Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Система управління освітленням з використанням IoT

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІз-21-1
Валерія БОГДАНОВА
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ст. викл. Артем ГУК
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Богдановій Валерії Віталіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Система управління освітленням з використанням IoT

затверджена наказом по університету від " 05 " травня 2025 р. № 72 СТз

2. Термін подання здобувачем роботи до екзаменаційної комісії 16 червня 2025 р.

3. Вхідні дані до роботи
вуличне освітлення
пішохідний перехід
UI flow діаграма

4. Перелік питань, що потрібно опрацювати у роботі
Аналіз предметної області та постановка завдання
Технологічна платформа реалізації
Програмна реалізація пристрою керування

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 10 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання та аналіз літератури	05.05.2025–10.05.2025	
2	Огляд існуючих аналогів	11.05.2025–03.06.2025	
3	Вибір алгоритмів	04.06.2025–06.06.2025	
4	Вибір програмних та апаратних засобів	07.06.2025–08.06.2025	
5	Програмна реалізація	09.06.2025–11.06.2025	
6	Аналіз отриманих результатів	12.06.2025–13.06.2025	
7	Оформлення записки	14.06.2025–16.06.2025	

Дата видачі завдання " 05 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Артем ГУК
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 54 с., 15 рис., 1 дод., 9 джерел.

ІОТ, КОМП'ЮТЕРНА СИСТЕМА, RASPBERRY PI, ОСВІТЛЕННЯ, КЕРУВАННЯ, ПІШОХІДНИЙ ПЕРЕХІД.

Метою кваліфікаційної роботи є проєктування та реалізація комп'ютеризованої інформаційної системи для ефективного управління вуличним освітленням.

У ході виконання кваліфікаційної роботи здійснено ґрунтовний огляд сучасних технічних і програмних рішень у сфері автоматизації систем освітлення. На основі проведеного аналізу було визначено оптимальні інструменти для розробки, а також сформовано архітектуру системи, яка забезпечує масштабованість, стійкість до збоїв та високий рівень інформаційної безпеки з урахуванням криптографічного захисту при взаємодії з численними користувачами.

ABSTRACT

Bachelor’s thesis: 54 pages, 15 figures, 1 appendices, 9 sources.

IoT, COMPUTER SYSTEM, RASPBERRY PI, LIGHTING, CONTROL, PEDESTRIAN CROSSING.

The major goal of this thesis is the design and implementation of a computerized information system for efficient street lighting control.

In order to a comprehensive review of existing hardware and software solutions in the field of lighting automation was conducted. Based on this analysis, the most appropriate development tools were selected, and a robust system architecture was developed to ensure scalability, fault tolerance, and a high level of information security, including cryptographic protection during interactions with multiple users.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ДТП – дорожньо-транспортна пригода

МК – мікроконтролер

ОП – освітлювальні прилади

ПВТ – пішохідне викличне табло

САПР – система автоматизованого проектування

РЕП – радіоелектронні пристрої

ТПВ – таблиця переходів-виходів

ЦП – цифровий пристрій

ARM – Advanced RISC Machine, (вдосконалена RISC-машина)

ASIC – application-specific integrated circuit, (інтегральна схема спеціального призначення)

GUI – graphical user interface,(графічний інтерфейс користувача)

IDE – Integrated Development Environment (інтегроване середовище розробки)

IoT – Internet of things, Інтернет речей

RISC – reduced instruction set computer (комп’ютер (процесор) зі скороченим набором команд)

ВСТУП

Розвиток технологій Інтернету речей відкрив нові горизонти для модернізації систем освітлення, зокрема через концепцію інтелектуального управління, орієнтованого на потреби людини. Зокрема, урбанізація і формування концепцій «розумного» середовища, таких як smart city, smart home і автоматизовані будівлі, істотно трансформували підхід до розробки освітлювальних систем. Інтелектуальні системи освітлення дедалі більше орієнтуються на використання бездротового зв'язку, що забезпечує широкі функціональні можливості. З огляду на це, виникають нові технічні виклики, пов’язані з інтеграцією сенсорів, забезпеченням сумісності обладнання, розробкою адаптивного програмного забезпечення, що відповідає вимогам Інтернету речей.

Особливе значення у цьому контексті набуває зовнішнє освітлення міського середовища, яке не лише забезпечує комфорт і безпеку в темний період доби, а й є ключовим елементом інженерної інфраструктури. Рівень нормативного регулювання цього питання підкреслює його важливість: у Державних будівельних нормах України виділено окремий розділ, присвячений нормам зовнішнього освітлення. Така система виконує критичну функцію – гарантує повноцінне функціонування міської інфраструктури за умов обмеженої видимості, одночасно створюючи передумови для зменшення дорожньо-транспортних пригод.

Практика показує, що належне зовнішнє освітлення здатне знизити загальний рівень аварійності на дорогах, причому на окремих ділянках, зокрема в зонах підвищеної небезпеки, таких як перехрестя, ця ефективність є ще вищою. Підвищення яскравості дорожнього покриття суттєво покращує візуальне сприйняття простору і, відповідно, підвищує безпеку пересування. Крім безпосереднього впливу на зниження рівня аварійності, система зовнішнього освітлення чинить позитивний вплив і на інші сфери:

оптимізується енергоспоживання, зменшуються витрати на експлуатацію, покращується екологічна ситуація завдяки зниженню теплового навантаження на довкілля, зменшенню викидів і світлового забруднення, а також активізується ділове, туристичне та інвестиційне середовище міста.

Однак незважаючи на ці переваги, гострою проблемою залишається недостатня видимість пішоходів у темний час доби, що є одним із ключових факторів смертності серед учасників дорожнього руху. Значна частка наїздів відбувається на нерегульованих пішохідних переходах, де відсутність належного освітлення у поєднанні з несприятливими погодними умовами, низькою якістю розмітки, перешкодами огляду та технічним станом автомобілів істотно ускладнюють виявлення пішоходів. Згідно зі статистикою, кількість смертельних випадків серед пішоходів у темний час доби суттєво перевищує показники денної аварійності

В умовах таких викликів усе частіше застосовуються інтелектуальні рішення на основі мікроконтролерних систем, здатні виконувати адаптивне керування освітленням з урахуванням аналізу навколишніх умов. Мікроконтролери, що є серцевиною таких систем, широко використовуються в різних галузях і постійно вдосконалюються. У зв’язку з динамічним розвитком електронних технологій, ці пристрої отримали нові можливості, зокрема в обробці аналогових сигналів у реальному часі. Одним із сучасних і перспективних напрямків є застосування мікроконтролерів сімейства ARM, зокрема STM32, які поєднують високу продуктивність, енергоефективність і здатність до реалізації складних алгоритмів керування. Компанія STMicroelectronics, як один із лідерів у галузі, запропонувала потужну лінійку мікроконтролерів STM32, побудованих на ядрі ARM Cortex-M3, які стали новим стандартом в області вбудованих систем керування.

Метою роботи є проєктування та реалізація комп’ютеризованої інформаційної системи для ефективного управління вуличним освітленням.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Периферійне обладнання для освітлення пішохідних переходів

Зовнішнє освітлення виконує ключову роль у забезпеченні функціональності міської інженерно-транспортної інфраструктури. Раціонально спроєктоване вуличне освітлення підвищує рівень безпеки та комфорту мешканців у темний час доби. Це твердження підтверджується фактом наявності окремого розділу, присвяченого нормуванню зовнішнього освітлення, у чинних Державних будівельних нормах України, що підкреслює його важливість на національному рівні. Пішохідні переходи в межах міських вулиць, навіть у випадках, коли вони оснащені світлофорами, залишаються зонами підвищеного ризику, особливо за умов зниженої видимості, що характерна для сутінків або нічного часу. Установлення спеціального освітлення в межах таких переходів здатне істотно знизити ймовірність виникнення аварійних ситуацій.

Проєктуючи освітлення для пішохідних переходів, слід враховувати особливості зорового сприйняття об'єктів у темряві. Ефективним є використання світла, що відрізняється за колірною температурою від загального освітлення вулиць, оскільки такий контраст посилює його сигнальну функцію та покращує сприйняття об'єкта. Це дозволяє виділити пішохідну зону на тлі навколишньої дорожньої інфраструктури. При цьому оптимального результату досягають шляхом створення позитивного контрасту, за якого пішохід сприймається як світлий об’єкт на темному фоні, що сприяє своєчасному виявленню його водієм. Таке візуальне виділення забезпечується розташуванням джерела світла між автомобілем та пішоходом і направленням світлового потоку в напрямку руху транспортного засобу. Висота розміщення світильників визначається типом використовуваного обладнання і варіюється в межах половини до повної

висоти стандартної опори.

Актуальність даної проблематики знаходить відображення у чинному законодавстві та нормативних документах України, де розгляд питань організації зовнішнього освітлення включено до складу заходів із підвищення безпеки дорожнього руху, а також у рамках благоустрою міських територій. Системи освітлення, що застосовуються на пішохідних переходах, мають забезпечити достатній рівень видимості, що дозволяє пішоходам безпечно перетинати проїжджу частину та розпізнавати можливі перешкоди або дефекти дорожнього покриття. Досягається це шляхом створення зони з яскравістю, що перевищує рівень освітлення прилеглої частини дороги щонайменше у півтора раза. Такий ефект забезпечують зменшенням міжопорної відстані, установленням додаткових джерел світла або використанням приладів підвищеної потужності.

Для забезпечення ефективного світлорозподілу необхідно враховувати орієнтацію світильників та їхнє розміщення відносно зони переходу. Джерела світла повинні підкреслювати силует пішохода на тлі проїжджої частини, водночас унеможливлюючи ефект засліплення для водіїв. Рекомендованим є монтаж освітлювальних приладів перед пішохідним переходом у напрямку руху транспорту, а для двостороннього руху — симетричне розміщення по обидва боки проїзної частини. Використання асиметричних приладів освітлення дозволяє додатково зменшити ризик засліплення.

Досягнення високого рівня безпеки на пішохідних переходах у темний час доби передбачає дотримання низки ключових принципів. Йдеться про забезпечення видимості, еквівалентної денному освітленню, створення позитивного контрасту між пішоходом і фоном, відсутність засліплення водіїв, а також необхідність завчасного візуального сприйняття пішохідного переходу. Регулювання пішохідного руху здійснюється відповідно до положень чинних правил дорожнього руху, згідно з якими пішоходи зобов’язані орієнтуватися на сигнали світлофора чи регулювальника. Спеціалізовані пішохідні світлофори, що використовуються для цього, мають

дві кольорові секції із символічним зображенням людини та можуть доповнюватися звуковою індикацією з варіативними інтервалами сигналу.

