УДК 004.932:004.89:656.056.4 DOI: 10.31673/2412-9070.2026.318120

А. П. БОНДАРЧУК[13], д-р техн. наук, професор;
ORCID: 0000-0001-5124-5102
А. А. СТРАЖНІКОВ[2], аспірант;
ORCID: 0009-0000-3492-2968
О. В. ПРОНЬКІН[2], аспірант;
ORCID: 0009-0004-8529-6919
М. М. ЛИСЕНКО[2], аспірант,
ORCID: 0009-0000-3734-8332
[1]Київський столичний університет імені Бориса Грінченка, Київ
[2]Державний університет інформаційно-комунікаційних технологій, Київ
[3]Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

ІНТЕЛЕКТУАЛЬНА СИСТЕМА КЕРУВАННЯ СВІТЛОФОРОМ НА ОСНОВІ КОМП'ЮТЕРНОГО ЗОРУ

У статті представлено розробку та експериментальну перевірку прототипу інтелектуальної системи керування світлофором на основі комп'ютерного зору та глибокого навчання. Неефективність традиційних світлофорів із фіксованим циклом спричиняє хронічні затори, призводить до втрат часу та палива і перешкоджає проїзду пріоритетних транспортних засобів (швидка допомога, пожежні машини, аварійні служби). Запропонована система вирішує цю проблему через адаптивне керування фазами на основі аналізу відеопотоку в реальному часі.

Архітектура реалізована на Python із використанням Ultralytics YOLOv8n для виявлення об'єктів, OpenCV для захоплення відео та візуалізації, FastAPI і Uvicorn для асинхронного REST API з автоматичною документацією. Модуль просторового аналізу спирається на полігональні зони («північ–південь», «схід–захід», пішохідна), конфігурація яких зберігається у форматі JSON відповідно до реальної геометрії перехрестя. Логіка керування реалізована як скінченний автомат із шістьма фазами: зелений і жовтий для напрямку «північ–південь», червоний для всіх напрямків, зелений і жовтий для напрямку «схід–захід», червоний для всіх напрямків. Модуль рішень поєднує часові обмеження (мінімальна і максимальна тривалість зеленої фази) з динамічним порівнянням завантаженості та забезпечує пріоритетний проїзд екстрених служб із негайним перемиканням фази. Усі рішення фіксуються з поясненням причини перемикання. Потокобезпечна архітектура синхронізує доступ до спільних даних між відеоциклом і вебсервером. REST API надає ендпоінти для моніторингу стану системи, отримання метрик (кількість транспортних засобів, пішоходи, пріоритетний транспорт, FPS) і ручного керування фазами. Проведено аналіз суміжних досліджень із YOLO, LLM-агентів і навчання з підкріпленням, що підтверджує відповідність рішень глобальним тенденціям. Визначено обмеження прототипу (одне перехрестя, відсутність трекінгу, емулятор контролера) й окреслено дорожню карту: інтеграція з промисловими контролерами, трекінг, підтримка кількох камер, збереження метрик у базі даних і резервування на випадок відмови штучного інтелекту. Прототип може слугувати основою для промислових систем керування транспортними потоками у розумних містах.

Ключові слова: штучний інтелект, адаптивне керування світлофором, комп'ютерний зір, глибинне навчання, детекція об'єктів, інформаційна система, управління, відеоаналітика.

Вступ

Сучасні міста стикаються з постійно зростаючим навантаженням на транспортну інфраструктуру. Щодня мільйони автомобілів простоюють у заторах, витрачаючи час своїх власників та забруднюючи повітря. Традиційні світлофори з фіксованим циклом перемикання, хоч і виконують базову функцію регулювання руху, не здатні адаптуватися до реальної ситуації на дорозі. Вони не бачать, чи є взагалі автомобілі на порожньому напрямку, і не можуть надати перевагу завантаженій смузі, що призводить до неефективного використання дорожнього часу та штучного створення заторів там, де їх могло б і не бути.

Саме для вирішення цієї проблеми було розроблено концепцію "розумного" перехрестя, а саме, систему, яка в реальному часі аналізує дорожню ситуацію за допомогою комп'ютерного зору та самостійно приймає рішення про оптимальну тривалість фаз світлофора. Замість жорсткого таймера, така система бачить автомобілі та пішоходів, оцінює завантаженість кожного напрямку та динамічно перемикає сигнали, мінімізуючи час очікування. Це не просто автоматизація, це надання світлофору "зору" та елементарного "інтелекту", здатного самостійно реагувати на зміну трафіку.

