Хмельницький національний університет
Факультет програмування та комп`ютерних і телекомунікаційних систем
Кафедра комп'ютерних наук та інформаційних технологій

ДИПЛОМНА РОБОТА

МАГІСТРА

на тему *Інформаційна система розумного світлофора для регулювання дорожнього трафіку*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Виконав: студент *2* курсу, група *КНм-19-1* _____ *Ю.Б. Михайляк*
Підпис Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КНІТ* _____ *О.А. Пасічник*
Підпис Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КНІТ* _____ *Р.О. Багрій*
Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КНІТ, д.т.н., професор _____ *О.В. Бармак*
Підпис Ініціали, прізвище
7 13 2020 р.

Хмельницький 2020

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *програмування та комп'ютерних і телекомунікаційних систем*
Кафедра *комп'ютерних наук та інформаційних технологій*
Освітній ступінь *магістр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук та інформаційних технологій*

(підпис)
д.т.н., професор О.В. Бармак
«7» 9 *2020* року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ МАГІСТРА

1. Тема дипломної роботи магістра: *«Інформаційна система розумного світлофора для регулювання дорожнього трафіку»*

2. Завдання видано студенту *Михайляку Юрію Богдановичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *к.т.н., доцент Пасічник Олександр Анатолійович*
(прізвище, ім'я, по батькові)

4. Затверджені наказом університету від «5» 9 202 р. № 12

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – розробка інформаційної системи розумного світлофора для регулювання дорожнього трафіку з використанням мультиканального режиму прийому даних, удосконалити існуючі методи регулювання дорожнього трафіку з використанням розумного світлофора у напрямку полегшення їх інтегрування в наявну транспортну інфраструктуру

Реферат

Дипломна робота магістра присвячена розробці інформаційної системи регулювання дорожнього трафіку за допомогою системи розумного світлофора.

Актуальність теми. Розробка нових інформаційних технологій дозволяє розвивати суспільство в тих чи інших напрямках. Вони активно перетворюють інші технології матеріального і нематеріального виробництва, в кінцевому підсумку формуючи новий стиль роботи, спосіб життя в цілому. Суть інформаційних технологій становлять методи і засоби формування та підтримки інформаційних потоків у системах управління об'єктами. Прогрес в світі технологій відбувається задля вдосконалення людського життя. Оптимізуючи певні речі можливо значно зекономити час, що є натепер найважливішим ресурсом. Задля мобільності люди почали користуватись автомобілями, сьогодні автомобіль це не елемент розкоші, а потреба. В деяких містах немає хороших автобусних чи залізничних сполучень, тому задля якогось пересування люди користуються автівками.

Автомобільні затори - це те, що забирає багато часу, крім того вони шкодять навколишньому середовищу, бо відбувається більший рівень викидів чадного газу з двигунів внутрішнього згоряння. Час перебування в заторах не може бути в будь-якій мірі ефективним, ви не можете відірватись від дорожнього обстановки, оскільки є ризик потрапити в ДТП.

Впровадження інформаційних технологій дозволяє підняти на якісно новий рівень автоматизацію тих процесів, які ще кілька років тому вважалися верхівкою і зразком в галузі автоматизації.

Сфера менеджменту дорожнього трафіку не виключення, завдяки інформаційним технологіям з'являється можливість до реалізації нових ідей, які змінюють всі процеси, які були до цього запроваджені в тій чи іншій галузі.

Світлофорні системи на перехрестях Україні працюють за найпростішим принципом пофазної дії первинного налаштування, не враховуючи вхідні дані, як завантаження доріг на момент часу, що призводить до суттєвої інерційності

процесу керування і відсутньої можливості оперативного реагування в режимі реального часу на дорожню ситуацію.

Завдяки сучасним технологіям можна змінити дану ситуацію, а саме оперативно регулювати дорожній рух спираючись на вхідні дані поточної завантаженості доріг. Це дозволить ефективно використовувати час учасників дорожнього руху і покращити екологічну ситуації в містах.

Мета і задачі роботи. Мета роботи полягає у реалізація інформаційної системи розумного світлофора для регулювання дорожнього трафіку з використанням мультиканального режиму прийому даних.

Для досягнення поставленої мети визначенні наступні задачі-дослідження:

- провести аналіз існуючих методів, технологій та рішень регулювання дорожнього трафіку;
- удосконалити існуючі методи регулювання дорожнього трафіку з використанням розумного світлофора у напрямку полегшення їх інтегрування в наявну транспортну інфраструктуру;
- розробити інформаційну систему розумного світлофора для регулювання дорожнього трафіку за допомогою отриманих моделей та методів;
- виконати експериментальну перевірку інформаційної системи регулювання дорожнього трафіку.

Об’єкт дослідження – процес регулювання дорожнього трафіку з використанням інформаційних технологій.

Предмет дослідження – моделі, методи, підходи та засоби інформаційної технології регулювання дорожнього трафіку.

Методи дослідження. Для розв’язання поставлених задач використовуються основні положення методів аналізу даних, дискретної математики; для вдосконалення аналізу дорожньої обстановки методи нейронних мереж, комп’ютерного зору; для реалізації інформаційної технології

методології проектування інформаційних систем та об’єктно орієнтований підхід.

Наукова новизна одержаних результатів. В результаті проведеної роботи були отримані наступні результати:

- Удосконалено існуючий метод регулювання дорожнього трафіку на перехрестях за допомогою розумного світлофора в мультиканальному режимі прийому даних.
- Досліджено ефективність використання часткового і повного режиму ідентифікації дорожнього трафіку,
- Розроблений алгоритм ручного керування світлофорною ділянкою поверх автоматичного режиму.

Практичне значення одержаних результатів. В результаті виконання дипломної роботи магістра розроблено відповідне експериментальне програмне забезпечення, яке підтвердило вірність запропонованих положень. Застосування інформаційної системи дає можливість здійснювати регулювання дорожнього руху в режимі реального часу, практично миттєво реагуючи на зміни дорожнього трафіку, а її інтеграція в наявну транспортну інфраструктуру не потребує її суттєвої модернізації. Наведено рекомендації щодо режиму роботи інформаційної системи.

Апробація результатів дипломної роботи магістра та публікації. Основні наукові та практичні результати опубліковані в наукових виданнях МОН України:

- публікація на тему “Інформаційна система розумного світлофора для регулювання дорожнього трафіку” в науковому журналі “Вісник Хмельницького національного університету”.

За темою дипломної роботи магістра автором виконано одну наукову публікацію[35].

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 35 найменувань та 3 додатків. Загальний обсяг дипломної

роботи магістра становить 94 сторінок, з них 68 сторінок основного тексту та 17 сторінок додатків. У роботі наведено 31 рисунків.

Ключові слова: інформаційна система, регулювання трафіку, розумний світлофор, мультикананний режим.

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
НМ	Нейронна мережа
ПЗ	Програмне забезпечення
ДТП	Дорожньо-транспортна пригода
ТЗ	Транспортний засіб

Вступ

Актуальність теми. Розробка нових інформаційних технологій дозволяє розвивати суспільство в тих чи інших напрямках. Вони активно перетворюють інші технології матеріального і нематеріального виробництва, в кінцевому підсумку формуючи новий стиль роботи, спосіб життя в цілому. Суть інформаційних технологій становлять методи і засоби формування та підтримки інформаційних потоків у системах управління об’єктами. Прогрес в світі технологій відбувається задля вдосконалення людського життя. Оптимізуючи певні речі можливо значно зекономити час, що є натепер найважливішим ресурсом. Задля мобільності люди почали користуватись автомобілями, сьогодні автомобіль це не елемент розкоші, а потреба. В деяких містах немає хороших автобусних чи залізничних сполучень, тому задля якогось пересування люди користуються автівками.

Автомобільні затори - це те, що забирає багато часу, крім того вони шкодять навколишньому середовищу, бо відбувається більший рівень викидів чадного газу з двигунів внутрішнього згоряння. Час перебування в заторах не може бути в будь-якій мірі ефективним, ви не можете відірватись від дорожнього обстановки, оскільки є ризик потрапити в ДТП.

Впровадження інформаційних технологій дозволяє підняти на якісно новий рівень автоматизацію тих процесів, які ще кілька років тому вважалися верхівкою і зразком в галузі автоматизації.

Сфера менеджменту дорожнього трафіку не виключення, завдяки інформаційним технологіям з’являється можливість до реалізації нових ідей, які змінюють всі процеси, які були до цього запроваджені в тій чи іншій галузі.

Світлофорні системи на перехрестях Україні працюють за найпростішим принципом пофазної дії первинного налаштування, не враховуючи вхідні дані, як завантаження доріг на момент часу, що призводить до суттєвої інерційності

процесу керування і відсутньої можливості оперативного реагування в режимі реального часу на дорожню ситуацію.

Завдяки сучасним технологіям можна змінити дану ситуацію, а саме оперативно регулювати дорожній рух спираючись на вхідні дані поточної завантаженості доріг. Це дозволить ефективно використовувати час учасників дорожнього руху і покращити екологічну ситуації в містах.

Мета і задачі роботи. Мета роботи полягає у реалізація інформаційної системи розумного світлофора для регулювання дорожнього трафіку з використанням мультиканального режиму прийому даних.

Для досягнення поставленої мети визначенні наступні задачі-дослідження:

- провести аналіз існуючих методів, технологій та рішень регулювання дорожнього трафіку;
- удосконалити існуючі методи регулювання дорожнього трафіку з використанням розумного світлофора у напрямку полегшення їх інтегрування в наявну транспортну інфраструктуру;
- розробити інформаційну систему розумного світлофора для регулювання дорожнього трафіку за допомогою отриманих моделей та методів;
- виконати експериментальну перевірку інформаційної системи регулювання дорожнього трафіку.

Об’єкт дослідження – процес регулювання дорожнього трафіку з використанням інформаційних технологій.

Предмет дослідження – моделі, методи, підходи та засоби інформаційної технології регулювання дорожнього трафіку.

Методи дослідження. Для розв’язання поставлених задач використовуються основні положення методів аналізу даних, дискретної математики; для вдосконалення аналізу дорожньої обстановки методи нейронних мереж, комп’ютерного зору; для реалізації інформаційної технології

методології проектування інформаційних систем та об’єктно орієнтований підхід.

Наукова новизна одержаних результатів. В результаті проведеної роботи були отримані наступні результати:

- Удосконалено існуючий метод регулювання дорожнього трафіку на перехрестях за допомогою розумного світлофора в мультиканальному режимі прийому даних.
- Досліджено ефективність використання часткового і повного режиму ідентифікації дорожнього трафіку,
- Розроблений алгоритм ручного керування світлофорною ділянкою поверх автоматичного режиму.

Практичне значення одержаних результатів. В результаті виконання дипломної роботи магістра розроблено відповідне експериментальне програмне забезпечення, яке підтвердило вірність запропонованих положень. Застосування інформаційної системи дає можливість здійснювати регулювання дорожнього руху в режимі реального часу, практично миттєво реагуючи на зміни дорожнього трафіку, а її інтеграція в наявну транспортну інфраструктуру не потребує її суттєвої модернізації. Наведено рекомендації щодо режиму роботи інформаційної системи.

Апробація результатів дипломної роботи магістра та публікації. Основні наукові та практичні результати опубліковані в наукових виданнях МОН України:

- публікація на тему “Інформаційна система розумного світлофора для регулювання дорожнього трафіку” в науковому журналі “Вісник Хмельницького національного університету”.

За темою дипломної роботи магістра автором виконано одну наукову публікацію[35].

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків,

переліку посилань із 35 найменувань та 2 додатків. Загальний обсяг дипломної роботи магістра становить 85 сторінок, з них 68 сторінок основного тексту та 17 сторінок додатків. У роботі наведено 31 рисунків.

Розділ 1

Аналіз сучасного стану проблеми автоматизації формування тестових завдань

1.1 Аналіз предметної області

Дорожній рух регулюється з допомоги дорожніх знаків, розмітки, дорожнього обладнання, світлофорів, а також регулювальниками вручну. Знаки дорожнього руху мають перевагу над розміткою і можуть бути перманентними чи тимчасовими [1]. Світлофори ж бувають кількох видів [2]:

1. Вуличні і дорожні світлофори.
    1.1. Автомобільні світлофори.
    1.2. Стрілки і стрілочні секції.
    1.3. Світлофор із мигаючим червоним сигналом.
    1.4. Світлофори на залізничних переїздах.
    1.5. Реверсійний світлофор.
    1.6. Світлофори для маршрутних транспортних засобів.
    1.7. Світлофори для пішоходів.
    1.8. Світлофори для велосипедистів.
    1.9. Трамвайний світлофор.
2. Залізничний світлофор.
    2.1. Класифікація.
    2.2. Сигнали.
3. Річковий світлофор.
4. Світлофори у автоспорті.

Найпоширеніші світлофори з сигналами (зазвичай круглими) трьох кольорів: червоного, жовтого і зеленого. У деяких країнах замість жовтого використовується помаранчевий колір. Сигнали можуть бути розташовані як вертикально (при цьому червоний сигнал завжди розташовується зверху, а

зелений — знизу), так і горизонтально (при цьому червоний сигнал завжди розташовується зліва, а зелений — справа). За відсутності інших, спеціальних світлофорів вони регулюють рух всіх видів транспортних засобів і пішоходів. Іноді сигнали світлофора доповнюють спеціальним табло зворотного відліку часу, який показує скільки часу ще горітиме сигнал. Частіше за все табло зворотного відліку роблять для зеленого сигналу світлофора, але у ряді випадків табло відображає і решту часу для червоного сигналу [2].

Практично повсюдно червоний сигнал світлофора забороняє рух, жовтий забороняє виїзд на ділянку, що охороняється світлофором, але допускає завершення його проїзду, а зелений — дозволяє рух. Поширене, але не повсюдне використання поєднання червоного і жовтого сигналів, що позначає майбутнє включення зеленого сигналу. Іноді зелений сигнал включається відразу після червоного без проміжного жовтого, але не навпаки. Деталі застосування сигналів розрізняються залежно від прийнятих в тій або іншій країні Правил дорожнього руху. [2]

Існують світлофори з двох секцій — червоної і зеленої. Такі світлофори зазвичай встановлюються на пунктах, де пропуск автомобілів проводиться в індивідуальному порядку, наприклад, на прикордонних переходах, при в'їзді або виїзді з автостоянки, території, що охороняється тощо [2].

Можуть також подаватися миготливі сигнали, значення яких залежить від місцевого законодавства. В Україні і в багатьох країнах Європи миготливий зелений сигнал означає майбутнє перемикання до жовтого. Автомобілі, що наближаються до світлофора з миготливим зеленим сигналом, можуть прийняти заходи до своєчасного гальмування, аби уникнути виїзду на перехрестя, що охороняється світлофором, або переходу на заборонений сигнал. Миготливий жовтий сигнал вимагає понизити швидкість для проїзду перехрестя або означає на пішохідному переході нерегульований світлофор (наприклад, вночі, коли регулювання не потрібне через низьку інтенсивність руху). Іноді для цих цілей

застосовуються спеціальні світлофори, що складаються з миготливої однієї або поперемінно миготливих двох жовтих секцій [2].

Світлофорний об'єкт – локальна ділянка вулично-дорожньої мережі, на якій послідовність руху конфліктуючих транспортних потоків або транспортних і пішохідних потоків регулюється світлофорною сигналізацією [3] (рис. 1.1).

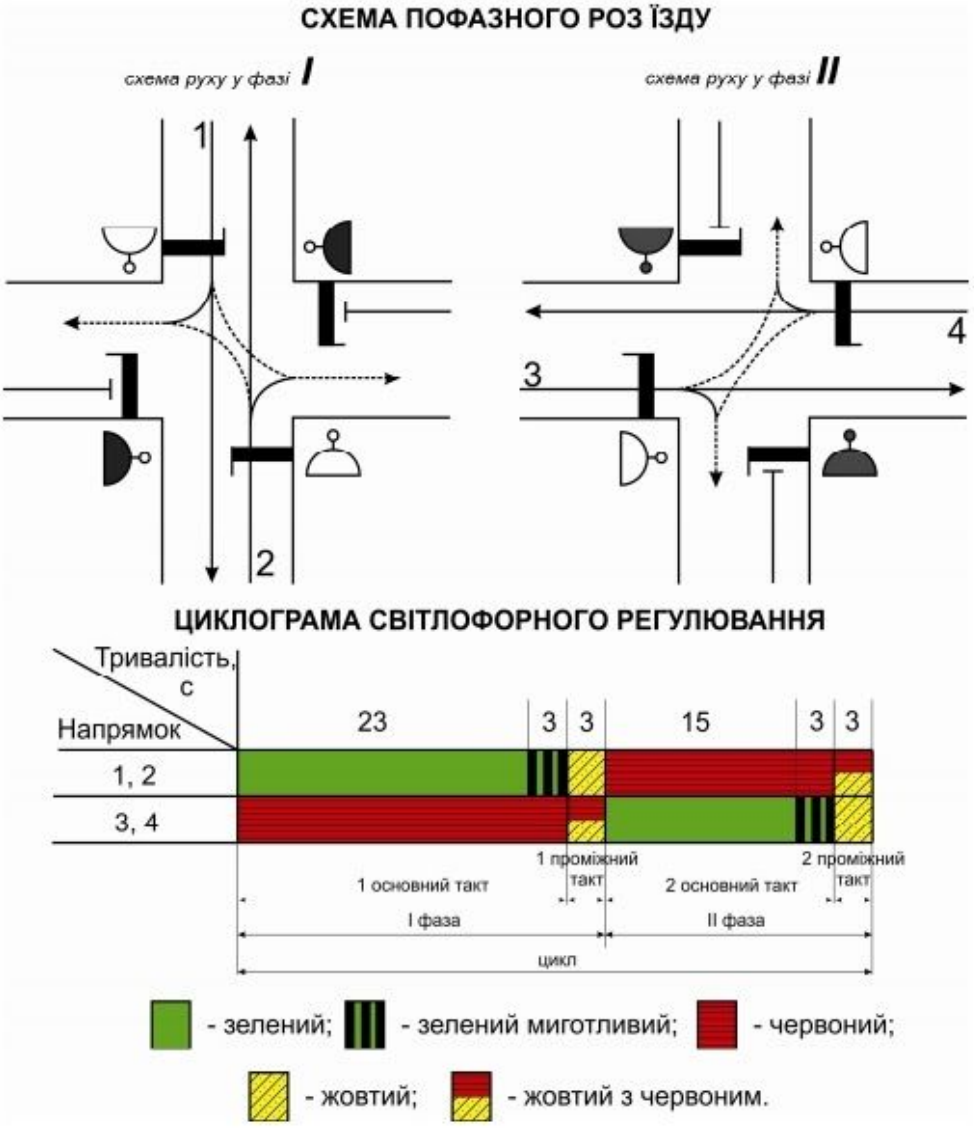


Рисунок 1.1 – Схеми основних термінів режиму регулювання світлофором [3]

Основні поняття авто-світлофора [3]:

- *такт* – період виконання конкретної комбінації індикаторів;
- *головний такт* – хронологічний період, під час котрого пересування транспортних засобів (інколи й пішохідних потоків) є дозволеним;

- *проміжний такт* – хронологічний період, під час котрого відбувається підготовка до перемикання дозволу на пересування наступній групі автомобілів (інколи й пішохідних потоків);
- *фаза регулювання* – сукупність головного такту й проміжного такту;
- *регулювальний цикл* – послідовність фаз яка повторюється з конкретною періодичністю;
- *режим світлофорного регулювання* – к-сть часу вказаних фаз, тактів і тривалість регулювального циклу;
- *схема роз'їзду* – візуальне хронологічне представлення конфліктних автомобільних потоків;
- *циклограма регулювання світлофора* – візуальна схема послідовності й тривалості горіння індикаторів світлофорів на всіх можливих маршрутах, розташованих на вулиці.

Незважаючи, на простоту й надійність механізму світлофорного регулювання, можна побачити значний недолік, а саме - неможливість регулювання трафіку на основі вхідних зовнішніх даних. Під зовнішніми даними мається на увазі такі елементи:

1. Завантаженість доріг.
2. Тип транспорту.

Спираючись на ці фактори можна визначати пріоритетність тої чи іншої частини дороги.

1.2 Аналіз існуючого програмного забезпечення предметної області

На даний час жодних відкритих розробок по системі розумної регуляції руху в Україні не ведеться. Існує кілька ідей на ресурсах для стартапів, які чекають інвестицій і пропонують розумну регуляцію руху на основі завантаженості доріг потоком автомобілів. Також, є опис, що розвиваючи цю ідею можна добавити розумну детекцію правопорушень і виписку штрафів.

Проте, за межами українського інтернет-простору можна знайти кілька схожих проектів, які розташовані на відкритому програмному ресурсі - github.com. Реалізації цих проектів мають свою плюси чи мінуси, як і будь-яке ПО. Перший з таких “Smart-traffic-light-2” [4].

Рисунок 1.2 – Аналог системи розумного світлофора [4]

Проект був розроблений в рамках Бельгійського університету. Плюси цього проекту:

1. Система інтегрована на платформу Raspberry Pi 3 з підключеною USB камерою.
2. Розпізнавання меж дороги, для точнішого обрахунки вхідних даних.
3. Код реалізований на мові Python, що дає легку підтримку і гнучкість в добавленні нових функціональних речей.

4. Можливість надання пріоритетності для спец. засобів (швидка допомога, поліція, тощо.)

Мінуси проекту:

1. Визначення дорожнього транспорту ведеться з камери, яка може розташовуватися лише зверху.
2. Неможливість використовувати по одній камері на одну сторону.
3. Визначення автомобілів ведеться за технологією HOG (Histogram of Oriented Gradients), що може призвести до неочікуваної поведінки під час негоди, тощо.

Ще одним прикладом є схожий проект, який реалізований на C++ “smart-traffic-signals”[5]. Реалізації проекту є доволі низькорівневою, що робить його надзвичайно гнучким. Реалізації проекту є доволі низькорівневою, що робить його надзвичайно гнучким. Плюси проекту:

1. C++ надає велику гнучкість редагування проекту.
2. C++ надає великий приріст в швидкості дії.
3. Інтеграція з RaspberryPi 2.

Мінуси:

1. C++ мова з низьким рівнем абстракції, тому навіть для реалізації банальних речей потрібно більше часу, ніж до прикладу на Python.
2. Визначення автомобілів ведеться за технологією Haar Cascade, що може призвести до неочікуваної поведінки під час негоди, тощо.

Отож, встановлено актуальність розробки програмного забезпечення предметної області з вузькоспеціалізованим функціоналом, що буде реалізовувати можливість регулювання дорожнього руху на основі точнішої системи визначення об’єктів - YOLO.

1.3 Аналіз сучасних засобів створення програмного забезпечення

За основу комп’ютерного зору була обрана OpenCV, яка давно вже являється де факто лідером ринку в цій галузі.

OpenCV — алгоритмічна бібліотека для обробки зображень. Бібліотека дає функціонал для редагування фреймів і аналізування зображень, також розпізнавання і знаходження об’єктів на фотографіях. OpenCV був побудований для забезпечення загальної інфраструктури для програм комп'ютерного зору та прискорення використання машинного сприйняття в комерційних продуктах. Як продукт із ліцензією BSD, OpenCV полегшує бізнесу використання та модифікацію коду. Бібліотека широко використовується у компаніях, дослідницьких групах та державних органах.

На даний момент є кілька варіантів взаємодії з OpenCV бібліотекою, найбільш популярнішими є Python і Java.

Мова Java – це об'єктно-орієнтована мова програмування, яка транслюються не безпосередньо в машинно-залежний код, а в так званий байт-код, виконуваний спеціальним інтерпретатором, віртуальної Java – машиною (Java Virtual Machine, JVM). Така організація роботи Java-програм дозволяє їм переноситися без змін і однаково працювати на різних платформах, якщо на цих платформах є реалізація JVM, відповідна опублікованими специфікаціями віртуальної машини.

Мова Python – це це високорівнева мова програмування. Ця мова вміщує структури даних і простий (без надлишкових речей, як в Java), але потужний підхід до ООП. Синтаксис продуманий, динамічні типи, інтерпретований тип мови - всі ці факти роблять її ідеальним варіантом для ефективної і швидкої розробки програм у будь-яких галузях на левовій частині платформ.

Відмінність Python’а від інших мов програмування:

1. Керування пам'яттю - автоматичне — не потрібно хвилюватися щодо розподілу або звільнення пам'яті. Немає загрози “небезпечного посилання”.
2. Типи зв'язані з об'єктами, а не зі змінними. Це означає, що змінній може бути призначене значення будь-якого типу, і що (наприклад) масив може містити об'єкти різних типів. Традиційні мови не надають такої можливості.
3. Операції звичайно виконуються в більш високому рівні абстракції. Це частково результат того, як написана мова, і частково результат розширеної стандартної бібліотеки кодів, що поставляється разом з Python.

Отже, прийнявши до уваги всі переваги та недоліки даних мов, було прийнято рішення використання мови Python для реалізації програмного продукту.

Висновки до розділу 1

Проведений аналіз літературних джерел засвідчує суттєву та зростаючу потребу у регулюванні дорожнього трафіку, це обумовлено стрімким збільшенням автопарку у поєднанні з відставанням розвитку інфраструктури. Розвиток і запровадження інформаційних технологій, та інформаційних систем створює широкі можливості для застосування нових, більш ефективних підходів до регулювання дорожнього руху, зокрема використання розумних світлофорів. Наявна дорожня інфраструктура є достатньо розгалуженою, тому забезпечення достатньо рівня економічної ефективності впроваджуваних інструментів регулювання руху повинні передбачати мінімальне втручання у вже наявні та розгорнуті системи.

В результаті проведеного аналізу існуючих підходів сформульовані такі завдання дослідження метою якого є реалізація інформаційної системи

розумного світлофора для регулювання дорожнього трафіку з використанням мультиканального режиму прийому даних:

1. Провести аналіз існуючих методів, технологій та рішень регулювання дорожнього трафіку.
2. Удосконалити існуючі методи регулювання дорожнього трафіку з використанням розумного світлофора у напрямку полегшення їх інтегрування в наявну транспортну інфраструктуру.
3. Розробити інформаційну систему розумного світлофора для регулювання дорожнього трафіку за допомогою отриманих моделей та методів.
4. Виконати експериментальну перевірку інформаційної системи регулювання дорожнього трафіку.

Розділ 2

Проектування структури інформаційної системи

2.1 Загальний опис інформаційної системи регулювання дорожнього трафіку

Для мультиканальності потрібно організувати інформаційні потоки в такому вигляді, щоб вони були не пов’язані між собою, але мали спільний вузол, який саме й буде вирішувати наступний стан дорожнього трафіку.

Ізольованість потоків слугує для того, щоб програму можна було легко розширяти під різні типи перехресть.

На рисунку 2.1 показано, що в нас є основний шар “Контролер”, який який отримує дані від екземплярів камер - інформаційних потоків. Спираючись на вхідні дані з потоків контролер вирішує стан світлофорної ділянки.

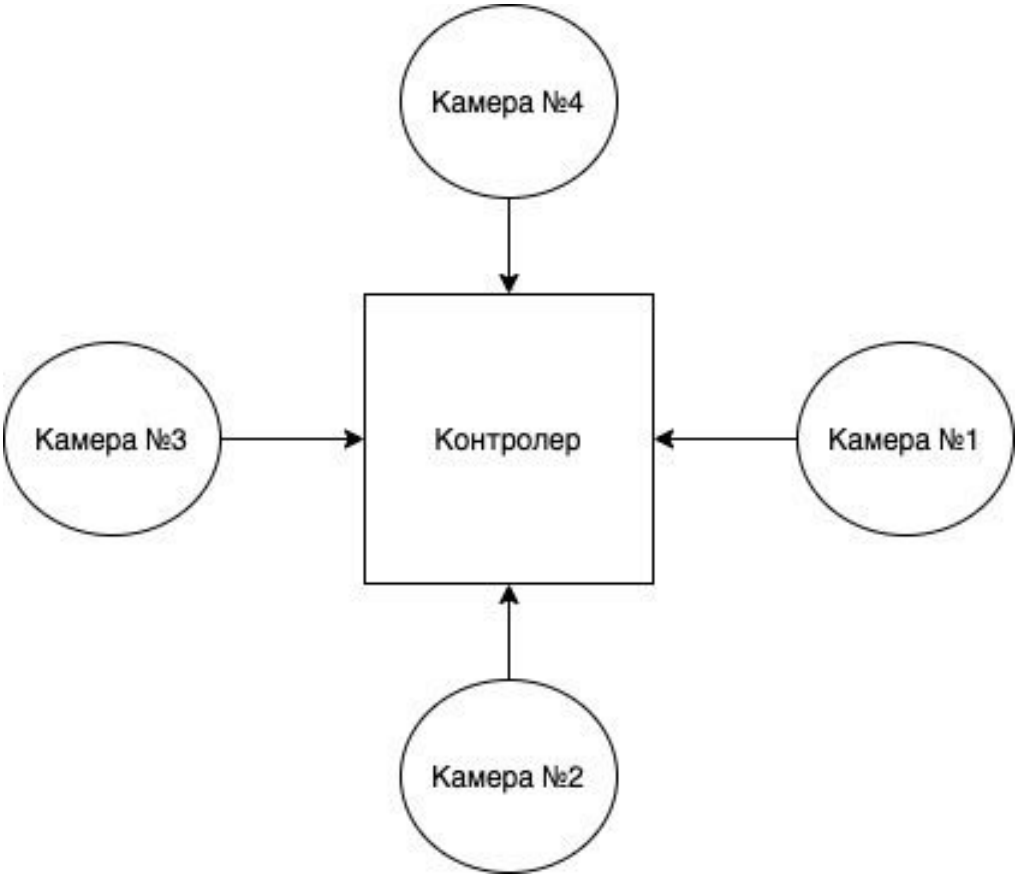


Рисунок 2.1 – Вхідні потоки і контролер

Кожний екземпляр камери це не просто відео-потік, це окрема підпрограма, яка віддає результат на контролер, результатом є кількість транспортних засобів на ділянці для підрахунку автомобілів на кожній осі.

Принцип роботи показаний на рисунку 2.2. Підпрограма отримує відео-потік. Кожен відео-фрейм відправляється на обробку в систему детекції об’єктів, виявивши об’єкти система робиться дві задачі:

1. Обраховує кількість об’єктів на поточній ділянці і відправляє їх.
2. Вимальовує знайдені елементи для візуалізації об’єктів на фреймі.

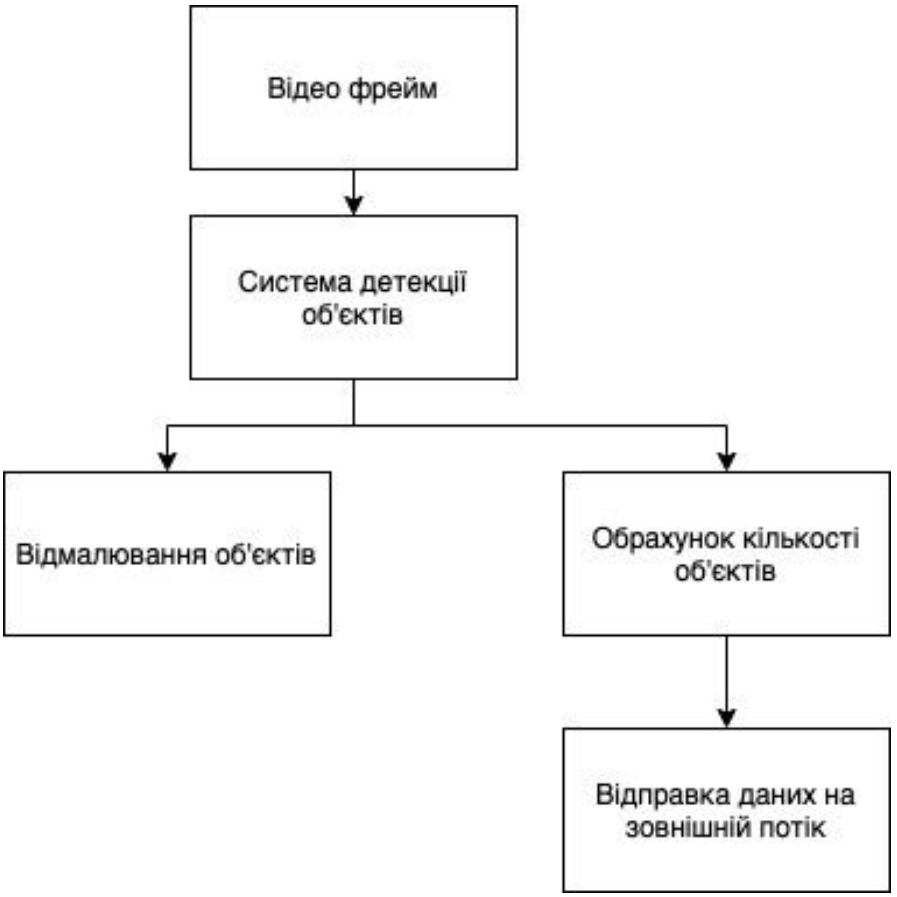


Рисунок 2.2 – Підпрограма камери

Принцип дії ж самої системи детекції описаний на рисунку 2.3. Детектор отримує з оболонки відео-файл і додаткові параметри, які на даному етапі не враховуються. Оболонка це батьківський процес нашого підпроцесу. Після прийому даних з батьківського процесу відбувається ініціалізація самого детектора, на цьому етапі створюється нейронна мережа, після ініціалізації з кожного відео-фрейму створюється blob-об’єкт.

Blob-об’єкт - це бінарний великий об’єкт, сукупність двійкових даних, що зберігаються як єдине ціле в системі управління базами даних. Дані, як правило, це зображення, аудіо чи інші мультимедійні об’єкти.

Пізніше цей blob-об’єкт передається в YOLO.

YOLO - ефективний алгоритм розпізнавання об'єктів, який базується на нейронній мережі, початково це Darknet, що є нейронною мережею з відкритим кодом, написана на мовах C та CUDA. Вона швидка, проста у встановленні та підтримує обчислення процесора та графічного процесора. Зазвичай такі мережі тренуються на даних COCO або Pascal VOC.

Є можливість натренувати нейронну мережу самостійно. Для роботи з YOLO в OpenCV не потрібно доставляти жодних пакетів, оскільки YOLO підтримується OpenCV самостійно. При цьому модель може бути створена в будь-якому з трьох фреймворків глибокого навчання - Caffe, TensorFlow або Torch; спосіб її завантаження і використання зберігається незалежно від того, де вона була створена.