Сучасна система автоматизованої індикації пішохідного переходу передбачає застосування інтегрованих технічних рішень для підвищення помітності переходу в темний час доби. Така система зазвичай включає підсвічувані дорожні знаки, сенсори присутності пішохода, блоки синхронізації індикації та освітлення, спрямовані світлодіодні ліхтарі й при необхідності автономне живлення на базі сонячних панелей. Активація відбувається або автоматично при наближенні пішохода, або вручну за допомогою кнопки виклику. У результаті запускається підсвічування знаків і вуличного освітлення над пішохідною зоною, яке зберігається протягом часу, необхідного для безпечного перетину проїжджої частини, після чого система автоматично деактивується.

Рисунок 1.1 – Приклад конструкцій освітлення пішохідних переходів

У межах підвищення ефективності організації дорожнього руху та забезпечення безпеки пішоходів поряд із пішохідними світлофорами передбачається можливість використання пішохідного викличного табло. Цей інтелектуальний пристрій спрямований на ініціювання запиту до

системи керування світлофорними об’єктами з метою активації дозволяючого сигналу для переходу через проїжджу частину. Основна перевага полягає в тому, що зелений сигнал вмикається лише в разі наявного запиту з боку пішохода, що сприяє раціоналізації транспортних потоків без погіршення умов для пішохідного руху.

Пішохідне викличне табло забезпечує формування керуючого сигналу, що надсилається до дорожнього контролера, ініціюючи активацію дозвільного світлофорного сигналу. Виклик відбувається в результаті натискання на кнопку, після чого система фіксує подію та подає запит на зміну фази. У пристрої передбачено візуальну індикацію, яка сповіщає пішохода про прийом запиту системою керування. Світловий індикатор активується у момент надсилання сигналу та залишається увімкненим до моменту появи зеленого сигналу на світлофорі, що підтверджує дозволений прохід.

Для забезпечення інклюзивності системи передбачено також підключення аудіопідтримки, що має на меті підвищення безпеки пересування людей із порушенням зору. Аудіосигнал синхронізується із фазами світлофора та активується під час увімкнення дозволяючого сигналу. Звуковий супровід у вигляді імпульсів або трелей повідомляє про можливість безпечного перетину, а з наближенням завершення дозволеної фази частота сигналу зростає, що виконує функцію попередження. Такий підхід значно зменшує ризик виникнення аварійних ситуацій у межах пішохідних переходів, підвищуючи загальний рівень дорожньої безпеки.

1.2 Автоматні пристрої керування

Згідно з класифікацією програмних систем, що охоплює переробку інформації та функціонування в умовах управління, усі програмні комплекси поділяються на трансформуючі, інтерактивні та реактивні. Перший тип передбачає одноразове перетворення вхідної інформації із завершенням

роботи після видачі результату. У таких випадках всі вхідні дані зазвичай наявні до початку обробки, а результат формується наприкінці виконання програми. Інтерактивні системи, навпаки, реалізують постійну взаємодію із зовнішнім середовищем, маючи змогу контролювати темп обміну інформацією та забезпечувати діалоговий режим. Реактивні системи функціонують в умовах зовнішньої ініціативи, тобто їх поведінка визначається не лише подіями, а й їхнім темпом, заданим зовнішнім середовищем.

Одним із ключових напрямів розвитку реактивних систем є побудова логіки управління, що реалізується за допомогою автоматів. Такі системи здатні обробляти сигнали в реальному часі, реагуючи на множину вхідних впливів шляхом формування відповідних вихідних реакцій. Особливістю подібних рішень є залежність вихідної дії не лише від поточного сигналу, але й від попередньої історії функціонування, яка акумулюється у стані системи. У випадку, коли реакція визначається виключно поточним входом, поведінку системи можна вважати простою; натомість залежність від стану вказує на складну поведінку, властиву динамічним системам із пам’яттю.

Поняття стану використовується для відображення сукупності змінних, які характеризують внутрішній зміст системи в конкретний момент часу. Саме стан забезпечує узагальнену інтерпретацію попередніх дій і сигналів, дозволяючи сформулювати поточну реакцію як функцію від входу та стану. В цьому сенсі стан виступає як своєрідна пам’ять, яка замінює необхідність уявного оперування часовою змінною, зберігаючи лише сутнісну інформацію про минуле.

У процесі розробки систем автоматичного управління прийнято розділяти структуру об’єкта на керуючу та керовану частини. Перша забезпечує логіку дій на основі аналізу поточного стану та вхідної інформації, тоді як друга реалізує фізичне виконання вибраних дій та, за необхідності, формує зворотні сигнали. У межах класичної теорії систем автоматичного керування ці компоненти визначаються як система керування

та об’єкт керування відповідно. У випадках, коли система керування реалізується у вигляді автомата, її ще називають керуючим автоматом, а сукупність керованого об’єкта та автомату – автоматизованим об’єктом керування.

Під час моделювання подібних систем час може розглядатися як дискретний або безперервний. Якщо система функціонує у дискретному режимі, то час подається як послідовність тактів, впродовж яких система переходить із одного стану в інший. У таких умовах модель системи описується у вигляді кінцевого автомата, в основі якого лежить математична структура з обмеженою кількістю вхідних, станів і вихідних значень. Абстрактна модель автомата передбачає визначення множин вхідних символів, станів та виходів, а також відповідних функцій переходів та виходів, які формалізують зміну стану й реакцію системи.

З огляду на характер логічного управління, особливого значення набуває точна розмірність множин входів, виходів та станів, оскільки це безпосередньо впливає на складність технічної реалізації. Кожен з вхідних або вихідних сигналів розглядається як змінна відповідного типу: вхідна, вихідна або внутрішня. Саме ці змінні відображають реальний фізичний або логічний процес у системі.

У відповідності до викладеного, структура автоматизованого об’єкта керування містить елементи, що забезпечують як керування, так і збирання інформації про стан середовища. До її складу входять сигнали керування, індикаційні сигнали та зовнішні події, які в сукупності формують повноцінне функціонування всієї системи. Графічне представлення такої структури наведено на рисунку 1.2.

У системах автоматизованого управління важливим напрямом є використання удосконалених моделей, що розширюють класичні уявлення про кінцеві автомати. Однією з таких концепцій є гіперавтомати, які спираються на уявлення про поліморфні процеси, визначені множинами функцій-станів.

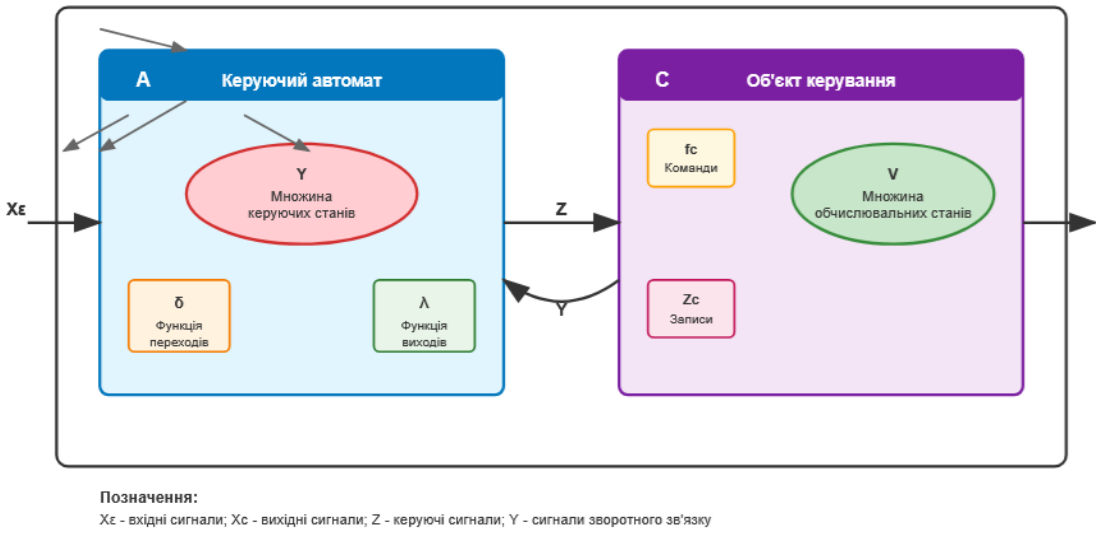

Рисунок 1.2 – Структура автоматизованого об’єкта керування

У модернізованій моделі автомат подається як впорядкована п’ятірка, що включає множину функцій-станів, множини вхідних і вихідних даних, функцію формування виходу та функцію переходів між станами. На відміну від класичних моделей, новий підхід не вимагає жорсткої прив’язки змінних до конкретних функцій-станів, дозволяючи кожній з них мати власний простір змінних. Це створює гнучке середовище для інтерпретації таких функцій як окремих процедур, що працюють у глобальному контексті.

Ключовим поняттям тут виступає процес – логічна сутність, що об’єднує всі альтернативні функції-стани у вигляді єдиної поліморфної структури. При активації процесу виконується лише одна з функцій, яка є поточною. Кожен процес має власний таймер, що обнуляється при зміні стану, дозволяючи відстежувати тривалість перебування у кожному з режимів. Статична структура процесу формується його множиною функцій-станів і початковим станом, тоді як динамічний аспект представлений поточним станом та значенням таймера.

Кожна окрема функція-стан задається як пара, що включає набір умов активації та відповідні реакції. Умови визначаються комбінацією значень інших процесів, значень змінних або таймерів, що фіксують тривалість бездіяльності. Реакції, у свою чергу, представляють сукупність операцій, які

змінюють стан змінних або активують переходи між функціями-станами. Деякі функції можуть не мати жодної реакції, що дає підстави вважати їх пасивними. Серед них виділяють спеціальні режими, які інтерпретуються як нормальне або аварійне завершення.

Сукупність взаємодіючих процесів утворює гіперпроцес – надбудову над множиною процесів, яка підтримує синхронізовану роботу компонентів за єдиним циклічним тактом. На відміну від класичних автоматів, гіперпроцес реалізує логіку логічного паралелізму, дозволяючи процесам бути слабко зв’язаними, з обміном інформацією через глобальні змінні. Початкова активація здійснюється із запуском одного процесу в конкретному стані, тоді як інші перебувають у стані готовності до активації. Пасивні функції відіграють ключову роль у забезпеченні комунікації та синхронізації.

Функціонування гіперпроцесу має подібності до традиційної реалізації автомата, але відрізняється за рахунок циклічності, множинності логічних потоків і наявності глобального періоду тактування. Незалежно від фізичної моделі виконання – послідовної чи паралельної – на рівні логіки гіперпроцес зберігає структурну модель кооперативної багатозадачності. Це означає, що кожен окремий процес функціонує як автономний логічний потік, тоді як загальна система координується централізовано, забезпечуючи черговість та узгодженість дій.

