Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Система автоматичного керування автомобільним транспортним засобом
(тема)

Виконав: студент 2 курсу, групи СКСм-19-1
Кривицький А.О.
(прізвище, ініціали)
Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва спеціальності)
Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Спеціалізовані комп'ютерні системи
(повна назва освітньої програми)
Керівник
доц. Філіппенко І.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С.В.
(підпис) (прізвище, ініціали)

2020 р.

Факультет Комп'ютерної інженерії та управління
Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти другий (магістерський)
Спеціальність 123 – Комп’ютерна інженерія
Тип програми Освітньо-професійна
Освітня програма Спеціалізовані комп’ютерні системи
Харківський національний університет радіоелектроніки

ЗАТВЕРДЖУЮ:
Зав.
кафедр
и _____
(підпис)

_____ _____2020 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Кривицькому Андрію Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Система автоматичного керування автомобільним транспортним

затверджена наказом по університету від 04 30 жовтня 2020 р. 1489Ст.
2. Термін подання студентом роботи (проекту) 13 грудня 2020 р.
3. Вихідні дані до роботи (проекту)
Апаратна платформа – вбудовані комп’ютери на архітектурі ARM
Підтримка актуаторів та сенсорів
Зовнішній інтерфейс взаємодії з клієнтськими додатками
Середа розробки Eclipse

4. Перелік питань, що потрібно опрацювати в роботі
Аналіз предметної області
Розробка алгоритму керування
Розробка архітектури апаратної платформи
Вибір компонентів
Написання програмної реалізації
Тестування та експериментальні дослідження

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)
14 слайдів

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		(підпис)	(дата)

КАЛЕНДАРНИЙ ПЛАН

№ п./п.	НАЗВА ЕТАПІВ РОБОТИ	ТЕРМІН ВИКОНАННЯ ЕТАПІВ РОБОТИ	Примітка
1	Видача теми проекту, узгодження і	03.09.2020 - 05.09.2020	
2	Аналіз предметної області та опрацювання	06.09. 2020 - 06.10.2020	
3	Розробка структури та алгоритмів для програмно-аппаратної платформи	7.10. 2020 - 13.10.2020	
4	Розробка програмного забезпечення	14.10.2020 - 25.10.2020	
5	Тестування та випробування програмної реалізації	28.10.2020 - 08.11.2020	
6	Оформлення пояснювальної записки	11.11.2020 - 29.11.2020	
7	Перевірка виконаного проекту керівником, допуск до захисту	2.12.2020 - 10.12.2020	
8	Захист проекту	13.12.2020 -22.12.2020	

Дата видачі завдання 3 вересня 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Філіппенко І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 54 сторінки, 14 рисунків, 4 таблиці, 17 джерела за переліком посилань.

АВТОНОМНИЙ АВТОМОБІЛЬ, АВТОНОМНИЙ ТРАНСПОРТ, ПРОГНОЗУВАННЯ, ПАРАЛЕЛЬНА ПРОГРАМА, ОПТИМІЗАЦІЯ, ВИСОКОПРОДУКТИВНА СИСТЕМА

Метою роботи є розробка моделі система автоматичного керування автомобільним транспортним засобом. В ході роботи були розглянуті сучасні реалізації систем у відомих автовиробників та підходи до реалізації систем автопілоту, питання безпеки. Проаналізовані можливості імплементації систем на двох типах апаратних платформ. Порівняні швидкодії моделей на різних імплементаціях.

Була розроблена модель та реалізація программно-аппаратного комплексу для управління транспортним засобом. Описана модель предметної області, модель блоку керування, створені абстракції та представлення пристроїв, інтеграції та інтерфейс програмної взаємодії (API), що використовуються для реалізації проекту.

На основі моделей та методів було реалізовано систему автоматизації керування транспортним, засобу.

ABSTRACT

Explanatory note contains 54 pages, 14 figures, 4 tables, 17 sources.

AUTONOMUS AUTOMOBILE, AUTONOMUS VEHICLE, FORECASTING, PARALLEL PROGRAM, OPTIMIZATION, HIGH-PRODUCTION SYSTEM, ARM, FPGA

The aim of the work is to develop a model of automatic vehicle control system. In the course of work modern implementations of systems at known automated manufacturers and approaches to realization of autopilot systems, safety questions were considered. Possibilities of system implementation on two types of hardware platforms are analyzed. Compared performance of models on different implementations.

A model and implementation of a software and hardware complex for vehicle control was developed. Describes the model of the subject area, the model of the control unit, created abstractions and representations of devices, integration and software interaction interface (API) used to implement the project.

On the basis of models and methods the system of automation of management of the vehicle was developed.

ЗМІСТ

ВСТУП

Сучасна автомобільна промисловість, як і будь-яка інша високотехнологічна галузь, розвивається у бік спрощення користування пристроєм і підвищення безпеки. Одним з найяскравіших представників такого руху є автомобільні автопілоти. Оскільки комп'ютер не може відчувати погіршення здоров'я, або бути втомленим, безпека дорожнього руху, за умови повсюдного впровадження автономних систем, може вийти на принципово новий рівень.

Основними завданнями автомобільного автопілота є визначення швидкості руху, напряму, визначення перешкод і позиціонування автомобіля в просторі.

Позиціонування в просторі відбувається завдяки технології GPS і бортових систем машини, а саме – камер і радарів. Камера, за умови наявності зазначених параметрів, може дати дуже багато інформації про положення транспортного засобу. Додаткове комбінування камери з радаром дає достатньо інформації для точного позиціонування транспортного засобу не лише відносно смуги руху а й відносно інших транспортних засобів та дорожньої обстановки. Використання добре навченої нейронної мережі разом з аппаратним комлексом зазначеним вище надає можливість точно та швидко реагувати на дорожнью ситуацію.

Реагування на дорожню обстановку є першочерговою задачею при реалізації програмно-аппаратного комплексу системи автоматичного керування автомобільним транспортним засобом, саме тому весь комплек має бути макимально швидким та точним при цьому, має навчатись на попередньо виявлених типових дорожніх ситуаціях.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальна характеристика автономних транспортних засобів

Автономні автомобілі – це автотранспортні засоби, в яких керування здійснюють внутрішні системи транспортних засобів, а не водій-людина. Вони виконують усі функції у міру руху транспортного засобу по дорогах загального користування. Такі транспортні засоби можуть набувати форми легкових автомобілів, великих або малих вантажівок, автобусів або інших видів наземного транспорту.

Автомобілі з автопілотом можуть перевозити або вантажі, або пасажирів, або і те, і інше, або ні те, ні інше. Різноманітна термінологія, що стосується автотранспортних засобів без водіїв, ускладнює розробку політики відносно таких транспортних засобів. Іноді ці транспортні засоби називають "повністю самокерованими", "повністю автономними" або навіть "повністю автоматизованими", на додаток до того, що їх доречніше називати "безпілотними транспортними засобами".

Негативна ознака відсутності водія повторює назву "безкінних возів" для автомобілів більше ста років тому. Обидві фрази припускають зручне перетворення з чогось знайомого в щось нове, що, по суті, є таким, що перетворюється. Деякі з тих же технологій, що використовуються в автомобілях без водія, забезпечують функції автоматизації в звичайних транспортних засобах. Але автоматизовані автомобілі не обов'язково є безпілотними. Вже наявні автоматизовані, напівавтономні або самокеровані системи допомагають водіям, контролюючи всі або деякі операції транспортних засобів.

Технології автоматичного управління допомагають виконувати певні дії використовуючи конкретні системи транспортного засобу, такі як гальмування або парковка, проте для управління основними операціями транспортного засобу як і раніше потрібна присутність водія-людини. Крім того, міжнародні конвенції і, наприклад, закони Сполучених Штатів прямо

вимагають, щоб водій керував транспортним засобом, що рухається на дорогах загального користування.

Різний вигляд автоматизованих, самокерованих, або напівавтономних систем управління, що мається нині, на звичайних транспортних засобах дасть розробникам досвід застосування деяких з видів технологій, які також застосовуються в транспортних засобах без водія. Експлуатація на дорогах цих обмежених можливостей автопілотування, автономних режимів роботи або автоматизованих операцій дозволить отримати дані, необхідні для ухвалення рішень з питань правової політики відносно транспортних засобів, які працюють виключно без водіїв-людей. Проте правові і політичні питання, які виникають у зв'язку з використанням транспортних засобів без водія, істотно відрізняються від питань, що виникають у зв'язку з транспортними засобами, які не повністю обходять стороною оператора.

1.2 Класифікація автономності автомобілів

Автомобілі, що не мають водія, забезпечують дорожні перевезення як для вантажів, так і для людей, при цьому здійснюють і контролюють власні операції і пересування. Автомобілі без водія, як очікується, будуть безпечніші, ефективніші і екологічніші, ніж звичайні автомобілі, керовані водіями. Вважається, що вони врятують тисячі життів і мільйони доларів завдяки уникненню аварій.

Існуючі нормативні вимоги до механічних транспортних засобів (такі, як велике устаткування пасивної безпеки, облаштування обмеження забруднення і так далі) можуть привести до того, що найбільш ранні безпілотні транспортні засоби можуть зовні виглядати схожими на звичайні транспортні засоби. Врешті решт, транспортні засоби без водія, ймовірно, виглядатимуть зовсім інакше, ніж автомобілі, керовані людиною.

Можливості автомобілів без водія із запобігання аварій повинні усувати необхідність в дорогих технологіях пасивної безпеки, таких як важкі

матеріали, бампери, подушки безпеки або навіть вітрові стекла меншого розміру. Проте, перш ніж такі фізичні зміни зможуть статися, необхідно провести велику кількість правових і нормативних змін.

Різні види застосування автоматизованих систем транспортних засобів – від електронного контролю стійкості до автоматичної системи утримання смуги руху, парковки і гальмування – дозволяють транспортним засобам виконувати конкретні завдання без втручання людини. Нещодавно розроблені автоматизовані системи, які контролюють деякі або усі операції з транспортними засобами на частині шляху або в конкретному дорожньому середовищі, також стають доступні в комерційних транспортних засобах.

В той же час нині в цілому контроль здійснюється діяльністю водіїв, особливо в надзвичайних ситуаціях. "Super Cruise" від General Motors і обіцяний Tesla "Autopilot"- це фірмові реалізації автоматичного управління, що рекламуються як самостійні або самокеровані автомобілі. Проте, навіть такі високоавтоматизовані функції автомобіля і режими автопілотування не дозволяють автомобілям з такими функціями бути безпілотними, тобто повністю позбавлятися від водія.

Водій-людина як і раніше має бути присутньою на цих автомобілях як на законних підставах, так і практично, і він може здійснювати позитивний оперативний контроль за ними. Навпаки, безпілотні транспортні засоби експлуатуються без якого-небудь контролю з боку людини і тому є правовими і політичними питаннями, відмінними від тих, які виникають у разі транспортних засобів, які як і раніше покладаються на присутність водія-людини.