Агрегація - це окремий блок, який створений для того, щоб агрегувати інформацію від окремих шарів з попередніх блоків (як показано на малюнку вище) для збільшення акуратності передбачення.

Dense prediction - це передбачення, яке складається з вектора, що містить координати передбачуваного обмежувального поля (центр, висота, ширина), оцінка достовірності передбачення та мітка.

Sparse prediction - визначають окремо регіони і потім класифікують ці регіони.

Після операцій YOLO відбувається фільтрація об’єктів по потрібних класах, а саме, для нашого завдання нам потрібно дивитися лише на ТЗ, тому система бачить лише автомобілі, автобуси, мотоцикли, вантажівки. Як тільки ми маємо координати об’єктів і клас, ми можемо обвести всі об’єкти на відео.

Місцезнаходження вказується, намалювавши обмежувальне поле навколо об’єкта. Обмежувальне поле може або не може точно визначати положення об'єкта. Можливість знаходити об'єкт всередині зображення визначає ефективність алгоритму, що використовується для виявлення.

Кінцевим етапом є обрахування всіх об’єктів і відправка їх на зовнішній рівень.

Для реалізації мультиканальності потрібно організувати інформаційні потоки в такому вигляді, щоб вони були не пов’язані між собою, але мали спільний вузол, який саме й буде вирішувати наступний стан дорожнього трафіку.

2.2 Розробка додаткових функцій інформаційної системи

Схема доступних функцій:

1. Регулювання трафіку - основна функція додатку.
2. Ручне керування - система надає можливість керувати потоком вручну, блокуючи чи розблоковуючи ті, чи інші ділянки.
3. Відеозапис дорожнього потоку - система зберігає відео з кожного потоку, це відео є відредагованим, оскільки воно містить рамки для кожного знайденого об’єкта за допомогою системи детекції.
4. Зберігання стану світлофора - текстовий формат зберігання стану світлофорної ділянки в певну одиницю часу.
5. Добавлення слухачів - цей функціонал дозволяє добавляти слухачів на події, які продюсуються контролером.

Схема доступних функцій регулювання дорожнього трафіку представлена на рис 2.4.

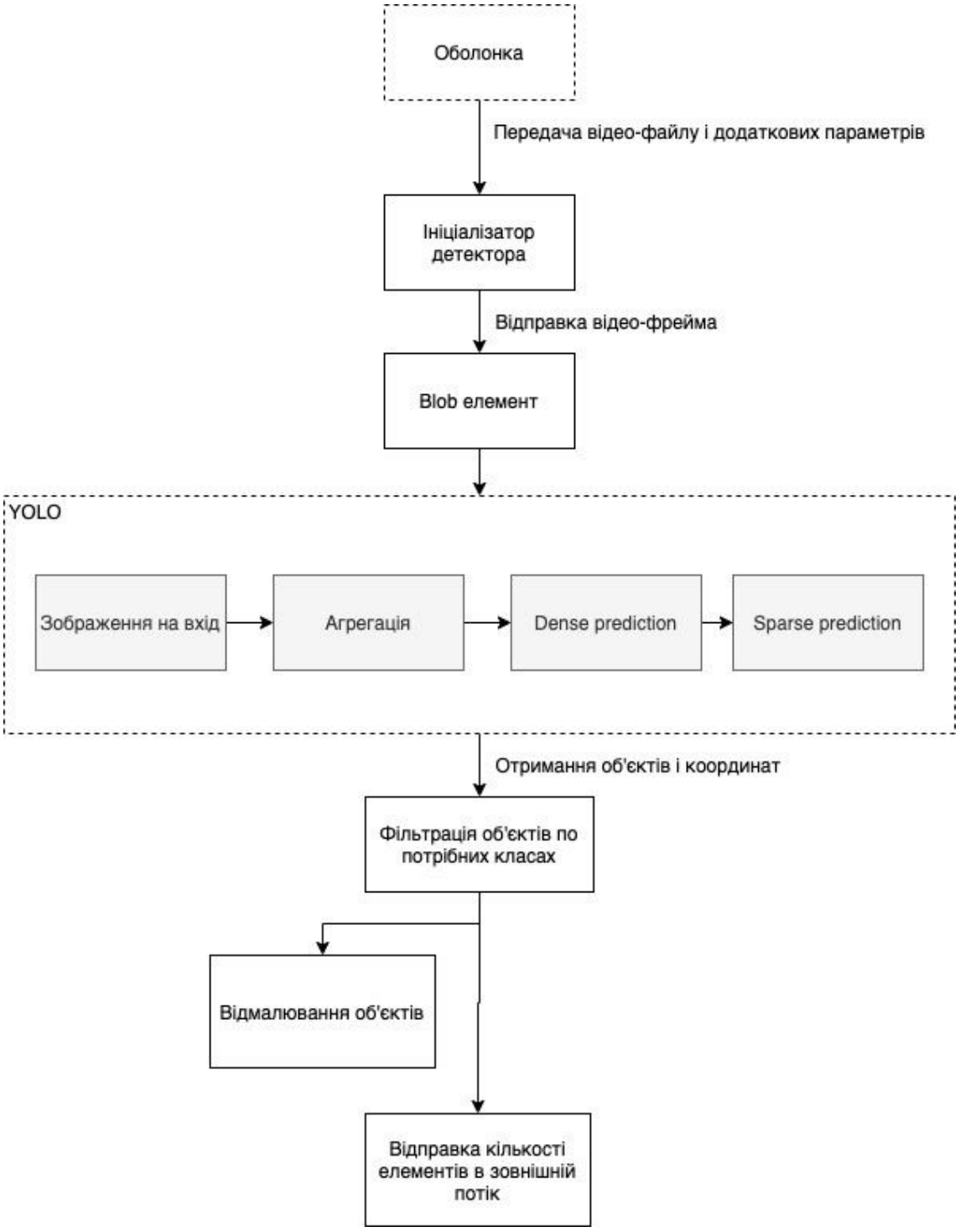


Рисунок 2.3 – Принцип дії ідентифікації об’єктів

Рисунок 2.4 – Функції системи розумного світлофора

Загальну схему регулювального алгоритму системи розумного світлофора представлено на рисунку 2.5.

Алгоритм визначення об’єктів на фреймі, буде визначати засоби транспорту, конкретна реалізації реагує лише на автомобілі різних габаритів і мотоцикли, можливість визначення сторонніх предметів - відключена.

Як тільки автомобілі були знайдені, система спробує просумувати автомобілі відповідно до осей: x і y, пізніше визначається сторона, яка вважається більш завантаженою, прохід дозволяється завантаженій стороні, при

тому з конкретним часом, який залежить від кількості автомобілів, цей час не має перевищувати максимальний час проходження. Зелене світло для одного потоку може повторитися лише 3 рази.

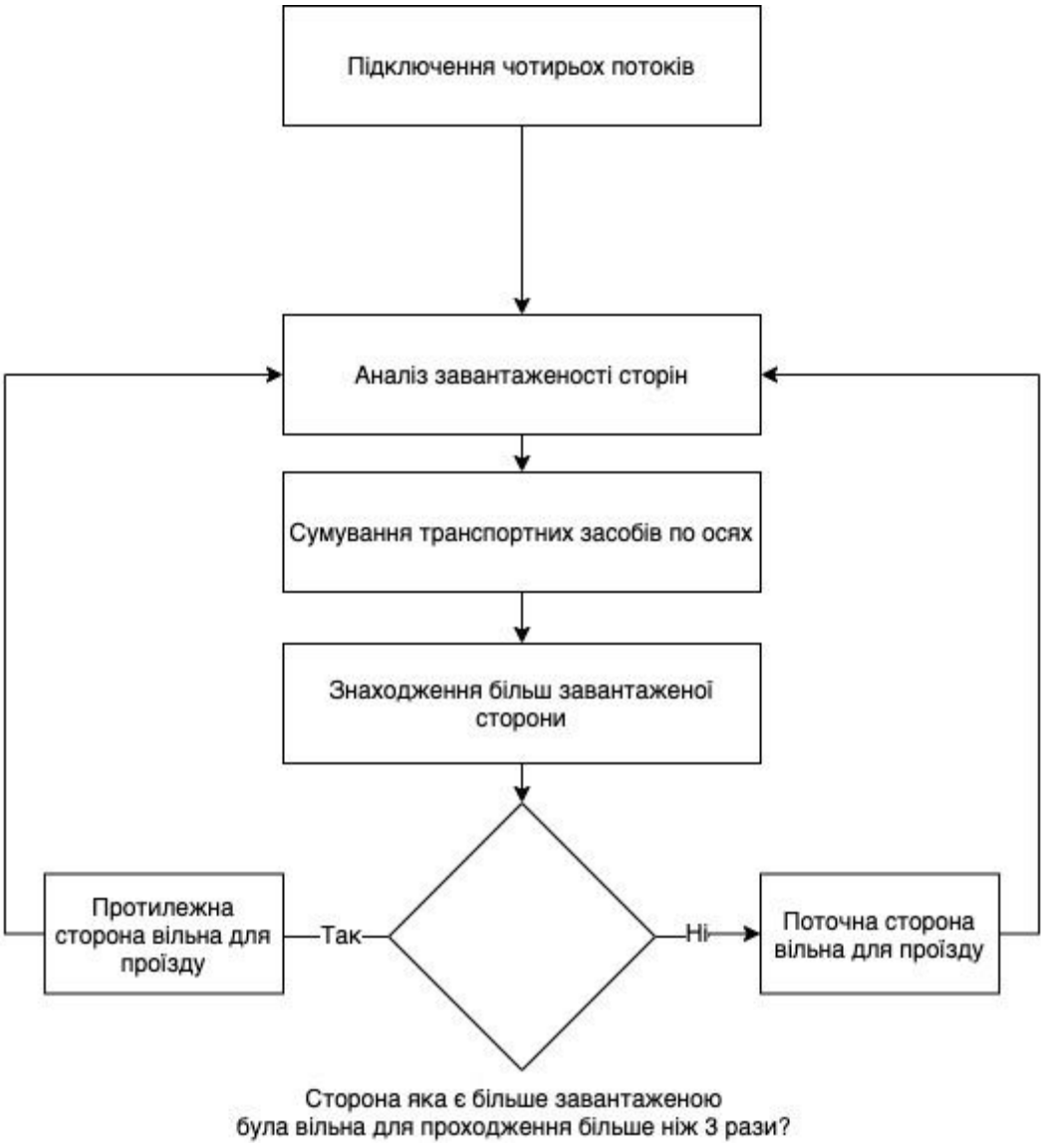


Рисунок 2.5. – Спрощена структура алгоритму системи регулювання дорожнього трафіку

Також існує алгоритм для визначення часу потрібного на проходження ділянки з світлофором він описується наступним чином: коефіцієнт проходження перехрестя одного автомобіля множиться на кількість авто, в

кінцевому результаті отримуємо час на проходження. Також є поріг, максимального значення часу, тобто, ми не можемо завжди тримати одну ділянку закритою, якщо на протилежній дуже багато автівок, тому, завжди є максимальне значення для зеленої зони світлофора. Схема алгоритму наведена на рис 2.6.

Рисунок 2.6. – Алгоритм обрахунку часу на проходження

Задля забезпечення можливості роботи програми в режимі реального часу із залученням незначних системних можливостей передбачається, у разі потреби, запуск режиму ітераційної детекції зображень з затримкою в потоковому форматі отримання даних. Процес роботи в такому режимі показаний на рисунку 2.7.

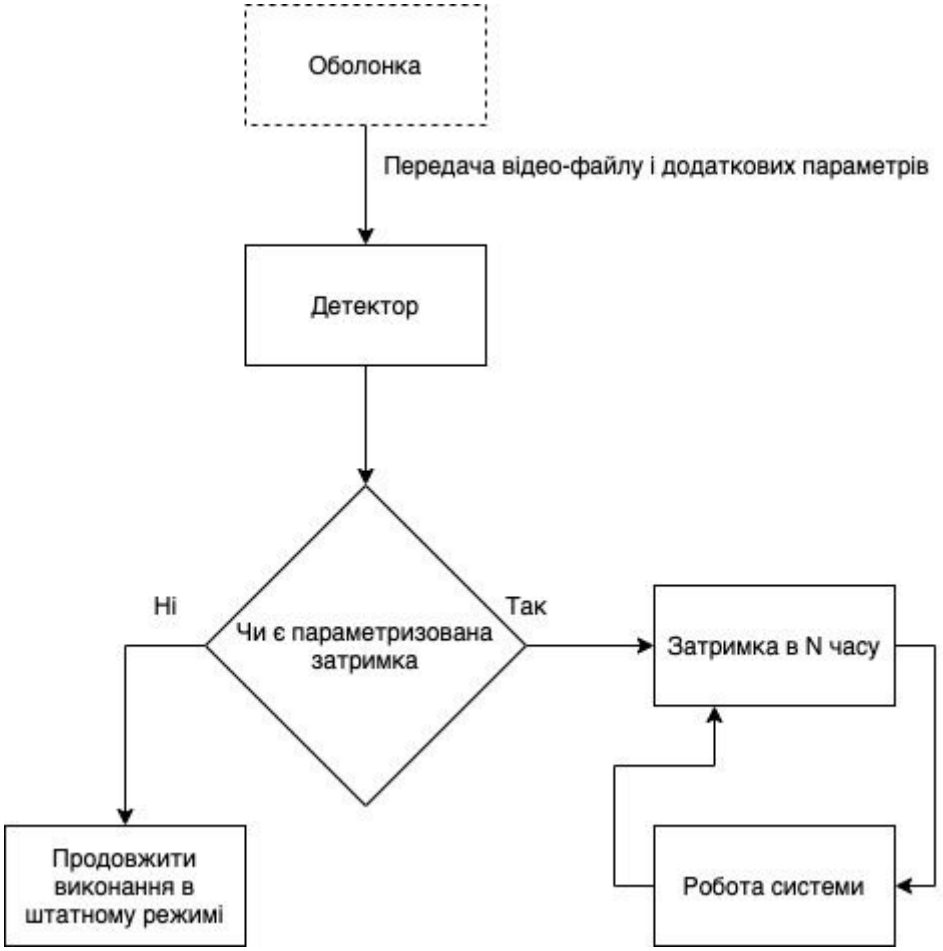


Рисунок 2.7. – Алгоритм дії детектора, якщо є параметризована затримка

Потреба у режимі ітераційної детекції пов’язана із тим, що виявлення об’єктів є достатньо ресурсоємкою задачею комп’ютерного зору. При ідентифікації об’єкту інформаційна система може знаходити та відстежувати об’єкт із заданого зображення чи відео. Особливим атрибутом виявлення об’єкта є те, що він визначає координати на конкретному зображенні та їх конкретні розташування.

Система роботи програми з затримкою в площині бізнес-логіки виглядала б так (рис. 2.8.).

Мається на увазі, що за якусь кількість часу ситуації завантаженості не зміниться, але при тому ми здобуваємо приріст в продуктивності самого додатку і можемо бути впевнені, що він не вийде з ладу через відсутність пам’яті.

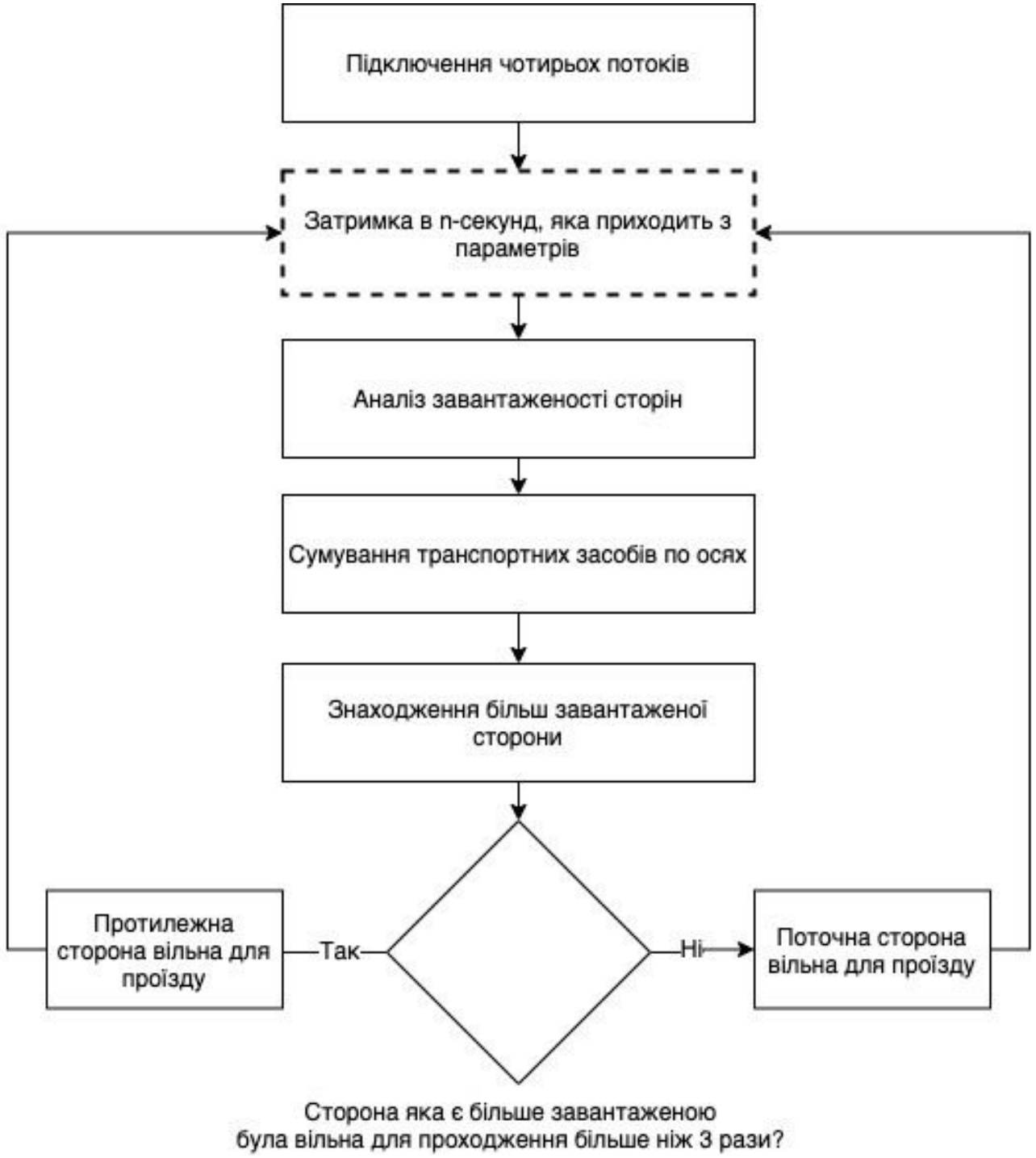


Рисунок 2.8. – Спрощена структура алгоритму системи регулювання дорожнього трафіку з внесенням додаткового шару з затримкою

Наступною додатковою функцією є керування дорожнього трафіку вручну. Цей функціонал дозволяє керувальнику змінити відкриту зону світлофорної ділянки, оскільки зона ділянки змінюється вручну - система

попадає в ручний режим керування, для того, щоб повернути систему назад, існує додатковий елемент інтерфейсу.

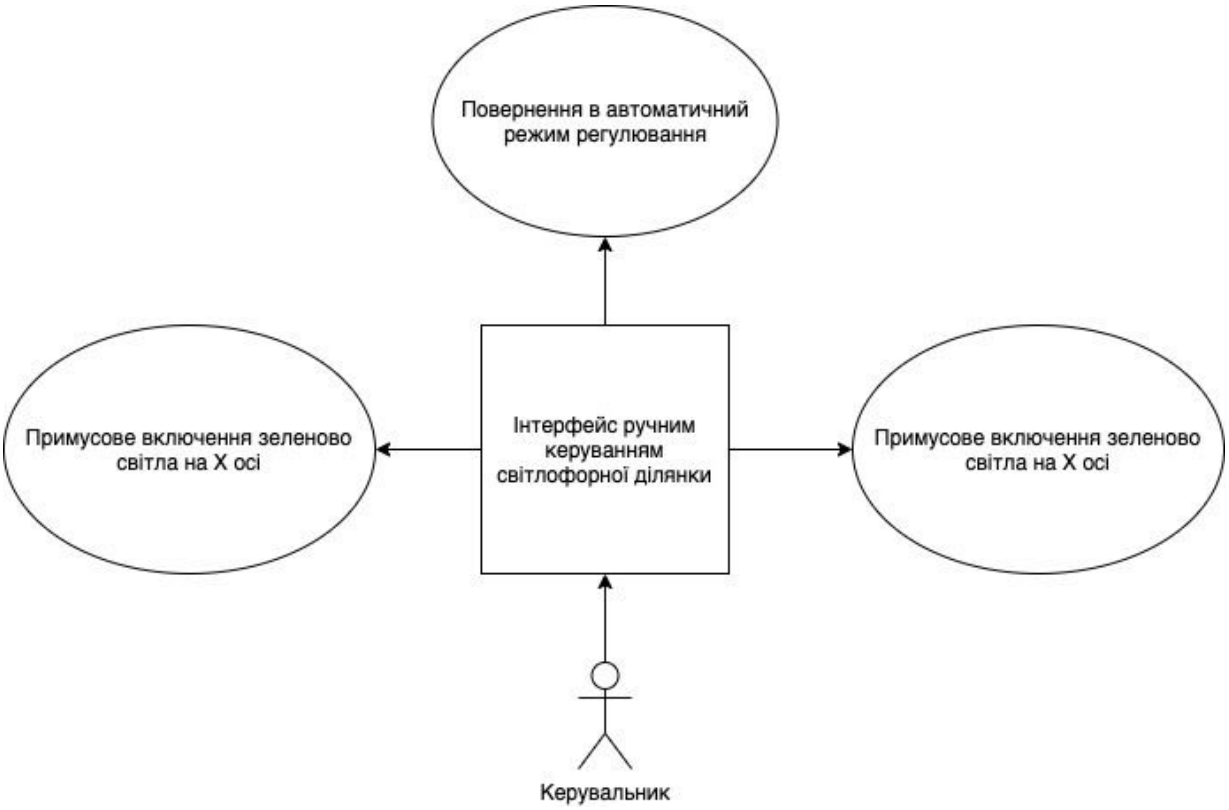


Рисунок 2.8. – Можливості ручного керування світлофорної ділянки

Наступною додатковою функцією є запис відео і текстових файлів. В межах цих функцій система записує відео і текстовий файл в файлову систему. Відео-файлом буде відео-потік, який знімався з камери впродовж роботи програми, текстовим файлом буде файл логування системи в конкретну одиницю часу. Журнал логування записує наступні дії, які показані на рисунку 2.9.

Формат запису в файл має наступний вигляд:

<РІК-МІСЯЦЬ-ДЕНЬ ЧАС>:ТИП ЛОГУВАННЯ:ПОВІДОМЛЕННЯ

Типом логування може бути:

1. Інформація.
2. Попередження.
3. Помилка.

Рисунок 2.9. – Дії, які записуються в журнал

Останньою додатковою функцією є можливість добавлення слухачів, які можуть реагувати на події спричинені контролером. Під такими діями мається на увазі:

1. Перехід світлофора в стандартний режим користування.
2. Трафік на двох ділянках був обрахований, як однаковий.
3. Стан світлофорної ділянки змінився.

Такі слухачі є незалежними одне від одного і можуть добавлятися на момент ініціалізації програми.

Висновки до розділу 2

Визначено принцип роботи системи і тип зберігання інформації. Побудовано інформаційну модель системи розумного світлофора. Визначені технології реалізації інформаційної системи розумного світлофора, з базовою

бібліотекою YOLO. Інформаційна система передбачається відкритою для забезпечення можливості розширення функціональності.

Розділ 3

Розробка методів та компонентів для системи регулювання дорожнього трафіку

3.1 Структура і функціональне призначення модулів системи, їх взаємозв’язок

Класи в проекті взаємодіють одне з однем через систему DI (Dependency injection), це покращує гнучкість проекту і unit-тестуванням.

У програмній інженерії введення залежностей - це техніка, при якій об’єкт отримує інші об’єкти, від яких він залежить. Ці інші об'єкти називаються залежностями. У типових відносинах "з використанням" отримуючий об'єкт називається клієнтом, а переданий (тобто "введений") об'єктом - службою. Код, який передає послугу клієнту, може бути різним, і називається інжектором. Замість того, щоб клієнт вказував, яку послугу він буде використовувати, інжектор повідомляє клієнту, яку послугу використовувати. "Введення" означає передачу залежності (послуги) в об'єкт (клієнт), який би її використовував.

Введення залежності - одна з форм більш широкої техніки інверсії контролю. Клієнт, який хоче викликати функцію переданого об’єкта, не повинен знати, як побудувати ці послуги. Натомість клієнт делегує відповідальність за надання своїх послуг зовнішньому коду (інжектор). Клієнту не дозволяється викликати код будь-якого метода напряму; саме інжектор створює послуги. Потім інжектор вводить (передає) послуги клієнту, які могли б уже існувати або можуть бути сконструйовані інжектором. Потім клієнт користується послугами. Це означає, що клієнту не потрібно знати про інжектор, як будувати послуги чи навіть які фактичні послуги він використовує. Клієнту потрібно лише знати про внутрішні інтерфейси служб, оскільки вони визначають, як клієнт може використовувати послуги. Це відокремлює відповідальність "використання" від відповідальності "будівництва".

Бізнес логіка зберігається в класі Controller, отримує дані, акумулює їх (дані з потоків можуть приходити в різному порядку, тому було реалізована логіка нагромадження данних). Діаграма класу Controller показана на рисунку 3.2.

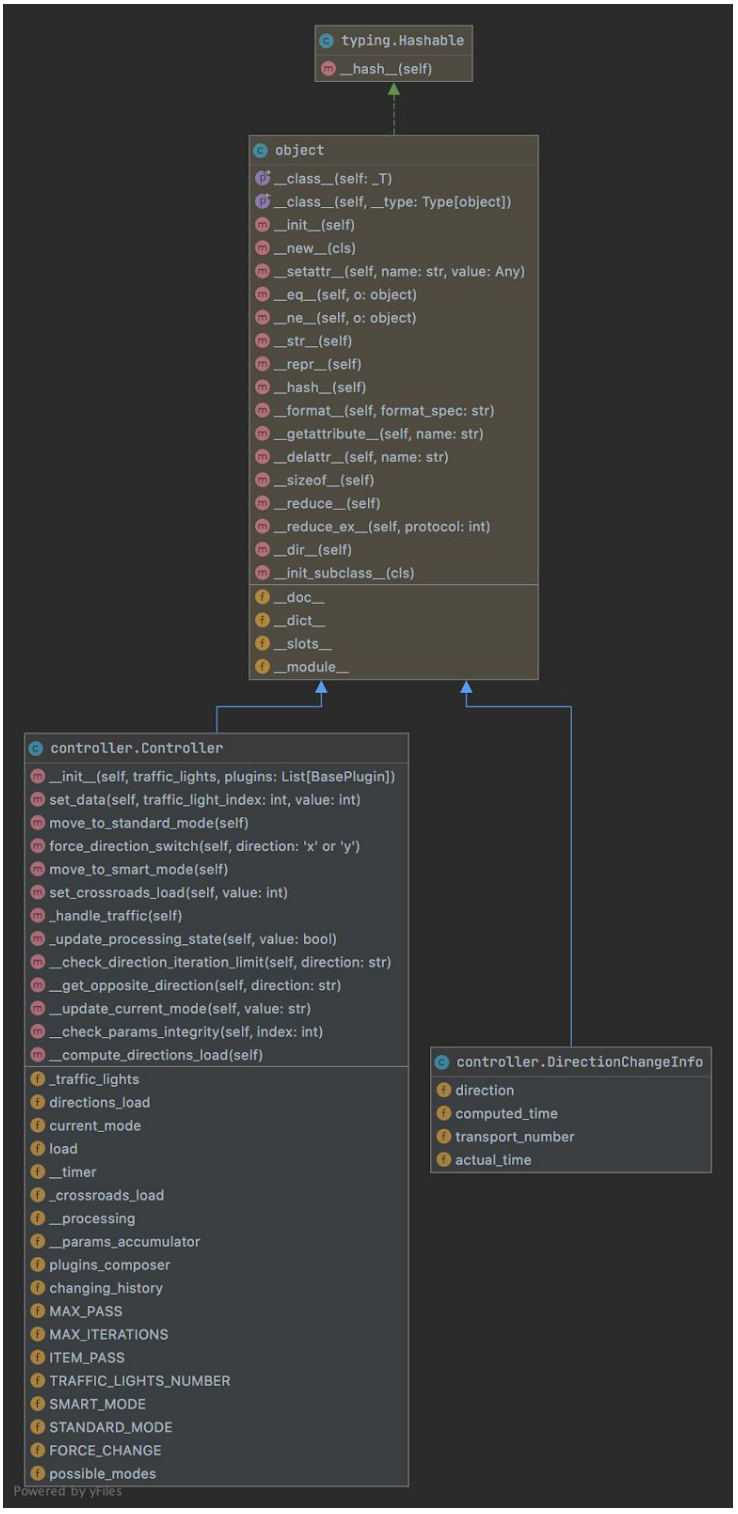


Рисунок 3.2 – Діаграма класу “Controller”

В Свою чергу Controller має залежність від класу TrafficLights та плагінів. Також є прості імпортовані залежності для модуля, а саме:

1. app_logger - локальний модуль, який пише інформамію в app.log файл і виводить інформацію в shell.
2. operator - модуль, який має можливість працювати з операндами.
3. numpy - модуль для роботи з масивами і матрицями.
4. threading - модуль для роботи з потоками, конкретно в цьому класі він використовується для таймера.

Controller залежить напряму від PluginsComposer класу, для чого потрібний цей клас буде описано нижче.

Важливо зауважити, що для всіх потоків сторін існує лише один екземпляр Controller'а. Поточний клас може розширювати дочірні класи, так як потрібні методи, де реалізується основна бізнес-логіка не є прихованими.

Також, в класі Controller зберігаються константи, які в перспективі можуть бути змінені, чи реалізовані через конфігурацію (наприклад, в конструкторі, при створенні екземпляра). Список констант:

1. MAX_PASS - максимальний час ітерації проходження для одного світлофора.
2. MAX_ITERATIONS - максимальна безперервна кількість ітерацій для одного світлофора.
3. ITEM_PASS - час для проходження одної одиниці транспортного засобу.
4. TRAFFIC_LIGHTS_NUMBER - кількість світлофорів.
5. SMART_MODE - назва розумного режиму (який реалізується проектом)
6. STANDARD_MODE - назва стандартного режиму.
7. possible_modes - можливі режими Controller'а.

Першочергово, нагромадження дорожнього руху по сторонах відбувається в масиві "load", індекси масиву відносяться до індифікаторів сторін

відповідно. Властивість “load” заповнюється інформацією в методі “set_data”, на вхід цей метод приймає індифікатор сторони і значення, також в цьому ж методі реалізовано нагромадження даних з всіх сторін, так як потрібно аналізувати всі дані одночасно. Для спрощення обрахунків існує властивість “directions_load”, поточна змінна зберігає в собі dictionary, з ключами x і y, значення кожного ключа - кількість автомобілів на відповідній осі. Якщо дані нагромадженні, то виконується метод “_handle_traffic”, цей метод може бути перезавантаженим в дочірному класі, якщо потрібно. Фактична реалізація в базовому класі шукає найбільш завантажену вісь і дозволяє проїзд для неї, на конкретну кількість часу (кількість транспортний засобів * ITEM_PASS), якщо час більший ніж MAX_PASS, то значення часу таймера буде вписано MAX_PASS. В випадку, якщо кількість транспортних засобів однакова - буде увімкнено стандартний режим світлофора. Під час процесу обробки, або процесу виконання приймання даних ігнорується, так як система входить в режим processing, хоча відправка даних з Detector екземплярів не припиняється. Для допоміжних цілей було додано властивість “current_mode”, яка відповідає за поточний режим системи.

Якщо ж кількість безперевних ітерацій на одній стороні, тобто одна сторона була дозволена для проїзду MAX_ITERATIONS разів, то наступна ітерація в будь якому випадку дозволить для проїзду протилежну сторону.

PluginsComposer - це клас який з'єднує список плагінів і надає інтерфейс для відправлення повідомлень всім плагінам за допомогою одного виклику. Цей має такий ж інтерфейс, як і самі плагіни - BasePlugin. Це важливо, оскільки він має підтримувати ідентичні події. Робота цього класу полягає в тому, що при отримані події від контролера, відбувається перебіг по всіх плагінах і відправляється ідентична подія, яка була відправлена самому екземпляру PluginsComposer. Діаграма класу вказана на рисунку 3.3.

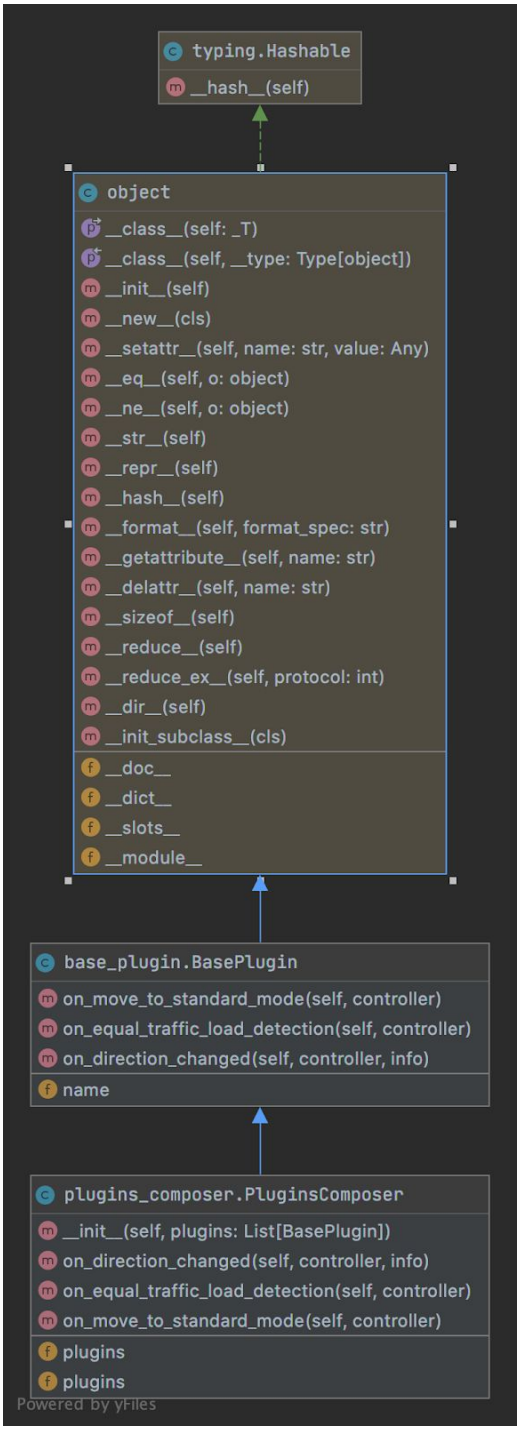


Рисунок 3.3 – Діаграма класу “PluginsComposer”

TrafficLights - це клас, який на даний час працює як заглушка, він реалізує стандартні дії світлофора і має режим стандартного регулювання трафіка.

