Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)
Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Модель керування автопілотом для наземного транспорту

(тема)

Виконав: студент 2 курсу, групи СКСм-20-1
Колковський В.І.
(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 роботи доц. Філіппенко І.В.
(посада, прізвище, ініціали)

Допускається до захисту

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

2021 р.

Харківський національний університет радіоелектроніки

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
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2021 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Колковському Віталію Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Модель керування автопілотом для наземного транспорту

затверджена наказом по університету від "04" 11 2021 р. № 1635 Ст.
2. Термін подання студентом роботи (проекту) 20.12.2021
3. Вихідні дані до роботи (проекту)
Нечітка логіка
Керування наземним транспортом
ATmega328
Arduino UNO

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити)
Процеси керування наземним транспортом що потребують автоматизації
Проектування систем на базі нечіткої логіки
Створення тестової платформи для імітації роботи транспортних засобів
Імплементація систем на базі нечіткої логіки на Arduino

Нормалізація даних для розпізнавання кольору
Перевірка функціонування розробленого пристрою

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 16 слайдів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	04.11.2021 - 05.11.2021	
2	Аналіз предметної області	06.11.2021 - 07.11.2021	
3	Аналіз джерел з проблемної галузі	08.11.2021 - 11.11.2021	
4	Розробка підходу для реалізації поставленої	12.11.2021 - 16.11.2021	
5	Реалізація поставленої задачі	17.11.2021 - 26.11.2021	
6	Оформлення пояснювальної записки	29.11.2021 - 05.12.2021	
7	Оформлення графічного матеріалу	06.12.2021 - 10.12.2021	
8	Перевірка виконаного проекту керівником	13.12.2021 - 15.12.2021	
9	Захист проекту	16.12.2021 – 20.12.2021	
10			

Студент _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 90 сторінок, 45 рисунків, 12 джерел за переліком посилань.

АВТОПІЛОТ, НАЗЕМНІ ТРАНСПОРТНІ ЗАСОБИ, НЕЧІТКА ЛОГІКА, ARDUINO, ATMEGA328, EFLL, BLUETOOTH КЕРУВАННЯ

Метою кваліфікаційної роботи є реалізація моделі керування автопілотом для наземного транспорту із використанням систем прийняття рішень на базі нечіткої логіки.

Для цього в роботі було розглянуто процеси користування транспортними засобами, що потребують автоматизації. Розглянуті переваги використання у даному проекті нечіткої логіки перед іншими математичними моделями. Виконане проектування систем прийняття рішень із використанням середи Fuzzy logic toolbox у складі програмного пакету MATLAB. Був реалізований прототип на базі мікроконтролеру ATmega328. Створені системи прийняття рішень реалізовані на пристрої із використанням бібліотеки eFLL. Також розглянутий процес нормалізації отриманих с датчиків даних для їх обробки системою на базі нечіткої логіки.

Робота кінцевого приладу була перевірена у ряді тестів із підтвердженням працездатності.

ABSTRACT

The explanatory note of the qualification work: 90 pages, 45 figures, 12 sources.

AUTOPILOT, GROUND VEHICLES, FUZZY LOGIC, ARDUINO, ATMEGA328, EFLL, BLUETOOTH DRIVING

The purpose of the qualification work is to implement an autopilot control model for ground transport using decision-making systems based on fuzzy logic.

To do this, the paper considers the processes of using vehicles that require automation. The advantages of using fuzzy logic in this project over other mathematical models are considered. Design of decision-making systems using the Fuzzy logic toolbox environment as part of the MATLAB software package was performed. A prototype based on the ATmega328 microcontroller was implemented. The created decision-making systems were implemented on the device using the eFLL library. The process of normalization of the data received from the sensors for their processing by the fuzzy logic system is also considered.

The operation of the end device was tested in a number of tests with confirmation of operability.

ЗМІСТ

ВСТУП

Технологічний світ не стоїть на місці і з кожним роком стрімко розвивається, впроваджуючи нові засоби як для вирішення наявних світових проблем, так і для спрощення, полегшення чи навіть автоматизації повсякденних різноманітних задач.

Навіть наземні транспортні засоби первинно були створені як альтернатива кінним візкам, а вже сьогодні вони заполонили світ і мільярдами використовуються для задач переміщення людей або речей, призвели до появи цілої автодорожньої інфраструктури та змінили відношення людей до мандрівок на великі дистанції.

При чому розроблені пристрої не залишилися на рівні перших прототипів, що являли собою прості відкриті візки з невеликими двигунами. За весь свій період існування приблизно у 150 років, винахід пройшов великий шлях та радикально змінився у порівнянні із своїми першими варіаціями. На сьогодні автомобілі це найчастіше комфортні та прості у використанні засоби, що забезпечують можливість вільно пересуватися на величезні дистанції.

Усі зміни, що втілюються в нових варіаціях автомобілів спрямовані в першу чергу на підвищення комфорту водія та пасажиру, підвищення безпечності руху та екологічності. Так, поява та широке розповсюдження автоматичної коробки перемикання передач значно спростили взаємодію з автівками, а поява ременів та подушок безпеки допомогла врятувати велику кількість життів у аваріях.

А за рахунок активного розвитку комп’ютерної техніки та появи нових методів та моделей обробки інформації в автомобільному світі також почалась активна комп’ютеризація. Сучасні автомобілі мають більше сенсорів, датчиків та обчислюваної потужності ніж перші космічні ракети. І

всі ці засоби використовуються для впровадження нових рішень та підходів, що повністю змінюють поняття водіння автомобілем.

За такою тенденцією вже через декілька десятків років від людини необхідно буде лише обрати кінцеву точку поїздки, а всі інші дії виконуватиме сам автомобіль не потребуючи втручання з боку користувача, якого в такому разі вже важко називати водієм.

Але на сьогоднішній день розроблені технології з автоматизації певних аспектів у керуванні автомобілем доступні лише невеликій кількості кінцевих користувачів, через високу ціну моделей із вбудованими рішеннями та відсутністю альтернатив.

В рамках даної роботи розглядається розробка моделі керування автопілотом для наземного транспорту із використанням нечіткої логіки для обробки даних та прийняття рішень з метою спрощення взаємодії водія з автомобілем.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Розвиток технологій, крім внесення значного прогресу у вирішенні трудомістких або раніше нездійсненних завдань, також великою мірою впливає на рівень і якість життя людей. Більшість останніх досягнень у технологічному світі знаходять застосування у побутових та повсякденних речах людей.

Так, сучасний смартфон повністю замінив цілий перелік пристроїв, вмістивши в себе їх функціонал – засіб зв'язку, записник, фотоапарат, радіо, засіб для доступу в інтернет, що розкриває ще більший функціонал, та інше.

Подібні покращення та рішення для спрощення простих побутових або робочих аспектів все частіше впроваджуються у нашому житті. Причому такі рішення можуть бути як глобальними, що мають ефект на широкій кількості людей, так і досить простими і локальними, але навіть без них життя здається вже зовсім іншим та недостатньо зручним.

Однією зі сфер, що найбільш активно націлені у бік спрощення взаємодії користувача с приладом, є транспортна. Якщо поставити поруч перші прототипи транспортних засобів та їх сучасні аналоги, то різниця буде колосальною.

Відмінним прикладом темпів автоматизації транспорту є літаки, перший із яких, створений братами Райт, піднявся в небо 17 грудня 1903 року. А вже через 9 років, 1912 року компанія Sperry Corporation представила світові перший аналог сучасного автопілота, який дозволяв у автоматичному режимі утримувати курс польоту та стабілізував крен літака [1].

Рисунок 1.1 – Панель керування однією з перших систем автопілотування

Подібними кроками та темпами розвиваються і автомобільний, водний, залізничний транспорт, будівельна та інша спеціалізована техніка.

Що стосується автомобілів, що являють собою максимально широке охоплення користувачів серед простих людей, то за дослідженням Іена Севеджа з університету Нортуестерн [2], легковий транспорт є найбільш летальним за співвідношенням смертності до 1 мільярда пройдених миль.

Як можна помітити з графіка, інші транспортні засоби являються набагато безпечнішими. Це насамперед пов'язано з тим, що керують ними спеціально навчені люди, які чітко розуміють як діяти в екстремальних ситуаціях, і суворо дотримуються встановлених правил користування транспортом.

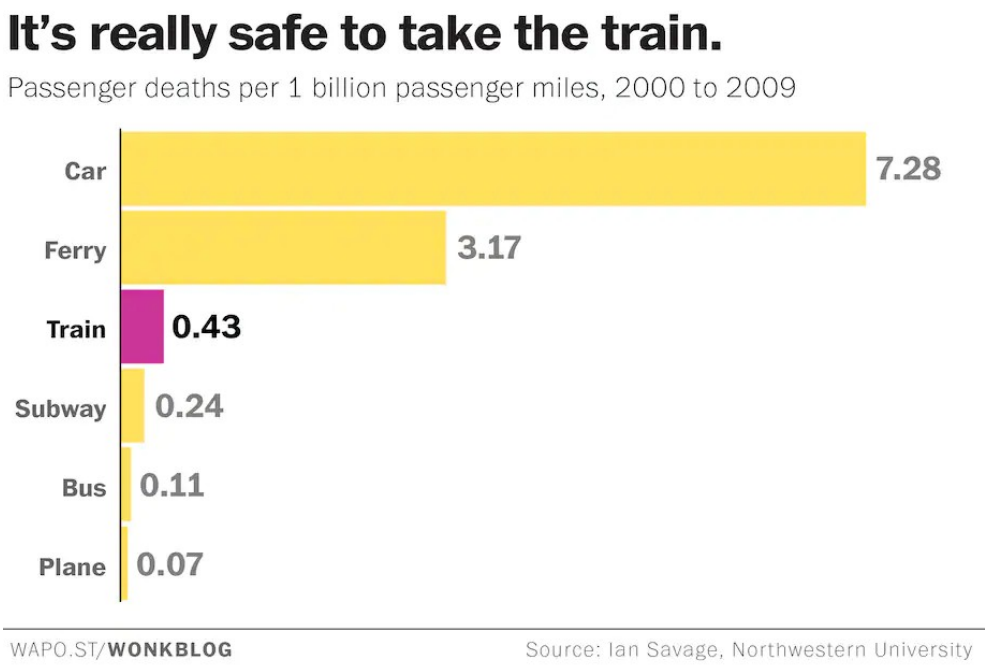


Рисунок 1.2 – Кількість летальних випадків на 1 мільярд пройдених миль

Крім цього, ті ж літаки, поїзди та водний транспорт оснащені великою кількістю аварійних систем для мінімізації збитків у разі виникнення поломок, аварій та інших небезпечних ситуацій.

Автомобілі ж керуються простими людьми. Згідно з дослідженням Міжнародної асоціації автовиробників (OICA), на 2015 рік в експлуатації знаходилися 947 млн легкових та 335 млн комерційних автомобілів [3].

З кожним роком кількість автомобілів, як і водіїв, тільки зростає. Держави вводять різні системи контролю та сертифікації автомобілів за ступенем їхньої безпеки. А автовиробники повинні відповідати їм під час випуску нової моделі ринку.

Так, зараз усі автомобілі оснащуються ременями та подушками безпеки, нові моделі проходять велику кількість краш-тестів для перевірки безпеки людей як усередині, так і зовні автомобіля у разі виникнення аварій різноманітних видів.

Сучасні варіанти передбачають системи аварійного гальмування, зміни

налаштувань підвіски для зменшення впливу бічного зіткнення, інформування про перешкоди у сліпих зонах. [4]. Сама конструкція та матеріали автомобілів підібрані з розрахунком на мінімізацію збитків життю у разі аварії.

Додатковою перевагою наявності таких систем в автомобілях є простота реалізації на їх основі функціоналу, спрямованого не лише на безпеку, а й комфорт управління транспортним засобом. Велика кількість датчиків дозволила реалізувати системи утримання в смузі, адаптивний круїз-контроль, системи автоматичного паркування, камери сліпих зон, камери нічного бачення, адаптивні фари з підсвічуванням важливих ділянок, системи розпізнавання перешкод, дорожніх знаків та інше.

Всі вони поступово з'являються в комплектаціях автомобілів різних брендів і з роками стають доступними дедалі більшій кількості користувачів. Таким чином, глобальна статистика щодо ситуацій на дорогах з кожним роком поступово покращується.

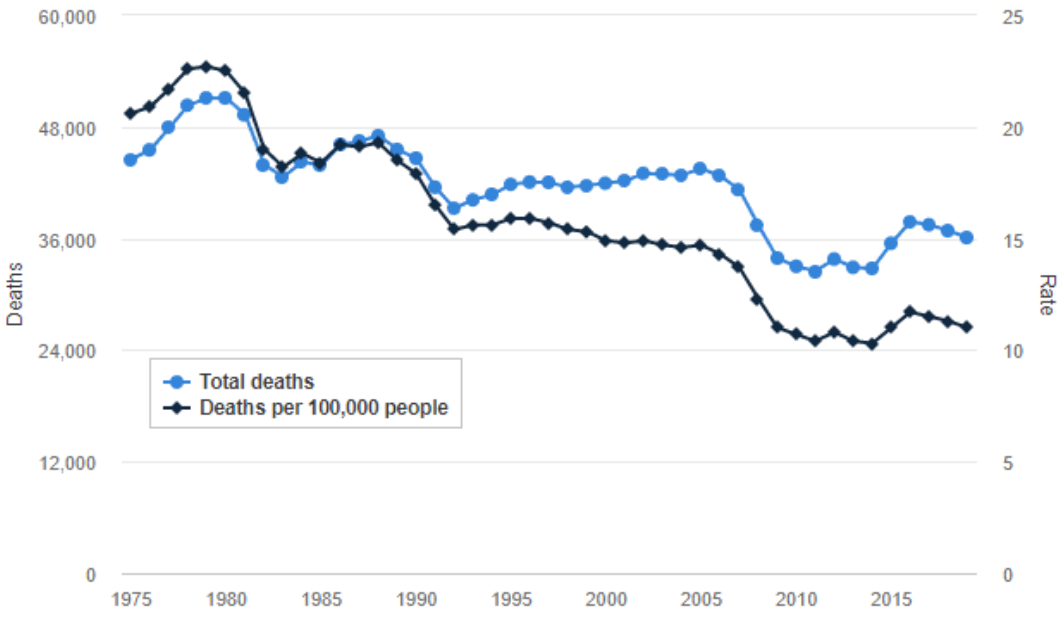


Рисунок 1.3 – Загальна кількість летальних випадків на 100000 людей з 1975 по 2019 рік, Міністерство транспорту США

Так, за інформацією міністерства транспорту США [5], у 1975 році в аваріях загинуло 44525 осіб, а коефіцієнт смертності на 100000 осіб склав 20,6. А у 2019 році, за загальної кількості летальних випадків 36096, коефіцієнт смертності становив лише 11.

Виходячи з такої статистики, легко зрозуміти, що навіть при значному збільшенні кількості транспортних засобів на дорогах з кожним роком і відповідно великою кількістю аварій, ймовірність учасників руху виявитися неушкодженими лише зростає за рахунок постійного підвищення стандартів безпеки та впровадження нових рішень для її підтримки.

Ідеальним рішенням для досягнення максимальної безпеки на дорогах і, до того ж, спрощення взаємодії людей з транспортом, буде повна його автоматизація, що відносить такий транспорт до 5 рівня автономності, або «steering wheel optional» згідно з класифікацією автоматизації автомобілів Спільноти автомобільних інженерів (SAE) [6].

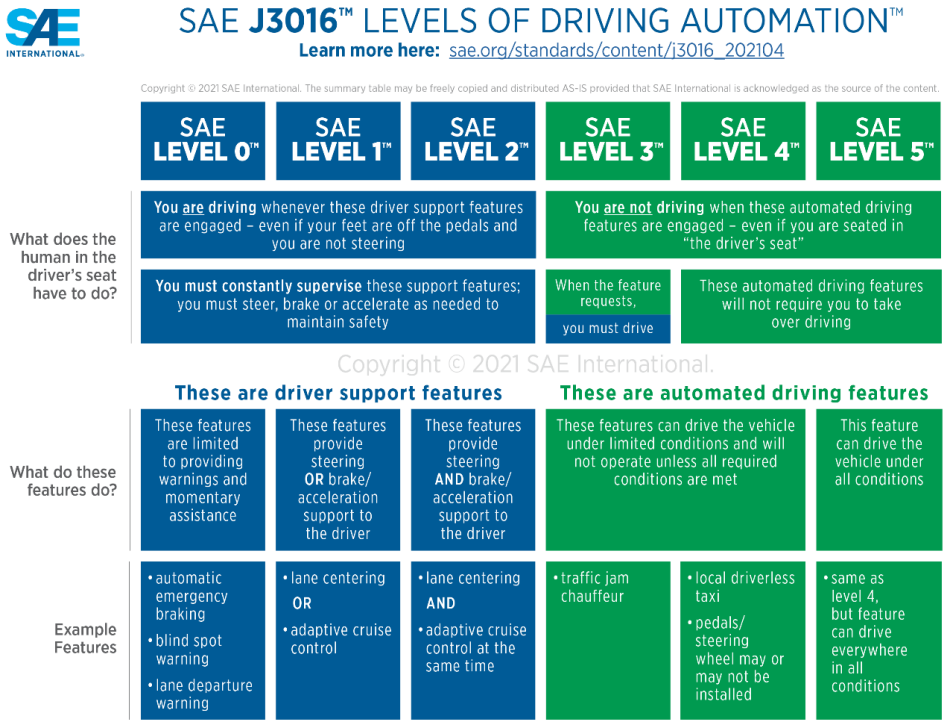


Рисунок 1.4 – Класифікація Спільноти автомобільних інженерів (SAE) рівнів автоматизації керування транспортними засобами

Але на сьогоднішній день такий рівень автоматизації являється або складно реалізованим або практично неможливим за наявної інфраструктури. Для повноцінного автопілотування, що не вимагає від автомобіля наявності суперкомп'ютера для аналізу дорожньої обстановки, потрібно враховувати багато умов і навіть невеликих умовностей. Такі системи складно та майже неможливо навчити діям при всіх можливих ситуаціях на дорозі, оскільки проблеми можуть викликати навіть найнесподіваніші і абсолютно унікальні ситуації.

Наприклад, вже існуючий автопілот компанії Tesla, що відноситься до другого рівня автоматизації, часто потрапляє в ситуації, які неможливо передбачити на стадії проектування. Так, система розпізнавання дорожніх знаків і світлофорів помилково спрацьовувала на місяць, що знаходиться на небі, оскільки той відбивав жовте сонячне світло, сприймаючи його як жовтий знак світлофора [7]. А вантажівка дорожньої служби, повна світлофорів також вводила систему в оману тим, що в її кузові знаходилось три непрацюючі світлофори. Їх вантажівка просто переміщувала до місця встановлення. Причому попередження про світлофор викликалось постійно та про кожен окремий світлофор у кузові автомобіля попереду [8].

Таким чином демонструється можливість сучасних систем якісно та точно визначати наявність важливих об’єктів, але точно проаналізувати їх значимість для прийняття рішень система самостійно не може.

Виходячи з наявних даних та пропозицій представників ринку, вчених та інженерів, виходом із такої ситуації буде налагодження повної комунікації між усіма автомобілями на дорозі для обміну даними та синхронізації їх дій. Такі системи також дозволять вирішити проблеми пробок, заторів та аварій на дорогах, оскільки система, за відсутності помилок у роботі, не припустить, або максимально знизить ймовірність таких подій.

Але на даному етапі побудова таких систем є лише далеким планом на майбутнє, тому що навіть серйозних прототипів, стандартів комунікації транспортних засобів або інших продуктів цього напрямку практично немає.

У зв'язку з цим, на сьогодні, найбільш реальним та затребуваним буде здешевлення та популяризація систем спрощення взаємодії та підвищення безпеки на індивідуальних транспортних засобах. Як говорилося раніше, до таких систем найчастіше відносяться: контроль швидкості автомобіля, аналіз дорожніх знаків та світлофорів, контроль сліпих зон, процес паркування, детектування зіткнень, системи утримання у смузі.

1.1 Контроль швидкості автомобіля

Рішення для автоматизації управління швидкістю з'явилися майже одночасно з появою самих автомобілів. Перші системи для підтримки заданої швидкості були встановлені в 1901 році на автомобілі марки Wilson-Pilcher і являли собою важіль біля рульової колонки, який дозволяв обмежити максимальну швидкість руху.

Варіант, який зараз прийнято називати «круїз-контролем», був вперше представлений компанією Chrysler у 1958 році на моделі Imperial. Він являв собою пристрій, який зчитував швидкість руху автомобіля зі спідометру і змінював положення дросельної заслінки для підтримки заданої водієм швидкості.

Сучасні системи є більш розвинутими, в першу чергу за рахунок комп'ютеризації автомобілів. Наприклад, електронні системи дозволили точніше регулювати задану швидкість за різних дорожніх умов і типу покриття. А установка в передній частині автомобілів різних датчиків вимірювання відстані дала можливість реалізувати адаптивний круїз-контроль, який в автоматичному режимі змінює свої налаштування, щоб безпечно слідувати за транспортним засобом попереду, пригальмовуючи і збільшуючи швидкість до заданої без участі водія. Та при зупинці автомобілю попереду система автоматично зупинить автомобіль на безпечній відстані.

1.2 Детектування зіткнень

Ті ж датчики відстані дозволили вирішити проблему виникнення різких перешкод на дорозі або компенсувати помилку водія, якщо той не зумів зреагувати на перешкоду, що наближається. Залежно від просунутості, такі системи або підказують звуковим, візуальним та тактильним способами про появу загрози безпечному руху, або перехоплюють контроль на себе та здійснюють екстрене гальмування чи маневрування для уникнення зіткнення.