У цій статті представлено прототип інтелектуальної системи керування світлофором, розроблений на Python. Прототип отримує відеопотік з камери або файлу, за допомогою нейромережі YOLO виявляє транспортні засоби та пішоходів, аналізує їхню кількість у кожному напрямку та на основі простих, але ефективних правил ухвалює рішення про зміну фаз. Система має власний REST API для моніторингу та ручного керування, а також візуальний інтерфейс із накладанням усієї службової інформації на відео.

Постановка проблеми. Сучасні міста стикаються з критичним викликом: транспортні потоки зростають швидше, ніж розвивається дорожня інфраструктура. Щодня мільйони водіїв простоюють у заторах, витрачаючи години робочого часу та палива. За даними аналітичних центрів, великий відсоток міських заторів створюються саме через неефективну роботу світлофорів, які не враховують реальну ситуацію на дорозі. Традиційні системи керування з фіксованим циклом перемикання фаз працюють за жорстким розкладом, незалежно від того, чи є взагалі автомобілі на порожньому напрямку, чи накопичилася багатокілометрова черга на іншому. Це призводить до парадоксальної ситуації, коли світлофор дає зелене світло порожній дорозі, змушуючи водіїв на завантаженій смузі чекати даремно.

Окремою проблемою є реагування на пріоритетний транспорт, тобто швидку допомогу, пожежні машини чи аварійні служби. У критичних ситуаціях кожна секунда може коштувати життя, однак звичайні світлофори не здатні ідентифікувати спецтранспорт і надати йому "зелену хвилю" автоматично. Водночас сучасні технології комп'ютерного зору та штучного інтелекту вже досягли рівня, коли можна в реальному часі розпізнавати об'єкти на дорозі з високою точністю. Виникає потреба у створенні інтелектуальної системи, яка бачила б дорожню ситуацію, аналізувала її та приймала адаптивні рішення, мінімізуючи затори та забезпечуючи пріоритетний проїзд спецтранспорту.

Аналіз останніх досліджень і публікацій

Проблема адаптивного керування світлофорами перебуває в центрі уваги наукової спільноти, особливо з розвитком технологій комп'ютерного зору та глибинного навчання. Сучасні дослідження демонструють стрімку еволюцію підходів: від простих правил на основі детекції до складних гібридних систем [1]-[2], що поєднують комп'ютерний зір з великими мовними моделями (LLM) та навчанням з підкріпленням (RL). Зокрема, робота Yao зі співавторами (2025) пропонує інтеграцію YOLOv11 з LLM-агентом, де мовна модель аналізує дорожню ситуацію на основі "індексу терміновості" звільнення фази, що дозволило зменшити середній час очікування понад 5% у звичайних умовах та 20% за наявності екстреного транспорту[3]. Це підтверджує перспективність поєднання правил з інтелектуальним аналізом контексту, що частково реалізовано і в нашому прототипу. Інший важливий напрям — використання чисто візуальних методів для динамічного керування фазами. Дослідження Majeed та інших (2025) із застосуванням YOLOv10 на реальних даних у Багдаді продемонструвало приголомшливе

скорочення часу очікування до 91,7% порівняно з фіксованим циклом [4]. Водночас група румунських дослідників представила фреймворк DeepSIGNAL-ITS, який поєднує YOLOv8 для детекції на периферійних пристроях (RSU) з алгоритмом Proximal Policy Optimization (PPO) для оптимізації сигналів, досягнувши зменшення часу очікування на 30,2% та помітного зниження викидів CO2 [5]. Це підкреслює правильність обраного нами вектору розвитку — інтеграції з реальним залізом та додавання алгоритмів навчання з підкріпленням для підвищення ефективності. Аналіз літератури також виявляє загальновизнані обмеження поточних систем, які збігаються з нашими спостереженнями. У більшості робіт наголошується на необхідності переходу від поодиноких перехресть до мережевої координації, використання трекінгу (ByteTrack/DeepSORT) для уникнення подвійного підрахунку та калібрування зон детекції під різні ракурси камер [7]-[8]. Крім того, оглядові статті з розвитку AIGC (AI Generated Content) в інтелектуальних транспортних системах вказують на потребу в пояснюваності рішень та етичній відповідності алгоритмів , що ми частково вирішили через фіксацію причин перемикання фаз у HUD та API. Таким чином, представлений прототип знаходиться в руслі сучасних світових тенденцій, пропонуючи працюючу реалізацію базових принципів, яка може бути розширена до рівня передових дослідницьких зразків.