Діаграма класу показана на рисунку 3.4.

Рисунок 3.4 – Діаграма класу “TrafficLights”

Передбачається, що світлофори мають мінімальне API, це може бути не обов’язково API з Python модуля. API може бути на низькорівневих мовах і обгорнутим в Python. Цей модуль має лише одну імпортовану залежність - app_logger, для інформування.

BasePlugin - це абстрактний клас, який забов’язує користувача описати в ньому кілька методів-подій, які викликаються контролером. Діаграма класу зображена на рисунку 3.5.

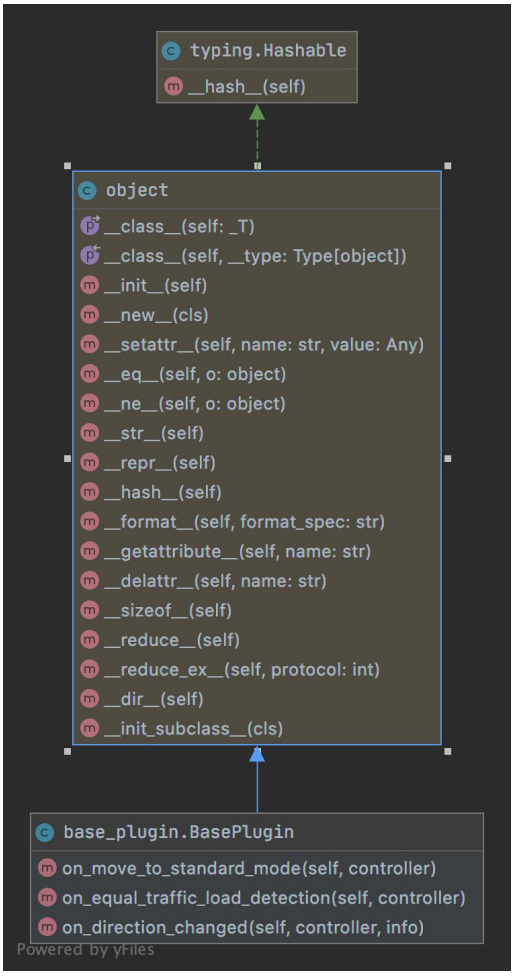


Рисунок 3.5 – Діаграма класу “BasePlugin”

На даний момент існує три події:

1. Світлофорна ділянка була переведена в стандартний режим.
2. Навантаження ділянки по всіх осях одинакове.
3. Стан світлофора змінився.

TrafficLightsResolver - клас, який надає UI представлення для того, щоб можна було організувати ручне керування світлофором. Для ініціалізації цей клас потребує контролер. Клас уніслідує QWidget для можливості побудови інтерфейсу.

Діаграма класу зображена на рисунку 3.6.

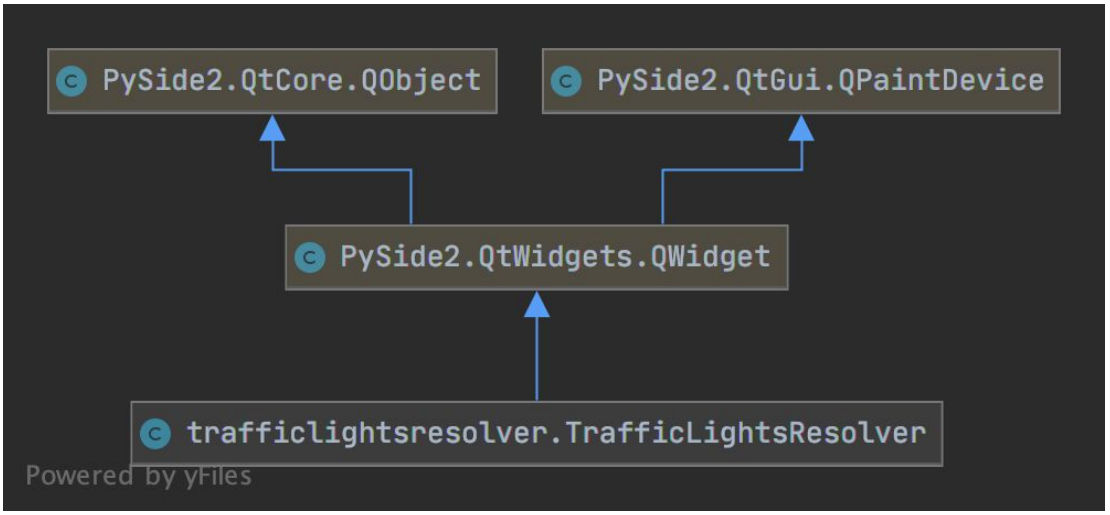


Рисунок 3.6 – Діаграма класу “TrafficLightsResolver”

Віджети - це основні будівельні блоки для програм графічного інтерфейсу (GUI), побудованих за допомогою Qt. Кожен компонент графічного інтерфейсу (наприклад, кнопки, мітки, текстовий редактор) - це віджет, який розміщується десь у вікні інтерфейсу користувача або відображається як незалежне вікно. Кожен тип віджетів забезпечується підкласом QWidget, який сам по собі є підкласом QObject.

QWidget не є абстрактним класом. Його можна використовувати як контейнер для інших віджетів, і його можна підкласифікувати з мінімальними зусиллями для створення нових, спеціальних віджетів. QWidget часто використовується для створення вікна, всередині якого розміщуються інші QWidgets.

TrafficLightsResolver клас створює інтерфейс і добавляє на кожний елемент обробник події на натискання. При кожному кліку буде викликатися відповідний метод класу Controller’а.

Logger - це клас, який реалізує інтерфейс абстрактного класу “BasePlugin”. Клас потрібний для того, щоб виводити в термінал поточний стан світлофора і показувати повідомлення, які відносяться до зміни стану світлофорної ділянки. Діаграма класу наведена на рисунку 3.7.

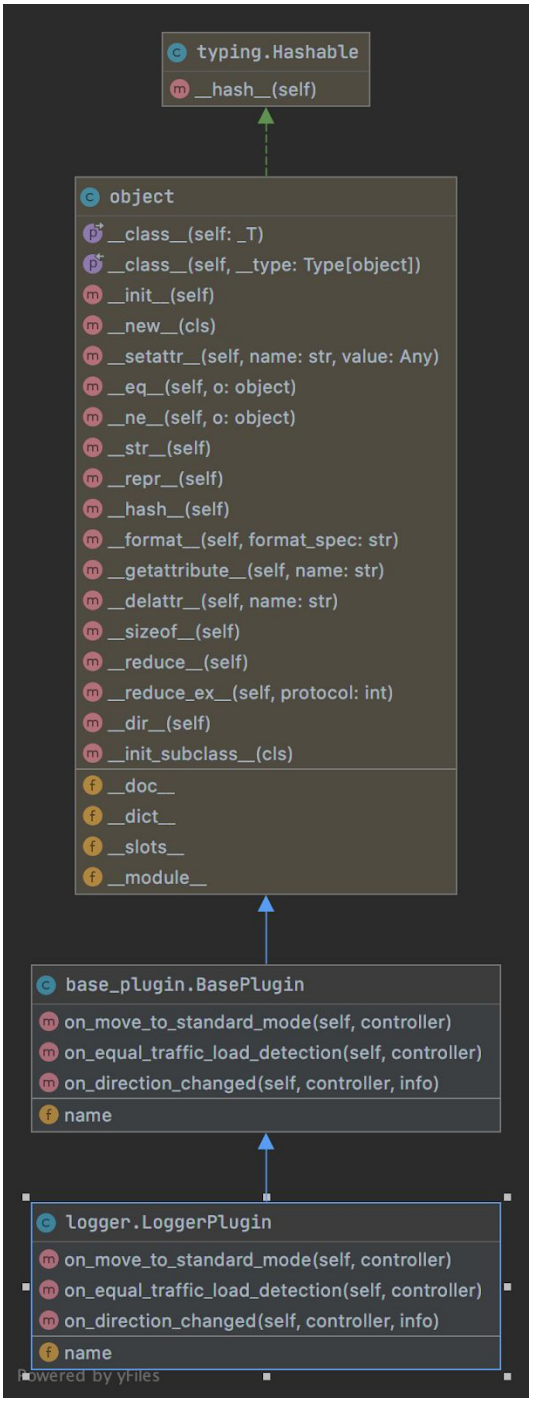


Рисунок 3.7 – Діаграма класу “LoggerPlugin”

Залежності модуля - “logging” модуль, який надає можливість налаштовувати всі вихідні повідомлення з рівнями, типами і форматуванням.

Detector - це клас (одночасно і самостійний модуль), який реалізує знаходження об’єктів на фреймі. Цей модуль може використовуватися

самостійно (передачею аргументів через bash) або як простий клас (передачею аргументів в конструктор). Діаграма класу показана на рисунку 3.8.

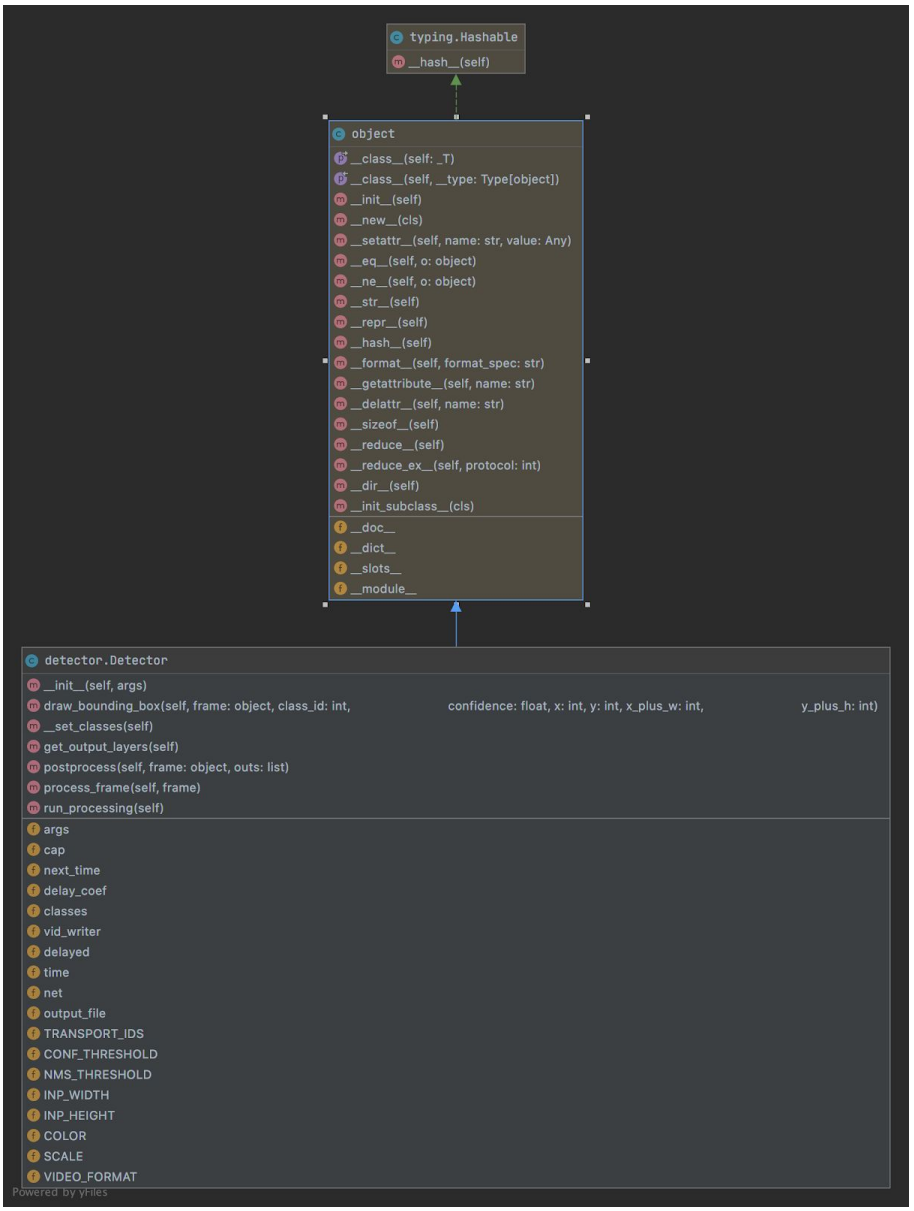


Рисунок 3.8 – Діаграма класу “Detector”

Імпортовані залежності:

1. sys - модуль для взаємодії з інтерпретатором.
2. os - модуль для взаємодії з операційною системою.
3. cv2 - модуль для взаємодії з OpenCV.
4. numpy - модуль для взаємодії з масивами і матрицями.

5. cli_args - локальний модуль, використовується для взяття агрументів з bash’у.
6. time - модуль для взаємодії з часом.

Оскільки модуль здебільшого працює з OpenCV є константи, які потрібні для взаємодії. Список констант:

1. TRANSPORT_IDS - масив індифікаторів транспортних засобів, оскільки нейронна мережа може розрізняти не лише ТЗ, то в поточній програмі потрібно ігнорувати інші об’єкти, які були зафіксовані. Пізніше, це можна розширити, можливості, так як базуватися можна буде й на пішоходах і т.п. Об’єкти, які можуть бути знайдені в реалізації на даний час:
    a. велосипед;
    b. автомобіль;
    c. мотоцикл;
    d. автобус;
    e. вантажівка;
2. CONF_THRESHOLD - коефіцієнт порогу для детекції. Важливим нюансом є те, що YOLO надає передбачення в коефіцієнті, на скільки знайдений об’єкт може відноситися до якогось відповідного класу. Щоб відсіяти “слабкі” детекції, ми використовуємо цей коефіцієнт.
3. NMS_THRESHOLD (Non-max suppression) - Алгоритм YOLO розбиває зображення на 7x7 ділянок, для того щоб запобігти дублюванню дуплікацій на сусідніх ділянках застосовується цей коефіцієнт. Якщо значення високе (близьке до одиниці), то детекції можуть накладатися одне на одну.
4. INP_WIDTH & INP_HEIGHT - для роботи НМ може використовувати лише обмежену кількість розширень, тому ці змінні зберігаються одну з таких значень. В цій реалізії, вони зберігають

найнижче можливе значення, оскільки для високих значень, процесорний час виконання збільшується, але, на жаль, точність зменшується.

5. COLOR - колір обведення для знайденого об’єкта.
6. SCALE - ця зміння також відносить до роботи НМ, так як крім ширини і висоти, нейронній мережі потрібно працювати лише з одним масштабом, значення якого зберігається в цій константі.

Аргументи в екземпляр потрапляють під час ініціалізації екземпляра параметром “args”. Перш за все, під час ініціалізації (виконання конструктора) добавляються класи по вказаному шляху до файла з ними, створюється екземпляр “VideoCapture”, який дозволяє читати відео-файл з файлової системи, створюється екземпляр “VideoWriter” класу, який дозволяє записувати відео в файлову систему, створюється екземпляр прочитаної нейронної моделі по вказаних параметрах з “args”.

“args” параметр реалізує потреби, які описані в cli_args.py:

1. config - шлях до конфігураційного файлу YOLO.
2. weights - шлях до файлу з вагами для нейронної мережі.
3. classes - шлях до файлу з класами.
4. video - шлях до відеофайла.
5. outputDir - шлях до директорії, де зберігатиметься записане відео.
6. record - визначає чи відео має записуватися чи ні.
7. id - ідентифікатор світлофорної ділянки.
8. delay - час затримки між ітераціями детекції

Якщо модуль detector.py запускається, як головний модуль, то дані будуть братися з bash’у і метод, який запускає детекцію - run_processing, буде запущений автоматично. Якщо ж Detector був імпортований, то аргументи будуть передаватися в конструктор, а запуск потрібно буде виконати власноруч.

Detector зберігає в собі “net” властивість, яка є нейронною мережею. OpenCV зберігає можливість використовувати НМ в об’єкті dnn. Основна

можливість dnn полягає, звичайно ж, в завантаженні і запуску нейронних мереж (inference).

Процес детекції є доволі простим, перш за все виконання йде в безкінечний while цикл, який читає фрейми з video. Потрібно зазначити, що реалізація визначення об’єктів для відео і для зображень виконується одним і тим алгоритмом, бо відео - це ніщо інше, як набір зображень (фреймів). Для проходження по фреймах й потрібний цей цикл. Якщо ж фрейми закінчилися, то виконання циклу переривається.

Для знаходження об’єктів з фрейма утворюється blob. Blob - бінарний набір даних, що розглядається як єдиний, неперервний об'єкт. Цей набір даних потрібний для того, щоб утворені дані до НМ. Фактично, НМ не розуміє інших форматів, крім blob. Дані додаються в НМ методом “setInput”, пізніше запускається механізм обробки методом “forward”, цьому методу на вхід потрібні шари верхнього рівня, їх отримуємо виконанням методу класу “get_output_layers”, виконання переходить в метод “postprocess”, цей метод відфільтровує всі знайдені об’єкти по порогу сходження з класами, по NMS (Non-max suppression) і по індифікаторам класів. Метод “postprocess” викликає “draw_bounding_box” метод, який виділяє рамкою з кольором COLOR (константа, яка зберігається в класі) і добавляє інформацію про збір, в кінцевому результаті “postprocess” повертає масив індифікаторів знайдених об’єктів, в свою чергу кількість елементів передається по stdout потоку через print метод.

В звичайному виконанні stdout буде записувати дані в shell, але якщо програма запущена в потоці, під іншою програмою, (в нашому випадку під main.py) то вивід відбувається в головну програму, також важливо пам’ятати, що потрібно скидувати буфер запису, по замовчуванні буфер накопичує дані перед тим, як викинути їх в потік, це зроблено в цілях оптимізації.

Кінцевими етапами є те, що зображення відображаються в окремих вікнах, назви цих вікон вказують на індифікатор сторони. Відображення реалізується за допомогою метода “imshow” екземпляра OpenCV. Якщо ж

конфігурація запуску вказує, що запис є потрібним, то, результат буде записаний функціоналом властивості “vid_writer”.

Процес трохи змінюється, якщо є опціональний параметр “delay”, за допомогою якого здійснюється затримка між ітераціями безкінечного циклу. Для реалізації затримки застосовуються властивості time і next_time, для кожного разу next_time дорівнює сумі значенню “delay” і поточного часу. Таким чином використовуючи час ми можемо досягнути ефекту відкладеного виконання коду.

Загальна взаємодія класів в межах програми наведена на рисунку 3.9.

3.2 Розробка програмних модулів

Для реалізації системи регулювання трафіку було обрано мову програмування Python, бібліотека OpenCV. Оскільки ні Python, ні OpenCV (так як являється бібліотекою, а не фреймворком) не накладає жодних обмежень на реалізацію на структуру, тому її реалізації була обрана власноруч, так, щоб задовольнити потреби.

Ми розглянули модулі, які містять в собі лише класи, тому модулі controller.py, detector.py і traffic_lights.py, plugins_composer.py, logger.py, base_plugin.py не будуть тут обговорюватися.

Модулі які не містять в собі класів, а виконують допоміжні або комбінаторні функції.

app_logger.py - модуль для логування інформації про виконання програми. Система логування для аплікації є розширенням стандартного модуля - logging. Журнал (Logger) записує три типи повідомлень: інформація, помилка і попередження, добавлено кілька обробників, які записують всю інформацію в файл app.log. Також добавлено точна дата і час до кожного з повідомлень.

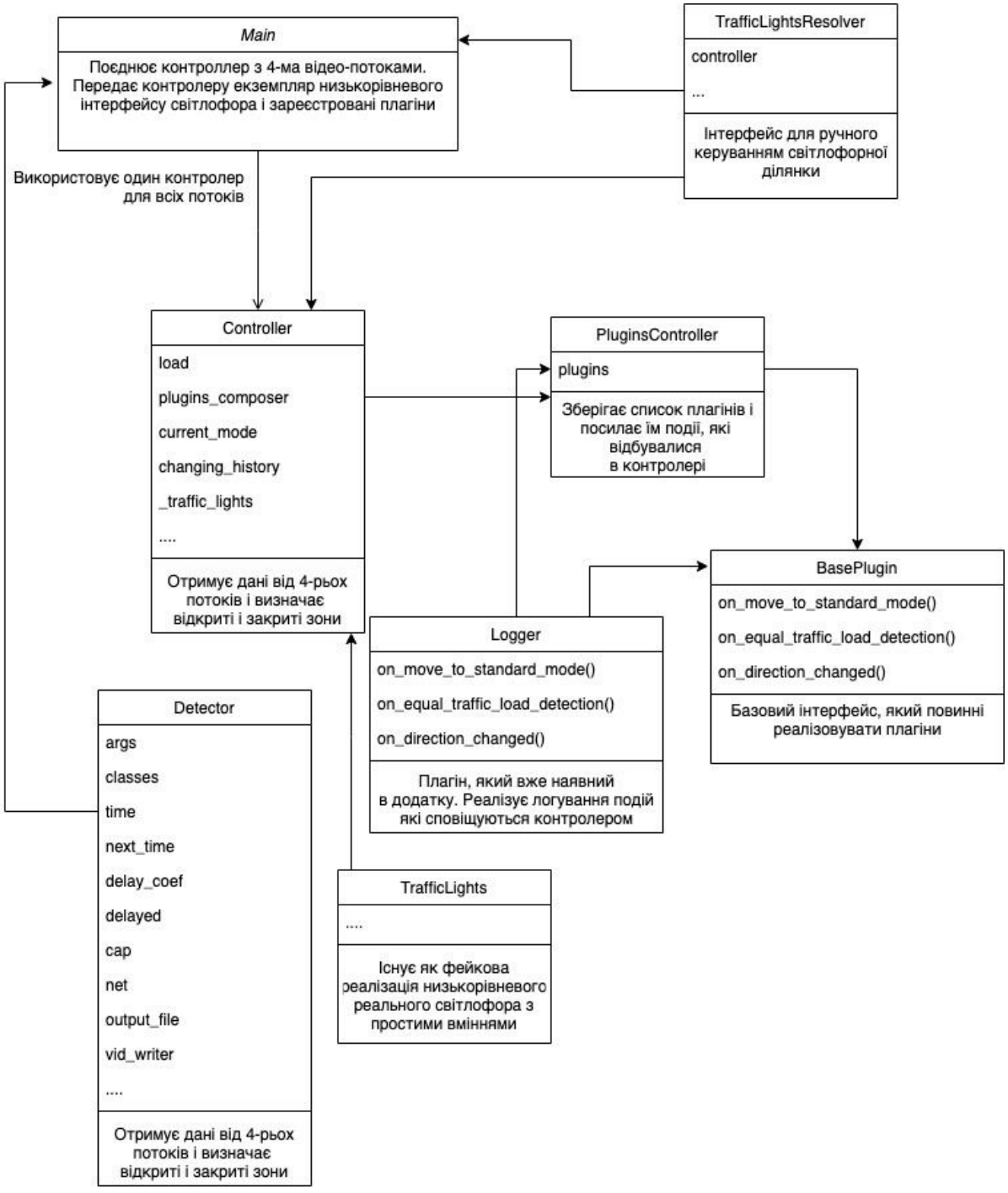


Рисунок 3.9 – Діаграма взаємодії класів без сторонніх модулів

Оскільки app_logger являє собою екземпляр logging модуля, то для всіх модулів, де він (app_logger) застосовується - доступні методи - debug, critical, error, info, warn, для типу повідомлень. Крім запису в файл, система виводить інформації на термінал, це корисно тим, що ти можеш бачити повідомлення в режимі реального часу під час моніторинга системи.

Модуль logging в Python - це готовий до використання, потужний модуль, призначений для задоволення потреб як початківців, так і корпоративних команд. Він використовується більшістю сторонніх бібліотек Python, тому ви можете інтегрувати ваші логи з повідомленнями з цих бібліотек для створення єдиного журналу логів в вашого додатку.

З імпортованим модулем logging ви можете використовувати те, що називається «logger», для логування повідомлень, які ви хочете бачити. За замовчуванням існує 5 стандартних рівнів важливості (перераховані вище), що вказують на важливість подій. У кожного є відповідний метод, який можна використовувати для логування подій на обраному рівні важливості.

Логування - це засіб відстеження подій, які трапляються під час запуску деякого програмного забезпечення. Подія описується повідомленням, яке за бажанням може містити змінні дані (тобто дані, які потенційно відрізняються для кожного входження події). Події також мають значення, яке розробник приписує події; важливість також можна назвати рівнем або важкістю.

Цей модуль визначає функції та класи, які реалізують гнучку систему реєстрації подій для програм та бібліотек.

Ключовою перевагою використання API ведення журналу, що надається стандартним бібліотечним модулем, є те, що всі модулі Python можуть брати участь у веденні журналу, тому журнал вашого додатку може включати ваші власні повідомлення, інтегровані з повідомленнями від сторонніх модулів.

Модуль забезпечує багато функціональних можливостей та гнучкості.

Основні класи, визначені модулем, разом із їх функціями, перелічені нижче.

1. Логери виставляють інтерфейс, який безпосередньо використовує код програми.
2. Обробники надсилають записи журналу (створені реєстраторами) до відповідного пункту призначення.

3. Фільтри забезпечують більш дрібну структуру для визначення того, які записи журналу виводити.
4. Форматори вказують макет записів журналу в кінцевому результаті.

Наступним допоміжним модулем є cli_args. Цей локальний модуль допомагає описати життєво-важливі параметри для компонента, тобто, якщо параметр, який позначений як “Required” в cli_args модулі не буде переданим, то додаток просто не запуститься. Сам ж модуль cli_args залежить від argparse.

Модуль argparse дозволяє легко писати зручні інтерфейси командного рядка. Програма визначає, які аргументи вона вимагає, і argparse зрозуміє, як проаналізувати їх із sys.argv. Модуль argparse також автоматично генерує повідомлення про допомогу та використання та видає помилки, коли користувачі надають програмі недійсні аргументи.

Заповнення ArgumentParser інформацією про аргументи програми здійснюється за допомогою викликів методу add_argument (). Як правило, ці виклики говорять ArgumentParser, як взяти рядки в командному рядку і перетворити їх на об'єкти. Ця інформація зберігається та використовується при виклику parse_args ().

ArgumentParser аналізує аргументи за допомогою методу parse_args (). Це перевірить командний рядок, перетворить кожен аргумент у відповідний тип, а потім викличе відповідну дію. У більшості випадків це означає, що простий об’єкт простору імен буде створений з атрибутів, проаналізованих з командного рядка.

У сценарії, parse_args (), як правило, викликається без аргументів, а ArgumentParser автоматично визначає аргументи командного рядка з sys.argv.

Передавати cli аргументи можна двома способами, коротким і довгим записом аргументів, для прикладу, щоб передати шлях до конфігураційного файлу YOLO в bash’і потрібно зробити наступне: -c path/to/config/file, або ж так: --config /path/to/config/file.

На даний момент є такі параметри:

1. config - обов’язковий параметр, шлях до конфігураційного файлу YOLO.
2. weights - обов’язковий параметр, шлях до файлу з вагами для нейронної мережі.
3. classes - обов’язковий параметр, шлях до файлу з класами.
4. video - обов’язковий параметр, шлях до відеофайла.
5. outputDir - необов’язковий параметр, шлях до директорії, де зберігатиметься записане відео.
6. record - необов’язковий параметр, визначає чи відео має записуватися чи ні. Початкове значення - False.
7. id - необов’язковий параметр, індифікатор світлофорної ділянки. Початкове значення - 0.

Послідовність передання аргументів не має значення.

Наступний модуль, який не складає окремі, ізольовані частини програми в одне ціле - main.py. Цей модуль створює екземпляр Controller’а і передає його в функцію, яка є головним виконавчим об’єктом для потоків. Через те, що екземпляр єдиний, об’єкти спілкуються з одним екземпляром бізнес-логіки.

Дочірні процеси запускаються за допомогою bash-команди, якій додатково передається назва відео, яке буде присвячене поточному потоку. Скрипт sh описаний в файлі start.sh. Його основна місія - це приймати відео параметр, параметр легкого режиму і індифіктор відео.

Поточна імплементація завдання має два режими: легкий (tiny) і повний (full). Ці режими існують для виконання роботи в різних середовищах. До прикладу, легкий режим доволі добре справляється на машинах з малою потужністю, при тому якість детекції різко знижується, іншими словами, кількість об’єктів, які знаходяться значно менша ніж в повному режимі, тому легкий режим детекції не рекомендується. Якщо ж ресурси дозволяють використовувати повний режим, то детекція буде значно точнішою.

Висновки до розділу 3

У розділі 3 були визначені та описані основні модулі. Вибрані засоби розробки інформаційної системи. Визначено структуру реалізованого програмного застосунку, включно із класами та взаємодія між ними. Описано макети користувацького інтерфейсу та принцип їх побудови. Представлено схему функцій користувача.

Реалізований програмний застосунок має відкриту архітектуру та можливості для зручного розширення функціоналу.

Розділ 4

Апробація програмної інформаційної системи

4.1 Основні функції програми

Система була розроблена на найбільш поширеному типі перехресть, де чотири ділянки дороги перетинаються, але з незначними змінами не буде проблем адаптації такої системи під інші види перехресть.

Оскільки система має 2 режима запуску відносно потужності пошуку об’єктів на відео-потоці, ми можемо перевірити як працює “легкий” і повний режим роботи.

Для перевірки бралися зображення з мережі Інтернет, які відповідають реальним умовам, коли автомобілі можуть перекривати інші ТЗ.

Список зображень:

- дві автівки [6];
- п’ять автівок [7];
- велика кількість трафіку [8];
- дев’ять автівок одна за одною [9].

На рисунку 4.1. зображено легкий режим, як можна побачити, що з 9 автівок система могла знайти лише 2. Оскільки детекція - це підпрограма, яка віддає результат кількості автомобілів назовні, то можна побачити кількість автомобілів на терміналі зліва.

Рисунок 4.1 – Детекція в “легкому” (tiny) режимі

На рисунку 4.2 зображено повний режим роботи детекції об’єктів, оскільки зображення не є тривіальним, то навіть повний режим програми не зміг знайти всі автівки, для повного режиму результат - 6 автівок, також кількість авто підпрограми наведена в терміналі, як і в попередньому прикладі.

Різниця між режимами значна, але варто зауважити, що на цьому зображенні доволі складно розрізнити автомобілі. До прикладу, детекція на основі HAAR каскаду не впоралася з задніми автомобілями, так значна їхня частина прихована.

Якщо змінити відео, де автівки показані в більш чіткому представленні, то легкий режим також показує знайдених 2 автівки з 5. Результат можна побачити на рисунку 4.3.

При зміні режиму на повний і при оновленому відео система детекції знаходить всі автомобілі, зображено на рисунку 4.4.

Комплексно, система дивиться на 4-ри різних потоки і старається виконати додавання автомобілів відповідно до осей.