Крім того, датчики зіткнень допомагають зменшити можливі пошкодження навіть від бічних аварій та ситуацій, коли автомобіль не рухається. Так, при виникненні загрози система може автоматично керувати конфігурацією підвіски, ременями безпеки та іншими доступними засобами для більш жорсткої фіксації людей в салоні автомобіля та мінімізації сили зіткнення.

1.3 Аналіз дорожніх знаків та світлофорів

Подібні системи лише починають впроваджуватися в сучасних автомобілях, оскільки складаються з камери, що зчитує зображення попереду автомобіля, та алгоритму, найчастіше нейронної мережі, для виявлення на отриманих кадрах знаків дорожнього руху або світлофорів. Крім того, системи можуть додатково отримувати інформацію з наявної бази даних дорожніх знаків, визначаючи їхнє положення по GPS. Такі системи являються невід'ємною частиною систем автопілотування, але навіть без них допомагають водієві краще реагувати на дорожні знаки у разі утрудненої видимості, або, наприклад, нагадують допустиму максимальну швидкість на ділянці дороги.

Рисунок 1.5 – Приклад системи визначення дорожніх знаків та розмітки в автомобілі Tesla

Крім того, на транспорті може встановлюватися додатковий модуль для «нічного бачення», який виводить водію змінене зображення попереду автомобіля, на якому чітко помітні об'єкти та відстань до них навіть за максимально обмеженої видимості, наприклад, уночі в снігопад.

Рисунок 1.6 – Відображення інформації з камери «нічного бачення» на панелі приладів автомобіля

1.4 Системи утримання у смузі руху та контроль сліпих зон

Такі системи також реалізовані за рахунок розміщення у передній частині автомобіля пристрою відеофіксації. З отриманих кадрів система отримує інформацію про наявну дорожню розмітку та положення інших транспортних засобів. Грунтуючись на цій інформації, автомобіль або автоматично підрулює, щоб залишатися в поточній смузі руху, або нагадує водієві про те, що він наблизився до краю або перетнув розмітку.

Для контролю сліпих зон використовуються датчики, найчастіше розташовані у бічних дзеркалах автомобіля, направлені назад. Ця система відмінно себе показує при перебудовах на трасі, оскільки допомагає додатково повідомляти водія про наявність перешкод ліворуч і праворуч позаду автомобіля навіть у зонах, що важко побачити у бокових дзеркалах.

1.5 Системи допомоги у паркуванні

Згідно зі звітами Національної ради з безпеки (NSC) США, на рік відбувається понад 50000 аварій на паркувальних майданчиках та в гаражах, понад 500 з яких закінчуються летально. Така висока небезпека пов'язана насамперед із неуважністю всіх учасників руху на паркувальних майданчиках.

За проведеним опитуванням, 66% людей готові розмовляти телефоном під час керування автомобілем на паркувальних майданчиках, а ще близько 50% – набирати текстові повідомлення [9]. Окрім зниженої уваги учасників руху, немалу роль відіграє обмежена видимість, стисліть простору для маневрів автомобілів та рівень навичок водіння у водія. Саме з паркуванням найчастіше виникають проблеми у більшості людей у перші роки керування автомобілем.

Для вирішення проблем з процесом паркування та її спрощення розроблено чимало різних рішень, що відрізняються рівнем автоматизації та

технологічною просунутістю.

Найбільш простими системами для допомоги в паркуванні є системи парктронік, що являють собою набір ультразвукових датчиків, встановлених у передньому та задньому бампері транспортного засобу. Вперше така технологія з'явилася в 1982 році на автомобілі Toyota Corona. Інформація з датчиків виводиться водієві та відображає поточну відстань до перешкоди у кожного з датчиків. За рахунок цього водій може рухатися практично в щільну до бордюрів, стінок або інших транспортних засобів, не боячись зачепити їх через обмежену видимість.

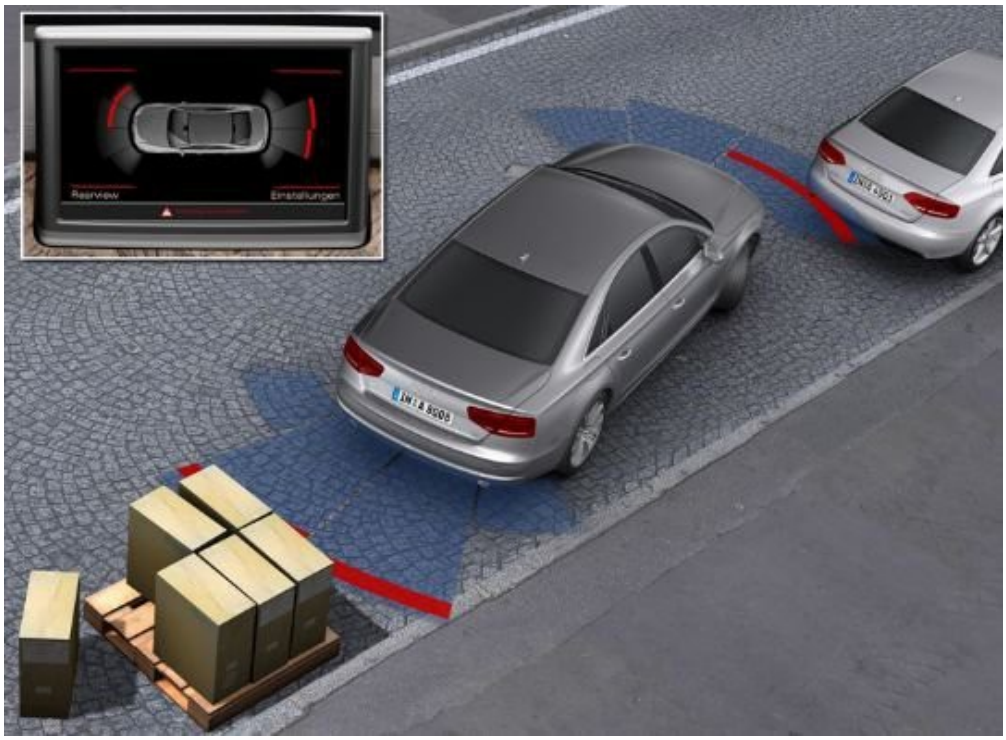


Рисунок 1.7 – Паркувальний радар від компанії Audi

Ще однією розробкою підвищення видимості можливих перешкод навколо автомобіля стали камери сліпих зон. Для цього на автомобілі встановлюються камери по периметру: попереду, позаду та навіть у дзеркалах у дорогих комплектаціях. Зображення з них виводиться на екран водію та дозволяє оцінити дорожню ситуацію по периметру всього

автомобіля у невеликому радіусі.

Крім того, на екрані найчастіше виводиться додаткова інформація, як, наприклад, вектор напрямку руху при поточному положенні керма, відстань до перешкод та інше. Сучасні системи, за рахунок установки більш просунутих камер, інфрачервоних датчиків та застосування різних алгоритмів обробки зображення дозволяють отримувати чітку та ясну картину того, що відбувається навіть з повністю неосвітлених та темних ділянок.

Рисунок 1.8 – Камери кругового огляду на сучасних автомобілях

Щодо досягнень останніх років, то за рахунок серйозної комп'ютеризації всіх установок автомобілів з'явилася можливість передавати повне керування маневрами внутрішнім системам. За рахунок наявності великої кількості різних датчиків, камер та сенсорів, надійних і швидких систем обробки інформації, у сучасних моделях почали з'являтися різні засоби автоматичного паркування.

Здебільшого вони схожі і відрізняються лише функціоналом та ступенем автоматизації. Наприклад, у деяких водій повинен контролювати

рух постійно натиснутою педаллю газу та тримати руки на кермі. Інші ж варіанти дозволяють водієві зовсім перебувати за межами автомобіля і лише утримувати кнопку активації функції на брелоку або в додатку на смартфоні.

Що стосується функціоналу, то існуючі системи можуть або запам'ятовувати здійснені водієм дії раніше і відтворювати їх у зворотному або тому самому порядку, наприклад, для автоматичного виїзду з місця паркування, або при паркуванні на знайоме місце біля будинку або в гаражі. Деякі системи взагалі не вимагають від водія установки транспорту в місце початку паркування, а здатні самі пересуватися по паркувальному майданчику в пошуках вільного місця для паркування.

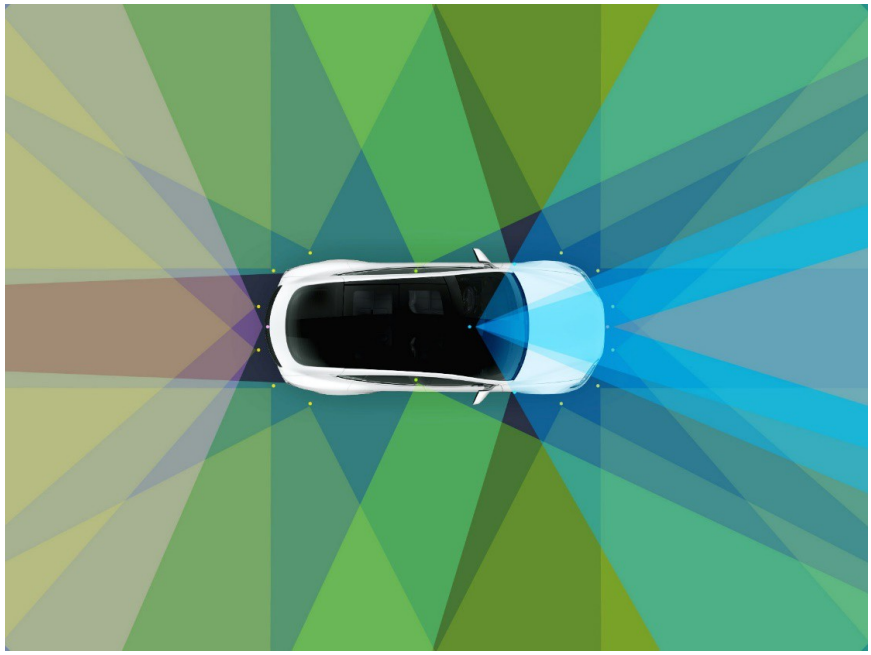


Рисунок 1.9 – Зона покриття вбудованих у автомобіль Tesla Model X датчиків

Така тенденція розвитку все-таки наближає автомобілі до повного автопілотування на дорогах, але системи подібного рівня вимагають дуже високої надійності всіх її компонентів, швидкості роботи та можливості правильно зреагувати у будь-якій, навіть не передбаченій на етапі проектування ситуації.

2 ОПИС ОБРАНОЇ СИСТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

2.1 Постанова задачі

Перед реалізацією проекту необхідно визначитися з вимогами до його влаштування та функціоналу, адже надалі для їх зміни потрібно додати чимало ресурсів.

Таким чином, система, що розробляється, повинна задовольняти наступним вимогам:

- відповідати фізично достовірним моделям транспортних засобів з актуальним влаштуванням механізму кермового керування для подальшої можливості простоти установки на більшість сучасних моделей;
- складатися з простих та дешевих компонентів, щоб бути доступним для більшої кількості кінцевих користувачів та уникнути можливих проблем із виходом з ладу наявних компонентів;
- при цьому всі компоненти повинні бути надійними і безвідмовними для забезпечення безпеки роботи системи.
- швидкість роботи системи має бути досить високою для коректної реакції у разі виникнення можливих раптових ситуацій;
- адаптивність базових пристроїв системи до впровадження нових компонентів або заміни старих, наприклад, можливість просто замінити датчики вимірювання відстані на аналоги з підвищеною надійністю або функціоналом;
- відповідно до вищесказаного, система повинна мати можливість легко масштабуватись для застосування на повнорозмірних транспортних засобах різного виду після успішного тестування прототипів.

Ключовим аспектом проекту, виходячи з перелічених умов, насамперед

буде можливість подальшого впровадження системи у вже існуючі транспортні засоби у вигляді додаткового обладнання. Також важливим фактором буде прагнення максимально здешевити кінцевий виріб за умови збереження бажаного функціоналу, а також надійності та безпеки роботи.

Щоб досягти поставлених вимог, необхідно ретельно поставитися до планування технічного пристрою проекту та розглянути такі його аспекти:

- проаналізувати наявні системи керування напрямом руху транспортних засобів та загального влаштування шасі і обрати відповідний варіанту для розробки та тестування системи на платформі, що підходить під фізичні моделі схожі, або ідентичні застосовним у реальному світі;

- провести вибір апаратної платформи, яка виконуватиме керування всіма компонентами системи та повинна відповідати поставленим функціональним вимогам;

- підібрати окремі компоненти системи для реалізації бажаного функціоналу;

- розглянути процес об'єднання отриманої системи з тестовою платформою, отримати інформацію про можливі проблеми інтеграції системи до існуючих транспортних засобів надалі;

- розробити програмну частину, яка відповідає заданому функціоналу, що має можливість підвищення доступних можливостей, наявності адаптивної системи управління, здатної в автоматичному режимі підлаштовуватися під поточні умови, дорожню ситуацію;

- реалізувати систему управління для дистанційного контролю та моніторингу за тестовою платформою.

Крім того, так як проект являє собою систему для керування транспортними засобами, фізичні властивості яких дозволяють завдавати істотної шкоди життю та майну через високу вагу та можливість розвивати велику швидкість руху, підсумковий виріб повинен бути ретельно протестований як у різних дорожніх ситуаціях, так і у разі можливої відмови компонентів для забезпечення максимальної безпеки під час функціонування

пристрою.

2.2 Вибір апаратної платформи

У якості головного пристрою, що отримує дані з датчиків, обробляє їх і формує подальші команди для управління рухом транспортного засобу вирішено було використовувати платформу Arduino.

Ця платформа поставляється у вигляді невеликих плат (Arduino UNO R3, Arduino PRO MINI, Arduino NANO та ін.), що мають широкий набір інтерфейсів для підключення зовнішніх пристроїв та програмного середовища Arduino IDE для написання, налагодження та компіляції коду, а також його запису на плату.

Серед доступних варіантів була обрана модель UNO R3, яка містить 14 цифрових входів та виходів, 6 з яких підтримують ШІМ та 6 аналогових входів. За рахунок великої кількості доступних портів з'являється можливість підключити велику кількість різних датчиків для збору інформації та пристроїв для керування транспортним засобом.

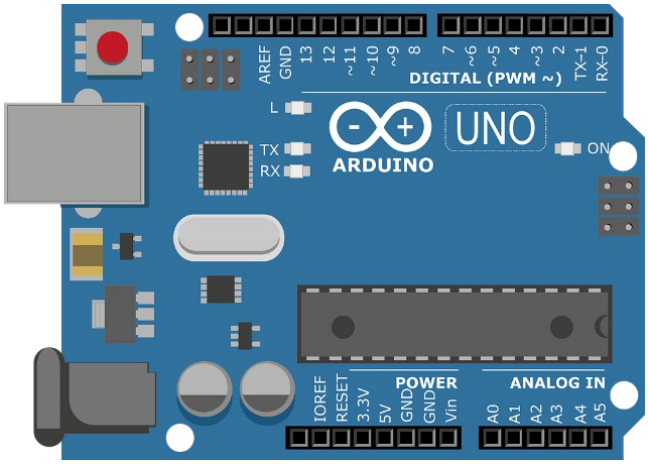


Рисунок 2.1 – Схематичне зображення плати Arduino UNO R3

Визначальною особливістю даної платформи на відміну від конкурентів є широка поширеність, дешевизна і простота. За рахунок цього

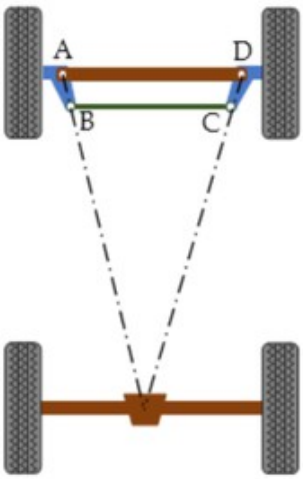
плата підтримує роботу з більшістю модулів і датчиків, що присутні на ринку. Крім того, для підключення та взаємодії із зовнішніми пристроями реалізовано велику кількість протестованих та оптимізованих бібліотек. Це дозволяє адаптувати систему для роботи з різними наборами датчиків, не вимагаючи детально розумітися у особливостях їх внутрішнього складу та функціонування.

На даному етапі обчислювальних потужностей платформи достатньо для реалізації поставленого функціоналу з достатньою точністю та швидкістю обчислень і за рахунок цього значно знижується ціна кінцевого виробу.

2.3 Вибір фізичної моделі

Для правильного розуміння поведінки системи під час тестування та подальшоъ імплементації на повнорозмірних транспортних засобах, фізична модель тестової платформи повинна максимально відповідати реальним аналогам у всіх своїх характеристиках окрім розміру.

Розмір зменшується з єдиною метою – спрощення тестування продукту на етапах проектування. При цьому масштаб все одно співвідноситься з реальними легковими транспортними засобами і становить 1/12 реального розміру аналогічного автомобіля.



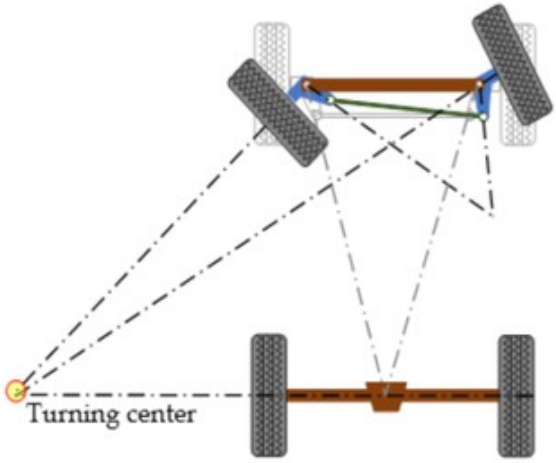


Рисунок 2.2 – Пристрій кермового керування транспортного засобу

Найбільш важливою для коректного моделювання особливістю є влаштування системи повороту транспортного засобу. Залежно від цього варіюватиметься мінімальний радіус повороту, від якого безпосередньо залежить загальна мобільність транспорту та всі розрахунки його руху при автоматизації.

Найпоширенішими варіантами зміни напрями руху колісного транспорту є рульове та «бортове» керування.

При рульовій системі керування напрямок руху транспорту визначається поточним положенням поворотних, найчастіше передніх коліс. Зміна їх кута виконується поворотом керма водієм і зазвичай повертає колеса на 30 градусів у крайніх положеннях керма. Центр повороту при такому компонуванні знаходиться за межами транспортного засобу, на місці перетину ліній проведених перпендикулярно центрам передніх та задніх коліс у їхньому поточному положенні. Такий варіант пристрою має різні варіанти виконання, що впливають як на надійність, так і ефективність, чутливість керування.

На додаток до поворотних передніх колес у деяких сучасних моделях почали встановлюватися механізми зміни кута повороту задніх коліс. Системи підрулювання задніх коліс, або 4 Wheel Steering (4WS) застосовуються як для підвищення стабільності на великих швидкостях, так і для підвищення маневреності на низьких. Так, при високій швидкості руху колеса повертаються в той же бік, що і передні, а при низькій – у протилежну.

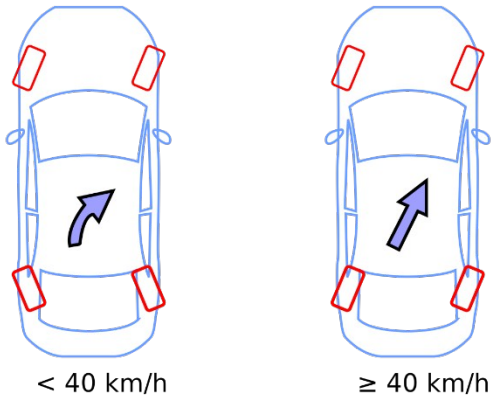


Рисунок 2.3 – Принцип роботи системи підрулювання задніх коліс

Що ж до «бортового» керування, воно найчастіше застосовується на гусеничній техніці, чи техніці спеціального призначення, наприклад, військовій. При такому пристрої контроль напрямком руху здійснюється шляхом зміни швидкості обертання всіх ведучих механізмів з лівого або правого борту транспортного засобу. Залежно від зміни їх швидкості центр повороту варіюється і, при одночасному русі бортів в різні сторони дозволяє транспорту повернути практично на місці.

Оскільки найбільш поширеним варіантом для зміни напрямку руху на сучасних автомобілях є застосування кермового керування, саме його варто використовувати в тестовій платформі. У такому разі при подальшому тестуванні та імплементації на реальних зразках не виникнуть проблеми з невідповідністю фізичної моделі.

Щоб спростити завдання та не займатися проектуванням тестової платформи з нуля, скористаємося вже розробленими рішеннями, що підходять під задані параметри. Серед таких максимально доступним та простим варіантом буде застосування шасі від машинки на радіокеруванні.

