призначення з однієї з категорій. Наприклад, якщо ви завантажили набір даних Detroit_BrightCleaners.gps, у категорії Ресторани виберіть гірське бістро Rojo, а потім натисніть Go. Якщо ви вибрали набір даних Wynn.gps, ви можете спробувати перейти до готелю Blair House Suites або медичного центру лікарні Санрайз.

Під час перезапуску навігаційного сеансу пам’ятайте завжди скидати дані GPS на екрані налаштувань.

Ви можете використовувати стандартні елементи масштабування та панорамування, щоб змінити розмір карти та змінити область, яку ви

переглядаєте. Торкніться центру регулятора руху, щоб скинути вигляд до поточного місцезнаходження. Ви також можете змінити відображення карти та додати дані про дорожній рух за допомогою супутника та управління дорожнім рухом.

Розвідник Теленав

Scout - це персоналізована навігаційна служба, яка забезпечує навігацію за бортом, покрокову навігацію. На інформаційній панелі відображається детальна дорожня карта, яка постійно оновлюється під час навігації. Сенсорні кнопки дозволяють отримати доступ до таких ключових функцій, як пошук та улюблені місця призначення.

Щоб користуватися Scout, ви повинні мати активне підключення до Інтернету. Скаут підключається до Інтернету під час запуску для завантаження навігаційної інформації і залишається на зв'язку, постійно завантажуючи статистику руху в режимі реального часу для постійної навігації.

Scout постачається із попередньо завантаженим навігаційним моделюванням, яке надає інформацію про маршрут поїздки між Уолл-стріт та аеропортом La Guardia в Нью-Йорку. Це моделювання показано під час першого завантаження програми.

Загалом, ви можете вибрати дім або роботу як наступне місце призначення за допомогою функції "Перейти до" наступним чином:

Натисніть кнопку Навігація до у верхньому правому куті головного екрана (за замовчуванням), щоб відкрити дві панелі, щоб вибрати дім або роботу як наступне місце призначення.

Натисніть кнопку «Перейти» на панелях «Дім» або «Робота», щоб переглянути список із трьох розрахованих маршрутів, кожен з яких має назву, загальну відстань та передбачуваний час у дорозі.

Торкніться запису для одного з цих списків маршрутів, а потім натисніть кнопку «Перейти», щоб розпочати навігацію з поточного місця до вибраного пункту призначення.

На головному екрані ви можете торкнутися кнопки пошуку, а потім торкнутися однієї з кнопок категорій зліва, щоб переглянути список найближчих

пунктів призначення. Ви можете шукати банки, ресторани, заправні станції тощо. Потім можна натиснути запис у списку результатів, щоб побачити адресу відповідного пункту призначення. Щоб додати цей пункт до списку Вибране, натисніть кнопку із значком серця.

Щоб отримати доступ до списків Вибране в будь-який час з головного екрана, просто натисніть кнопку Вибране. На екрані "Вибране" можна перерахувати всі улюблені пункти призначення, які знаходяться поблизу вашого поточного місцезнаходження, нещодавно використані вибрані місця та найближчі аеропорти.

Кнопка «Аудіо», яка також відображається на головному екрані, відкриває панель у лівій частині екрану для керування функціями навігації на основі аудіо Scout. Однак зверніть увагу, що голосові вказівки та попередження про дорожній рух, що виводяться через динаміки автомобіля, не працюють.

Ви можете змінити інформацію, намальовану на карті, перемикаючи ряд елементів керування параметрами відображення. Знайдіть кнопку із піктограмою, що показує два прямокутники, що перекриваються, у правому нижньому куті головного екрану розвідника, а потім натисніть цю кнопку, щоб відкрити ще три кнопки. Вони дозволяють вмикати / вимикати світлофори, відеокамери та супутникові знімки на дорогах.

Щоб переміщатися по карті, торкніться та перетягніть. Щоб збільшити або зменшити масштаб, торкніться кнопок + і -.

Директор вулиці Електробіт (ЕБ)

Призначене для автомобільних систем, це навігаційне програмне забезпечення дозволяє шукати пункт призначення та одним натисканням кнопки розраховувати оптимальний маршрут для досягнення цього пункту та активувати навігаційну службу для відстеження прогресу подорожі. Поки ви їдете, карта анімує прогрес подорожі в режимі реального часу, відображаючи поточну вулицю, наступний поворот та передбачуваний час та відстань до пункту призначення.

