

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ
УПРАВЛІННЯ РОБОТОМ-ДЕЗІНФЕКТОРОМ НА ОСНОВІ АНАЛІЗУ
ВІДЕОПОТОКУ
RESEARCH AND SOFTWARE IMPLEMENTATION OF DISINFECTION
ROBOT CONTROL SYSTEM BASED ON VIDEO FLOW ANALYSIS

Виконав: студент(ка) 2 курсу, групи КНТ-119м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Селіна А.О.

(прізвище та ініціали)

Керівник Степаненко О.О.

(прізвище та ініціали)

Рецензент Гофман Є.О.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Селіної Аліни Олександрівни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація системи управління роботом-дезінфектором на основі аналізу відеопотоку
Research and software implementation of disinfection robot control system based on video flow analysis

керівник проєкту (роботи) Степаненко Олександр Олексійович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “3” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) 10 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження.
2. Матеріали і методи. 3. Розробка архітектури програми. 4. Розробка програми.
5. Експлуатація та тестування програми. 6. Економіко-організаційна частина.
7. Охорона праці та безпека в надзвичайних ситуаціях.
8. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Степаненко О.О., доцент		
Економіко-організаційна частина	Пожуєва Т.О., професор		
Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Андрєєв М.О., асистент		

7. Дата видачі завдання “18” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4–5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Розробка економіко-організаційної частини	10 тиждень	Розділ 6
8	Розробка заходів з охорони праці та безпеки в надзвичайних ситуаціях.	11 тиждень	Розділ 7
9	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування.	14–15 тижні	
11	Захист роботи.	16 тиждень	

Студент(ка)

(підпис) Селіна А.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Степаненко О.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
132 с., 46 рис., 10 табл., 3 дод., 47 джерел.

СИСТЕМА УПРАВЛІННЯ, РОБОТОТЕХНІКА, РОБОТ-ДЕЗІНФЕКТОР,
АНАЛІЗ ЗОБРАЖЕНЬ, КЛАСИФІКАЦІЯ ОБ'ЄКТІВ, C++, ROS, DARKNET,
YOLOV4, OPENCV.

Об'єктом дослідження є система управління роботом-дезінфектором.

Предметом дослідження є програмне забезпечення для управління роботом-дезінфектором на основі аналізу відеопотоку.

Мета роботи – програмна реалізація системи управління бактерицидною ультрафіолетовою (УФ) лампою робота-дезінфектора на основі аналізу відеопотоку для забезпечення оптимального захисту людей, що знаходяться у приміщенні, яке дезінфікується.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, модель YOLOv4, DL-фреймворк Darknet, мова програмування C++, бібліотека Qt, бібліотека OpenCV, фреймворк Robotics Operating System (ROS), камера з кутом огляду 360° Insta360 Air, мікрокомп'ютер Jetson Nano Developer Kit.

Результати. Проаналізовано методи обробки й аналізу відеопотоку, які можуть застосовуватися для детектування людей, спроектовано та реалізовано програмний застосунок.

Висновки. Розроблено програмну систему, що на основі аналізу відеопотоку керує УФ лампою робота-дезінфектора, забезпечуючи захист людей від впливу ультрафіолетового випромінювання, що знаходяться у приміщенні під час дезінфекції.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 132 pages, 46 figures, 10 tables, 3 appendixes, 47 sources.

CONTROL SYSTEM, ROBOTICS, DISINFECTION ROBOT, IMAGE ANALYSIS, OBJECT CLASSIFICATION, C++, ROS, DARKNET, YOLOV4, OPENCV.

The object of the research is the control system of the disinfection robot.

The subject of the research is the software for controlling the disinfection robot based on video stream analysis.

The purpose of the work is the software implementation of the control system of the bactericidal ultraviolet lamp of the disinfection robot based on the video stream analysis to ensure optimal protection of people in the room that is being disinfected.

Materials, methods and technical means: object-oriented programming, YOLOv4 model, DL-framework Darknet, programming language C++, Qt library, OpenCV library, Robotics Operating System (ROS) framework, camera with 360° viewing angle Insta360 Air , microcomputer Jetson Nano Developer Kit.

Results. The methods of video stream processing and analysis that can be used to detect people are analyzed, the software application is designed and implemented.

Conclusions. The software system has been developed that, based on video stream analysis, controls the disinfection robot's ultraviolet lamp, protecting people from the effects of ultraviolet radiation in the room during disinfection.

ЗМІСТ

	С.
Перелік скорочень та умовних познач 8	8
Вступ..... 9	9
1 Аналіз проблеми та постановка завдань дослідження 10	10
1.1 Правила безпеки під час дезінфекції 10	10
1.2 Огляд існуючих роботів-дезінфекторів 10	10
1.3 Висновки за розділом 1 14	14
2 Матеріали і методи..... 15	15
2.1 Нейромережеві методи розпізнавання образів 15	15
2.2 Згорткові нейронні мережі..... 26	26
2.3 Модель YOLOv4 29	29
2.4 Метод обробки відеокадру..... 32	32
2.5 Система керування УФ лампою 37	37
2.6 Висновки за розділом 2 40	40
3 Опис програми..... 41	41
3.1 Визначення вимог до програмного забезпечення 41	41
3.2 Вибір мови програмування 42	42
3.3 Вибір фреймворку..... 45	45
3.4 Вибір бібліотеки для обробки відеопотоку..... 47	47
3.5 Висновки за розділом 3 49	49
4 Розробка програми 50	50
4.1 Опис реалізації структури програмного забезпечення 50	50
4.2 Опис розроблених модулів 51	51
4.3 Висновки за розділом 4 54	54
5 Експлуатація та тестування програми..... 55	55
5.1 Призначення програми..... 55	55
5.2 Умови виконання програми..... 55	55
5.3 Виконання програми 55	55
6 Економіко-організаційна частина 58	58

6.1 Планування розробки програмного виробу	58
6.2 Визначення витрат на розробку програми	61
6.3 Розрахунок капітальних вкладень.....	68
6.4 Оцінка економічної ефективності.....	69
7 Охорона праці та безпека в надзвичайних ситуаціях.....	71
7.1 Аналіз потенційних небезпек	71
7.2 Заходи із забезпечення безпеки.....	72
7.3 Заходи з виробничої санітарії і гігієни праці.....	75
7.4 Заходи безпеки у надзвичайних ситуаціях.....	85
Висновки	90
Перелік джерел посилання	92
Додаток А Технічне завдання	96
Додаток Б Текст програми	101
Додаток В Слайди презентації.....	125

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
ROS	– Robotics Operating System;
ЕОМ	– електронно-обчислювальна машина;
КТ	– комп'ютерна техніка;
ОЦ	– обчислювальний центр;
ПК	– персональний комп'ютер;
ПУЕ	– правила улаштування електроустановок;
УФ	– ультрафіолетове випромінювання або ультрафіолет.

ВСТУП

В медичних установах, які відвідують люди з різноманітними хворобами, гостро стоїть питання запобігання поширенню інфекції. В умовах скупчення великої кількості людей, особливо під час епідеміологічної ситуації, існує високий ризик зараження медичного та обслуговуючого персоналу, неінфекційних хворих і супроводжуючих осіб. Окрім використання засобів індивідуального захисту до важливих заходів профілактики відноситься знезараження повітря та поверхонь. Для цих цілей можуть застосовуватися УФ лампи, адже ультрафіолетове випромінювання може ефективно знищувати бактерії, мікроби і віруси. Їх також можна використовувати не лише в сфері охорони здоров'я, а й в дитячих установах, будь-яких місцях масового скупчення людей, й навіть в домашніх умовах.

Все більше лікарень, компаній та установ обирають автоматизовані рішення для дезінфекції приміщень і громадських місць.

Потреба в додаткових заходах санітарної обробки, обумовлена пандемією COVID-19, відкрила нові області застосування інноваційної робототехніки для дезінфекції різних приміщень [1].

Суворі вимоги до санітарної обробки, що висуваються лікарням і громадським установам в зв'язку з пандемією, створюють певні труднощі, що пов'язані з пошуком нових підходів і технологій. Мобільні роботи, оснащені УФ лампами, ідеально підходять для дезінфекції приміщень лікарень, торгових центрів та інших громадських місць. Автоматизація процесів дезінфекції за допомогою ультрафіолетового випромінювання підвищує ефективність заходів з охорони здоров'я та забезпечення безпеки, тим самим сприяючи боротьбі з COVID-19.

Тож мобільні роботи добре підходять для застосування з УФ лампами, які становлять небезпеку для здоров'я людини при прямому контакті [2].

АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Правила безпеки під час дезінфекції

Як зазначалось у минулому розділі, мобільні роботи-дезінфектори мають багато переваг: вони легкокеровані, автономні, дезінфекція приміщень роботами може здійснюватися як автоматично за заздалегідь заданою програмою, так і в режимі ручного дистанційного керування, роботи дозволяють більш ефективно планувати роботу офісного або медичного персоналу та мінімізувати контакт між людьми.

Але випромінювання бактерицидної УФ лампи є небезпечним для здоров'я людини – може викликати пошкодження очей та захворювання шкіри. Тому, обов'язковою умовою використання бактерицидних УФ ламп є виключення можливості шкідливого впливу на людину УФ опромінення [2].

Оскільки неможливо стовідсотково гарантувати відсутність людей у приміщенні під час дезінфекції, то розроблюваний робот-дезінфектор має бути обладнано системою безпеки, що буде виявляти, якщо хтось увійде у приміщення, де проводиться дезінфекція, і негайно припиняти дезінфекцію.

1.2 Огляд існуючих роботів-дезінфекторів

Мобільний робот-дезінфектор Model C (рис. 1.1) від компанії UVD Robots було розроблено на основі відгуків спеціалістів з профілактики інфекцій для регулярної дезінфекції та його ціллю є зменшення поширення інфекційних захворювань, бактерій та інших видів шкідливих органічних мікроорганізмів та патогенів у навколишньому середовищі [3].

До основних особливостей робота належать:

– залежно від тривалості дезінфекції, робот може знищити до 99,99% патогенів, таких як метицилін-резистентний золотистий стафілокок,

Clostridioides difficile та *Candida* phyla radiation. Він дезінфікує всі середовища, які потенційно містять небезпечні мікроорганізми;

- Model C має невеликі розміри і добре підходить для невеликих проїздів, як в поїздах і літаках. Його ширина складає 55 см і він може легко проїхати через стандартні двері;

- робот обладнано лазерними сканерами та 3D-камерами, які розпізнають перешкоди на своєму шляху. Ним можна керувати вручну дистанційно. За допомогою камер оператор може керувати роботом через планшет;

- робот живиться від Li-MNC акумуляторів, що дає змогу дезінфікувати більші площі, такі як аеропорти, супермаркети, конференц-зали та інше;

- він поставляється з док-станцією, яку робот самостійно знаходить, коли йому потрібно підзарядитись. Щоб переконатися, що він повністю заряджений і готовий до наступної дезінфекції, заощаджуючи час, оскільки не вимагає участі оператора;

- якщо хтось увійде в кімнату, де проводиться дезінфекція, то датчики робота це виявлять і припинять дезінфекцію;

- система безпеки буде спрацьовувати, якщо оператор вручну підведе робота занадто близько до стіни, людини або іншого об'єкту висотою більше 15 см. Якщо це станеться, передній і задній датчики робота змусять його зупинитися, запобігаючи зіткненню з об'єктом [3].