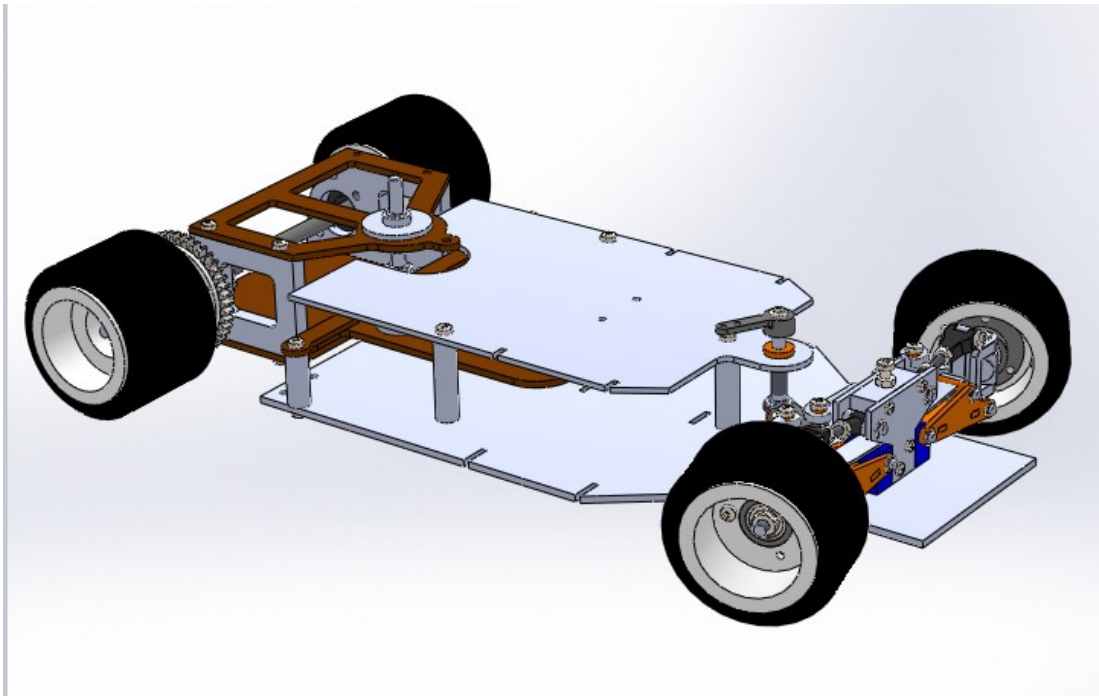


Рисунок 2.4 – Схематичне зображення каркаса проекту

У таких моделей параметри шасі найчастіше співвідносяться з деякими реальними моделями автомобілів, є кермо з необхідними параметрами кута повороту коліс і принципом роботи. Також вже наявний двигун, найчастіше електричний, розташований у задній частині та обертаючий задню вісь.

Таким чином, єдине, що значною мірою відрізняє шасі радіокерованої моделі від реальних зразків, крім розміру, це розподіл маси автомобіля між передньою та задньою осями. Але в разі потреби він може бути легко відкоригованим встановленням додаткових вантажів на каркасі.

2.3.1 Система керування платформою

На вибраній платформі для побудови системи були встановлені компоненти:

- сервопривід для повороту передніх коліс;

- плата управління для зв'язку з пультом дистанційного керування та виконання поданих із нього команд;

- електродвигун на задній осі для приведення системи у рух;
- акумуляторна батарея для живлення;
- Корпус.

Існуюча на платформі радіокерованої машини плата управління не мала відповідних для розроблюваної системи інтерфейсів, а також не дозволяла чітко управляти швидкістю руху і кутом повороту коліс, лише приймаючи фіксовані значення.

Також, сервопривід, що відповідав за поворот коліс, не дозволяв виставляти конкретний кут, а приймав одне з трьох положень: крайнє ліве, крайнє праве і посередині. Швидкість роботи сервоприводу також не підходила для реалізації проекту.

Щоб зайві компоненти не заважали при проектуванні системи, вони були демонтовані та замінені на більш просунуті. Зовнішній корпус довелося прибрати зовсім, тому що він виконував лише декоративну функцію та сильно обмежував доступне для встановлення компонентів місце.

Наявна акумуляторна батарея мала характеристики, недостатні для живлення всіх компонентів схеми і також була замінена.

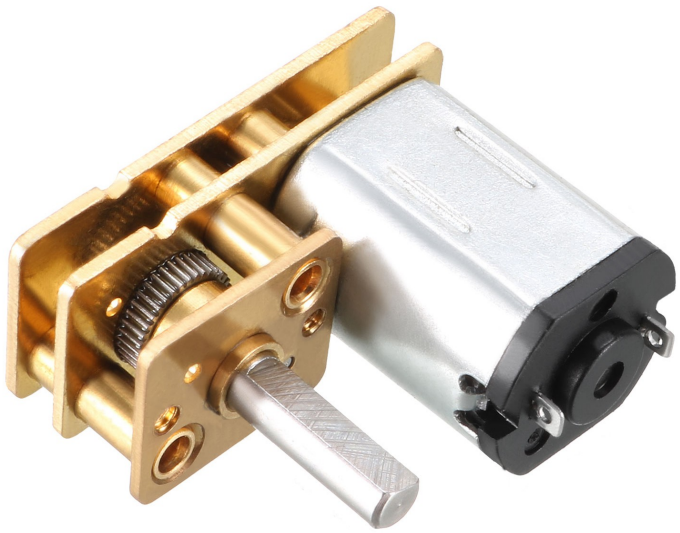


Рисунок 2.5 – Зовнішній вигляд двигуна без корпусу

Для приведення платформи в рух застосовується невеликий електродвигун постійного струму, що обертає задню вісь. У складі двигуна є простий диференціал, що дозволяє заднім колесам обертатися незалежно один від одного. Така особливість дозволяє зменшити вплив пробуксовування коліс під час руху. Крім того, такий пристрій задньої осі широко поширений на більшості сучасних транспортних засобів та наявність диференціала підвищить результативність подальшого проектування та тестування.

Двигун підтримує широкий діапазон роботи на різних швидкостях, що управляється поданою напругою. Це дозволяє краще імітувати рух транспортних засобів як на великій, так і низькій швидкості і точніше здійснювати бажані маневри.

Щоб керувати роботою електродвигуна, застосуємо модуль L298N. Дана мікросхема дозволяє контролювати як швидкість, так і напрямок обертання одночасно двох наборів двигунів постійного струму. Необхідність керувати декількома електродвигунами відсутня, але може бути застосована при необхідності тестування роботи пристрою з повнопривідними транспортними засобами, або з «бортовою» системою управління рухом.

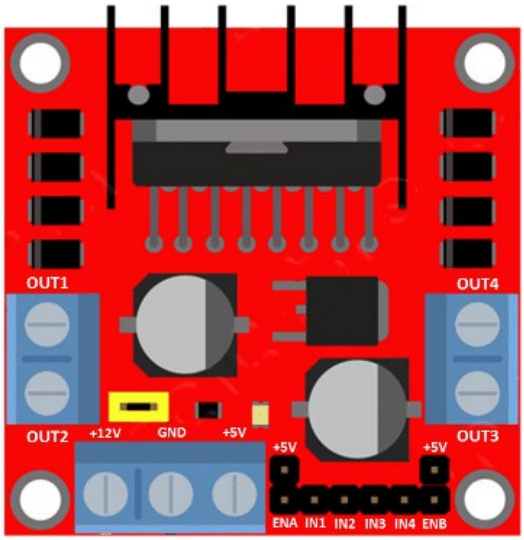


Рисунок 2.6 – Схематичне зображення плати L298N

Для живлення схеми підходить будь-яке джерело з напругою від 5 до 35 В і підключається в роз'єм +12V. Вбудований засіб зниження напруги дозволяє забезпечити стабільні 5 В. Так як при наявному джерелі живлення, напруга менше 12 В, то плата, крім управління рухом двигунів, використовується і функцію стабілізації напруги. Для ввімкнення такого режиму на контактах 5V enable встановлюється перемичка. З роз'єму +5V у такому разі можна отримати живлення для решти всіх компонентів пристрою.

Що стосується управління двигуном, то плата має 4 доступні комбінації на входах:

- IN1 = 1, IN2 = 1 – двигун стоїть на місці;
- IN1 = 0, IN2 = 0 – двигун стоїть на місці;
- IN1 = 1, IN2 = 0 – двигун крутитися в одному напрямку;
- IN1 = 0, IN2 = 1 – двигун крутитися в іншому напрямку.

Для керування швидкістю руху застосовується ШІМ сигнал, поданий на вхід EnableA. Входи IN3, IN4 та EnableB у нашому випадку залишаються непідключеними, але відповідають за керування другим двигуном.

Таким чином, щоб керувати роботою двигуна, на плату керування необхідно подавати обрані комбінації імпульсів та ШІМ сигнал. Це дозволяє спростити код проекту, оскільки жодні сторонні бібліотеки або складні контролери сигналів не потрібні.

Наявний на платформі сервопривід за багатьма характеристиками не підходив для проекту: низька швидкість роботи, велика кількість роз'ємів необхідних підключення, неможливість керувати кутом повороту, повністю пластикові шестірні, великий розмір.

Для виправлення всіх цих проблем був застосований сервопривід MG90S з наступними характеристиками:

- розмір 22,8 мм х 12,2 мм х 28,5 мм;
- швидкість обертання 60 градусів за 0,1 секунду;
- металеві складові;

- використовує два роз'єми для живлення та лише один для управління;

- підтримує напругу в діапазоні 4,8-6 В.

Для керування сервоприводом існує набір бібліотек під різні пристрої керування. Наприклад, для середи розробки Arduino IDE доступна бібліотека Servo.h.

Рисунок 2.7 – Зовнішній вигляд сервоприводу MG90S і його штатної комплектації

Щоб керувати сервоприводом, достатньо його проініціалізувати за допомогою функції servo.attach(N), де N – порядковий номер роз'єму на платі, до якого підключено порт керування сервоприводом.

Решта взаємодії зводиться до виклику функції servo.write(M), із зазначенням кута повороту щодо початкового положення замість символу M.

Наявність вже готової та протестованої бібліотеки дозволяє як спростити код проекту, так і не переживати за надійність роботи пристрою імітації рульового керування.

2.3.2 Засіб дистанційного керування

Як під час проектування, так і під час тестування необхідно мати повний контроль над пристроєм під час його роботи. Також для підвищення безпеки та зручності як при тестуванні малогабаритного прототипу, так і при подальших тестах на повнорозмірних транспортних засобах, найкращим варіантом буде реалізація дистанційного керування.

Для вирішення цього завдання підійде модуль HC-06, що дозволяє передавати дані між пристроями за протоколом Bluetooth 2.0. Хоча стандарт вже досить старий і програє своїм новим версіям Bluetooth 4.0 та 5.0 та альтернативам на кшталт WiFi або WiMAX за швидкістю та максимально доступним радіусом дії, обраний модуль виграє в ціні, а радіус дії та швидкість достатні для поставлених раніше завдань.

Крім того, при необхідності модуль може бути замінений у майбутньому на інший з мінімальними змінами у схемі та коді пристрою.

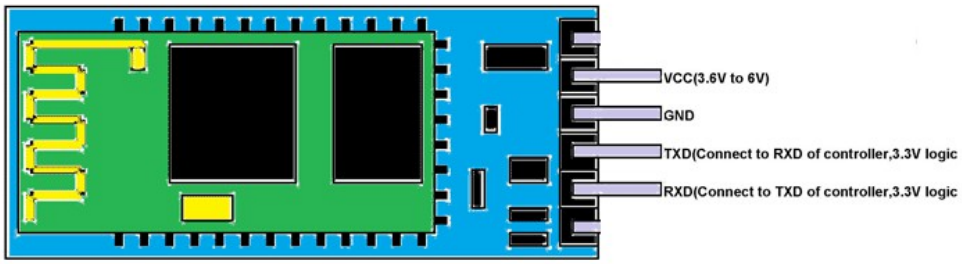


Рисунок 2.8 – Схематичне зображення плати HC-06

На відміну від модуля HC-05, плата постійно функціонує в режимі Веденого (Slave) та не підтримує режим Ведучого (Master). Таким чином, роль Ведучого виконуватиме зовнішній пристрій керування, який здійснюватиме пошук та підключення до нашої системи.

Плата HC-06 має наступні порти:

- RX – логічний вихід отриманих даних;
- TX – логічний вхід даних на передачу;
- GND – заземлення;
- + 5V – живлення.

Взаємодія плати Arduino з будь-яким модулем передачі даних здійснюється через цифрові порти введення та виведення Serial 0 (RX) та 1 (TX). Процес передачі даних у такому разі зводиться до простого зчитування повідомлень, що надійшли на модуль, і їх обробка. Для зчитування використається функція Serial.read(), що входить до складу базових функцій Arduino IDE.

2.3.3 Необхідні сенсори

Більшість функціоналу, який необхідно реалізувати, передбачає наявність у системі інформації про наявність перешкод та відстані до них. Одним із найкращих варіантів для таких датчиків буде ультразвуковий сенсор HC-SR04.

Для визначення відстані в датчику є приймач і випромінювач звукових імпульсів. При надходженні сигналу, випромінювач надсилає звукові хвилі з частотою 40 кГц, після цього, за наявності об'єкта на шляху руху хвиль, вони відбиваються і повертаються до пристрою. Там їх уловлює приймач, а система підраховує час, протягом якого повернувся сигнал і, знаючи швидкість звуку, визначає відстань до об’єкту.

Максимально допустима відстань до об'єкта у такого датчика становить 400 см, а кут охоплення від 22,5 до 30 градусів в залежності від бажаної точності. Таких характеристик більш ніж достатньо для аналізованих масштабів.

Проблеми датчика можуть виникати з об'єктами, які розсіюють або поглинають звукові хвилі. У такому разі результати виміру можуть не

відповідати реальним даним. Для вирішення цієї проблеми в сучасних автомобілях застосовується одночасно кілька типів датчиків, наприклад на додаток до ультразвукових можуть використовуватися інфрачервоні датчики, відеокамери, високоточні лідари. Таким чином вирішуються проблеми кожного з датчиків, адже інфрачервоні сенсори і лідари проблематично функціонують з об'єктами, що добре поглинають світло інфрачервоне випромінювання, а відеокамери погано збирають інформацію в середовищі, що слабко освітлено, наприклад вночі.

Для аналізованої системи реалізація кількох дублюючих систем вимірювання відстані буде надмірною і надто ресурсомісткою, тому на даному етапі найкращим вибором буде застосування саме ультразвукових датчиків за рахунок простоти, дешевизни та великого кута охоплення.

При інтеграції системи в повнорозмірні транспортні засоби, найкращим варіантом буде використання вже наявних у комплектації автомобілю датчиків та отримання інформації безпосередньо з них.

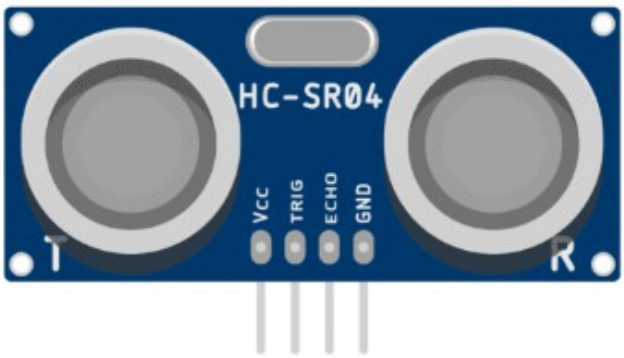


Рисунок 2.9 – Схематичне зображення датчику HC-SR04

Що стосується підключення датчиків HC-SR04 до загальної схеми, вони містять наступний набір роз'ємів:

- Trig – вхід для ініціалізації роботи датчика;

- Echo – вихід датчика, що стає логічною одиницею при поверненні сигналу;

- GND – заземлення;

- + 5V – живлення.

Для керування платою необхідно подати сигнал на порт Trig і засікти час до його повернення та появи сигналу на порту Echo. Після цього, отриманий час за формулою переводиться у відстань:

$$S = \frac{t * v_{sound}}{2}, \quad (2.1)$$

де S – відстань, t – час повернення сигналу, v_{sound} – швидкість звуку (дорівнює 340 м/с). Отримана відстань поділяється на два оскільки звуковий імпульс пройшов її двічі – до об'єкта і назад.

Оскільки у складі системи буде одночасно декілька датчиків, для спрощення взаємодії із ними використовуємо бібліотеку NewPing.h. Тоді все, що необхідно для знаходження відстані, це ініціалізувати кожен датчик, створивши для нього об'єкт типу sonar і викликати функцію sonar.ping_cm(). В результаті функція поверне вже відстань від датчика до об'єкта, а в разі його відсутності – 0.

Щоб реалізувати розпізнавання світлофорів необхідний засіб для отримання візуальної інформації, або інформації про світло і параметри його кольору.

Для таких цілей підходить багатофункціональний датчик APDS-9960. Він дозволяє визначати прості жести, наближення, рівень освітленості та складові кольору у колірній моделі RGB.

Всі вимірювання в датчику здійснюються за рахунок наявності інфрачервоного світлодіода для підсвічування об'єкта та набору з чотирьох фоторезисторів: по одному на кожен базовий колір та загальну освітленість. Завдяки різному розташуванню цих сенсорів, з'являється можливість визначати напрямок руху жестів або об'єктів перед модулем.

А наявність інфрачервоного світлодіода дозволяє виміряти відстань за

схожим з ультразвуковим датчиком принципом. Відмінність лише тому, що у розрахунках використовується швидкість світла, а не звуку.

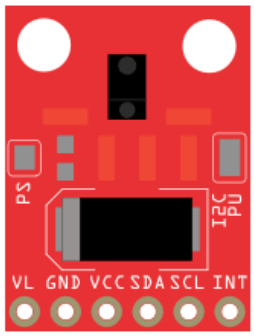


Рисунок 2.10 – Схематичне зображення датчику APDS-9960

Датчик працює при напрузі 3.3 В, так що його необхідно запитати незалежно від інших модулів – з самої плати Arduino. Для керування міститься три порти підключення:

- SDA – контакт для підключення до I2C для передачі даних;
- SCL – контакт для підключення до I2C для підрахунку часу;
- INT – виведення сигналу переривання.

Також плата може містити опціональний порт VL для зовнішнього живлення інфрачервоного світлодіода. А для взаємодії з датчиком у коді необхідно використовувати одну із запропонованих бібліотек, наприклад SparkFun_APDS9960.h

2.3.4 Живлення схеми

Щоб сам пристрій та тестова платформа залишалися мобільними під час роботи, живлення має здійснюватися від акумуляторних батарей. При встановленні на реальний транспортний засіб джерелом живлення виступатимуть внутрішні системи автомобіля: генератор та акумуляторна батарея.

У нашому випадку для цього було обрано батареї формату Li-ion

18650. Такий тип батарей широко поширений і варіюється за доступними характеристиками при однаковому розмірі. Для проекту були взяті два акумулятори ємністю 2400mah з номінальною напругою 3,6 В, чого більш ніж достатньо для живлення пристрою, що розробляється.

Щоб отримати необхідну для роботи компонентів напругу 5 В, батареї в схемі з'єднані послідовно. Таким чином їх напруга складається і при повному заряді дорівнює 8,4 В. Вибраний драйвер двигунів L298N містить засіб для зниження напруги до 5 В, так що акумулятори підключаються безпосередньо до нього. До самої плати управління двигунами у такому разі можна підключати живлення інших компонентів системи.

Крім того, наявність напруги в 8,4 В дозволяє підвищити максимальну швидкість обертання двигунів, адже вона залежить якраз від поданої напруги.

2.3.5 Підсумкова схема пристрою

Після завершення підбору компонентів та їх компонування в єдиний пристрій отримано наступну схему (рисунок 2.11).

Для простоти з'єднань і можливості швидкої заміни і установки компонентів, наприклад додаткових датчиків, всі з'єднання виконані за допомогою pin-коннектрів. Такі з'єднання фіксуються до наявних конекторів і забезпечують достатньо надійне з'єднання, не вимагаючи паяння компонентів.

Для підключення живлення до драйвера двигунів та самого двигуна використовуються передбачені на платі гвинтові з'єднання. Вони також не вимагають паяння та забезпечують ще більш жорстке та надійне з'єднання компонентів.

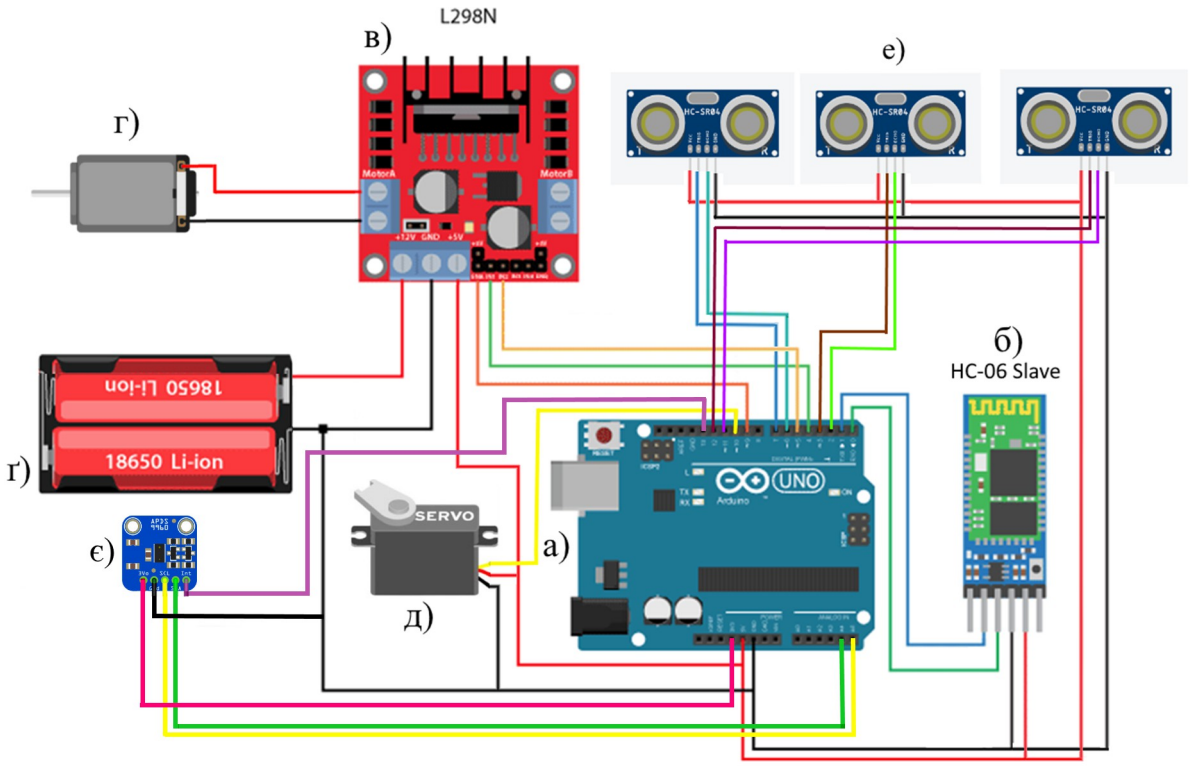


Рисунок 2.11 – Схематичне зображення пристрою та його компонентів: а) плата Arduino UNO R3; б) Bluetooth-модуль HC-06; в) драйвер двигунів L298N; г) електродвигун; ґ) блок акумуляторних батарей стандарту 18650; е) набір датчиків HC-SR04; є) датчик кольору APDS-9960

Крім того, за рахунок загального невеликого розміру пристрою, його достатньо легко розмістити на будь-якому каркасі або вбудувати в транспортний засіб без необхідності внесення кардинальних змін у внутрішній склад пристрою.