Карта Street Director дає надзвичайно масштабований вид на ваше оточення. Ви можете збільшити (натиснувши кнопку +) вигляд на рівні вулиці та

переглянути деталі окремих доріг та будівель або зменшити (натиснувши кнопку -) для перегляду країни чи навіть планети.

На головному екрані відображається інформація про поворот та відстань, коли активна навігація Street Director.

Окрім можливості пошуку за призначенням на основі імен, Street Director має функції для відстеження ваших улюблених напрямків та останніх подорожей. Ці функції забезпечують швидкий спосіб знайти та вибрати одне з найбільш часто відвідуваних місць як наступне місце призначення.

Цей випуск платформи QNX CAR постачається із зразками картографічних даних для Торонто, Онтаріо, Канада, включаючи інформацію про місцезнаходження для багатьох визначних місць, демонструючи реалістичну навігацію на основі карт. Дані карти зберігаються у проекті Foundry27:

Медіа плеєр

Екран медіаплеєра містить такі програми: Play Radio, Play Music, Play Video та Search.

Щоб відкрити відповідну програму, торкніться одного з активних елементів керування або використовуйте голосові команди. Щоб повернутися до головного екрана медіаплеєра, торкніться вкладки Медіаплеєр на панелі завдань.

Меню мультимедійних джерел доступне за допомогою програм Play Radio, Play Music та Play Video. Це меню полегшує перемикання між відтворенням музики та відео з різних джерел. Детальніше див. У розділі меню Джерело медіа.

Грати на радіо

Коли ви торкаєтесь опції меню Радіо, ви побачите підменю з двома параметрами:

AM / FM - торкніться, щоб переглянути радіо за замовчуванням

Pandora — торкніться, щоб переглянути версію Pandora, інтегровану в медіаплеєр

Інтегрована версія Pandora виглядає та функціонує як автономна версія.

Якщо Pandora не налаштовано, натисканням кнопки «Радіо» ви потрапите безпосередньо на екран, на якому відображається посилальний інтерфейс для радіопрограми:

Ця програма є імітацією на платформах розробки Panda та i.MX6, які не мають радіоапаратури.

Додаток Radio розроблено як додаток Sencha Touch, використовуючи архітектурний шаблон MVC фреймворку. Додаток має два профілі:

висока - представляє вдосконалену анімацію та графічно насичений скін

середина - має більш базовий зовнішній вигляд

На дошці Panda профіль за замовчуванням для програми Radio є високим, але ви можете налаштувати програму на використання середнього профілю. Ці профілі (і профіль, що використовується іншими компонентами HMI) визначаються об'єктом PPS. Для отримання додаткової інформації про встановлення типу профілю див. Запис / pps / qnxcar / system / settings у Довіднику об’єктів PPS.

Усі елементи керування радіопрограмою, включаючи повзунок, циферблат, AM, FM, пошук, сканування та попередні налаштування, функціонують. Але знову ж таки, сама радіо є лише імітацією на платах Panda та i.MX6.

Грати музику

Натисніть кнопку Відтворити музику, щоб відкрити програвач музики:

База даних Juke Box завантажена кількома музичними файлами, які HMI показує в активній каруселі. За допомогою голосових або сенсорних елементів керування ви можете запускати, призупиняти або пропускати доріжки (перемішування та повторення недоступні як голосові команди). Ви можете використовувати голосові команди для відтворення мультимедіа за названою піснею, альбомом чи виконавцем.

Відтворити відео

Торкніться елемента керування Відтворити відео, щоб відкрити програвач відео. Цей випуск включає таке відео:

"Виготовлення еталонного автомобіля QNX: Jeep Wrangler"

Елементи керування "Відтворити", "Пауза", "Вперед" та "Назад" функціонують. Повноекранний контроль не працює.

Відтворення відео на i.MX6

Кодеки, необхідні для відтворення відео на платформі i.MX6 Sabre Lite, доступні як окремий пакет на сайті платформи QNX CAR.