Рисунок 1.1 – Мобільний робот-дезінфектор Model C від компанії UVD Robots [3]

Робот TMiRob Intelligent Disinfection Robot (рис. 1.2) від компанії Shanghai TMI Robotics було розроблено спеціально для застосування в лікарнях і клінічних установах для дезінфекції повітря і поверхонь об'єктів [4].

Він має наступні особливості:

- робот об'єднує 3 режими дезінфекції: ультрафіолет, перекис водню і фільтрацію повітря, щоб задовольнити суворі вимоги до дезінфекції в секторі охорони здоров'я [5];

- автономний рух: карту плану поверху можна імпортувати в робота або він може працювати в режимі автономного виявлення. Користувач також може налаштувати систему за допомогою настроюваного користувачем режиму дезінфекції, позиційного контролю, автоматичного маршруту, автоматичного повернення додому та підзарядки;

- висока ефективність: робот може максимізувати ефективність дезінфекції, вивчаючи карту і мінімізуючи час дезінфекції;

– точність: робот має розпилювальну головку для дезінфекції з кутом повороту 360 градусів;

– робот пройшов клінічні та польові випробування, які підтвердили, що знищує 99,99% усіх бактерій та виконує дезінфекцію за 10 хвилин (для площі 5x5 метрів). Також доступний швидкий режим для отримання 95% дезінфекції протягом 6 хвилин;

– за допомогою датчика технічного зору та алгоритму розпізнавання зображень, робот забезпечує безпеку персоналу в робочій зоні, нагадуючи персоналу про евакуацію перед дезінфекцією. Якщо хто-небудь із співробітників виявиться в робочій зоні, час початку дезінфекції буде відкладено до тих пір, доки персонал не залишить об'єкт. В процесі дезінфекції, якщо хтось раптово втрутиться, робот зупинить дезінфекцію ультрафіолетом або перекисом водню і подасть голосове повідомлення, щоб переконатись в безпеці персоналу в процесі дезінфекції [6].



Рисунок 1.2 – Робот TMI Rob Intelligent Disinfection Robot від компанії Shanghai TMI Robotics [6]

1.3 Висновки за розділом 1

Розглянуто проблему забезпечення захисту людей від УФ опромінення при використанні мобільних роботів-дезінфекторів.

Виконано огляд існуючих роботів, які можуть використовуватися для дезінфекції приміщень. Model C від компанії UVD Robots є більш універсальним рішенням, що може використовуватися у лікарнях, офісах, готелях, аеропортах, торговельних центрах, житлових будинках та інших приміщеннях, а Робот TMI Rob Intelligent Disinfection Robot від компанії Shanghai TMI Robotics – в медичних установах.

Далі для забезпечення виконання цілі роботи необхідно реалізувати виконання наступних завдань.

Дослідження методів обробки та аналізу відеокadrів. В роботі необхідно дослідити, які засоби можуть використовуватися для конвертування зображень отриманих за допомогою об'єктива “fish-eye” до вигляду сферичної проекції та аналізу цих зображень для детектування на них людей.

Дослідження розподіленої системи на базі системи ROS, а саме обмін повідомленнями між різними вузлами, які відповідають за різні системи робота-дезінфектора.

Розробка архітектури програмного забезпечення. Необхідно визначити структуру програмного забезпечення, вибрати мову програмування, бібліотеки та фреймворки для реалізації та визначити функціональні вимоги до програмного забезпечення.

Розробка програмного забезпечення. Необхідно реалізувати проектовану систему у вигляді програмного застосунку.

Тестування та відлагодження програми. Необхідно запровадити на практиці розроблений застосунок.

Економічне обґрунтування доцільності розробки та забезпечення заходів охорони праці та безпеки в надзвичайних ситуаціях.

МАТЕРІАЛИ І МЕТОДИ

Керування УФ лампою робота-дезінфектора базується на аналізі 360° відеопотоку і розпізнаванні людей на відеокадрах. При детектуванні людини у приміщенні, що дезінфікується, дезінфекція припиняється та УФ лампа робота вимикається. Якщо людей у приміщенні більше немає, то робот продовжить дезінфекцію приміщення та увімкне УФ лампу.

2.1 Нейромережеві методи розпізнавання образів

2.1.1 Нейронні мережі та глибоке навчання

Штучна нейронна мережа – це спрощена модель біологічної нейронної мережі, що являє собою сукупність штучних нейронів, що взаємодіють між собою [7].

Основні принципи роботи нейронних мереж були описані в 1943 році Уорреном Мак-Каллоком і Уолтером Питтсом. У 1957 році нейрофізіолог Френк Розенблат розробив першу нейронну мережу. У 2006 були відкриті техніки навчання так званих глибоких нейромереж. Тепер ці техніки відомі, як глибоке навчання. А в 2010 році великі обсяги даних для навчання відкрили можливість використовувати нейронні мережі для машинного навчання [7].

Нейронні мережі не запрограмовані в звичному розумінні цього слова, а навчені. Навчання є однією з основних переваг нейромереж над традиційними алгоритмами. Процес навчання полягає в тому, щоб знайти коефіцієнти зв'язків між нейронами. Під час навчання нейромережа може виявляти складні залежності між вхідними та вихідними даними та виконувати узагальнення. Отже, в разі успішного навчання, нейромережа зможе повернути коректний результат на основі даних, яких не було в навчальній вибірці, а також неповних або частково спотворених даних [8].

Глибоке навчання – це сукупність методів машинного навчання, заснованих на імітації роботи людського мозку в процесі обробки даних і

створення патернів, що використовуються для прийняття рішень. Глибоке навчання призначене для роботи з великими обсягами даних і використовує складні алгоритми для навчання моделі. На великих датасетах глибоке навчання показує більш високу точність результатів в порівнянні з традиційним машинним навчанням. Залежність якості результатів від обсягу даних представлена на малюнку 2.1 [9].

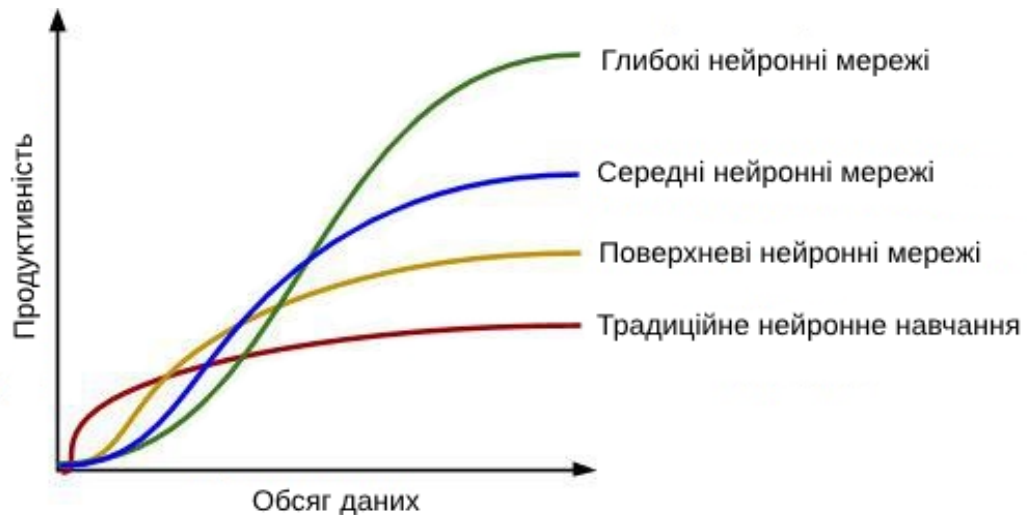


Рисунок 2.1 – Залежність якості результатів від обсягу даних [9]

На даний момент нейронні мережі і глибоке навчання забезпечують найкраще рішення багатьох завдань з областей розпізнавання зображень, голосу і обробки природної мови [10].

2.1.2 Архітектури нейронних мереж

Існує величезна різноманітність різних архітектур нейронних мереж, ми розглянемо декілька основних.

Перша класична архітектура – повнозв'язні нейронні мережі прямого поширення, або Fully Connected Feed-Forward Neural Network, FFNN (рис. 2.2). У повнозв'язній мережі кожен нейрон пов'язаний з усіма нейронами попереднього шару. FFNN зазвичай навчається за допомогою метода зворотного поширення

помилки, в якому неймережа отримує множини вхідних та вихідних даних. Даний спосіб називається навчанням з учителем, або контрольованим навчанням. Вищезгадана помилка є різницею між входом і виходом. Якщо мережа має достатню кількість прихованих нейронів, то вона спроможна змодельовати взаємодію між вхідними та вихідними даними. На практиці такі мережі використовуються нечасто, проте їх зазвичай комбінують з іншими типами для отримання нових архітектур [11].

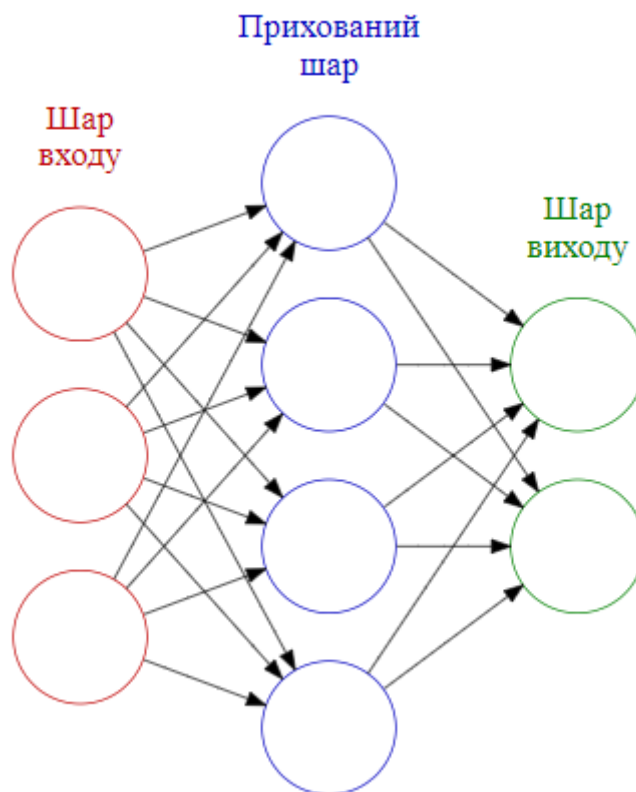


Рисунок 2.2 – Повнозв'язна нейронна мережа прямого поширення (персептрон) [11]

Автокодувальник (autoencoder, AE) чимось схожий на FFNN (рис. 2.3), бо це скоріше варіація FFNN, ніж фундаментально інша архітектура. Основною ідеєю даної архітектури є автоматичне кодування (в сенсі стиснення, а не шифрування) інформації. Сама неймережа за формою нагадує пісочний годинник, в ній приховані шари менше вхідного і вихідного шару, причому вона

симетрична. Ця архітектура дозволяє застосовувати навчання без вчителя при використанні методу зворотного поширення помилки [11].