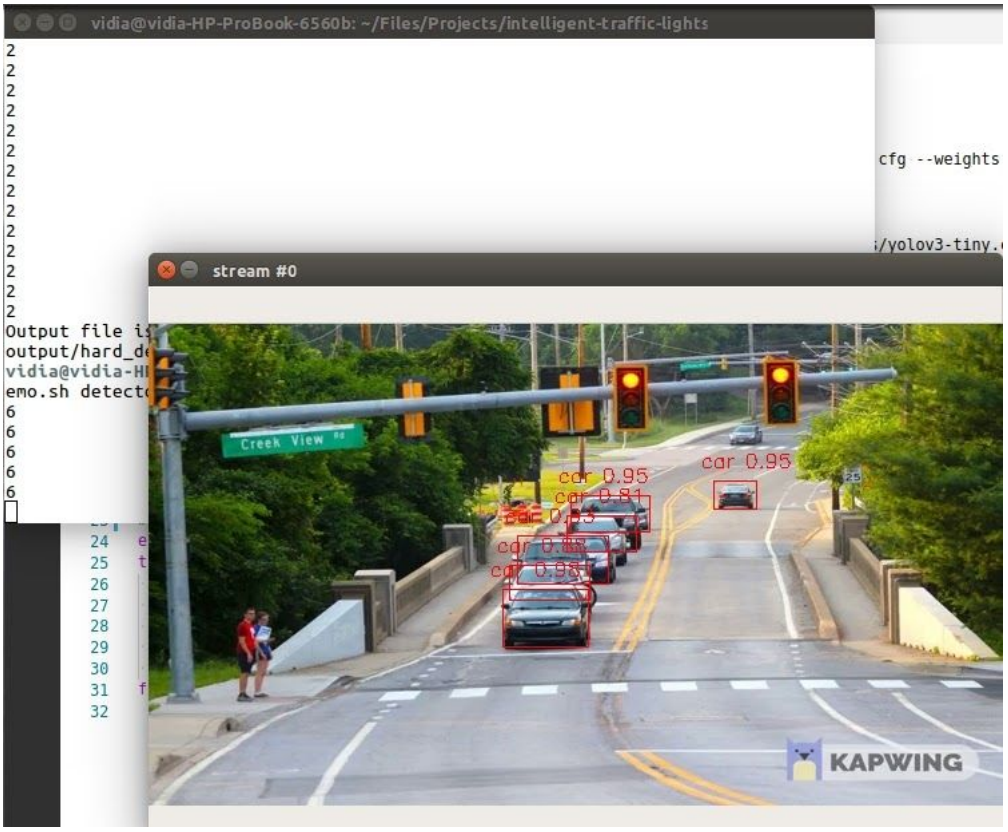


Рисунок 4.2 – Детекція в “повному” (full) режимі

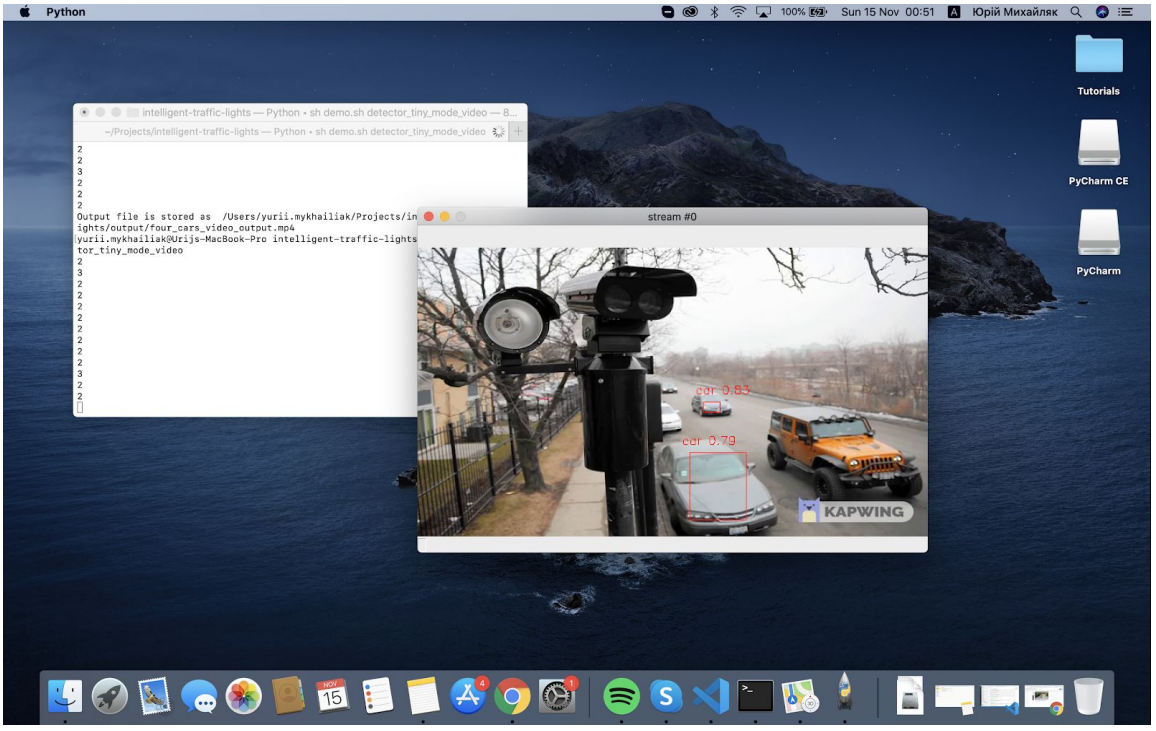


Рисунок 4.3 – Детекція в “легкому” (tiny) режимі при зміні відео

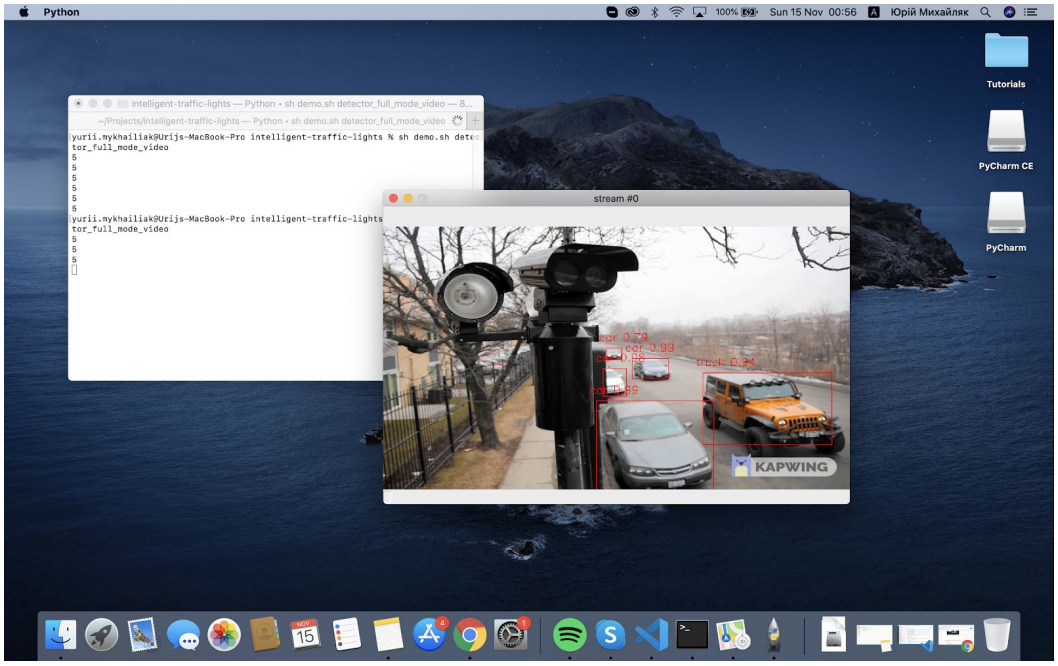


Рисунок 4.4 – Детекція в “повному” (full) режимі при зміні відео

На рисунку 4.5. зображено повністю запущено програму в повному режимі, але відео-файли складаються з одного фрейму, іншими словами, це відео з одним зображенням.

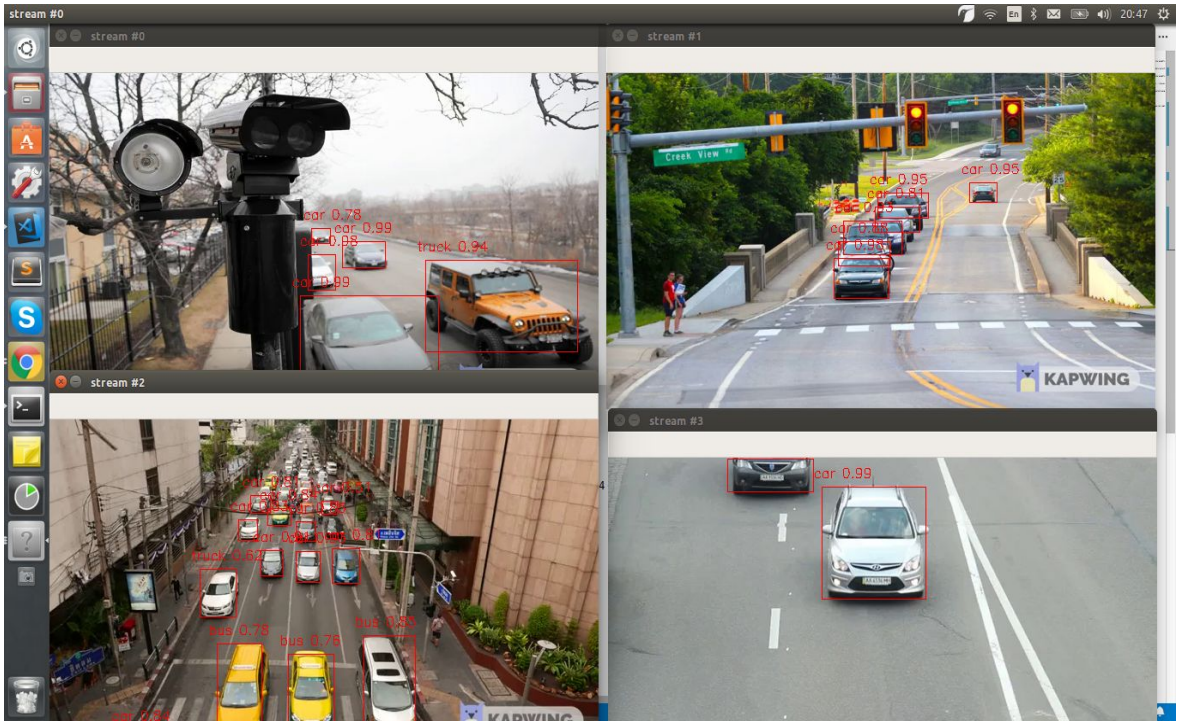


Рисунок 4.5 – Система в “повному” (full) режимі з однофреймовими відео-файлами

Знаходження автомобілів є доволі точним, на рисунках 4.6. - 4.7. можна побачити інформацію щодо поведінки програми по журналу (logger).

```
2019-05-19 20:44:57,762:INFO:Directions state: {'y': 18, 'x': 8}
2019-05-19 20:44:57,764:INFO:
 Visualization of crossroad state

                   (0)
                    |
                    o
                    |
         (3)----x---C---x----(1)
                    |
                    o
                    |
                   (2)


2019-05-19 20:45:52,074:INFO:
                Green direction: y;
                Needed time: 90;
                Count of transport: 18;
                Actual time: 50

2019-05-19 20:45:52,075:INFO:Directions state: {'y': 18, 'x': 8}
2019-05-19 20:45:52,103:INFO:
 Visualization of crossroad state

                   (0)
                    |
                    o
                    |
         (3)----x---C---x----(1)
                    |
                    o
                    |
                   (2)


2019-05-19 20:46:46,925:INFO:
                Green direction: y;
                Needed time: 90;
                Count of transport: 18;
                Actual time: 50

2019-05-19 20:46:46,927:INFO:Directions state: {'y': 18, 'x': 8}
2019-05-19 20:46:46,932:INFO:
 Visualization of crossroad state

                   (0)
                    |
                    o
                    |
         (3)----x---C---x----(1)
                    |
                    o
                    |
                   (2)
```

Рисунок 4.6 – Перших три ітерації програми з однофреймовими відео-файлами

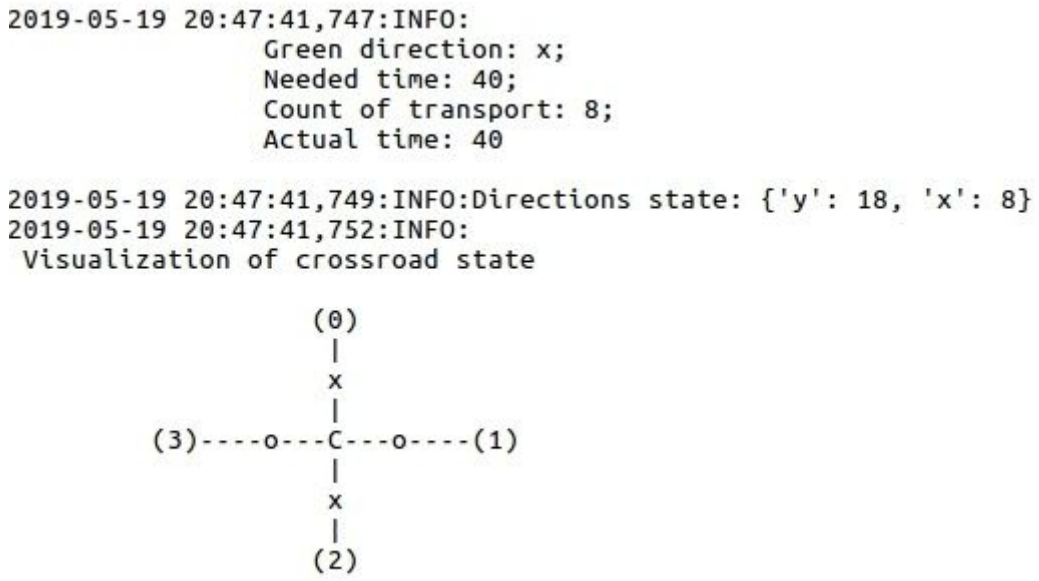


Рисунок 4.7 – Четверта ітерація програми з однофреймовими відео-файлами

Якщо порівняти рисунки 4.6. і 4.7. можна побачити, що перших три ітерації були закінчені в користь осі у, остання ж була в користь осі х. Оскільки відео однофреймові, дані на відео не змінюються, вони залишаються постійними, тому повідомлення “Directions state” залишається незмінним, це повідомлення відображає поточну кількість автомобілів на осях. “Green direction” вказує на вісь, яка в даний час доступна для проїзду, “Needed time” вказує на час який потрібно для проїзду автомобілів, “Count of transport” вказує на кількість транспортних засобів на осі, яка є відкрита для проїзду і “Actual time” вказує не час, який відведено для проїзду.

Під списком знаходиться відображення системи в конкретний період часу. Цифри біля сторін означають ідентифікатор потоку і відповідають до заголовків відео-файлів на рисунку 4.5. Також, для умовних позначень використовується “o” і “x”. Символ “o” означає, що сторони вільні для проїзду, а “x” - сторони закриті для проїзду.

Зауважте, що потрібний і відведений час для трьох перших ітерацій на рисунку 4.6. відрізняється, це зумовлене тим, що потрібний час - 90 секунд, а максимальний - 50, тому відведеному часу було присвоєно 50 секунд. Також

можна побачити, що четверта ітерація має інший результат ніж перших три, така поведінка зумовлена тим, що максимальна безперервна кількість позитивних ітерацій для однієї осі не має перевищувати трьох разів. Тобто, три рази вісь у була відкрита для проїзду, четвертий раз - x.

4.2 Додаткові функції програми

Програма записує файл app.log, який зберігає в собі стан світлофорної ділянки, кількість машин на кожну вісь, час на проходження для відкритої сторони. На рисунку 4.8 зображений файл app.log.

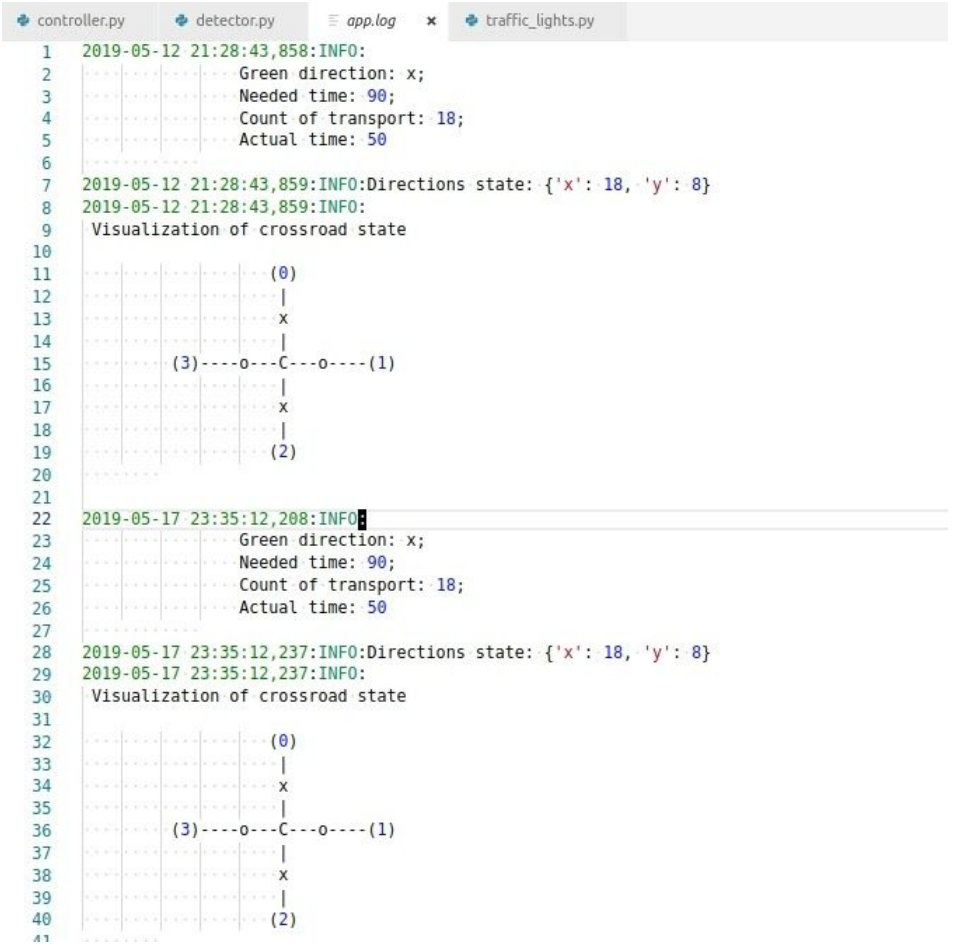


Рисунок 4.8 – Файл-журнал app.log

При записування відео-потоку, програма зберігає результуючий відео-файл в файловій системі, результат на рисунку 4.9.

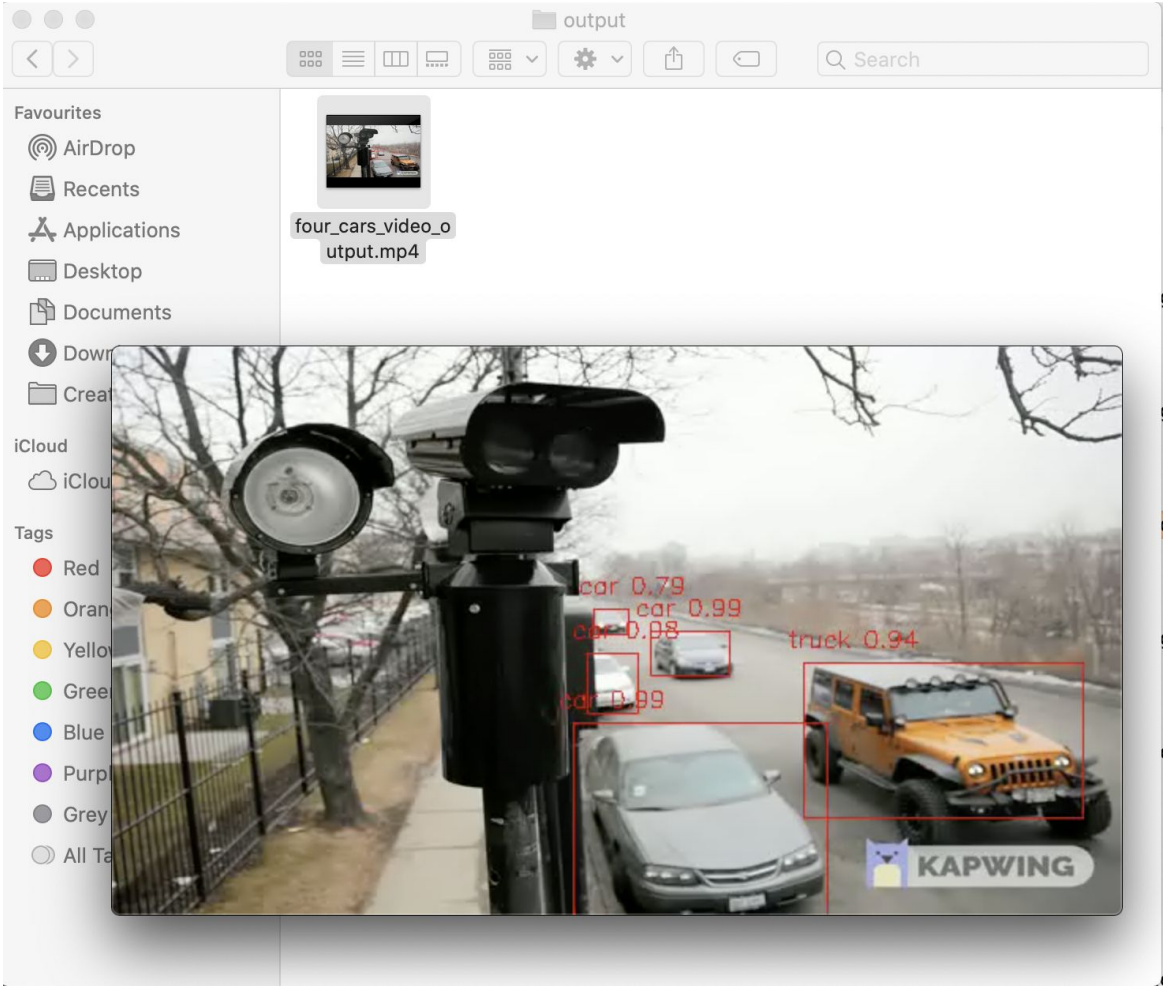


Рисунок 4.9 – Результуючий файл запису відео-потоку

Наступною додатковою функцією є можливість ручного керування. Використання такої можливості рекомендується при високому рівні завантаженості доріг. Немає потреби використовувати таку можливість при майже пустих дорогах, оскільки є можливість, що саме в час затримки система пропустить поодинокі ТЗ на високій швидкості, що може призвести до ДТП.

Було також добавлено окрему система на випадок надзвичайних ситуацій, коли потрібно заблокувати чи розблокувати якусь конкретну ділянку дороги.

Нижче показана схема роботи ручного керування світлофорної ділянкою, інтерфейс - це три кнопки (рисунок 4.10):

1. Activate X Axis.
2. Activate Y Axis.
3. Activate Auto Resolve.

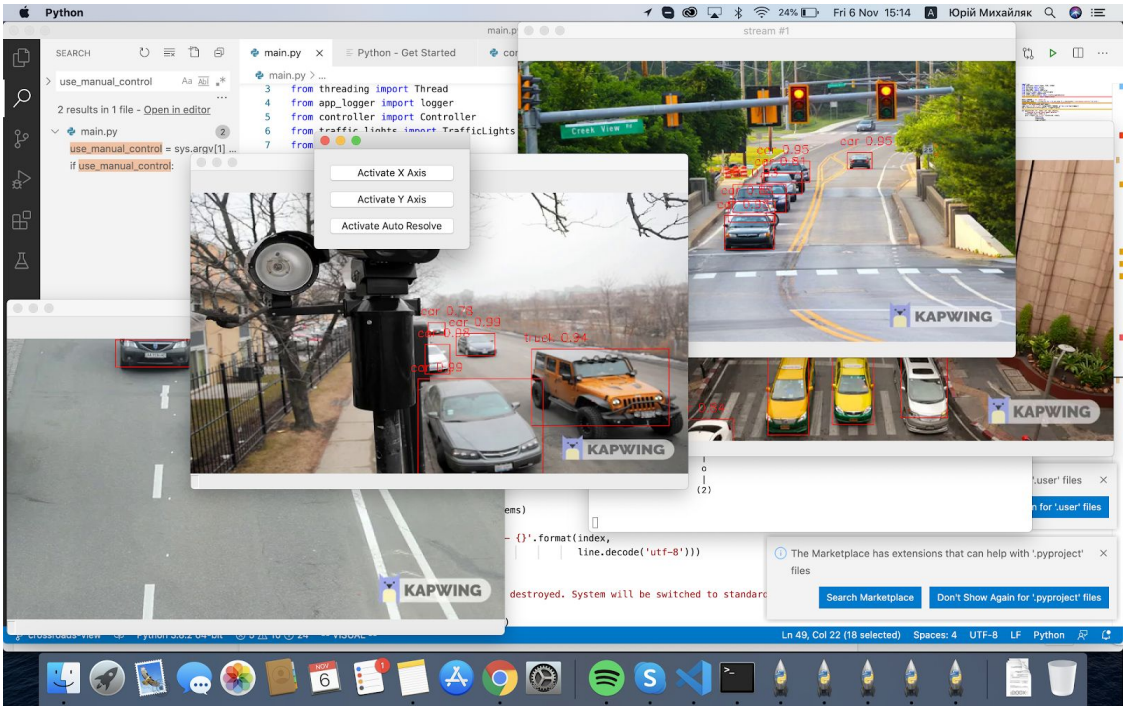


Рисунок 4.10 – UI для ручного керування станом світлофора

Користувачу надається можливість працювати в ручному або автоматизованому режимі. “Active X Axis” і “Active Y Axis” переводять систему в ручний режим і не в залежності від даних буде доступна та ділянка дороги, яку обрав контролер. На рисунку 4.11 показано, що при таких ж даних, як на рисунку 4.5, користувач зумів змінити зелену ділянку з осі Y на X.

Оскільки натискання на одну з перших двох фото переводить систему в ручний режим, то при потребі повернення її в автоматичний можна скористатись допомогою кнопки “Activate Auto Resolve”.

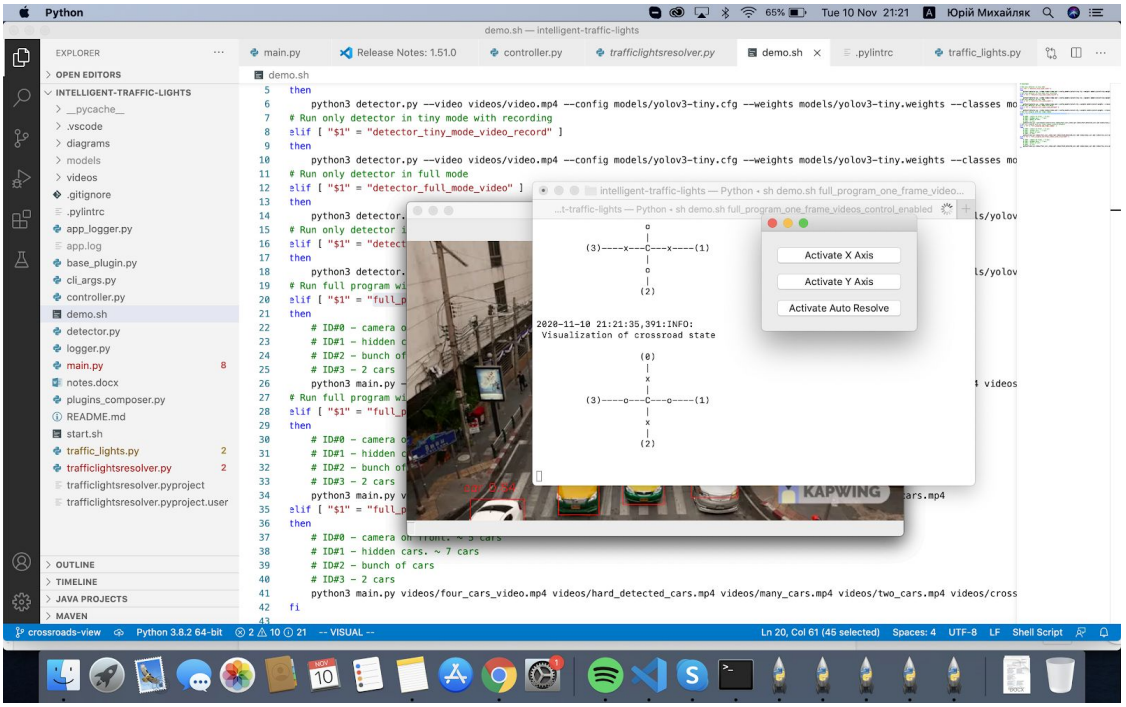


Рисунок 4.11 – Зміна зеленої ділянки дороги за допомогою ручного керування

Висновки до розділу 4

Проведена експериментальна апробація результатів роботи реалізованої інформаційної системи регулювання дорожнього трафіку в реальному часі. Як приклад був виконаний аналіз однофреймових відео. Для оцінки ефективності роботи системи виконано аналіз ряду зображень з кількістю об’єктів для ідентифікації (автомобілів) складала від 2 до ЧИСЛО. У випадку застосування важкого режиму в усіх проаналізованих модельних ситуаціях було отримано повне співпадіння, відсутні випадки хибної ідентифікації та випадки пропущених об’єктів для ідентифікації. При застосуванні легкого режиму, який потребує суттєво менших апаратних ресурсів, мало місце до 40% пропущених об’єктів.

Загальні висновки

На основі проведеного аналізу літературних джерел визначено суттєву та зростаючу потребу у регулюванні дорожнього трафіку, що обумовлено стрімким збільшенням автопарку у поєднанні з відставанням розвитку інфраструктури. Розвиток і впровадження інформаційних технологій, та інформаційних систем створює широкі можливості для застосування нових, більш ефективних підходів до регулювання дорожнього руху, зокрема використання розумних світлофорів. Наявна дорожня інфраструктура є достатньо розгалуженою, тому забезпечення достатньо рівня економічної ефективності впроваджуваних інструментів регулювання руху повинні передбачати мінімальне втручання у вже наявні та розгорнуті системи.

Реалізовано інформаційну системи розумного світлофора для регулювання дорожнього трафіку з використанням мультиканального режиму прийому даних, та:

- проведено аналіз існуючих методів, технологій та рішень регулювання дорожнього трафіку;
- удосконалено існуючі методи регулювання дорожнього трафіку з використанням розумного світлофора у напрямку полегшення їх інтегрування в наявну транспортну інфраструктуру;
- розроблено інформаційну систему розумного світлофора для регулювання дорожнього трафіку за допомогою отриманих моделей та методів;
- виконано експериментальну перевірку інформаційної системи регулювання дорожнього трафіку.

Побудовано інформаційну модель системи розумного світлофора з відкритою архітектурою.

Визначені та описані основні модулі інформаційної системи, включно із класами та взаємодія між ними. Описано макети користувацького інтерфейсу та принцип їх побудови. Представлено схему функцій користувача.

Проведена експериментальна апробація результатів роботи реалізованої інформаційної системи регулювання дорожнього трафіку в реальному часі. Як приклад був виконаний аналіз однофреймових відео. Для оцінки ефективності роботи системи виконано аналіз ряду зображень з кількістю об’єктів для ідентифікації (автомобілів) складала від 2-ох до 13-ти. У випадку застосування важкого режиму в усіх проаналізованих модельних ситуаціях було отримано повне співпадіння, відсутні випадки хибної ідентифікації та випадки пропущених об’єктів для ідентифікації. При застосуванні легкого режиму, який потребує суттєво менших апаратних ресурсів, мало місце до 40% пропущених об’єктів, тому він не може бути рекомендований для практичного застосування.

Перелік посилань

1. Правила дорожнього руху. Режим доступу: http://pdd.ua/ua/8/
2. Загальні положення про світлофор. Режим доступу: https://uk.wikipedia.org/wiki/%D0%A1%D0%B2%D1%96%D1%82%D0%BB%D0%BE%D1%84%D0%BE%D1%80
3. Основи світлофорного регулювання. Режим доступу: http://ea.donntu.org:8080/jspui/bitstream/123456789/28329/4/%D0%A2%D0%A1%D0%9E%D0%94%D0%94_%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_3.pdf
4. Аналог системи розумного світлофора. Режим доступу: https://github.com/Twitwi96/Smart-traffic-light-2
5. Другий аналог системи розумного світлофора. Режим доступу: https://github.com/JayLohokare/smart-traffic-signals
6. Зображення двох автівок. Режим доступу: https://www.shutterstock.com/ru/video/clip-12934481-bangkok---march-24-2015-cars-being
7. Зображення п’яти автівок. Режим доступу: https://www.chicagotribune.com/business/ct-biz-mayoral-candidates-transportation-getting-around-20190129-story.html
8. Зображення великої кількості автівок. Режим доступу: https://www.shutterstock.com/ru/video/clip-12934481-bangkok---march-24-2015-cars-being
9. Зображення дев’яти автівок одна за одною. Режим доступу: https://www.newarkpostonline.com/news/new-paper-mill-road-traffic-signal-part-of-broader-safety-plan/article_e35b6719-a81c-548c-9975-8cc2582a1365.html
10. Схеми основних термінів режиму регулювання. Режим доступу: http://ea.donntu.org:8080/jspui/bitstream/123456789/28329/4/%D0%A2%D0%A1%D

0%9E%D0%94%D0%94_%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_3.pdf

11. Грицунов О. В. Інформаційні системи та технології: навч. посіб. для студентів за напрямом підготовки «Транспортні технології» / О. В. Грицунов; Харк. нац. акад. міськ. госп-ва. – Х.: ХНАМГ

12. Принцип роботи YOLO. Режим доступу: https://appsilon.com/wp-content/uploads/2018/08/types.png

13. QT Widget схема. Режим доступу: https://doc.qt.io/qt-5/images/parent-child-widgets.png

14. Шаблон “Observer”. Режим доступу: https://upload.wikimedia.org/wikipedia/commons/0/01/W3sDesign_Observer_Design_Pattern_UML.jpg

15. Павлиш В. А. Основи інформаційних технологій і систем: Навчальний посібник. / Павлиш В. А., Гліненко Л. К. - Львів: Видавництво Львівської політехніки, 2013. − 500 с

16. Обробка геофізичних сигналів у сучасних автоматизованих комплексах: Навчальний посібник / В.А. Кирилюк, М.Ф. Пічугін, О.А. Машков та ін. – Житомир: ЖВІРЕ, 2007. – 176 с

17. Fast heuristics for large geometric traveling salesman problems. / Reinelt, Gerhard (1992) // ORSA Journal on computing, 4:206-217

18. Годун В.М. Інформаційні системи і технології в статистиці: навч. посіб. / В.М. Годун, Н.С. Орленко, М. А. Сендзюк; за ред. В.Ф. Ситника. – К.: КНЕУ, 2003. – 267 с.

19. Hemming R. V. Theory of coding and theory of information / R. V. Hemming // Trans. eng. – M.:Radio & communication, 1983. – 176 p.