Щоб увімкнути відтворення відео на i.MX6:

Отримайте відеокомпоненти Freescale з сайту платформи QNX CAR:

Платформа QNX CAR 2.0 RR - Відеокомпоненти Freescale

Створіть каталог / base / lib / firmware / vpu /.

Помістіть двійковий файл кодека VPU (vpu_fw_imx6q.bin) в каталог / base / lib / firmware / vpu /.

Перезапустіть систему за допомогою команди перезавантаження.

Пошук

Торкніться елемента керування пошуком, щоб отримати доступ до функції пошуку. HMI відображає список джерел пошуку, таких як iPod, USB-накопичувач або Juke Box. Торкніться запису для пристрою, який потрібно шукати. Потім HMI відображає текстове поле, яке дозволяє знайти виконавця, альбом або доріжку за іменем.

Меню джерела медіа

Торкніться Меню у верхній частині екрана Play Radio, Play Media або Play Video, щоб переглянути список джерел мультимедіа. Медіа-джерела можуть включати: радіо, Apple iPod (якщо підключено), USB-накопичувач (якщо підключено), пристрої DMS (якщо ввімкнено DLNA) або базу даних Juke Box. Потім можна зробити наступне:

Торкніться Радіо, щоб отримати доступ до інтерфейсу з посиланнями на радіо, описаного раніше (або Pandora, якщо він налаштований).

Торкніться iPod, USB-накопичувача, запису для будь-якого локального пристрою DMS або Juke Box, щоб переглянути список музичних носіїв, відсортованих за списками відтворення, виконавцями, альбомами, піснями та жанрами.

Щоб повернутися до попереднього екрана, торкніться елемента керування меню внизу екрана.

На головному екрані відображається інформація про вибрану пісню, якщо музичний плеєр активний.

Підтримка DLNA

Наразі медіаплеєр підтримує технологію Digital Living Network Alliance (DLNA) із використанням Twonky DMS. Коли на вашій цілі ввімкнено DLNA, будь-які локальні пристрої DMS з’являться у списку доступних джерел медіа в меню.

Усі елементи керування відтворенням, включаючи функцію Skip and Seek, функціонують для пристроїв DLNA.

Запуск DLNA

DLNA не налаштовано на запуск за замовчуванням. Щоб дозволити сервісам запускатися при завантаженні, відредагуйте файл / var / etc / services-enabled та встановіть для параметра конфігурації DLNA значення true:

3.2. Використання репозиторію для зміни регіональних налаштувань

На прикладі зміни регіональних налаштувань розглянемо роботу з модулями репозиторію і збереженими конструкціями.

На рис. 3.8. представлено базову версію ПЗ мультимедійної інформаційної системи *Audi MIB*-2. Дане ПЗ має регіональне налаштування Америка.

Рис. 3.8. Регіональне налаштування системи Америка

Для переведення даного ПЗ на налаштування під Європу (рис. 3.9)необхідно з репозиторію ПЗ витягнути нову версію ПЗ для даного типу ГП (головного пристрою) та внести зміни в файлах даного репозиторію.

Рис. 3.9. Регіональне налаштування системи Європа

Приклад даної зміни представлено на рис. 3.10.