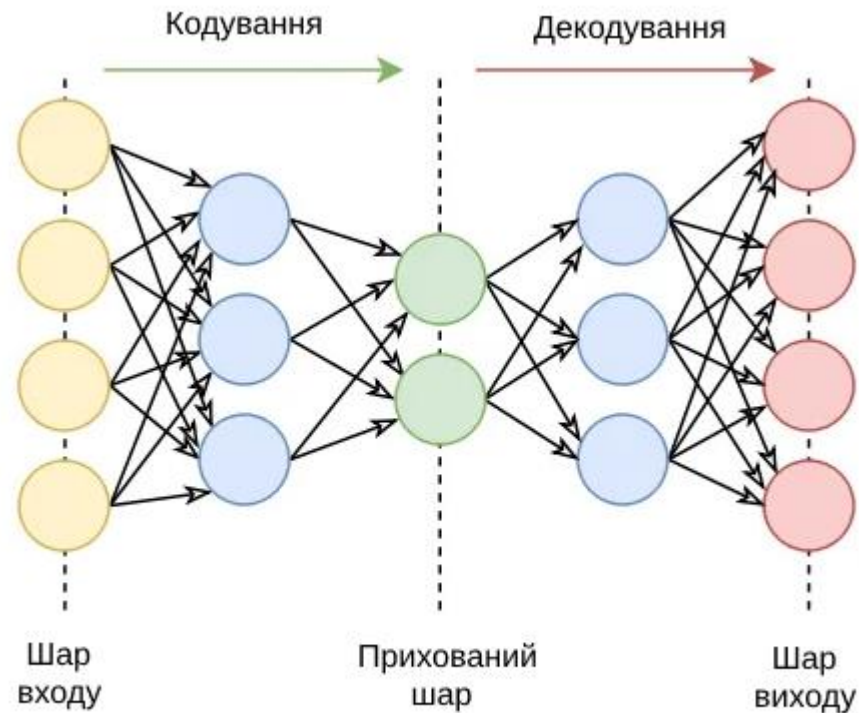


Рисунок 2.3 – Автокодувальник [11]

Рекурентні нейронні мережі (Recurrent neural network, RNN) – нейронні мережі, де зв'язки між елементами утворюють спрямовану послідовність (рис. 2.4). Завдяки цьому з'являється можливість обробляти серії подій у часі або послідовні просторові ланцюжки. На відміну від багатошарових перцептронів, рекурентні мережі можуть використовувати свою внутрішню пам'ять для обробки послідовностей довільної довжини. Тому мережі RNN можуть застосовуватись в тих задачах, де щось цілісне розбите на частини, наприклад: розпізнавання рукописного тексту або розпізнавання мови. Порядок надання об'єктів в такій мережі має важливе значення [11].

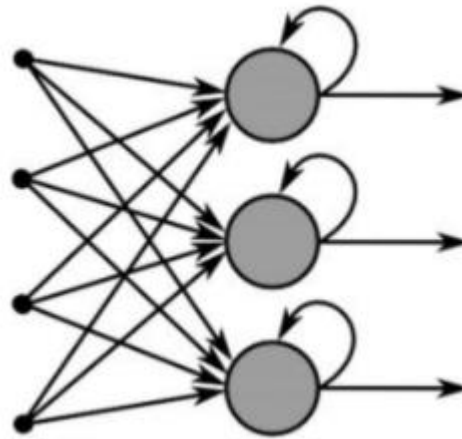


Рисунок 2.4 – Рекурентна нейронна мережа [11]

Згорткова нейронна мережа (convolutional neural network, CNN) – архітектура нейронних мереж, запропонована Яном Лекуном, від самого початку націлена на ефективне розпізнавання зображень (рис. 2.5) [12].

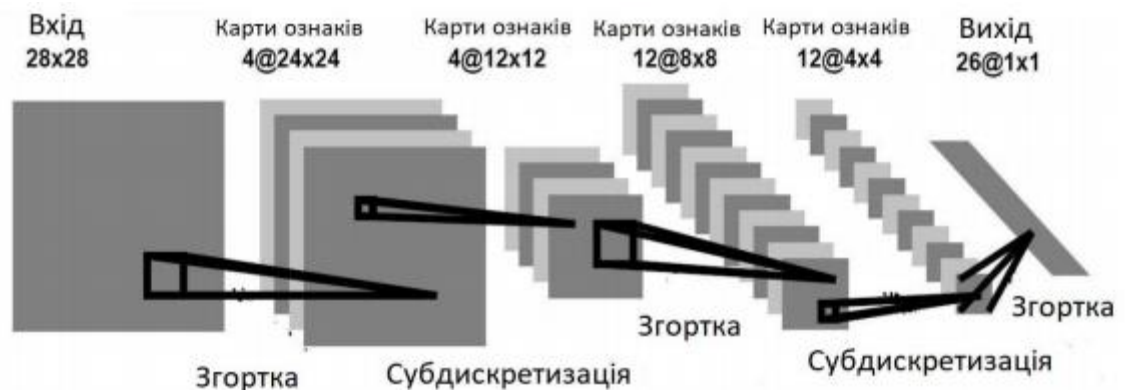


Рисунок 2.5 – Згорткова нейронна мережа [12]

CNN вдається набагато точніше розпізнавати об'єкти на зображеннях, бо, на відміну від багат шарового персептрона, враховується двовимірна топологія зображення. При цьому згорткові мережі стійкі до невеликих зсувів, змін масштабу і поворотів об'єктів на вхідних зображеннях. На даний момент CNN та її модифікації вважаються найкращими за точністю та швидкістю алгоритмами знаходження об'єктів на сцені. Починаючи з 2012 року, нейронні мережі

займають перші місця на відомому міжнародному конкурсі з розпізнавання образів ImageNet (рис 2.6) [13].

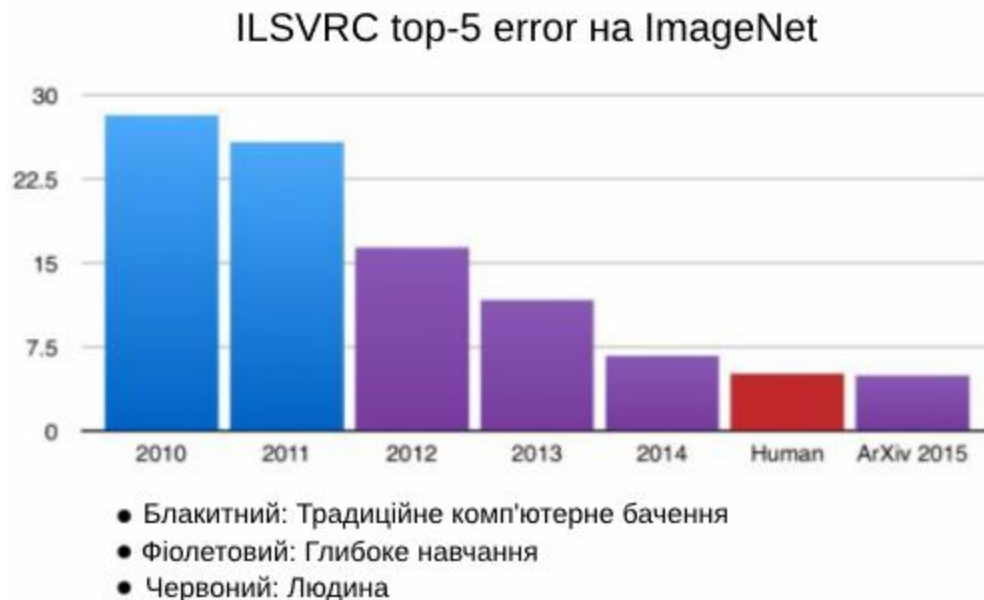


Рисунок 2.6 – Порівняння точності розпізнавання зображень нейронними мережами, традиційними методами та людиною [13]

Ідея згорткових нейронних мереж полягає в чергуванні згорткових шарів і агрегувальних шарів. Структура мережі – односпрямована (не має зворотних зв'язків), багатошарова. Для навчання використовуються стандартні методи, здебільшого метод зворотного поширення помилки. Функція активації нейронів може бути будь-якою (ReLU є однією з найбільш популярних) [12].

Назва архітектури нейронної мережі була обумовлена наявністю операції згортки. Зміст згортки полягає в наступному: кожен фрагмент зображення множиться на матрицю (ядро) згортки поелементно, а результат підсумовується та записується в аналогічному положенні вихідного зображення [12]. На малюнках 2.8 та 2.9 наведено приклад, що наочно представляє операцію згортки. Фільтр, відповідальний за пошук лівосторонніх кривих, переміщується вздовж вихідного зображення (рис. 2.7). Коли фрагмент, розглядається в даний момент, містить шукану криву, результатом згортки буде велике число (6600 в даному

випадку). Але коли фільтр переміщається на позицію, де немає лівосторонньої кривої, результатом згортки виступає невелике число (в даному випадку – 0) [14].

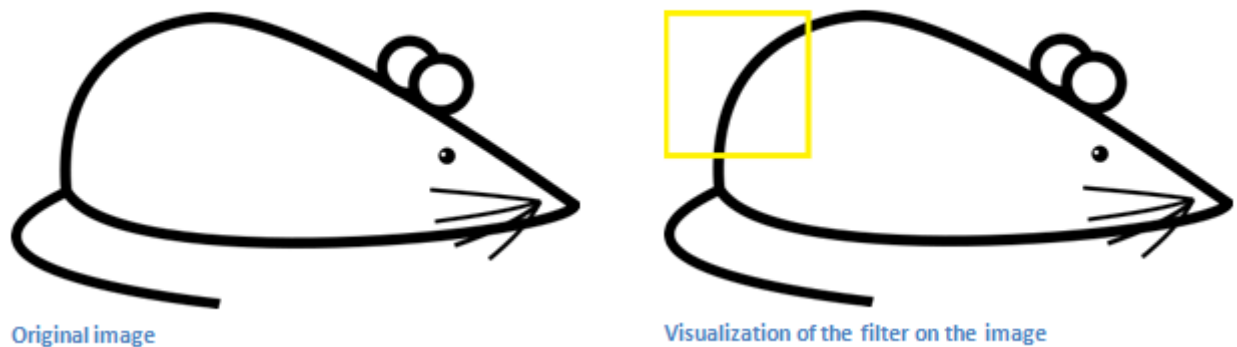


Рисунок 2.7 – Вихідне зображення [14]

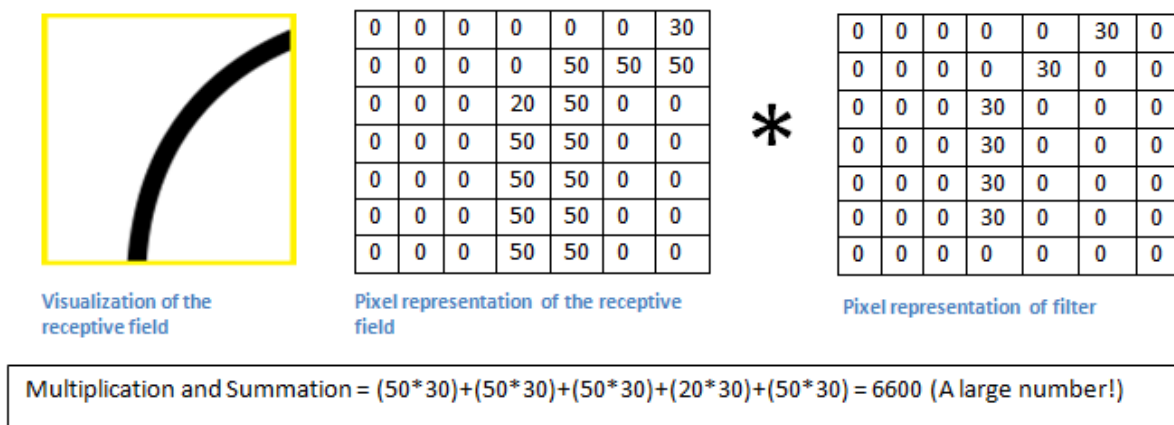


Рисунок 2.8 – Фільтр "шукає" лівосторонні криві; результат позитивний [14]