Для багатьох просунутих розробників автомобільних систем "автономність" стала досить неоднозначною, щоб органи стандартизації і регулювання не застосували її на користь "ряду засобів автоматизації". Підвищення міри автоматизації транспортних засобів, в зворотному зв'язку з управлінням людьми, представляється корисним в описі усе більш складних етапів технологій автоматизації транспортних засобів.

Насправді існує два варіанти рівнів автоматизації транспортних засобів. Для обох з них, безпілотні транспортні засоби знаходяться на найвищому рівні повної автоматизації – це означає, що автомобіль повністю контролює усі функції управління у будь-який час. У 2013 році Національна адміністрація безпеки дорожнього руху (НАБДД) США [12] запропонувала рівні автоматизації транспортних засобів в підготовленому агентством "Попередній заяві про політику відносно автоматизованих транспортних засобів". У січні 2014 року Міжнародне суспільство інженерів автомобільної промисловості (SAE) запропонувало дещо інший варіант "Таксономії і визначень термінів, що стосуються автоматизованих системи керування транспортних засобів", як стандарт SAE J3016.185 [11] Два конкуруючі набори категорій або рівнів:

1) рівні автоматизації SAE:

рівень 0 – без автоматизації;

рівень 1 – допомога водію;

рівень 2 – часткова автоматизація;

рівень 3 – автоматизація умов;

рівень 4 – висока автоматизація;

рівень 5 – повна автоматизація [без водія];

2) рівні автоматизації NHTSA:

рівень 0 – без автоматизації;

рівень 1 – автоматизація для певниї функцій;

рівень 2 – автоматизація комбінованиї функцій;

рівень 3 – обмежена автоматизація;

рівень 4 – повна автоматизація без керування.

Слід зазначити, що автомобілі без водія займають найвищий рівень автоматизації в обох системах. У категорії NHTSA наявні нині технології автоматизації транспортних засобів знаходяться на рівні 2 і швидко переходять на рівень 3, але ще не наблизилися до верхнього рівня автоматизації NHTSA 4 – робота без втручання людини.

Аналогічним чином, існуюча автоматизація транспортних засобів нині знаходиться між рівнями SAE 2 і 3. Федеральні і державні регулюючі органи в Сполучених Штатах зазвичай посилаються на рівні автоматизації транспортних засобів NHTSA. Виробники транспортних засобів часто використовують категорії SAE, які аналогічні категоріям автоматизації транспортних засобів, що використовуються в Європі.

1.3 Складові програмно апаратної платформи

Автономний транспортний засіб є складною програмно-апаратною системою, основними складовими якої наступні компоненти.

Сховище даних – це твердотілі або портативні накопичувачі для використання в автомобілях і центрах обробки даних. Спеціалізована конструкція для використання в транспортних засобах.

"Drive-by-wire" – електроніка і контролери, що забезпечують взаємодію між обробкою і роботизованим управлінням автомобілем. Управління, гальмування, прискорення здійснюється за допомогою дротяних інтерфейсів для конкретних транспортних засобів або за допомогою протоколів відкритого стандарту.

Позиціонування – включає супутникове позиціонування, вимір інерції.

Апаратне забезпечення, що здійснює обробку даних – спеціалізоване апаратне забезпечення або одного із вже з існуючих виробників, або спеціалізоване для певного виробника.

Програмне забезпечення – алгоритмічна частина, що сполучає усе вищевикладене і датчики в підсумкову систему.

Також система повинна бути обладнана датчиками, такими як:

камера – найпоширеніший датчик машинного зору, що імітує людське око;

LiDAR – інфрачервоний хвилевий лазерний випромінювач і сенсорна система для побудови тривимірної карти оточення;

радар – короткохвильовий, високоенергетичний сенсор, пристосований для проникнення в погану погоду і мандрівок великої дальності;

ультразвукові сенсори – високочастотний аудіосигнал є найкращим датчиком малого радіусу дії для виявлення близьких об'єктів.

2 СКЛАДОВІ АПАРАТНОЇ ПЛАТФОРМИ

2.1 Спеціалізовані апаратні платформи

Багато автовиробників анонсували або демонструють самокеровані автомобілі. Але більшість з цих компаній не створюють свої власні самокеровані апаратні і програмні системи; вони або придбали спеціалізовані фірми для створення таких систем, або передають роботу на аутсорсинг третім особам. Деякі з найбільш шанованих лідерів в цьому зростаючому сегменті ринку знаходяться в Сполучених Штатах і Канаді.

Сьогодні багато говорять про автономні транспортні засоби та значна кількість прототипів легкових і вантажних автомобілів нині знаходиться на дорогах загального користування. Існують також повністю функціонуючі автономні робо-таксі сервіси, що перевозять пасажирів по різних регіонах світу, іноді без водія.

Але для усіх розмовних і тестових автомобілів на ринку кількість реальних "платформ" – інтелектуальних технологій в основі – залишається обмеженою. Причина цього проста – створення цих платформ є складним процесом, що вимагає значної капіталізації, ретельного проектування систем і ретельного тестування. На даний момент ця робота краще підходить технологічним компаніям (що мають досвід роботи в області комп'ютерного устаткування і програмного забезпечення), а не автовиробникам.

Це пояснює, чому деякі з найбільших автомобільних компаній у світі або віддали на аутсорсинг свої технологічні платформи, або придбали компанії, що спеціалізуються на їх створенні.

Розуміння обмежень цієї технології привело до того, що багато виробників зосередили свої зусилля на постачаннях автомобілів з робо-таксі, а не на виробництві автомобілів, щоб отримати цінний досвід реального

тестування. Ще однією причиною такої уваги до автопарків і послуг таксі є те, що в майбутньому індивідуальна мобільність може зменшитися на користь мобільності між декількома людьми або групами.

Деякі компанії вирішили співпрацювати або обмінюватися даними зі своїми колишніми конкурентами, щоб знизити витрати на розробку своїх власних автономних автомобільних платформ. Це привело до створення часто заплутаної мережі союзів, ліцензійних угод і інвестиційних партнерствах в автомобільній промисловості.

Наступні компанії розробили широкий спектр AV- платформ для різних виробничих цілей. Для деяких фірм вироблювані автомобілі для споживачів є основною цільовою сферою застосування. Для інших основним напрямом може бути автономний рух вантажівок, приміські поїздки, доставка середньої милі або низько швидкісні шатли.

2.1.1 NVIDIA

NVIDIA DRIVE AGX – це масштабована, відкрита автономна обчислювальна платформа, яка служить мозком для автономних транспортних засобів (рис 2.2) [16].

Рисунок 2.2 – Тестовий автомобіль Nvidia

Спеціалізований виробник комп'ютерного устаткування NVIDIA вже деякий час співпрацює з фірмами, котрі розроблюють автономні транспортні засоби [13]. За станом на 2020 рік Reuters повідомляло, що компанія має більше 370 окремих бізнес-клієнтів для своїх продуктів AV платформи. Серед цих клієнтів Audi, Daimler, ZF і Volkswagen, Aurora, Uber ATG і Baidu.

У 2019 році компанія NVIDIA показала на виставці CES в Лас-Вегасі першу комерційну відкриту платформу SAE Level 2+ AV під назвою AutoPilot. "+" означає, що платформа підтримує функції, що вимагають

глибокого навчання, такі як моніторинг водія і сприйняття навколишнього простору.

Як заявив Роб Ксонгор, віце-президент NVIDIA по автономних автомобілях: "Повнофункціональна система рівня 2+ вимагає значно більшої обчислювальної потужності і складного програмного забезпечення, ніж система, існуюча сьогодні. NVIDIA AutoPilot надає ці можливості, що дозволяє виробникам автомобілів швидко розгорнути передові автономні рішення до 2020 року і швидше масштабувати це рішення до більш високих рівнів автономності".

Платформа AutoPilot включає високошвидкісний процесор NVIDIA Xavier з низьким енергоспоживанням і паралельний процесор SoC та програмне забезпечення DRIVE AI. Вони дозволяють платформі обробляти глибокі нейронні мережі (DNN) для сприйняття з датчиків камери, встановлених на автомобілі і усередині нього. До числа таких DNN входять SignNet, DriveNet, OpenRoadNet, LaneNet і WaitNet. Ці DNN допомагають системі DRIVE зрозуміти, де знаходяться інші транспортні засоби, зчитувати дорожню розмітку, визначати пішоходів і велосипедистів, розрізняти різні типи вогнів та кольорів, розпізнавати дорожні знаки і розуміти складні умови водіння.

У свою чергу, автомобіль може виконувати функції автономного управління, такі як об'єднання шосе, зміну смуги руху, розділення смуги руху і індивідуальне картування. Також в комплект входить програмне забезпечення DRIVE IX для моніторингу і сповіщення водіїв, копіювання ШІ і візуалізації комп'ютерного зору платформи в салоні.

Крім того, NVIDIA пропонує платформу DRIVE Constellation для центрів обробки даних. Сервер симулятора використовує графічні процесори NVIDIA, що працюють під управлінням програмного управління Drive Sim. Другий сервер транспортного засобу містить комп'ютер Drive AGX, який обробляє дані змодельованого датчика. Рішення, що приймаються на автомобільному сервері, вирушають на симулятор для проведення точного і

побітового тестування устаткування в петлі. Для роботизованого таксі NVIDIA створила більш швидку платформу DRIVE AGX Pegasus, яка може бути використана для забезпечення функціональності SAE рівня 5 AV.

Система DRIVE від NVIDIA – це "мозок" для автомобілів з штучним інтелектом від компанії Optimus Ride, що базується у Бостоні, заснованої ветеранами програм MIT з виробництва електромобілів і самокерованих технологій. Optimus Ride має групу з автономних транспортних засобів, працюючих в невеликих районах, таких як житлові громади, промислові парки, шкільні городки, аеропорти, змішані розробки і порти. Нині компанія працює в п'яти американських регіонах: Бостон, морський район ОЭ; Південний Уэймут, центр програми ОЭ; Бруклін, Нью-Йорк; Рестон, "Брукфілд Хеллі Райз" Вірджинія; і Райській долини, райські долини Центральної Азії.

Деякі компанії, такі як шведська Volvo (належить китайській Geely), використовують усі технологічні компоненти NVIDIA. Інші, такі як німецькі Bosch, ZF і Continental, використовують тільки деякі з них в якості частин своїх власних платформ AV. Деякі виробники також використовують технологію NVIDIA для персоналізації автомобіля, управління енергоспоживанням і до мережі при зарядці елементів живлення.

З 2017 року Bosch і німецький автовиробник Daimler працюють спільно з NVIDIA над автономною платформою, керованою Pegasus, з первинною метою виробництва автомобілів SAE рівнів 4 і 5 для індивідуальних споживачів. Проте з часом обидві компанії зрозуміли, що ця мета може бути нереалістичною, принаймні в короткостроковій перспективі.