20. Patrick E. Fundamentals of the theory of pattern recognition. Moscow: Sov.radio, 1980 .

21. Lainiotis D. G. A Class of Upper Bounds on Probability of Error for Multi –Hypotheses Pattern Recognition // IEEE Transaction on Information Theory, IT-15, №5, 1969

22. QtWdiget. Режим доступу: https://doc.qt.io/qt-5/qtwidgets-index.html

23. OpenCV. Режим доступу: https://opencv.org/

24. OpenCV Detection. Режим доступу: https://docs.opencv.org/master/d2/d64/tutorial_table_of_content_objdetect.html

25. Python. Режим доступу: https://www.python.org/

26. Python GUI. Режим доступу: https://www.tutorialspoint.com/python/python_gui_programming.htm

27. Yolo object detection. Режим доступу: https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/

28. Python vs Java. Режим доступу: https://raygun.com/blog/java-vs-python/

29. Object detection. Режим доступу: https://www.fritz.ai/object-detection/

30. Traffic lights logic. Режим доступу: https://www.sciencedirect.com/science/article/pii/B9780128153024000029

31. Правила дорожнього руху. Режим доступу: https://vodiy.ua/pdr/

32. Tiny Yolo. Режим доступу: https://www.pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-ncs/

33. Darknet neural network. Режим доступу: https://pjreddie.com/darknet/

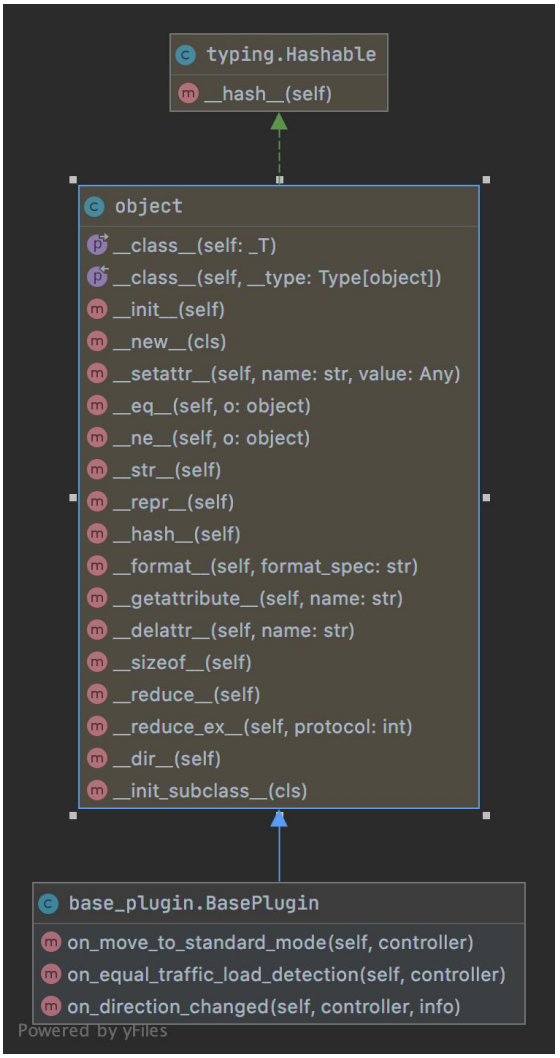
34. COCO dataset. Режим доступу: https://cocodataset.org/#home

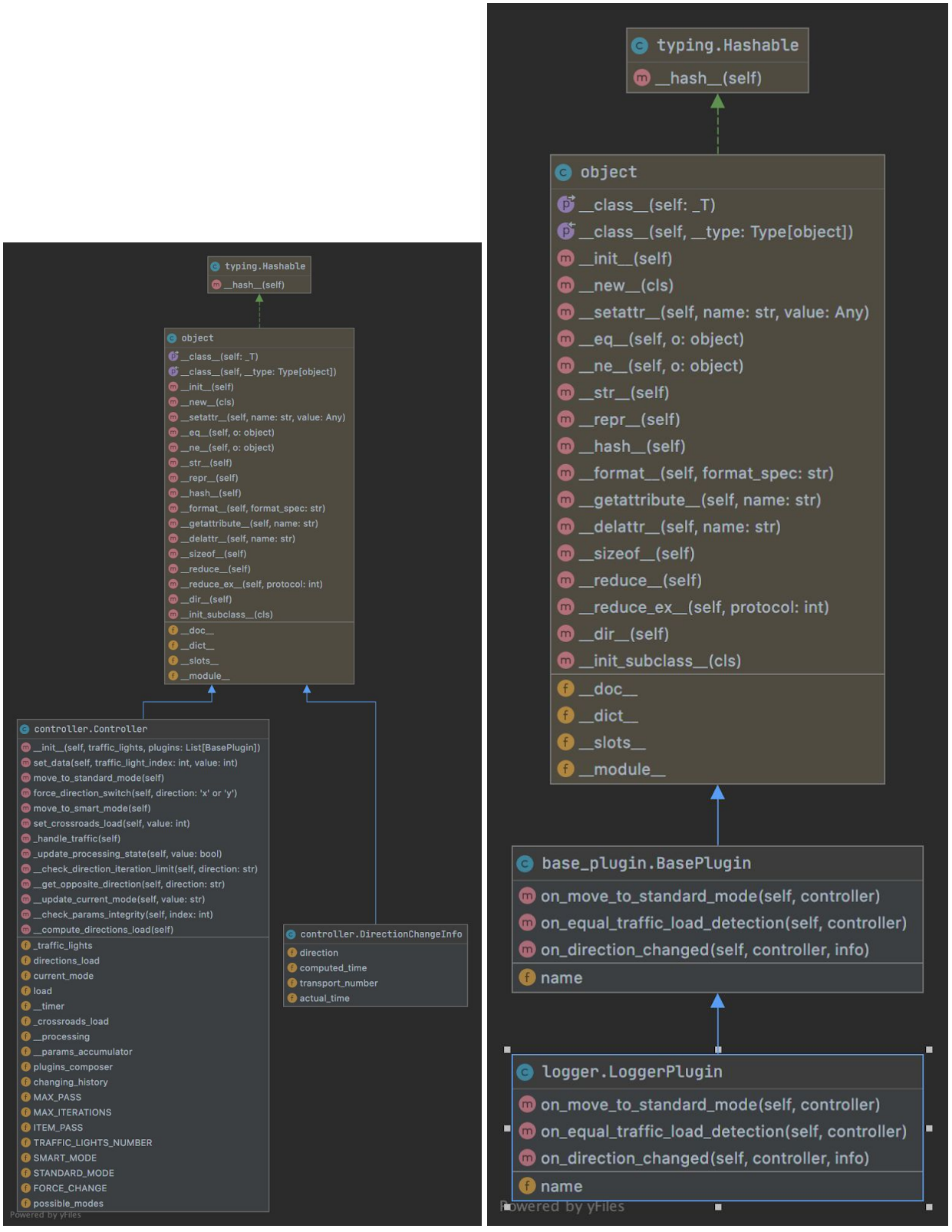
35. Ю.Б. Михайляк, О.А. Пасічник, Т.К. Скрипник. Інформаційна система розумного світлофора для регулювання дорожнього трафіку // Вісник Хмельницького національного університету. 2020 (подано до редакції).

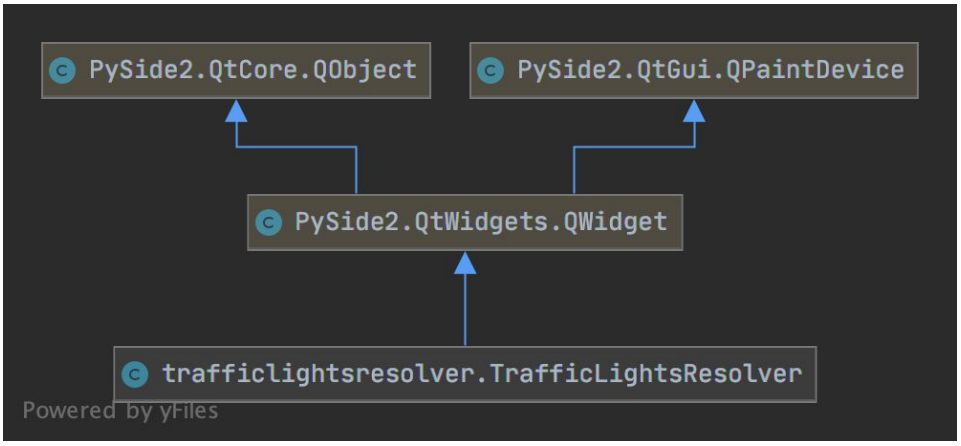
ДОДАТКИ

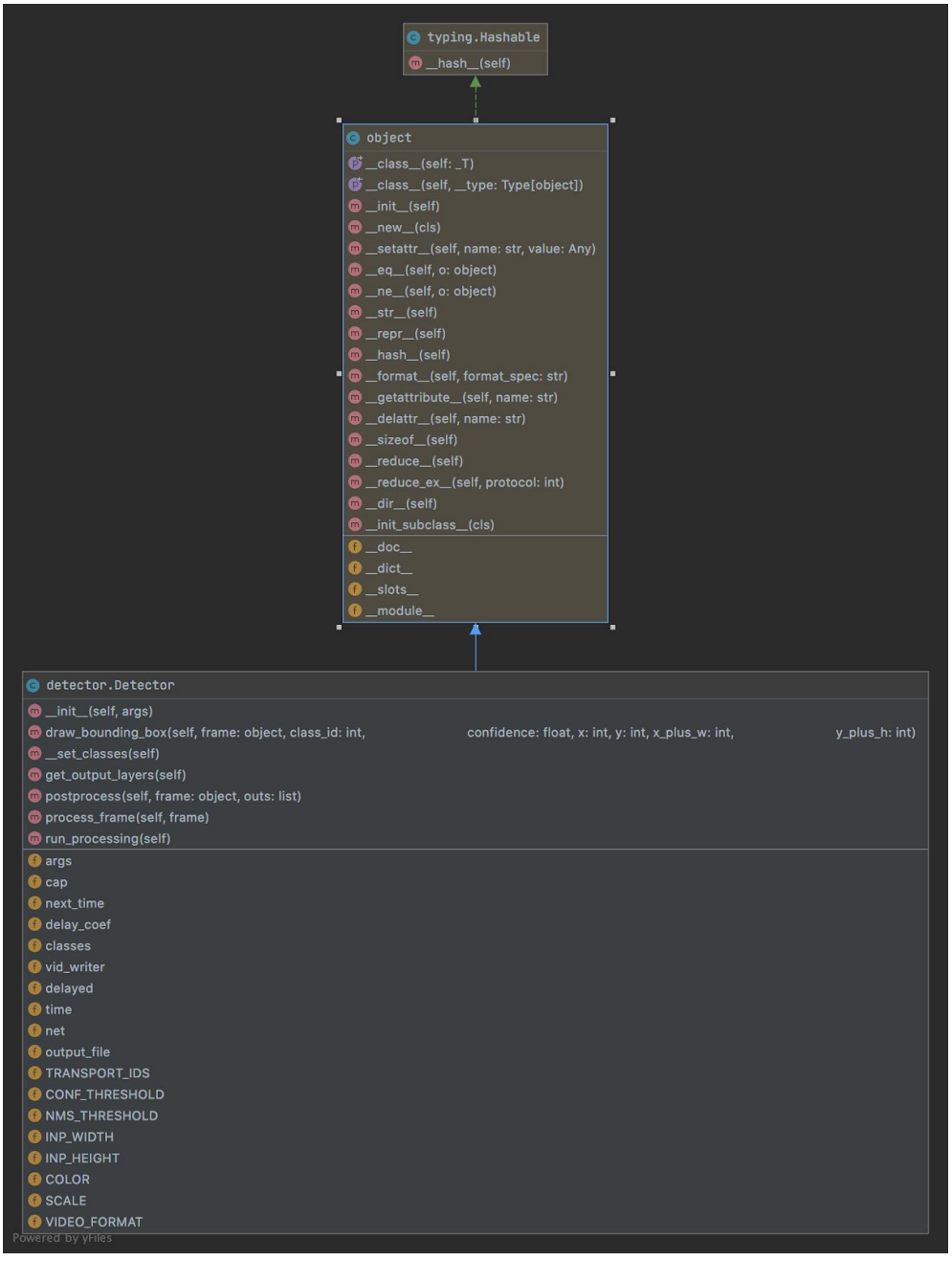
Додаток А

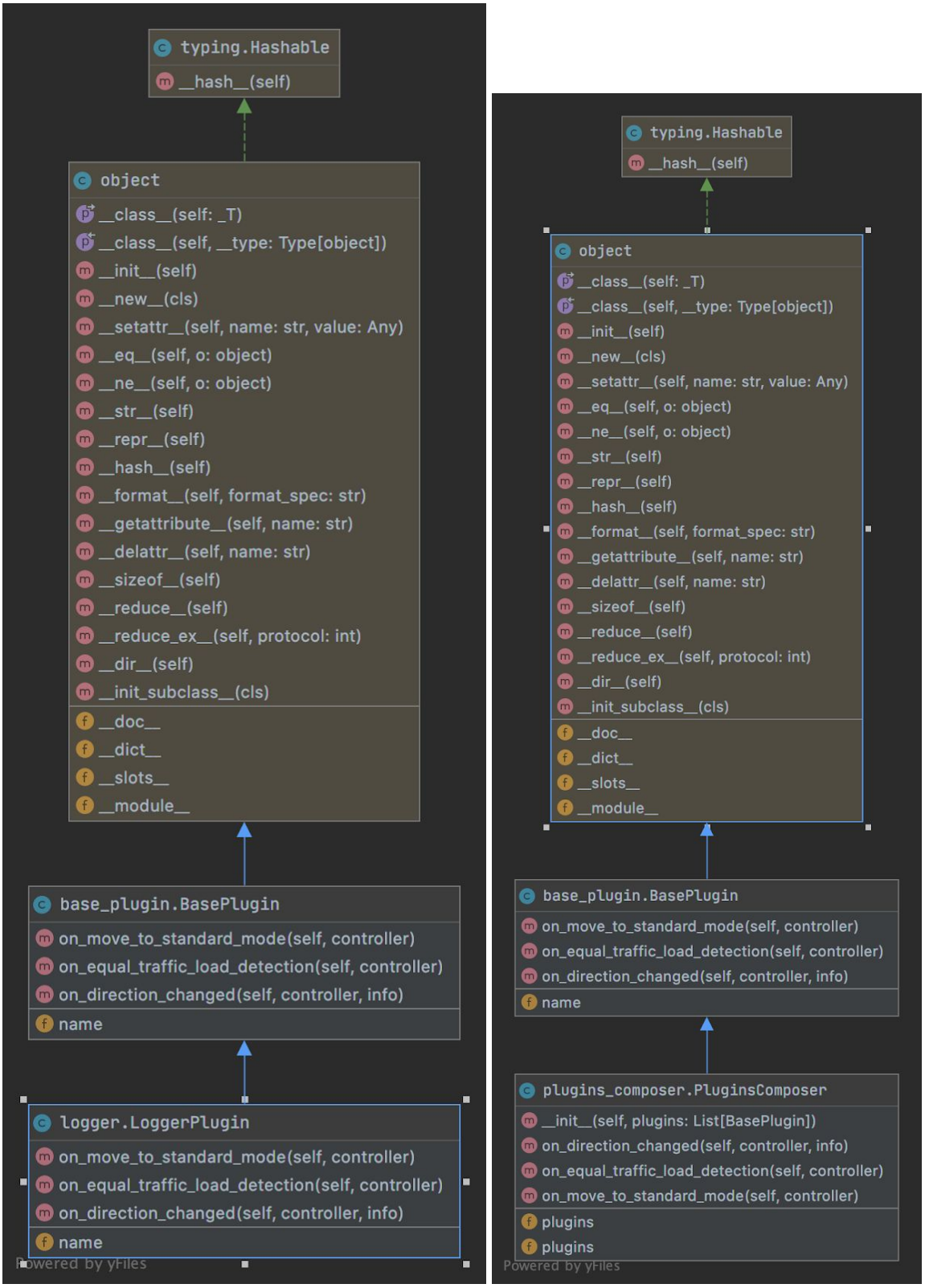
Діаграми класів




typing.Hashable
__hash__(self)
object
__class__(self: _T)
__class__(self, __type: Type[object])
__init__(self)
__new__(cls)
__setattr__(self, name: str, value: Any)
__eq__(self, o: object)
__ne__(self, o: object)
__str__(self)
__repr__(self)
__hash__(self)
__format__(self, format_spec: str)
__getattribute__(self, name: str)
__delattr__(self, name: str)
__sizeof__(self)
__reduce__(self)
__reduce_ex__(self, protocol: int)
__dir__(self)
__init_subclass__(cls)
__doc__
__dict__
__slots__
__module__
controller.Controller
__init__(self, traffic_lights, plugins: List[BasePlugin])
set_data(self, traffic_light_index: int, value: int)
move_to_standard_mode(self)
force_direction_switch(self, direction: 'x' or 'y')
move_to_smart_mode(self)
set_crossroads_load(self, value: int)
_handle_traffic(self)
_update_processing_state(self, value: bool)
__check_direction_iteration_limit(self, direction: str)
__get_opposite_direction(self, direction: str)
__update_current_mode(self, value: str)
__check_params_integrity(self, index: int)
__compute_directions_load(self)
_traffic_lights
directions_load
current_mode
load
__timer
_crossroads_load
__processing
__params_accumulator
plugins_composer
changing_history
MAX_PASS
MAX_ITERATIONS
ITEM_PASS
TRAFFIC_LIGHTS_NUMBER
SMART_MODE
STANDARD_MODE
FORCE_CHANGE
possible_modes
controller.DirectionChangeInfo
direction
computed_time
transport_number
actual_time
base_plugin.BasePlugin
on_move_to_standard_mode(self, controller)
on_equal_traffic_load_detection(self, controller)
on_direction_changed(self, controller, info)
name
logger.LoggerPlugin
on_move_to_standard_mode(self, controller)
on_equal_traffic_load_detection(self, controller)
on_direction_changed(self, controller, info)
name
Powered by yFiles


PySide2.QtCore.QObject
PySide2.QtGui.QPaintDevice
PySide2.QtWidgets.QWidget
trafficlightsresolver.TrafficLightsResolver
Powered by yFiles


typing.Hashable
__hash__(self)
object
__class__(self: _T)
__class__(self, __type: Type[object])
__init__(self)
__new__(cls)
__setattr__(self, name: str, value: Any)
__eq__(self, o: object)
__ne__(self, o: object)
__str__(self)
__repr__(self)
__hash__(self)
__format__(self, format_spec: str)
__getattribute__(self, name: str)
__delattr__(self, name: str)
__sizeof__(self)
__reduce__(self)
__reduce_ex__(self, protocol: int)
__dir__(self)
__init_subclass__(cls)
__doc__
__dict__
__slots__
__module__
detector.Detector
__init__(self, args)
draw_bounding_box(self, frame: object, class_id: int, confidence: float, x: int, y: int, x_plus_w: int, y_plus_h: int)
__set_classes(self)
get_output_layers(self)
postprocess(self, frame: object, outs: list)
process_frame(self, frame)
run_processing(self)
args
cap
next_time
delay_coef
classes
vid_writer
delayed
time
net
output_file
TRANSPORT_IDS
CONF_THRESHOLD
NMS_THRESHOLD
INP_WIDTH
INP_HEIGHT
COLOR
SCALE
VIDEO_FORMAT
Powered by yFiles


typing.Hashable
__hash__(self)
object
__class__(self: _T)
__class__(self, __type: Type[object])
__init__(self)
__new__(cls)
__setattr__(self, name: str, value: Any)
__eq__(self, o: object)
__ne__(self, o: object)
__str__(self)
__repr__(self)
__hash__(self)
__format__(self, format_spec: str)
__getattribute__(self, name: str)
__delattr__(self, name: str)
__sizeof__(self)
__reduce__(self)
__reduce_ex__(self, protocol: int)
__dir__(self)
__init_subclass__(cls)
__doc__
__dict__
__slots__
__module__
base_plugin.BasePlugin
on_move_to_standard_mode(self, controller)
on_equal_traffic_load_detection(self, controller)
on_direction_changed(self, controller, info)
name
logger.LoggerPlugin
on_move_to_standard_mode(self, controller)
on_equal_traffic_load_detection(self, controller)
on_direction_changed(self, controller, info)
name
Powered by yFiles
typing.Hashable
__hash__(self)
object
__class__(self: _T)
__class__(self, __type: Type[object])
__init__(self)
__new__(cls)
__setattr__(self, name: str, value: Any)
__eq__(self, o: object)
__ne__(self, o: object)
__str__(self)
__repr__(self)
__hash__(self)
__format__(self, format_spec: str)
__getattribute__(self, name: str)
__delattr__(self, name: str)
__sizeof__(self)
__reduce__(self)
__reduce_ex__(self, protocol: int)
__dir__(self)
__init_subclass__(cls)
__doc__
__dict__
__slots__
__module__
base_plugin.BasePlugin
on_move_to_standard_mode(self, controller)
on_equal_traffic_load_detection(self, controller)
on_direction_changed(self, controller, info)
name
plugins_composer.PluginsComposer
__init__(self, plugins: List[BasePlugin])
on_direction_changed(self, controller, info)
on_equal_traffic_load_detection(self, controller)
on_move_to_standard_mode(self, controller)
plugins
plugins
Powered by yFiles


typing.Hashable
__hash__(self)
object
__class__(self: _T)
__class__(self, __type: Type[object])
__init__(self)
__new__(cls)
__setattr__(self, name: str, value: Any)
__eq__(self, o: object)
__ne__(self, o: object)
__str__(self)
__repr__(self)
__hash__(self)
__format__(self, format_spec: str)
__getattribute__(self, name: str)
__delattr__(self, name: str)
__sizeof__(self)
__reduce__(self)
__reduce_ex__(self, protocol: int)
__dir__(self)
__init_subclass__(cls)
__doc__
__dict__
__slots__
__module__
traffic_lights.TrafficLights
__init__(self)
turn_on_lights(self, dir: str)
switch_lights(self)
visualize_state(self)
state
Powered by yFiles

Додаток Б

Ксерокопії наукових публікацій, виконаних при роботі над дипломною роботою магістра

УДК 004.931+004.932+656.054

Ю.Б. МИХАЙЛЯК, О.А. ПАСІЧНИК, Т.К. СКРИПНИК,
Хмельницький національний університет

ІНФОРМАЦІЙНА СИСТЕМА РОЗУМНОГО СВІТЛОФОРА ДЛЯ РЕГУЛЮВАННЯ ДОРОЖНЬОГО ТРАФІКУ

Реалізована інформаційна технологія регуляції дорожнього трафіку. Програмний застосунок забезпечує керування світлофором на перехресті базуючись на завантаженості доріг. Завантаженість доріг визначається кількістю транспортних засобів на ділянці. Додатковими функціями є логування стану світлофора і зберігання відео в файловій системі комп'ютера. Реалізовано інтеграцію з сторонніми бібліотеками.

YU.B. MYKHAILIAK, O.A. PASICHNYK, T.K. SKRYPNYK
Khmelnytsky National University

**PLANNING INFORMATION SYSTEM
FOR THE BEST WAY TO DELIVER CARGO
WITH THE HELP OF THE TRAVELER'S TASK**

Today, cars are as much a part of our lives as a luxury. It is difficult to understand that the first cars were perceived as a manifestation of evil spirits. A little later, cars were available only to the rich. Now cars cover all parts of our society. The main reason for the popularity of cars was the ability to be mobile in a given situation. Undoubtedly, your own car provides a lot of time.

Cars are also used in businesses of various sizes, the main field of use - transportation of goods, transportation of people and others.

The development of road surface is developing in parallel with consumers. In the civilized world, people began to use means to prevent traffic accidents, so there were traffic rules, road markings, lying police and road signs. Probably, one of the main attributes of traffic regulation is a traffic light - a device that gives a certain lane permission to move, while blocking the movement perpendicular in order to avoid a traffic accident. The traffic light mechanism is quite simple, it does its job well.

It is a clear fact that with the growth of cars, the load on the roads increases. Often the result of heavy traffic is congestion, which causes a significant difference between the expected time and the actual time of getting from point A to point B. Significant congestion is observed in large, densely populated cities. The peak time for such cities is 9:00 and 18:00, at which time residents go or return from work.

You can often see that some minibuses are more popular than others. It is assumed that the load on one axle is much greater than the other, usually in such situations, the waiting time for the "green light" for the unloaded axle is inefficient. Since, in our period - time is an exhausted resource, we need to optimize the work of small things that could save on it.

In this case, the automated traffic manager is useful because the system can analyze the input data and based on them to build the optimal state of the traffic light.

Keywords: traffic, traffic lights, vehicles, traffic jams, optimization, vehicles detection.

Аналіз предметної області

Світлофорний об'єкт – територія дорожнього сполучення, на котрому послідовність пересування конфліктуючих автомобільних потоків або автомобільних і пішохідних потоків врегульовується світлофорною сигналізацією.

Основні поняття авто-світлофора:

- такт – період виконання конкретної комбінації індикаторів;
- головний такт – хронологічний період, під час котрого пересування транспортних засобів (інколи й пішохідних потоків) є дозволеним;
- проміжний такт – хронологічний період, під час котрого відбувається підготовка до перемикання дозволу на пересування наступній групі автомобілів (інколи й пішохідних потоків);
- фаза регулювання – сукупність головного такту й проміжного такту;
- регулювальний цикл – послідовність фаз яка повторюється з конкретною періодичністю;
- режим світлофорного регулювання – к-сть часу вказаних фаз, тактів і тривалість регулювального циклу;
- схема роз'їзду – візуальне хронологічне представлення конфліктних автомобільних потоків;
- циклограма регулювання світлофора – візуальна схема послідовності й тривалості горіння індикаторів світлофорів на всіх можливих маршрутах, розташованих на вулиці.

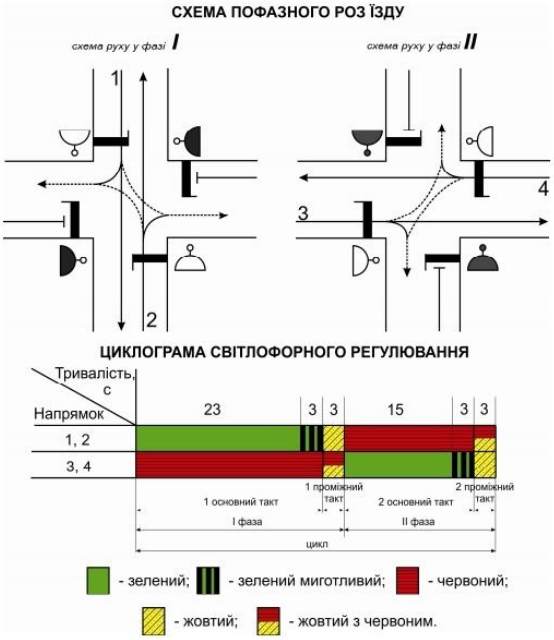


Рисунок 1.1 – Схеми основних термінів режиму регулювання світлофором[2]

Незважаючи, на простоту й надійність механізму світлофорного регулювання, можна побачити значний недолік, а саме - неможливість регулювання трафіку на основі вхідних зовнішніх даних. Під зовнішніми даними мається на увазі такі елементи:

- завантаженість доріг;
- тип транспорту;

Аналіз існуючого програмного забезпечення предметної област

На даний час глобальних розробок по системі розумної регуляції руху в Україні не ведеться. Існує кілька ідей на ресурсах для стартапів, які чекають інвестицій і пропонують розумну регуляцію руху на основі завантаженості доріг потоком автомобілів. Також, є опис, що розвиваючи цю ідею можна добавити розумну детекцію правопорушень і виписку штрафів.

Проте, за межами українського інтернет-простору можна знайти кілька схожих проектів, які розташовані на відкритому програмному ресурсі - *github.com*. Реалізації цих проектів мають свою плюси чи мінуси, як і будь-яке ПО. Перший з таких “Smart-traffic-light-2”[1].

Рисунок 1.2 – Аналог системи розумного світлофора[3]

Проект був розроблений в рамках Бельгійського університету.
Плюси цього проекту:

1. Система інтегрована на платформу Raspberry Pi 3 з підключеною USB камерою.
2. Розпізнавання меж дороги, для точнішого обрахунки вхідних даних.
3. Код реалізований на мові Python, що дає легку підтримку і гнучкість в добавленні нових функціональних речей.
4. Можливість надання пріоритетності для спец. засобів (швидка допомога, поліція, тощо.)

Мінуси проекту:

1. Визначення дорожнього транспорту ведеться з камери, яка може розташовуватися лише зверху.
2. Неможливість використовувати по одній камері на одну сторону.
3. Визначення автомобілів ведеться за технологією HOG (Histogram of Oriented Gradients), що може призвести до неочікуваної поведінки під час негоди, тощо.

Ще одним прикладом є схожий проект, який реалізований на C++ “smart-traffic-signals”[4]. Реалізації проекту є доволі низькорівневою, що робить його надзвичайно гнучким.
Плюси проекту:

1. C++ надає велику гнучкість редагування проекту.
2. C++ надає великий приріст в швидкості дії.
3. Інтеграція з RaspberryPi 2.

Мінуси:

1. C++ мова з низьким рівнем абстракції, тому навіть для реалізації банальних речей потрібно більше часу, ніж до прикладу на Python.
2. Визначення автомобілів ведеться за технологією Haar Cascade, що може призвести до неочікуваної поведінки під час негоди, тощо.

Поточний проект позбавлений деяких вад, які є описані в аналогах вище, а саме: використовується проста платформа для підтримки коду, технологія розпізнавання автомобілів ведеться за допомогою сторонньої бібліотеки, що зменшує кодову базу. Конструктивно проект простий тим, що потребує монтування чотирьох камер на опори світлофорів (в першому аналозі спостереження дозволяються лише зверху, що в кілька разів збільшує кошторис реалізації проекту).

Описання бізнес-логіки

Якщо розглядати роботу звичайного світлофора, то всюди можна побачити систему, яка з періодичністю перемикає індикатори для проходження руху на тій чи іншій ділянці руху. Трапляються ситуації, коли працівникам потрібно відновити історію дії (наприклад ДТП), яка відбулася якийсь час назад. Єдиним варіантом цього досягнути є отримання даних з сусідніх камер будівель, що уповільнює аналіз в кілька разів, так як не на всіх будівлях є камери, також потрібно всі ці дані співставляти.

Інша дія, якій потрібно автоматизація - збір інформації про поломку світлофора. Якщо світлофор вийшов з ладу, маючи інформацію про минулі дії і помилки в системі, тех. працівнику не складе зусиль полагодити проблему.

Для того, щоб автоматизувати роботу збору інформації, дані потрібно організувати у файловій системі. На даний час потреби в створенні бази даних немає, так як системи не є централізованою, на даний час. В подальшому, є сенс перевести зберігання інформації в БД.

На даному етапі аплікація пише два файли під час виконання, перший файл - це логування, в логуванні можна переглянути стан системи в конкретний відрізок часу, також, якщо виникнуть якісь помилки, вони миттєво запишуться в *app.log* файл, який саме й слугує для зберігання інформації. Другим файлом є відео-файл, який записується з відеопотоку, єдина відмінність відеопотоку, від результуючого файлу є те, що на фінальному відео буде накладатися система визначення автомобілів в потоці.

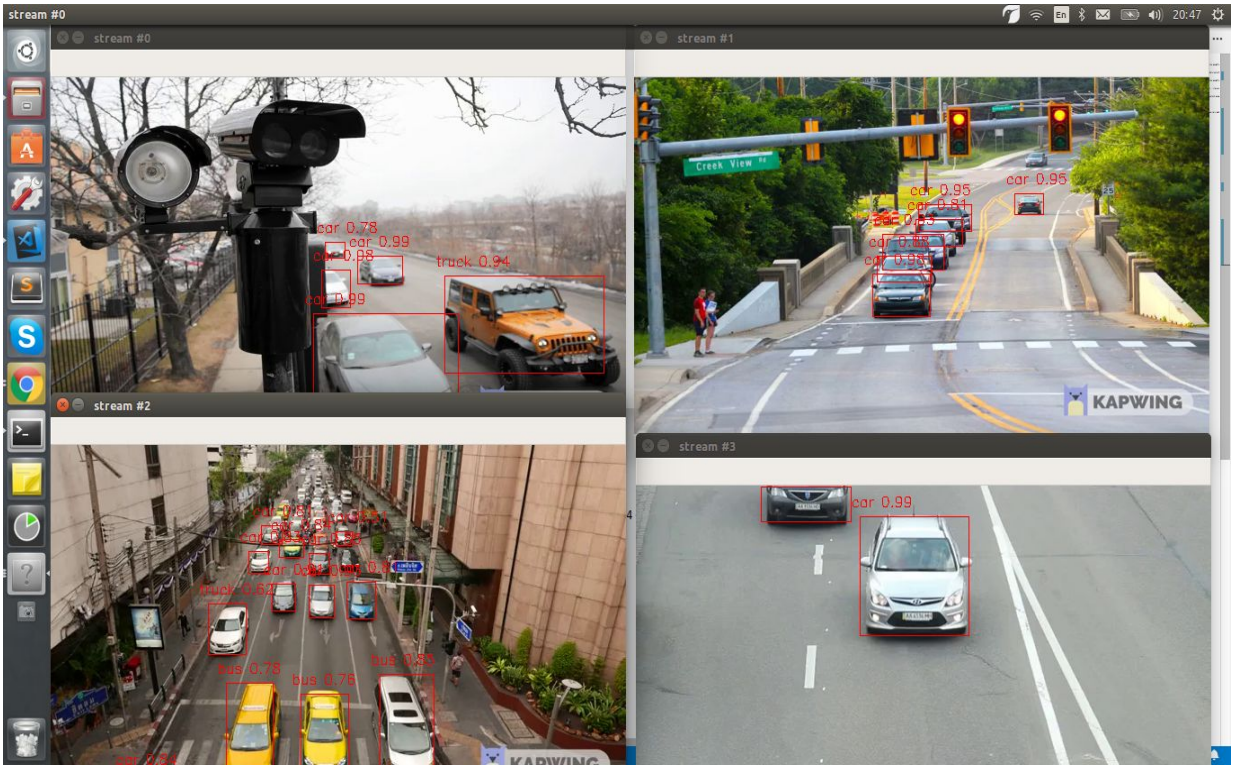


Рисунок 2.1 – відео-файл, який записується з відеопотоку

```
controller.py   detector.py   app.log   traffic_lights.py

2019-05-12 21:28:43,858:INFO:
                Green direction: x;
                Needed time: 90;
                Count of transport: 18;
                Actual time: 50

2019-05-12 21:28:43,859:INFO:Directions state: {'x': 18, 'y': 8}
2019-05-12 21:28:43,859:INFO:
 Visualization of crossroad state

                (0)
                 |
                 x
                 |
        (3)----o---C---o----(1)
                 |
                 x
                 |
                (2)


2019-05-17 23:35:12,208:INFO:
                Green direction: x;
                Needed time: 90;
                Count of transport: 18;
                Actual time: 50

2019-05-17 23:35:12,237:INFO:Directions state: {'x': 18, 'y': 8}
2019-05-17 23:35:12,237:INFO:
 Visualization of crossroad state

                (0)
                 |
                 x
                 |
        (3)----o---C---o----(1)
                 |
                 x
                 |
                (2)
```

Рисунок 2.2 – Логування для системи регулювання

Створення нових інформаційних технологій має велике значення для розвитку суспільства. Вони активно перетворюють інші технології матеріального і нематеріального виробництва, в кінцевому підсумку формуючи новий стиль роботи, спосіб життя в цілому. Суть інформаційних технологій становлять методи і засоби формування та підтримки інформаційних потоків у системах управління об'єктами, у тому числі регулювання дорожнього трафіку.

Інша дія, якій потрібно автоматизація - збір інформації про поломку світлофора. Якщо світлофор вийшов з ладу, маючи інформацію про минулі дії і помилки в системі, тех. працівнику не складе зусиль полагодити

проблему.

Для поточної роботи, як і у всіх програмах є споживач (клієнт). В цьому випадку користувачем програми є державна дорожня служба / приватне підприємство. Схема доступних функцій для гостя (клієнта) зображено на рисунку 2.3.