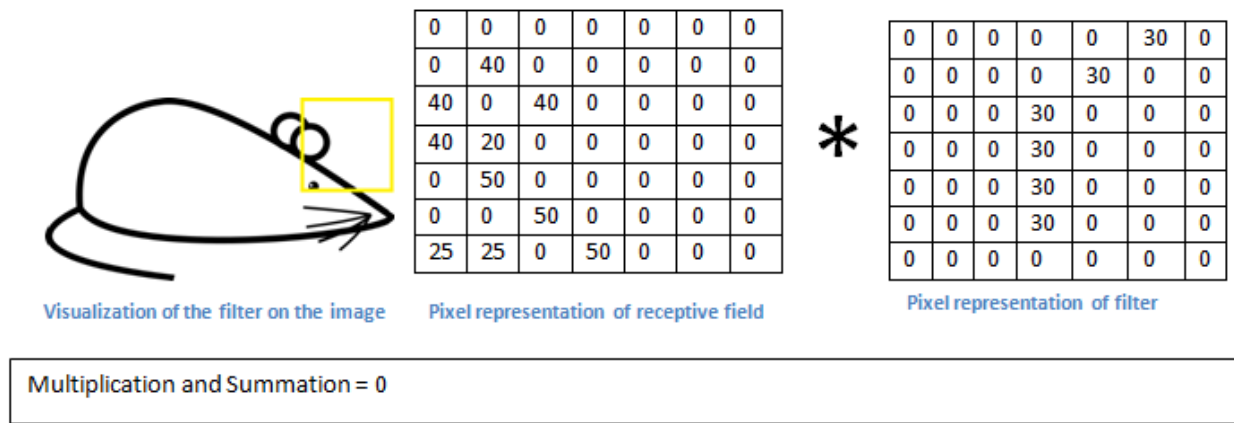


Рисунок 2.9 – Фільтр "шукає" лівосторонні криві; результат негативний [14]

Результатом переміщення даного фільтра уздовж всього зображення є карта ознак. У найпростішому випадку, при наявності одного фільтра згортки (і якщо цей фільтр – детектор кривої), карта ознак покаже області, в яких більше ймовірність наявності кривих. У цьому прикладі в лівому верхньому куті значення карти ознак буде дорівнювати 6600. Це велике значення вказує, що, можливо, щось схоже на лівосторонню криву присутнє на зображенні, і така ймовірність активувала фільтр. У правому верхньому куті значення у карти ознак буде дорівнювати 0, тому що на зображенні не було нічого, що могло активувати фільтр (простіше кажучи, в цій області не було кривої). Чим більше фільтрів, тим більше глибина карти ознак, і тим більше інформації ми маємо про вхідне зображення [14].

Фільтри першого згорткового шару виявляють властивості базового рівня, такі як межі та криві. Як можна собі уявити, щоб припустити який тип об'єкта зображений на малюнку, необхідна мережа, здатна розпізнавати властивості вищого рівня, як наприклад руки, лапи або вуха. Коли картинка проходить через один згортковий шар, вихід першого шару (одна або кілька карт ознак) стає вхідним значенням другого шару. При застосованні набору фільтрів поверх цього (при обробці зображення другим згортковим шаром), на виході будуть активовані фільтри, які представляють властивості вищого рівня. Чим

більше згорткових шарів проходить зображення і чим далі воно рухається по мережі, тим складніші характеристики виводяться в картах активації [14].

Загалом згорткова нейронна мережа складається з шарів входу і виходу та декількох прихованих шарів. Приховані шари CNN складаються із згорткових шарів, агрегувальних шарів, повноз'єднаних шарів та шарів нормалізації [12].

Згортковий шар нейронної мережі являє собою застосування операції згортки до виходів з попереднього шару, де вагові коефіцієнти ядра згортки є параметрами, що навчаються. Ще один ваговий коефіцієнт, що навчається, використовується в якості константного зсуву (bias). При цьому є кілька важливих деталей:

- один згортковий шар може мати декілька згорток. Тоді, для кожної згортки на виході вийде своє зображення. Наприклад, якщо вхід мав розмірність $w * h$, а в шарі було n згорток з ядром розмірності $k_x * k_y$, то вихід буде мати розмірність $n * (w - k_x + 1) * (h - k_y + 1)$;

- ядра згортки можуть бути тривимірними. Згортка тривимірного входу з тривимірним ядром відбувається аналогічно, але скалярний добуток розраховується також по всіх шарах зображення. Наприклад, для усереднення інформації про кольори початкового зображення, на першому шарі можна використовувати згортку розмірності $3 * w * h$. На виході цього шару буде тільки одне зображення (замість трьох) (рис. 2.10);

- застосування операції згортки зменшує зображення. Також пікселі, які знаходяться на межі зображення приймають участь в меншій кількості згорток, ніж ті, що знаходяться всередині. Через це в згорткових шарах використовується доповнення зображення (padding). Виходи з попереднього шару доповнюються пікселями таким чином, щоб після згортки зберігся розмір зображення. Такі згортки називають однаковими (same convolution), а згортки без доповнення зображення називаються правильними (valid convolution). Серед способів, якими можна заповнити нові пікселі, можна виділити наступні:

- zero shift: 00[ABC]00;
- border extension: AA[ABC]CC;

– mirror shift: BA[ABC]CB;

– cyclic shift: BC[ABC]AB.

– ще одним параметром згорткового шару є зсув (stride). Хоча зазвичай згортка застосовується для кожного пікселя поспіль, іноді використовується зсув, що відрізняється від одиниці – скалярний добуток розраховується не з усіма можливими положеннями ядра, а тільки з положеннями, кратними деякому зсуву s [15].

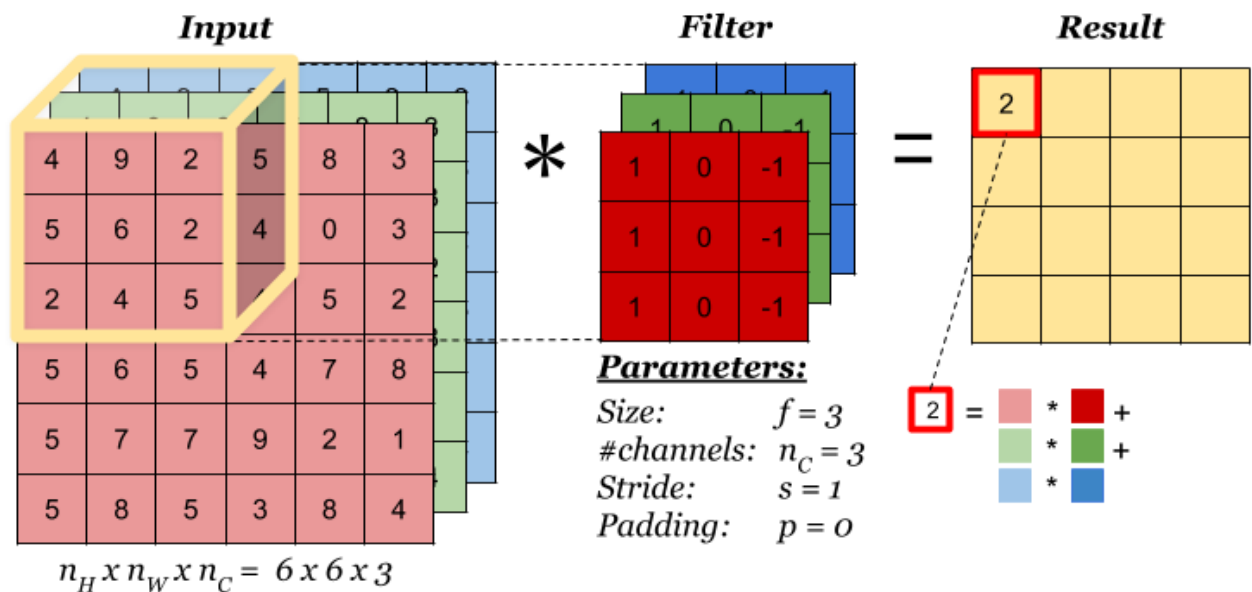


Рисунок 2.10 – Приклад згортки з тривимірним ядром [15]

Агрегувальний шар (pooling layer) покликаний знижувати розмірність зображення. Якщо деякі ознаки вже були знайдені на попередній операції згортки, то для подальшої обробки таке детальне зображення вже не потрібно, і воно ущільнюється до менш детального. Вихідне зображення ділиться на блоки розміром $w * h$ і для кожного блоку обчислюється деяка функція. Найчастіше використовується функція максимуму (max pooling) або (зваженого) середнього ((weighted) average pooling) (рис. 2.11). Параметрів, що навчаються, у цього шару немає. Основні цілі агрегувального шару:

– зменшення зображення, щоб наступні згортки могли оперувати над більшою областю вихідного зображення;

- збільшення інваріантності виходу нейромережі відносно малого переносу входу;
- прискорення обчислень [15].

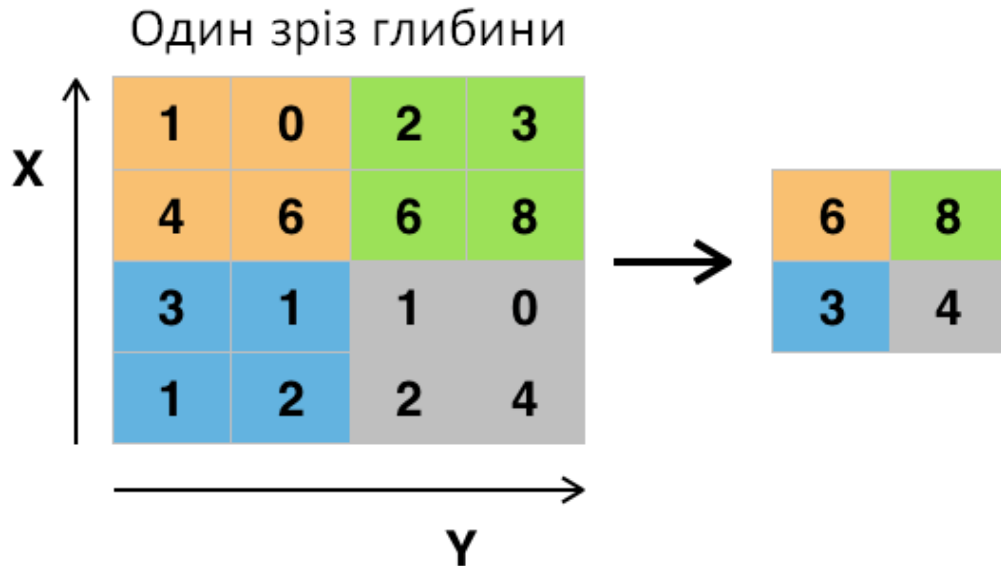


Рисунок 2.11 – Максимізаційне агрегування (max pooling) із фільтром 2×2 та кроком, що дорівнює 2 [15]

Після кількох згорткових та агрегувальних шарів, в нейронній мережі високорівневі міркування здійснюються за допомогою повноз'єднаних шарів. Повноз'єднаний шар бере вхідні дані і виводить N -мірний вектор, де N – кількість класів, з яких програма обирає потрібний. Наприклад, для розпізнавання цифр, у N буде значення 10, тому що цифр 10. Кожне число в цьому N -мірному векторі являє собою ймовірність певного класу. Тобто, якщо результуючий вектор для програми розпізнавання цифр це $[0, 0,1, 0,2, 0,65, 0, 0, 0, 0, 0,05]$, то існує 10% ймовірність, що на зображенні цифра "1", 20% ймовірність, що на зображенні цифра "2", 65% ймовірність – "3", і 5% ймовірність – "9" [14].