Функції-стани в межах гіперавтомата описуються за допомогою конструкцій умовного переходу мовою програмування С, що включає використання умовних операторів з довільною глибиною вкладеності. Це дозволяє будувати гнучку, легко модифіковану структуру алгоритмічного управління, що відповідає принципам відкритості, залежності від подій, логічної циклічності та асинхронної взаємодії, характерних для сучасних систем автоматичного керування.

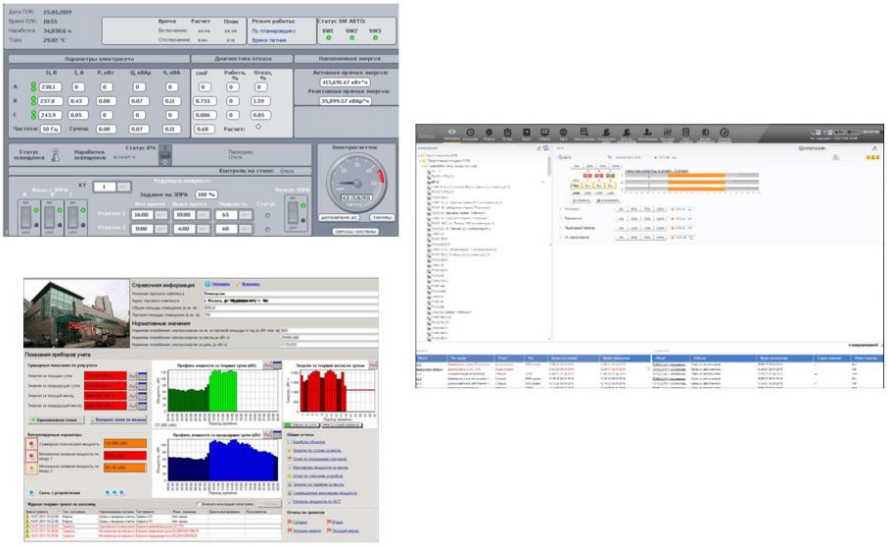

Рисунок 1.3 – Огляд існуючих систем

1.3 Обробка подій у системах керування

У системах логічного управління, особливо в контексті пристроїв контролю периметра, логіка функціонування часто базується на зовнішніх подіях, які ініціюють певну реакцію з боку системи. Події як абстрактне поняття слугують сигналами змін зовнішнього середовища, на які система має відреагувати відповідним чином. У сфері програмування, особливо в розробці інтерфейсів користувача або в середовищах з інтенсивною взаємодією, події можуть проявлятися у вигляді дій користувача, завершення обчислювальних процесів, комунікацій у мережевому середовищі або активації різноманітних сенсорних систем.

Обробка подій, подібно до обробки виключень, реалізується через спеціальні механізми, які забезпечують зворотній зв’язок із зовнішнім середовищем. Взаємодія, що виникає під час взаємного обміну між інтерфейсом та користувачем, розглядається як ключова причина активації обробників подій. У динамічних веб-застосунках, наприклад, генерація внутрішніх реакцій відбувається у відповідь на дії користувача, що ініціює зміни у формі або змістовій структурі документа.

Механізм подієвої взаємодії в системах програмування включає кілька

логічно пов’язаних компонентів. Подія виникає як сигнал, який генерується певним джерелом, наприклад, при взаємодії з графічним компонентом. Цей сигнал приймається слухачем, який організовує його маршрутизацію до обробника. Такий обробник реалізує логіку реагування на зміну в системі, адаптуючи поведінку відповідно до її поточного стану. У багатьох випадках між слухачем і обробником подій діє проміжна структура у вигляді адаптера, що забезпечує узгодженість інтерфейсу, використовуючи порожні методи, які перевизначаються у конкретних реалізаціях.

Складність обробки подій зростає при наявності спільного доступу до системних ресурсів, що вимагає ієрархізації за пріоритетами або часом реакції. Використання анонімних об’єктів або внутрішніх класів дозволяє інкапсулювати логіку взаємодії, обмежуючи доступ до неї ззовні, що відповідає принципам безпечної інкапсуляції програмного коду. Обробка подій також має гнучкість у плані синхронного або асинхронного виконання, що особливо важливо для систем реального часу.

Візуалізація процесу подієвої взаємодії відображає типовий ланцюг активації, починаючи з джерела сигналу, який відстежується слухачем. Надалі подія передається через адаптер до обробника, який реалізує необхідну реакцію системи. Така реакція, залежно від контексту, може призвести до зміни стану керованого об’єкта або до активації нового етапу роботи системи. У певних реалізаціях деякі ролі можуть об'єднуватися в межах одного програмного компонента, що забезпечує компактнішу реалізацію, зберігаючи при цьому загальну структурну логіку.

Таким чином, подієво-орієнтована модель взаємодії в логічних системах управління забезпечує ефективну організацію реакцій на зовнішні впливи, дозволяючи будувати адаптивні, гнучкі й масштабовані архітектури як у сфері побутової автоматизації, так і в більш складних інтерактивних системах.

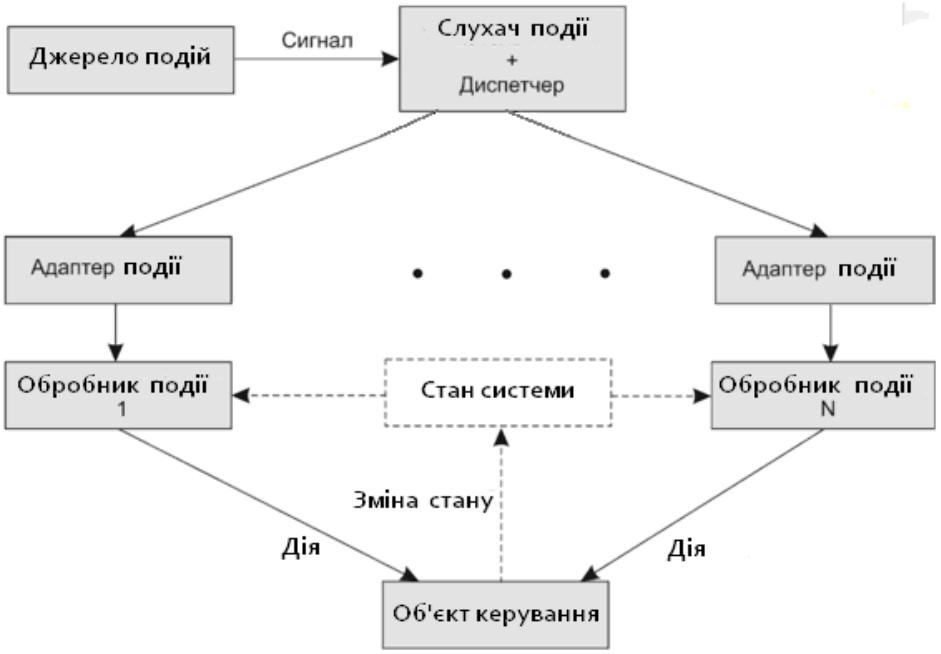

Рисунок 1.4 – Модель програмування на основі подій

У системах реального часу обробка подій відіграє ключову роль, оскільки саме динамічні зміни середовища визначають характер реакцій, які система має здійснити у відповідь. Подія в такому контексті трактується як будь-яке значуще збурення або зміна параметрів зовнішнього чи внутрішнього середовища, яка потребує відповідної реакції керуючої системи. Важливо усвідомлювати, що подія є абстрактним поняттям і може бути актуалізована як зовнішніми джерелами, так і внутрішніми механізмами самої системи. Наприклад, натискання кнопки користувачем виступає зовнішньою подією, яка фіксується системою з певною затримкою, що і визначає параметри реакції в реальному часі. Однак ця ж дія може спричинити ланцюг внутрішніх змін, таких як фіксація зміни рівня сигналу на вході мікроконтролера або ініціація вторинних функціональних процесів.

Обробка подій у системах керування зазвичай ієрархізується відповідно до рівнів абстракції. Низькорівнева подія, пов’язана із фізичним сигналом, наприклад зростанням або спадом напруги, трансформується засобами фільтрації або стабілізації в події середнього рівня, що відповідають конкретним діям користувача. Згодом вони можуть бути

використані для генерації більш узагальнених високорівневих подій, таких як серія натискань або тривале утримання елемента управління, які вже мають семантичну інтерпретацію в рамках логіки системи.

У програмованих мікроконтролерах обробка подій здійснюється за допомогою послідовного виконання інструкцій, де кожна подія викликає зміну поточного стану системи, відповідно до заздалегідь визначеного сценарію. Характерною ознакою таких систем є їх дискретна природа, обумовлена періодичністю обробки інструкцій у рамках тактової частоти процесора. Таким чином, мінімальна одиниця часу, за яку може бути здійснена реакція системи, визначається інтервалом виконання однієї машинної інструкції, що притаманно усім кінцевим динамічним системам.

В умовах експлуатації системи часто мають справу з двома типами часової чутливості: жорстким та м’яким реальним часом. Перший тип пов'язаний з точним дотриманням моментів активації та синхронізації, що характерно для цифрової обробки сигналів або синтезу, де події регламентуються тайм-контролем. Натомість м’який реальний час пов'язаний з обробкою асинхронних подій, що надходять у довільні моменти, як-от взаємодія з користувачем або зовнішніми пристроями. Це зумовлює поділ функцій на синхронні та асинхронні, що об’єднуються в окремі підсистеми в рамках загальної архітектури.

Визначальним чинником ефективності системи є час реакції на подію, що складається із затримки її фіксації та часу обробки. Ці показники залежать не лише від фізичних властивостей апаратної платформи, але й від специфіки алгоритмів, які використовуються в реалізації логіки управління. Процес фіксації події може реалізовуватися кількома методами. Наприклад, при використанні техніки опитування програмний цикл періодично перевіряє значення вхідних параметрів і за відсутності змін виконується безрезультатно. Натомість механізм переривання забезпечує негайну реакцію на зовнішню подію шляхом призупинення основного виконання і переходу до спеціалізованої підпрограми обробки. У складніших випадках

застосовуються комбіновані підходи, як-от прямий доступ до пам’яті, який дозволяє оптимізувати продуктивність.

Окремий інтерес становлять системи з поділом на фонову та активну частини. У таких структурах основний цикл виконує фонову функцію, тоді як критичні події обробляються через переривання, що мають пріоритет над основною логікою. Такий підхід дозволяє ефективно поєднувати безперервність функціонування з оперативністю реагування, що робить його придатним для реалізації у вбудованих системах, орієнтованих на реальний час.