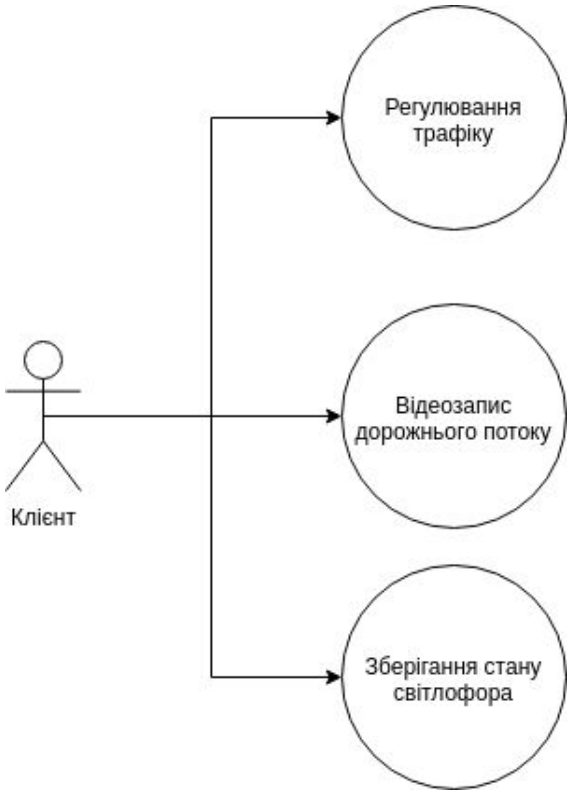


Рисунок 2.3 – Можливості користувача

Загальну схему регулювального алгоритму системи розумного світлофора представлено на рисунку 2.4.

Алгоритм визначення об’єктів на фреймі, буде визначати засоби транспорту, конкретна реалізації реагує лише автомобілі різних габаритів і мотоцикли, можливість визначення сторонніх предметів - відключена.

Як тільки автомобілі були знайдені, система спробує просумувати автомобілі відповідно до осей: x і y, пізніше визначається сторона, яка вважається більш завантаженою, прохід дозволяється завантаженій стороні, при тому з конкретним часом, який залежить від кількості автомобілів, цей час не має перевищувати максимальний час проходження. Зелене світло для одного потоку може повторитися лише 3 рази.

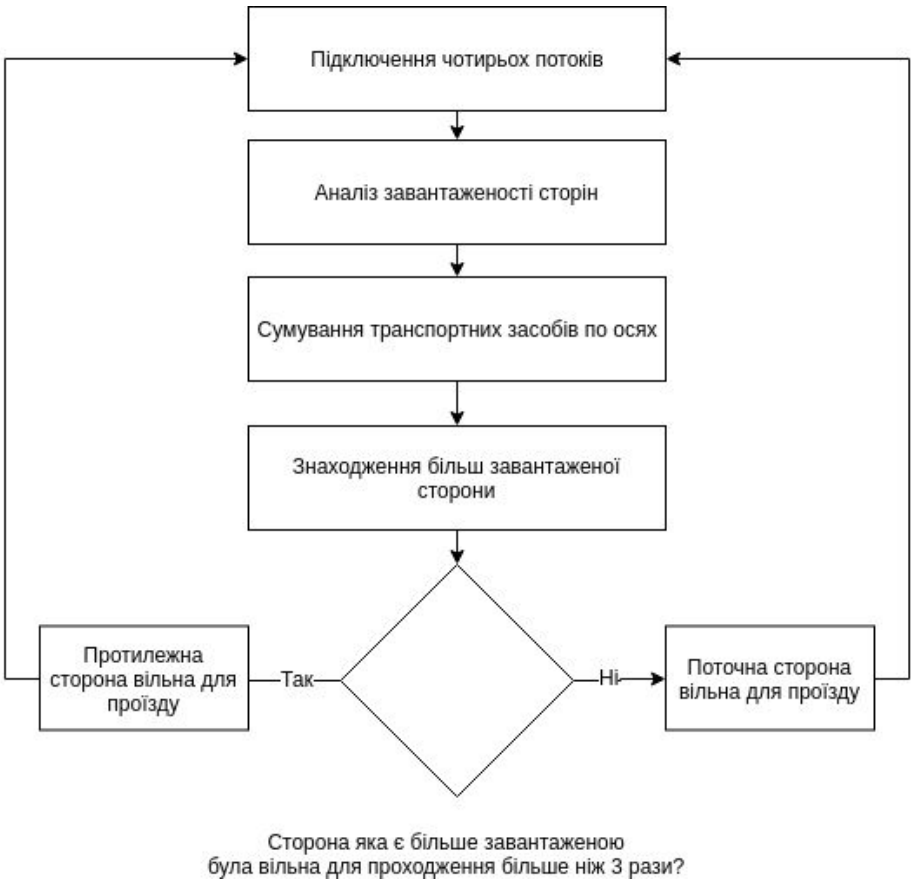


Рисунок 2.4 – Спрощена структура алгоритму системи регулювання дорожнього трафіку

Також існує алгоритм для визначення часу потрібного на проходження ділянки з світлофором він описується наступним чином: системою детекції YOLO визначається число автомобілів, які перебувають на світлофорній ділянці, в класі контролера є константа, яка відповідає за проходження автомобілів, коефіцієнт проходження множиться на кількість авто, в кінцевому результаті отримуємо час на проходження. Варто зауважити, що існує інша константа, яка вказує на максимальний час проходження, вона запобігає протяжну в часі відкритість завантаженої осі. Схема алгоритму наведена нижче

Рисунок 2.5 – Алгоритм обрахунку часу на проходження

Система також зберігає стан світлофора в конкретну ділянку часу, записуючи дані в файлову систему.

Для зберігання файлів в файловій системі не використовується жодна з обгорток, оскільки операції з файлами є простими - створення, оновлення, видалення, читання. До застосування був обраний стандартний пакет - “os”.

Можливість інтеграції з іншими системами

Незначним, але потужним додатком до системи є можливість підключення модулів, які можуть слухати події, які продюсує контроллер.

На даний момент часу таким подій не так багато, але вони можуть їхній список може збільшуватись по мірі росту самої програми, кожен слухач отримує всю інформацію про стан контроллера, а остання подія й додаткову інформацію, а саме:

- ось, яка планується бути активною для проїзду;
- обрахований час для проїзду;
- кількість транспортних засобів;
- фактичний час на проходження з урахуванням ліміту на максимальний час;
- події які підтримуються:
- світлофор переведений в стандартный режим роботи;
- трафік по всіх сторонах є одинаковим;
- світлофори змінили свій стан;

На вищевказані події можна добавити свій власний “слухач”, який буде обробляти їх. На даному етапі існує простий плагін логування даних (тобто виводу інформації на в термінал).

За основу був взятий поведінковий патерн “Спостерігач”.

Спостерігач — це поведінковий патерн проектування, який створює механізм підписки, що дає змогу одним об’єктам стежити й реагувати на події, які відбуваються в інших об’єктах.

При реалізації шаблону «спостерігач» зазвичай використовуються такі класи:

Subject — інтерфейс, що визначає методи для додавання, видалення та оповіщення спостерігачів.

Observer — інтерфейс, за допомогою якого спостережуваний об'єкт сповіщає спостерігачів.

ConcreteSubject — конкретний клас, який реалізує інтерфейс Subject.

ConcreteObserver — конкретний клас, який реалізує інтерфейс Observer.

При зміні спостережуваного об'єкту, оповіщення спостерігачів може бути реалізоване за такими сценаріями:

Спостережуваний об'єкт надсилає, кожному із зареєстрованих спостерігачів, всю потенційно релевантну інформацію (примусове розповсюдження).

Спостережуваний об'єкт надсилає, кожному із зареєстрованих спостерігачів, лише повідомлення про те що інформація була змінена, а кожен із спостерігачів, за необхідності, самостійно здійснює запит необхідної інформації у спостережуваного об'єкта (розповсюдження за запитом).

Висновок

Реалізовано інформаційну систему розумного світлофора для регулювання дорожнього трафіку з використанням мультиканального способу формування даних про обстановку на перехресті. Програмний застосунок забезпечує регулювання дорожнього трафіку за алгоритмом, який дозволяє розвантажувати напрямки руху на перехресті, Розроблено відповідну структуру додатку, що дозволяє зберігати всю потрібну інформацію та автоматизувати необхідні процеси.

Література

1. Smart-light-light-2. Режим доступу: https://github.com/Twitwi96/Smart-traffic-light-2.
2. Схеми основних термінів режиму регулювання. Режим доступу: http://ea.donntu.org:8080/jspui/bitstream/123456789/28329/4/%D0%A2%D0%A1%D0%9E%D0%94%D0%94_%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_3.pdf.
3. Аналог системи розумного світлофора. Режим доступу: https://github.com/Twitwi96/Smart-traffic-light-2.
4. Другий аналог системи розумного світлофора. Режим доступу: https://github.com/JayLohokare/smart-traffic-signals.
5. Грицунов О. В. Інформаційні системи та технології: навч. посіб. для студентів за напрямом підготовки «Транспортні технології» / О. В. Грицунов; Харк. нац. акад. міськ. госп-ва. – Х.: ХНАМГ
6. Павлиш В. А. Основи інформаційних технологій і систем: Навчальний посібник. / Павлиш В. А., Гліненко Л. К. - Львів: Видавництво Львівської політехніки, 2013. − 500 с
7. Обробка геофізичних сигналів у сучасних автоматизованих комплексах: Навчальний посібник / В.А. Кирилюк, М.Ф. Пічугін, О.А. Машков та ін. – Житомир: ЖВІРЕ, 2007. – 176 с
8. Fast heuristics for large geometric traveling salesman problems. / Reinelt, Gerhard (1992) // ORSA Journal on computing, 4:206-217
9. Годун В.М. Інформаційні системи і технології в статистиці: навч. посіб. / В.М. Годун, Н.С. Орленко, М. А. Сендзюк; за ред. В.Ф. Ситника. – К.: КНЕУ, 2003. – 267 с.
10. Hemming R. V. Theory of coding and theory of information / R. V. Hemming // Trans. eng. – M.:Radio & communication, 1983. – 176 p.
11. Patrick E. Fundamentals of the theory of pattern recognition. Moscow: Sov.radio, 1980 .
12. Lainiotis D. G. A Class of Upper Bounds on Probability of Error for Multi –Hypotheses Pattern Recognition // IEEE Transaction on Information Theory, IT-15, №5, 1969

References

1. Smart-light-light-2. Access mode: https://github.com/Twitwi96/Smart-traffic-light-2.
2. Schemes of the main terms of the regulation mode. Access mode: http://ea.donntu.org:8080/jspui/bitstream/123456789/28329/4/%D0%A2%D0%A1%D0%9E%D0%94%D0%94_%D0%9B% D0% B5% D0% BA% D1% 86% D0% B8% D1% 8F_3.pdf.
3. An analogue of a smart traffic light system. Access mode: https://github.com/Twitwi96/Smart-traffic-light-2.

4. The second analogue of the smart traffic light system. Access mode: https://github.com/JayLohokare/smart-traffic-signals.
5. Gritsunov OV Information systems and technologies: textbook. way. for students in the field of training "Transport Technologies" / OV Gritsunov; Hark. nat. acad. city households. - H .: KNAMG
6. Pavlysh VA Fundamentals of information technology and systems: Textbook. / Pavlysh VA, Glinenko LK - Lviv: Lviv Polytechnic Publishing House, 2013. - 500 p.
7. Processing of geophysical signals in modern automated complexes: Textbook / V.A. Кирилюк, M.Ф. Pichugin, OA Mashkov and others. - Zhytomyr.
8. Fast heuristics for large geometric traveling salesman problems. / Reinelt, Gerhard (1992) // ORSA Journal on computing, 4:206-217
9. Godun VM Information systems and technologies in statistics: textbook. way. / B.M. Godun, N.S. Orlenko, MA Sendzyuk; for order. V.F. Sitnika. - K .: KHEU, 2003. - 267 c.
10. Hemming R. V. Theory of coding and theory of information / R. V. Hemming // Trans. eng. – M.: Radio & communication, 1983. – 176 p.
11. Patrick E. Fundamentals of the theory of pattern recognition. Moscow: Sov.radio, 1980 .
12. Lainiotis D. G. A Class of Upper Bounds on Probability of Error for Multi –Hypotheses Pattern Recognition // IEEE Transaction on Information Theory, IT-15, №5, 1969