Замість цього фірми створюють службу доставки для мобільних застосувань 4 рівні в Сан-Хосе, Каліфорнія. Ця служба використовуватиме досвід, отриманий на випробувальному полігоні Daimler площею 100 000 квадратних метрів в Еммендінгині, Німеччина.

Врешті-решт, вважається, що, коли сервіс з автономних автомобілів таксі зарекомендує себе в Каліфорнії, Daimler зможе узяти платформу і

перетворити її на пакет, орієнтований на сторонніх операторів таксі. У свою чергу, є надія, що ці оператори куплять самокеровані автомобілі у Daimler.

2.1.2 Tesla

Усі нові автомобілі Tesla (рис 2.3) поставляються в стандартній комплектації з вдосконаленим устаткуванням, здатним надати функції автоматичного управління вже сьогодні та стати повністю автопілотованими в майбутньому – за допомогою оновлень програмного забезпечення, розробленого для поліпшення функціональності з часом.

Рисунок 2.3 – Ілюстрація роботи датчиків автомобіля Tesla

Ілон Маск, генеральний директор і співзасновник Tesla, заявив, що до кінця 2020 року його компанія буде мати на дорогах до мільйона автомобілів SAE рівня 5 (повністю автономних), які будуть перебувати у спільному використанні, але при цьому являтися індивідуальною власністю.

Проте багато галузевих оглядачів рахують заяву І.Маска диким перебільшенням. І.Маск відтоді уточнив свої слова, сказавши, що повна автономність здійснюватиметься окремо у кожному конкретному випадку через відмінності в правилах країни і штату.

Хоча компанія Tesla, безумовно, проводить велику кількість тестувань автоматичних засобів, стан і удосконалення її платформи, що виходить за рамки того, що вбудовано в сучасні моделі Tesla, залишаються під сумнівами.

Проте невідомо, чи буде Tesla ділитися або ліцензувати свою платформу Autopilot з іншими виробниками автомобілів в майбутньому. Autopilot нині надає функціональність SAE рівня 3 (умовна автоматизація) в 2020 модельному році автомобілів Tesla.

У 2019 році при проведенні заходу, який Tesla називає Днем Автономії, компанія заявила, що комп'ютерний компонент другого покоління з повним приводом свого Autopilot перевершує платформу Pegasus від NVIDIA.

Продуктивність системи Тесла є дуже близькою до Pegasus від NVIDIA. На даний момент, принаймні, здається, що ці дві компанії дуже схожі в продуктивності. (Tesla раніше використовувала компоненти NVIDIA, до розробки власних.)

2.2 Протоколи передачі даних

Частина підключених до мережі транспортних засобів швидко росте. До 2020 року, за прогнозами галузевих аналітиків, на дорогах буде 250 мільйонів транспортних засобів підключених до мережі і 10 мільйонів самокерованих автомобілів, а до 2025 року – 470 мільйонів підключених/автономних транспортних засобів.

Рисунок 2.4 – Графіки Big Data у автономних ТС

Аналітики Statistica підготували графіки (рис. 2.4) на основі даних з оцінки McKinsey, згідно якої підключені автомобілі створюють до 25 гігабайт даних в годину. Це еквівалентно майже 30 годинам відтворення відео високої чіткості і більш ніж місячної вартості 24-годинної потокової музики.

У міру збільшення числа підключених автомобілів росте і обсяг виробництва, передачі і отримання даних. Компанія McKinsey & Company повідомляє, що об'єм даних, що передаються через підключений транспортний засіб (а також в хмару і з нього), складає приблизно 25 Гбайт даних в годину, і прогнозує, що ця цифра зросте до майже 500 Гбайт даних в годину після того, як транспортні засоби стануть дійсно автономними.

У сучасних автомобільних мережах фактично використовується комбінація декількох різних протоколів передачі даних. Деякі з них існують вже декілька десятиліть. Серед них – мережа контролерів (CAN), яка виконує функції силового агрегату і приводу; локальна мережа міжзв'язків (LIN), переважно робота з додатками, не залежними від часу, такими як клімат-контроль, освітлення навколишнього повітря, регулювання сидінь і так далі; перенесення системи мультимедіа (MOST) для розваг; і FlexRay для антиблокувального гальмування, електронного підсилювача рульового управління і забезпечення стійкості автомобіля.

Приклад використання Ethernet в автомобільній промисловості: Шлюз Molex 10 Gb Ethernet з підтримкою 25 Гбіт/с і рішенням Molex HSAutoGig. Він підтримує поля цілісності сигналу, необхідні для автомобільних датчиків, камер, шлюзів і перемикачів, які є важливими компонентами майбутніх автономних автомобілів.

Одним з результатів використання різних протоколів є те, що мережі транспортних засобів повинні включати шлюзи для передачі даних усередині інфраструктури. Ця практика не є оптимальною на декількох рівнях, один з яких полягає в тому, що проводка для кожної мережі додає вагу автомобілю. Дротяні джгути є третім за важкістю елементом типового автомобіля (важче тільки двигун і шасі).

Автомобільні мережі сьогодні також включають численні електронні блоки управління (ЭБУ), і кількість ЭБУ в типовому автомобілі також росте. Нерідкі випадки, коли на дорогих моделях налічується 150 електронних обчислювальних пристроїв (ECU), а на звичайних автомобілях – до 90.

Автоконструктори дійшли висновку, що додатки, що інтенсивною мірою використовують дані і підтримують вдосконалену систему допомоги водіям (ADAS), не можуть бути оброблені з використанням сучасних технологій автомобільних мереж. Увесь підхід до створення мереж усередині транспортних засобів повинен докорінно змінитися як з точки зору топології, так і з точки зору базової мережевої технології.

Сьогодні мережева структура, використовувана в транспортних засобах, рухається у бік варіанту так званої доменної архітектури. Архітектура домена характеризується різними доменами для кожної ключової функції: один блок управління ECU і мережею, один блок ECU для інформаційно-розважальних програм, один блок для телематики, один блок живлення і так далі У міру ускладнення мереж цей підхід на основі домена стає менш ефективним. Таким чином, станеться перехід від доменної архітектури до типу, що називається зональною архітектурою.

Зональна архітектура сполучає дані з різних традиційних доменів з одним і тим же ECU на основі місця (зони) розташування ECU в автомобілі. Ця схема значно зменшує кількість дроту, тому що багато функцій, які тепер обробляються з дискретною проводкою, перейдуть до технології Ethernet.

На відміну від інших мережевих протоколів для транспортних засобів, Ethernet має чітку схему розвитку для досягнення більш високих швидкостей. На відміну від цього, традиційні автомобільні протоколи, такі як CAN і LIN, знаходяться в точці, де плановані застосування перевищують свої можливості без чіткого шляху розвитку для вирішення проблеми.

Очікується, що велика частина передачі даних усередині транспортних засобів здійснюватиметься через Ethernet. Таким чином, планується створити єдину однорідну мережу на усьому транспортному засобі. Ця мережа в автомобілі буде масштабована таким чином, що вона буде здатна обробляти функції, якы вимагають більш високих швидкостей (наприклад, 10 Гігабіт в секунду), з наднизькою затримкою, але також буде здатна обробляти повільніші функції. Розробники вибирають фізичний рівень Ethernet (PHY) для виконання певних функцій відповідно до вимог до смуги пропускання. Таким чином, датчики зображення з великим об'ємом даних, такі як радар і лідар, можуть використовувати інтерфейс 1 Гбіт/с, де датчики з низькою швидкістю передачі даних можуть потребувати тільки підключення 10 Мбіт/с.

Тенденція до використання мереж транспортних засобів привела до спрямування зусиль на стандартизацію автомобільного Ethernet. Автомобільна версія тепер називається стандартом IEEE 802.3ch для високошвидкісних автомобільних додатків Ethernet (Multi - Gig Automotive Ethernet на 2.5g, 5g і 10g) з додатковим PoDL. Перша версія цього стандарту вийде до кінця цього року. Це продукт Multi - Gig Automotive Ethernet Task Force (NGAUTO), який був створений для визначення конкретних характеристик продуктивності і роботи на швидкості 2,5 Гбіт/с, 5 Гбіт/с і 10 Гбіт/с.

IEEE також використала дослідницьку групу у рамках робочої групи 802.3 Ethernet для автомобільної мережі Ethernet із швидкістю більше 10 Гбіт/с. Цей порівняно недавній розвиток, і, ймовірно, пройде деякий час, перш ніж з'явиться який-небудь стандарт для Ethernet 10 Гбіт/с.

Ще однією тенденцією, що стосується підключених транспортних засобів, є перехід на 5g. Великі оператори зв'язку почали активувати безпровідні мережі п'ятого покоління (5g) в містах. Завдяки появі безпровідних даних 5g, інтелектуальне і автономне управління автомобілем може піднятися на новий рівень завдяки більш високій швидкості передачі даних, захищеному з'єднанню типу "автомобіль-автомобіль" і з'єднанню типу "автомобіль-інфраструктура". Планована до широкого впровадження впродовж декількох років, 5g потребує постійного прогресу в розвитку потужної мережевої інфраструктури і технологій внутрішньо корабельної обробки для забезпечення надійної швидкості передачі сигналу з наднизькою затримкою смуги пропускання.

Для пояснення, що 5g може означати для автомобільного зв'язку, може бути корисно навести реальні приклади можливостей 5g, які нині знаходяться у стадії прототипу. Одним з таких проектів є AutoAir, який проходить випробування на полігоні Міллбрук у Великобританії. Дослідники встановили базові станції, антени і інше устаткування для створення

сценаріїв, що включають підтримку з'єднання на швидкості 1 Гбіт/с з автомобілями, швидкість яких досягає 160 миль/г.

Ще одна реалізація 5g під назвою Invisible - to - Visible була описана на CES 2019 компанією Nissan. Він використовує швидкий зв'язок, щоб бачити по кутах і попереджати водія про потенційну небезпеку перешкод на дорозі або пішоходів, прихованих транспортними засобами. Аналогічно, Ford Motor працює з Vodafone над системою, яка використовує 5g для попередження автомобілів про наближення до аварійних ділянок.

Ці високошвидкісні автомобільні рішення вимагають досвіду і інженерних навичок, які перевершують багато інших сфер застосування. Багато в чому збільшення складності конструкції досягається за рахунок використання роз'ємів, кабелів і модулів з невеликими розмірами. Тепер для сполучного устаткування недостатньо простих параметрів, таких як опір контактам і корозії. Устаткування, яке обробляє високошвидкісні дані, має бути захищене від електромагнітних завад, перехресних перешкод, імпедансу і інших проблем, які можуть вплинути на передачу сигналу. Багато що залежить від конекторів, здатних надійно передавати високошвидкісні сигнали з високою пропускною спроможністю і потужністю на кожен датчик в автомобілі і назад.