Також використана плата Arduino UNO дозволяє підключити до роз'ємів, що залишилися вільними, додаткові компоненти, наприклад датчики або інші рішення. А при необхідності значного підвищення кількості датчиків може бути використана плата розширення, що дає збільшити кількість доступних роз'ємів практично вдвічі.

3 МАТЕМАТИЧНА МОДЕЛЬ ДЛЯ ПРИЙНЯТТЯ РІШЕНЬ

Для того, щоб розроблюваний пристрій не виконував дії згідно із чітким алгоритмом, який дозволяє системі реагувати лише на передбачені на етапі написання коду події, застосовуються різні математичні абстракції та моделі прийняття рішень.

Такі системи можуть відрізнятися за способом внутрішньої організації, особливо системою прийняття рішень. Загальною рисою являється те, що рішення приймаютьсявиходячи з наявних у системи даних.

Найбільш поширеними в сучасній комп'ютерній техніці математичними моделями для прийняття рішення є кінцеві автомати, контролери на базі нечіткої логіки та нейронні мережі.

3.1 Кінцеві автомати

Кінцеві автомати являють собою абстрактну модель пристрою, що містить один вхід, один вихід і множину внутрішніх станів. Для переходу між станами вказуються різні умови, що відповідають за сигнали, що надходять на вхід. Також кожна вершина стану містить набір здійснюваних системою дій під час переходу до цього стану.

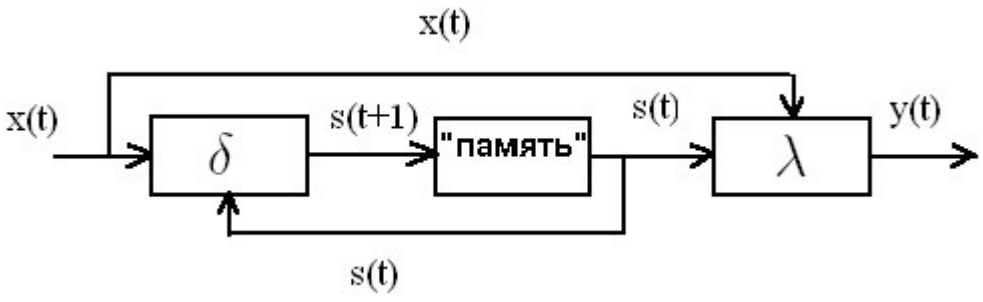


Рисунок 3.1 – Структура кінцевого автомату

Згідно з типовою структурою, кінцевий автомат складається з наступних компонентів:

- S – кінцева множина станів автомата;
- δ – функція переходів;
- λ – функція виходів;
- блок «пам'ять» – сукупність внутрішніх станів, де $S(t)$ – поточний стан, а $S(t+1)$ – наступний.

Крім того, кінцеві автомати поділяють на два класи: автомати Мілі та Мура. Вони відрізняються один від одного однією особливістю: у першого вихідний сигнал залежить лише від внутрішнього стану, а у другого – як від внутрішнього, так і від стану входу.

Для спрощення проектування та опису таких моделей, без використання формул з множинами, використовують два поширені способи завдання функціонування.

- Граф переходів, або діаграма станів – графічне представлення внутрішнього пристрою кінцевого автомата, у якому стани являються вершинами графа, а функції переходів – дугами між вершинами.
- Таблиця переходів – табличне уявлення функції δ. Кожен рядок таблиці співвідноситься з конкретним станом, а кожен стовпчик – з вхідним символом. У комірках на перетині станів і сигналів вказуються стани, у яких у разі відповідності інших параметрів повинен перейти автомат.

Як можна помітити з особливостей структурної реалізації, при проектуванні таких систем необхідно описати повний набір дій з умовами переходів між станами.

У зв'язку з цим дані, що отримуються на вхід, повинні бути чітко визначені на проектуванні і коректно вказані допустимі значення або діапазони для кожного переходу. В іншому випадку, залишаються можливі

ситуації, у яких ніякої дії не відбудеться, наприклад, при надходженні неврахованого сигналу.

Також, набір здійснюваних дій залишається обмеженим тим, що був реалізованим на етапі створення кінцевого автомата, і у разі непокриття всього можливого і необхідного функціоналу, вимагатиме повної перепроектування системи.

3.2 Нейронні мережі

З метою імітації роботи біологічних процесів живих істот, що протікають у мозку, були розроблені штучні нейронні мережі. За своєю суттю вони являються математичною моделлю, яка намагається точно повторити внутрішній пристрій нейронних мереж у мозку.

Основною особливістю такого підходу є навчання кінцевої моделі, що дозволяє досягти бажаної точності обчислень при достатній кількості ітерацій навчання, розмірі даних для навчання та ресурсах системи.

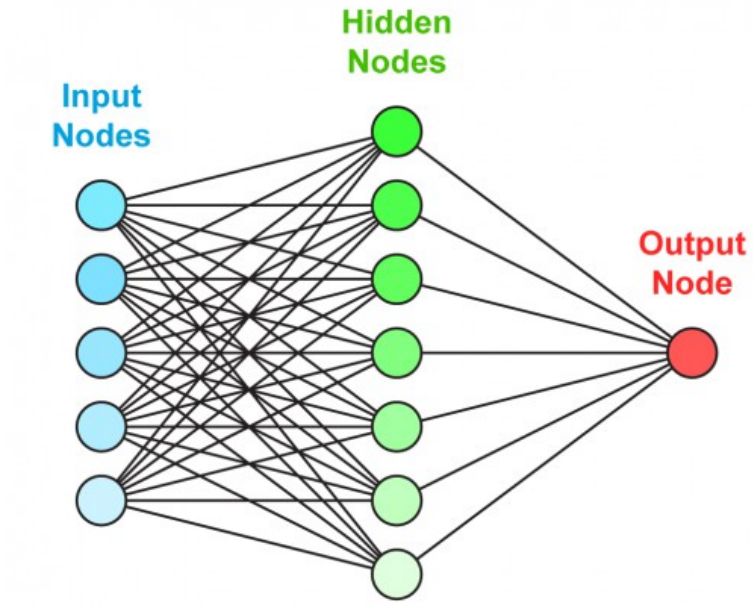


Рисунок 3.2 – Структура штучної нейронної мережі

Навчальні штучні нейронні мережі за своєю структурою являють собою перцептрон – математичну модель сприйняття інформації мозком. Він, у свою чергу, складається з трьох типів шарів, що відрізняються за своїми функціональними завданнями.

Вхідний шар – являє собою набір нейронів, що відповідають за надходження інформації в математичну модель. Вони не містять у собі обчислювальних процесів, а лише є індикаторами появи впливу на систему і передають дані далі.

Прихований шар – містить у собі множину проміжних нейронів, що виконують основні операції перетворення над сигналами. Таких шарів може бути декілька, що позитивно впливає на навчання та універсальність нейронної мережі, але дається взнаки значного зниження продуктивності.

Вихідний шар – набір нейронів, відповідальних за отримання кінцевих даних. Так само як і в прихованому шарі виконують перетворення отриманих сигналів.

Кожен нейрон на вхідному рівні просто передає дані далі. На двох інших на вхід нейрона потрапляє інформація з усіх нейронів попереднього шару. Після цього відбувається нормалізація за допомогою функції активації та переход на наступний рівень.

Дані, що передаються між нейронами, є числами в діапазонах [0,1] або [-1,1]. Тому, щоб вхідні дані відповідали цій умові, вони нормалізуються із застосуванням різних по своєму вигляду функцій.

Зв'язки між нейронами називаються синапсами та мають параметр ваги. Він дозволяє визначати, інформація з якого нейрона буде важливішою на наступному рівні, відповідно до зазначеного коефіцієнта ваги кожного зв'язку.

Саме за рахунок наявності ваги інформація всередині нейронних мереж обробляється в необхідний результат. Крім того, чим точніше і якісніше налаштовані ваги кожного зв'язку, тим точнішим і кращим буде результат.

Навчання здійснюється для встановлення вагів у автоматичному

режимі і виконується двома способами: з наявністю вчителя та без нього. У першому випадку нейронної мережі надаються набори вхідних даних і бажані при таких параметрах вихідні. Використовуючи внутрішні алгоритми, система розставляє первинні ваги і, після цього, донавчається на основі отриманих даних поступово змінюючі ваги для підвищення точності кінцевого результату.

У другому випадку виходи формуються самостійно, а ваги налаштовуються за алгоритмом, що враховує тільки вхідні та внутрішні сигнали.

3.3 Система підтримки прийняття рішень на основі нечіткої логіки

Ще одним способом реалізації системи управління є застосування нечіткої логіки. Нечітка логіка є середнім між теорією множин і логікою, включаючи їх принципи функціонування для взаємодії з введеним поняттям нечіткої множини.

Остання, в свою чергу, є об'єктом, належність до множини яких визначається функцією, яка приймає не лише значення 0 чи 1, а й значення у їх інтервалі, тобто [0,1].

За рахунок такого пристрою з'являється можливість реалізувати лінгвістичні змінні для роботи з «людським» сприйняттям значень. Наприклад, температуру в приміщенні можна описати трьома варіантами: холодно, тепло та спекотно.

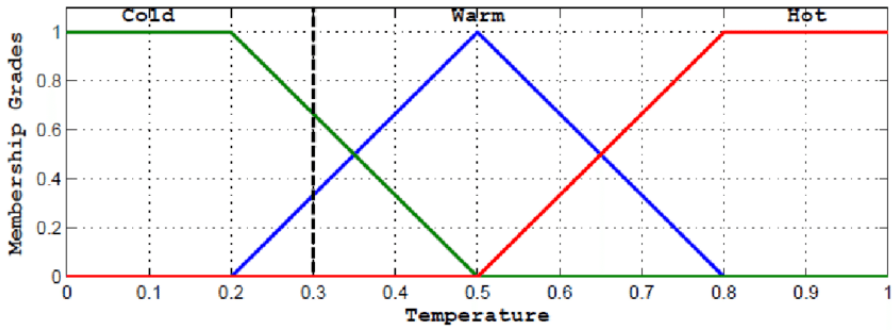


Рисунок 3.3 – Приклад лінгвістичної змінної та її функцій належності до різних множин

Причому перехід від значення до іншого відбувається не різко, а плавно, що залежить від застосовної функції належності. Тим самим, будь-яке подане значення має ступінь належності до множин усередині змінної.

За рахунок такої реалізації змінних утворюється можливість виконання логічних операцій над ними, наприклад І, АБО, НЕ. А вже за рахунок наявності таких операцій можна реалізувати систему нечіткої логіки, рішення в якій приймаються, ґрунтуючись на базі нечітких правил ЯКЩО-ТО.

Всі правила формуються за загальним шаблоном і дозволяють просто описати бажані від системи дії «людськими словами». Приклад правила для функції оцінки показників здоров'я людини:

ЯКЩО Зріст є Високий І Вага є Середній ТО Здоров'я є В_нормі.

У такому разі, при надходженні значень на вхід системи вони обробляються відповідно до базису правил, при яких над множинами виконуються зазначені в правилах операції. В результаті виходить нечітка множина значень, що результують по кожному з правил.

Головною проблемою такої системи обробки даних являється те, що входи та виходи є нечіткими множинами, а більшість комп'ютерних систем вміють коректно працювати і підтримують тільки реальні значення.

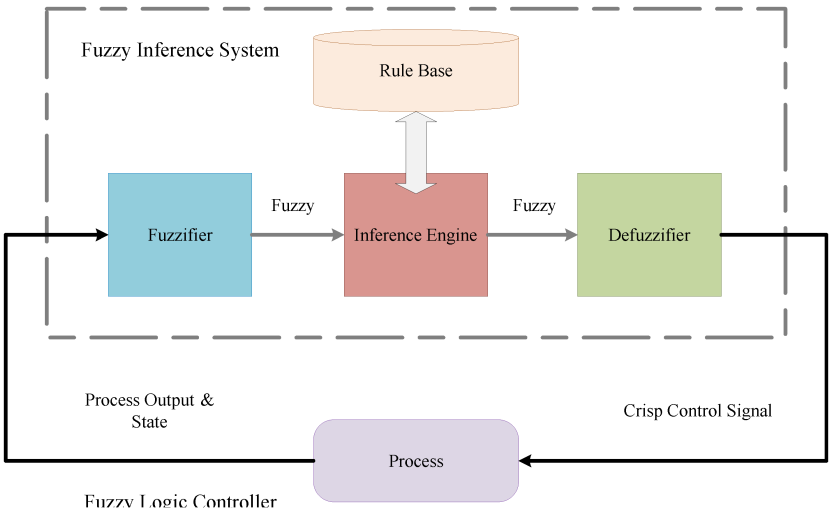


Рисунок 3.4 – Система нечіткої логіки з фазифікатором та дефаззифікатором

Для вирішення цієї проблеми до складу системи додаються два модулі: фазифікатор на вході та дефаззифікатор на виході. Як випливає з назв, модулі являються засобами для переведення даних з одного виду в інший. Фаззифікатор відображає чітке значення змінної у нечіткій множині, а дефаззифікатор – нечіткі множини у чітке значення виходу. Всі інші компоненти системи залишаються такими самими і не вимагають змін.

Отримані системи дозволяють застосовувати їх під час роботи з реальними значеннями, тобто у більшості комп'ютерних систем. Також є доступ до використання нечітких ЯКЩО-ТО правил. Пристрій фазифікатора, механізму нечіткого виведення та дефаззифікатора може змінюватися для кращої відповідності до поставленої задачі. Для ефективного поєднання чисельної та лінгвістичної інформації можуть бути розроблені додаткові алгоритми налаштування системи нечіткої логіки.

3.4 Вибір математичної моделі для реалізації

Виходячи з усіх викладених описів та особливостей реалізованого проекту, найбільш доречною буде система на основі нечіткої логіки. Це пов'язано в першу чергу з тим, що інформація, що отримується з датчиків системи, що розробляється, знаходиться в відносно обмежених діапазонах.

При цьому розробка кінцевого автомата для охоплення всіх можливих станів буде досить трудомістким та неефективним до комп'ютерних ресурсів завданням. Дані, отримані з сенсорів, не являються стандартизованими, а входять у діапазони значень, допустимих технічним пристроєм датчика.

Якщо розглядати такий автомат у вигляді графа, необхідно буде реалізувати окремий стан під кожне значення і практично кожна вершина буде взаємопов'язана з іншими. За рахунок цього у загальному пристрої можливі помилки та неточності умов переходів. А також значною мірою

знизиться швидкодія системи.

Що стосується нейронних систем, то через складність отримання коректного набору навчальних даних та обмеженість продуктивності системи, їх застосування для автоматизації всіх розглянутих процесів у взаємодії з транспортними засобами так само не є оптимальним.

Отримання набору навчальних даних передбачає наявність внутрішньої системи їхнього збору, наприклад за рахунок запису всіх дій водія та часу їх виконання. Навіть за наявності засобу для їх збору в системі, немає конкретного зв'язку між ними і даними з датчиків по периметру пристрою. У такому випадку велика ймовірність, що навчання системи не призведе до бажаного результату, або вимагатиме занадто великої кількості ітерацій та ресурсів від системи.

Таким чином, саме система на базі нечіткої логіки дозволить значною мірою спростити розробку, є оптимальною для наявних обчислювальних потужностей системи і дозволить з великою ефективністю обробляти велику кількість можливих ситуацій при застосуванні з розглянутим функціоналом кінцевого пристрою. Крім того, за рахунок використання базису правил ЯКЩО-ТО, всі дії, необхідні для виконання системою, досить просто і зрозуміло описуються, співвідносячись з досвідом і відчуттями реального водія, буть то відчуття швидкості, відстані, достатнього місця чи інші.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ

Для написання програмної частини проекту під платформу Arduino UNO використовується інтегроване середовище розробки Arduino IDE. Програмний продукт підтримує мови програмування C і C++ і містить редактор коду, систему пошуку помилок та інструментарій для автоматичного перетворення виконуваного коду в текстовий файл у шістнадцятковому кодуванні та його запису на плату Arduino.

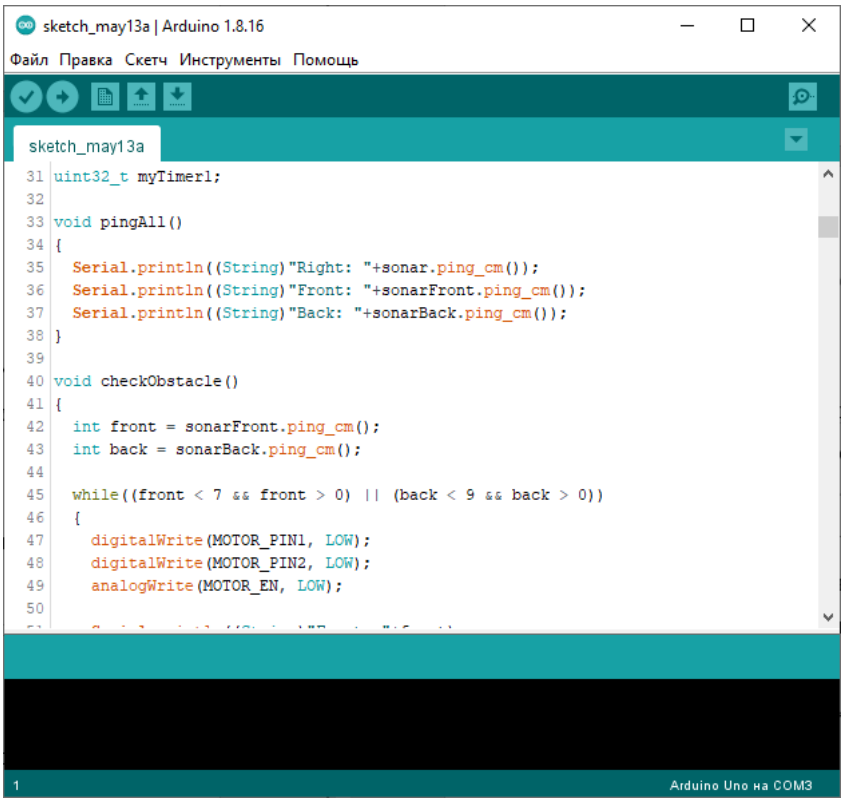


Рисунок 4.1 – Інтерфейс середовища розробки Arduino IDE

Що стосується вимог до програмного продукту, що розробляється, то він повинен відповідати наступному переліку вимог:

- код повинен мати просту та зрозумілу структуру, мати невеликий розмір та бути оптимізованим, щоб без проблем поміщатися та швидко оброблятися на платформі Arduino;

- структура коду має бути модульною, із застосуванням універсальних алгоритмів та з можливістю при необхідності легко та швидко внести зміни для модифікації застосовуваних рішень, підвищення наявного функціоналу та для впровадження підтримки додаткових компонентів;

- програмний код повинен містити засоби для спрощення процесу тестування та налаштування пристрою, методи для простої діагностики та збору інформації з датчиків;

- отриманий програмний продукт повинен виконувати поставлені в проекті завдання.

4.1 Реалізація системи на базі нечіткої логіки

Оскільки було вирішено реалізувати систему управління на базі нечіткої логіки, то для цього існує готова протестована бібліотека з відкритим вихідним кодом eFLL (Embedded Fuzzy Logic Library). Вона містить весь необхідний функціонал та типи даних для роботи з нечіткими множинами, лінгвістичними змінними та правилами. Бібліотека містить натсупні компоненти:

- Fuzzy – основний об'єкт, є підсумковою системою прийняття рішень на основі нечіткої логіки;
- FuzzyInput - об'єкт, що групує нечіткі множини в єдиний вхід системи;
- FuzzyOutput – ідентичний до FuzzyInput, але групує множини у виходи;
- FuzzySet – нечітка множина, підтримує три види функцій приналежності: трикутну, трапецієподібну та лінійну, що визначається переданими параметрами;
- FuzzyRule – об'єкт для реалізації нечітких правил та прив'язки їх до системи нечіткої логіки;
- FuzzyRuleAntecedent – компонент нечіткого правила, що містить

логічну операцію з входами системи;

- FuzzyRuleConsequent – компонент нечіткого правила, використовується для прив'язки вихідної множини до правила.