Метою даного дослідження є розробка та експериментальна верифікація прототипу інтелектуальної системи адаптивного керування світлофором, яка на основі аналізу відеопотоку в реальному часі з використанням технологій комп'ютерного зору, здатна автоматично виявляти транспортні засоби та пішоходів, оцінювати завантаженість напрямків руху, динамічно змінювати фази світлофора з урахуванням часових обмежень та пріоритетності спецтранспорту, а також надавати інтерфейс для моніторингу та ручного керування через REST API, що дозволить підвищити ефективність використання дорожньої інфраструктури та зменшити затримки транспортних потоків у міських умовах.

Основна частина

Архітектура системи побудована за конвеєрним принципом: відеопотік (файл, IP-камера, RTSP) через OpenCV надходить до детектора YOLO, який виявляє об'єкти та їхні координати. Далі модуль аналізу трафіку за допомогою геометричних тестів відносить об'єкти до зон "північ-південь", "схід-захід" або пішохідної зони, формуючи метрики завантаженості. Модуль прийняття рішень на основі часових обмежень та порівняння черг приймає рішення про зміну фаз, а контролер світлофора реалізує скінченний автомат із шести станів, забезпечуючи потокобезпечний доступ через блокування. Паралельно REST API через спільний об'єкт стану надає доступ до стану та метрик, дозволяючи моніторинг і ручне керування.

```
def run_processing_loop(runtime, source, model_path, show, output_path):
    detector = YoloDetector(model_path)
    cap = cv2.VideoCapture(source)
    cfg = build_default_intersection_config(width, height)
    analyzer = TrafficAnalyzer(cfg)
    decision_engine = DecisionEngine(runtime.controller)

    while True:
        ok, frame = cap.read()
        detections = detector.detect(frame)
        metrics = analyzer.analyze(detections, fps)
        decision_engine.update(metrics)

        draw_polygon(frame, ...)
        draw_detection(frame, ...)
        draw_hud(frame, runtime.state)

        with runtime.lock:
            runtime.last_frame = frame.copy()
            runtime.last_metrics = metrics
```

Рис. 1. Фрагмент коду основного циклу run_processing_loop

Python обрано завдяки багатій екосистемі бібліотек для машинного навчання та комп'ютерного зору, низькому порогу входження та динамічній типізації, що пришвидшує розробку прототипу. Бібліотека Ultralytics YOLO (модель YOLOv8n) забезпечує оптимальне співвідношення швидкості та точності для виявлення транспорту та пішоходів у реальному часі. OpenCV виконує захоплення відео з різних джерел, геометричні перевірки та візуалізацію (зони, рамки виявлених об'єктів, інформаційне табло). FastAPI разом із сервером Uvicorn створюють асинхронний REST API з автоматичною документацією. Модуль роботи з потоками забезпечує паралельний запуск відеообробки та API, а механізм блокувань — потокобезпечну синхронізацію спільного стану [9].

Конфігурація зон реалізована через класи-структури: головний клас перехрестя об'єднує зони «північ-південь», «схід-захід» та пішохідну, кожна з яких має назву та багатокутник координат. Базова конфігурація генерується пропорційно до розмірів кадру (вертикальний та горизонтальний коридори 35–65%). Для точного налаштування передбачено експорт у JSON із можливістю ручного редагування. Належність об'єкта до зони визначається геометричною функцією OpenCV, яка перевіряє потрапляння центру рамки об'єкта в багатокутник, що забезпечує високу продуктивність та однозначність.

Логіка керування реалізована через скінченний автомат із шести циклічних фаз (зелений-жовтий-червоний для двох напрямків), що керується контролером з підтримкою ручного режиму та блокуваннями для потокобезпечного доступу (рис. 2).