Спосіб, за допомогою якого працює повноз'єднаний шар – це звернення до виходу попереднього шару (який повинен виводити високорівневі карти властивостей) і визначення властивостей, які більше пов'язані з певним класом. Наприклад, якщо програма передбачає, що на якомусь зображенні собака, то у

карт ознак, які відображають високорівневі характеристики, такі як лапи, повинні бути високі значення [14].

2.2 Згорткові нейронні мережі

Сьогодні дуже багато завдань в області комп'ютерного зору вирішуються за допомогою згорткових нейронних мереж. CNN використовуються в задачах класифікації, розпізнавання, сегментації і ще в безлічі інших.

Нижче перераховані популярні архітектури CNN для розпізнавання об'єктів:

R-CNN. Одна з перших моделей для вирішення даної задачі. На вхід нейромережі подаються різні регіони зображення та для них робиться передбачення. Доволі повільна, бо опрацьовує одне зображення кілька тисяч разів [16].

Fast R-CNN. Покращена та швидша версія R-CNN, яка працює за схожим принципом, але спочатку все зображення подається на вхід згорткової нейронної мережі для отримання загальної карти ознак, а потім із отриманого внутрішнього уявлення генеруються регіони. Досить повільна для задач реального часу [16].

Faster R-CNN. На відміну від попередніх, замість алгоритму селективного пошуку використовує для вибору регіонів нейронну мережу для їх «заучування».

YOLO. Ця архітектура була першою спробою зробити можливою детекцію об'єктів в реальному часі. Має зовсім інший принцип роботи в порівнянні з попередніми, не використовує регіони взагалі [16]. Найбільш швидка (рис. 2.12) [17].

Method	Backbone	Size	FPS	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
YOLOv4: Optimal Speed and Accuracy of Object Detection									
YOLOv4	CSPDarknet-53	416	38 (M)	41.2%	62.8%	44.3%	20.4%	44.4%	56.0%
YOLOv4	CSPDarknet-53	512	31 (M)	43.0%	64.9%	46.5%	24.3%	46.1%	55.2%
YOLOv4	CSPDarknet-53	608	23 (M)	43.5%	65.7%	47.3%	26.7%	46.7%	53.3%
Learning Rich Features at High-Speed for Single-Shot Object Detection [84]									
LRF	VGG-16	300	76.9 (M)	32.0%	51.5%	33.8%	12.6%	34.9%	47.0%
LRF	ResNet-101	300	52.6 (M)	34.3%	54.1%	36.6%	13.2%	38.2%	50.7%
LRF	VGG-16	512	38.5 (M)	36.2%	56.6%	38.7%	19.0%	39.9%	48.8%
LRF	ResNet-101	512	31.3 (M)	37.3%	58.5%	39.7%	19.7%	42.8%	50.1%
Receptive Field Block Net for Accurate and Fast Object Detection [47]									
RFBNet	VGG-16	300	66.7 (M)	30.3%	49.3%	31.8%	11.8%	31.9%	45.9%
RFBNet	VGG-16	512	33.3 (M)	33.8%	54.2%	35.9%	16.2%	37.1%	47.4%
RFBNet-E	VGG-16	512	30.3 (M)	34.4%	55.7%	36.4%	17.6%	37.0%	47.6%
YOLOv3: An incremental improvement [63]									
YOLOv3	Darknet-53	320	45 (M)	28.2%	51.5%	29.7%	11.9%	30.6%	43.4%
YOLOv3	Darknet-53	416	35 (M)	31.0%	55.3%	32.3%	15.2%	33.2%	42.8%
YOLOv3	Darknet-53	608	20 (M)	33.0%	57.9%	34.4%	18.3%	35.4%	41.9%
YOLOv3-SPP	Darknet-53	608	20 (M)	36.2%	60.6%	38.2%	20.6%	37.4%	46.1%
SSD: Single shot multibox detector [50]									
SSD	VGG-16	300	43 (M)	25.1%	43.1%	25.8%	6.6%	25.9%	41.4%
SSD	VGG-16	512	22 (M)	28.8%	48.5%	30.3%	10.9%	31.8%	43.5%
Single-shot refinement neural network for object detection [95]									
RefineDet	VGG-16	320	38.7 (M)	29.4%	49.2%	31.3%	10.0%	32.0%	44.4%
RefineDet	VGG-16	512	22.3 (M)	33.0%	54.5%	35.5%	16.3%	36.3%	44.3%
M2det: A single-shot object detector based on multi-level feature pyramid network [98]									
M2det	VGG-16	320	33.4 (M)	33.5%	52.4%	35.6%	14.4%	37.6%	47.6%
M2det	ResNet-101	320	21.7 (M)	34.3%	53.5%	36.5%	14.8%	38.8%	47.9%
M2det	VGG-16	512	18 (M)	37.6%	56.6%	40.5%	18.4%	43.4%	51.2%
M2det	ResNet-101	512	15.8 (M)	38.8%	59.4%	41.7%	20.5%	43.9%	53.4%
M2det	VGG-16	800	11.8 (M)	41.0%	59.7%	45.0%	22.1%	46.5%	53.8%
Parallel Feature Pyramid Network for Object Detection [34]									
PFPNet-R	VGG-16	320	33 (M)	31.8%	52.9%	33.6%	12%	35.5%	46.1%
PFPNet-R	VGG-16	512	24 (M)	35.2%	57.6%	37.9%	18.7%	38.6%	45.9%
Focal Loss for Dense Object Detection [45]									
RetinaNet	ResNet-50	500	13.9 (M)	32.5%	50.9%	34.8%	13.9%	35.8%	46.7%
RetinaNet	ResNet-101	500	11.1 (M)	34.4%	53.1%	36.8%	14.7%	38.5%	49.1%
RetinaNet	ResNet-50	800	6.5 (M)	35.7%	55.0%	38.5%	18.9%	38.9%	46.3%
RetinaNet	ResNet-101	800	5.1 (M)	37.8%	57.5%	40.8%	20.2%	41.1%	49.2%
Feature Selective Anchor-Free Module for Single-Shot Object Detection [102]									
AB+FSAF	ResNet-101	800	5.6 (M)	40.9%	61.5%	44.0%	24.0%	44.2%	51.3%
AB+FSAF	ResNeXt-101	800	2.8 (M)	42.9%	63.8%	46.3%	26.6%	46.2%	52.7%
CornerNet: Detecting objects as paired keypoints [37]									
CornerNet	Hourglass	512	4.4 (M)	40.5%	57.8%	45.3%	20.8%	44.8%	56.7%

Рисунок 2.12 – Порівняння продуктивності на наборі даних COCO [17]

За допомогою алгоритму YOLO вихідне зображення спочатку розбивається на сітку з $N \times N$ комірок. Якщо центр об'єкта потрапляє всередину координат комірки, то ця комірка вважається відповідальною за визначення

параметрів місцезнаходження об'єкта. Кожна комірка описує декілька варіантів місця розташування обмежувальних рамок (bounding boxes) для одного і того ж об'єкта. Кожен з цих варіантів характеризується п'ятьма значеннями – координатами центру обмежувальних рамок, його шириною і висотою, а також ступенем впевненості в тому, що обмежувальна рамка містить в собі об'єкт. Також необхідно для кожної пари класу об'єктів і комірки визначити ймовірність того, що комірка містить в собі об'єкт цього класу. Таким чином, останній шар мережі, що приймає кінцеве рішення щодо обмежувальних рамок і класифікації об'єктів працює з тензором розмірності $N * N * (5B + C)$, де B – кількість обмежувальних рамок для комірки, що передбачаються, а C – кількість класів об'єктів, визначених спочатку (рис. 2.13). Перевага даного підходу полягає в тому, що мережа дивиться на все зображення відразу і враховує контекст при детектуванні і розпізнаванні об'єкта. Також YOLO в 1000 разів швидше ніж R-CNN і близько 100х швидше ніж Fast R-CNN [18].

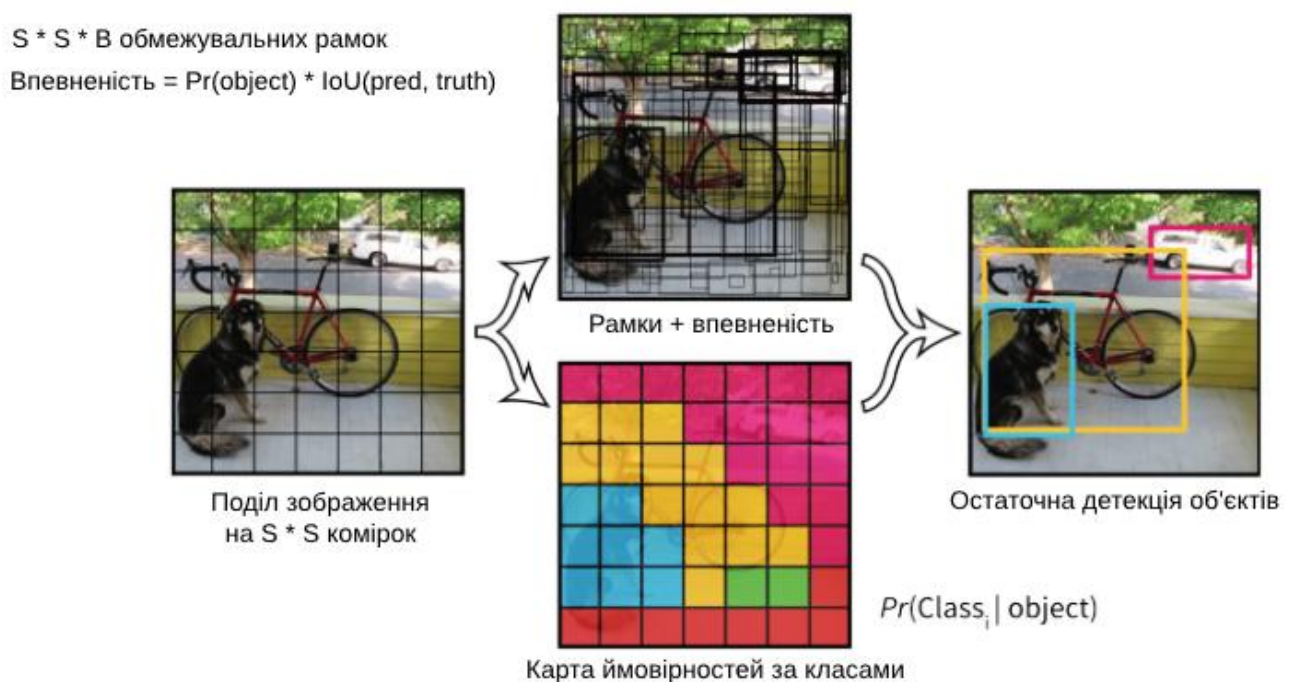


Рисунок 2.13 – Схема роботи алгоритму YOLO

SSD. За принципом роботи схожа на YOLO, але в якості мережі для виділення ознак використовує VGG16. Також є досить швидкою та придатною для роботи в режимі реального часу [16].