В результаті автовиробники звертаються до постачальників з проханням створити рішення націлені на короткострокову перспективу, які можуть бути впроваджені з існуючою інфраструктурою, але можуть бути перетворені в рішення, що відповідають вимогам завтрашнього дня. Постачальники докладають усіх зусиль, щоб забезпечити і розширити можливості високошвидкісної передачі даних, щоб задовольнити вимоги автовиробників, що прагнуть до більш високих рівнів підключення, включаючи автомобіль-автомобіль, автомобіль-інфраструктура і повну автономність водіння.

3 СТРУКТУРНА МОДЕЛЬ СИСТЕМИ УПРАВЛІННЯ

3.1 Принципи проектування програмного забезпечення

Програмно-апаратний комплекс автономного транспортного засобу є дуже складною системою, яка з розвитком технологій стає ще складнішою: саме тому система має бути розділена на окремі частини – модулі [5]. Крім того, коли розробка подібного продукту є настільки складною, що не тільки продукт, але й діяльність з розробки повинна бути структурована, щоб бути керованою. Як у галузі розробки програмного забезпечення, так і системної інженерії існує безліч таких процесів. Відповідний процес може бути обраний на основі потреб проекту з точки зору складності, критичності та терміновості продукту. Однак, через різну природу “віртуального” програмного забезпечення та фізично існуючого обладнання, ці процеси розвитку значно відрізняються.

Процес розробки програмного забезпечення може бути ітеративним та інкрементальним, що дозволяє забезпечити еволюційний розвиток до стабільного, надійного та зрілого результату. Сучасні ітеративні процеси розробки процеси, такі як XP, Scrum або RUP [8], підтримують ітерації на всіх етапах розробки з різною тривалістю ітерацій, яку встановлює менеджмент. Повторні дії необхідні для постійної оцінки поточного стану потреб замовника, з одного боку. Крім того, ітерації є важливою частиною процесу, за допомогою якої стаж можливим швидко реагувати на зміни та постійно інтегрувати актуальні розробки у функціональну систему.

В проектах з інтегрованим програмним забезпеченням та системах зі складним програмним забезпеченням потрібно використовувати ітеративний процес розробки програмного забезпечення замість традиційного інженерного процесу. Найкращий підхід – це якомога більше відокремити

підпроекти, щоб можна було стежити за окремими процесами в кожному з них та відповідно реагувати, а не робити проект в цілому і наприкінці виявиться, що в одній з підсистем щось пішло не так. Набір важливих етапів (точок відліку, англ. “milestone”) що є спільними для усіх підпроектів пов’язує процеси розробки кожної команди в одну систему, але в межах кожного підпроекту можуть використовуються різні форми процесів, згідно з потребами кожного окремого випадку. Крім того, абсолютно необхідно відокремити ітеративну розробку програмного забезпечення від апаратного забезпечення, наприклад за допомогою віртуалізації.

Розробляючи автономні керування автомобілів, автоматичне тестування має вирішити низку проблем. Перш за все програмне забезпечення інтегровано і тісно пов'язане з апаратним забезпеченням, таким як датчики та виконавчі механізми, а через них і з оточуючим середовищем.

Для ефективної роботи тестів необхідні відповідні абстракції для різних частин програмного та апаратного забезпечення. Для «інтелектуальної частини» програмного забезпечення не потрібно проводити тести, засновані лише на інформації з сенсорів, а надавати дистильовану, сукупну інформацію про можливі перешкоди, а також шлях проїзду через них. Дуже важливою абстракцією для ефективної автоматизації тестів є заміна реального обладнання на імітацію. Моделювання апаратного забезпечення дозволяє проводити автоматизовані тести на звичайних комп’ютерах і, отже, доступне для кожного розробника самостійно.

3.2 Структура комплексу

Сучасні автомобілі дедалі частіше використовують технології керування за дротом (англ. Drive-by-wire). Експлуатація автономного транспортного засобу вимагає спрацьовування трьох основних елементів управління:

– прискорення – спрацьовування дросельної заслонки;

– напрямку – спрацьовування керма;

– зупинки – спрацьовування гальма.

Приведення в дію дросельної заслінки досягається в більшості сучасних автомобілів за допомогою електронного управління. Електропідсилювач рульового керування витіснив гідравлічні системи підсилювання керма. Керування гальмівною системою досягається за допомогою електронного контролю стійкості.

Різноманітні датчики та комп'ютерні системи потрібні для того, щоб транспортний засіб міг відтворювати традиційний процес подібний до того, як людина керує транспортним засобом [2] – розуміння поточного місцезнаходження транспортного засобу, бажаного місця розташування та способу безпечного проходження маршрутів та небезпек на їх протязі. Серед наявних на даний момент датчиків жоден готовий модуль не здатний інтерпретувати своє середовище таким чином, щоб забезпечити достатньо інформації для повністю автономної їзди. Система повинна мати комбінацію датчиків. Структурна модель автономного транспортного засобу наведена на рисунку 3.1.

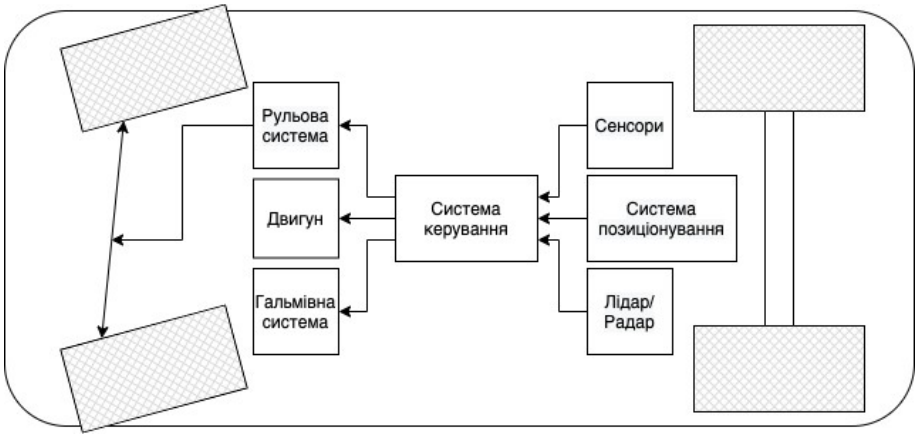

Рисунок 3.1 – Структура автономного автомобіля

3.3 Блок керування системи

В автономних транспортних засобах однією з найважливіших особливостей є правильне та точне виявлення перешкод, а також смуги руху транспортного засобу. Транспортний засіб або автомобіль повинні мати можливість вчасно і добре виявити наявність перешкоди, щоб він міг зупинитися на безпечній відстані, або уникнути зіткнення [17]. Виявлення смуги руху також є дуже важливим фактором, оскільки транспортний засіб повинен мати можливість утримуватись у межах смуги та слідувати лініям розмітки на дорозі, щоб правильно залишатися на колії та також слідувати за смугою руху.

Запропонована структурна модель системи автоматичного керування автомобільним транспортним засобом наведена на рисунку 3.2. Систему можна умовно розділити на п’ять основних блоків. Камера, або інший КМОП сенсор для зчитування даних відеопотоку у видимому діапазоні кольорів. Лідар – допомагає будувати трьохмірну карту оточуючого середовища, що використовується для побудови маршруту. Радар використовується для точного визначення дистанції до перешкод. Блок виділення ліній знаходить розмітку смуг руху та відповідно до них видає керуючі сигнали до електронного блоку керування. Блок запобігання зіткнень відповідно до сигнали з камери, лідару та радару приймає рішення про гальмування і так само як і блок виділення ліній через блок пріоритетів видає керуючі сигнали на органи управління автомобіля.

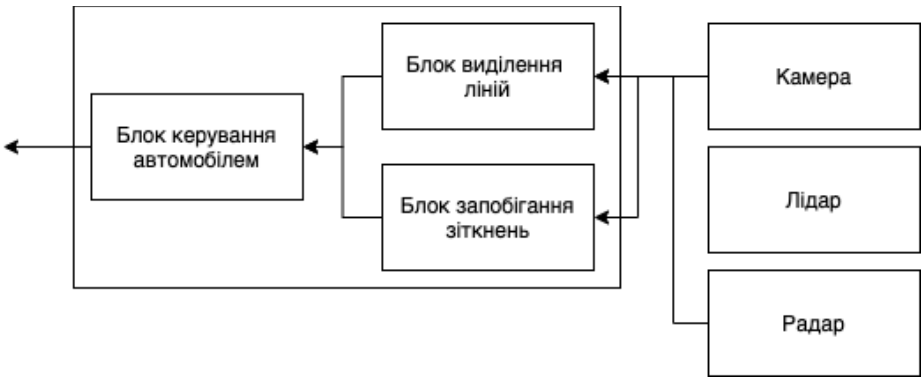

Рисунок 3.2 – Структурна модель системи автоматичного керування автомобільним транспортним засобом

3.4 Програмне забезпечення на алгоритми управління системи

Для розробленої моделі була запропонована програмна реалізація, яка визначає смуги руху, прораховує шлях [10] та реагує на можливі зіткнення. Програмна система працює за алгоритмом представленим на рисунку 3.3.

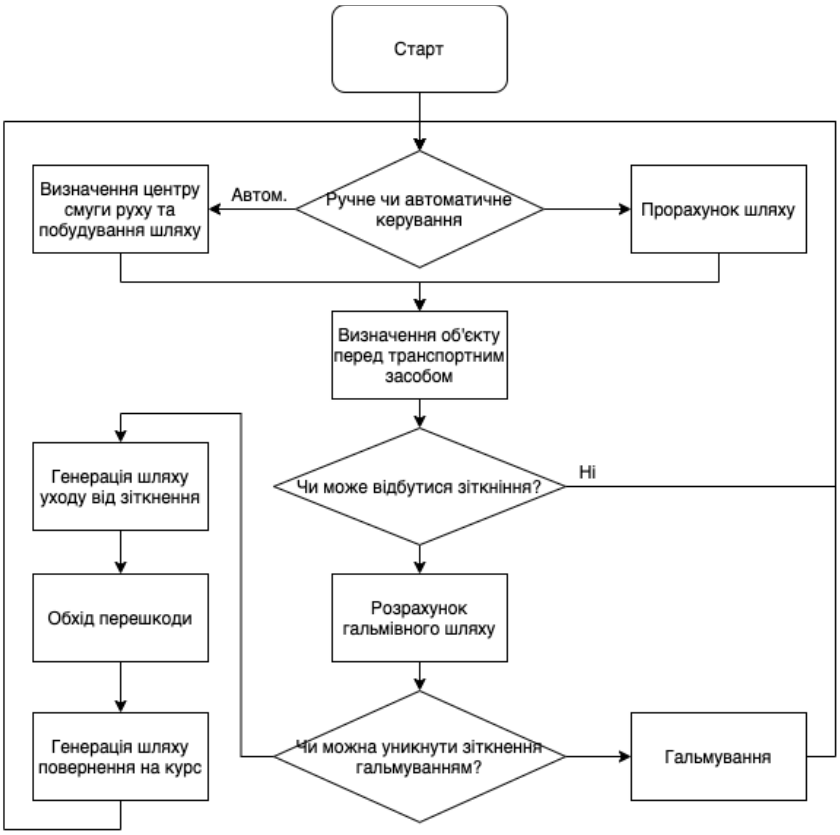

Рисунок 3.3 – Граф схема алгоритму роботи блоку керування

4 ІМПЛЕМЕНТАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

4.1 Структура програмно-апаратного комплексу

Для порівняння швидкодії програмної реалізації було розглянуто дві апаратні платформи – на основі мікроконтролера та ПЛІС.