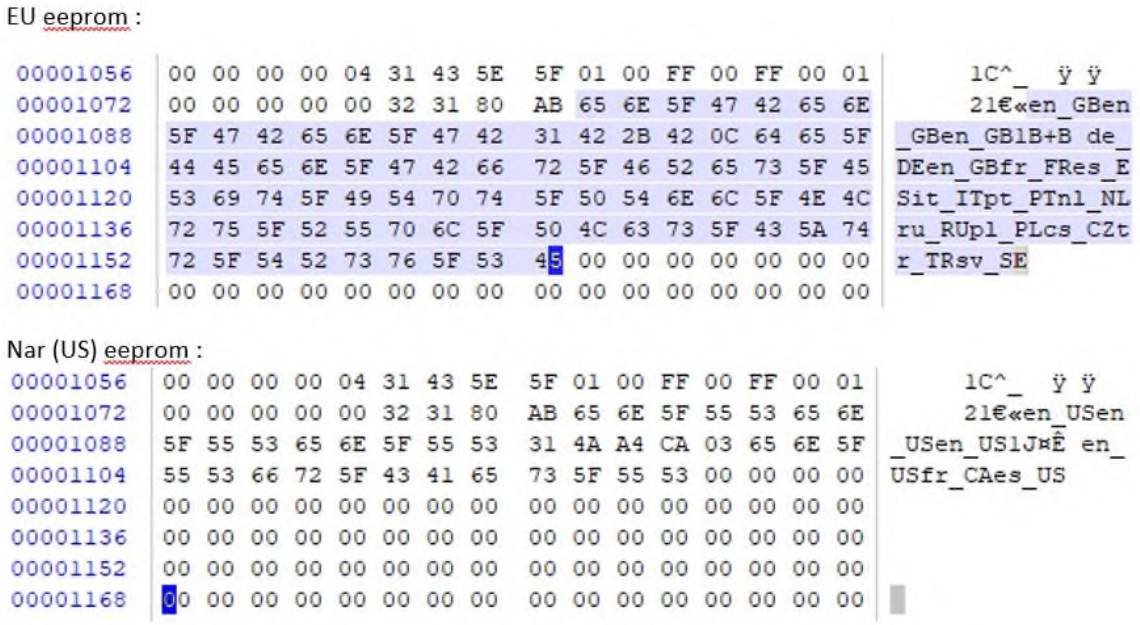

Рис. 3.10. Зміни регіональних налаштувань у файлі *eeprom*

Перевага використання репозиторію полягає у простоті повторення дій для різних ГП. На рис. 3.11 представлені приклади змін у різних ГП.

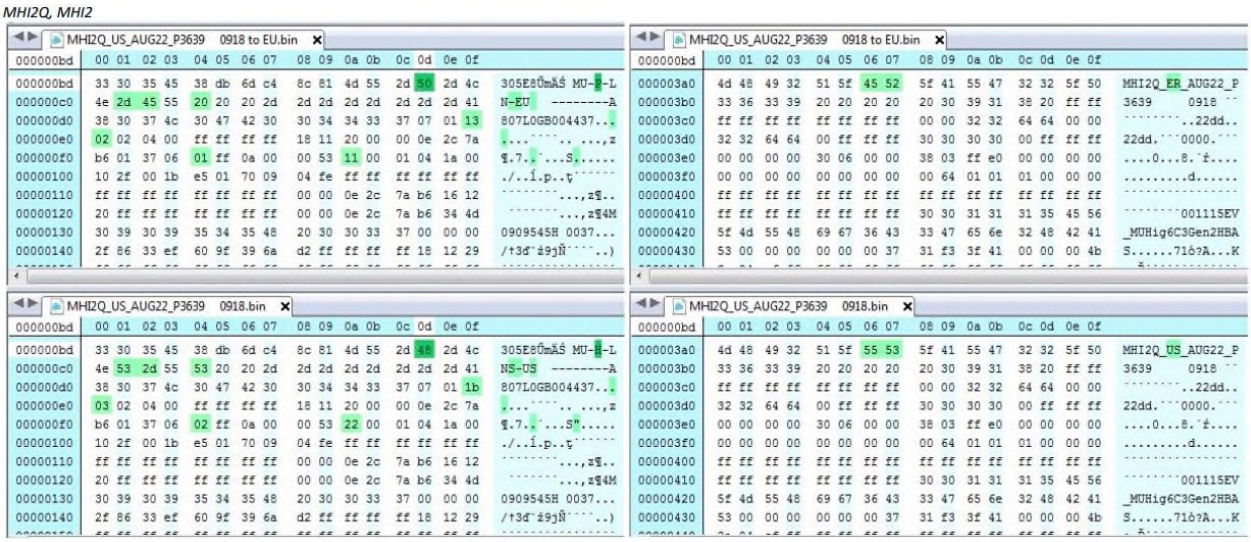

Рис. 3.11. Результати змін файлів *eeprom* на різних ГП

На рис. 3.12 показані спрацьовування даного модулю з репозиторію на різних версіях ПЗ різнотипних марок ГП.

Рис. 3.12. Результат оновлення програмного модулю

На рис. 3.13 показані переліки оновлених модулів.