Feature Pyramid Networks (FPN). Ще один різновид мережі типу Single Shot Detector (SSD), що через особливості виділення ознак краще за SSD розпізнає невеликі об'єкти [16].

RetinaNet. Використовує комбінацію FPN + ResNet і завдяки спеціальній функції помилки (focal loss) має більш високу точність [16].

2.3 Модель YOLOv4

YOLOv4 – це вдосконалена версія архітектури YOLO. Основні модифікації даної моделі починаючи з другої до четвертої версії наведено нижче.

YOLOv2 відрізняється від попередньої версії використанням пакетної нормалізації (batch-normalization) на згорткових шарах, навчанням моделі на зображеннях з високою роздільною здатністю, використанням ключових рамок для передбачення місцезнаходження об'єктів, використанням кластеризації алгоритмом k-середніх для навчання більш ефективного вибору розмірів обмежувальних рамок на тренувальній вибірці з використанням функції відстані на основі IoU (2.1):

$$dist(x, c_i) = 1 - IoU(x, c_i), \quad (2.1)$$

де x – справжня обмежувальна рамка, c_i – центроїд кластера.

Кількість обмежувальних рамок-центроїдів обирається за допомогою "методу ліктя" (elbow method). Також в YOLOv2 використовується припущення, що обмежувальні рамки не дуже відхиляються від місця розташування центру, що забезпечує стабільність на тлі менш ефективного рівномірного вибору рамок-кандидатів по всьому вихідному зображенню. YOLOv2 використовує деревоподібну структуру класів, враховуючи їх вкладеність. Наприклад, якщо

серед класів є мітка "Персидська кішка", це буде означати, що знайдений об'єкт буде підкласом мітки "Кішка". Таким чином, не виникає взаємної винятковості класів, і softmax до всіх класів не застосовується. Щоб передбачити ймовірність вузла класу, ми можемо йти шляхом від вузла до кореня (2.2):

$$p(\text{persian cat}|\text{object}) = p(\text{persian cat}|\text{cat}) \cdot p(\text{cat}|\text{animal}) \cdot p(\text{animal}|\text{object}) \cdot p(\text{object}), \quad (2.2)$$

де $p(\text{object})$ – ймовірність виявлення об'єкта, обчислена на етапі генерації обмежувальних рамок. Шлях прогнозування умовної ймовірності може зупинитися на будь-якому етапі, в залежності від того, які мітки доступні [16].

YOLOv3, в свою чергу, є невеликим поліпшенням YOLOv2 – використовується логістична регресія для оцінок достовірності обмежувальних рамок замість суми квадратів помилок для умов класифікації в YOLO і YOLOv2. Також використовуються декілька незалежних логістичних класифікаторів для кожного класу замість одного шару softmax. Додано міжрівневі з'єднання між рівнями прогнозування обмежувальних рамок. А також використовуються архітектури DarkNet і ResNet для згорткових мереж [16].

YOLOv4 – це вдосконалення YOLOv3. Впровадження нової архітектури в Backbone (рис. 2.14, 2.15) та модифікацій в Neck покращили mAP на 10%, а кількість FPS на 12%. Крім того, стало легше навчати цю нейронну мережу на одному графічному процесорі [17].

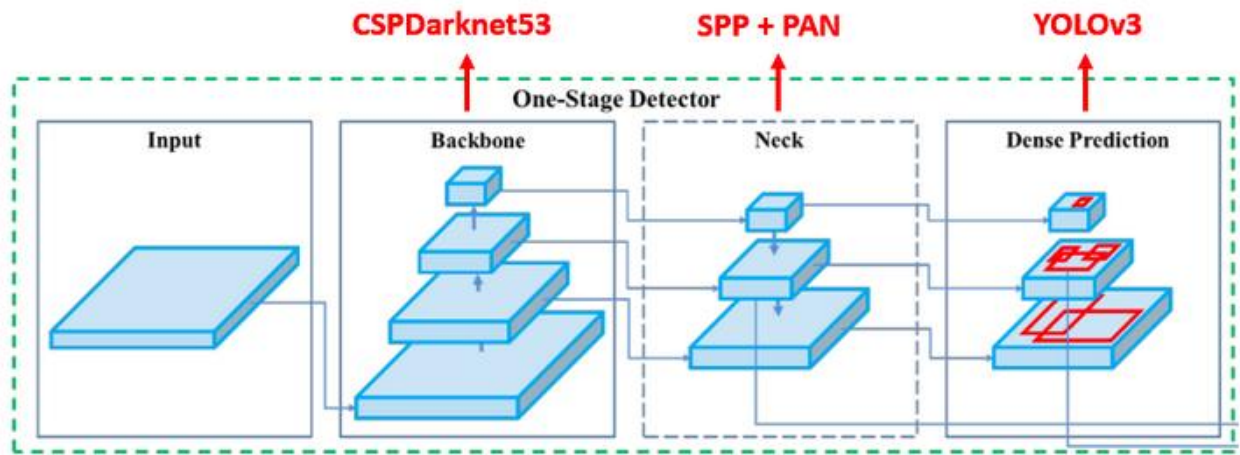


Рисунок 2.14 – Архітектура YOLOv4 [17]

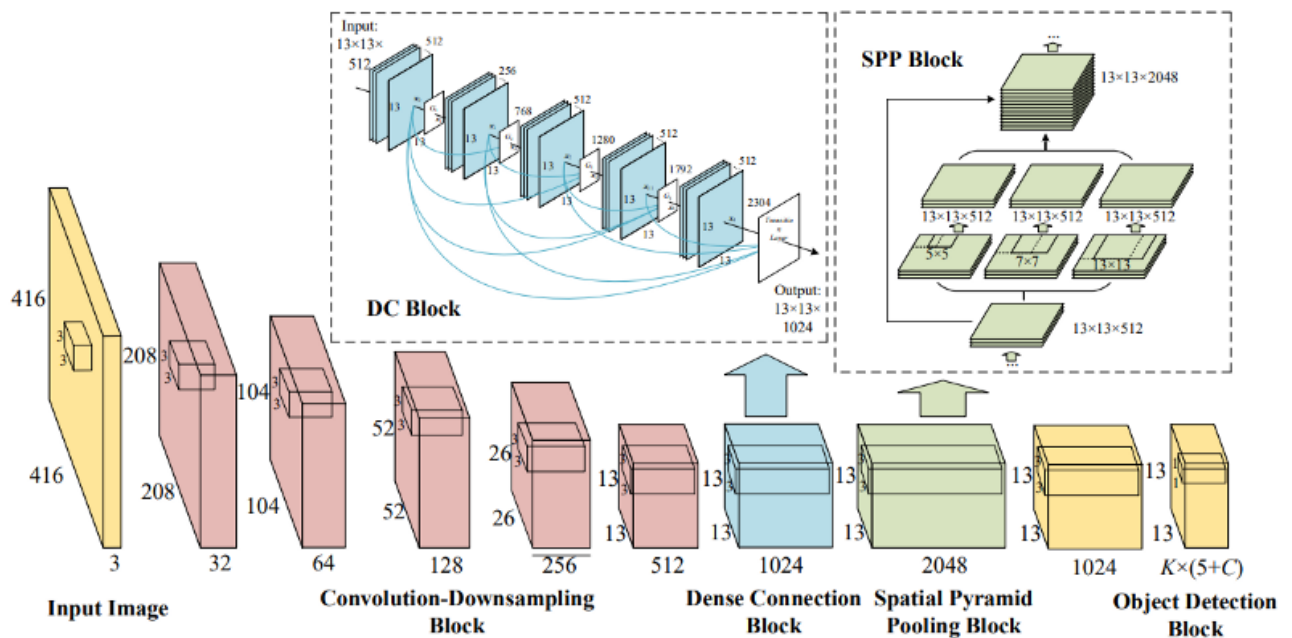


Рисунок 2.15 – Структура YOLOv4 [17]

Для згорткових мереж в Backbone застосовано архітектуру CSPDarknet53 замість Darknet53. YOLOv4 використовує в Neck PANet для покращення процесу сегментації шляхом збереження просторової інформації. В якості функції активації використовується Mish, яка може бути визначена як $f(x) = x \tanh(\text{softplus}(x))$. Для більш точного прогнозування обмежувальних рамок застосовується CIoU loss. Також замість пакетної нормалізації використовується крос-міні-пакетна нормалізація (Cross mini Batch Normalization) [17].

2.4 Метод обробки відеокадру

Згорткова неймережа забезпечує часткову стійкість до змін масштабу, зсувів, поворотів та інших спотворень зображення [19]. Оскільки використовується відеокамера з кутом огляду 360° , то зображення отримані за допомогою об'єктива “fish-eye” (рис. 2.16) мають характерні спотворення, за виглядом схожі на відображення в дзеркальній сфері [20]. Це може бути причиною погіршення точності детектування людей на відеокадрах згортковою неймережею. Тому, перш ніж передати відеокادر до неймережі, необхідно його привести до вигляду сферичної проекції (рис. 2.17) для зменшення спотворень. Оскільки, відеопотік — це часова послідовність кадрів, то далі оброблюватиметься саме кадр. Алгоритм конвертації зображення отриманого за допомогою об'єктива “fish-eye” до вигляду сферичної проекції наведено нижче.



Рисунок 2.16 – Зображення, отримане за допомогою об'єктива “fish-eye”



Рисунок 2.17 – Зображення, приведені до вигляду сферичної проекції

Проекція "fish-eye" – один із багатьох способів відображення точок у 3D на площину. У контексті зображення, знятого камерою з об'єктивом "fish-eye", ключовими параметрами є положення камери, орієнтація та поле зору об'єктива "fish-eye". Хоча найчастіше поле зору "fish-eye" становить 180 градусів, деякі фізичні лінзи "fish-eye" розширюються до 250 градусів [21].

Умовні позначення, що використовуються для зображення "fish-eye" для отримання математичних розрахунків, наведено на рис. 2.18. Коло "fish-eye" лежить у площині (x, z) , розташованій на початку координат, а межі зображення нормалізовані так, щоб знаходились від -1 до 1. Камера знаходиться на початку координат і направлена донизу на позитивну вісь y у [21].