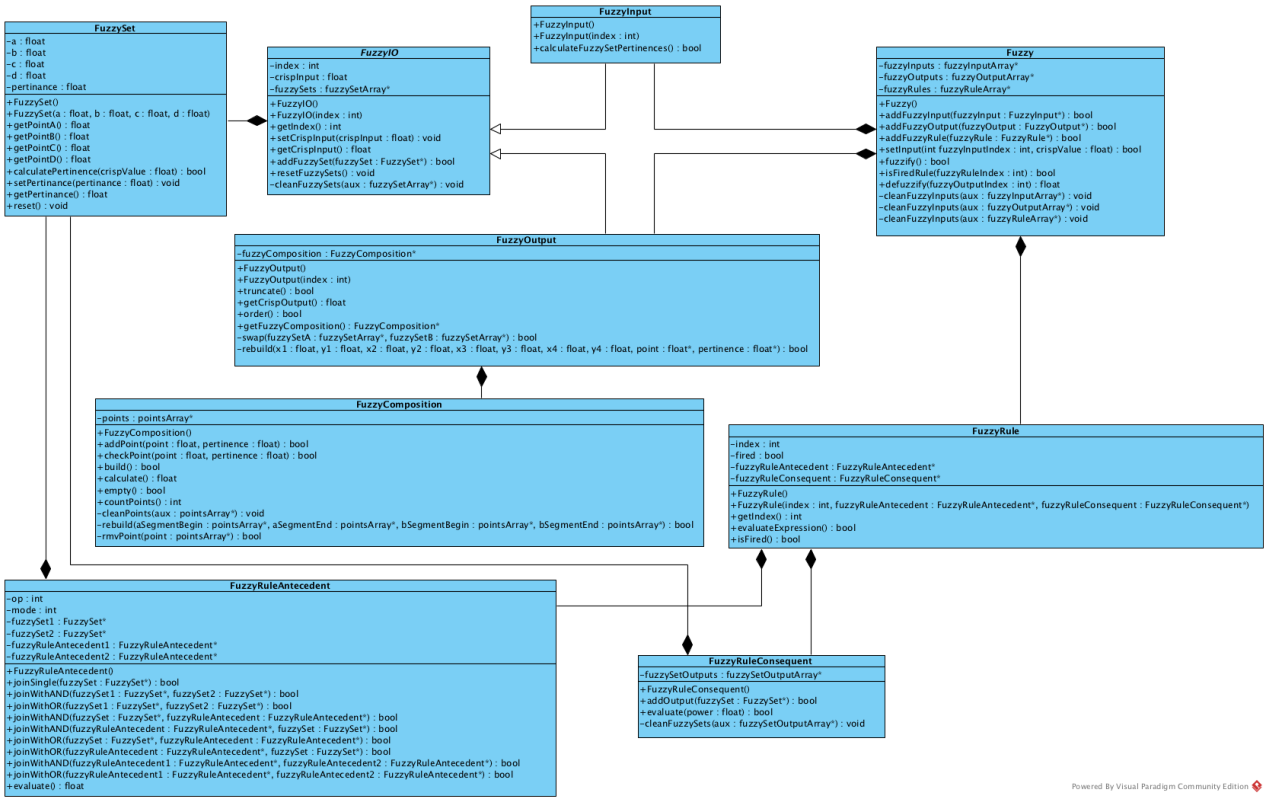


Рисунок 4.2 – Схема взаємодії, поля та методи об'єктів класу Fuzzy.h

Крім того, об'єкт Fuzzy містить у собі методи для фазифікації та дефазифікації даних, що дозволяє легко впровадити рішення на базі нечіткої логіки в будь-який існуючий проект.

Для використання в проекті достатньо вказати її імпорт на початку проекту: #include <Fuzzy.h>. А потім створити необхідний об'єкт і виконати налаштування системи, в якому беруть участь усі описані раніше об'єкти.

Лістинг 4.1 – Створення та часткова ініціалізація системи підтримки прийняття рішень на основі нечіткої логіки

```
Fuzzy *fuzzy = new Fuzzy();
. . .
void setup() {
. . .
      //depth
      FuzzySet *narrowD = new FuzzySet(0, 0, 0, 30);
      FuzzySet *enoughD = new FuzzySet(23, 37, 38, 53);
      FuzzySet *deepD = new FuzzySet(45, 50, 70, 100);
. . .
      FuzzyInput *depth = new FuzzyInput(1);
      depth->addFuzzySet(narrowD);
      depth->addFuzzySet(enoughD);
      depth->addFuzzySet(deepD);
      fuzzy->addFuzzyInput(depth);
. . .
      FuzzyRuleConsequent     *thenparkingMethodParal     =     new
FuzzyRuleConsequent();
      thenparkingMethodParal->addOutput(paral);

      FuzzyRuleAntecedent    *ifLengthShortORDepthNarrow    =    new
FuzzyRuleAntecedent();
      ifLengthShortORDepthNarrow->joinWithOR(shortL, narrowD);
      FuzzyRule      *fuzzyRule1      =      new      FuzzyRule(1,
ifLengthShortORDepthNarrow, thenparkingMethodNotEnough);
      fuzzy->addFuzzyRule(fuzzyRule1);
```

Що стосується проектування системи, реалізації нечітких правил, вибору функцій власності, то для цього завдання найкраще підходить пакет прикладних програм для вирішення завдань технічних обчислень MATLAB.

Цей пакет містить у собі програмний продукт Fuzzy logic toolbox для проектування систем прийняття рішень на базі нечіткої логіки. Для його виклику необхідно написати команду fuzzy у консолі.

В результаті відкриється візуальний інтерфейс для проектування та налаштування системи. Для реалізації бажаного функціоналу, а точніше автоматичного паркування, круїз-контролю та визначення сигналу світлофора, необхідно реалізувати три окремі системи, оскільки необхідні на вході дані та дані на виході у всіх випадках не співпадають.

4.2 Керування моделлю

Програмний продукт на цьому етапі розробляється для перевірки функціонування концепції та надійності можливих до застосування алгоритмів роботи системи.

Насамперед необхідно зробити підключення наявних на шасі компонентів до системи, їх інтеграцію в код та первинне налаштування. Як було зазначено раніше, для полегшення взаємодії з компонентами найкращим рішенням являється використання готових та протестованих бібліотек.

Таким чином, на самому початку проекту відбувається імпорт бібліотек Servo.h та NewPing.h, які відповідають за керування сервоприводом, тобто рульовим управлінням, та ультразвуковими датчиками, встановленими по периметру пристрою. Також імпортуються бібліотека SparkFun_APDS9960.h для керування датчиком кольору та Fuzzy.h для створення контроллерів на базі нечіткої логіки

Усі підключені пристрої вказуються через директиву препроцесора #define, де кожному порту підключення вказується відповідний номер порту на платі.

Перевагою такої вказівки є простота зміни компонування схеми у майбутньому, адже не доведеться вносити виправлення в інший код, а лише поміняти порт підключення в директиві, тобто всього в одному місці. Сам код також стає більш читабельним, адже замість постійних звернень до роз'ємів під номерами відразу вказуються їх назви з легко зрозумілим призначенням і функціоналом.

Також зазначені у директиві значення не займають програмну пам'ять – при компіляції всі застосування цих директив замінюються на відповідні звернення до портів.

На цьому етапі таким же чином вказуються налаштування внутрішніх систем, наприклад допустимі розміри при пошуку паркувального місця і

відстань до перешкод. Таким чином, код легко модифікується для функціонування з транспортними засобами іншого розміру, як більшого, так і меншого.

Лістинг 4.2 – Ініціалізація компонентів

```
#include <NewPing.h>
#include <Servo.h>
#include <SparkFun_APDS9960.h>
#include <Fuzzy.h>

#define PIN_TRIG 12       //правий ультразвуковий
#define PIN_ECHO 13
#define MAX_DISTANCE 200

...

#define MIN_SPOT_SIZE 50  //ширина парковочного місця
#define MIN_SPOT_DEPTH 25 //глибина парковочного місця
#define MIN_DIST 7        //відстань до машины зліва

Servo servo1;
NewPing sonar(PIN_TRIG, PIN_ECHO, MAX_DISTANCE);
SparkFun_APDS9960 apds = SparkFun_APDS9960();
...
Fuzzy *trafficLightDetector = new Fuzzy();
```

Для спрощення взаємодії з двигунами була створена невелика функція movVeh(int _M1, int _M2), що приймає напрямок руху у якості параметрів. Вона необхідна щоб щоразу не використовувати в коді три команди для керування двигуном: дві для вибору напрямку та одна для встановлення швидкості.

Для зупинки двигуна та повернення керма в пряме положення наявна функція stopVeh().

Щоб точніше контролювати пройдену відстань, використовується функція для руху вказаний час – movTime(int _M1, int _M2, int _time). У ролі третього параметра, на відміну від простої функції руху, додається значення часу в мілісекундах.

Також у функції руху додано перевірку наявності перешкод на шляху прямування. Для цього необхідний для проходження інтервал розбивається

на частини, між якими здійснюється перевірка викликом функції checkObstacle().

Вона у свою чергу отримує інформацію з розташованого за напрямком руху датчика про доступну відстань до перешкоди і, залежно від отриманого значення, зупиняє транспортний засіб доти, доки об'єкт на шляху не зникне або продовжує рух за відсутності перешкод.

Лістинг 4.3 – Функції руху та перевірки відсутності перешкод

```
void movVeh(int _M1, int _M2) //рух у вказаному напрямку
{
  digitalWrite(MOTOR_PIN1, _M1);
  digitalWrite(MOTOR_PIN2, _M2);
  analogWrite(MOTOR_EN, MOTOR_SPEED);
}

void stopVeh() //зупинка
{
  digitalWrite(MOTOR_PIN1, LOW);
  digitalWrite(MOTOR_PIN2, LOW);
  analogWrite(MOTOR_EN, LOW);
  servo1.write(55);
}
void movTime(int _M1, int _M2, int _time) //рух вказаний час
{
  int t;
  for(t = 0; t < _time; t+=50)
  {
    checkObstacle();
    movVeh(_M1, _M2);
    delay(50);
  }
  stopVeh();
}

void checkObstacle() //перевірка перешкод
{
  int front = sonarFront.ping_cm();
  int back = sonarBack.ping_cm();

  while((front < 7 && front > 0) || (back < 9 && back > 0))
  {
    digitalWrite(MOTOR_PIN1, LOW);
    digitalWrite(MOTOR_PIN2, LOW);
    analogWrite(MOTOR_EN, LOW);
    front = sonarFront.ping_cm();
    back = sonarBack.ping_cm();
    delay(1000);
  }
}
```

Однією з функцій пристрою, що розробляється, є система автоматичного паркування. Автомобіль повинен сам рухатися, знаходити відповідне місце для паркування і, не зачепивши інші транспортні засоби або стіни, акуратно заїхати в його межі.

Це досить важке завдання навіть для водіїв, адже залежно від доступного місця, габаритів транспортного засобу та навіть стану дорожнього полотна, маневри необхідно коригувати та постійно стежити за дзеркалами, щоб нічого не зачепити.

Існує кілька типів паркувальних місць, як і способів паркування. Вони варіюються залежно від способу розташування транспортних засобів відносно один одного.

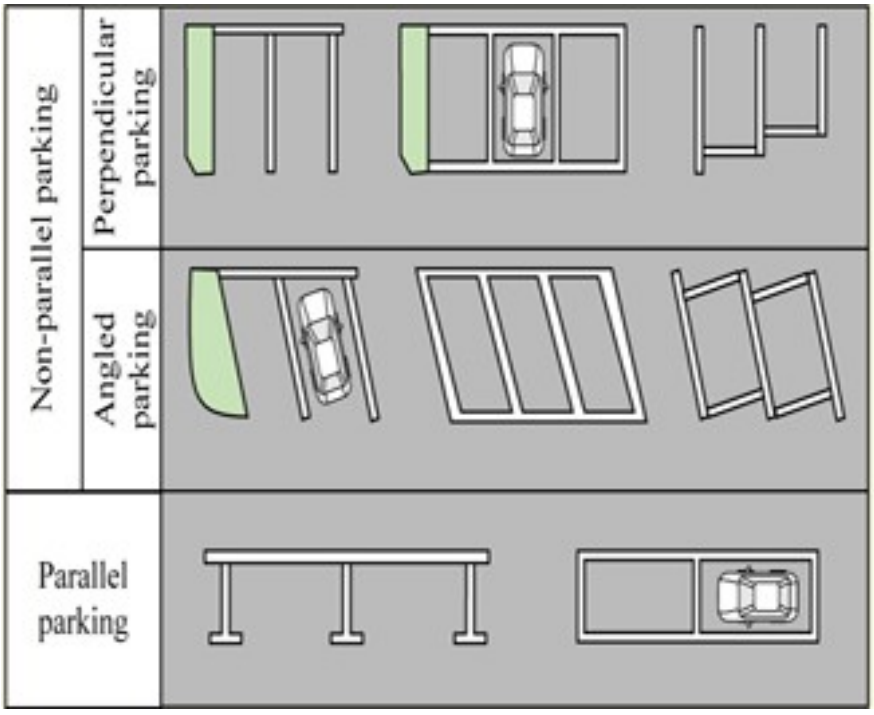


Рисунок 4.3 – Найбільш типові місця для паркування та їх розмітка

Види паркувальних місць поділяються на дві групи: паралельні та перпендикулярні. Напрямок залежить від орієнтації у просторі місця по відношенню до дороги: при паралельному місце розташовано вздовж дороги, а машини розміщені у напрямку руху; при перпендикулярному –

перпендикулярно до напрямку руху, причому як під прямим кутом, так і можливі варіанти паркування під кутом, «ялинкою».

Що стосується складності виконання маневрів, то паралельне паркування та паркування під прямим кутом вимагають найбільших навичок від водія, особливо при малій кількості наявного простору біля паркувального місця та його невеликих габаритах. Парковка під кутом, навпаки, є найпростішою і легко здійсненною навіть у вузьких проїздах. Тому, на стадії планування було прийнято рішення не реалізовувати функціонал для паркування «ялинкою», а зосередитися на двох інших варіантах, оскільки вони складніші, а можливість виникнення аварійних ситуацій у них вище.

4.3 Алгоритм паралельного паркування

Першим етапом здійснення будь-якої парковки являється пошук та вибір відповідного місця. Його має бути достатньо для розміщення автомобіля, як за шириною, так і за довжиною, при чому повинен залишатися запас по довжині мінімум в 1 м попереду та позаду від автомобілю. Додаткове місце необхідно для маневрів і без нього паркування здійснити не вийде або буде дуже важко.

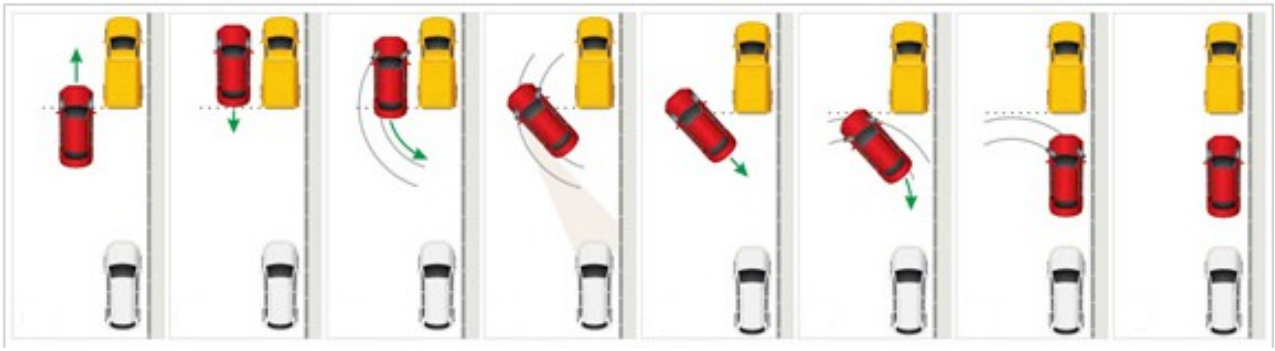


Рисунок 4.4 – Процес паралельного паркування

Весь процес паркування паралельно до дороги може бути поділений на наступні етапи:

- після знаходження відповідного місця, автомобіль необхідно встановити паралельно з розміщеним попереду, на одній лінії із заднім бампером, або навіть трохи попереду, це залежить від індивідуальних параметрів кожного транспортного засобу: габаритів та особливостей кермового управління;
- наступним етапом буде поворот керма та передніх коліс вправо та здійснення руху назад доти, доки автомобіль не встановиться під кутом 45 градусів по відношенню до місця стоянки;
- потім, кермо переводиться в початкове положення і автомобіль здає назад до перетину межі місця паркування заднім бампером;
- кермо повертається вліво, а автомобіль продовжує рух назад по дузі, до повного перетину меж паркувального місця та встановлення транспорту паралельно дорозі та припаркованим попереду та ззаду автомобілям;
- останнім етапом є корекція положення – при необхідності можна проїхати трохи вперед або назад, щоб залишити достатню відстань для вільного виїзду автомобілів попереду та позаду.

Як можна помітити, процес справді складний і складається з багатьох етапів, а водій повинен залишатися зосередженим на всіх його етапах.

Що стосується автоматизації цього процесу, то вона також не найпростіша, адже через велику кількість дій легше припуститися помилки в русі, а органи зору автоматики замінені сенсорами, які дають дещо інше уявлення про поточну дорожню ситуацію.

Змінна paral приймає значення 1, воно необхідно надалі у разі використання функції автоматичного виїзду з місця паркування.

Лістинг 4.4 – Функція паралельної парковки

```
void  paralPark()
{
    delay(1000);
    servo1.write(15);
    delay(1000);
    movTime(LOW, HIGH, 1000);   //назад та направо
    servo1.write(55);
    movTime(LOW, HIGH, 400);    //назад
    servo1.write(100);
    delay(1000);
    movTime(LOW, HIGH, 750);    //назад та наліво
    movTime(HIGH, LOW, 100);    //зупинка
    movTime(HIGH, LOW, 500);    //вперед
    paral = 1;
}
```

4.4 Алгоритм перпендикулярного паркування

Перпендикулярне паркування найбільш поширене на стоянках, наприклад біля великих супермаркетів, офісних центрів та стадіонів. Також воно застосовується у будь-якому критому чи підземному паркінгу.

Таке розміщення автомобілів дозволяє на досить невеликому просторі розмістити велику кількість автомобілів, тому широко використовується в місцях, де потенційне скупчення транспортних засобів велике.

Що стосується маневрів при здійсненні такого паркування, то вони дещо простіші, ніж при паралельному. З іншого боку, залежно від наявного простору дій може бути як більше, так і менше.

Процес складається з наступних етапів:

- спочатку також вибирається місце, достатнє по глибині та ширині під габарити автомобіля;
- транспортний засіб встановлюється біля переднього краю місця паркування;
- після цього колеса повертаються вліво і автомобіль рухається вперед дугою, поки не встановиться під кутом 45 градусів відносно місця

паркування;

- колеса повертаються вправо і автомобіль виконує рух назад, доки не вирівняється паралельно з бічними напрямними паркувального місця;
- відстань, що залишилася до заднього краю доступного місця, автомобіль проїжджає з колесами в початковому положенні і рухається доти, доки повністю не перетне передню межу паркувального місця.

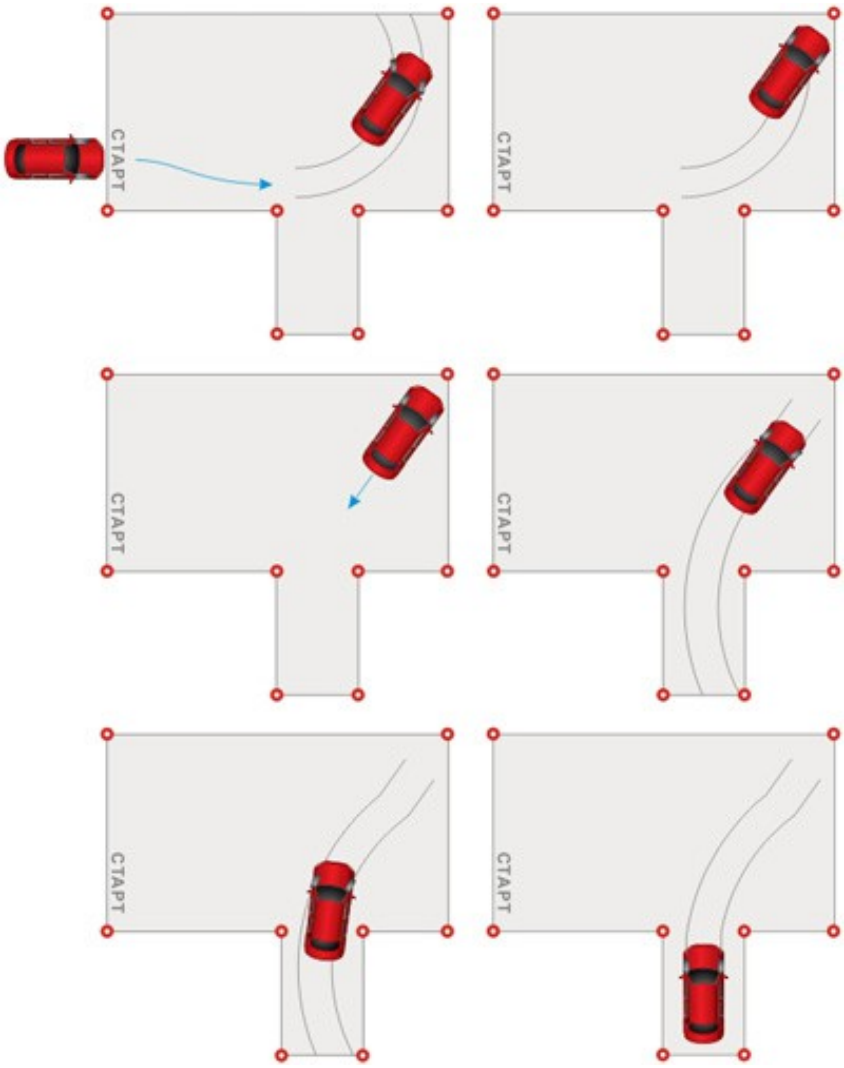


Рисунок 4.5 – Схематичне зображення заїзду в гараж

Алгоритм автоматичного паркування в розроблюваному пристрої схожий на паралельний. Основна відмінність полягає в наборі виконуваних маневрів.