Оскільки механізми обробки переривань реалізуються на апаратному рівні, це забезпечує значні переваги щодо ефективності та швидкодії при організації багатозадачного виконання у вбудованих системах. Такий підхід дозволяє уникнути складної координації потоків або контекстного перемикання, характерного для програмного забезпечення, що працює у середовищах з повноцінною підтримкою багатозадачності. Реакція на події в цих системах відбувається майже миттєво, що є критично важливим для задач реального часу, де визначальною є мінімальна затримка між моментом виникнення події та початком її обробки.

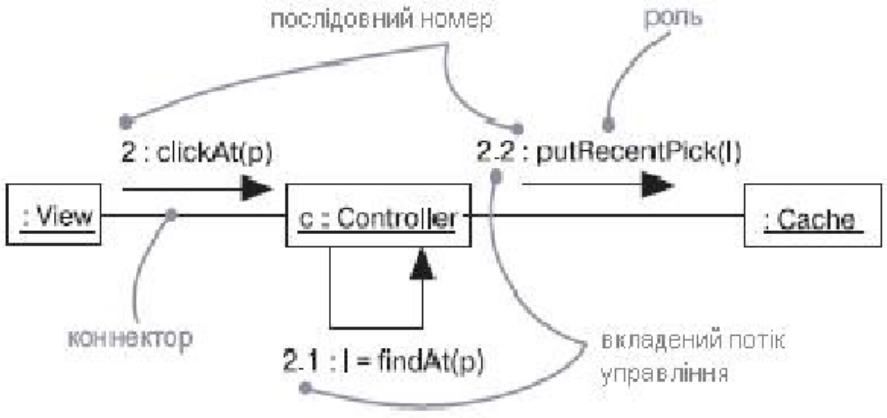

Рисунок 1.5 – Виконання програми з вкладеністю переривань

Для забезпечення коректності обробки декількох одночасних або майже одночасних подій у таких системах застосовується механізм

маскування переривань. Його суть полягає у тимчасовому блокуванні певних класів переривань під час виконання обробника, що запобігає конфліктам доступу до спільних ресурсів і порушенням логіки виконання. Маскування може бути повним або частковим, залежно від ієрархії пріоритетів, яка встановлена для переривань. Це забезпечує ефективну реалізацію взаємного виключення без потреби у додаткових програмних засобах синхронізації.

З метою збереження поточного стану програми при переході до обробника переривання використовується єдиний стек викликів. Він виконує функцію тимчасового сховища для ключових параметрів виконання, включно з вмістом регістрів, лічильників команд та змінних, що визначають контекст роботи основної програми. У випадках, коли система допускає вкладені переривання, обсяг цього стеку має бути достатнім для розміщення кількох рівнів контексту, що дозволяє забезпечити безпомилкове повернення до відповідного рівня виконання після завершення кожного з обробників. Таким чином, завдяки апаратним засобам підтримки переривань можлива реалізація ефективної багаторівневої обробки подій у режимі жорсткого реального часу з мінімальними витратами ресурсів.

1.4 Технічне завдання на проектування

Метою проєктування є створення ефективного мікроконтролерного пристрою, що забезпечує автоматизоване керування освітленням пішохідного переходу, орієнтоване на підвищення безпеки руху в темний час доби. Реалізоване рішення не лише вмикає зовнішнє освітлення переходу, але й динамічно підсвічує смуги «зебри», що дозволяє візуально супроводжувати пішохода під час його руху через проїжджу частину, роблячи його переміщення прогнозованим та помітним для водіїв транспортних засобів.

Алгоритм функціонування пристрою передбачає його активацію лише в умовах недостатнього освітлення, що забезпечується шляхом ручного вмикання відповідного режиму. Подальше керування здійснюється за

ініціативою пішохода через натискання кнопки викличного табло, після чого вмикається червоне світло для транспорту та активується зовнішнє освітлення переходу. Додатково задається пауза, що дозволяє пішоходу підготуватись до перетину, а транспорту – залишити перехрестя.

Ключовим етапом є активація підсвічування смуг переходу, що реалізується у вигляді послідовної зміни інтенсивності або кольору світлових елементів уздовж траєкторії руху пішохода. Підсвічування має двосторонній напрямок: від краю проїжджої частини до її центру і в зворотному напрямку, що дозволяє враховувати різні напрямки руху. Темп підсвічування адаптується до ширини переходу, а отже, і до кількості смуг для транспорту, що забезпечує належну тривалість супроводу пішохода.

Для уникнення повторного виклику підсвічування впродовж часу, коли ще триває процес переходу або фаза розвантаження транспортного потоку, кнопка блокується. У разі повторного натискання запит зберігається в пам’яті і активується автоматично після завершення поточної фази. У випадку відсутності нових запитів система переходить до стану очікування.

Підсвічування може бути реалізовано як за рахунок надсмугового розміщення світлових елементів, що створюють ілюзію освітлених смуг, так і шляхом інтеграції джерел світла безпосередньо у покриття переходу. Такий варіант надає додаткову варіативність реалізації залежно від конкретних умов місцевості.

Пріоритетом у проєктуванні є використання максимально простих і доступних апаратних засобів. У якості керувального елементу обрано мікроконтролер на платформі Arduino, що дозволяє швидко реалізувати макет системи з мінімальними витратами. Візуалізація роботи системи на етапі розробки здійснюється за допомогою кольорових світлодіодів на відлагоджувальній платі, що дозволяє моделювати як основне освітлення, так і ефект динамічного підсвічування смуг пішохідного переходу.

2 ТЕХНОЛОГІЧНА ПЛАТФОРМА РЕАЛІЗАЦІЇ

2.1 Функціональна схема системи автоматизованого керування освітленням пішохідного переходу

В сучасному ландшафті проектування цифрових пристроїв спостерігається значне різноманіття підходів до формального опису проектів. Серед найбільш поширених у світовій практиці методологій варто відзначити аналітичні мови опису апаратури, а також графічні та візуальні засоби представлення у вигляді ієрархічних цифрових структур, схем та графів переходів автоматів. Провідні промислові системи автоматизованого проектування зазвичай прагнуть забезпечити максимально широкий спектр інтерфейсів для задоволення потреб найвибагливіших користувачів на ринку проектувальних систем.

Одним з широко застосовуваних методів вхідного опису моделей спеціалізованих цифрових обчислювальних пристроїв обробки даних та логічного керування виступає концепція абстрактного кінцевого автомата. Такий автомат характеризується сукупністю взаємопов'язаних компонентів, що включають множину символів вхідного алфавіту, множину станів автомату, множину символів вихідного алфавіту, а також функції переходів та виходів автомату. Найпоширенішими формами представлення абстрактного автомату в практиці проектування є таблиця переходів-виходів та граф переходів автомату.

Структурні моделі автоматів, що застосовуються в задачах логічного управління, оперують принципово відмінною термінологією. У цьому підході символи вхідного та вихідного алфавітів співвідносяться з векторами вхідних та вихідних змінних, які при схемній реалізації відповідають зовнішнім входам та виходам схеми. Функціонування структурного автомата відбувається в автоматному часі, що вимірюється дискретними тактами, при

цьому перехід між станами здійснюється протягом одного автоматного такту.

Відображення станів абстрактного автомату на структурному рівні становить більш складну задачу. Кожен стан абстрактного автомата кодується відповідним вектором внутрішніх змінних, що дозволяє представити модель структурного автомату в аналітичному вигляді через систему функцій виходів та переходів.

Загальноприйнята модель структурного автомата, відома як модель Хаффмена, базується на поєднанні комбінаційного та послідовнісного компонентів. Послідовнісний компонент інкорпорує елементи пам'яті, зокрема синхронні тригери, які забезпечують збереження стану та дозволяють здійснювати його синхронні зміни. Комбінаційний компонент реалізується через систему логічних елементів, що виконують дві ключові логічні функції: обчислення значень вихідних сигналів та визначення нових значень елементів пам'яті, що відповідають наступному стану автомата.

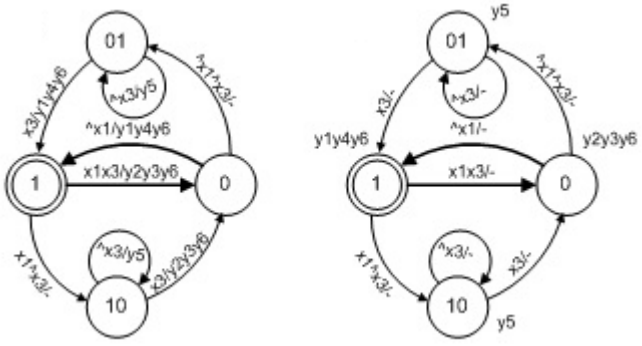

Рисунок 2.1 – Модель структурного автомату

Тривалість автоматного такту в структурних автоматах може характеризуватися різними підходами. У простішому випадку вона залишається однаковою для всіх станів автомату та визначається виключно частотою синхропослідовності. Альтернативний підхід передбачає індивідуальну тривалість для кожного окремого стану автомату.

Розвиток концепції структурного автомату для динамічних систем

реального часу потребує розширення традиційної моделі шляхом інтеграції параметру автоматного часу безпосередньо у функцію переходів. Таке вдосконалення призводить до трансформації класичного графу переходів у темпоральний граф, що враховує часові характеристики системи.

Для коректного застосування концепції темпорального графу переходів необхідно уточнити фундаментальні поняття та визначення. Стан автомату представляє математичну абстракцію, що характеризується сукупністю стійких параметрів із урахуванням передісторії динамічної системи та часу існування зазначеної сукупності параметрів.

Вхідні сигнали структурного автомата класифікуються за двома категоріями: значення та події. Значення корелюють із вхідними змінними автомату та зазвичай існують протягом повної тривалості автоматного такту, здійснюючи вплив на вихідні сигнали. Події характеризуються потенційно меншою тривалістю порівняно з автоматним тактом та впливають на процеси зміни стану автомату. Типовими прикладами подій виступають синхропослідовності або зовнішні керуючі сигнали.

Тривалість кожного автоматного такту приписується конкретному стану автомату, що відповідає вершині графу переходів. Цей параметр визначає часовий інтервал незмінності стану автомату. Події володіють здатністю переривати перебування автомату в певному стані та ініціювати перехід безпосередньо в момент їх виникнення. Вихідні сигнали автомату можуть впливати на події через механізми умов скидання.

Концепція гіперавтомату розширює традиційну модель автомату введенням гіперстанів. Гіперстан являє собою сукупність станів автомату, переходи між якими здійснюються безумовно з урахуванням виключно часу існування кожного окремого стану. Стани автомату, що входять до гіперстану, характеризуються повною або частковою цикличністю. Вхід до гіперстану завжди відбувається через перший стан із множини станів гіперстану, незалежно від початкового стану основного автомату. Вихід з гіперстану реалізується або по завершенні часу існування гіперстану, або за

наявності зовнішньої події, що формується поза множиною вихідних оповіщувальних сигналів автомату.