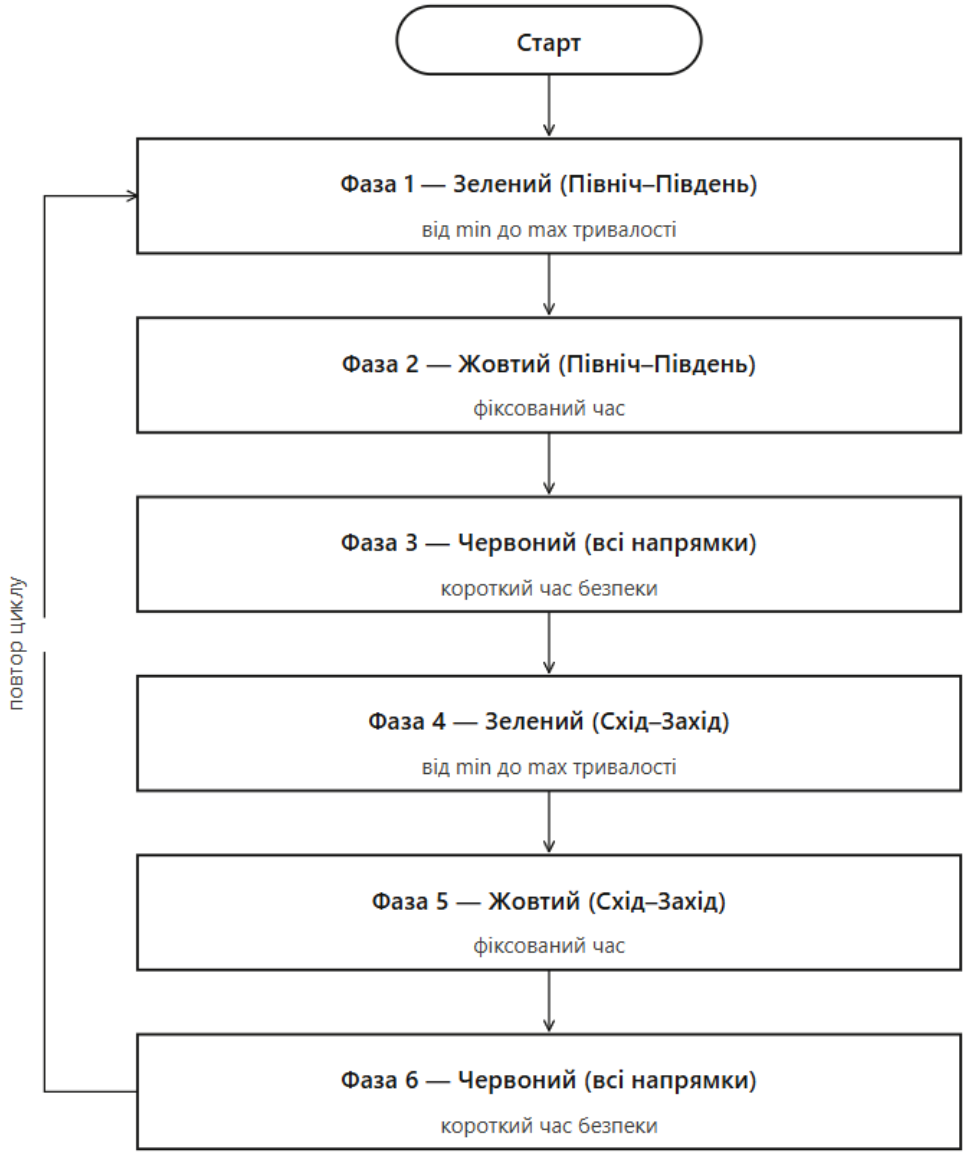

Рис. 2. Схема логіки керування (скінченний автомат)

Модуль прийняття рішень — центральний елемент адаптивної логіки. Він реалізує гібридний підхід, поєднуючи часові обмеження з динамічним аналізом ситуації. Ключові параметри (мінімальна/максимальна тривалість зеленої фази, тривалість жовтого та час повного червоного) створюють «безпечну оболонку»: жодна фаза не триває менше мінімуму (для безпеки водіїв) і не перевищує максимуму, щоб уникнути надмірного очікування іншого напрямку.

Коли система перебуває в автоматичному режимі та активована зелена фаза для певного напрямку, модуль постійно оцінює доцільність її продовження. Після завершення мінімального часу горіння зеленого активується логіка порівняння завантаженості протилежних напрямків. Модуль обчислює різницю між чергою на очікуючому напрямку та поточному. Якщо черга на східно-західному напрямку стає суттєво більшою (наприклад, перевищує на два транспортні засоби), система ініціює перемикання, переходячи до жовтої фази. Такий підхід дозволяє динамічно реагувати на зміну трафіку, надаючи перевагу більш завантаженим напрямкам.

```
class DecisionEngine:
    def __init__(self, controller, min_green_sec=12, max_green_sec=45,
                 yellow_sec=3, all_red_sec=2, queue_delta_threshold=2):
        self.controller = controller
        self.min_green_sec = min_green_sec
        self.max_green_sec = max_green_sec
        self.yellow_sec = yellow_sec
        self.all_red_sec = all_red_sec
        self.queue_delta_threshold = queue_delta_threshold

    def update(self, metrics):
        state = self.controller.state
        state.metrics = metrics

        if state.mode == "MANUAL" and state.forced_phase is not None:
            return

        phase = state.current_phase
        elapsed = state.phase_elapsed()

        # Пріоритетний транспорт
        if metrics.priority_vehicle_count > 0:
            if metrics.ns_vehicle_count >= metrics.ew_vehicle_count:
                self.controller.set_phase(Phase.NS_GREEN, "priority vehicle bias to NS")
            else:
                self.controller.set_phase(Phase.EW_GREEN, "priority vehicle bias to EW")
            return

        if phase == Phase.NS_GREEN:
            if elapsed < self.min_green_sec:
                return
            if elapsed >= self.max_green_sec:
                self.controller.set_phase(Phase.NS_YELLOW, "NS max green timeout")
                return
            if metrics.ew_queue_score - metrics.ns_queue_score >= self.queue_delta_threshold:
                self.controller.set_phase(Phase.NS_YELLOW, "EW demand higher than NS")
                return
        # ... аналогічно для інших фаз
```