Мікроконтролером для тестування було обрано модуль NVIDIA Jetson (рис. 4.1) через найкраще співвідношення ціни та швидкодії, а також має підтримку технологію NVIDIA CUDA[1]. Він був доповненим дистрибутивом ROS (Robotic Operation System), що була налаштована під необхідності автономного транспортного засобу та нейронною мережею навченою на відкритих даних з мережі.

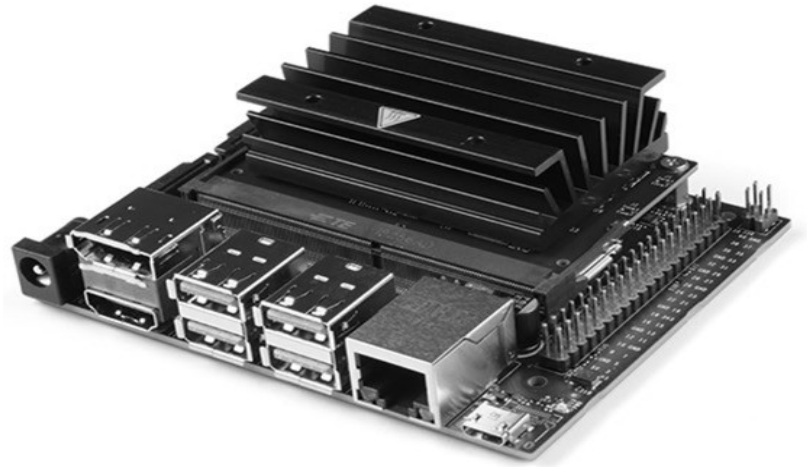

Рисунок 4.1 – NVIDIA Jetson Nano Developer Kit

Реалізація на ПЛІС була протестована на Arty S7-25, Spartan-7 FPGA, завдяки використання якого вдалось значно підвищити швидкість та точність розпізнавання смуги руху та своєчасно видавати реакцію на рульове керування автомобілем.

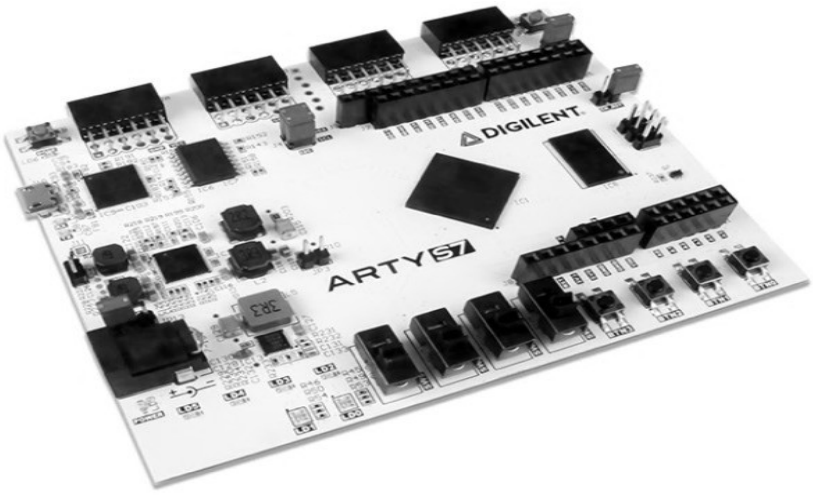

Рисунок 4.2 – Arty S7-25, Spartan-7 FPGA

4.2 Методи розпізнавання смуг руху

4.2.1 Розпізнавання країв

Будь-який розрив лінії розглядається як край. Тобто традиційні підходи до виявлення країв – це процес знаходження локальних максимумів у першій похідній або нульових перетинів у другій похідній шляхом перетворення зображення за допомогою якоїсь форми лінійного фільтра, який апроксимує першу або другу похідні. Хоча непарна симетрична функція може апроксимувати першу похідну, друга похідна апроксимується парною симетричною функцією.

В дискретному вигляді градієнт зображення можна просто розрахувати, взявши різницю значень сірого на зображенні. [3] Ця процедура дорівнює згортанню зображення однією з масок з таблиці 4.1. Очевидним недоліком цього спрощення є те, що незрозуміло, з яким пікселем пов'язаний результат. Існують різні підходи для вирішення цього питання, серед яких наступні маски фільтрів пропонують перше похідне рішення.

Таблиця 4.1 – Фільтри масок

	$\left(\frac{dI}{dx}\right)$	$\left(\frac{dI}{dy}\right)$
Robert	$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Prewitt	$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$
Sobel	$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$

Щоб вказати на карту градієнта кадру, в кожному випадку величина карти градієнта обчислюється за рівнянням 4.1

$$\sqrt{\left(\frac{dI}{dx}\right)^2 + \left(\frac{dI}{dy}\right)^2}\sqrt{\left(\frac{dI}{dx}\right)^2 + \left(\frac{dI}{dy}\right)^2} \quad (4.1)$$

Основним ідеєю при виявленні похідного краю є вплив шуму, оскільки локальні максимуми, зумовлені білим шумом, можуть маскувати реальні максимуми градієнта у краях. Ось чому необхідно згорнути зображення за допомогою функції згладжування, наприклад Функція Гауса, щоб ефект білого шуму був мінімізований. Обидва згадані оператори, наприклад, градієнт і Гаусова, є лінійними, і тому з точки зору обчислювальної складності ефективніше їх поєднати.

4.2.2 Виділення смуги руху

Виявлення смуг може бути здійснено різними способами, для цього був використаний підхід - Трансформація Хофа [15]. Перетворення Хофа – це техніка вилучення особливостей, яка використовується в аналізі зображень, комп'ютерному зорі та цифровій обробці зображень. Ціллю методики є пошук недосконалих екземплярів предметів у межах певного класу фігур за допомогою процедури голосування [16].

Найпростіший випадок перетворення Хофа – виявлення прямих ліній. Загалом, пряму лінію $y = mx + b$ можна представити як точку (b, m) у просторі параметрів. Однак вертикальні лінії створюють проблему. Вони могли б спричинити значення параметра нахилу m, що наближається до нескінченності. Таким чином, з обчислювальних причин використовується нормальна формула Гессе:

$$r = x\cos\theta + y\sin\theta \quad (4.2)$$

де r – відстань від початку координат до найближчої точки на прямій, а θ (тета) – кут між віссю x та лінією, що з'єднує початок координат із найближчою точкою. Отже, можна пов'язати з кожною лінією зображення пару (r, θ). Площину (r, θ) іноді називають простором Хофа для безлічі прямих у двох вимірах. Це подання робить перетворення Хофа концептуально дуже близьким до двовимірного перетворення Радона.

Беручи за базис одну точку на площині, тоді множина всіх прямих, що проходять через цю точку, відповідає синусоїдальній кривій у площині (r, θ), яка є унікальною для цієї точки [6]. Набір з двох або більше точок, які утворюють пряму лінію, дасть синусоїди, які перетинаються в (r, θ) для цієї лінії. Таким чином, проблему виявлення колінеарних точок можна перетворити на задачу пошуку паралельних кривих.

4.3 Реалізація модулю контролю смуги руху на мікроконтролері

4.3.1 Отримання початкових даних з відеопотоку

Nvidia Jetson Nano є дуже потужним модулем, з високопродуктивним графічним процесором, що дозволяє обробляти відеопотік майже з еквівалентною до стаціонарного комп’ютера швидкодією, саме тому серед доступних на ринку рішень було обрано саме його. Як початкові дані

використовується відеопотік, який може отримуватися з різних джерел, в залежності від типу компіляції. В демонстраційному режимі відеопотік отримуєтсья з файлу, що знаходиться на жорсткому диску комп'ютера, у режимі компіляції для реального пристрою відео отримується з підключеної камери. Таким чином отримання даних зводиться до відкриття відеопотоку, в даному випадку класом cv::VideoCapture, та послідовної обробки усіх кадрів.

4.3.2 Реалізація алгоритму для виділення смуги руху

Для людського ока смуга руху очевидна, проте цього не можна сказати про комп’ютер. Смуга руху [4] виділяється з кадру, який для машини не більше ніж матриця з кольорами. Обробка відеопотоку відбувається згідно до ГСА з розділу 2.6.

На початку відбувається виділення базових фігур і маскування непотрібних кольорів. Розмітка у більшості випадків рисується двома кольорами, білим і жовтим. Таким чином можна виділити ці кольори у бітову маску і накласти її на кадр. Параметри бітової маски для кольорів в RGB форматі наведені в таблиці 4.2.

Таблиця 4.2 – Параметри бітової маски для маскування кольорів

	Жовтий колір	Білий колір
Нижній поріг	255, 180, 0	100, 100, 200
Верхній поріг	255, 255, 170	255, 255, 255

Після створення бітових масок для кожного кольору робиться комбінування цих масок в одну за допомогою логічного складання (бітова операція «OR»). Готова маска накладається на початковий кадр [7] за допомогою логічного множення (бітова операція «AND»), причому ця операція робиться після перекладу зображення в чорно-білий спектр. Алгоритм, реалізований в функції edgeDetector, наведено у лістингу 4.1.

Лістинг 4.1 – Масковане зображення

```
cv::Mat LaneDetector::edgeDetector( cv::Mat img_noise ) {
        cv::Mat whiteMask, yellowMask, mask;
        cv::Mat output( img_noise.size(), CV_8UC1, cv::Scalar(0) );
        cv::Mat kernel;
        cv::Point anchor;
        cv::Scalar lowerb1( 100, 100, 200 );
        cv::Scalar upperb1( 255, 255, 255 );
        cv::inRange( img_noise, lowerb1, upperb1, whiteMask );
        cv::Scalar lowerb2( 255, 180, 0 );
        cv::Scalar upperb2( 255, 255, 170 );
        cv::inRange(img_noise, lowerb2, upperb2, yellowMask );
        cv::bitwise_or( whiteMask, yellowMask, mask );
        cv::cvtColor( img_noise, output, cv::COLOR_RGB2GRAY );
        cv::bitwise_and( output, mask, output );
        return mask;
}
```

Результатом виконання алгоритму є масковане зображення, що містить тільки білий і жовтий колір, необхідний для подальшого виділення розмітки. В результаті перетворення був отриманий наступний проміжний кадр (рис. 4.3).