Гіперстан у програмній реалізації представляє структуру, інверсну до конструкції процесу в мовах опису апаратури. Якщо в процесі обчислення значень сигналів та змінних відбувається незалежно від параметру часу, то в гіперстані функції переходів обчислюються виключно від параметру часу. Стани автомату в межах гіперстану відображуються моделлю без вхідних сигналів.

Гіперстан не володіє окремими вихідними змінними, використовуючи виключно вихідні змінні основного автомату з можливістю взаємного використання. У гіперстані допускається зміна оповіщувальних сигналів основного автомату, проте вихід з гіперстану здійснюється або за параметром часу, або за зовнішнім сигналом, що не є оповіщувальним для жодного зі станів автомату.

На основі розглянутих моделей абстрактного та структурного автоматів може бути формалізований алгоритм керування освітленням пішохідного переходу з підсвіченням смуг зебри. Множина вхідних сигналів включає сигнали включення та виключення пристрою освітлення переходу, а також сигнал натиснення кнопки, що ініціює основний режим освітлення переходу та підсвічення його смуг. Множина вихідних сигналів охоплює сигнал включення аварійного освітлення кнопки при включенні системи за відсутності пішоходів, сигнал включення основного освітлення всього переходу прожектором, сигнал включення забороняючого червоного світла дорожнього світлофору, а також сукупність сигналів підсвічення пар смуг зебри переходу.

Початковий стан системи характеризується перебуванням у неактивному режимі. Активація системи сигналом включення ініціює перехід до режиму очікування, при якому здійснюється освітлення виключно кнопки виклику, що сигналізує про готовність системи до роботи.

Натиснення кнопки виклику, що розглядається як подія, призводить до

переходу системи в активний режим функціонування. У цьому стані активується основне зовнішнє освітлення переходу та забороняючий червоний сигнал дорожнього світлофору для автотранспорту. Система надає певний часовий інтервал для підготовки пішохода до переходу, протягом якого транспортні засоби повинні залишити зону пішохідного переходу.

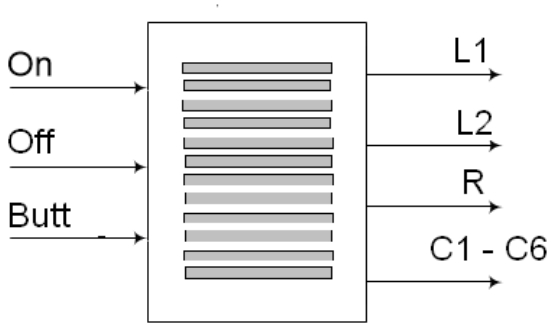

Рисунок 2.2 – Інтерфейс пристрою керування освітленням пішохідного переходу

По завершенні підготовчого періоду система переходить до гіперстану, що характеризується зустрічним підсвічуванням смуг зебри переходу. Цей динамічний процес супроводжує пішохода при перетині проїжджої частини автодороги, створюючи візуальний ефект руху світла, що досягає середини переходу та повертається назад, забезпечуючи зручність для пішоходів, які перетинають дорогу з різних напрямків.

Часові параметри системи визначаються характеристиками конкретної дорожньої інфраструктури. Для дороги із двома смугами руху, що відповідає дванадцяти смугам зебри, тривалість гіперстану становить приблизно п'ятнадцять секунд. При цьому час повернення до попереднього стану складає три секунди, формуючи повний цикл роботи системи. Збільшення тривалості гіперстану пропорційно впливає на час повернення до початкового стану.

Завершення циклу гіперстану призводить до переходу системи в режим очікування наступного натиснення кнопки або отримання сигналу

виключення системи. Така циклічна організація роботи забезпечує ефективне функціонування системи освітлення пішохідного переходу.

Для практичної реалізації керуючого автомату обрано модель автомату Мура, оскільки вихідні сигнали пристрою керування залежать виключно від поточних станів автомату, а їх тривалість визначається внутрішньою затримкою відповідного стану автомату. Такий підхід забезпечує стабільність та передбачуваність роботи системи керування освітленням пішохідного переходу

3 РЕАЛІЗАЦІЯ ПРИСТРОЮ УПРАВЛІННЯ

3.1 Принципи автоматного програмування

Сучасні системи логічного управління характеризуються дуальною структурою, що включає апаратну та програмну складові. Апаратна компонента відрізняється специфічною організаційною особливістю, коли розробка здійснюється одними фахівцями, а виготовлення та комплектування реалізується іншими спеціалістами. Така організація процесу зумовлює необхідність проведення розробки у формі детального проектування, що передбачає створення значної кількості різнотипних схем та конструкторських документів, які відповідають чинним стандартам та всебічно відображають усі аспекти життєвого циклу апаратури. Подібний підхід забезпечує можливість порівняно простої модифікації апаратної частини системи навіть через тривалий час та силами інших фахівців.

Принципово відмінна ситуація характеризує створення програмного забезпечення, оскільки його розробка та виготовлення здійснюються одними й тими ж спеціалістами. Внаслідок цього розробка програм зазвичай не виконується у формі проекту з тим же ступенем деталізації, що застосовується для апаратури, що часто призводить до значних труднощів при необхідності подальшої модифікації програмних компонентів.

Для систем логічного управління з введенням вхідних параметрів за запитом була запропонована SWITCH-технологія, спеціально призначена для алгоритмізації та програмування завдань логічного управління. У цій технології базовим виступає поняття стану, до якого додаються концепції вхідної та вихідної дій, формуючи комплексне уявлення про автомат як сукупність станів, вхідних та вихідних дій.

Відповідний підхід до програмування отримав назву автоматного програмування. Центральною ідеєю автоматного програмування є

відокремлення опису логіки поведінки від опису його семантики. Логіка поведінки визначає умови, за яких необхідно виконувати ті чи інші дії, тоді як семантика розкриває власне зміст кожної з дій. Опис логіки при автоматному підході характеризується жорсткою структурованістю, що робить автоматний опис складної поведінки наочним та зрозумілим.

Представлення автоматних моделей базується на використанні поняття автомата як фундаментальної концепції, на відміну від традиційних понять класу, об'єкта, алгоритму чи агента. Застосовується модель змішаного автомата, поведінка якого описується відповідним графом переходів. У загальному випадку автомати розглядаються не ізольовано, а як складові частини взаємопов'язаної системи автоматів, поведінка якої формалізується через систему взаємопов'язаних графів переходів.

Графи переходів автомата виступають не лише візуальним відображенням алгоритму функціонування, але й повною математичною моделлю системи. Стани кожного автомата первинно визначаються через виділені стани об'єкту управління або його частини, а при значній їх кількості через алгоритм управління, побудований в альтернативній нотації. На етапі проектування, на відміну від традиційного програмування, використовується кодування станів автомата із застосуванням багатозначного коду для розрізнення станів через десяткові номери.

Вхідні дії класифікуються за характером дії на події короткочасної дії та вхідні змінні, що вводяться шляхом опитування. Доцільність реалізації вхідних дій у вигляді вхідних змінних поєднується з можливістю застосування подій для скорочення часу реакції системи. При цьому одна вхідна дія може одночасно представлятися як подією, так і вхідною змінною. Усі вихідні дії характеризуються як дії, а не як діяння, при цьому групи вхідних та вихідних дій пов'язуються зі станами, виділеними для кожного автомата.

Переривання в системі обробляються операційною системою та передаються програмі у формі повідомлень, після чого підлягають обробці як

події за допомогою спеціалізованих обробників. Формування деяких вхідних змінних може здійснюватися внаслідок порівняння вхідних аналогових сигналів з еталонними значеннями.

Реалізація функціонування автоматів передбачає можливість їх одноразового запуску з передачею певної події або багаторазового циклічного запуску з передачею однієї події. При практичній реалізації системи необхідно враховувати нереєнтерабельність функцій, що реалізовують автомати, тобто неможливість їх повторного запуску до завершення попереднього виконання. Кожен автомат при запуску реалізує не більше одного переходу, після обробки чергової події зберігає свій стан та переходить у режим очікування наступної події.

Дуги та петлі графів переходів позначаються довільними логічними формулами, які можуть включати вхідні змінні та предикати для перевірки номерів станів інших автоматів і номерів подій. Дуги та петлі, окрім умов переходів, можуть містити послідовно виконувані вихідні дії. Вершини графа переходів можуть включати вкладені автомати, що запускаються послідовно, та послідовно виконувані вихідні дії.

Програмна реалізація на етапі впровадження передбачає побудову програми на відповідній мові програмування або опису апаратури, в якій графи переходів, вхідні змінні, обробники подій та вихідні дії реалізуються у формі функцій. Програма додатково включає допоміжні модулі, що складаються з наборів відповідних функцій, зокрема модулів управління таймерами. Обробники подій містять виклики підпрограм, що реалізовують графи переходів з передачею відповідних подій. Функції вхідних та вихідних дій викликаються з підпрограм реалізації автоматів, тоді як функції допоміжних модулів викликаються з функцій вхідних та вихідних дій.

Кожен граф переходів формально та ізоморфно реалізується окремою функцією, створеною за шаблоном, який містить дві конструкції та умовний оператор. Перша конструкція викликає вкладені автомати та реалізує переходи і дії на дугах і петлях. Умовний оператор перевіряє зміну стану, і у

випадку його зміни друга конструкція активізує вкладені автомати та реалізує дії у новій вершині.

Взаємодія з оточенням організована таким чином, що система взаємопов'язаних автоматів утворює системонезалежну частину програми, яка реалізує алгоритм функціонування системи управління незалежно від операційної системи. Реалізація вхідних змінних, обробників подій, вихідних дій, допоміжних модулів та призначених для користувача інтерфейсів формує системозалежну частину програми.

Верифікація передбачає перевірку кожного графа переходів на досяжність, несуперечність, повноту та відсутність генеруючих контурів. Після реалізації графа переходів текст підпрограми підлягає корегуванню для забезпечення безповторності опитування вхідних змінних, що позначають дуги, які виходять з одного стану, вирішуючи таким чином проблему ризику.

Документування проекту створення програмного забезпечення забезпечує можливість внесення змін через тривалий термін після випуску, включаючи роботу фахівців, які не брали участь у первинному проектуванні. Склад документації включає структурну схему системи, схему розробленого програмного забезпечення, роздруківки екранів користувацьких інтерфейсів, переліки подій, вхідних змінних та вихідних дій, діаграму взаємодії автоматів, опис термінології та форм представлення у графах переходів, шаблони для реалізації графів переходів змішаних автоматів довільного рівня вкладеності. Для кожного автомата документація містить словесний опис як коментар до технічного завдання, схему зв'язків автомата, граф переходів та початковий текст функції реалізації автомата. Додатково включаються алгоритми у формі графів переходів, початкові тексти допоміжних модулів та функцій реалізації вхідних змінних, обробників подій та вихідних дій, протоколи сертифікації програми як контрольні приклади, керівництво програміста та посібник користувача.