Рис. 3.13. Перелік оновлених модулів

3.3. Висновки до розділу

Підчас розробки репозиторію ПЗ автомобільних мультимедійних систем були проведені обстеження та детальний аналіз предметної області, були створені функціональна, концептуальна, логічна та фізична моделі бази даних. Також були детально проаналізовані усі елементи, з яких складається даний

програмний продукт та залежності між ними. Для розробки даного програмного продукту було використано *MySQL* та *php* – відкриті, прозорі, безкоштовні та переносимі рішення.

Створена електронна база даних значно підвищує ефективність праці у процесі обліку програмного забезпечення, оскільки дозволяє швидко реалізувати можливості пошуку ПЗ: за автором, назвою, датою створення; можливість реєстрації дистрибутива, а саме фіксування таких даних, як: назва, версія, *ftp*-адреса дистрибутива; можливість реєстрації автора, а саме занесення до бази даних такої інформації: ім”я, по-батькові, та прізвище автора, його *email*-адресу; можливість реєстрації програмного забезпечення, а саме занесення до бази даних такої інформації: назву, версію, дату створення, ліцензію та *ftp*-адресу вихідних файлів програми, автора та дистрибутив із вже наявних у таблиці бази даних; можливість редагування інформації про вже зареєстроване програмне забезпечення; можливість переглядати статистичні дані про все наявне програмне забезпечення.

Програма має зручний інтерфейс та меню користувача, проста у використанні, враховує найважливіші потреби користувача, який працює з великою базою даних, має інструкцію для користувача. Програма максимально переносима оскільки потребує тільки підключення до мережі інтернет та браузер. Програма підтримує використання багатьма користувачами в реальному часі.

Отже, розроблене програмне забезпечення повністю задовольняє поставлені до нього вимоги. Може бути використане за призначенням як примітивний репозитарій та як приклад для інших проектів.

ВИСНОВКИ

В результаті роботи над дипломною роботою був проведений огляд літератури з метою пошуку існуючих методів оновлення версій ПЗ автомобільних мультимедійних систем.

Інформаційно-розважальна програма у традиційному розумінні – це засоби масової інформації, які передають споживачам інформативний та розважальний контент. Побутова електроніка, така як планшети та смартфони, змінила ландшафт інформаційно-розважального комплексу, пропонуючи платформу, яка забезпечує величезний обсяг персоніфікованих даних, будь то ціни на акції, інформація про погоду або останні меми. Автомобільна промисловість швидко рухається до адаптації цієї концепції до сучасних транспортних засобів. Інформаційно-розважальна система в автомобілі означає сучасну пропозицію автомобілів (зробіть глибокий вдих): підключені пристосовані до власного пристрою сумісні настроювані бездротові найсучасніші голосово-активовані сенсорні екрани з голосовою активацією, інтегровані мультимедійні системи,

Сьогодні на частку електроніки припадає близько 80% функціональних інновацій автомобільної галузі, і програмне забезпечення - це ключ до більшості з них. У міру того як ПЗ стає все більш істотною частиною вартості обладнання, в бізнес-моделях починають враховувати необхідність повторного використання та обміну програмним забезпеченням.

Високошвидкісні шини, такі як *Ethernet*, все ширше використовуються сьогодні в автомобілебудуванні для підтримки взаємодії між *ECU* і розробки нових функцій, особливо в області безпеки. Інформація з різних джерел аналізується і консолідується для формування повної моделі середовища, дозволяючи розробляти нові функції, які підтримують водія в критичних ситуаціях. Наприклад, якщо увагу водія відволікає пасажир, то програма може визначити, що їде попереду гальмує, і попередити про це водія або ж автономно включити гальмування. Водій ніколи не здогадається про існування такого програмного забезпечення, поки не виникне небезпечна ситуація.

Швидкий цикл розвитку побутової електронної промисловості спонукав споживачів очікувати нових, кращих та швидших функціональних продуктів протягом декількох місяців. На відміну від циклу розробки автомобільної промисловості, де найшвидший час, коли новий автомобіль переходить від дизайну до виробництва, становить близько двох років – і це його штовхає.