Рис. 3. Фрагмент коду модуля прийняття рішень DecisionEngine

Особливий пріоритет у логіці прийняття рішень мають транспортні засоби екстрених служб. Якщо модуль аналізу трафіку фіксує наявність такого транспорту, стандартна логіка порівняння черг тимчасово ігнорується на користь якнайшвидшого звільнення шляху. Модуль аналізує, на якому напрямку знаходиться пріоритетний транспорт: якщо більше таких машин на північ-південь, система негайно перемикається (або залишає) зелений для цього напрямку, якщо на схід-захід - відповідно для нього. Це моделює поведінку реальних систем "зеленої хвилі" для спецтранспорту.

Усі рішення модуля супроводжуються фіксацією причини перемикання. Це забезпечує прозорість роботи системи: оператор завжди може побачити, чи змінилася фаза через досягнення максимального часу, через перевагу іншого напрямку чи через пріоритетний транспорт. Така діагностична інформація виводиться на інформаційне табло при візуалізації та доступна через API, що критично важливо для налагодження та вдосконалення логіки роботи світлофора в реальних умовах.

Перша група ендпоїнтів (запит перевірки працездатності) забезпечує базовий моніторинг працездатності системи та доступу до поточного стану світлофора. Ендпоїнт перевірки стану виконує роль "пинг-запиту", повертаючи простий об'єкт із статусом "ok", прапорцем, який вказує, чи працює цикл обробки відео, та часовою міткою. Це дозволяє зовнішнім системам моніторингу швидко перевірити, чи функціонує сервіс належним чином. Ендпоїнт отримання стану надає детальнішу інформацію про поточне керування: значення активної фази світлофора, режим роботи (автоматичний або ручний), інформацію про примусово встановлену фазу, якщо система в ручному режимі, час, що минув з початку поточної фази, та причину останнього перемикання. Цей ендпоїнт є основним джерелом інформації для панелі керування оператора.

Для отримання аналітичних даних про дорожню ситуацію призначений ендпоїнт отримання показників. Він повертає повний набір метрик, зібраних модулем аналізу трафіку. Відповідь включає кількість транспортних засобів на кожному напрямку, кількість пішоходів, наявність пріоритетного транспорту, показники черги, середню впевненість виявлення об'єктів та поточну швидкість обробки відео. Це забезпечує зручний доступ до даних для зовнішніх систем аналітики.

Третя група ендпоїнтів реалізує функцію ручного керування світлофором, що критично важливо для тестування та екстрених ситуацій. Запит на примусове перемикання дозволяє оператору встановити будь-яку з шести фаз світлофора. При отриманні такого запиту система переводиться в ручний режим, запам'ятовує примусову фазу та негайно виконує перемикання. Якщо передано некоректне значення фази, API повертає помилку з детальним поясненням. Запит на повернення в автоматичний режим виконує зворотну дію — очищає значення примусової фази та дозволяє модулю прийняття рішень знову працювати на основі аналітики трафіку. Така комбінація автоматичного та ручного режимів забезпечує гнучкість керування та безпеку експлуатації системи.

Ключовою технічною проблемою при розробці системи стала необхідність організувати безпечний обмін даними між двома паралельними потоками: основним циклом обробки відео та потоками веб-сервера, які обслуговують HTTP-запити. Вирішенням цієї проблеми став спеціалізований об'єкт, який виступає єдиним джерелом правди для всієї системи [10]. Цей об'єкт містить посилання на поточний стан світлофора, контролер, а також зберігає останній оброблений кадр та метрики. Для забезпечення захисту від одночасного доступу кожне звернення до цих даних захищене блокуванням, що гарантує узгодженість інформації при одночасному читанні з API та записі з циклу обробки відео.