3.2 Програмування мікроконтролерного пристрою

3.2.1 Середовище розробки програмного забезпечення

У сучасному технологічному ландшафті спостерігається стрімке зростання популярності систем на основі вбудованих процесорів, які призначені для ефективного виконання вузькоспеціалізованого класу завдань в умовах жорстких обмежень щодо співвідношення продуктивності, енергоспоживання, габаритних розмірів та вартості виготовлення.

У процесі створення вбудованих систем ключову роль відіграє інструментарій розробки, який забезпечує можливість виконання розробки, налагодження та профілювання програм для цільової системи з використанням інструментальної машини. Цей інструментарій включає асемблер, компонувальник, симулятор, налагоджувач та профілірувальник як основні компоненти. Функцію інструментальної машини зазвичай виконує стандартна робоча станція.

Метою застосування крос-інструментів є створення на інструментальній машині файлу з двійковим образом початкового вмісту пам'яті, що включає як машинні команди, так і дані для пристроїв на основі вбудованих процесорів. Створений файл надалі використовується для завантаження в конкретні цільові пристрої.

Процес взаємодії програміста з крос-інструментами зазвичай здійснюється через візуальне інтегроване середовище розробки з вбудованим редактором, механізмами підтримки проектів та різноманітними засобами керування й аналізу результатів роботи окремих інструментів. Реальні програми характеризуються модульною структурою, при якій кожному модулю відповідає окремий файл із вихідними текстами програм, написаними мовами високого рівня або асемблера.

Типова схема розробки програм мікроконтролерів за допомогою крос-інструментарію демонструє послідовність етапів від створення вихідного

коду до отримання готового двійкового образу для завантаження в цільовий пристрій. Цей процес характеризується високим ступенем автоматизації та інтеграції різних компонентів інструментарію, що забезпечує ефективність розробки вбудованих систем.

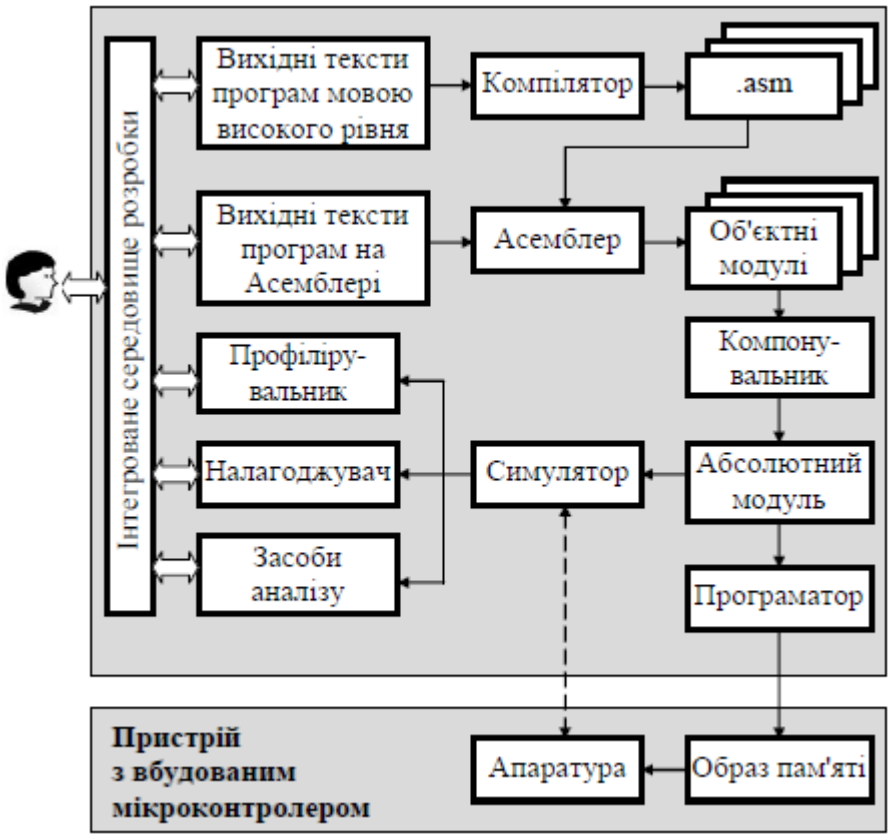

Рисунок 3.1 – Розробка програм за допомогою крос-інструментів

Процес взаємодії програміста з крос-інструментами зазвичай здійснюється через візуальне інтегроване середовище розробки з вбудованим редактором, механізмами підтримки проектів та різноманітними засобами керування й аналізу результатів роботи окремих інструментів. Реальні програми характеризуються модульною структурою, при якій кожному модулю відповідає окремий файл із вихідними текстами програм, написаними мовами високого рівня або асемблера.

Компілятор здійснює трансляцію модулів мовою високого рівня в проміжні асемблерні модулі. Асемблер забезпечує перетворення

асемблерних модулів, як написаних вручну, так і згенерованих компілятором, в об'єктні модулі з машинними кодами та даними для цільової апаратури. Компонувальник виконує складання декількох об'єктних модулів в один абсолютний модуль з об'єднанням відповідних секцій вхідних об'єктних файлів. При цьому здійснюються переміщення символів за абсолютними адресами пам'яті автоматично або відповідно до заданої програмістом карти пам'яті з відповідними виправленнями залежних від них кодів команд і значень даних, що завершує етап складання програми.

Отриманий абсолютний модуль підлягає перетворенню в образ пам'яті для безпосереднього завантаження в цільову апаратуру за допомогою програматора. Взаємодія програміста з моделлю апаратури здійснюється через налагоджувач, який забезпечує можливість перегляду стану моделі та здійснення керованого, включаючи покрокове, виконання цільової програми на рівні окремих команд або рядків вихідного коду. Спільно з налагоджувачем використовуються різноманітні профілірувальники та засоби аналізу для візуалізації отриманих характеристик моделі.

Роль моделі цільової системи для крос-налагоджувача найчастіше виконує симулятор, який дозволяє повністю моделювати цільову апаратуру на інструментальній машині. Існують симулятори різного рівня абстракції від функціонального симулятора на рівні всієї системи до симуляторів рівня системи команд та симуляторів, що емулюють точну структуру апаратури на рівні функціональних блоків і конкретних вентилів. У складі інструментарію крос-розробки зазвичай використовується симулятор рівня системи команд, включаючи урахування конвеєрних ефектів з потактовою точністю.

Середовище розробки Keil являє собою комплексний набір утиліт для виконання повного спектру заходів щодо написання програмного забезпечення для мікроконтролерів. Keil забезпечує роботу з проектами будь-якого ступеня складності, від введення та виправлення вихідних текстів до внутрішньосхемного налагодження коду та програмування мікроконтролера. Від розробника приховується значна частина другорядних функцій, що

суттєво розвантажує інтерфейс та робить керування інтуїтивно зрозумілим. При зростанні складності реалізованих завдань завжди можна задіяти повний потенціал модулів, що функціонують під керуванням єдиної оболонки.

Серед основних програмних засобів Keil виділяється база даних мікроконтролерів, що містить докладну інформацію про всі підтримувані пристрої, включаючи їхні конфігураційні дані та посилання на джерела інформації з додатковими технічними описами. При додаванні нового пристрою в проект усі його унікальні опції встановлюються автоматично. Менеджер проектів служить для об'єднання окремих текстів програмних модулів і файлів у групи, що обробляються за єдиними правилами, дозволяючи значно краще орієнтуватися серед множини файлів.

Вбудований редактор полегшує роботу з вихідним текстом завдяки використанню багатовіконного інтерфейсу та виділенню синтаксичних елементів шрифтом і кольором з можливістю налаштування згідно з уподобаннями розробника. Редагування залишається доступним під час налагодження програми, що дозволяє відразу виправляти помилки або відзначати проблемні ділянки коду.

Засоби автоматичної компіляції, асемблювання та компонування проекту призначені для створення завантажувального модуля програми. Між файлами автоматично генеруються нові асемблерні та компіляційні зв'язки, що дозволяють обробляти виключно ті файли, у яких відбулися зміни або файли, що знаходяться в залежності від змінених. Функція глобальної оптимізації проекту дозволяє досягти найкращого використання регістрів мікроконтролера шляхом багатократної компіляції вихідного коду.

Налагоджувач-симулятор забезпечує налагодження роботи скомпільованої програми на віртуальній моделі мікропроцесора з достатньо вірогідним моделюванням роботи ядра контролера та його периферійного устаткування. Для полегшення комплексного налагодження програмного забезпечення можливе підключення програмних моделей нестандартного устаткування.

Додаткові утиліти полегшують виконання найпоширеніших завдань та включають базу даних програмних символів для швидкого пошуку, засоби пошуку заданого коду у всіх файлах зазначеної папки або проекту, можливість використання утиліт сторонніх виробників та програмування Flash-пам'яті мікроконтролерів.

Середовище програмування було розроблене мюнхенською компанією Keil у 1982 році, а у жовтні 2005 року Keil увійшла до складу американської корпорації ARM. На сьогоднішній день це середовище надає широкий спектр різноманітних засобів для розробки програм, що працює на персональних комп'ютерах під керуванням операційної системи Windows.