Лістинг 4.5 – Функція паркування у гараж

```
void garagePark()
{
    delay(1000);
    servo1.write(100);
    delay(1000);
    movTime(HIGH, LOW, 1500); //вперед та наліво
    delay(1000);
    servo1.write(15);
    delay(1000);
    movTime(LOW, HIGH, 1400); //назад та направо
    delay(1000);
    servo1.write(55);
    delay(1000);
    movTime(LOW, HIGH, 500);  //назад
    paral = 0;
}
```

Крім того, оскільки алгоритм виїзду з місця паркування повністю ідентичний заїзду на нього і має зворотний порядок, був реалізований функціонал для автоматичного залишення місця стоянки.

Функція unpark() запускає той чи інший алгоритм залежно від здійснених раніше дій і не допускає запуску методу, який не підходить для поточного способу паркування.

Лістинг 4.8 – Функції залишення місця паркування

```
void paralOut()
{
    movTime(LOW, HIGH, 600); //назад
    movTime(HIGH, LOW, 100); //зупинка
    delay(1000);
    servo1.write(100);
    delay(1000);
    movTime(HIGH, LOW, 1000); //вперед і ліворуч
    delay(1000);
    servo1.write(55);
    movTime(HIGH, LOW, 800); //вперед
    servo1.write(15);
    delay(1000);
    movTime(HIGH, LOW, 900); //вперед і праворуч
}

void garageOut()
```

```
{
    movTime(HIGH, LOW, 700); //вперед
    delay(1000);
    servo1.write(15);
    delay(1000);
    movTime(HIGH, LOW, 1500);//вперед і праворуч
    delay(1000);
    servo1.write(55);
    delay(1000);
    movTime(LOW, HIGH, 400); //назад
    delay(1000);
    servo1.write(100);
    delay(1000);
    movTime(LOW, HIGH, 1400);//назад і ліворуч
}

void unpark()
{
  if (paral == 1)
    paralOut();
  else
    garageOut();
}
```

4.5 Загальний алгоритм паркування

Щоб ще більше спростити взаємодію та звільнити водія від процесу пошуку паркувального місця, було вирішено реалізувати необхідний функціонал. Таким чином від водія достатньо встановити транспортний засіб паралельно дорозі, розмітці або іншим припаркованим автомобілям і передати управління автоматиці.

Система пошуку також працює за досить суворим алгоритмом:

- транспортний засіб збирає поточну інформацію з датчиків про об'єкти навколо та починає рух уперед;
- у разі відсутності об'єктів праворуч від автомобілю система починає підрахунок доступного місця;
- проїжджаючи невеликі ділянки, система вимірює глибину доступного місця для паркування, а з пройденої відстані визначає його довжину;
- у разі появи іншого об'єкта, що заважає здійсненню паркування в даному місці, система продовжує рух до появи нового

доступного простору;

- якщо місця достатньо для здійснення того, або іншого методу паркування, то система в автоматичному режимі робить вибір відповідного методу і переходить до його виконання.

Лістинг 4.6 – Функціъ пошуку та аланізу наявного місця для паркування

```
void park()
{
  while(true) {
    int tempDist = sonar.ping_cm();
    if (tempDist > MIN_DIST || tempDist == 0) {
      checkSpace();
      fuzzy->setInput(1, spotSize);
      fuzzy->setInput(2, spotDepth);
      fuzzy->fuzzify();
      float parkingM = fuzzy->defuzzify(1);

      if (parkingM >= 0.5) {
        movTime(HIGH, LOW, myDelay);
        paralPark();
        break;
      }
      else if (parkingM > 0.4) {
        movTime(LOW, HIGH, myDelay);
        garagePark();
        break;
      }
    }
    movTime(HIGH, LOW, 300);
    delay(300);
  }
}
void checkSpace() {
  spotSize = 0;
  unsigned int tempDist = 100;
  float depth = 0;
  int iterCount = 0;

  while(tempDist > MIN_DIST) {
    if(spotSize >= MIN_SPOT_SIZE) {
      break;
    }
    movTime(HIGH, LOW, 300);
    spotSize+= 5;
    tempDist = sonar.ping_cm();
    depth+=tempDist;
    iterCount++;
    delay(350);
```

```
    }
}
```

Вся інформація про наявне місце береться з ультразвукового датчика, розташованого по правому борту транспортного засобу. У перевірках доступної відстані, вона порівнюється із зазначеними в директивах достатніми габаритами для паркування та нулем, оскільки у разі великої відстані або відсутності об'єкта на шляху звукових хвиль датчик повертає значення 0.

Функція перевірки доступного місця винесена окремо і виконується до тих пір, поки не з'явиться перешкода, або поки доступне місце відповідатиме мінімально необхідному. На основі отриманих даних виконується вибір наступного кроку.

Система, що використовується для прийняття рішення про придатний спосіб паркування була спершу спроектована в програмному пакеті MATLAB. Вона містить в собі дві вхідні множини – довжина і ширина місця, і одна вихідна – придатний спосіб паркування.

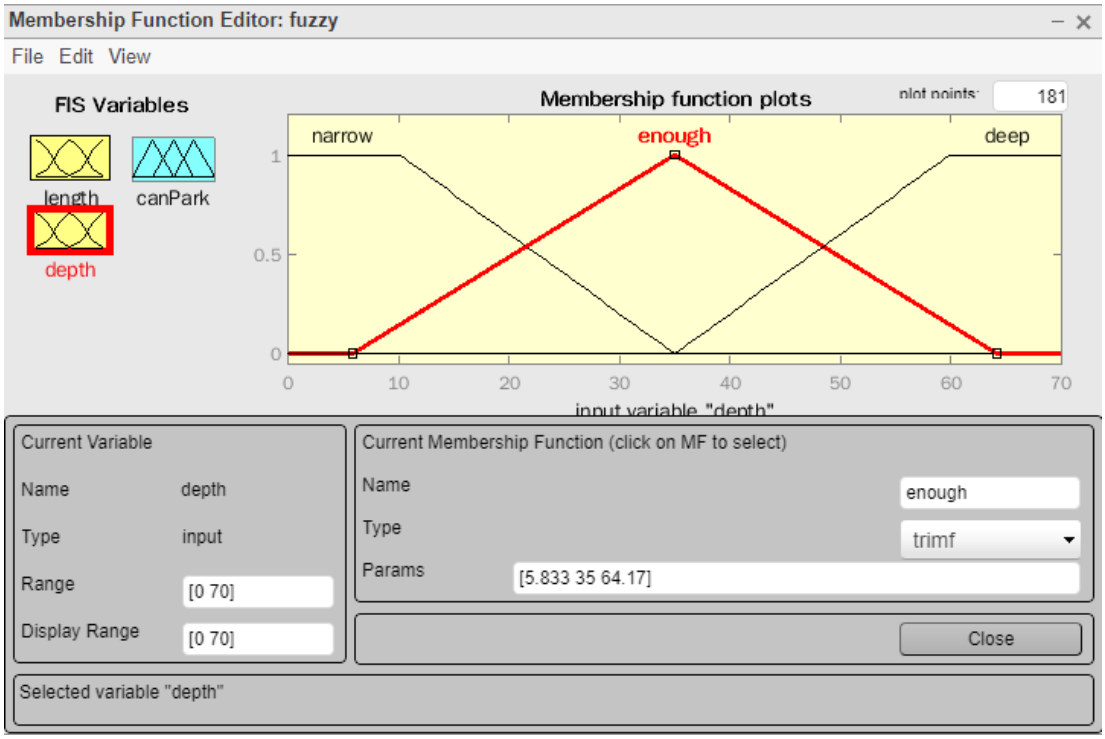


Рисунок 4.6 – Функції належності для параметра глибини паркувального місця

Кожна з лінгвістичних змінних містить у собі по 3 функції належності: недостатньо місця, достатньо, більш ніж достатньо. У випадку вихідної змінної функції означають брак місця для будь-якого виду паркування, придатність для перпендикулярного паркування і придатність для паралельного паркування.

Після цього було реалізовано набір нечітких правил прийняття рішень. Дані правила досить прості та зрозумілі і в більш «людському» вигляді виглядають так:

- якщо довжини недостатньо і глибини недостатньо, то місце не підходить;
- якщо довжини достатньо та глибини достатньо, то місце не підходить;
- якщо довжини достатньо і глибини більш ніж достатньо, то місце підходить для перпендикулярного паркування;
- якщо довжини більш ніж достатньо і глибини достатньо, то місце підходить для паралельного паркування;
- якщо довжини більш ніж достатньо і глибини більш ніж достатньо, то місце підходить для паралельного паркування.

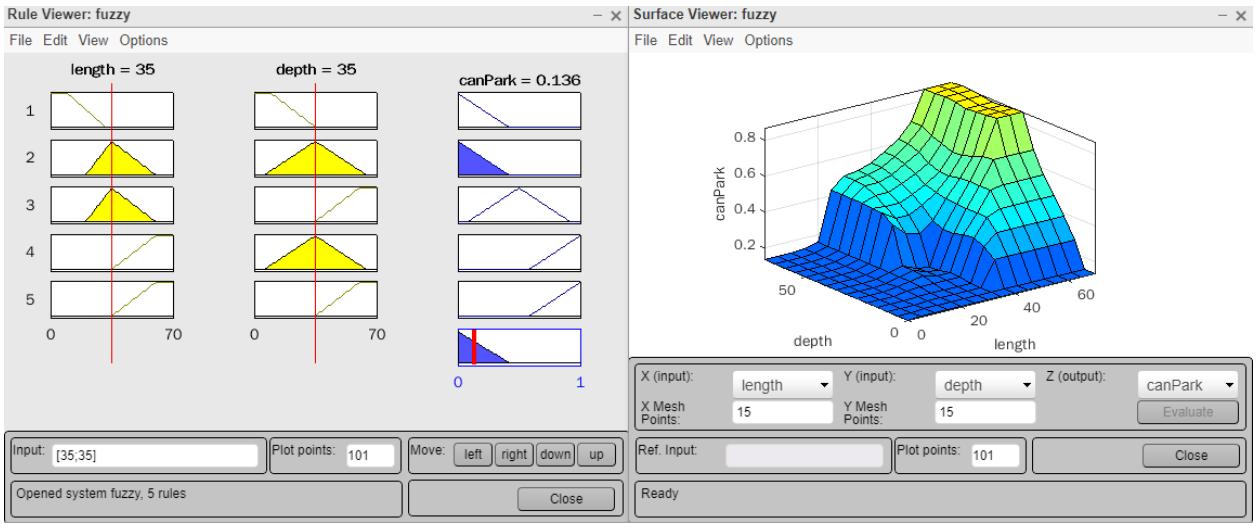


Рисунок 4.7 – Візуалізація нечіткого логічного виводу та поверхня «входи-вихід»

В результаті отримано необхідну базу знань, візуалізацію нечіткого логічного виводу, за допомогою якої можна провести тестування на довільних наборах, та поверхню «входи-вихід», що відповідає синтезованій нечіткій системі.

Для подальшої реалізації в коді використовується проініціалізована раніше система на базі нечіткої логіки. Отримані з датчиків дані про доступне місце передаються в об'єкт fuzzy і відбувається їх фазифікація.

Отримані дані після цього обробляються за встановленим набором правил. Потім, вихід системи дефаззифицируется значення, яке означає прийнятний для доступного простору спосіб паркування.

4.6 Функція круїз-контролю

Як розглядалося раніше, круїз-контроль стає все більш поширеною системою всередині сучасних автомобілів і дозволяє значно полегшити взаємодію під час руху по довгих прямих ділянках доріг, наприклад автомагістралях чи шосе.

Для реалізації такої системи найкраще підійде система на основі нечіткої логіки. Грунтуючись на параметрі поточної швидкості та відстані до об'єкта на шляху руху, система генеруватиме новий параметр швидкості, що підходить для поточної ситуації.

Спочатку, як і у випадку системи вибору методу паркування, потрібно зробити проектування системи всередині пакету MATLAB. Система має два виходи та один вхід: поточну швидкість, відстань до об'єкта попереду та необхідну швидкість відповідно.

Обидві вхідні нечіткі множини містять по три функції належності: низьке, середнє і високе значення. Вихідна множина містить функції належності: зниження швидкості, збереження поточної швидкості,

підвищення швидкості.

База знань містить 9 правил для визначення дії за наявними вхідними значеннями:

- при будь-якій швидкості та низькій відстані, швидкість необхідно знизити;
- якщо швидкість низька та відстань середня, то швидкість зберегти;
- якщо швидкість середня та відстань середня, то швидкість знизити;
- якщо швидкість висока та відстань середня, то швидкість знизити;
- якщо швидкість низька і відстань велика, то швидкість підвищити;
- якщо швидкість середня та відстань велика, то швидкість підвищити;
- якщо швидкість висока та відстань велика, то швидкість зберегти.

В результаті отримуємо візуалізацію нечіткого логічного виводу та поверхню «входи-вихід» функціонування системи.

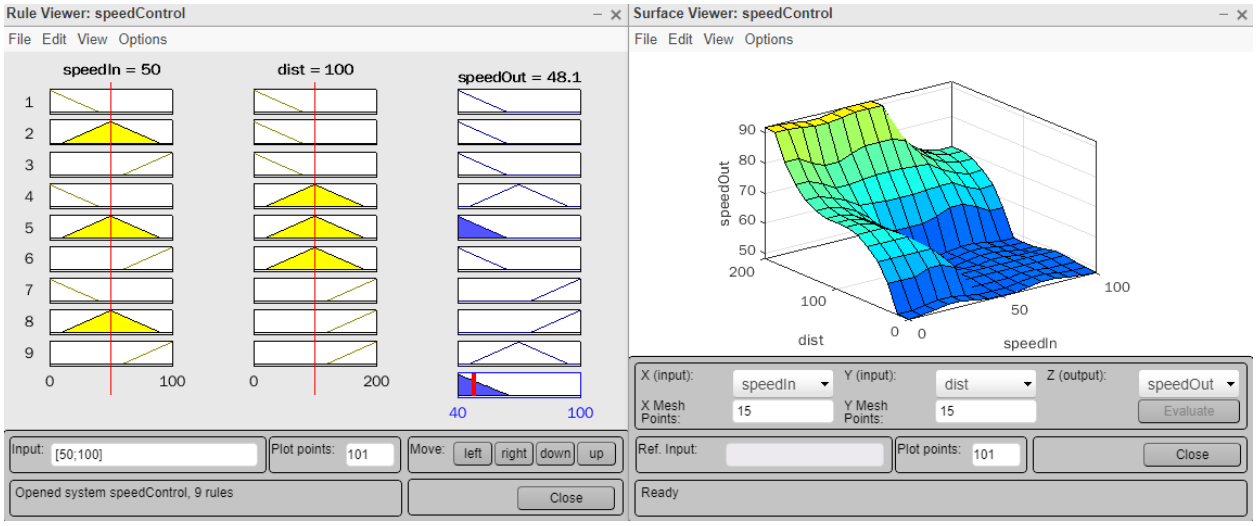


Рисунок 4.8 – Візуалізація нечіткого логічного виводу та поверхня «входи-вихід»

Для імплементації реалізованої системи в кінцевому пристрої проведена ініціалізація об'єкта типу Fuzzy за аналогією з системою вибору способу паркування. До нього були підключені необхідні входи, вихід та правила.

Сама функція круїз-контролю містить збір інформації про поточну швидкість та відстань до об'єкта. Якщо дані задовольняють умовам продовження руху, відбувається їх обробка в реалізованій системі на основі нечіткої логіки. Внаслідок дефазифікації на кожній ітерації перевірки показань системи визначається нове значення швидкості, яке присвоюється двигуну.

Лістинг 4.7 - Функція круїз-контролю

```
void crouiseControl () {
  bool obstacle = false;
  int front = sonarFront.ping_cm();
  int speed = 100;

  int prevFront = 100;

  while(front > 7 || front == 0) {
    front = sonarFront.ping_cm();

    int tempFront = front;

    if (front == 0) {
        tempFront = 200;
    }

    crouiseSpeed->setInput(1, speed);
    crouiseSpeed->setInput(2, tempFront);
    crouiseSpeed->fuzzify();
    speed = int(crouiseSpeed->defuzzify(1));

    if (speed <= 50) {
        stopVeh();
    }
    else {
        digitalWrite(MOTOR_PIN1, HIGH);
        digitalWrite(MOTOR_PIN2, LOW);
```

```
            analogWrite(MOTOR_EN, speed);
        }
        delay(100);
    }
    stopVeh();
}
```

4.7 Визначення світла світлофора

Засіб для перевірки наявності та зчитування сигналу світлофора також є корисним інструментом підвищення безпеки на дорогах. Реалізація такої системи не є тривіальним завданням, адже комп'ютер, на відміну від людини, легко відрізняти кольори один від одного не може.

Для отримання інформації про колір використовується модуль на основі датчика APDS-9960. Для роботи з ним існують готові бібліотеки, наприклад SparkFun_APDS9960.h.

Після підключення бібліотеки необхідно створити змінну для роботи з датчиком та його підключення.

Лістинг 4.8 – Команди для керування модулем APDS-9960

```
SparkFun_APDS9960 apds = SparkFun_APDS9960();
apds.init();
apds.enableLightSensor(false);
apds.readRedLight(red_light);
apds.readGreenLight(green_light);
apds.readBlueLight(blue_light);
```

Датчик повертає значення у колірній моделі RGB, тобто співвідношення червоного, зеленого і синього, що входять до складу зафіксованого датчиком світла.

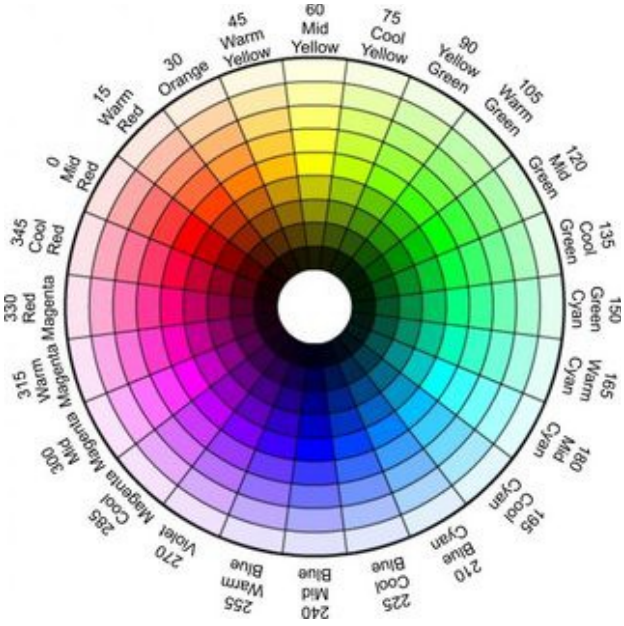


Рисунок 4.9 – Залежність кольору від значення колірного тону колірної моделі HSV

Для більшості завдань така колірна модель буде придатна, але у разі визначення кольору, реалізувати точний поділ на червоний жовтий та зелений буде досить важко. Щоб полегшити подальше завдання визначення кольору, необхідно конвертувати отримані дані в іншу колірну модель з окремим параметром тону кольору, наприклад HSV або HSB.

У таких моделях замість значень в діапазоні від 0 до 255 для трьох ключових кольорів є параметри насиченості (saturation), яскравості (value/brightness) в діапазоні від 0 до 100 і параметр колірного тону (hue) в діапазоні [0;360], що визначає положення кольору на шкалі кольорів.

Листинг 4.9 – Функція зміни колірного діапазону

```
void rgb2hsv(double r, double g, double b){
    double min, max, delta;

    min = r < g ? r : g;
    min = min  < b ? min  : b;
    max = r > g ? r : g;
    max = max  > b ? max  : b;
    v = max/255*100;          // v
    delta = max - min;
    if (delta < 0.00001) {
```

```
        s = 0;
        h = 0;
        return 0;
    }
    if( max > 0.0 ) {
        s = (delta / max)*100;  // s
    } else {
        s = 0.0;
        h = NAN;
        return 0;
    }
    if( r >= max )
        h = ( g - b ) / delta;        // between yellow & magenta
    else
    if( g >= max )
        h = 2.0 + ( b - r ) / delta;  // between cyan & yellow
    else
        h = 4.0 + ( r - g ) / delta;  // between magenta & cyan

    h *= 60.0;                // degrees
    if( h < 0.0 )
        h += 360.0;
    return 0;
}
```

Отримані значення колірного діапазону HSV придатні для подальшої обробки.

У якості системи визначення кольору також використаємо нечітку логіку, а для проектування системи – середовище MATLAB. На вхід системи подаються зібрані значення, переведені в HSV, а у якості виходу – значення, що означає один із трьох можливих сигналів світлофора або його відсутність. Нечітка змінна колірного тону містить 5 функцій приналежності: по одній на кожен колір що визначається, одна для всіх інших і одна додаткова для червоного, так як до нього відносяться значення і в кінці колірного діапазону.