Рисунок 4.3 – Масковане зображення

Кадр із замаскованими кольорами для людського ока виглядає як хаос білого та чорного кольорів, але для комп'ютера це оранізована структура, що допомагає подальшій оброці та розумінню того, де в кадрі знаходяться координати предмета інтересу, в даному випадку – розмітки. Проте ж кадр вийшов досить різким, отже, при подальшій обробці таких "різких" координат, точність обробки значно знизиться, що зробить використання такої системи небезпечним. Щоб уникнути такого ефекту було використано згладжування, в даному випадку, розмиття Гауса.

Розмиття Гаусса (також відоме як згладжування Гаусса) є результатом розмивання зображення за допомогою функції Гаусса (названої на честь математика і вченого Карла Фрідріха Гаусса). Цей алгоритм широко використовується в графічному програмному забезпеченні, як правило, для зменшення шуму зображення та зменшення деталізації. Візуальним ефектом цієї техніки розмивання є плавне розмиття, що нагадує перегляд зображення через напівпрозорий екран, чітко відрізняється від ефекту боке, який створюється поза фокусом або тінь об'єкта під звичайним освітленням. Гаусівське згладжування також використовується як етап попередньої обробки в алгоритмах комп'ютерного зору з метою підвищення структури зображень в різних масштабах.

Математично, застосування розмиття Гауса до зображення є таким же, як згортання зображення за функцією Гаусса. Це також відоме як двовимірне перетворення Вейєрштрасса, яке визначається за формулою:

(4.3)

Для досягняння ефекту розмиття було виористано стандартну функцію з бібліотеки OpenCV (лістинг 4.2).

Лістинг 4.2 – Накладання розмиття Гауса

```
cv::Mat LaneDetector::deNoise( cv::Mat inputImage ) {
        cv::Mat output;
        cv::GaussianBlur(inputImage,output,cv::Size(3,3),0,0);
        return output;
}
```

Після накладання розмиття отримується проміжний кадр (рис. 4.4) з набагато меньшою кількістю шумів та завад, що допомагає підвищити точнітсть визначення положення смуг руху та зменьшує ймовірність хибного визначення ліній.

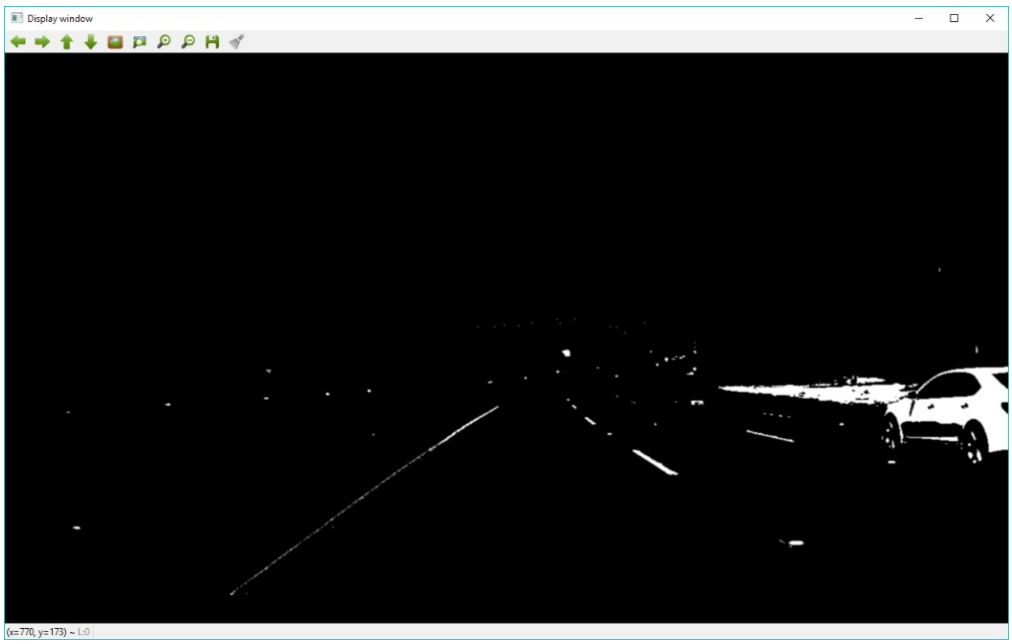

Рисунок 4.4 – Кадр з накладеним розмиттям Гауса

Після згладжування кадру відбувається виділення області інтересу. У зв'язку з особливостями сприйняття каринки (у наслідок наявності перспективи) камерою, розмітка, що необхідна для обробки програмою, не може знаходись вище за середину кадру і преставляє собою умовну "трапецію" в котрій і відбуватимуться усі обчислення. Для вирішення цієї

задачі був використана методика ліній Хофа, зазначена в розділі 4.2. Бібліотека комп'ютерного зору OpenCV має вкрай велику кількість готових функцій, які можна викростовувати. Так, для цього методу можна використати вбудовану функцію cv::houghLines, яка реалізує вище зазначений метод та приймає в себе стандартний формат кадру для бібліотеки. Кадр з виділенною областю інтересу наведено у лістингу 4.3.

Лістинг 4.3 – Виділення розмітки за допомогою ліній Хофа

```
std::vector<cv::Vec4i>LaneDetector::houghLines(cv::Mat img_mask)
{
	std::vector<cv::Vec4i> line;
	HoughLinesP(img_mask,line,1,CV_PI/180,20,20,300);
	return line;
}
```

Після виконання операції виділення ліній отримується проміжний кадр, що містить у собі лише розмітку, яка знаходяться у безпосередньо цікавому для обробки полі зору. Опрацьований кадр готовий для будь-яких подальших обчислень, як з визначення позиції відносно розмітки, так і визначення поточного вигину дорожнього полотна відносно осі руху. Результуючий кадр, готовий до подальшої обробки наведено на рисуноку 4.5.

Рисунок 4.5 – Маскований кадр зі смугами розмітки

4.3.3 Реалізація алгоритму визначення повороту

Алгоритм визначення повороту був реалізований на основі порівняння кута, за яким лінії розмітки знаходяться відносно нижній грані кадру. Таким чином алгоритм полягає у визначенні кута між шістьма відомими точками. Після визначення напряму руху, з маскованого кадру вибираються точні позиції ліній розмітки і виконується перевірка про можливий вихід за кордони розмітки. Алгоритм представлений у лістингу 4.4.

Лістинг 4.4 – Алгоритм визначення напряму руху

```
std::string LaneDetector::predictTurn() {
	std::string output;
	double vanish_x;
	double thr_vp = 10;

	vanish_x = static_cast<double>(
		(
			( right_m*right_b.x )
		-	( left_m*left_b.x )
		-	right_b.y + left_b.y )/( right_m - left_m )
	);
```

```
        if ( vanish_x < ( img_center - thr_vp ) )
                output = "Left Turn";
        else if ( vanish_x > ( img_center + thr_vp ) )
                output = "Right Turn";
        else if (
                        vanish_x >= ( img_center - thr_vp )
                &&      vanish_x <= ( img_center + thr_vp ) )
                output = "Straight";

        return output;
}
```

У разі компіляції в режимі демонстрації, поверх основного кадру по координатах ліній розмітки рисується полігон, що показує передбачувану траєкторію руху, напис з поточним поворотом та дистанцією до небезпечної перешкоди, якщо така існує. Вихідний кадр в режимі демонстрації зображений на рисунку 4.6.

Рисунок 4.6 – Тестове відображення смуг руху

4.3.4 Аналіз точності прогнозування повороту

Аналіз проводився наступним чином, у програмну систему почергово подавався набір відео потоків з різноманітними умовами тестування, поганим

освітленням (сутінки або ніч), складними погодними умовами, наприклад, під час дощу світло відбивається від поверхонь краще, що сильно ускладнює розпізнавання будь яких кольорів та об’єктів.

В ході виконання аналізу була сформована таблиця результатів (таблиця 4.3), де значення у відсотках – співвідношення вірно коректно визначеної смуги руху до явних помилок. Окремо варто зазначити вкрай сильні помилки вночі при освітленні від вуличних ліхтарів, причиною якого є нерівномірність освітлення, і як наслідок, неточність визначення об’єктів.

Таблиця 4.3 – Результати аналізу

	Погодні умови			
Освітлення	Ясно	Дощ	Сніг	Туман
Ранок	90%	60%	20%	40%
День	100%	70%	40%	50%
Вечір	90%	60%	20%	40%
Ніч	50%	40%	0%	20%

Варто зазначити, що співвідношення різних погодних умов різне і сильно відрізняється від кліматичних умов, у яких використовується автомобіль та пори року, саме тому найбільш раціональним є обчислення середньої точності за усіма можливими факторами, що було вирахувано за формулою 4.4.

$$\sum \frac{\begin{pmatrix} Mc & Mr & Ms & Mf \\ Dc & Dr & Ds & Df \\ Ec & Er & Es & Ef \\ Nc & Nr & Ns & Nf \end{pmatrix}}{n} \sum \frac{\begin{pmatrix} Mc & Mr & Ms & Mf \\ Dc & Dr & Ds & Df \\ Ec & Er & Es & Ef \\ Nc & Nr & Ns & Nf \end{pmatrix}}{n} \tag{4.4}$$

де M-ранок, D-день, E-вечір, N-ніч, c-ясно, r-дощ, s-сніг, f-туман, n-загальна кількість всіх знайдених вірогідностей.

Результатом розрахунку стала ймовірність помилки у 30%. Таким чином, завдяки даним з таблиці та фінальній ймовірності помилок можна зробити висновок, що точність визначення смуги руху та кута повороту залежить від якості відео потоку і від рівня освітлення в кадрі, але це також відносяться до якості відео. Можна майже повністю позбавитись цих помилок використанням інфрачервоної камери, або тепловізору.

4.4 Програмна реалізація на ПЛІС

4.4.1 Розпізнавання ліній

Алгоритм лінійного перетворення Хофа використовує двовимірний масив, який називається акумулятором, для виявлення існування лінії, описаної рівнянням нижче:

$$r = x\cos\theta + y\sin\theta r = x\cos\theta + y\sin\theta \quad (4.5)$$

Розмір акумулятора дорівнює кількості невідомих параметрів, тобто двох, враховуючи квантовані значення r, а θ - пара (r, θ). Для кожного пікселя в точці (x, y) та його сусідства алгоритм перетворення Хофа визначає, чи є достатньо доказів виявлення прямої лінії на цьому пікселі. Якщо так, він обчислює параметри (r, θ) цього рядка, а потім шукає буфери, до якого потрапляють параметри, і збільшує значення цього відділу аккумулятору. Знаходячи буфери з найвищими значеннями, як правило, шукаючи локальні максимуми в просторі акумулятора, можна витягти найімовірніші рядки та зчитати їх (приблизні) геометричні визначення. Найпростіший спосіб знайти ці піки – це застосувати одну з наявних форм порогового значення, але інші методи можуть дати кращі результати за різних обставин – визначаючи, які лінії знайдені, а також скільки. Оскільки повернуті рядки не містять інформації про довжину, часто на наступному кроці необхідно знайти, які частини зображення збігаються з якими рядками. Більше того, через помилки

та недосконалості на кроці виявлення краю, як правило, виникають помилки в буфері акумулятора, що може зробити нетривіальним пошук відповідних піків, а отже, і відповідних ліній [17].