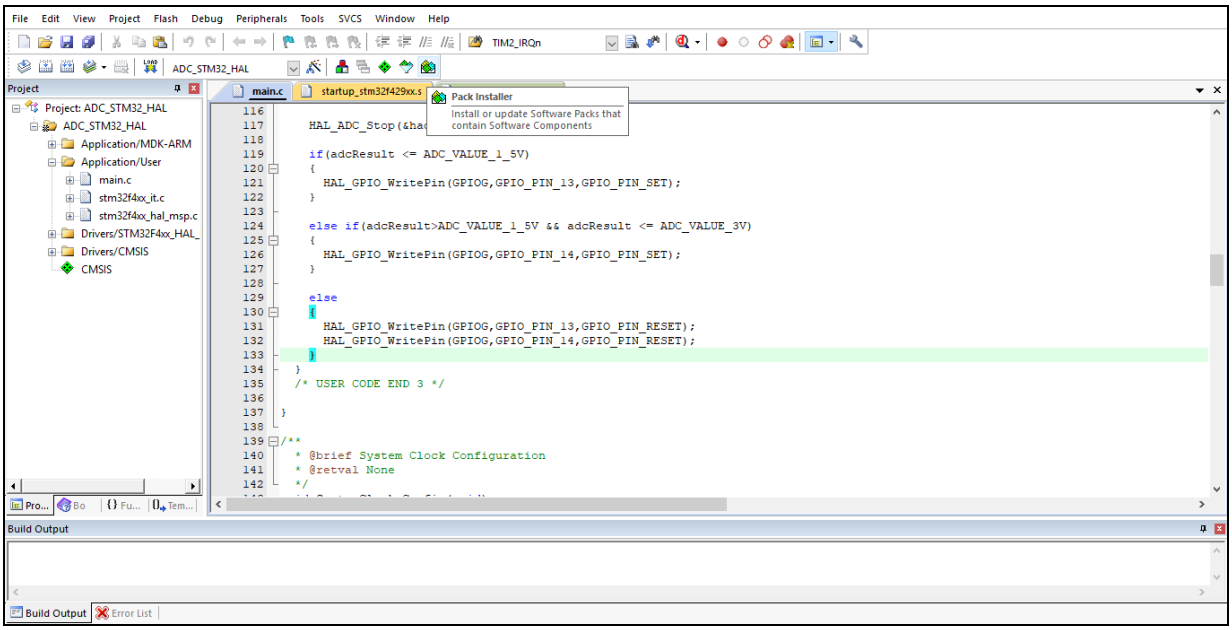

Рисунок 3.2 – Вигляд середовища

Особливою відмінністю Keil for ARM є бібліотека мікроконтролерів сімейства STM, доступна розробнику, яка містить всі мікроконтролери сімейства STMicroelectronics. Компанія STM надає бібліотеку драйверів периферійних пристроїв, бібліотеку для USB та вихідні коди, сумісні з попередніми бібліотеками для мікроконтролерів STM7 і STM9. Для сімейства Cortex доступні численні відкриті та комерційні операційні системи реального часу та сполучне програмне забезпечення.

У Cortex-M3 використовується принципово нова система налагодження CoreSight, доступ до якої здійснюється через порт Debug Access, що підтримує з'єднання через стандартний JTAG або послідовний двохпровідний інтерфейс. Система CoreSight забезпечує управління ходом налагодження та відповідає за точки перегляду даних і трасування, передаючи певну інформацію про програму в середу налагодження для використання під час тестування програмного забезпечення.

Загальний вигляд вікна середовища Keil for ARM представлений на рис. 3.2.

3.2.2 Програмна модель пристрою та її моделювання

Реалізація програмного забезпечення для системи керування освітленням пішохідного переходу базується на мікроконтролері архітектури MCS 51 з використанням мови програмування С в середовищі розробки Keil.

Ініціалізація системи розпочинається з початкового стану повної деактивації всіх компонентів, за якого дорожній рух здійснюється в стандартному режимі функціонування. Активація системи керування освітленням переходу в довільний момент часу шляхом натиснення кнопки виклику може призвести до аварійної ситуації та потенційно до дорожньо-транспортної пригоди. Вирішення цієї проблематики реалізується через впровадження постійного очікуючого таймера з тридцятисекундним інтервалом відліку, який здійснює моніторинг активації системи в цілому та подальшого виконання алгоритму керування освітленням переходу.

При активації системи відбувається перерахунок часових параметрів, що забезпечує відстеження моменту включення самої системи та часу натиснення кнопки ініціації пішохідного переходу. Система налаштована на обробку переривань від таймера для забезпечення точної синхронізації процесів.

Наступний етап функціонування характеризується переходом системи

в режим очікування активації кнопки виклику для здійснення пішохідного переходу. Візуальна індикація стану системи реалізується через спеціальне кольорове позначення біля кнопки, що інформує про її поточний стан активації. Червоне кольорове маркування області навколо кнопки сигналізує про неактивований стан елемента керування, інформуючи користувачів про готовність системи до прийому команди активації пішохідного переходу.

Така організація початкового етапу функціонування системи забезпечує безпечний запуск та готовність до виконання основних функцій керування освітленням пішохідного переходу з урахуванням вимог дорожньої безпеки.

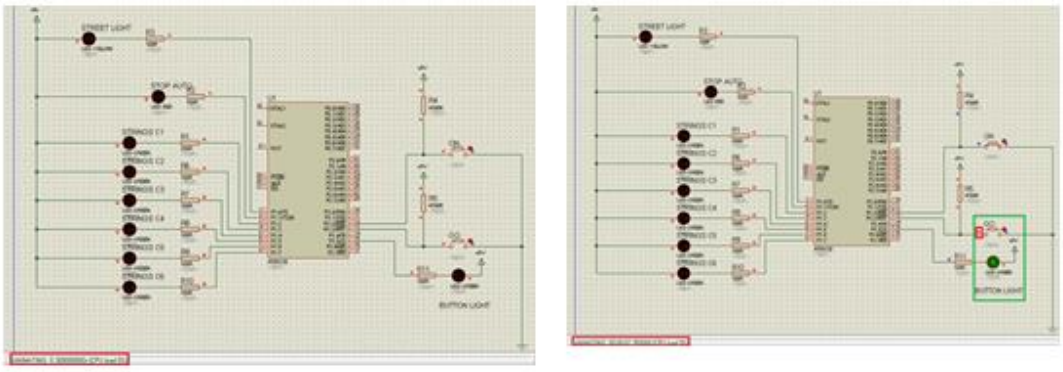

Рисунок 3.3 – Старт системи та очікування натискання кнопки

Система переходить у режим очікування активації кнопки виклику протягом тридцятисекундного інтервалу, встановленого таймером. По завершенні зазначеного часового проміжку система здійснює перевірку стану кнопки активації. У випадку виявлення факту натиснення система ініціює перехід у режим очікування підготовки до пішохідного переходу або активує додатковий п'ятисекундний інтервал очікування для проведення додаткових верифікаційних перевірок стану кнопки.

Активація кнопки користувачем не супроводжується жодними візуальними або звуковими інформаційними сигналами для пішохода. Водночас система отримує повідомлення про зовнішню подію натиснення

кнопки через механізм переривань та після завершення встановленого системою часового інтервалу ініціює процес підготовки до здійснення пішохідного переходу.

Аналогічним чином система не забезпечує жодних інформаційних індикаторів або сповіщень для пішохода в момент відпускання кнопки. Варто зазначити, що кнопка повертається у відпущений стан після незначного часового проміжку від моменту натиснення, що обумовлено необхідністю запобігання явищу дребезжіння контактів та некоректного розпізнавання системою факту активації кнопки ініціації пішохідного переходу.

Така конфігурація системи забезпечує стабільне функціонування механізму активації з урахуванням технічних особливостей електромеханічних компонентів та мінімізацією ризиків помилкової інтерпретації сигналів користувача. Використання переривань для обробки сигналів кнопки гарантує надійність реєстрації команд активації незалежно від поточного стану виконання основного циклу програми.

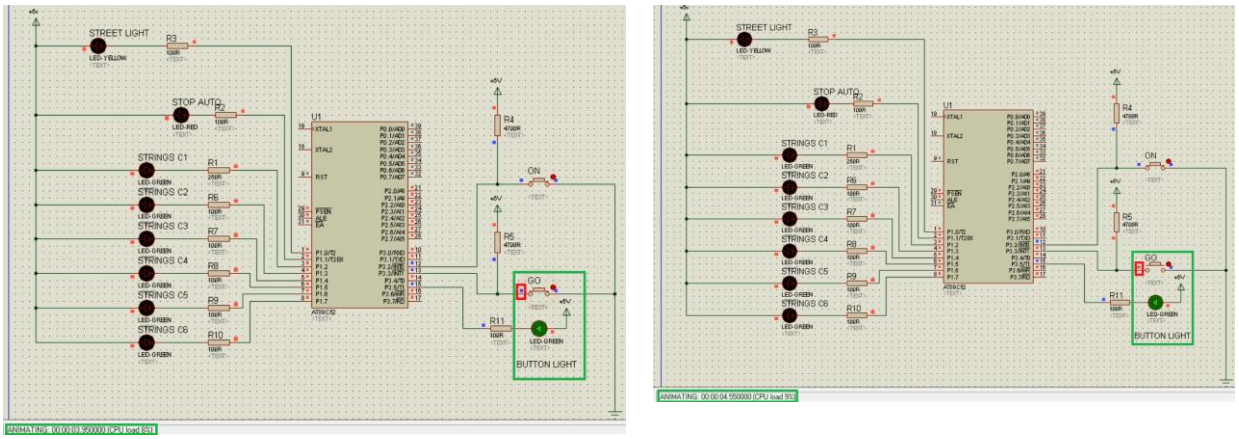

Рисунок 3.4 – Момент натискання та віджимання кнопки

По завершенні основного тридцятисекундного інтервалу для проїзду автотранспорту система припиняє реагування на активацію кнопки виклику, унеможливлюючи будь-які зміни в алгоритмі функціонування незалежно від кількості повторних натиснень. Одночасно система активує режим підготовки до переходу, що створює необхідні умови для готування пішохода до перетину проїжджої частини та надає можливість водіям

транспортних засобів завершити проїзд зони переходу.

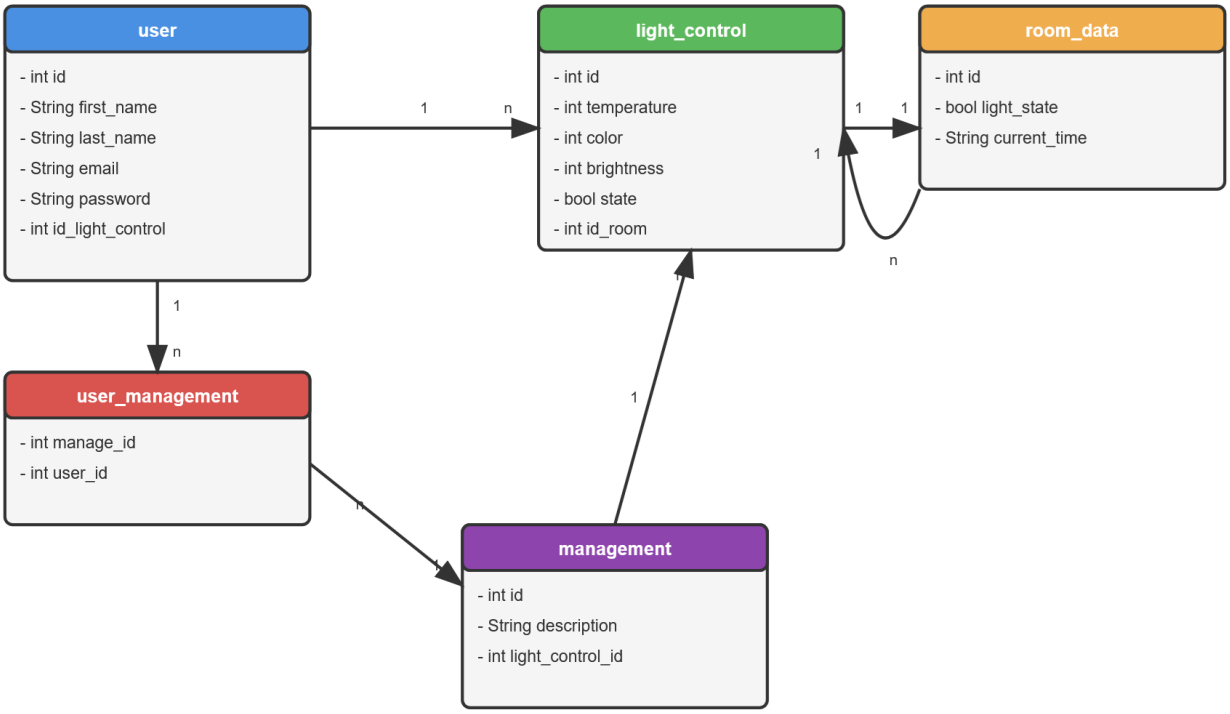

Рисунок 3.5 – Інформаційна модель системи

Наступний етап функціонування характеризується активацією освітлення пішохідного переходу та увімкненням забороняючого сигналу для автотранспорту через червоний сигнал світлофора. На цьому етапі система реалізує п'ятисекундну затримку, призначену для запобігання раптовому виходу пішоходів на проїжджу частину в момент завершення руху транспортних засобів, водії яких не встигли покинути зону переходу своєчасно.