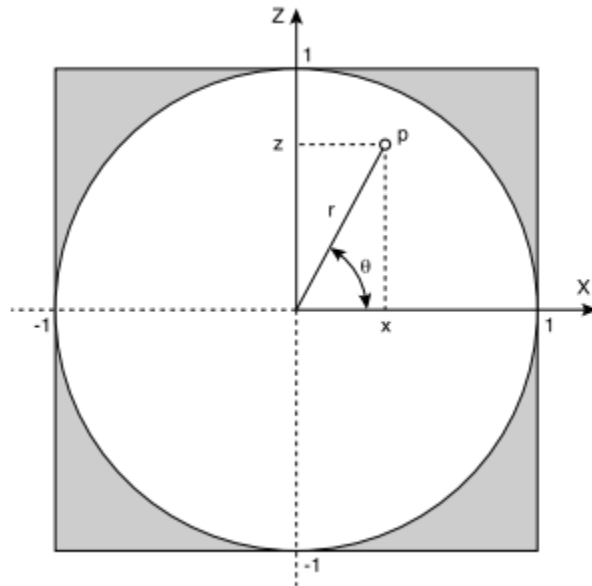


Рисунок 2.18 – Схема зображення "fish-eye" [21]

Будь-яку 2D точку p на цьому зображенні "fish-eye" можна параметризувати в полярних координатах (r, θ) . Ці r, θ задаються наступним чином (2.3), (2.4) [21]:

$$r = \sqrt{x^2 + z^2} \quad (2.3)$$

$$\theta = \text{atan2}(z, x) \quad (2.4)$$

Визначимо додатковий кут ϕ за формулою (2.5):

$$\phi = \frac{\phi_{\max} r}{2}, \quad (2.5)$$

де $\phi_{\max}$ – поле зору лінзи "fish-eye". Саме лінійна залежність між радіусом r на зображенні "fish-eye" та ϕ є ключовим відображенням для проєкції "fish-eye" (рис 2.19) [21].

Відповідний тривимірний вектор P і значення ϕ та θ проілюстровано нижче, а також умовні позначки камери. Деякі особливі випадки:

– коли $r = 0$, це відповідає центру зображення "fish-eye", а $\phi = 0$ відповідає тривимірному вектору вздовж оптичної осі лінзи "fish-eye".

–коли $r = 1$, це відповідає положенням по краю кола "fish-eye". Це відповідає тривимірним векторам під кутами в половину максимального поля зору.

– використовуючи домовленості, представлені тут, θ є однаковим значенням у 2D-зображенні та 3D-векторі [21].

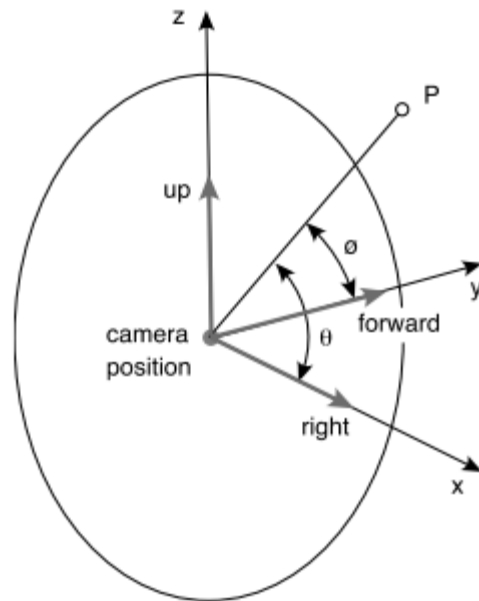


Рисунок 2.19 –Розташування камери на зображенні "fish-eye" [21]

При відображенні зображення "fish-eye" в рівнопрямокутну проекцію потрібно розрахувати визначену довготу та широту, тут використовуються домовленості, як показано нижче (рис. 2.20) [21].

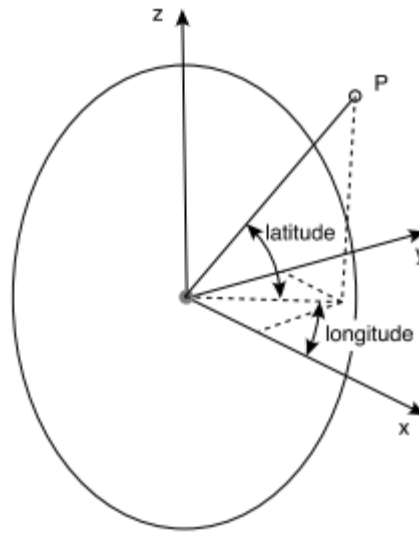


Рисунок 2.20 – Довгота та широта на зображенні "fish-eye" [21]

Таким чином (2.6), (2.7),

$$longitude = atan2(P_y, P_x), \quad (2.6)$$

$$latitude = atan2(P_z, \sqrt{P_x^2 + P_y^2}) \quad (2.7)$$

Точне визначення довготи може варіюватися залежно від того, де потрібно розмістити довготу "0". Якщо 0 градусів довготи слід розташовувати в центрі горизонтально рівнопрямокутного зображення (рис. 2.21), тоді кут довготи в наведеному вище прикладі повинен бути між віссю у та проекцією на площину x-y [21].

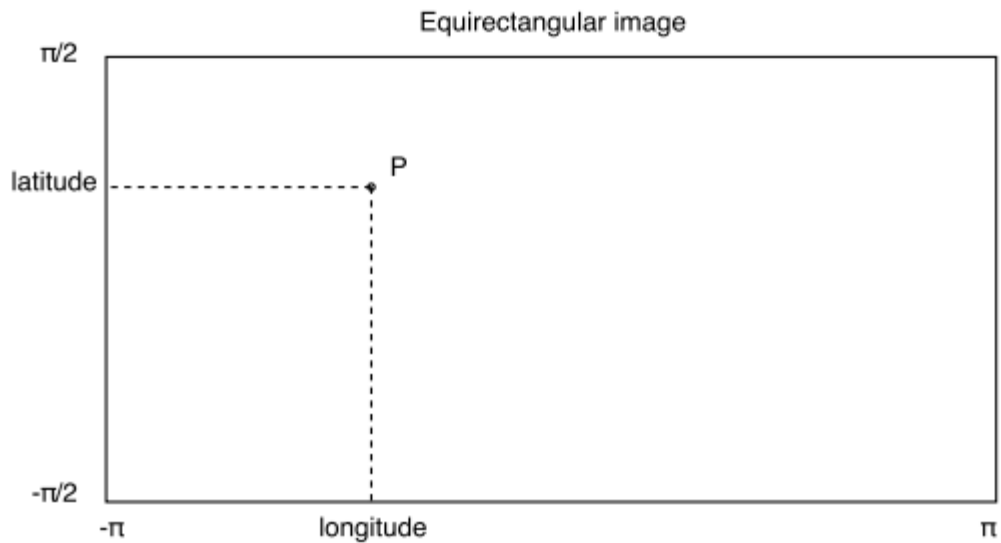


Рисунок 2.21 – Рівнопрямокутне зображення (сферична проекція) [21]

2.5 Система керування УФ лампою

Для керування УФ лампою робота-дезінфектора потрібно спочатку отримати зображення від панорамної камери. Отриманий кадр оброблюється наступним чином: ділиться на дві частини, кожна з частин кадру приводиться до вигляду сферичної проєкції, потім частини склеюються. Це допомагає зменшити спотворення, що збільшує ймовірність детектування об'єктів. Далі оброблений кадр передається до нейронної мережі YOLOv4. Нейронна мережа визначає, чи є на відеокадрі об'єкти, що відносяться до класу "person". Потім, якщо нейронна мережа детектувала на відеокадрі хоча б один об'єкт, що відносяться до класу "person", то система вмикає лампу та припиняє дезінфекцію. У випадку, коли людей на зображенні не знайдено система вмикає лампу, якщо вона була вимкнена, та продовжує дезінфекцію (рис. 2.22).

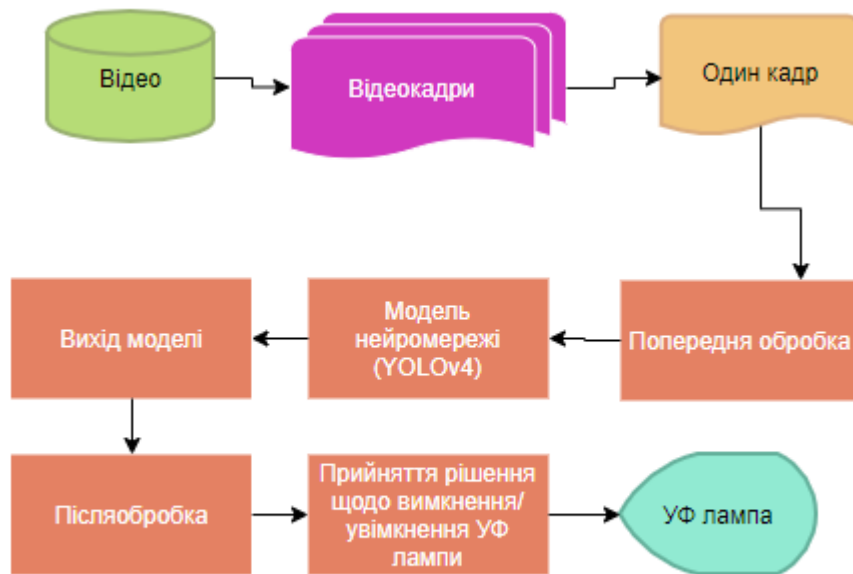


Рисунок 2.22 – Структура системи

Схема роботи системи керування УФ лампою робота-дезінфектора приведена на рис. 2.23 та рис. 2.24.