Механізм взаємодії побудований за принципом "один пише — багато читають". Потік обробки відео на кожній ітерації оновлює спільні дані, захоплюючи блокування лише на час копіювання кадру та метрик у відповідні поля. Це мінімізує час блокування, дозволяючи відеопотоку працювати з максимальною швидкістю. У свою чергу, обробники запитів веб-сервера при отриманні запиту звертаються до спільного об'єкта, через його блокування отримують доступ до актуальних даних і повертають їх клієнту. Така архітектура повністю розділяє відповідальність: відеопотік турбується лише про актуальність даних, а API — про їх оперативне надання, при цьому обидва компоненти працюють незалежно та не блокують один одного.

Візуальний інтерфейс системи відіграє критичну роль як для налагодження, так і для демонстрації роботи прототипу, перетворюючи сирі дані виявлення об'єктів на інтуїтивно зрозумілу картину дорожньої ситуації. За допомогою графічних функцій на кожен кадр накладаються багатокутники зон інтересу з чітким кольоровим кодуванням: північно-південний напрямок позначається синім контуром, східно-західний — червоним, а пішохідна зона — жовтим.

Така кольорова диференціація дозволяє оператору миттєво орієнтуватися в просторовому розподілі об'єктів. Кожен виявлений об'єкт супроводжується зеленою рамкою з підписом класу

```
class TrafficController:
    def __init__(self, state: TrafficState):
        self.state = state
        self.lock = threading.Lock()

    def set_phase(self, phase: Phase, reason: str) -> None:
        with self.lock:
            if self.state.current_phase != phase:
                self.state.current_phase = phase
                self.state.phase_started_at = time.time()
                self.state.last_decision_reason = reason

    def force_phase(self, phase: Phase) -> None:
        with self.lock:
            self.state.mode = "MANUAL"
            self.state.forced_phase = phase
            self.set_phase(phase, "forced by API")

class SharedRuntime:
    def __init__(self, state: TrafficState, controller: TrafficController):
        self.state = state
        self.controller = controller
        self.lock = threading.Lock()

# У головному циклі:
with runtime.lock:
    runtime.last_frame = frame.copy()
    runtime.last_metrics = metrics
```

Рис. 4. Фрагмент коду класів TrafficController та SharedRuntime

(наприклад, "автомобіль", "людина") та значенням впевненості виявлення у відсотках, що дає змогу візуально оцінити якість роботи моделі в реальному часі.

Найбільш інформативним елементом інтерфейсу є динамічне табло у верхньому лівому кутку екрану, яке відображає поточну фазу світлофора, режим керування (автоматичний/ручний), кількість транспортних засобів на кожному напрямку, пішоходів та пріоритетного транспорту. Особливу цінність мають технічні показники, такі як, частота кадрів та причина останнього перемикання фази, що дозволяє оператору не лише бачити ситуацію, а й розуміти логіку рішень системи.

Перед запуском необхідно встановити бібліотеки для виявлення об’єктів, роботи з відео, візуалізації, REST API та математичних обчислень, бажано у віртуальному середовищі. Система підтримує різні сценарії через параметри командного рядка: обробку відеофайлу з API, роботу з веб-камерою для налагодження, потоковий протокол для IP-камер та збереження обробленого відео. Для точного налаштування зон виявлення передбачено генерацію конфігураційного файлу з координатами багатокутників, який можна відредагувати вручну під геометрію конкретного перехрестя [11].

Основними обмеженнями прототипу є підтримка лише одного перехрестя з однією камерою, відсутність алгоритмів відстеження об’єктів (що спричиняє подвійний підрахунок), а також відсутність бази даних та інтеграції з реальним обладнанням, система працює з емулятором контролера. Подальший розвиток передбачає інтеграцію з промисловими контролерами, впровадження трекінгу об’єктів, використання багатьох камер, додавання бази даних для історичних показників, розмежування прав доступу до API та забезпечення надійності з резервуванням і безпечним режимом роботи.

Нарешті, промислова версія має відповідати високим стандартам надійності. Це передбачає створення резервних копій критичних компонентів, автоматичне перезавантаження при збоях, механізми поступового зниження функціональності при втраті відеопотоку та "безпечний режим" роботи світлофора з фіксованим циклом у разі повної відмови системи штучного інтелекту [13], [14]. Важливо також реалізувати розподілену архітектуру, де обробка відео може виконуватися на периферійних пристроях, а центральний сервер відповідає за координацію та збір аналітики. Такий підхід забезпечить живучість системи навіть при втраті зв'язку з окремими компонентами.