Додаток В

Лістинги програмного коду

```
import logging

log_format = '%(asctime)s:%(levelname)s:%(message)s'
formatter = logging.Formatter(log_format)

logging.basicConfig(filename='app.log', level=logging.INFO,
format=log_format)

logger = logging.getLogger()

info_stream_handler = logging.StreamHandler()
info_stream_handler.setLevel(logging.INFO)
info_stream_handler.setFormatter(formatter)

error_stream_handler = logging.StreamHandler()
error_stream_handler.setLevel(logging.ERROR)
error_stream_handler.setFormatter(formatter)

warn_stream_handler = logging.StreamHandler()
warn_stream_handler.setLevel(logging.WARN)
warn_stream_handler.setFormatter(formatter)

logger.addHandler(info_stream_handler)
logger.addHandler(error_stream_handler)
logger.addHandler(warn_stream_handler)
import argparse

ap = argparse.ArgumentParser()
ap.add_argument('-c',
                '--config',
                required=True,
                help='path to yolo config file')
ap.add_argument('-w',
                '--weights',
                required=True,
                help='path to yolo pre-trained weights')
ap.add_argument('-cl',
                '--classes',
                required=True,
                help='path to text file containing class
names')
ap.add_argument('-v', '--video', required=True, help='path
to video file')
ap.add_argument('-o',
                '--outputDir',
                default='output',
                help='Path to output directory')
ap.add_argument('-r', '--record', default=False,
help='Determines recording')
ap.add_argument('-i', '--id', default=0, help='ID of
thread')

args = ap.parse_args()
import unittest
from controller import Controller
from traffic_lights import TrafficLights


class SetDataTestCase(unittest.TestCase):
    def test_partial_set_of_params(self):
        inst = Controller(TrafficLights())
        inst.set_data(0, 1)
        result = inst.set_data(0, 2)
        self.assertFalse(result)

    def test_setting_data(self):
        inst = Controller(TrafficLights())
        inst.set_data(0, 1)
        inst.set_data(1, 2)
        inst.set_data(2, 3)
        result = inst.set_data(3, 4)
        self.assertEqual(inst.load, [1, 2, 3, 4])
        self.assertTrue(result)
        self.assertEqual(inst.directions_load, {"x": 6,
"y": 4})
        self.assertEqual(inst.current_mode,
inst.SMART_MODE)

    def test_mode_moving_to_standard_mode(self):
        inst = Controller(TrafficLights())
        inst.move_to_standard_mode()
        self.assertEqual(inst.current_mode,
inst.STANDARD_MODE)

    def test_handling_traffic_with_the_same_load(self):
        inst = Controller(TrafficLights())
        inst.directions_load = {"x": 5, "y": 5}
        inst._handle_traffic()
        self.assertEqual(inst.current_mode,
inst.STANDARD_MODE)
        self.assertEqual(inst.changing_history, [
            {'direction': None, 'mode': 'standard'}
        ])

    def test_handling_traffic_with_different_load(self):
        traffic_lights = TrafficLights()
        inst = Controller(traffic_lights)
        inst.directions_load = {"x": 5, "y": 6}
        inst._handle_traffic()
        self.assertEqual(inst.current_mode,
inst.SMART_MODE)
        self.assertEqual(inst.changing_history, [
            {'direction': 'y', 'mode': 'auto'}
        ])
        self.assertEqual(traffic_lights.state, [True,
False, True, False])
from app_logger import logger
import operator
import numpy
import threading

class Controller:
    """Bussines logic layer.

    Determines the state of directions.
    Attributes:
        current_mode (string): Current state of behavior.
Should be either 'smart' or 'standard'.
        changing_history (list): List of changing history.
        _traffic_lights (object): TrafficLights instance.
API of traffic lights.
        MAX_PASS (int): Maximal time (seconds) for passing
the strech of road.
        MAX_ITERATIONS (int): Maximal iterations value
which indicates how many times one direction may be
selected.
        ITEM_PASS (int): Time for passing the strech of
road for one transport.
        TRAFFIC_LIGHTS_NUMBER (int): Number of traffic
lights.
        SMART_MODE (string): Smart mode value.
        STANDARD_MODE (str): Standard mode value.
        possible_modes (list): List of possible modes.
    """

    MAX_PASS = 50
    MAX_ITERATIONS = 3
    ITEM_PASS = 5
    TRAFFIC_LIGHTS_NUMBER = 4
    SMART_MODE = 'smart'
    STANDARD_MODE = 'standard'
    possible_modes = [SMART_MODE, STANDARD_MODE]

    def __init__(self, traffic_lights):
        """The constructor for Controller class.

        Parameters:
            traffic_lights (object): TrafficLights
instance. API of traffic lights.
        """
        self.load = [0 for _ in
range(self.TRAFFIC_LIGHTS_NUMBER)]
        self.current_mode = self.possible_modes[0]
        self.changing_history = []
        self._traffic_lights = traffic_lights
        self.__processing = False
        self.__timer = None
        self.__params_accumulator = set()
        self.__compute_directions_load()

    def set_data(self, traffic_light_index: int, value:
bool) -> bool:
        """Set data for processing.

        Parameters:
            traffic_light_index (int): Index of traffic
light.
            value (int): Value of appropriate traffic light
        Returns:
            boolean: False if system in processing state or
params are being accumulated
        """
        self.load[traffic_light_index] = value
        self.__compute_directions_load()

        if self.__processing or not
self.__check_params_integrity(
                traffic_light_index):
            return False

        self.__update_current_mode(self.SMART_MODE)
        self._update_processing_state(True)
        self._handle_traffic()
        return True

    def move_to_standard_mode(self):
        """Move system to standard behavior"""
        logger.info('System has been moved to standard
mode')
        self.__update_current_mode(self.STANDARD_MODE)

    def _handle_traffic(self):
        """Handle the traffic.

        Handler makes a decision which direction should be
allowed for passing.
        """
        if self.directions_load['x'] ==
self.directions_load['y']:
            self.move_to_standard_mode()
            self.changing_history.append({
                'direction': None,
                'mode': 'standard',
            })
            self.__timer = threading.Timer(self.MAX_PASS,
self._update_processing_state,
                                           args=[False])
            logger.info('Traffic load is the same; System
was moved to standard behavior for {}
seconds'.format(self.MAX_PASS))
        else:
            direction = max(self.directions_load.items(),
                            key=operator.itemgetter(1))[0]
            direction = direction if
self.__check_direction_iteration_limit(
                    direction) else
self.__get_opposite_direction(direction)
            transport_number =
self.directions_load[direction]
            computed_time = transport_number *
self.ITEM_PASS
            actual_time = self.MAX_PASS if computed_time >
self.MAX_PASS else computed_time
            self.changing_history.append({
                'direction': direction,
                'mode': 'auto',
            })
            self._traffic_lights.turn_on_lights(direction)
            self.__timer = threading.Timer(actual_time,
self._update_processing_state,
                                           args=[False])
            logger.info('''
                Green direction: {0};
                Needed time: {1};
                Number of vehicles: {2};
                Actual time: {3}
            '''.format(direction, computed_time,
transport_number, actual_time))
        logger.info('Directions state:
{}'.format(self.directions_load))
        self.__timer.start()
        self._traffic_lights.visualize_state()

    def _update_processing_state(self, value: bool):
        """Update processing state.

        Parameters:
            value (boolean): Identifies whether the
processing is in progress or no.
        """
        self.__processing = value

    def __check_direction_iteration_limit(self, direction:
str) -> bool:
        """Check limit of iterations for specific
direction.

        Parameters:
            direction (string): Needed direction. Should be
either 'x' or 'y'.
        Returns:
            boolean: True if direction is allowed to be
green one more time.
        """
        return len(self.changing_history) <
self.MAX_ITERATIONS or not all([
            h['direction'] == direction
            for h in
self.changing_history[:self.MAX_ITERATIONS]
        ])

    def __get_opposite_direction(self, direction: str) ->
str:
        """Get oposite direction.

        Parameters:
            direction (string): Direction. Should be either
'x' or 'y'.
        Returns:
            string: Opposite direction to passed direction.
        """
        return 'x' if direction == 'y' else 'y'

    def __update_current_mode(self, value: str):
        """Update current mode of the state

        Parameters:
            value (string): New mode.
        """
        self.current_mode = value

    def __check_params_integrity(self, index: int) -> bool:
        """Check if all needed params have been provided.

        Parameters:
            index (int): Index of traffic light.
        Returns:
            boolean: True if params are ready for
processing.
        """
        self.__params_accumulator.add(index)

        if len(self.__params_accumulator) ==
self.TRAFFIC_LIGHTS_NUMBER:
            self.__params_accumulator = set()
            return True
        return False

    def __compute_directions_load(self):
        """Compute directions.

        Allows to get needed direction by using a
dictionary
        """
        self.directions_load = {
            'x': numpy.sum(self.load[1::2]),
            'y': numpy.sum(self.load[::2]),
        }
#!/bin/bash

# Runs only detector in tiny mode
if [ "$1" = "detector_tiny_mode_video" ]
then
    python3.5 detector.py --video videos/video.mp4 --config
models/yolov3-tiny.cfg --weights models/yolov3-tiny.weights
--classes models/yolov3.txt
```

```
# Runs only detector in tiny mode one frame video
elif [ "$1" = "detector_tiny_mode_one_frame_video" ]
then
    python3.5 detector.py --video
videos/hard_detected_cars.mp4 --config
models/yolov3-tiny.cfg --weights models/yolov3-tiny.weights
--classes models/yolov3.txt --record True
# Runs only detector in tiny mode with recording
elif [ "$1" = "detector_tiny_mode_video_record" ]
then
    python3.5 detector.py --video videos/video.mp4 --config
models/yolov3-tiny.cfg --weights models/yolov3-tiny.weights
--classes models/yolov3.txt --record True
# Runs only detector in full mode
elif [ "$1" = "detector_full_mode_video" ]
then
    python3.5 detector.py --video videos/video.mp4 --config
models/yolov3.cfg --weights models/yolov3.weights --classes
models/yolov3.txt
# Runs only detector in full mode one frame video
elif [ "$1" = "detector_full_mode_one_frame_video" ]
then
    python3.5 detector.py --video
videos/hard_detected_cars.mp4 --config models/yolov3.cfg
--weights models/yolov3.weights --classes models/yolov3.txt
--record True
# Runs full program with one frame videos
elif [ "$1" = "full_program_one_frame_videos" ]
then
    # ID#0 - camera on front. ~ 5 cars
    # ID#1 - hidden cars. ~ 7 cars
    # ID#2 - bunch of cars
    # ID#3 - 2 cars
    python3.5 main.py videos/four_cars_video.mp4
videos/hard_detected_cars.mp4 videos/many_cars.mp4
videos/two_cars.mp4
fi
import sys
import os
import cv2
import numpy as np
import cli_args


class Detector:
    """Detect objects from video source

    Emits data by using stdout stream.
    Attributes:
        args (dictionary): Configuration dictionary.
        classes (list): List of object classes.
        cap (object): VideoCapture instance.
        net (object): DNN instance.
        output_file (string): Path to output file.
        vid_writter (object): VideoWriter instance.
        TRANSPORT_IDS (list): List of ints. Ids of
transports.
        CONF_THRESHOLD (float): Confidence threshold value.
        NMS_THRESHOLD (float): Non-maximal suppression
value.
        INP_WIDTH (int): Width for dnn input.
        INP_HEIGHT (int): Height for dnn input.
        COLOR (int): Color of text.
        SCALE (float): Scale for dnn input.
        VIDEO_FORMAT (string): Format of video.
    """
    TRANSPORT_IDS = [1, 2, 3, 5, 7]
    CONF_THRESHOLD = 0.5
    NMS_THRESHOLD = 0.3
    INP_WIDTH, INP_HEIGHT = (320, 320)
    COLOR = (0, 0, 255)
    SCALE = 0.00392
    VIDEO_FORMAT = 'mp4'

    def __init__(self, args):
        """The constructor for Detector class.

        Parameters:
            args (dictionary): Configuration dictionary.
            args.video (string): Path to video.
            args.weights (string): Path to DNN weights.
            args.config (string): Path to DNN config.
            args.classes (string): Path to object classes.
            args.id (int): ID of thread.
            args.record (boolean): Determines recording.
            args.outputDir (string): Path to output dir.
        """
        self.args = args
        self.classes = None
        self.__set_classes()
        self.cap = cv2.VideoCapture(args.video)
        self.net = cv2.dnn.readNet(args.weights,
args.config)
        self.output_file = os.path.join(
            os.getcwd(), args.outputDir,
            args.video[:-4].split('/').pop() + '_output.' +
self.VIDEO_FORMAT)
        self.vid_writer = cv2.VideoWriter(
            self.output_file,
cv2.VideoWriter_fourcc(*self.VIDEO_FORMAT +
                                                'v'),
20.0,
            (round(self.cap.get(cv2.CAP_PROP_FRAME_WIDTH)),
             round(self.cap.get(
                 cv2.CAP_PROP_FRAME_HEIGHT)))) if
args.record else None

    def draw_bounding_box(self, frame: object, class_id:
int,
                          confidence: float, x: int, y:
int, x_plus_w: int,
                          y_plus_h: int):
        """Draw bounding box and appropriate label.

        Parameters:
            frame (object): Current frame.
            class_id (int): Class is of detected object.
            confidence (float): Detection confidence for
current object.
            x (int): X coordinate of bounding box.
            y (int): Y coordinate of bounding box.
            x_plus_w (int): X + detecion width coordinate
of bounding box.
            y_plus_h (int): Y + dection height coordinate
of bounding box.
        """
        label = str(self.classes[class_id]) + '
{:0.2f}'.format(confidence)

        cv2.rectangle(frame, (x, y), (x_plus_w, y_plus_h),
self.COLOR, 1)
        cv2.putText(frame, label, (x - 10, y - 10),
cv2.FONT_HERSHEY_SIMPLEX,
                    0.5, self.COLOR, 1)

    def __set_classes(self):
        """Set classes by using `args.classes` file"""
        with open(self.args.classes, 'r') as f:
            self.classes = [line.strip() for line in
f.readlines()]

    def get_output_layers(self) -> list:
        """Find output layers and return them"""
        layer_names = self.net.getLayerNames()
        output_layers = [
            layer_names[i[0] - 1] for i in
self.net.getUnconnectedOutLayers()
        ]

        return output_layers

    def postprocess(self, frame: object, outs: list) ->
list:
        """Find needed objects which is introduced in
`self.TRANSPORT_IDS`

        Parameters:
            frame (object): Current frame.
            outs (list): Result of `net.forward` method.

        Returns:
            list: List of class ids.
        """
        frame_height, frame_width = frame.shape[:2]

        class_ids = []
        confidences = []
        boxes = []
        for out in outs:
            for detection in out:
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence = scores[class_id]
                if class_id in self.TRANSPORT_IDS and
confidence > self.CONF_THRESHOLD:
                    center_x = int(detection[0] *
frame_width)
                    center_y = int(detection[1] *
frame_height)
                    width = int(detection[2] * frame_width)
                    height = int(detection[3] *
frame_height)
                    left = int(center_x - width / 2)
                    top = int(center_y - height / 2)
                    class_ids.append(class_id)
                    confidences.append(float(confidence))
                    boxes.append([left, top, width,
height])

        indices = cv2.dnn.NMSBoxes(boxes, confidences,
self.CONF_THRESHOLD,
                                   self.NMS_THRESHOLD)
        for i in indices:
            i = i[0]
            box = boxes[i]
            left = box[0]
            top = box[1]
            width = box[2]
            height = box[3]
            self.draw_bounding_box(frame, class_ids[i],
confidences[i], left,
                                   top, left + width, top +
height)

        return [class_ids[i[0]] for i in indices]

    def run_processing(self):
        """Start objects detection"""
        while cv2.waitKey(1) < 0:
            has_frame, frame = self.cap.read()

            if not has_frame:
                print('Output file is stored as ',
self.output_file)
                break

            blob = cv2.dnn.blobFromImage(frame,
                                         self.SCALE,
                                         (self.INP_WIDTH,
self.INP_HEIGHT),
                                         (0, 0, 0),
                                         True,
                                         crop=False)
            self.net.setInput(blob)

            outs =
self.net.forward(self.get_output_layers())
            detected_objects = self.postprocess(frame,
outs)

            print(len(detected_objects))
            sys.stdout.flush()
            cv2.imshow('stream #{}'.format(self.args.id),
frame)
            if self.vid_writer:
self.vid_writer.write(frame.astype(np.uint8))


if __name__ == '__main__':
    Detector(cli_args.args).run_processing()
import sys
from subprocess import Popen, PIPE, STDOUT
from threading import Thread
from app_logger import logger
from controller import Controller
from traffic_lights import TrafficLights

BASIC_COMMAND = 'sh ./start.sh'
videos = sys.argv[1:]
cmds_list = ['{0} -v {1}'.format(BASIC_COMMAND, p) for p in
iter(videos)]
app = Controller(TrafficLights())

def execute(cmd: str, index: int, app: object):
    """Execute sh script which opens detector"""
    proc = Popen('{0} -i {1}'.format(cmd, index),
shell=True, stdout=PIPE, stderr=STDOUT)

    for line in iter(proc.stdout.readline, b''):
        try:
            app.set_data(index, int(line.rstrip()))
        except ValueError:
            logger.info('Camera ID#{} - {}'.format(index,
line.decode('utf-8')))
        except Exception:
            logger.error('Camera ID#{} has been destroyed.
System will be switched to standard mode'.format(index))
            app.move_to_standard_mode()

for idx, cmd in enumerate(cmds_list):
    p = Thread(target=execute, args=[cmd, idx, app])
    p.start()
#!/bin/bash

TINYMODE=0
ID=0
while getopts "ti:v:" OPTION
do
    case $OPTION in
        t)
            TINYMODE=1
            ;;
        i)
            ID=$OPTARG
            ;;
        v)
            VIDEO=$OPTARG
            ;;
    esac
done

if [ "$TINYMODE" -eq 1 ]
then
    python3.5 detector.py --id $ID --video $VIDEO --config
models/yolov3-tiny.cfg --weights models/yolov3-tiny.weights
--classes models/yolov3.txt
else
    python3.5 detector.py --id $ID --video $VIDEO --config
models/yolov3.cfg --weights models/yolov3.weights --classes
models/yolov3.txt
fi
from app_logger import logger

class TrafficLights:
    """Traffic lights' API class."""

    def __init__(self):
        """The constructor for TrafficLights class."""
        self.state = [False, True] * 2

    def turn_on_lights(self, dir: str):
        """Turn on green lights on direction

        Parameters:
            dir (string): Needed direction. Should be
either 'x' or 'y'.
        """
        if dir == 'x':
            self.state = [False, True] * 2
        elif dir == 'y':
            self.state = [True, False] * 2
        else:
            raise ValueError('direction should either "x"
or "y"')

    def switch_lights(self):
        """Switch green according to directions"""
        self.state = [not v for v in self.state]

    def visualize_state(self):
        """Present current state of the system"""
        logger.info("""\n Visualization of crossroad
state\n
                    (0)
                     |
                    {0}
                     |
         (3)----{3}---C---{1}----(1)
                     |
                    {2}
                     |
                    (2)
        \n""".format(*map(lambda i: 'o' if i else 'x',
self.state)))
```

```
# This Python file uses the following encoding: utf-8
import sys
from PySide2.QtWidgets import QApplication, QWidget, QPushButton, QVBoxLayout
from controller import Controller

class TrafficLightsResolver(QWidget):
    def __init__(self, controller: Controller):
        QWidget.__init__(self)
        self.controller = controller
        self.load_ui()

    @staticmethod
    def exec(controller: Controller):
        app = QApplication([])
        window = TrafficLightsResolver(controller)
        window.show()
        sys.exit(app.exec_())

    def x_axis_click_handler(self):
        self.controller.force_direction_switch('x')

    def y_axis_click_handler(self):
        self.controller.force_direction_switch('y')

    def auto_resolve_click_handler(self):
        self.controller.move_to_smart_mode()

    def load_ui(self):
        layout = QVBoxLayout(self)
        x_axis_direction = QPushButton("Activate X Axis")
        x_axis_direction.clicked.connect(self.x_axis_click_handler)
        layout.addWidget(x_axis_direction)
        y_axis_direction = QPushButton("Activate Y Axis")
        y_axis_direction.clicked.connect(self.y_axis_click_handler)
        layout.addWidget(y_axis_direction)
        auto_resolve = QPushButton("Activate Auto Resolve")
        auto_resolve.clicked.connect(self.auto_resolve_click_handler)
        layout.addWidget(auto_resolve)
```

```
{
    "files": ["trafficlightsresolver.py"]
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE QtCreatorProject>
<!-- Written by QtCreator 4.13.2, 2020-10-24T20:45:07. -->
<qtcreator>
 <data>
  <variable>EnvironmentId</variable>
  <value type="QByteArray">{52d5526e-c2df-4e4c-86d3-ecfadea7b0a0}</value>
 </data>
 <data>
  <variable>ProjectExplorer.Project.ActiveTarget</variable>
  <value type="int">0</value>
 </data>
 <data>
  <variable>ProjectExplorer.Project.EditorSettings</variable>
  <valuemap type="QVariantMap">
   <value type="bool" key="EditorConfiguration.AutoIndent">true</value>
   <value type="bool" key="EditorConfiguration.AutoSpacesForTabs">false</value>
   <value type="bool" key="EditorConfiguration.CamelCaseNavigation">true</value>
   <valuemap type="QVariantMap" key="EditorConfiguration.CodeStyle.0">
    <value type="QString" key="language">Cpp</value>
    <valuemap type="QVariantMap" key="value">
     <value type="QByteArray" key="CurrentPreferences">CppGlobal</value>
    </valuemap>
   </valuemap>
   <valuemap type="QVariantMap" key="EditorConfiguration.CodeStyle.1">
    <value type="QString" key="language">QmlJS</value>
    <valuemap type="QVariantMap" key="value">
     <value type="QByteArray" key="CurrentPreferences">QmlJSGlobal</value>
    </valuemap>
   </valuemap>
   <value type="int" key="EditorConfiguration.CodeStyle.Count">2</value>
   <value type="QByteArray" key="EditorConfiguration.Codec">UTF-8</value>
   <value type="bool" key="EditorConfiguration.ConstrainTooltips">false</value>
   <value type="int" key="EditorConfiguration.IndentSize">4</value>
   <value type="bool" key="EditorConfiguration.KeyboardTooltips">false</value>
   <value type="int" key="EditorConfiguration.MarginColumn">80</value>
   <value type="bool" key="EditorConfiguration.MouseHiding">true</value>
   <value type="bool" key="EditorConfiguration.MouseNavigation">true</value>
   <value type="int" key="EditorConfiguration.PaddingMode">1</value>
   <value type="bool" key="EditorConfiguration.ScrollWheelZooming">true</value>
   <value type="bool" key="EditorConfiguration.ShowMargin">false</value>
   <value type="int" key="EditorConfiguration.SmartBackspaceBehavior">0</value>
   <value type="bool"
key="EditorConfiguration.SmartSelectionChanging">true</value>
   <value type="bool" key="EditorConfiguration.SpacesForTabs">true</value>
   <value type="int" key="EditorConfiguration.TabKeyBehavior">0</value>
   <value type="int" key="EditorConfiguration.TabSize">8</value>
   <value type="bool" key="EditorConfiguration.UseGlobal">true</value>
   <value type="int" key="EditorConfiguration.Utf8BomBehavior">1</value>
   <value type="bool" key="EditorConfiguration.addFinalNewLine">true</value>
   <value type="bool" key="EditorConfiguration.cleanIndentation">true</value>
   <value type="bool" key="EditorConfiguration.cleanWhitespace">true</value>
   <value type="QString" key="EditorConfiguration.ignoreFileTypes">*.md, *.MD,
Makefile</value>
   <value type="bool" key="EditorConfiguration.inEntireDocument">false</value>
   <value type="bool"
key="EditorConfiguration.skipTrailingWhitespace">true</value>
  </valuemap>
 </data>
 <data>
  <variable>ProjectExplorer.Project.PluginSettings</variable>
  <valuemap type="QVariantMap">
   <valuemap type="QVariantMap" key="AutoTest.ActiveFrameworks">
    <value type="bool" key="AutoTest.Framework.Boost">true</value>
    <value type="bool" key="AutoTest.Framework.Catch">true</value>
    <value type="bool" key="AutoTest.Framework.GTest">true</value>
    <value type="bool" key="AutoTest.Framework.QtQuickTest">true</value>
    <value type="bool" key="AutoTest.Framework.QtTest">true</value>
   </valuemap>
   <valuemap type="QVariantMap" key="AutoTest.CheckStates"/>
   <value type="int" key="AutoTest.RunAfterBuild">0</value>
   <value type="bool" key="AutoTest.UseGlobal">true</value>
   <valuelist type="QVariantList" key="ClangCodeModel.CustomCommandLineKey"/>
   <value type="bool" key="ClangCodeModel.UseGlobalConfig">true</value>
   <value type="QString"
key="ClangCodeModel.WarningConfigId">Builtin.Questionable</value>
   <valuemap type="QVariantMap" key="ClangTools">
    <value type="bool" key="ClangTools.BuildBeforeAnalysis">true</value>
    <value type="QString"
key="ClangTools.DiagnosticConfig">Builtin.DefaultTidyAndClazy</value>
    <value type="int" key="ClangTools.ParallelJobs">4</value>
    <valuelist type="QVariantList" key="ClangTools.SelectedDirs"/>
    <valuelist type="QVariantList" key="ClangTools.SelectedFiles"/>
    <valuelist type="QVariantList" key="ClangTools.SuppressedDiagnostics"/>
    <value type="bool" key="ClangTools.UseGlobalSettings">true</value>
   </valuemap>
  </valuemap>
 </data>
 <data>
  <variable>ProjectExplorer.Project.Target.0</variable>
  <valuemap type="QVariantMap">
   <value type="QString" key="DeviceType">Desktop</value>
   <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Desktop
(x86-darwin-generic-mach_o-32bit)</value>
   <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DisplayName">Desktop
(x86-darwin-generic-mach_o-32bit)</value>
   <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">{b82cd0f6-29c0-4869-a021-ece7ebcf37a
7}</value>
   <value type="int"
key="ProjectExplorer.Target.ActiveBuildConfiguration">-1</value>
   <value type="int"
key="ProjectExplorer.Target.ActiveDeployConfiguration">0</value>
   <value type="int" key="ProjectExplorer.Target.ActiveRunConfiguration">0</value>
   <value type="int"
key="ProjectExplorer.Target.BuildConfigurationCount">0</value>
   <valuemap type="QVariantMap"
key="ProjectExplorer.Target.DeployConfiguration.0">
    <valuemap type="QVariantMap"
key="ProjectExplorer.BuildConfiguration.BuildStepList.0">
     <value type="int" key="ProjectExplorer.BuildStepList.StepsCount">0</value>
     <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Deploy</value>
     <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DisplayName">Deploy</value>
     <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Deploy</v
alue>
    </valuemap>
    <value type="int"
key="ProjectExplorer.BuildConfiguration.BuildStepListCount">1</value>
    <valuemap type="QVariantMap"
key="ProjectExplorer.DeployConfiguration.CustomData"/>
    <value type="bool"
key="ProjectExplorer.DeployConfiguration.CustomDataEnabled">false</value>
    <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.DefaultDeployConfigu
ration</value>
   </valuemap>
   <value type="int"
key="ProjectExplorer.Target.DeployConfigurationCount">1</value>
   <valuemap type="QVariantMap" key="ProjectExplorer.Target.PluginSettings"/>
   <valuemap type="QVariantMap" key="ProjectExplorer.Target.RunConfiguration.0">
    <value type="QString" key="Analyzer.Perf.CallgraphMode">dwarf</value>
    <valuelist type="QVariantList" key="Analyzer.Perf.Events">
     <value type="QString">cpu-cycles</value>
    </valuelist>
    <valuelist type="QVariantList" key="Analyzer.Perf.ExtraArguments"/>
    <value type="int" key="Analyzer.Perf.Frequency">250</value>
    <valuelist type="QVariantList" key="Analyzer.Perf.RecordArguments">
     <value type="QString">-e</value>
     <value type="QString">cpu-cycles</value>
     <value type="QString">--call-graph</value>
     <value type="QString">dwarf,4096</value>
     <value type="QString">-F</value>
     <value type="QString">250</value>
    </valuelist>
    <value type="QString" key="Analyzer.Perf.SampleMode">-F</value>
    <value type="bool" key="Analyzer.Perf.Settings.UseGlobalSettings">true</value>
    <value type="int" key="Analyzer.Perf.StackSize">4096</value>
    <value type="bool" key="Analyzer.QmlProfiler.AggregateTraces">false</value>
    <value type="bool" key="Analyzer.QmlProfiler.FlushEnabled">false</value>
    <value type="uint" key="Analyzer.QmlProfiler.FlushInterval">1000</value>
    <value type="QString" key="Analyzer.QmlProfiler.LastTraceFile"></value>
    <value type="bool"
key="Analyzer.QmlProfiler.Settings.UseGlobalSettings">true</value>
    <valuelist type="QVariantList" key="Analyzer.Valgrind.AddedSuppressionFiles"/>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.CollectBusEvents">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.CollectSystime">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.EnableBranchSim">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.EnableCacheSim">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.EnableEventToolTips">true</value>
    <value type="double"
key="Analyzer.Valgrind.Callgrind.MinimumCostRatio">0.01</value>
    <value type="double"
key="Analyzer.Valgrind.Callgrind.VisualisationMinimumCostRatio">10</value>
    <value type="bool" key="Analyzer.Valgrind.FilterExternalIssues">true</value>
    <value type="QString"
key="Analyzer.Valgrind.KCachegrindExecutable">kcachegrind</value>
    <value type="int" key="Analyzer.Valgrind.LeakCheckOnFinish">1</value>
    <value type="int" key="Analyzer.Valgrind.NumCallers">25</value>
    <valuelist type="QVariantList"
key="Analyzer.Valgrind.RemovedSuppressionFiles"/>
    <value type="int" key="Analyzer.Valgrind.SelfModifyingCodeDetection">1</value>
    <value type="bool"
key="Analyzer.Valgrind.Settings.UseGlobalSettings">true</value>
    <value type="bool" key="Analyzer.Valgrind.ShowReachable">false</value>
    <value type="bool" key="Analyzer.Valgrind.TrackOrigins">true</value>
    <value type="QString"
key="Analyzer.Valgrind.ValgrindExecutable">valgrind</value>
    <valuelist type="QVariantList" key="Analyzer.Valgrind.VisibleErrorKinds">
     <value type="int">0</value>
     <value type="int">1</value>
     <value type="int">2</value>
```

```
     <value type="int">3</value>
     <value type="int">4</value>
     <value type="int">5</value>
     <value type="int">6</value>
     <value type="int">7</value>
     <value type="int">8</value>
     <value type="int">9</value>
     <value type="int">10</value>
     <value type="int">11</value>
     <value type="int">12</value>
     <value type="int">13</value>
     <value type="int">14</value>
    </valuelist>
    <valuelist type="QVariantList" key="CustomOutputParsers"/>
    <value type="int" key="PE.EnvironmentAspect.Base">2</value>
    <valuelist type="QVariantList" key="PE.EnvironmentAspect.Changes"/>
    <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DisplayName">trafficlightsresolver</valu
e>
    <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">PythonEditor.RunConfiguration./Users
/yurii.mykhailiak/Projects/intelligent-traffic-lights/trafficlightsresolver.py</va
lue>
    <value type="QString"
key="ProjectExplorer.RunConfiguration.BuildKey">/Users/yurii.mykhailiak/Projects/i
ntelligent-traffic-lights/trafficlightsresolver.py</value>
    <value type="bool" key="PythonEditor.RunConfiguation.Buffered">false</value>
    <value type="QString"
key="PythonEditor.RunConfiguation.Interpreter">{fd053d1a-0bec-4e9e-8e0d-39e55f1ae3
60}</value>
    <value type="QString"
key="PythonEditor.RunConfiguation.Script">/Users/yurii.mykhailiak/Projects/intelli
gent-traffic-lights/trafficlightsresolver.py</value>
    <value type="QString" key="RunConfiguration.Arguments"></value>
    <value type="bool" key="RunConfiguration.Arguments.multi">false</value>
    <value type="QString" key="RunConfiguration.OverrideDebuggerStartup"></value>
    <value type="bool" key="RunConfiguration.UseCppDebugger">false</value>
    <value type="bool" key="RunConfiguration.UseCppDebuggerAuto">true</value>
    <value type="bool" key="RunConfiguration.UseMultiProcess">false</value>
    <value type="bool" key="RunConfiguration.UseQmlDebugger">false</value>
    <value type="bool" key="RunConfiguration.UseQmlDebuggerAuto">true</value>
    <value type="QString" key="RunConfiguration.WorkingDirectory"></value>
    <value type="QString"
key="RunConfiguration.WorkingDirectory.default">/Users/yurii.mykhailiak/Projects/i
ntelligent-traffic-lights</value>
   </valuemap>
   <value type="int" key="ProjectExplorer.Target.RunConfigurationCount">1</value>
  </valuemap>
 </data>
 <data>
  <variable>ProjectExplorer.Project.TargetCount</variable>
  <value type="int">1</value>
 </data>
 <data>
  <variable>ProjectExplorer.Project.Updater.FileVersion</variable>
  <value type="int">22</value>
 </data>
 <data>
  <variable>Version</variable>
  <value type="int">22</value>
 </data>
</qtcreator>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE QtCreatorProject>
<!-- Written by QtCreator 4.13.2, 2020-10-24T20:45:07. -->
<qtcreator>
 <data>
  <variable>EnvironmentId</variable>
  <value type="QByteArray">{52d5526e-c2df-4e4c-86d3-ecfadea7b0a0}</value>
 </data>
 <data>
  <variable>ProjectExplorer.Project.ActiveTarget</variable>
  <value type="int">0</value>
 </data>
 <data>
  <variable>ProjectExplorer.Project.EditorSettings</variable>
  <valuemap type="QVariantMap">
   <value type="bool" key="EditorConfiguration.AutoIndent">true</value>
   <value type="bool" key="EditorConfiguration.AutoSpacesForTabs">false</value>
   <value type="bool" key="EditorConfiguration.CamelCaseNavigation">true</value>
   <valuemap type="QVariantMap" key="EditorConfiguration.CodeStyle.0">
    <value type="QString" key="language">Cpp</value>
    <valuemap type="QVariantMap" key="value">
     <value type="QByteArray" key="CurrentPreferences">CppGlobal</value>
    </valuemap>
   </valuemap>
   <valuemap type="QVariantMap" key="EditorConfiguration.CodeStyle.1">
    <value type="QString" key="language">QmlJS</value>
    <valuemap type="QVariantMap" key="value">
     <value type="QByteArray" key="CurrentPreferences">QmlJSGlobal</value>
    </valuemap>
   </valuemap>
   <value type="int" key="EditorConfiguration.CodeStyle.Count">2</value>
   <value type="QByteArray" key="EditorConfiguration.Codec">UTF-8</value>
   <value type="bool" key="EditorConfiguration.ConstrainTooltips">false</value>
   <value type="int" key="EditorConfiguration.IndentSize">4</value>
   <value type="bool" key="EditorConfiguration.KeyboardTooltips">false</value>
   <value type="int" key="EditorConfiguration.MarginColumn">80</value>
   <value type="bool" key="EditorConfiguration.MouseHiding">true</value>
   <value type="bool" key="EditorConfiguration.MouseNavigation">true</value>
   <value type="int" key="EditorConfiguration.PaddingMode">1</value>
   <value type="bool" key="EditorConfiguration.ScrollWheelZooming">true</value>
   <value type="bool" key="EditorConfiguration.ShowMargin">false</value>
   <value type="int" key="EditorConfiguration.SmartBackspaceBehavior">0</value>
   <value type="bool"
key="EditorConfiguration.SmartSelectionChanging">true</value>
   <value type="bool" key="EditorConfiguration.SpacesForTabs">true</value>
   <value type="int" key="EditorConfiguration.TabKeyBehavior">0</value>
   <value type="int" key="EditorConfiguration.TabSize">8</value>
   <value type="bool" key="EditorConfiguration.UseGlobal">true</value>
   <value type="int" key="EditorConfiguration.Utf8BomBehavior">1</value>
   <value type="bool" key="EditorConfiguration.addFinalNewLine">true</value>
   <value type="bool" key="EditorConfiguration.cleanIndentation">true</value>
   <value type="bool" key="EditorConfiguration.cleanWhitespace">true</value>
   <value type="QString" key="EditorConfiguration.ignoreFileTypes">*.md, *.MD,
Makefile</value>
   <value type="bool" key="EditorConfiguration.inEntireDocument">false</value>
   <value type="bool"
key="EditorConfiguration.skipTrailingWhitespace">true</value>
  </valuemap>
 </data>
 <data>
  <variable>ProjectExplorer.Project.PluginSettings</variable>
  <valuemap type="QVariantMap">
   <valuemap type="QVariantMap" key="AutoTest.ActiveFrameworks">
    <value type="bool" key="AutoTest.Framework.Boost">true</value>
    <value type="bool" key="AutoTest.Framework.Catch">true</value>
    <value type="bool" key="AutoTest.Framework.GTest">true</value>
    <value type="bool" key="AutoTest.Framework.QtQuickTest">true</value>
    <value type="bool" key="AutoTest.Framework.QtTest">true</value>
   </valuemap>
   <valuemap type="QVariantMap" key="AutoTest.CheckStates"/>
   <value type="int" key="AutoTest.RunAfterBuild">0</value>
   <value type="bool" key="AutoTest.UseGlobal">true</value>
   <valuelist type="QVariantList" key="ClangCodeModel.CustomCommandLineKey"/>
   <value type="bool" key="ClangCodeModel.UseGlobalConfig">true</value>
   <value type="QString"
key="ClangCodeModel.WarningConfigId">Builtin.Questionable</value>
   <valuemap type="QVariantMap" key="ClangTools">
    <value type="bool" key="ClangTools.BuildBeforeAnalysis">true</value>
    <value type="QString"
key="ClangTools.DiagnosticConfig">Builtin.DefaultTidyAndClazy</value>
    <value type="int" key="ClangTools.ParallelJobs">4</value>
    <valuelist type="QVariantList" key="ClangTools.SelectedDirs"/>
    <valuelist type="QVariantList" key="ClangTools.SelectedFiles"/>
    <valuelist type="QVariantList" key="ClangTools.SuppressedDiagnostics"/>
    <value type="bool" key="ClangTools.UseGlobalSettings">true</value>
   </valuemap>
  </valuemap>
 </data>
 <data>
  <variable>ProjectExplorer.Project.Target.0</variable>
  <valuemap type="QVariantMap">
   <value type="QString" key="DeviceType">Desktop</value>
   <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Desktop
(x86-darwin-generic-mach_o-32bit)</value>
   <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DisplayName">Desktop
(x86-darwin-generic-mach_o-32bit)</value>
   <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">{b82cd0f6-29c0-4869-a021-ece7ebcf37a
7}</value>
   <value type="int"
key="ProjectExplorer.Target.ActiveBuildConfiguration">-1</value>
   <value type="int"
key="ProjectExplorer.Target.ActiveDeployConfiguration">0</value>
   <value type="int" key="ProjectExplorer.Target.ActiveRunConfiguration">0</value>
   <value type="int"
key="ProjectExplorer.Target.BuildConfigurationCount">0</value>
   <valuemap type="QVariantMap"
key="ProjectExplorer.Target.DeployConfiguration.0">
    <valuemap type="QVariantMap"
key="ProjectExplorer.BuildConfiguration.BuildStepList.0">
     <value type="int" key="ProjectExplorer.BuildStepList.StepsCount">0</value>
     <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Deploy</value>
     <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DisplayName">Deploy</value>
     <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Deploy</v
alue>
    </valuemap>
    <value type="int"
key="ProjectExplorer.BuildConfiguration.BuildStepListCount">1</value>
    <valuemap type="QVariantMap"
key="ProjectExplorer.DeployConfiguration.CustomData"/>
    <value type="bool"
key="ProjectExplorer.DeployConfiguration.CustomDataEnabled">false</value>
    <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.DefaultDeployConfigu
ration</value>
   </valuemap>
   <value type="int"
key="ProjectExplorer.Target.DeployConfigurationCount">1</value>
   <valuemap type="QVariantMap" key="ProjectExplorer.Target.PluginSettings"/>
   <valuemap type="QVariantMap" key="ProjectExplorer.Target.RunConfiguration.0">
    <value type="QString" key="Analyzer.Perf.CallgraphMode">dwarf</value>
    <valuelist type="QVariantList" key="Analyzer.Perf.Events">
     <value type="QString">cpu-cycles</value>
    </valuelist>
    <valuelist type="QVariantList" key="Analyzer.Perf.ExtraArguments"/>
    <value type="int" key="Analyzer.Perf.Frequency">250</value>
    <valuelist type="QVariantList" key="Analyzer.Perf.RecordArguments">
     <value type="QString">-e</value>
     <value type="QString">cpu-cycles</value>
     <value type="QString">--call-graph</value>
     <value type="QString">dwarf,4096</value>
     <value type="QString">-F</value>
     <value type="QString">250</value>
    </valuelist>
    <value type="QString" key="Analyzer.Perf.SampleMode">-F</value>
    <value type="bool" key="Analyzer.Perf.Settings.UseGlobalSettings">true</value>
    <value type="int" key="Analyzer.Perf.StackSize">4096</value>
    <value type="bool" key="Analyzer.QmlProfiler.AggregateTraces">false</value>
    <value type="bool" key="Analyzer.QmlProfiler.FlushEnabled">false</value>
    <value type="uint" key="Analyzer.QmlProfiler.FlushInterval">1000</value>
    <value type="QString" key="Analyzer.QmlProfiler.LastTraceFile"></value>
    <value type="bool"
key="Analyzer.QmlProfiler.Settings.UseGlobalSettings">true</value>
    <valuelist type="QVariantList" key="Analyzer.Valgrind.AddedSuppressionFiles"/>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.CollectBusEvents">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.CollectSystime">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.EnableBranchSim">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.EnableCacheSim">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.EnableEventToolTips">true</value>
    <value type="double"
key="Analyzer.Valgrind.Callgrind.MinimumCostRatio">0.01</value>
```

```
    <value type="double"
key="Analyzer.Valgrind.Callgrind.VisualisationMinimumCostRatio">10</value>
    <value type="bool" key="Analyzer.Valgrind.FilterExternalIssues">true</value>
    <value type="QString"
key="Analyzer.Valgrind.KCachegrindExecutable">kcachegrind</value>
    <value type="int" key="Analyzer.Valgrind.LeakCheckOnFinish">1</value>
    <value type="int" key="Analyzer.Valgrind.NumCallers">25</value>
    <valuelist type="QVariantList"
key="Analyzer.Valgrind.RemovedSuppressionFiles"/>
    <value type="int" key="Analyzer.Valgrind.SelfModifyingCodeDetection">1</value>
    <value type="bool"
key="Analyzer.Valgrind.Settings.UseGlobalSettings">true</value>
    <value type="bool" key="Analyzer.Valgrind.ShowReachable">false</value>
    <value type="bool" key="Analyzer.Valgrind.TrackOrigins">true</value>
    <value type="QString"
key="Analyzer.Valgrind.ValgrindExecutable">valgrind</value>
    <valuelist type="QVariantList" key="Analyzer.Valgrind.VisibleErrorKinds">
     <value type="int">0</value>
     <value type="int">1</value>
     <value type="int">2</value>
     <value type="int">3</value>
     <value type="int">4</value>
     <value type="int">5</value>
     <value type="int">6</value>
     <value type="int">7</value>
     <value type="int">8</value>
     <value type="int">9</value>
     <value type="int">10</value>
     <value type="int">11</value>
     <value type="int">12</value>
     <value type="int">13</value>
     <value type="int">14</value>
    </valuelist>
    <valuelist type="QVariantList" key="CustomOutputParsers"/>
    <value type="int" key="PE.EnvironmentAspect.Base">2</value>
    <valuelist type="QVariantList" key="PE.EnvironmentAspect.Changes"/>
    <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DisplayName">trafficlightsresolver</valu
e>
    <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">PythonEditor.RunConfiguration./Users
/yurii.mykhailiak/Projects/intelligent-traffic-lights/trafficlightsresolver.py</va
lue>
    <value type="QString"
key="ProjectExplorer.RunConfiguration.BuildKey">/Users/yurii.mykhailiak/Projects/i
ntelligent-traffic-lights/trafficlightsresolver.py</value>
    <value type="bool" key="PythonEditor.RunConfiguation.Buffered">false</value>
    <value type="QString"
key="PythonEditor.RunConfiguation.Interpreter">{fd053d1a-0bec-4e9e-8e0d-39e55f1ae3
60}</value>
    <value type="QString"
key="PythonEditor.RunConfiguation.Script">/Users/yurii.mykhailiak/Projects/intelli
gent-traffic-lights/trafficlightsresolver.py</value>
    <value type="QString" key="RunConfiguration.Arguments"></value>
    <value type="bool" key="RunConfiguration.Arguments.multi">false</value>
    <value type="QString" key="RunConfiguration.OverrideDebuggerStartup"></value>
    <value type="bool" key="RunConfiguration.UseCppDebugger">false</value>
    <value type="bool" key="RunConfiguration.UseCppDebuggerAuto">true</value>
    <value type="bool" key="RunConfiguration.UseMultiProcess">false</value>
    <value type="bool" key="RunConfiguration.UseQmlDebugger">false</value>
    <value type="bool" key="RunConfiguration.UseQmlDebuggerAuto">true</value>
    <value type="QString" key="RunConfiguration.WorkingDirectory"></value>
    <value type="QString"
key="RunConfiguration.WorkingDirectory.default">/Users/yurii.mykhailiak/Projects/i
ntelligent-traffic-lights</value>
   </valuemap>
   <value type="int" key="ProjectExplorer.Target.RunConfigurationCount">1</value>
  </valuemap>
 </data>
 <data>
  <variable>ProjectExplorer.Project.TargetCount</variable>
  <value type="int">1</value>
 </data>
 <data>
  <variable>ProjectExplorer.Project.Updater.FileVersion</variable>
  <value type="int">22</value>
 </data>
 <data>
  <variable>Version</variable>
  <value type="int">22</value>
 </data>
</qtcreator>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE QtCreatorProject>
<!-- Written by QtCreator 4.13.2, 2020-10-24T20:45:07. -->
<qtcreator>
 <data>
  <variable>EnvironmentId</variable>
  <value type="QByteArray">{52d5526e-c2df-4e4c-86d3-ecfadea7b0a0}</value>
 </data>
 <data>
  <variable>ProjectExplorer.Project.ActiveTarget</variable>
  <value type="int">0</value>
 </data>
 <data>
  <variable>ProjectExplorer.Project.EditorSettings</variable>
  <valuemap type="QVariantMap">
   <value type="bool" key="EditorConfiguration.AutoIndent">true</value>
   <value type="bool" key="EditorConfiguration.AutoSpacesForTabs">false</value>
   <value type="bool" key="EditorConfiguration.CamelCaseNavigation">true</value>
   <valuemap type="QVariantMap" key="EditorConfiguration.CodeStyle.0">
    <value type="QString" key="language">Cpp</value>
    <valuemap type="QVariantMap" key="value">
     <value type="QByteArray" key="CurrentPreferences">CppGlobal</value>
    </valuemap>
   </valuemap>
   <valuemap type="QVariantMap" key="EditorConfiguration.CodeStyle.1">
    <value type="QString" key="language">QmlJS</value>
    <valuemap type="QVariantMap" key="value">
     <value type="QByteArray" key="CurrentPreferences">QmlJSGlobal</value>
    </valuemap>
   </valuemap>
   <value type="int" key="EditorConfiguration.CodeStyle.Count">2</value>
   <value type="QByteArray" key="EditorConfiguration.Codec">UTF-8</value>
   <value type="bool" key="EditorConfiguration.ConstrainTooltips">false</value>
   <value type="int" key="EditorConfiguration.IndentSize">4</value>
   <value type="bool" key="EditorConfiguration.KeyboardTooltips">false</value>
   <value type="int" key="EditorConfiguration.MarginColumn">80</value>
   <value type="bool" key="EditorConfiguration.MouseHiding">true</value>
   <value type="bool" key="EditorConfiguration.MouseNavigation">true</value>
   <value type="int" key="EditorConfiguration.PaddingMode">1</value>
   <value type="bool" key="EditorConfiguration.ScrollWheelZooming">true</value>
   <value type="bool" key="EditorConfiguration.ShowMargin">false</value>
   <value type="int" key="EditorConfiguration.SmartBackspaceBehavior">0</value>
   <value type="bool"
key="EditorConfiguration.SmartSelectionChanging">true</value>
   <value type="bool" key="EditorConfiguration.SpacesForTabs">true</value>
   <value type="int" key="EditorConfiguration.TabKeyBehavior">0</value>
   <value type="int" key="EditorConfiguration.TabSize">8</value>
   <value type="bool" key="EditorConfiguration.UseGlobal">true</value>
   <value type="int" key="EditorConfiguration.Utf8BomBehavior">1</value>
   <value type="bool" key="EditorConfiguration.addFinalNewLine">true</value>
   <value type="bool" key="EditorConfiguration.cleanIndentation">true</value>
   <value type="bool" key="EditorConfiguration.cleanWhitespace">true</value>
   <value type="QString" key="EditorConfiguration.ignoreFileTypes">*.md, *.MD,
Makefile</value>
   <value type="bool" key="EditorConfiguration.inEntireDocument">false</value>
   <value type="bool"
key="EditorConfiguration.skipTrailingWhitespace">true</value>
  </valuemap>
 </data>
 <data>
  <variable>ProjectExplorer.Project.PluginSettings</variable>
  <valuemap type="QVariantMap">
   <valuemap type="QVariantMap" key="AutoTest.ActiveFrameworks">
    <value type="bool" key="AutoTest.Framework.Boost">true</value>
    <value type="bool" key="AutoTest.Framework.Catch">true</value>
    <value type="bool" key="AutoTest.Framework.GTest">true</value>
    <value type="bool" key="AutoTest.Framework.QtQuickTest">true</value>
    <value type="bool" key="AutoTest.Framework.QtTest">true</value>
   </valuemap>
   <valuemap type="QVariantMap" key="AutoTest.CheckStates"/>
   <value type="int" key="AutoTest.RunAfterBuild">0</value>
   <value type="bool" key="AutoTest.UseGlobal">true</value>
   <valuelist type="QVariantList" key="ClangCodeModel.CustomCommandLineKey"/>
   <value type="bool" key="ClangCodeModel.UseGlobalConfig">true</value>
   <value type="QString"
key="ClangCodeModel.WarningConfigId">Builtin.Questionable</value>
   <valuemap type="QVariantMap" key="ClangTools">
    <value type="bool" key="ClangTools.BuildBeforeAnalysis">true</value>
    <value type="QString"
key="ClangTools.DiagnosticConfig">Builtin.DefaultTidyAndClazy</value>
    <value type="int" key="ClangTools.ParallelJobs">4</value>
    <valuelist type="QVariantList" key="ClangTools.SelectedDirs"/>
    <valuelist type="QVariantList" key="ClangTools.SelectedFiles"/>
    <valuelist type="QVariantList" key="ClangTools.SuppressedDiagnostics"/>
    <value type="bool" key="ClangTools.UseGlobalSettings">true</value>
   </valuemap>
  </valuemap>
 </data>
 <data>
  <variable>ProjectExplorer.Project.Target.0</variable>
  <valuemap type="QVariantMap">
   <value type="QString" key="DeviceType">Desktop</value>
   <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Desktop
(x86-darwin-generic-mach_o-32bit)</value>
   <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DisplayName">Desktop
(x86-darwin-generic-mach_o-32bit)</value>
   <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">{b82cd0f6-29c0-4869-a021-ece7ebcf37a
7}</value>
   <value type="int"
key="ProjectExplorer.Target.ActiveBuildConfiguration">-1</value>
   <value type="int"
key="ProjectExplorer.Target.ActiveDeployConfiguration">0</value>
   <value type="int" key="ProjectExplorer.Target.ActiveRunConfiguration">0</value>
   <value type="int"
key="ProjectExplorer.Target.BuildConfigurationCount">0</value>
   <valuemap type="QVariantMap"
key="ProjectExplorer.Target.DeployConfiguration.0">
    <valuemap type="QVariantMap"
key="ProjectExplorer.BuildConfiguration.BuildStepList.0">
     <value type="int" key="ProjectExplorer.BuildStepList.StepsCount">0</value>
     <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DefaultDisplayName">Deploy</value>
     <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DisplayName">Deploy</value>
     <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.BuildSteps.Deploy</v
alue>
    </valuemap>
    <value type="int"
key="ProjectExplorer.BuildConfiguration.BuildStepListCount">1</value>
    <valuemap type="QVariantMap"
key="ProjectExplorer.DeployConfiguration.CustomData"/>
    <value type="bool"
key="ProjectExplorer.DeployConfiguration.CustomDataEnabled">false</value>
    <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">ProjectExplorer.DefaultDeployConfigu
ration</value>
   </valuemap>
   <value type="int"
key="ProjectExplorer.Target.DeployConfigurationCount">1</value>
   <valuemap type="QVariantMap" key="ProjectExplorer.Target.PluginSettings"/>
   <valuemap type="QVariantMap" key="ProjectExplorer.Target.RunConfiguration.0">
    <value type="QString" key="Analyzer.Perf.CallgraphMode">dwarf</value>
    <valuelist type="QVariantList" key="Analyzer.Perf.Events">
     <value type="QString">cpu-cycles</value>
    </valuelist>
    <valuelist type="QVariantList" key="Analyzer.Perf.ExtraArguments"/>
    <value type="int" key="Analyzer.Perf.Frequency">250</value>
    <valuelist type="QVariantList" key="Analyzer.Perf.RecordArguments">
     <value type="QString">-e</value>
     <value type="QString">cpu-cycles</value>
     <value type="QString">--call-graph</value>
     <value type="QString">dwarf,4096</value>
     <value type="QString">-F</value>
     <value type="QString">250</value>
    </valuelist>
    <value type="QString" key="Analyzer.Perf.SampleMode">-F</value>
    <value type="bool" key="Analyzer.Perf.Settings.UseGlobalSettings">true</value>
```

```
    <value type="int" key="Analyzer.Perf.StackSize">4096</value>
    <value type="bool" key="Analyzer.QmlProfiler.AggregateTraces">false</value>
    <value type="bool" key="Analyzer.QmlProfiler.FlushEnabled">false</value>
    <value type="uint" key="Analyzer.QmlProfiler.FlushInterval">1000</value>
    <value type="QString" key="Analyzer.QmlProfiler.LastTraceFile"></value>
    <value type="bool"
key="Analyzer.QmlProfiler.Settings.UseGlobalSettings">true</value>
    <valuelist type="QVariantList" key="Analyzer.Valgrind.AddedSuppressionFiles"/>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.CollectBusEvents">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.CollectSystime">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.EnableBranchSim">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.EnableCacheSim">false</value>
    <value type="bool"
key="Analyzer.Valgrind.Callgrind.EnableEventToolTips">true</value>
    <value type="double"
key="Analyzer.Valgrind.Callgrind.MinimumCostRatio">0.01</value>
    <value type="double"
key="Analyzer.Valgrind.Callgrind.VisualisationMinimumCostRatio">10</value>
    <value type="bool" key="Analyzer.Valgrind.FilterExternalIssues">true</value>
    <value type="QString"
key="Analyzer.Valgrind.KCachegrindExecutable">kcachegrind</value>
    <value type="int" key="Analyzer.Valgrind.LeakCheckOnFinish">1</value>
    <value type="int" key="Analyzer.Valgrind.NumCallers">25</value>
    <valuelist type="QVariantList"
key="Analyzer.Valgrind.RemovedSuppressionFiles"/>
    <value type="int" key="Analyzer.Valgrind.SelfModifyingCodeDetection">1</value>
    <value type="bool"
key="Analyzer.Valgrind.Settings.UseGlobalSettings">true</value>
    <value type="bool" key="Analyzer.Valgrind.ShowReachable">false</value>
    <value type="bool" key="Analyzer.Valgrind.TrackOrigins">true</value>
    <value type="QString"
key="Analyzer.Valgrind.ValgrindExecutable">valgrind</value>
    <valuelist type="QVariantList" key="Analyzer.Valgrind.VisibleErrorKinds">
     <value type="int">0</value>
     <value type="int">1</value>
     <value type="int">2</value>
     <value type="int">3</value>
     <value type="int">4</value>
     <value type="int">5</value>
     <value type="int">6</value>
     <value type="int">7</value>
     <value type="int">8</value>
     <value type="int">9</value>
     <value type="int">10</value>
     <value type="int">11</value>
     <value type="int">12</value>
     <value type="int">13</value>
     <value type="int">14</value>
    </valuelist>
    <valuelist type="QVariantList" key="CustomOutputParsers"/>
    <value type="int" key="PE.EnvironmentAspect.Base">2</value>
    <valuelist type="QVariantList" key="PE.EnvironmentAspect.Changes"/>
    <value type="QString"
key="ProjectExplorer.ProjectConfiguration.DisplayName">trafficlightsresolver</valu
e>
    <value type="QString"
key="ProjectExplorer.ProjectConfiguration.Id">PythonEditor.RunConfiguration./Users
/yurii.mykhailiak/Projects/intelligent-traffic-lights/trafficlightsresolver.py</va
lue>
    <value type="QString"
key="ProjectExplorer.RunConfiguration.BuildKey">/Users/yurii.mykhailiak/Projects/i
ntelligent-traffic-lights/trafficlightsresolver.py</value>
    <value type="bool" key="PythonEditor.RunConfiguation.Buffered">false</value>
    <value type="QString"
key="PythonEditor.RunConfiguation.Interpreter">{fd053d1a-0bec-4e9e-8e0d-39e55f1ae3
60}</value>
    <value type="QString"
key="PythonEditor.RunConfiguation.Script">/Users/yurii.mykhailiak/Projects/intelli
gent-traffic-lights/trafficlightsresolver.py</value>
    <value type="QString" key="RunConfiguration.Arguments"></value>
    <value type="bool" key="RunConfiguration.Arguments.multi">false</value>
    <value type="QString" key="RunConfiguration.OverrideDebuggerStartup"></value>
    <value type="bool" key="RunConfiguration.UseCppDebugger">false</value>
    <value type="bool" key="RunConfiguration.UseCppDebuggerAuto">true</value>
    <value type="bool" key="RunConfiguration.UseMultiProcess">false</value>
    <value type="bool" key="RunConfiguration.UseQmlDebugger">false</value>
    <value type="bool" key="RunConfiguration.UseQmlDebuggerAuto">true</value>
    <value type="QString" key="RunConfiguration.WorkingDirectory"></value>
    <value type="QString"
key="RunConfiguration.WorkingDirectory.default">/Users/yurii.mykhailiak/Projects/i
ntelligent-traffic-lights</value>
   </valuemap>
   <value type="int" key="ProjectExplorer.Target.RunConfigurationCount">1</value>
  </valuemap>
 </data>
 <data>
  <variable>ProjectExplorer.Project.TargetCount</variable>
  <value type="int">1</value>
 </data>
 <data>
  <variable>ProjectExplorer.Project.Updater.FileVersion</variable>
  <value type="int">22</value>
 </data>
 <data>
  <variable>Version</variable>
  <value type="int">22</value>
 </data>
</qtcreator>
```

```
## Intelligent traffic lights
#### Diploma project

System determines traffic load and resolves it.
Detection is based on [YOLO](https://pjreddie.com/darknet/yolo/) system with
pre-trained dnn model.

To deploy the project on your machine do items below:
- `sudo apt-get install python3`
- `pip install numpy`
- `pip install opencv-python`
- Create a `models` directory and move model to it
- `wget https://pjreddie.com/media/files/yolov3.weights && mv yolov3.weights
models/`
- Install
[yolo3.cfg](https://raw.githubusercontent.com/pjreddie/darknet/master/cfg/yolov3.c
fg) to `models`
- Install
[yolov3.txt](https://raw.githubusercontent.com/arunponnusamy/object-detection-open
cv/master/yolov3.txt) to `models`
- Create a `videos` directory and move all needed videos to it
- Set needed video names to `VIDEOS` array
- `python3 main.py`
- Enjoy it :relaxed:
from base_plugin import BasePlugin
from typing import List

class PluginsComposer(BasePlugin):
    plugins = []

    def __init__(self, plugins: List[BasePlugin]):
        self.plugins = plugins

    def on_direction_changed(self, controller, info):
        for plugin in self.plugins:
            plugin.on_direction_changed(self, controller, info)

    def on_equal_traffic_load_detection(self, controller):
        for plugin in self.plugins:
            plugin.on_equal_traffic_load_detection(self, controller)

    def on_move_to_standard_mode(self, controller):
        for plugin in self.plugins:
            plugin.on_move_to_standard_mode(self, controller)

class BasePlugin:
    """BasePlugin

    Introduces event-driven logic.
    Possible handlers:
        on_move_to_standard_mode
        on_equal_traffic_load_detection
        on_direction_changed
    """
    name = 'Basic plugin'

    def on_move_to_standard_mode(self, controller):
        """Handler for changing system to default mode event"""
        pass

    def on_equal_traffic_load_detection(self, controller):
        """Handler for detection of equal traffic load event"""
        pass

    def on_direction_changed(self, controller, info):
        """Handler for changing traffic state event"""
        pass
MASTER]

# A comma-separated list of package or module names from where C extensions may
# be loaded. Extensions are loading into the active Python interpreter and may
# run arbitrary code
extension-pkg-whitelist=cv2,cv2.dnn

# Add files or directories to the blacklist. They should be base names, not
# paths.
ignore=CVS

# Add files or directories matching the regex patterns to the blacklist. The
# regex matches against base names, not paths.
ignore-patterns=

# Python code to execute, usually for sys.path manipulation such as
# pygtk.require().
#init-hook=

# Use multiple processes to speed up Pylint.
jobs=1

# List of plugins (as comma separated values of python modules names) to load,
# usually to register additional checkers.
load-plugins=

# Pickle collected data for later comparisons.
persistent=yes

# Specify a configuration file.
#rcfile=

# When enabled, pylint would attempt to guess common misconfiguration and emit
# user-friendly hints instead of false-positive error messages
suggestion-mode=yes

# Allow loading of arbitrary C extensions. Extensions are imported into the
# active Python interpreter and may run arbitrary code.
unsafe-load-any-extension=no


[MESSAGES CONTROL]

# Only show warnings with the listed confidence levels. Leave empty to show
# all. Valid levels: HIGH, INFERENCE, INFERENCE_FAILURE, UNDEFINED
confidence=

# Disable the message, report, category or checker with the given id(s). You
# can either give multiple identifiers separated by comma (,) or put this
# option multiple times (only on the command line, not in the configuration
# file where it should appear only once).You can also use "--disable=all" to
# disable everything first and then reenable specific checks. For example, if
# you want to run only the similarities checker, you can use "--disable=all
# --enable=similarities". If you want to run only the classes checker, but have
# no Warning level messages displayed, use"--disable=all --enable=classes
# --disable=W"
disable=print-statement,
        parameter-unpacking,
        unpacking-in-except,
        old-raise-syntax,
        backtick,
        long-suffix,
        old-ne-operator,
        old-octal-literal,
```

```
        import-star-module-level,
        non-ascii-bytes-literal,
        invalid-unicode-literal,
        raw-checker-failed,
        bad-inline-option,
        locally-disabled,
        locally-enabled,
        file-ignored,
        suppressed-message,
        useless-suppression,
        deprecated-pragma,
        apply-builtin,
        basestring-builtin,
        buffer-builtin,
        cmp-builtin,
        coerce-builtin,
        execfile-builtin,
        file-builtin,
        long-builtin,
        raw_input-builtin,
        reduce-builtin,
        standarderror-builtin,
        unicode-builtin,
        xrange-builtin,
        coerce-method,
        delslice-method,
        getslice-method,
        setslice-method,
        no-absolute-import,
        old-division,
        dict-iter-method,
        dict-view-method,
        next-method-called,
        metaclass-assignment,
        indexing-exception,
        raising-string,
        reload-builtin,
        oct-method,
        hex-method,
        nonzero-method,
        cmp-method,
        input-builtin,
        round-builtin,
        intern-builtin,
        unichr-builtin,
        map-builtin-not-iterating,
        zip-builtin-not-iterating,
        range-builtin-not-iterating,
        filter-builtin-not-iterating,
        using-cmp-argument,
        eq-without-hash,
        div-method,
        idiv-method,
        rdiv-method,
        exception-message-attribute,
        invalid-str-codec,
        sys-max-int,
        bad-python3-import,
        deprecated-string-function,
        deprecated-str-translate-call,
        deprecated-itertools-function,
        deprecated-types-field,
        next-method-defined,
        dict-items-not-iterating,
        dict-keys-not-iterating,
        dict-values-not-iterating,
        deprecated-operator-function,
        deprecated-urllib-function,
        xreadlines-attribute,
        deprecated-sys-function,
        exception-escape,
        comprehension-escape

# Enable the message, report, category or checker with the given id(s). You can
# either give multiple identifier separated by comma (,) or put this option
# multiple time (only on the command line, not in the configuration file where
# it should appear only once). See also the "--disable" option for examples.
enable=c-extension-no-member


[REPORTS]

# Python expression which should return a note less than 10 (10 is the highest
# note). You have access to the variables errors warning, statement which
# respectively contain the number of errors / warnings messages and the total
# number of statements analyzed. This is used by the global evaluation report
# (RP0004).
evaluation=10.0 - ((float(5 * error + warning + refactor + convention) /
statement) * 10)

# Template used to display messages. This is a python new-style format string
# used to format the message information. See doc for all details
#msg-template=

# Set the output format. Available formats are text, parseable, colorized, json
# and msvs (visual studio).You can also give a reporter class, eg
# mypackage.mymodule.MyReporterClass.
output-format=text

# Tells whether to display a full report or only the messages
reports=no

# Activate the evaluation score.
score=yes


[REFACTORING]

# Maximum number of nested blocks for function / method body
max-nested-blocks=5

# Complete name of functions that never returns. When checking for
# inconsistent-return-statements if a never returning function is called then
# it will be considered as an explicit return statement and no message will be
# printed.
never-returning-functions=optparse.Values,sys.exit


[TYPECHECK]

# List of decorators that produce context managers, such as
# contextlib.contextmanager. Add to this list to register other decorators that
# produce valid context managers.
contextmanager-decorators=contextlib.contextmanager

# List of members which are set dynamically and missed by pylint inference
# system, and so shouldn't trigger E1101 when accessed. Python regular
# expressions are accepted.
generated-members=

# Tells whether missing members accessed in mixin class should be ignored. A
# mixin class is detected if its name ends with "mixin" (case insensitive).
ignore-mixin-members=yes

# This flag controls whether pylint should warn about no-member and similar
# checks whenever an opaque object is returned when inferring. The inference
# can return multiple potential results while evaluating a Python object, but
# some branches might not be evaluated, which results in partial inference. In
# that case, it might be useful to still emit no-member and other checks for
# the rest of the inferred objects.
ignore-on-opaque-inference=yes

# List of class names for which member attributes should not be checked (useful
# for classes with dynamically set attributes). This supports the use of
# qualified names.
ignored-classes=optparse.Values,thread._local,_thread._local

# List of module names for which member attributes should not be checked
# (useful for modules/projects where namespaces are manipulated during runtime
# and thus existing member attributes cannot be deduced by static analysis. It
# supports qualified module names, as well as Unix pattern matching.
ignored-modules=

# Show a hint with possible names when a member name was not found. The aspect
# of finding the hint is based on edit distance.
missing-member-hint=yes

# The minimum edit distance a name should have in order to be considered a
# similar match for a missing member name.
missing-member-hint-distance=1

# The total number of similar names that should be taken in consideration when
# showing a hint for a missing member.
missing-member-max-choices=1


[MISCELLANEOUS]

# List of note tags to take in consideration, separated by a comma.
notes=FIXME,
      XXX,
      TODO


[LOGGING]

# Logging modules to check that the string format arguments are in logging
# function parameter format
logging-modules=logging


[VARIABLES]

# List of additional names supposed to be defined in builtins. Remember that
# you should avoid to define new builtins when possible.
additional-builtins=

# Tells whether unused global variables should be treated as a violation.
allow-global-unused-variables=yes

# List of strings which can identify a callback function by name. A callback
# name must start or end with one of those strings.
callbacks=cb_,
          _cb

# A regular expression matching the name of dummy variables (i.e. expectedly
# not used).
dummy-variables-rgx=_+$|(_[a-zA-Z0-9_]*[a-zA-Z0-9]+?$)|dummy|^ignored_|^unused_

# Argument names that match this expression will be ignored. Default to name
# with leading underscore
ignored-argument-names=_.*|^ignored_|^unused_

# Tells whether we should check for unused import in __init__ files.
init-import=no

# List of qualified module names which can have objects that can redefine
# builtins.
redefining-builtins-modules=six.moves,past.builtins,future.builtins,io,builtins


[SIMILARITIES]

# Ignore comments when computing similarities.
ignore-comments=yes

# Ignore docstrings when computing similarities.
ignore-docstrings=yes

# Ignore imports when computing similarities.
ignore-imports=no

# Minimum lines number of a similarity.
min-similarity-lines=4


[FORMAT]

# Expected format of line ending, e.g. empty (any line ending), LF or CRLF.
expected-line-ending-format=

# Regexp for a line that is allowed to be longer than the limit.
ignore-long-lines=^\s*(# )?<?https?://\S+>?$
```

```
# Number of spaces of indent required inside a hanging  or continued line.
indent-after-paren=4

# String used as indentation unit. This is usually "    " (4 spaces) or "\t" (1
# tab).
indent-string='    '

# Maximum number of characters on a single line.
max-line-length=100

# Maximum number of lines in a module
max-module-lines=1000

# List of optional constructs for which whitespace checking is disabled. `dict-
# separator` is used to allow tabulation in dicts, etc.: {1  : 1,\n222: 2}.
# `trailing-comma` allows a space between comma and closing bracket: (a, ).
# `empty-line` allows space-only lines.
no-space-check=trailing-comma,
               dict-separator

# Allow the body of a class to be on the same line as the declaration if body
# contains single statement.
single-line-class-stmt=no

# Allow the body of an if to be on the same line as the test if there is no
# else.
single-line-if-stmt=no


[BASIC]

# Naming style matching correct argument names
argument-naming-style=snake_case

# Regular expression matching correct argument names. Overrides argument-
# naming-style
#argument-rgx=

# Naming style matching correct attribute names
attr-naming-style=snake_case

# Regular expression matching correct attribute names. Overrides attr-naming-
# style
#attr-rgx=

# Bad variable names which should always be refused, separated by a comma
bad-names=foo,
          bar,
          baz,
          toto,
          tutu,
          tata

# Naming style matching correct class attribute names
class-attribute-naming-style=any

# Regular expression matching correct class attribute names. Overrides class-
# attribute-naming-style
#class-attribute-rgx=

# Naming style matching correct class names
class-naming-style=PascalCase

# Regular expression matching correct class names. Overrides class-naming-style
#class-rgx=

# Naming style matching correct constant names
const-naming-style=UPPER_CASE

# Regular expression matching correct constant names. Overrides const-naming-
# style
#const-rgx=

# Minimum line length for functions/classes that require docstrings, shorter
# ones are exempt.
docstring-min-length=-1

# Naming style matching correct function names
function-naming-style=snake_case

# Regular expression matching correct function names. Overrides function-
# naming-style
#function-rgx=

# Good variable names which should always be accepted, separated by a comma
good-names=i,
           j,
           k,
           ex,
           Run,
           _

# Include a hint for the correct naming format with invalid-name
include-naming-hint=no

# Naming style matching correct inline iteration names
inlinevar-naming-style=any

# Regular expression matching correct inline iteration names. Overrides
# inlinevar-naming-style
#inlinevar-rgx=

# Naming style matching correct method names
method-naming-style=snake_case

# Regular expression matching correct method names. Overrides method-naming-
# style
#method-rgx=

# Naming style matching correct module names
module-naming-style=snake_case

# Regular expression matching correct module names. Overrides module-naming-
# style
#module-rgx=

# Colon-delimited sets of names that determine each other's naming style when
# the name regexes allow several styles.
name-group=

# Regular expression which should only match function or class names that do
# not require a docstring.
no-docstring-rgx=^_

# List of decorators that produce properties, such as abc.abstractproperty. Add
# to this list to register other decorators that produce valid properties.
property-classes=abc.abstractproperty

# Naming style matching correct variable names
variable-naming-style=snake_case

# Regular expression matching correct variable names. Overrides variable-
# naming-style
#variable-rgx=


[SPELLING]

# Limits count of emitted suggestions for spelling mistakes
max-spelling-suggestions=4

# Spelling dictionary name. Available dictionaries: none. To make it working
# install python-enchant package.
spelling-dict=

# List of comma separated words that should not be checked.
spelling-ignore-words=

# A path to a file that contains private dictionary; one word per line.
spelling-private-dict-file=

# Tells whether to store unknown words to indicated private dictionary in
# --spelling-private-dict-file option instead of raising a message.
spelling-store-unknown-words=no


[DESIGN]

# Maximum number of arguments for function / method
max-args=5

# Maximum number of attributes for a class (see R0902).
max-attributes=7

# Maximum number of boolean expressions in a if statement
max-bool-expr=5

# Maximum number of branch for function / method body
max-branches=12

# Maximum number of locals for function / method body
max-locals=15

# Maximum number of parents for a class (see R0901).
max-parents=7

# Maximum number of public methods for a class (see R0904).
max-public-methods=20

# Maximum number of return / yield for function / method body
max-returns=6

# Maximum number of statements in function / method body
max-statements=50

# Minimum number of public methods for a class (see R0903).
min-public-methods=2


[CLASSES]

# List of method names used to declare (i.e. assign) instance attributes.
defining-attr-methods=__init__,
                      __new__,
                      setUp

# List of member names, which should be excluded from the protected access
# warning.
exclude-protected=_asdict,
                  _fields,
                  _replace,
                  _source,
                  _make

# List of valid names for the first argument in a class method.
valid-classmethod-first-arg=cls

# List of valid names for the first argument in a metaclass class method.
valid-metaclass-classmethod-first-arg=mcs


[IMPORTS]

# Allow wildcard imports from modules that define __all__.
allow-wildcard-with-all=no

# Analyse import fallback blocks. This can be used to support both Python 2 and
# 3 compatible code, which means that the block might have code that exists
# only in one or another interpreter, leading to false positives when analysed.
analyse-fallback-blocks=no

# Deprecated modules which should not be used, separated by a comma
deprecated-modules=regsub,
                   TERMIOS,
                   Bastion,
                   rexec

# Create a graph of external dependencies in the given file (report RP0402 must
# not be disabled)
ext-import-graph=

# Create a graph of every (i.e. internal and external) dependencies in the
# given file (report RP0402 must not be disabled)
import-graph=
```

```
# Create a graph of internal dependencies in the given file (report RP0402 must
# not be disabled)
int-import-graph=

# Force import order to recognize a module as part of the standard
# compatibility libraries.
known-standard-library=

# Force import order to recognize a module as part of a third party library.
known-third-party=enchant


[EXCEPTIONS]

# Exceptions that will emit a warning when being caught. Defaults to
# "Exception"
overgeneral-exceptions=Exception
Classes:
person
bicycle
car
motorcycle
airplane
bus
train
truck
boat
traffic light
fire hydrant
stop sign
parking meter
bench
bird
cat
dog
horse
sheep
cow
elephant
bear
zebra
giraffe
backpack
umbrella
handbag
tie
suitcase
frisbee
skis
snowboard
sports ball
kite
baseball bat
baseball glove
skateboard
surfboard
tennis racket
bottle
wine glass
cup
fork
knife
spoon
bowl
banana
apple
sandwich
orange
broccoli
carrot
hot dog
pizza
donut
cake
chair
couch
potted plant
bed
dining table
toilet
tv
laptop
mouse
remote
keyboard
cell phone
microwave
oven
toaster
sink
refrigerator
book
clock
vase
scissors
teddy bear
hair drier
toothbrush
[net]
# Testing
batch=1
subdivisions=1
# Training
# batch=64
# subdivisions=2
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 500200
policy=steps
steps=400000,450000
scales=.1,.1

[convolutional]
batch_normalize=1
filters=16
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=1

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

###########

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=255
activation=linear



[yolo]
mask = 3,4,5
anchors = 10,14,  23,27,  37,58,  81,82,  135,169,  344,319
classes=80
```

```
num=6
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1

[route]
layers = -4

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[upsample]
stride=2

[route]
layers = -1, 8

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=255
activation=linear

[yolo]
mask = 0,1,2
anchors = 10,14,  23,27,  37,58,  81,82,  135,169,  344,319
classes=80
num=6
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
from base_plugin import BasePlugin
from app_logger import logger

class LoggerPlugin(BasePlugin):
    """LoggerPlugin"""
    name = 'Logger plugin'

    def on_move_to_standard_mode(self, controller):
        """Handler for changing system to default mode event"""
        logger.info('System has been moved to standard mode')

    def on_equal_traffic_load_detection(self, controller):
        """Handler for detection of equal traffic load event"""
        logger.info('Traffic load is the same; System was moved to standard
behavior for {} seconds'.format(controller.MAX_PASS))

    def on_direction_changed(self, controller, info):
        """Handler for changing traffic state event"""
        direction, computed_time, transport_number, actual_time = info
        logger.info('''
            Green direction: {0};
            Needed time: {1};
            Count of transport: {2};
            Actual time: {3}
        '''.format(direction, computed_time, transport_number, actual_time))
        logger.info('Directions state: {}'.format(controller.directions_load))
import sys
from subprocess import Popen, PIPE, STDOUT
from threading import Thread
from app_logger import logger
from controller import Controller
from traffic_lights import TrafficLights
from logger import LoggerPlugin
from trafficlightsresolver import TrafficLightsResolver
from PySide2.QtWidgets import QApplication

BASIC_COMMAND = 'sh ./start.sh'
videos = sys.argv[1:]
cmds_list = ['{0} -v {1}'.format(BASIC_COMMAND, p) for p in iter(videos)]
app = Controller(TrafficLights(), [LoggerPlugin])


def execute(cmd: str, index: int, app: object):
    """Execute sh script which opens detector"""
    proc = Popen('{0} -i {1}'.format(cmd, index),
                 shell=True,
                 stdout=PIPE,
                 stderr=STDOUT)

    for line in iter(proc.stdout.readline, b''):
        try:
            app.set_data(index, int(line.rstrip()))
        except ValueError:
            logger.info('Camera ID#{} - {}'.format(index,
                                                   line.decode('utf-8')))
        except Exception:
            logger.error(
                'Camera ID#{} has been destroyed. System will be switched to
standard mode'
                .format(index))
            app.move_to_standard_mode()


for idx, cmd in enumerate(cmds_list):
    p = Thread(target=execute, args=[cmd, idx, app])
    p.start()

# Manual control
if use_manual_control:
    TrafficLightsResolver.exec(app)
```