Для того, щоб залишатися конкурентоспроможними, *OEM*-виробники швидко працюють над впровадженням підключених функцій у нові транспортні засоби. Деякі виробники, такі як *Ford* та *Mercedes Benz*, зазнали критики за впровадження інформаційно-розважальних систем. *Mercedes Benz* критикували за інтеграцію *Facebook* та *Twitter* у їхню інформаційно-розважальну систему. *Ford* виявив недолік за програмні помилки та заплутаний користувальницький інтерфейс. Поспішаючи бути першим на ринку досвіду автомобілів *XXI* століття, виробники автомобілів ризикували зіткнутися з нестандартними, випадковими та непродуктивними.

На даний момент *In-Vehicle Infotainment* (*IVI*) представляється Диким Заходом автопрому. Різні партнерські відносини між виробниками обладнання та постачальниками програмного забезпечення, а також нестандартні користувальницькі інтерфейси (різні піктограми та жести для сенсорного екрану) відрізняються від фактичного стандарту, який *iOS* та *Android* створили у галузі смартфонів та планшетів.

В автомобілебудуванні сьогодні назріла чергова програмна революція - все ширше починають застосовуватися засоби мультимедіа та побутової електроніки. Автомобілі будуть підключатися до Інтернету і до всіх видів мобільних і встановлених будинку пристроїв, причому неухильно зростатиме частка рішень на базі вільного програмного забезпечення. Кожного дня з’являються нові версії ПЗ для автомобільних систем, а фахівці, що працюють з налагодженням електроніки змушені ставати фахівцями з встановлення і налаштування ПЗ. Саме тому сьогодні проблема відстеження версій і оновлень автомобільного ПЗ є доволі актуальною задачею.

На основі приведеного аналізу зроблено висновки щодо необхідності формування реєстру програм на хмарному сервері і перевірки їх оновлень. Даний

реєстр повинен бути розміщений на серверах з доступам через загальну або закриту мережу.

Результати дипломної роботи рекомендується використовувати при розробці програмних систем контролю версій автомобільного програмного забезпечення.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Andrew Patterson*. *Automotive infotainment systems*: *Open source drives innovation*. *Embedded Computing*. *Accessed*: 2015-03-12. [*Online*]. *Available*: *http*://*embedded-computing.com*/*articles*/*automotive-source-drives-innovation*/

2. *Michael Kerrisk* (2012-08-8). *GENIVI*: *Moving an industry to open source*. *Accessed*: 2015-02-11. [*Online*]. *Available*: *http*://*www.genivi.org* /*sites*/ *default* /*files* / *in-the-news*/2012 08 08 *GENIVI*%20 *Moving*% 20*to*%20*Open* %20*Source* %20-%20*Michael*%20*Kerrisk.pdf*

3. *BMW Case Study*. *GENIVI Alliance*. *Accessed*: 2015-02-19. [*Online*]. *Available*: *http*://*www.genivi.org*/*sites*/*default*/*files*/*BMW Case Study Download* 040914.*pdf*

4. *An Architecture for In-Vehicle Infotainment Systems*. *Dr. Dobb's*. *Accessed*: 2020-11-23. [*Online*]. *Available*: *http*://*www.drdobbs.com*/*embedded-systems*/*anarchitecture-for-in-vehicle-infotainm*/222600438

5. *Application software*. *Accessed*: 2020-11-23. [*Online*]. *Available*: *http*://*www.drdobbs.com*/*embedded-systems/Application software*

6. *Concepts of the Middleware*. *Accessed*: 2020-11-23. [*Online*]. *Available*: *http*://*www.drdobbs.com*/*Middleware*

7. *Theoretical Concepts of the Operating system*. *Accessed*: 2020-11-23. [*Online*]. *Available*: *http*://*theoretic_os.org*/*operating system*

8. *Analyze of the Board Support Package. Accessed*: 2020-11-23. [*Online*]. *Available*: *http*://*articles.bsp.org*/*board_support_package*

9. *Modern Device driver*. *Accessed*: 2020-11-23. [*Online*]. *Available*: *http*://*articles.bsp.org*/*Device driver*

10. *Doug Newcomb*. *The next big OS war is in your dashboard*. *Wired*. *Accessed*: 2015-03-12. [*Online*]. *Available*: *http*://*www.wired.com*/2012/12/*automotive-oswar*/*all*/ 50