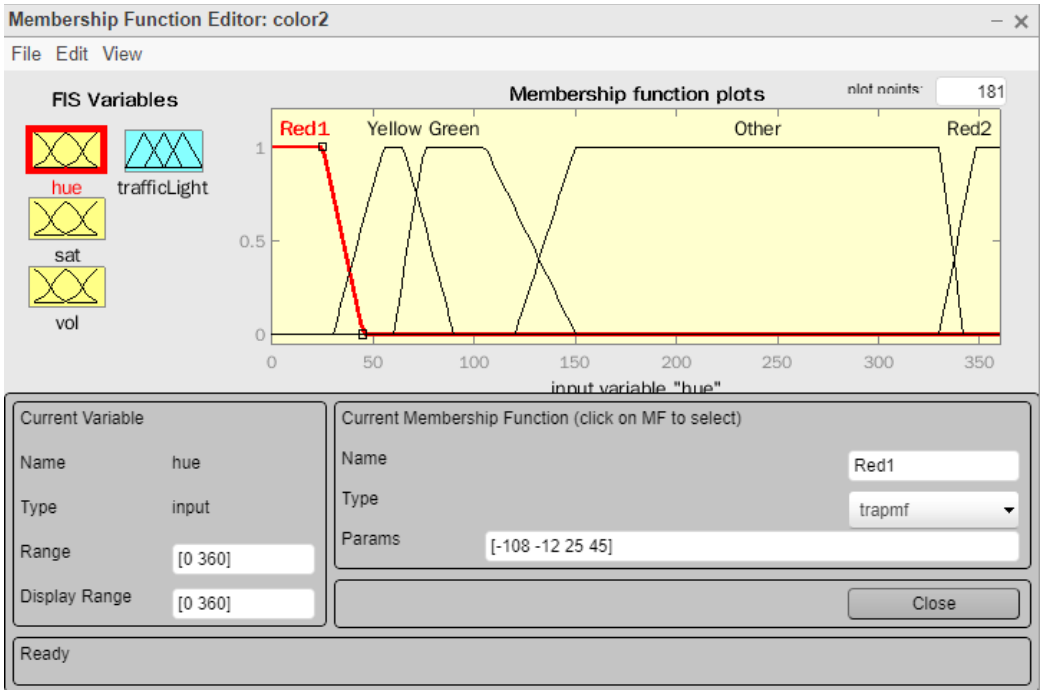


Рисунок 4.10 – Функції належності параметра колірного тону

Дві інші нечіткі змінні містять функції належності для низького, середнього та високого значення відповідних параметрів.

Для подальшого визначення кольорів застосовуються такі правила:

- якщо H червоний та S середній та V середній, то сигнал відсутній;
- якщо H червоний та S високий та V середній, то сигнал червоний;
- якщо H червоний та S середній та V високий, то сигнал червоний;
- якщо H червоний та S високий та V високий, то сигнал червоний;
- якщо H жовтий та S середній та V середній, то сигнал відсутній;
- якщо H жовтий та S середній та V високий, то сигнал червоний;
- якщо H жовтий та S високий та V високий, то сигнал червоний;
- якщо H зелений та S середній та V середній, то сигнал відсутній;
- якщо H зелений та S високий та V середній, то сигнал зелений;
- якщо H зелений та S середній та V високий, то сигнал зелений;
- якщо H зелений та S високий та V високий, то сигнал зелений;
- якщо H інший або S низький іди V низький, сигнал відсутній,

де H – колірний тон, S – насиченість, V – яскравість.

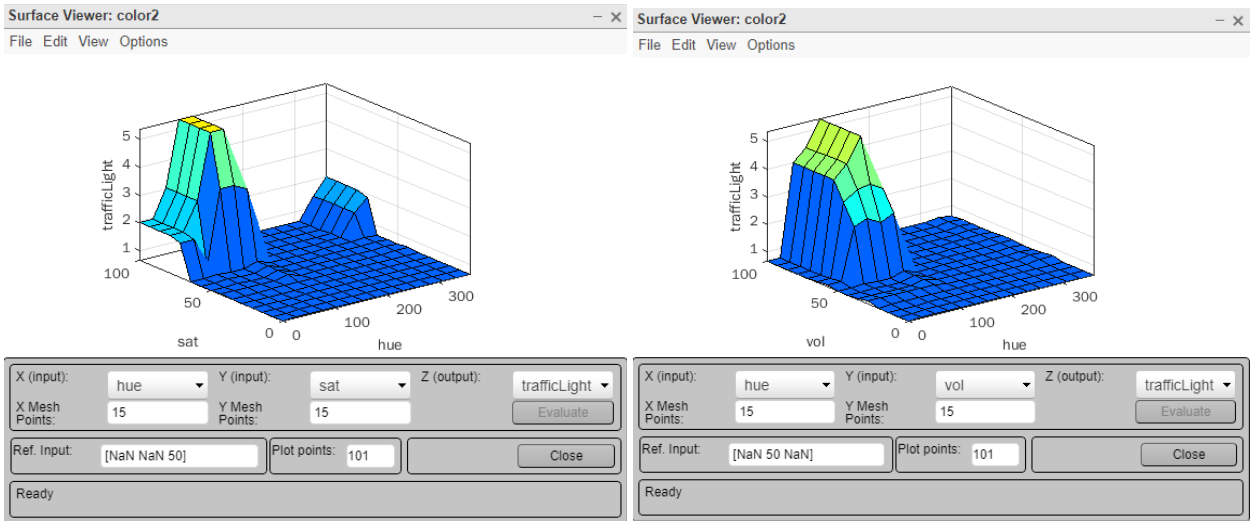


Рисунок 4.11 – Візуалізація поверхонь «входи-вихід» для двох комбінацій вхідних полів

За аналогією з іншими системами на основі нечіткої логіки в даному проекті, для імплементації даного функціоналу на пристрої виконуються ті ж дії: створюється об'єкт-контролер, задаються всі необхідні налаштування, діапазони та правила.

Єдиним обмеженням та відмінністю від двох попередніх контролерів стала зміна у наборі правил. Через наявність трьох входів та набору з 12 правил, кожне з яких також приймає по три значення, можливе виникнення помилок у роботі системи. Воно пов'язане з невеликою кількістю доступної на платі Arduino Uno пам'яті та недостатньою оптимізацією бібліотеки, що використовується.

Для вирішення цієї проблеми було здійснено скорочення кількості команд з бази знань системи нечіткої логіки. Також було вирішено відмовитись від окремого визначення зеленого сигналу. Хоча він і дозволяє системі розпізнати наявність світлофора та перехрестя, сигнал дозволяє рух і для коректної роботи системи автоматичної зупинки на червоний та жовтий сигнал від нього можна відмовитись.

Щоб точність обробки, у цьому разі, збереглася на достатньо високому рівні, правила не прибиралися, а об'єднувалися або спрощувалися за загальними ознаками.

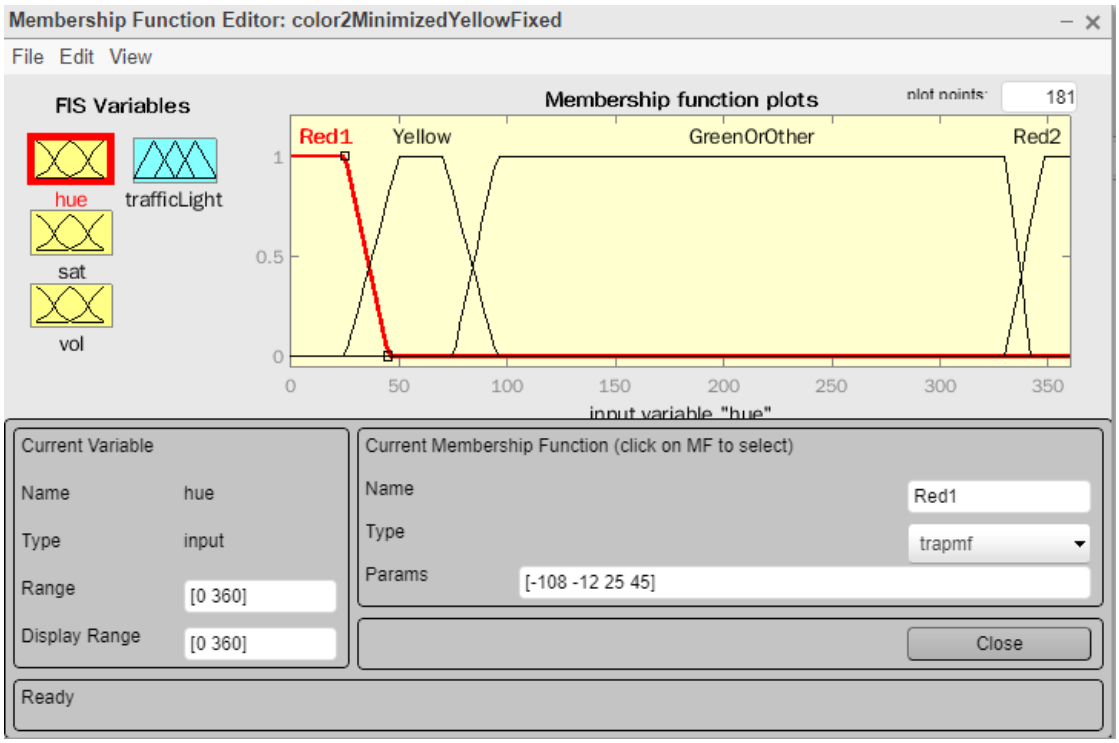


Рисунок 4.12 – Змінена функція належності параметра колірного тону

В результаті отримано набір із п'яти правил, що покриває необхідні ситуації із можливих:

- якщо H червоний та S середній, то сигнал відсутній;
- якщо H червоний та S високий, то сигнал червоний;
- якщо H жовтий та V середній, то сигнал відсутній;
- якщо H жовтий та V високий, то сигнал відсутній;
- якщо H інший або S низький іди V низький, сигнал відсутній.

Лістинг 4.10 – Функція визначення сигналу світлофора

```
void checkTrafficLight()
{
```

```
  apds.readRedLight(red_light);
  apds.readGreenLight(green_light);
  apds.readBlueLight(blue_light);

  rgb2hsv(red_light, green_light, blue_light);

  trafficLight->setInput(1, int(h));
  trafficLight->setInput(2, int(s));
  trafficLight->setInput(3, int(v));
  trafficLight->fuzzify();
  int color = int(trafficLight->defuzzify(1));

  switch (color) {
    //yellow
    case 2:
      stopVeh();
      delay(3000);
      checkTrafficLight();
      break;
    //red
    case 1:
      stopVeh();
      delay(3000);
      checkTrafficLight();
      break;
    //green or none
    case 0:
      break;
  }
}
```

4.8 Керування моделлю

Розміщення органів керування розроблюваної системи на самому пристрої хоч і здається логічним, але викликає велику кількість незручностей при використанні пристрою на етапі проектування та тестування. Кінцевий пристрій повинен мати органи керування всередині автомобіля легко доступні водієві. Але у разі використання тестової платформи їх краще винести на зовнішній пристрій, не пов'язаний з платформою, що рухається.

Саме для цього у схемі пристрою наявний модуль HC-06, що дозволяє здійснювати бездротове підключення по протоколу передачі даних Bluetooth 2.0.

У програмному коді, у разі такої реалізації, вбудований цикл обробки одержуваних з пристрою управління повідомлень. Повідомлення являють собою різні текстові символи. Залежно від символу залежить дія, що

здійснюється тестовою платформою.

Лістинг 4.11 – Цикл обробки повідомлень

```
     if       (Serial.available()       >       0)       {serialA       =
Serial.read();Serial.println(serialA);}

         switch (serialA) {
         // forward
         case 'F':
              movVeh(HIGH, LOW);
              servo1.write(55);
              break;

          // left
          case 'L':
              servo1.write(100);
              break;

          // right
          case 'R':
              servo1.write(15);
              break;

          // forward left
          case 'G':
              movVeh(HIGH, LOW);
              servo1.write(100);
              break;
     ...
     }
```

Для зручнішої взаємодії модель містить набір команд для керування рухом та включення до виконання заданого функціоналу. Наявний набір команд може бути легко розширений за необхідності, наприклад для керування додатковим функціоналом. Потрібно лише додати виконання необхідних дій при надходженні нового варіанту керуючої команди.

На даний момент пристрій містить наступний перелік команд для виконання:

- F – рух уперед;
- L – поворот наліво;
- R – поворот праворуч;
- B – рух назад;

- G – рух уперед і зліва;
- I – рух уперед і праворуч;
- H – рух назад і ліворуч;
- J – рух назад і праворуч;
- S – зупинка;
- P – переведення в режим автоматичного паркування;
- U – автоматичний виїзд з місця паркування;
- C – ввімкнення системи круїз-контролю;
- зміна швидкості руху від 0% до 100% відповідно сигналами 0-9 та Q.

4.9 Пристрій керування

Для керування тестовою платформою у якості пристрою, що містить органи управління функціоналом і засіб передачі повідомлень на модель вирішено використовувати телефон на базі операційної системи Android 6.0.1.

Це дозволяє спростити розробку модуля керування, а також подальше тестування пристрою. Пристрої на базі системи Android мають вбудовані модулі Bluetooth, сумісні з використовуваним на нашій моделі. Крім того, користувач має велику мобільність при використанні компактного телефону або планшета на відміну від громіздких спеціалізованих пультів дистанційного керування або власноруч створених пристроїв.

Операційна система Android на пристрої також дозволяє скористатися великою кількістю безкоштовних сервісів та програмних продуктів як для прямої взаємодії з пристроями Bluetooth, так і для створення власного програмного забезпечення з необхідним функціоналом.

Крім того, за останні роки простежується тенденція на появу додаткових помічників для моніторингу та управління транспортними засобами з мобільних пристроїв. Такі продукти дозволяють віддалено підключатися до автомобіля, заводити його, активувати систему клімат

контролю та використовувати для доступу до авто замість ключа. Також програми дозволяють отримувати сповіщення про події з автомобілем, дублюючи сигналізацію.

Варіант від автовиробника Tesla також містить інформацію про поточний заряд електромобіля та карту доступних зарядних станцій. Також є функція виклику транспортного засобу, при активації якої транспорт починає рухатися у бік водія. Це може бути корисно у разі утрудненого доступу до автомобіля на паркувальному майданчику.

Мобільний додаток, що розробляється в рамках проекту, повинен відповідати наступним умовам:

- мати відкритий програмний код щоб мати можливість вносити зміни при необхідності як під час розробки так і надалі, наприклад для реалізації додаткового функціоналу;
- містити повний набір необхідних органів управління для повного контролю над пристроєм, що розробляється;
- з'єднання з пристроєм має бути простим і зрозумілим, причому повинен бути доступ підключати різні пристрої, а не один;
- у програми повинен бути простий і зрозумілий інтерфейс, не перевантажений зайвими деталями, щоб спростити розробку, так і звільнити кінцевого користувача від необхідності читати інструкцію з використання програми.

Для досягнення поставленого завдання та виконання всіх умов було вирішено скористатися середовищем розробки MIT App Inventor під час створення програми. Цей програмний продукт дозволяє створювати прості програми, використовуючи візуальне програмування.

У такому разі, замість написання коду, структура додатка є набором візуальних блоків, що відповідають за різні функції та обробку дій користувача. За рахунок з'єднання блоків у різні структури виходить реалізувати необхідний функціонал.

Із застосуванням такого середовища розробки значно зростає

швидкість створення програм, оскільки немає необхідності вивчати особливості складних компіляторів та мов програмування, що застосовуються для створення програм під операційну систему Android.

Головною проблемою даного продукту є мала різноманітність доступних до використання готових модулів та відсутність підтримки множинних натискань на екран.

При цьому можливостей середовища MIT App Inventor більш ніж достатньо для поставлених у проекті завдань і його застосування є чудовим рішенням при створенні пристрою керування системою.

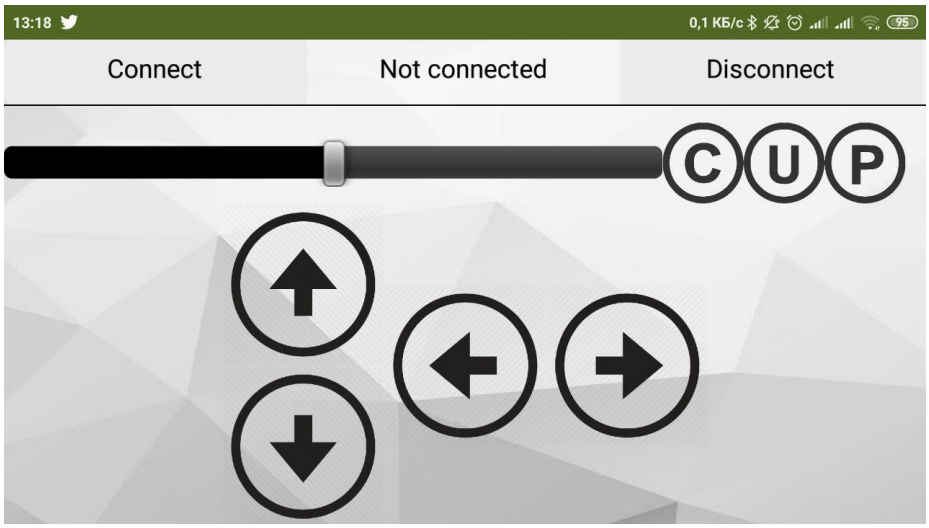


Рисунок 4.12 – Інтерфейс створеного додатку

Для контролю роботи пристрою у додатку передбачені всі необхідні органи управління. У верхній частині розміщений інтерфейс для здійснення підключення до пристрою та відображення його статусу. На основному сегменті вікна програми є два варіанти управління рухом: стрілочки з напрямком, або джойстик.

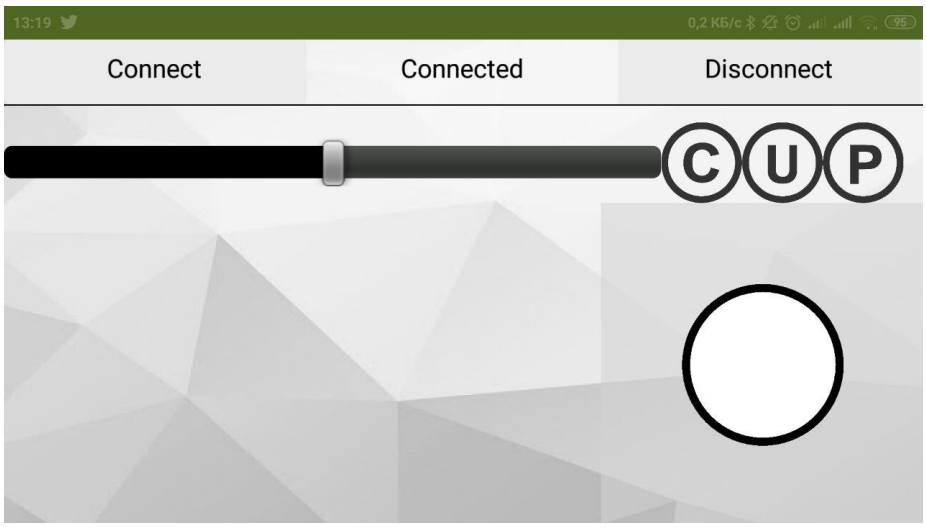


Рисунок 4.13 – Керування джойстиком та відображення підключення додатку до приладу

Наявна шкала установки швидкості руху транспортного засобу та кнопки активації різних функцій: "C" – круїз-контроль, "P" – автоматичне паркування, "U" – автоматичний виїзд з місця для паркування.

Програмна частина проекту являє собою обробкою різних подій, таких як натискань на певні області екрана, і пересилання текстових повідомлень на пристрій.

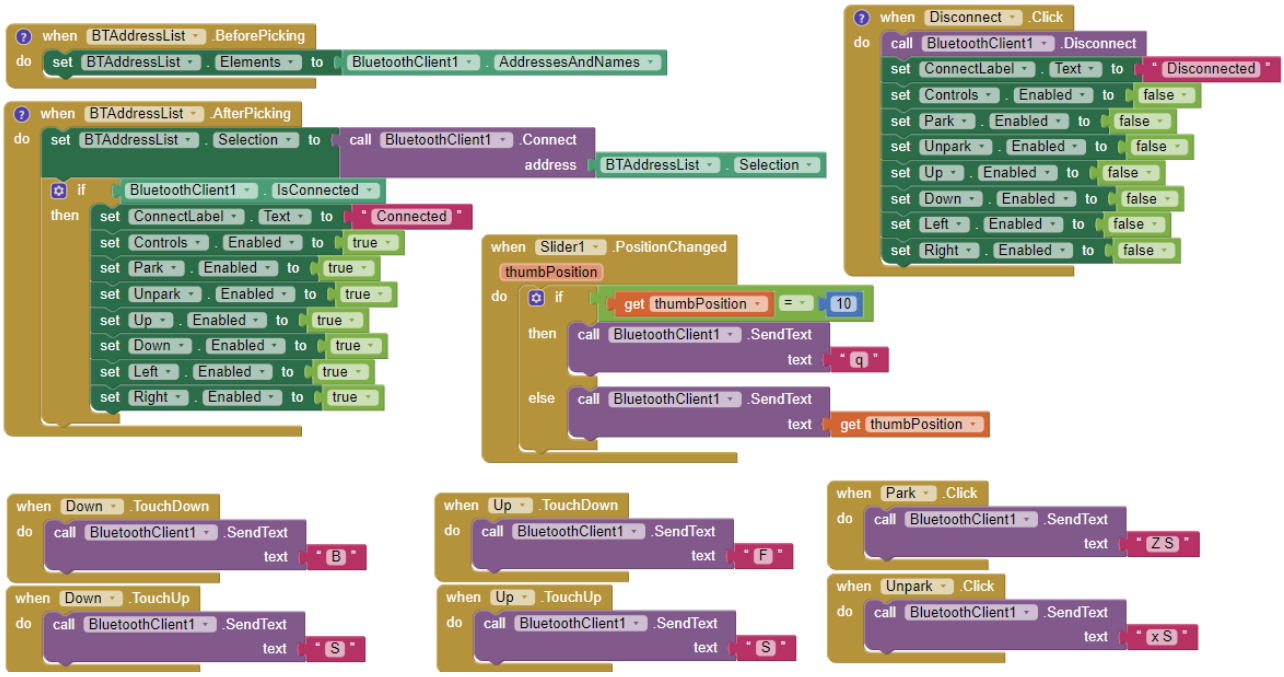


Рисунок 4.14 – Функції підключення та відключення від пристрою; обробка натискань на кнопки керування та зміна позиції повзунка швидкості

Через особливості функціонування порту обміну даними та Bluetooth-модуля, передана команда виконуватиметься циклічно до надходження нової. У зв'язку з цим для коректної роботи органів управління обробляються як натискання на них, так і відпускання кнопок. Під час виконання останніх на пристрій передається команда припинення руху.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРИЛАДУ

5.1 Постанова задач на тестування

Перед застосуванням розробленого пристрою або програмного продукту на практиці, а тим більше випуском його на ринок у вигляді готового, конкурентного продукту, його функціонування має бути ретельно протестоване за різних умов, можливих ситуацій та інших штатних та позаштатних обставин.