Висновки

У результаті роботи створено функціональний прототип (MVP) адаптивної системи керування світлофором на основі комп'ютерного зору та глибокого навчання. Прототип успішно виконує весь цикл завдань: захоплення відеопотоку, детекцію об'єктів засобами YOLOv8n, аналіз завантаженості напрямків і прийняття рішень про зміну фаз у реальному часі. Система підтверджує ключову концепцію: нейромережева детекція забезпечує достатню точність для ухвалення осмислених керуючих рішень, включно з пріоритетним пропуском екстрених служб. Моніторинг здійснюється через REST API та візуалізацію з HUD-дисплеєм.

Архітектура забезпечує надійну основу для подальшого розвитку завдяки чіткому розмежуванню відповідальності між модулями, потокобезпечній взаємодії компонентів та гнучкій системі конфігурації. Гібридна логіка прийняття рішень, що поєднує часові обмеження з динамічним аналізом завантаженості та пріоритетів, демонструє правильний підхід до побудови адаптивних алгоритмів, де безпека поєднується з ефективністю.

Попри статус прототипу, продукт уже придатний для дослідницьких цілей, навчання та демонстрації замовникам. Визначені обмеження формують чіткий план розвитку, а запропонована дорожня карта окреслює реалістичний шлях перетворення прототипу на промислове рішення для керування транспортними потоками у розумних містах.

Внесок авторів

Андрій БОНДАРЧУК – наукове керівництво, формування загальної концепції дослідження, визначення архітектури інтелектуальної системи, аналіз сучасних підходів до адаптивного керування світлофорами, постановка експерименту, узагальнення результатів, підготовка висновків та наукове редагування статті; Андрій СТРАЖНІКОВ – розробка програмної реалізації прототипу, створення модулів детекції об’єктів, просторового аналізу та логіки скінченного автомата, проведення обчислювальних експериментів, підготовка первинного варіанту рукопису; Олександр ПРОНЬКІН – реалізація REST API та механізмів потокобезпечної синхронізації, розробка конфігураційних JSON-файлів для зон перехрестя, візуалізація результатів, участь у налагодженні системи та підготовці ілюстративного матеріалу; Микола ЛИСЕНКО – аналіз ефективності запропонованих рішень, порівняння з аналогами, документування обмежень прототипу, формування дорожньої карти подальшого розвитку, підготовка списку літератури та технічне доопрацювання тексту статті.

Декларація про штучний інтелект

Автори не використовували штучний інтелект при створенні матеріалів статті.

Конфлікт інтересів

Автори заявляють про відсутність конфлікту інтересів та підтверджують, що під час підготовки цієї роботи не існувало жодних комерційних, фінансових чи інших взаємовідносин, які могли б бути розцінені як такі, що здатні вплинути на результати дослідження або їх інтерпретацію. Робота виконана відповідно до принципів академічної доброчесності, етичних норм проведення наукових досліджень та вимог редакційної політики щодо запобігання конфлікту інтересів.

Список використаної літератури

1. *Супрун, О., Огірко, О., & Кримська, А. (2025). Розроблення гібридних нейромережевих моделей для прогнозування в інтелектуальних системах. Вісник Херсонського національного технічного університету, 2(1 (92)), 222-229.* https://doi.org/10.35546/kntu2078-4481.2025.1.2.30

2. *Алексіна, Л. Т., & Бондарчук, А. П. (2024). Оптимізація гіперпараметрів для машинного навчання. Зв'язок, (2), 18-22.* DOI: 10.31673/2412-9070.2024.021822

3. *Yao, J., Li, J., Xu, X., Tan, C., Yap, K. H., & Su, R. (2025). Incorporating vision-based artificial intelligence and large language model for smart traffic light control. Applied Soft Computing, 179, 113333. https://doi.org/10.1016/j.asoc.2025.113333*

4. *Majeed, A., Naeem, S., Saeed, E., & Al-Shanoon, A. (2025). Real-Time Adaptive Traffic Signal Control with YOLOv10 and Image Processing. Al-Khwarizmi Engineering Journal, 21(4), 65-81. doi:* 10.22153/kej.2025.09.006.