11. *Lucas Mearian*. *Your car is about to go open source*. *Accessed*: 2015-01-17. [*Online*]. *Available*: *http*://*www.computerworld.com*/ *article*/ 2485817/ *emergingtechnology*/*your-car-is-about-to-go-open-source.html*

12. *GENIVI*. *GENIVI Alliance*. *Accessed*: 2015-02-11. [*Online*]. *Available*: *http*: //*www.genivi.org*

13. *AGL*. *Automotive Grade Linux* (*AGL*). *Accessed*: 2015-02-20. [*Online*]. *Available*: *https*://*www.automotivelinux.org*/

14. *AGA Wiki*. *Automotive Grade Android* (*AGA*). *Accessed*: 2015-03-15. [*Online*]. *Available*: *https*://*developer*.*lindholmen*.*se*/*redmine*/*projects*/*aga*/*wiki*

15. *In-Vehicle Infotainment* (*IVI*). *Techopedia*. *Accessed*: 2015-03-22. [*Online*]. *Available*: *http*://*www.techopedia.com*/*definition*/27778/*in-vehicle-infotainmentivi*

16. Молинаро Энтони. *SQL*. Сборник рецептов / Энтони Молинаро. – *O'Reilly,* 2009. – 672 с.

17. Оппенгейн А.В., Шафер Р.В. Цифрова обробка сигналів, М.: Радіо та зв'язок, 1979., 347 с.

18. Павловская Т.А. *C#*. Программирование на языке высокого уровня: Учебник для вузов / Т.А. Павловская. – Питер, 2009. – 432 с.

19. Рихтер Джеффри. *CLR via C#*. Программирование на платформе *Microsoft.NET Framework* 4.5 на языке *C#*. 4-е изд. / Джеффри Рихтер. – Питер, 2013. – 896 с.

20. Гуляєв А.О. Система контролю версій автомобільних мультимедійних систем // Тези доповідей наук.-практ. конф. "Сучасні тенденції розвитку системного програмування" (25-26 листопада 2020 р.). – К.: НАУ, 2020. – С. 7.

21. ДСТУ 3008-95 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

22. НД ТЗІ 1.1-003-99. Термінологія у області захисту інформації в комп'ютерних системах від несанкціонованого доступу. // Департамент спеціальних телекомунікаційних систем і захисту інформації Служби безпеки України. – Київ, 1999.

23. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

Додаток А

Схема алгоритму обробки запиту на пошук версій ПЗ

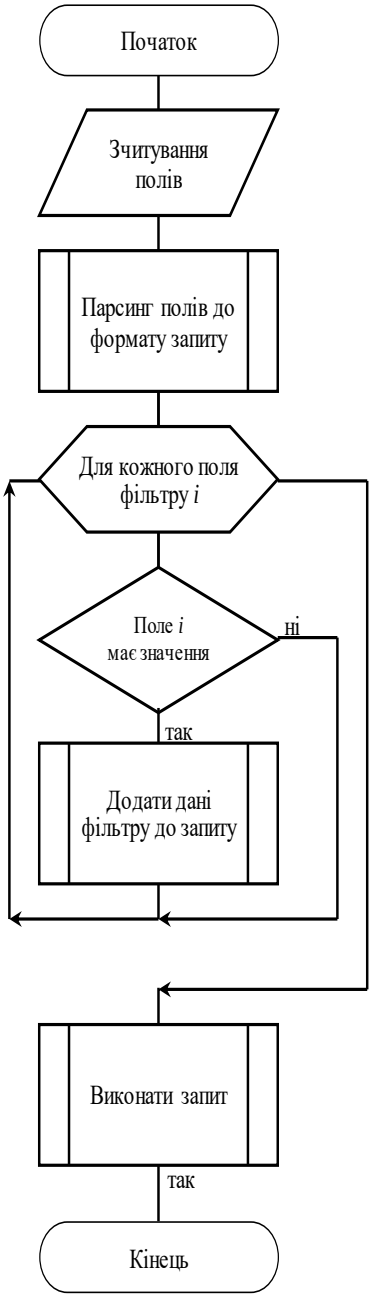

Рис. А.1. Схема алгоритму обробки запиту на пошук версій ПЗ