Цілісне тестування особливо важливо для пристроїв і систем, що включають великогабаритні елементи, що приводяться в рух, здатні завдати значної шкоди майну і здоров'ю людей у разі відмови.

Саме з цієї причини сучасні автомобілі проходять широкий набір різноманітних тестів, щоб визначити придатність кінцевого продукту до застосування на дорогах загального користування та забезпечення безпеки як водія та пасажирів, так і інших учасників руху.

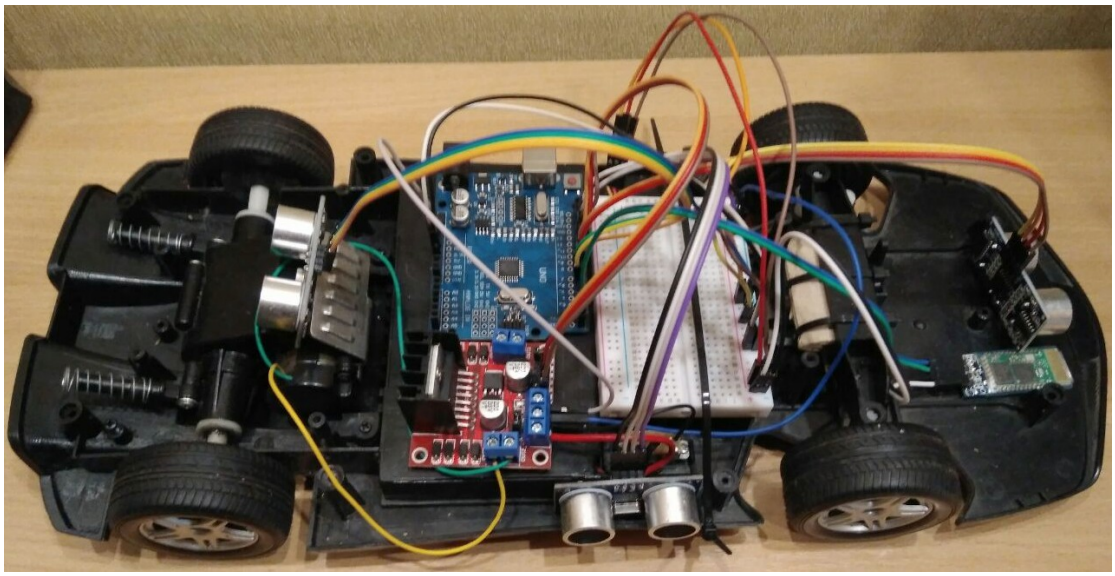


Рисунок 5.1 – Зовнішній вигляд кінцевого приладу

Крім того, відповідальні комісії з безпеки на дорогах з кожним роком вносять зміни до умов та характеристик, яким має відповідати новий транспортний засіб.

Щоб відповідати такому підходу і переконатися не тільки у працездатності, а й у безпечності розробленого пристрою, він має бути протестованим. Перевірці повинен підлягати весь наявний функціонал пристрою:

- правильність підключення пристрою керування до моделі;
- коректний рух моделі за отриманими командами;
- відповідність зазначеної на пристрої керування швидкості руху реальній;
- правильність функціонування загальної системи паркування, пошуку місця, вибору способу, здійснення маневрів;
- коректність автоматичного визначення швидкості руху виходячи з поточної швидкості та наявності перешкоди на шляху руху;
- безвідмовність визначення сигналу світлофора та виконання відповідних йому дій.

5.2 Тестування загальних систем

Для перевірки коректності функціонування підключення, руху та налаштування швидкості достатньо наявності відкритого рівного простору.

Під час тестування модель встановлюється на дану поверхню. З пристрою керування здійснюється спроба підключення до моделі. У разі успіху перевіряється дальність безвідмовного підключення. Вона має становити приблизно 10 м.

Після цього до перевірки піддаються системи руху. При правильній роботі, за натисканням на кнопки керування, модель повинна рухатися у зазначеному напрямку. А регулювання швидкості також повинне впливати на швидкість руху моделі відповідно до зазначених значень.

5.3 Тестування процесу автоматичного паркування

Розглянута модель побудована на базі шасі радіокерованої машини та має розміри 35 на 18 див. Максимальний кут повороту коліс становить 25 градусів. Виходячи з наявних параметрів, були обрані розміри інших об'єктів для застосування в тестуванні – 25 на 35 см. Загальна довжина смуги перешкод при перевірці функцій паркування становить 250 см, ширина – 100 см.

У якості типових дорожніх ситуацій, які необхідно змоделювати, були обрані:

- тестовий полігон порожній та перешкод для паркування не представляється;
- є транспортні засоби спереду та позаду від доступного місця;
- нестача місця для здійснення паркування;
- присутність перешкод на шляху руху під час будь-якого з етапів руху.

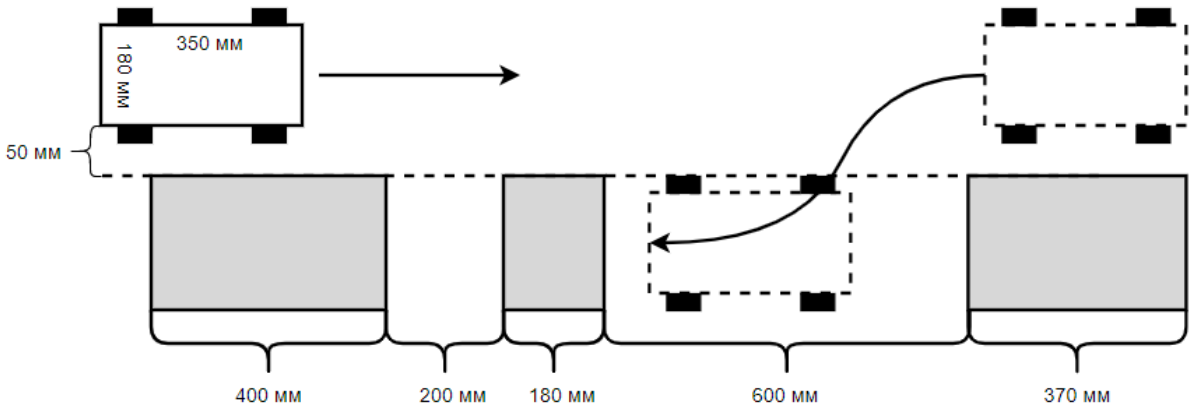


Рисунок 5.2 – Тестування паралельного паркування та системи визначення наявного місця

Відповідно до зазначених ситуацій була складена «смуга перешкод»

для послідовного тестування функціонування в розглянутих ситуаціях.

Модель транспортного засобу встановлюється на початку даної ділянки та запускається система автоматичного паркування. При правильності виконання дій пристрій здійснює рух по прямій і сканує доступний простір. У разі задоволення умов, що визначається системою на базі нечіткої логіки, проводиться парковка у придатному місці.

Під час руху та паркувальних маневрів, на шляху руху моделі, встановлюється додатковий об'єкт-перешкода для перевірки системи забезпечення безпеки руху.

Для перевірки функціонування перпендикулярного паркування застосовується подібний за своїми параметрами набір об'єктів. Відрізняється лише доступне місце для паркування. Всі інші етапи, перевірки та вимоги залишаються незмінними.

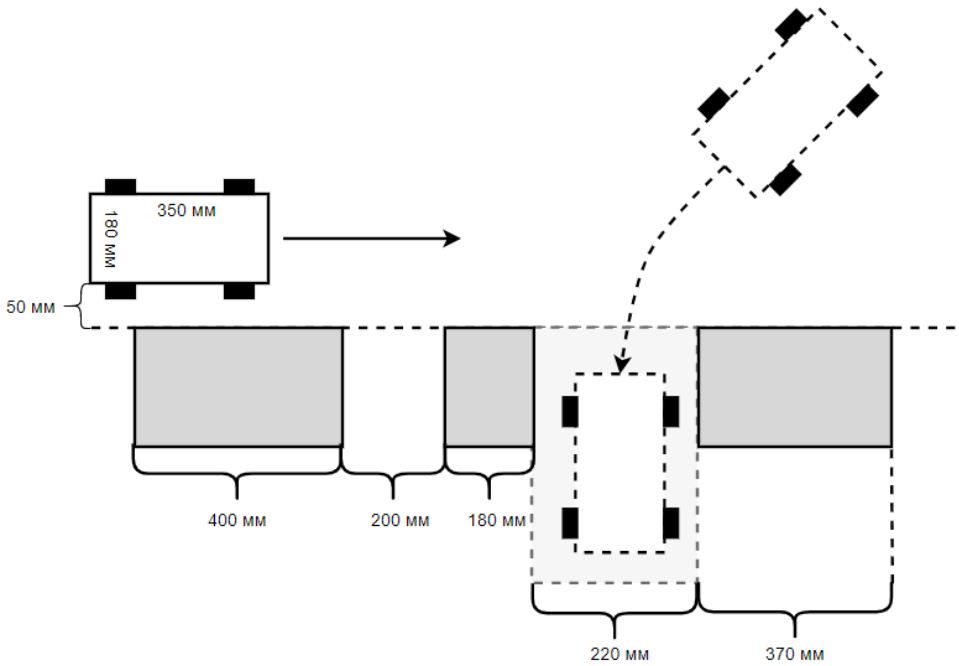


Рисунок 5.3 – Тестування перпендикулярного паркування та системи визначення наявного місця

У якості додаткового тестування всіх систем, пов'язаних з паркуванням, була створена узагальнююча смуга перешкод.

Вона передбачає почергову перевірку функціонування всіх дій системи, включаючи пошук місця, перевірку його придатності для різних методів паркування, здійснення маневрів для встановлення та залишення місця стоянки двох розглянутих у проекті видів, а також реакції системи на перешкоди, що виникають на шляху руху.

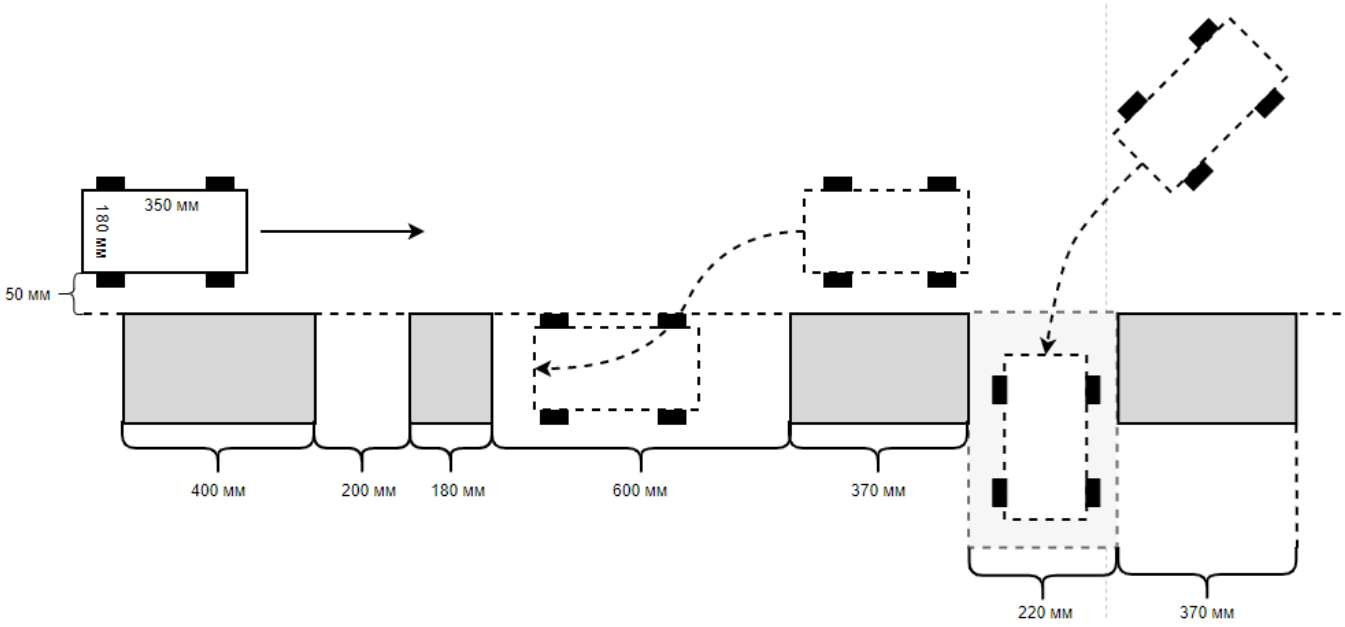


Рисунок 5.4 – Створена смуга перешкод для тестування роботи всіх аспектів роботи системи паркування

5.4 Перевірка системи круїз-контролю

Для перевірки системи автоматичного регулювання швидкості руху моделі необхідна досить довга ділянка рівної та відкритої місцевості. Транспортний засіб у такому разі рухається прямо, а розміщений перед ним об'єкт переміщається вручну з різною швидкістю.

При правильності функціонування системи на основі нечіткої логіки для визначення придатної для руху швидкості, модель повинна в автоматичному режимі знижувати або підвищувати швидкість руху в залежності від відстані до об'єкта попереду і поточної швидкості. А у разі

зупинки об'єкта також зупинитися перед ним.

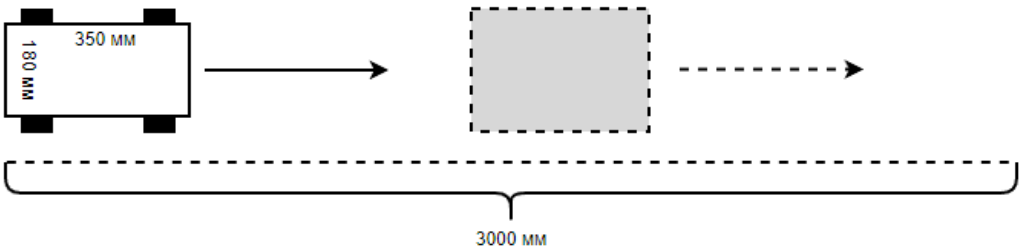


Рисунок 5.5 – Тестування системи круїз-контролю

5.5 Перевірка системи визначення сигналу світлофора

Ця система передбачає наявність світлофора або іншого пристрою для відображення колірних та світлових сигналів під час руху.

У якості такого пристрою використовується дисплей, на якому виводитимуться відповідні до різних сигналів світлофора кольорові зображення.

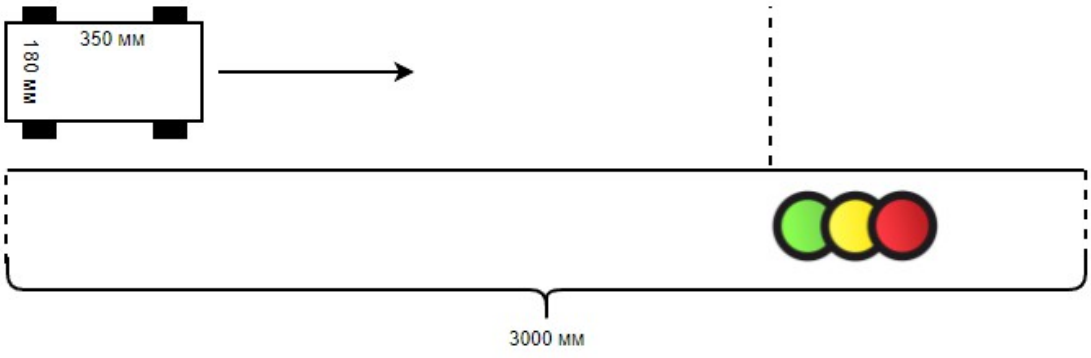


Рисунок 5.6 – Тестування системи зчитування сигналу світлофору

Транспортний засіб повинен здійснювати рух у напрямку даного дисплея та при наближенні зчитувати поточний сигнал. При правильності

функціонування системи модель повинна зупинитись у разі наявності червоного або жовтого сигналу та очікувати появи зеленого. В іншому випадку транспортний засіб повинен продовжити подальший рух.

Тестовий майданчик для даного випробування ідентичний використаній при перевірці системи круїз-контролю. Замість об'єкта-перешкоди, що рухається, на ньому розміщений дисплей-світлофор.

5.6 Результати тестування

В результаті проведених тестів пристрій коректно продемонстрував весь наявний функціонал. Усі перешкоди, що виникають на шляху, були коректно розпізнані системою і здійснені відповідні дії.

Система автоматичного паркування також коректно пройшла смугу перешкод та виконала всі дії точно і без непередбачуваних подій.

Система круїз-контролю також показала свою працездатність та коректність регулювання швидкості руху.

Система визначення сигналів світлофора також безпомилково визначила сигнали, що подаються світлофором, при середньому денному освітленні і напівтемряві. У разі високої яскравості система неправильно розпізнала жовтий колір, прийнявши його за зелений.

Ця помилка пов'язана з особливістю розпізнавання кольору та заданими діапазонами функцій належності нечіткої змінної колірного тону. Після внесення виправлень та підвищення допустимих значень жовтого кольору система почала працювати коректно.

ВИСНОВКИ

В даній кваліфікаційній роботі було розглянуто використання нечіткої логіки для створення моделі керування автопілотом для наземного транспорту. У якості кінцевих пристроїв, для яких реалізовувалось рішення стали автомобілі із механізмом кермового керування передніх коліс.

Для побудови системи і подальшого тестування була розроблена платформа, що відповідає наявним у проекті вимогам та фізичній моделі повнорозмірних наземних транспортних засобів.

Були розглянуті сучасні рішення у сфері автоматизації руху, їх тенденції розвитку та актуальність. На основі розглянутого, був запропонований функціонал, необхідний для реалізації компетентного та конкурентного рішення.

Також були розглянуті математичні моделі, що використовуються для створення систем автоматичного прийняття рішень та контролю над системою. За результатами їх аналізу було обрано системи із використанням нечіткої логіки для імплементації у розроблюваному рішенні.

Були розглянуті компоненти, необхідні для реалізації прототипу. Так, було обране шасі від радіокерованої машинки, обладнання для імітації руху транспортних реальних засобів. У якості пристрою керування було використано мікроконтролер ATmega328P у складі плати Arduino UNO. Для збору інформації про зовнішню ситуацію обрані ультразвукові датчики HC-SR04 та багатофункціональний датчик APDS-9960. Виходячи з наявних компонентів була запропонована схема кінцевого приладу та створений прототип із їх використанням.

У середовищі Fuzzy logic toolbox, що входить до складу програмного пакету MATLAB були спроектовані системи прийняття рішень для автоматизації задач паркування, контролю швидкості, та аналізу сигналів світлофорів. Під час проектування були розглянуті основні питання по

реалізації таких систем, нормалізації вхідних даних, внутрішнім правилам для прийняття рішень.

Спроектовані системи були відтворені у середовищі Arduino IDE для подальшого запису на плату. Реалізовані системи були протестовані у різноманітних можливих ситуаціях для підтвердження коректності функціонування пристрою. За результатами тестування були внесені невеликі зміни у налаштування внутрішніх систем для підвищення точності аналізу даних.

У разі подальшого розвитку проект може буди масштабований для інтеграції до систем повнорозмірного транспортного засобу та тестування працездатності у реальних умовах. Також може бути реалізований додатковий функціонал для автоматизації більшої кількості процесів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Little B. Automation of planes began 9 years after the wright bros took flight–but it still leads to baffling disasters. HISTORY. URL: https://www.history.com/news/plane-automation-autopilot-flight-302-610

2. Ingraham C. The safest – and deadliest – ways to travel. Washington Post. URL: https://www.washingtonpost.com/news/wonk/wp/2015/05/14/the-safest-and-deadliest-ways-to-travel/.

3. Savage I. Comparing the fatality risks in United States transportation across modes and over time. Research in transportation economics elsevier. 2013. Vol. 1, no. 43. P. 9–22. URL: https://faculty.wcas.northwestern.edu/~ipsavage/436.pdf.

4. Anderson B. Audi A8 lifts itself up in case of side impact to protect you | carscoops. Carscoops. URL: https://www.carscoops.com/2019/04/audi-a8s-pre-sense-side-system-shows-how-far-car-safety-has-come/.

5. Fatality facts 2019: yearly snapshot. IIHS-HLDI crash testing and highway safety. URL: https://www.iihs.org/topics/fatality-statistics/detail/yearly-snapshot.

6. SAE levels of driving automation™ refined for clarity and international audience. SAE International. URL: https://www.sae.org/blog/sae-j3016-update.

7. Jain S. Watch: tesla autopilot feature mistakes moon for yellow traffic light. NDTV.com. URL: https://www.ndtv.com/offbeat/watch-tesla-autopilot-feature-mistakes-moon-for-yellow-traffic-light-2495804.

8. Watch tesla autopilot get bamboozled by a truck hauling traffic lights. Futurism. URL: https://futurism.com/the-byte/tesla-autopilot-bamboozled-truck-traffic-lights.

9. Why hundreds are killed in crashes in parking lots and garages every year. CBS News - Breaking news, 24/7 live streaming news & top stories. URL: https://www.cbsnews.com/news/parking-lot-accidents-distracted-drivers-national-

safety-council/.

10. Valavanidis A. The shift to diesel fuel engines and how the emission scandal of diesel vehicles unfolded. world energy consumption of transportation sector. Scientific reviews. 2018. URL: https://www.researchgate.net/publication/322926517_The_Shift_to_Diesel_Fuel_Engines_and_How_the_Emission_Scandal_of_Diesel_Vehicles_Unfolded_World_Energy_Consumption_of_Transportation_Sector.

11. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений. Москва : Мир, 1976. 165 с.

12. Fuzzy logic toolbox. MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink. URL: https://www.mathworks.com/help/fuzzy/.