Кінцевим результатом лінійного перетворення Хофа є двовимірний масив (матриця) – одним виміром цієї матриці є квантований кут θ, а іншим виміром є квантована відстань r. Кожен елемент матриці має значення, рівне сумі точок або пікселів, які розташовані на лінії, представленій квантованими параметрами (r, θ). Отже, елемент з найбільшим значенням позначає пряму лінію, яка найбільше представлена на вхідному зображенні.

4.4.2 Апаратна архітектура системи виявлення ліній

Модель апаратної архітектури представлена на рис. 4.7. Система розділена на чотири блоки: блок введення камери, блок вилучення краї, блок перетворення Хофа та блок ідентифікації ліній. Послідовність введення та виведення кожного модуля та внутрішні операції описані в наступних підрозділах.

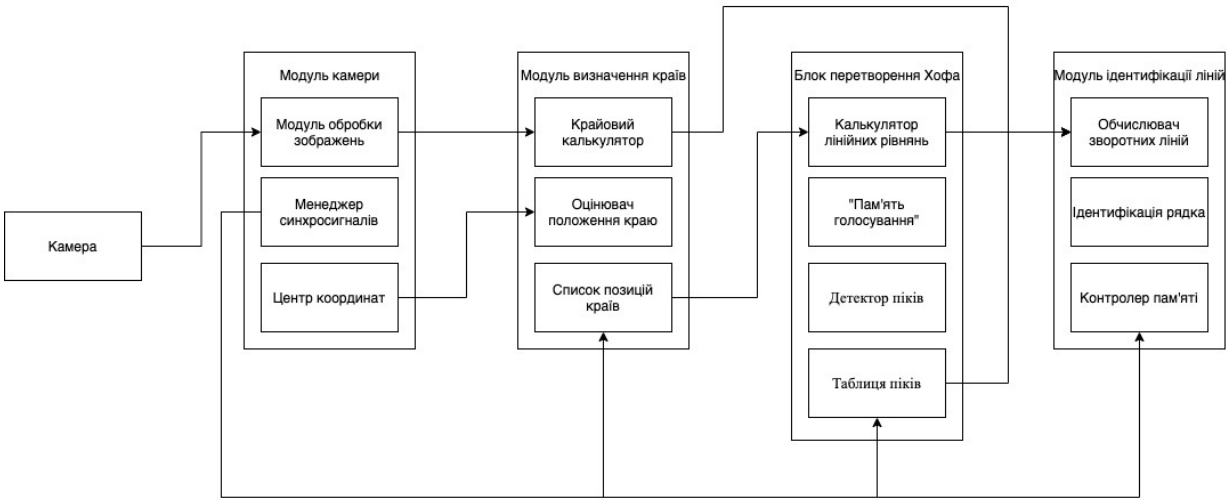

Рисунок 4.7 – Модель архітектури системи виявлення ліній

4.4.3 Блок визначення країв

4.4.3.1 Крайовий калькулятор

Калькулятор країв витягує зображення краю із вихідного зображення [8] за допомогою обробки вікон у режимі реального часу. Піксель необробленого зображення та його положення передаються з блоку введення камери. Калькулятор країв може зберігати локальну область необробленого зображення за допомогою віконного буфера, але при цьому він не використовує буфер кадру. За допомогою алгоритму Собеля калькулятор країв визначає, чи є центральний піксель буфера вікна крайовою точкою. Якщо цей піксель є краєм, тоді калькулятор ребер виводить «1» і навпаки - якщо цей піксель не є крайовою точкою, калькулятор країв видає «0», тому калькулятор країв перетворює вихідне зображення 1920 * 1080 * 8 біт у 1920 * 1080 * 1 біт.

4.4.3.2 Оцінювач положення краю

Оцінювач положення краю робить список положень краю із зображення краю. Зображення ребра не підходить для обчислення за допомогою лінійного рівняння, оскільки для рівняння лінії потрібні координати x та y, але зображення ребра – це лише двійкова інформація. Оцінювач положення краю перевіряє результат калькулятора країв. Якщо за допомогою калькулятора країв цей піксель визначається як ребро, тоді оцінювач положення краю зберігає положення цього пікселя у списку положень краю. В кінці зображення ребра оцінювач положення краю записує кінцеві дані біта ‘1’ у список положень краю, щоб вказати кінець списку. У кінцевих даних, якщо всі біти дорівнюють «1», це неприпустима позиція, тому система може розрізнити кінцеві дані та позиції готові для обробки.

4.4.3.3 Список позицій країв

Список позицій країв зберігає інформацію про положення кожної точки краю на одному зображенні краю. Список позицій краю підтримує два подібних списки для запобігання конфліктам під час операцій читання та запису. Неможливо виконувати операції читання і запису одночасно. Один список призначений для читання; інший – для запису. За допомогою цього методу подвійної буферизації запобігається затримки на синхронізації та підвищується швидкость обробки кадру. Більше того, блок перетворення Хафі може працювати незалежно від синхросигналу камери, тому перетворення Хофа може оброблятися з більш високою швидкістю, використовуючи більш високочастотний синхронізуючий імпульс.

4.4.4 Блок перетворення Хофа

Блок перетворення Хофа [9] виконаний у вигляді чотирьох модулів. Одним із них є «калькулятор лінійних рівнянь», який обчислює значення rho за допомогою лінійного рівняння. Другий – це «пам'ять для голосування», яка створює розподілений простір параметрів за допомогою значення rho. Третій – це «детектор піків», який знаходить пікове положення в розділеному просторі параметрів. Четвертий модуль – це «таблиця піків» для зберігання значень тета та rho виявленого піку. Останній – це «контролер перетворення Хофа», який керує цими модулями. Таблиця піків передає інформацію в блок ідентифікації лінії, щоб визначити точне положення лінії.

На рис. 4.8 наведена модель блоку перетворення Хофа. Блок перетворення Хофа містить 16-лінійний калькулятор рівнянь і пари «пам'яті для голосування», тому він може працювати в 16 разів швидше. Калькулятор лінійних рівнянь та пам’ять для голосування працюють паралельно, щоб одночасно обчислити 16 значень тета, тому цю операцію потрібно повторити 21 раз, щоб обчислити 336 значень тета.

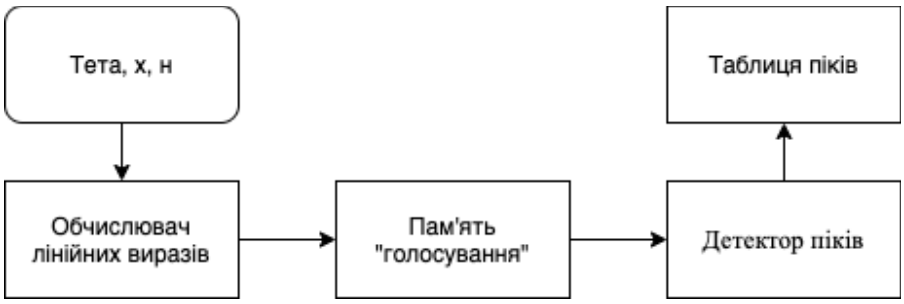
Тета, х, н
Обчислювач лінійних виразів
Пам'ять "голосування"
Детектор піків
Таблиця піків

Рисунок 4.8 – Модель модуля Хофа

4.4.4.1. Калькулятор лінійних рівнянь

Калькулятор лінійних рівнянь працює відповідно до рівняння 4.6. У ньому є таблиця синусів, таблиця косинусів, два множники та один суматор. Таблиця синусів і косинусів побудована з використанням пам'яті лише для читання (ПЗУ). Ці підмодулі налаштовані як конвеєрна структура. Після визначеної затримки алгоритму кінцевий результат, що складається із значення rho, отримується на кожному тактовому циклі. Загальна затримка конвеєра, яка бере участь у перетворенні інформації (x, y, тетта) у значення rho, становить 7 тактових циклів.

$$p = x * Cos(\varphi) + y * Sin(\varphi)\, p = x * Cos(\varphi) + y * Sin(\varphi) \quad (4.6)$$

4.4.4.2 Пам'ять голосування

Пам'ять для голосування налаштована в одному блоці пам'яті ПЛІС. Пам'ять для голосування має два режими роботи. Один режим – збільшити, а другий – очистити. Ця пам'ять для голосування накопичує введені значення rho в режимі збільшення. По-перше, він зчитує значення комірки пам'яті в голосуючій пам'яті, яке позначається значенням rho, і значення цієї комірки пам'яті збільшується та оновлюється. У режимі очищення вміст пам'яті для голосування передається на детектор піків, а потім дані очищаються. Адреса пам'яті для голосування послідовно збільшується в режимі очищення.

4.4.4.3 Детектор піків

Детектор піків виявляє точку піку в пам'яті для голосування, яка розділена на простір параметрів. Розмір розділеного простору параметрів становить 16 * 800. На Рисунку 4.9 показана конфігурація детектора піків. Детектор піків знаходить локальні максимуми в просторі параметрів 16 * 16,

а потім порівнює це максимальне значення з заздалегідь визначеним пороговим значенням.

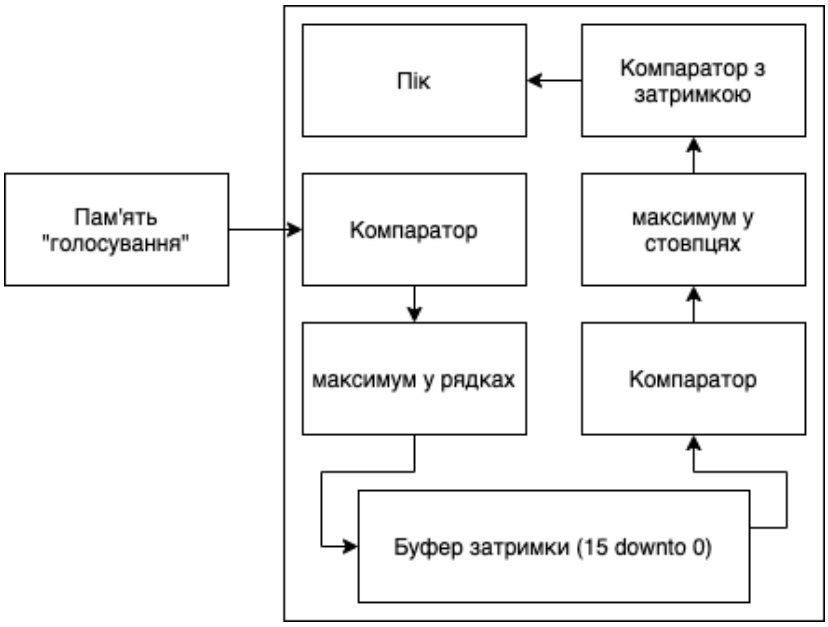

Рисунок 4.9 – Структурна модель детектору піків.