Інформаційна система розумного світлофора для регулювання дорожнього трафіку

Актуальність

Світлофорні системи на перехрестях Україні працюють за найпростішим принципом пофазної дії первинного налаштування, не враховуючи вхідні дані, як завантаження доріг на момент часу, що призводить до суттєвої інерційності процесу керування і відсутньої можливості оперативного реагування в режимі реального часу на дорожню ситуацію.

Завдяки сучасним технологіям можна змінити дану ситуацію, а саме оперативно регулювати дорожній рух спираючись на вхідні дані поточної завантаженості доріг. Це дозволить ефективно використовувати час учасників дорожнього руху і покращити екологічну ситуації в містах.

Мета і завдання

Мета роботи полягає у реалізація інформаційної системи розумного світлофора для регулювання дорожнього трафіку з використанням мультиканального режиму прийому даних.

Для досягнення поставленої мети визначенні наступні задачі-дослідження:

- провести аналіз існуючих методів, технологій та рішень регулювання дорожнього трафіку;
- удосконалити існуючі методи регулювання дорожнього трафіку з використанням розумного світлофора у напрямку полегшення їх інтегрування в наявну транспортну інфраструктуру;
- розробити інформаційну систему розумного світлофора для регулювання дорожнього трафіку за допомогою отриманих моделей та методів;
- виконати експериментальну перевірку інформаційної системи регулювання дорожнього трафіку.

Наукова новизна

В результаті проведеної роботи були отримані наступні результати:

- удосконалено існуючий метод регулювання дорожнього трафіку на перехрестях за допомогою розумного світлофора в мультиканальному режимі прийому даних;
- досліджено ефективність використання часткового і повного режиму ідентифікації дорожнього трафіку;
- розроблений алгоритм ручного керування світлофорною ділянкою поверх автоматичного режиму.

Практична цінність

В результаті виконання дипломної роботи магістра розроблено відповідне експериментальне програмне забезпечення, яке підтвердило вірність запропонованих положень. Застосування інформаційної системи дає можливість здійснювати регулювання дорожнього руху в режимі реального часу, практично миттєво реагуючи на зміни дорожнього трафіку, а її інтеграція в наявну транспортну інфраструктуру не потребує її суттєвої модернізації. Наведено рекомендації щодо режиму роботи інформаційної системи.

Публікація

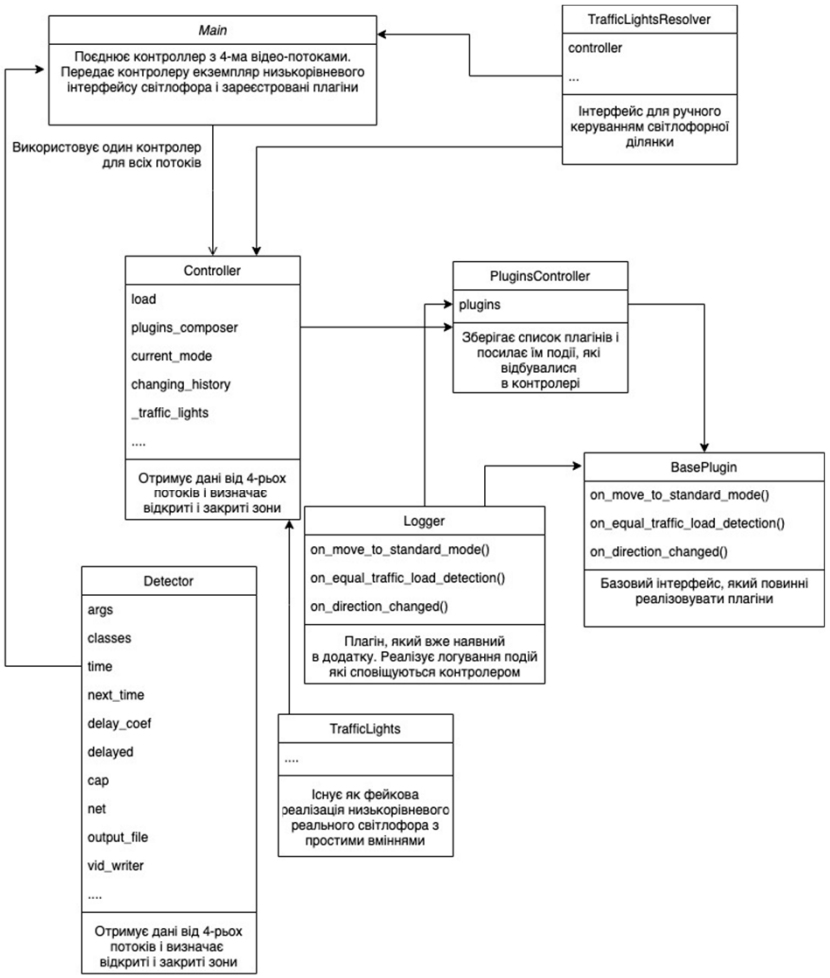
Основні наукові та практичні результати опубліковані в наукових виданнях МОН України:

- публікація на тему “Інформаційна система розумного світлофора для регулювання дорожнього трафіку” в науковому журналі “Вісник Хмельницького національного університету”.

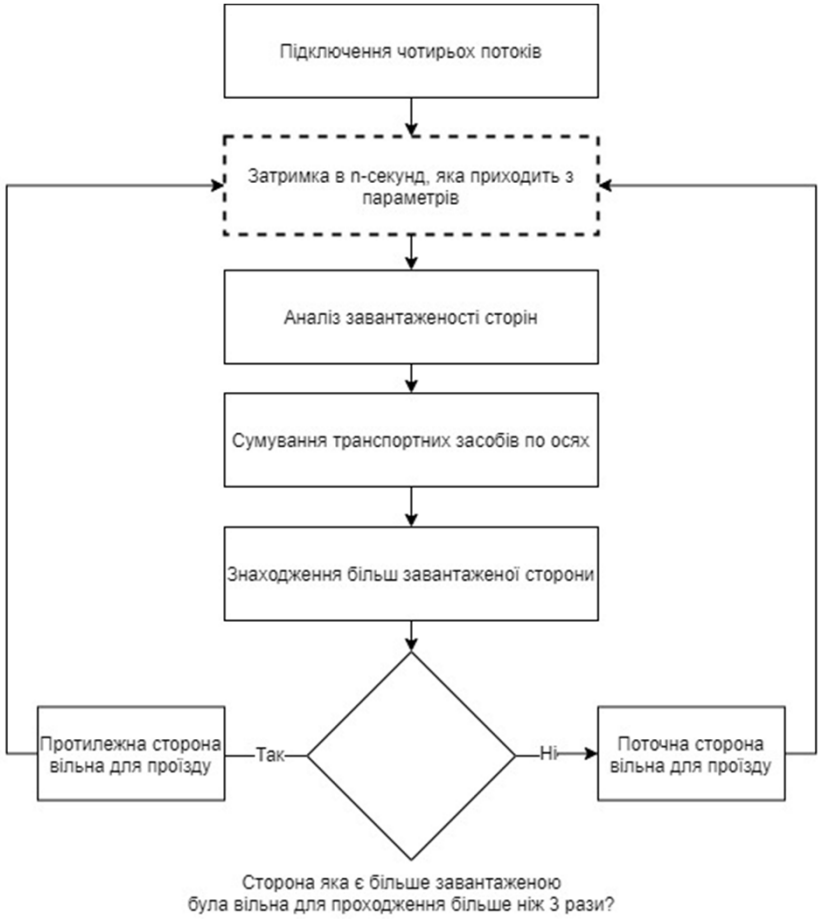
За темою дипломної роботи магістра автором виконано одну наукову публікацію:

Ю.Б. Михайляк, О.А. Пасічник, Т.К. Скрипник. Інформаційна система розумного світлофора для регулювання дорожнього трафіку // Вісник Хмельницького національного університету. 2020

Діаграма класів



Основний Принцип роботи



Результати роботи

Система була розроблена на найбільш поширеному типі перехресть, де чотири ділянки дороги перетинаються, але з незначними змінами не буде проблем адаптації такої системи під інші види перехресть.

В усіх проаналізованих ситуаціях відповідного високого режиму роботи було отримано повне співпадіння, відсутні випадки хибної ідентифікації та випадки пропущених об’єктів для ідентифікації. При застосуванні легкого режиму, який потребує суттєво менших апаратних ресурсів, мало місце до 40% пропущених об’єктів.

Загальні висновки

Реалізовано інформаційну системи розумного світлофора для регулювання дорожнього трафіку з використанням мультиканального режиму прийому даних з відкритою архітектурою.

Проведена експериментальна апробація результатів роботи реалізованої інформаційної системи регулювання дорожнього трафіку в реальному часі. Як приклад був виконаний аналіз однофреймових відео. Для оцінки ефективності роботи системи виконано аналіз ряду зображень з кількістю об’єктів для ідентифікації (автомобілів) складала від 2-ох до 13-ти. У випадку застосування важкого режиму в усіх проаналізованих модельних ситуаціях було отримано повне співпадіння, відсутні випадки хибної ідентифікації та випадки пропущених об’єктів для ідентифікації. При застосуванні легкого режиму, який потребує суттєво менших апаратних ресурсів, мало місце до 40% пропущених об’єктів, тому він не може бути рекомендований для практичного застосування.

Mon Nov 30 12:59:41 EET 2020, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 4.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

	Документ		Сумарний збіг по Базі Даних	
ID: 81682 Назва: Інформаційна система розумного світлофора для регулювання дорожнього трафіку Додано в БД: 2020-11-30 Автора: Михайляк Юрій Богданович Керівники: Пасічник О.А., Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	52237	437	3991 (8%)	40 (9%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

РІШЕННЯ КАФЕДРИ **КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ** ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: **Інформаційна система розумного світлофора для регулювання дорожнього трафіку**
Автор: **Михайляк Ю.Б.**
Спеціальність: **122 Комп'ютерні науки**
Науковий керівник: **к.т.н., доцент Пасічник О.А.**

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	-
3	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	-
4	Інше:	-

Підтвердження: Виявленні запозичення не є плагіатом т.я. розміщенні в розділах, які не описують безпосередньо авторське дослідження, складають 5,86% та мають посилання на приведений список літературних джерел

02.12.2020
Дата

Підпис керівника

Підпис завідувача кафедри

Хмельницький національний університет

Кафедра комп'ютерних наук та інформаційних технологій

ВІДГУК ОПОНЕНТА

на дипломну роботу магістра

Магістра *гр. КНм-19-1 Михайляка Юрія Богдановича*

На тему: *Інформаційна система розумного світлофора для регулювання дорожнього трафіку.*

1. Актуальність і значення теми

Завдяки новітнім технологіям, стало можливо автоматизувати систему регулювання дорожнього трафіку за допомогою системи розумного світлофора. Ці технології вже давно використовуються за кордоном, де вони знайшли широке застосування. Це значно покращує стан дорожнього трафіку в густонаселених містах, також покращується екологічна ситуація, оскільки зменшується викид чадного газу під час очікування автівок на дозвіл світлофора про проходження перехрестя. На даний час в Україні цей метод регулювання дорожнього трафіку не використовується.

2.Оцінка якості та достовірності проведених досліджень.

Отримані результати добре співвідносяться з результатами, наведеними в наукових роботах і довідниках.

3. Оцінка запропонованих заходів та пропозицій, практичної цінності та ефективності.

Проведені дослідження представляють науково-технічну цінність. Дослідження в галузі регулювання дорожнього руху є ефективними, їх можна використати з метою підвищення кількості транспортних засобів, які проходять світлофорну ділянку за одиницю часу.

4. Загальний висновок та оцінка

Робота виконана в повному обсязі. Досліджені та результати входять в рамки допустимих відхилень. Пояснювальна записка оформлена в відповідності з нормами. Відмічені недоліки не знижують цінності дипломної роботи. За своєю структурою, практичними цінностями, поставленій меті та вирішеними задачами робота відповідає вимогам вищої школи і вимогам, що пред'являються до освітньо-кваліфікаційного рівня «магістр», а її автор Михайляк Ю.Б. заслуговує присвоєння кваліфікації магістра з комп'ютерних наук та інформаційних технологій.

Робота заслуговує на оцінку « добре ».

Опонент Мартинак В.В., д.т.н., проф.