Після забезпечення проїзду всіх транспортних засобів, що не завершили рух до моменту активації забороняючого сигналу, система дозволяє пішоходам здійснювати перетин проїжджої частини. Цей етап супроводжується підсвічуванням смуг пішохідного переходу для забезпечення максимального освітлення зони переходу та покращення видимості пішоходів для водіїв транспортних засобів.

Рисунок 3.6 – Діаграма компонентів системи

Система надає пішоходам часовий інтервал для здійснення переходу, що становить половину від тривалості стандартного дорожнього руху, з додатковим часовим резервом для остаточного покидання проїжджої частини. Цей додатковий інтервал забезпечує безпечне завершення пішохідного переходу перед відновленням руху автотранспорту, гарантуючи дотримання вимог безпеки дорожнього руху та мінімізацію ризиків виникнення конфліктних ситуацій між учасниками руху.

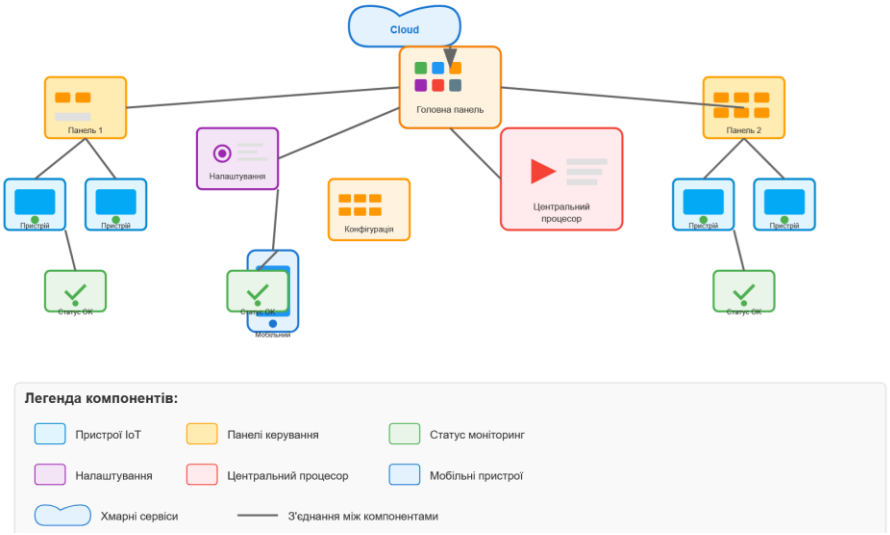

Рисунок 3.7 – Архітектура розподіленої системи

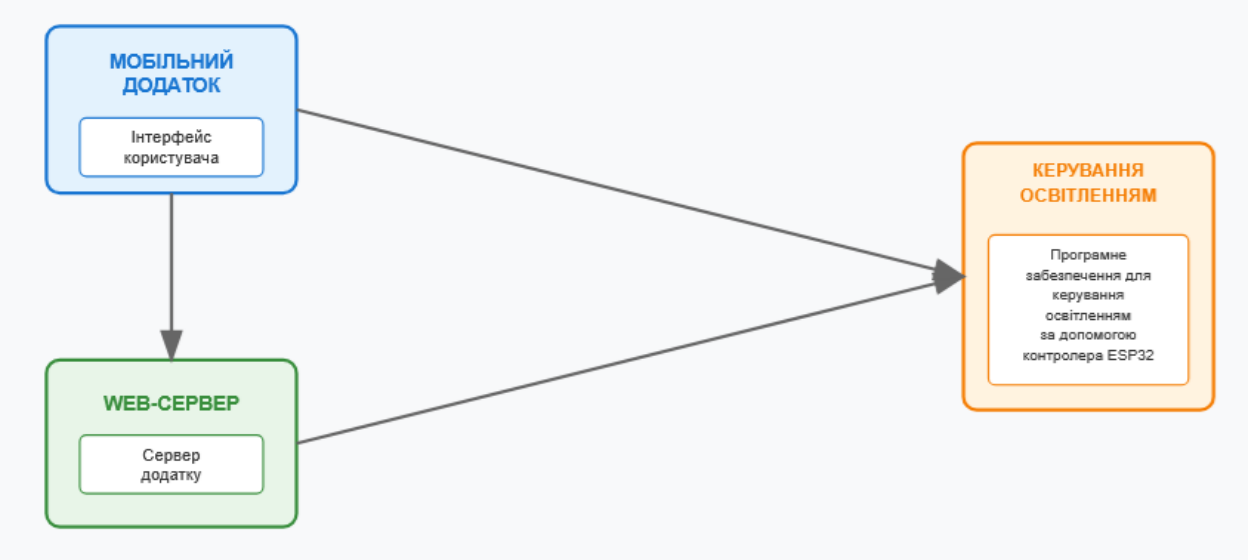

Рисунок 3.8 – Архітектура комп’ютерної інформаційної системи

Завершення процесу пішохідного переходу характеризується поверненням системи до початкового режиму очікування активації після звільнення проїжджої частини та відновлення стандартного дорожнього руху. Повний операційний цикл переходу реалізується через тридцять секунд від моменту готовності системи або після закінчення основного інтервалу з урахуванням додаткових п'ятисекундних періодів повторних очікувань.

Аналіз потенційних порушень функціонування системи та відповідних захисних механізмів виявляє декілька критичних сценаріїв. Першим проблемним випадком є ситуація активації кнопки користувачем з подальшим відключенням та повторним включенням системи внаслідок збоїв електроживлення або перепадів напруги. Протидією такому порушенню служить таймер з функцією обнулення при кожному відключенні живлення, що забезпечує повернення системи до режиму очікування з нульовим лічильником. Принципово важливо зазначити, що активація кнопки при відключеній системі або в режимі очікування не реєструється.

Другим критичним сценарієм є спроба користувача активувати кнопку під час підготовки до переходу або безпосередньо в момент перетину з метою продовження часового інтервалу. Захисний механізм реалізується через деактивацію функції прийому зовнішніх сигналів від кнопки від

початку підготовки до перетину до повернення в режим очікування. Така конфігурація забезпечує повернення системи до тридцятисекундного очікувального режиму для забезпечення проїзду транспорту.

Третім проблемним сценарієм є відключення або збій системи безпосередньо під час здійснення пішохідного переходу. Рішенням цієї проблематики є реалізація одноразового виконання повного циклу з подальшим переходом у режим очікування або повним відключенням системи. Така конфігурація мінімізує ризики виникнення аварійних ситуацій та забезпечує контрольоване завершення операційного циклу навіть за умов технічних збоїв.

ВИСНОВКИ

У рамках виконання кваліфікаційної роботи здійснено комплексне дослідження наявних програмно-апаратних рішень для автоматизації систем управління вуличним освітленням. Дослідження включало систематизацію та критичний аналіз сучасних технологічних підходів, методологій та інструментальних засобів, що застосовуються в галузі автоматизованого керування освітлювальними мережами.

На основі проведеного аналітичного огляду визначено оптимальний набір інструментальних засобів та технологічних рішень для створення комп'ютеризованої системи управління вуличним освітленням. Розроблена архітектурна концепція характеризується високим рівнем надійності та масштабованості, що забезпечує ефективне функціонування системи в умовах інтенсивного користувацького навантаження.

Архітектурне рішення інтегрує передові криптографічні механізми захисту інформації та алгоритми забезпечення безперервності функціонування, що гарантує стабільну експлуатацію системи в різноманітних операційних умовах. Запропонована архітектура демонструє здатність до ефективної обробки запитів множини користувачів при одночасному збереженні високих стандартів інформаційної безпеки та експлуатаційної надійності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мисюк Ю.П., Зовнішнє освітлення міст та безпека дорожнього руху / Ю. П. Мисюк // Світлотехніка та електроенергетика. – 2010. – № 3-4. –С. 33-39.

2. Безпека дорожнього руху. Розмітка дорожня. Загальні технічні вимоги. Методи контролювання. Правила застосування : ДСТУ 2587:2010. – Чинний від 2010-12-27. – К.: Держспоживстандарт України, 2011. – 56 с.

3. Harel D. Statechars: a Visual Formalism for complex systems // Science of Computer Programming. – 1987. – Vol. 8. – P. 231-274.

4. Шалыто А. А. Автоматное программирование / Н.И. Поликарпова, А.А. Шалыто. – Спб.: Питер, 2008. – 167 с.

5. Зюбин В.Е. Программирование информационно-управляющих систем на основе конечных автоматов: учеб.-метод. пособие / В.Е. Зюбин. – Новосибирск: Новосиб. гос. ун-т., 2006. – 96 с.

6. Орлов С.А. Теория и практика языков программирования: учебник для вузов. Стандарт 3-го поколения. – СПб.: Питер, 2013. – 688 с.

7. Матюшин А.О. Программирование микроконтроллеров: Стратегия и тактика / А.О. Матюшин. – М.: ДМК Пресс, 2017. – 356 с.

8. Карпов Ю.Г. Теория автоматов / Ю.Г. Карпов. – СПб.: Питер, 2003. – 208 с.

9. Insider's Guide to the STM32 ARM®Based Microcontroller [Електронний ресурс] / Hitex development tools. – Режим доступу: www / URL: https:// www.hitex.com – 20.09.2018 г. – Загол. з екрану.