4.4.4 4. Таблиця піків

Таблиця піків зберігає параметри лінії виявленої точки піку в розділеному просторі параметрів. Вміст таблиці піків передається в блок ідентифікації лінії для ідентифікації точного положення лінії.

4.4.5 Модуль ідентифікації лінії

Компонент ідентифікації лінії визначає точне положення лінії та її пікові параметри лінії на зображенні краю. Блок ідентифікації лінії складається з трьох модулів. Перший – це калькулятор рівнянь зворотної лінії, щоб знайти положення домену зображення за допомогою параметрів лінії. Другий – це ідентифікатор рядка, який перевіряє існування лінії на крайовому зображенні зі значенням положення обчислювача рівняння зворотної лінії. Останній – це «контролер пам’яті», який зберігає зображення краю і передає пікселі краю в ідентифікатор рядка.

4.4.5.1. Обчислювач зворотних виразів

Використовуючи наступні рівняння, можна дізнатися положення лінії, використовуючи параметри лінії, які є rho та theta. Індекс рівняння 5 змінюється, і це функція тета. Якщо діапазон тета – діапазон між 0,79 і 2,35, координата x використовується як індекс, а рівняння 5 обчислює відповідну позицію y. У цьому випадку діапазон значення індексу x становить від 1 до 1920.

$$y = \frac{\rho - \mathbf{i}\cos\theta}{\sin\theta} y = \frac{\rho - \mathbf{i}\cos\theta}{\sin\theta} \quad (4.7)$$

$$y = \frac{\rho - \mathbf{i}\sin\theta}{\cos\theta} y = \frac{\rho - \mathbf{i}\sin\theta}{\cos\theta} \quad (4.8)$$

Як варіант, координата y може бути використана як індекс для рівняння 6 зі значенням y в діапазоні від 1 до 1080. Рівняння зворотної лінії формує положення (x, y) параметрів лінії. Дані про цю позицію передаються до ідентифікатора лінії та контролера пам'яті, щоб знайти точну позицію лінії. Параметр рядка надходить із таблиці піків блоку перетворення Хофа. Якщо індекс досягає максимального значення (або 1080, або 1920), калькулятор рівняння зворотної лінії зчитує наступний параметр рядка в таблиці піків. Калькулятор рівняння зворотної лінії повторює ці операції, поки в таблиці піків не залишиться більше параметрів.

4.4.5.2 Ідентифікатор рядка

Ідентифікатор рядка ідентифікує рядок, використовуючи параметри рядка та зображення краю. Ідентифікатор рядка отримує дані про положення параметрів лінії від калькулятора рівняння зворотної лінії, і він зчитує дані ребра цієї позиції від контролера пам'яті. Крайове зображення в контролері пам'яті – це зображення 1920 * 1080 * 8 біт. Якщо піксель є ребром, тоді значення пікселя дорівнює "11111111", інакше значення пікселя дорівнює "00000000".

4.4.5.3 Контролер пам'яті

Контролер пам'яті зберігає крайове зображення і передає піксель краю в це крайове зображення із запитом ідентифікатора. Контролер пам'яті управляє двома буферами пам’яті FIFO та двома BRAM. FIFO – це буфер кадру, який відокремлює робочу швидкість блоку ідентифікації лінії від тактової частоти камери. BRAM зберігає крайове зображення, а також одночасно зчитує та записує, тому контролер пам'яті має два BRAM і перемикається між ними під час виконання операцій читання та запису. На рисунку 4.6 показана конфігурація контролера пам'яті.

Ця техніка подвійного буфера використовується для зберігання простору НТ. Використовується два екземпляри пам'яті НТ; перший оновлюється з використанням поточних даних кадру, тоді як другий зберігає дані НТ попереднього кадру, який потім використовується для подальшої обробки. З кожним новим кадром буфери перемикаються за допомогою мультиплексорів.

4.5 Порівняння швидкодії імплементацій

Було проведено тестування на різних наборах даних з спеціально введеними випадковими похибками. Очевидно, що ПЛІС через високу паралельність значно переважає у швидкості визначення смуг руху, результати порівняння імплементацій представлені у таблиці 4.4.

Таблиця 4.4 – Порівняння швидкодії імплементацій

Рівень похибки	Тип випадку	Імплементація	Час (ms)	Прискорення
Випадкова 0%	Гірший	Програмна	80	71
		Апаратна	1.1252	
	Кращий	Програмна	39.1	74
		Апаратна	0.522	
Випадкова 10%	Гірший	Програмна	75.2	70
		Апаратна	1.061	
	Кращий	Програмна	30.8	65
		Апаратна	0.471	
Випадкова 20%	Гірший	Програмна	65.9	59
		Апаратна	1.084	
	Кращий	Програмна	22.1	45
		Апаратна	0.491	
Випадкова 30%	Гірший	Програмна	57.8	50
		Апаратна	1.158	
	Кращий	Програмна	20.2	47
		Апаратна	0.431	
Випадкова 40%	Гірший	Програмна	51.8	45
		Апаратна	1.145	
	Кращий	Програмна	19.8	48
		Апаратна	0.411	
Випадкова 50%	Гірший	Програмна	40.4	36
		Апаратна	1.091	
	Кращий	Програмна	13.9	41
		Апаратна	0.338	

Відповідно до результатів тестування, очевидно, що у результуючій системі для виділення смуг руху раціональніше використовувати апаратні прискорювачі на ПЛІС, а для визначення зіткнень простіше і логічніше використовувати класичний мікропроцесор з комбінацією нейронної мережі що оброблює відеопотік та радару.

ВИСНОВКИ

В результаті виконання атестаційної роботи був розроблений програмно-аппаратний комплекс, який виконує аналіз відеопотоку, визначає дистанцію до перешкод та здійснює прогнозування кута повороту, що зебезпечує безпечне автоматичне керування транспортного засобу на основі вхідних даних.

В роботі були проаналізовані технології аналізу відеопотоку на основі бібліотек комп'ютерного зору та алгоритмів з відкритих джерел. Були розроблені моделі блоків виделення смуг руху, що виконують функції для фільтрації зображення, виділення з нього розмітки прогнозування куту повроту.

Було порівняно два типи апаратних засобів для реалізації моделі системи автоматичного керування транспортного засобу у смузі руху.

Запропонована система була протестована на двох апаратних платформах: одноплатному комп’ютері на архітектурі ARM та ПЛІС. Була перевірена робота системи з тестовими пристроями. Також була перевірена працездатність системи обробки відео потоку та автоматичного керування транспортним засобом у лабораторних умовах. Проведено порівняння швидкодії обробки відеопотоку на тестових платформах.

Згідно з результатами порівняння швидкодії апаратних платформ було зроблено висновок, що у фінальній системі раціональніше використовувати ПЛІС для визначення смуг руху, а мікроконтролер для обходу перешкод.

Наведена система автоматичного руху може у реальному часі обробляти відео потік, визначати позицію автомобіля відносно меж смуги руху, корегувати траєкторію в залежності від вигину дорожнього полотна та перешкод. Також, використання ультразвукового датчику допомагає тримати дистанцію до транспортних засобів попереду та виконувати маневр по запобіганню зіткнення з перешкодами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Nvidia CUDA Toolkit Documentation – URL: https://docs.nvidia.com/cuda/ – Загл. с экрана.

2. Habr.com – URL: https://habr.com/company/toshibarus/blog/431388/ – Мир глазами автомобиля. Каким его видят беспилотники?

3. A Medium Corportaion – URL: https://medium.com/@cacheop/advanced-lane-detection-for-autonomous-cars-bff5390a360f – Advanced Lane Detection for Autonomous Cars

4. A lane Detection, Tracking and Recognition System for Smart Vehicles – Guangqian L. Школа Електроніки та Комп'ютерних наук / L. Guangqian. – Оттава, Канада: Факультет інженерії, 2015. – 120 с. – (Університет Оттава).

Конспект лекцій з дисципліни «Інженерія програмного забезпечення» для студентів усіх форм навчання спеціальності 6.050102 «Комп’ютерна інженерія» [Електронне видання] / Упорядн.: Зайченко С.О. – Харків: ХНУРЕ, 2016. – 235 с.

Л. Шапиро, Дж. Стокман. Компьютерное зрение = Computer Vision. – М. : Бином. Лаборатория знаний, 2006. – 752 с

Девід Форсайт, Жан Понс. Комп’ютерний зір. Сучасні засоби = Computer Vision: A Modern Approach. – М. : «Вильямс», 2004. – 928 с.

Benjamin F. D. A Year in Computer Vision. The M Tank. [Електронний ресурс] / D. Benjamin F., F. Daniel R.. – 2017. – Режим доступу до ресурсу: http://www.themtank.org/a-year-in-computer-vision.

Azriel R. Digital Picture Processing / R. Azriel, K. Avinash. – Сан Франциско: Morgan Kaufmann Publishers Inc., 1982. – 435 с. – (Перше).

Chen, C. Tang, L. Xin, S. E. Li, and M. Tomizuka. // IEEE Intell. Veh. Symp.. – 2018. – №29. – С. 1651–1658.

Levels of Driving Automation [Електронний ресурс] // SAE International.. – 2018. – Режим доступу до ресурсу: https://www.sae.org/news/press-

room/2018/12/sae-international-releases-updated-visual-chart-for-its-“levels-of-driving-automation”-standard-for-self-driving-vehicles.

Automated Vehicles for Safety [Електронний ресурс] // National Highway Traffic Safety Administration. – 2020. – Режим доступу до ресурсу: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

Software-defined autonomy [Електронний ресурс] // NVIDIA. – 2020. – Режим доступу до ресурсу: https://www.nvidia.com/en-us/self-driving-cars/.

Autonomus driving [Електронний ресурс] // Qualcomm. – 2020. – Режим доступу до ресурсу: https://www.qualcomm.com/products/automotive/autonomous-driving.

HoughTransformWiki. [Електронний ресурс] // Qualcomm. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Hough_ transform.

Yang B. The design of 3 * 3 real time Videoimage convolver on the basis of FPGA”. In: Proceedings of 2011 International Conference on Electronic and Mechanical Engineering and Information Tech / Baohai Yang. // EMEIT. – 2011. – C. 4381–4384.

Ronith Raj Ram Prakash. FPGA Based Lane Tracking system for Autonomous Vehicles / Ronith Raj Ram Prakash. – Стокольм: Kth Royal Institute of Technology, 2019. – 53 с.