5. *Medvei, M. M., Bordei, A. V., Niță, Ș. L., & Țăpuș, N. (2025). DeepSIGNAL-ITS—Deep Learning Signal Intelligence for Adaptive Traffic Signal Control in Intelligent Transportation Systems. Applied Sciences, 15(17), 9396.* https://doi.org/10.3390/app15179396

6. *Harefa, M. T., & Wijaya, D. R. (2025, August). A Real-Time Multi-Object Tracker Based on Hybrid Integration of ByteTrack and DeepSORT Using YOLOv8. In 2025 IEEE International Conference on Artificial Intelligence for Learning and Optimization (ICoAILO) (pp. 244-250). IEEE.* DOI: 10.1109/ICoAILO66760.2025.11155979

7. *Kadam, S. S., Jadhav, T., Patil, L., Saw, P., & Landage, A. (2024, June). Crowd counting using yolov8 and various tracking algorithms. In 2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0 (pp. 1-9). IEEE.* DOI: 10.1109/OTCON60325.2024.10688023

8. *Cao, Y., Li, S., Liu, Y., Yan, Z., Dai, Y., Yu, P. S., & Sun, L. (2023). A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt. arXiv preprint arXiv:2303.04226.* https://doi.org/10.48550/arXiv.2303.04226

9. *Shantyr, A., Zinchenko, O., Storchak, K., Bondarchuk, A., & Pepa, Y. (2025). Prediction of quality software quality indicators with applied modifications of integrated gradiates methods. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska, 15(2), 139-146.*

10. *Стрельніков, В. І., & Бондарчук, А. П. (2025). Комплексний підхід до інтелектуального управління, моделювання та виявлення мережних аномалій на основі ентропійних та нейромережевих підходів. Телекомунікаційні та інформаційні технології, (2), 100-107.* DOI 10.31673/2412-4338.2025.025703

11. *Довженко, Т. П., & Бондарчук, А. П. (2025). Аналіз сучасного етапу розвитку штучного інтелекту в телекомунікаціях. Зв'язок, (1), 43-48.* DOI: 10.31673/2412-9070.2025.019552

12. *Вайшле, В. О. (2025). Автоматизована система регулювання дорожнього руху (Doctoral dissertation, Тернопіль, ЗУНУ). DOI:10.15407/emodel.47.01.040.*

13. *Устименко, А. В. (2024). Стійкий розвиток транспортної інфраструктури в контексті забезпечення досягнення цілей сталого розвитку України. Науковий вісник Ужгородського національного університету. 3(85),156-161.* https://doi.org/10.24144/2307-3322.2024.85.3.24

14. *Славич, В. П., & Савченко, М. О. (2024). Модель управління параметрами світлофорної сигналізації в залежності від встановленої пропускної здатності. Вісник Херсонського національного технічного університету, (3 (90)), 118-122.* https://doi.org/10.35546/kntu2078-4481.2024.3.15

A. Bondarchuk, A. Strazhnikov, O. Pronkin, M. Lysenko

INTELLIGENT TRAFFIC LIGHT CONTROL SYSTEM BASED ON COMPUTER VISION

The article presents the development and experimental verification of a prototype of an intelligent traffic light control system based on computer vision and deep learning. The inefficiency of traditional fixed-cycle traffic lights causes chronic congestion, leads to losses of time and fuel, and hinders

the passage of priority vehicles (ambulances, fire trucks, emergency services). The proposed system addresses this problem through adaptive phase control based on real-time video stream analysis.

The architecture is implemented in Python using Ultralytics YOLOv8n for object detection, OpenCV for video capture and visualization, FastAPI and Uvicorn for an asynchronous REST API with automatic documentation. The spatial analysis module relies on polygonal zones ("north–south", "east–west", pedestrian), whose configuration is stored in JSON format according to the actual geometry of the intersection. The control logic is implemented as a finite state machine with six phases: green and yellow for the "north–south" direction, red for all directions, green and yellow for the "east–west" direction, red for all directions. The decision module combines time constraints (minimum and maximum green phase duration) with dynamic comparison of directional loads and ensures priority passage for emergency vehicles with immediate phase switching. All decisions are recorded with an explanation of the reason for switching. The thread-safe architecture synchronizes access to shared data between the video processing loop and the web server. The REST API provides endpoints for system health monitoring, retrieval of metrics (vehicle count, pedestrians, priority transport, FPS) and manual phase control. An analysis of related research on YOLO, LLM agents and reinforcement learning was conducted, confirming the alignment of the proposed solutions with global trends. Prototype limitations are identified (single intersection, no object tracking, controller emulator) and a development roadmap is outlined: integration with industrial controllers, object tracking, multi-camera support, metrics database storage and redundancy in case of AI system failure. The prototype can serve as a foundation for industrial traffic flow management systems in smart cities.

Keywords: AI, adaptive traffic light control, computer vision, deep learning, object detection, information system, management, video analytics.

Надійшла до редакції: 24.04.2026
Прийнята до друку: 13.06.2026
Опубліковано: 29.06.2026