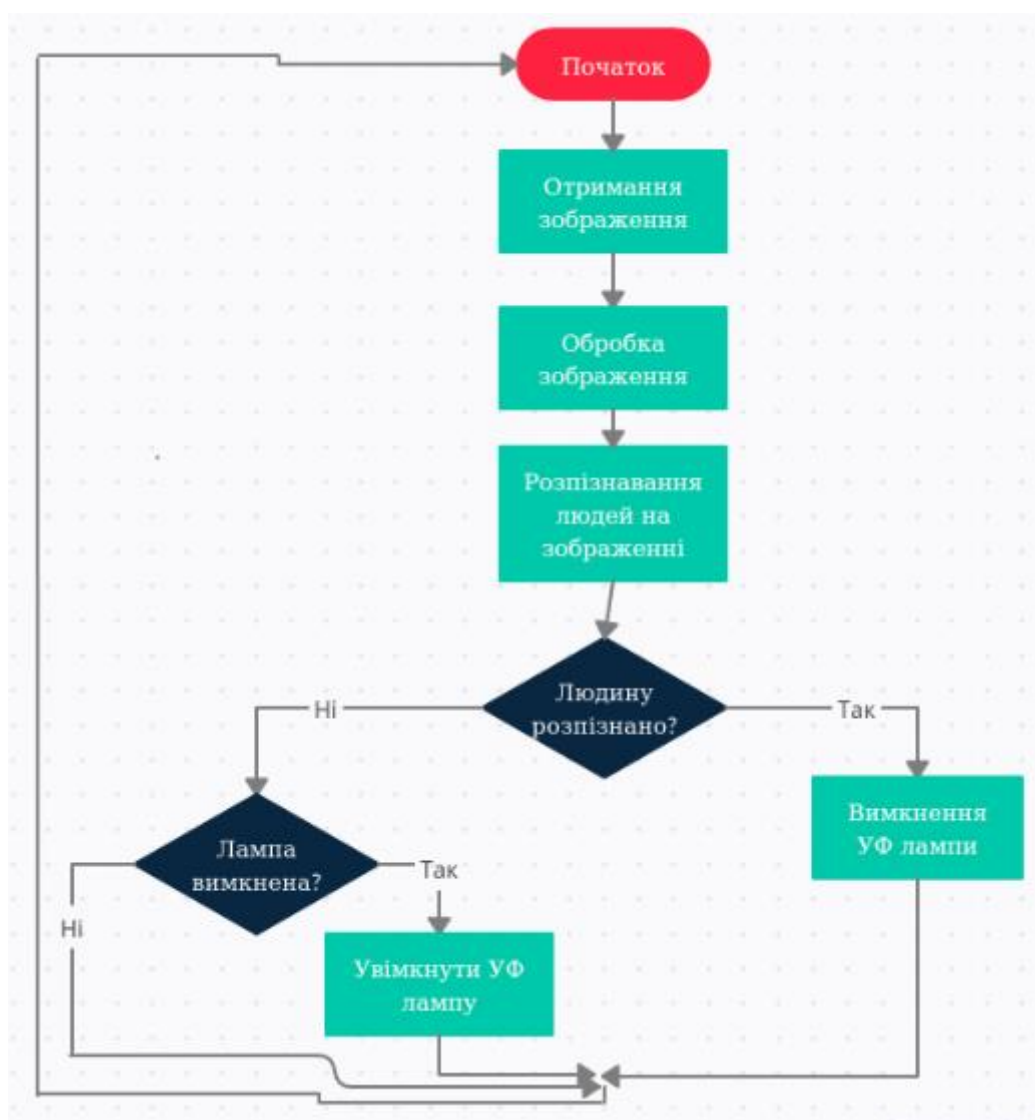


Рисунок 2.23 – Схема роботи системи керування УФ лампою робота

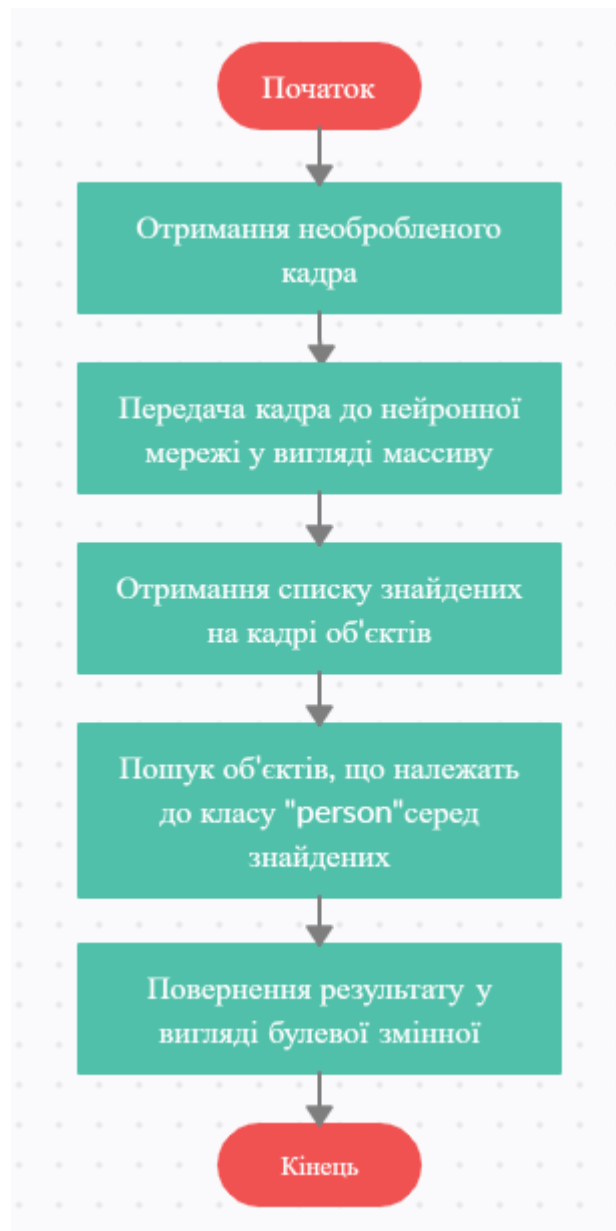


Рисунок 2.24 – Детальна схема блоку “Розпізнавання людей на зображенні”

2.6 Висновки за розділом 2

Проаналізовано нейромережеві методи, які можуть застосовуватись для розпізнавання людей на відеокadraх. Проаналізовано метод обробки відеокadру, що застосовується для приведення кадру до вигляду сферичної проекції. Розроблено систему для керування УФ лампою робота-дезінфектора. Робота системи базується на розпізнаванні людей на відеокadрах за допомогою нейронної мережі YOLOv4, які попередньо було конвертовано до вигляду сферичної проекції.

ОПИС ПРОГРАМИ

3.1 Визначення вимог до програмного забезпечення

Розроблюване програмне забезпечення має виконувати наступні функції:

- оброблювати відеокадри;
- розпізнавати людей на відеокадрі;
- вмикати УФ лампу;
- вимикати УФ лампу.

На рис. 3.1 приведено діаграму прецедентів.



Рисунок 3.1 – Діаграма прецедентів

3.2 Вибір мови програмування

У процесі вибору мови програмування для реалізації програмного забезпечення порівнювались мови програмування Python, C++ та Java, які можуть використовуватись для створення застосунків для системи ROS.

Основні особливості мови програмування Python перераховано нижче:

- мову програмування Python було створено у 1991 році голландцем Гвідо ван Россумом [22];
- має такі архітектурні риси, як: динамічна типізація та автоматичне керування пам'яттю, механізм обробки виключень, підтримка багатопоточних обчислень, повна інтроспекція та високорівневі структури даних [22];
- інтерпретована мова програмування. Це означає, що вихідний код перетворюється в машинний частинами в процесі його читання спеціальною програмою – інтерпретатором [23];
- має лаконічний та зрозумілий синтаксис. Читати код на Python легше, ніж на інших мовах програмування, бо в ній мало використовуються такі допоміжні синтаксичні елементи як дужки та крапки з комою. З іншого боку, правила Python примушують програміста робити відступи для позначення вкладених конструкцій, а добре оформлений текст із невеликою кількістю відволікаючих елементів легше читати та розуміти [23];
- універсальна мова програмування, що може використовуватись в різних сферах. Основна підтримувана мовою парадигма – об'єктно-орієнтоване програмування (також підтримуються інші парадигми) [23];
- інтерпретатор Python поширюється вільно на підставі ліцензії подібної GNU General Public License [23];

У свою чергу мова програмування C++ має наступні особливості:

- мова програмування була розроблена Бйорном Страуструпом на початку 80-х років XX століття як ряд вдосконалень для мови C [24];
- компільована мова програмування загального призначення із статичною типізацією. Вона може використовуватись для створення найрізноманітніших

застосунків. На сьогодні C++ є однією з найбільш популярних та поширених мов програмування [25];

- потужна мова, що успадкувала від C багато можливостей роботи з пам'яттю. Саме тому C++ доволі часто застосовується в системному програмуванні. Сфера її застосування включає в себе створення операційних систем, драйверів пристроїв, різних утиліт, антивірусів та додатків для вбудованих систем. Проте системним програмуванням використання даної мови програмування не обмежено. C++ використовується в будь-яких програмах, де продуктивність та швидкість роботи мають важливе значення. Нерідко вона застосовується для розробки прикладних програм та графічних додатків [25];

- кросплатформна: стандарт мови програмування накладає мінімальні вимоги на ЕОМ для запуску скомпільованих програм. У стандартній бібліотеці є функції, що визначають реальні властивості системи виконання (наприклад, `std::numeric_limits`). Існують компілятори для великої кількості платформ, тому на C++ розробляють програми для найрізноманітніших платформ і систем [25];

- має високу сумісність з мовою C, що дозволяє використовувати весь існуючий код, написаний на C [25];

- у порівнянні із своєю попередницею, мовою C, підтримує об'єктно-орієнтовану парадигму програмування, що дозволяє представляти програму як сукупність класів та об'єктів, що взаємодіють між собою. Це робить більш простим створення великих програмних додатків [25].

Основні особливості мови програмування Java наведено нижче:

- мова Java була розроблена компанією Sun Microsystems на початку 90-х років XX століття [26];

- надійна: мова Java строго типізована, тобто будь-яка змінна або вираз має певний тип вже на момент компіляції, що спрощує виявлення будь-яких проблем. Компілятор сам підказує програмісту, де той припускається помилки, і не дає її здійснити [26];

- не залежить від платформи, бо код, написаний на Java, спочатку транслюється в байт-код, який сумісний з різними платформами (Linux,

Windows, Mac OS та іншими). А вже потім отриманий байт-код виконується віртуальною машиною Java Virtual Machine (JVM). У зв'язку з цим Java відрізняється від інтерпретованих мов програмування, код яких одразу ж виконується інтерпретатором. Водночас, Java не є суто компільованою мовою, як, наприклад, C або C++ [27];

- відносно проста: мова має доволі простий синтаксис, який легко вивчити [27];

- підтримує автоматичне збирання сміття. Це означає, що програмісту не потрібно звільняти вручну пам'ять від об'єктів, що раніше використовувалися, як в C++, тому, що збирач сміття (garbage collector) це зробить автоматично [27];

- об'єктно-орієнтована мова. Об'єктно-орієнтований підхід дозволяє вирішувати завдання з побудови великих, але разом з тим масштабованих, гнучких та розширюваних додатків [27].

Описані мови порівнювались за основними критеріями, за якими має прийматися рішення щодо вибору. Результати зведено до табл. 3.1.

Таблиця 3.1 – Порівняння мов програмування

Критерій	Python	C++	Java
Об'єктно-орієнтована парадигма програмування	+	+	+
Безкоштовне рішення	+	+	+
Багатопотоковість	+	+	+
Робота з обраними інструментами (OpenCV, YOLOv4, Darknet)	+	+	+/-

Продовження табл. 3.1

Критерій	Python	C++	Java
Більш висока швидкість виконання	—	+	-

Для розробки основних вузлів системи – обробки кадру та детектування людей на ньому – обрано мову C++, оскільки на відміну від мови Python, вона є компільованою і забезпечує більш високу швидкість виконання [28], що важливо для систем реального часу. Мова Java не дозволяє працювати з деякими з обраних інструментів, а також має нижчу швидкість виконання, ніж C++. За всіма іншими критеріями описані мови достатньо співмірні. Тому, для вузла, який відповідає за керування УФ лампою було обрано мову Python.

3.3 Вибір фреймворку

В якості фреймворку для розробки програмного забезпечення роботів було обрано ROS, бо цей фреймворк забезпечує:

- повторне використання програмних модулів. Розроблений програмний модуль, легко запускається та повторно використовується у будь-якому іншому додатку. Питання встановлення залежностей та інших бібліотек добре досліджені та автоматизовані;

- готовий протокол комунікації. Основна проблема комплексних робототехнічних систем, це вирішення задач комунікації в рамках однієї програми. Для вирішення цих завдань ROS містить всі необхідні утиліти. Будь-який програмний модуль може бути представлений як окремий процес, що взаємодіє з іншими процесами за допомогою мережевого протоколу. Такий підхід дозволяє створювати незалежні і прості у повторному використанні програмні модулі, які можливо запустити/зупинити/модифікувати на будь-якому пристрої;

– розвиненість засобів розробки і налагодження програм. ROS надає готові інструменти для налагодження, інструмент 2D-візуалізації (rqt) та інструмент 3D-візуалізації (RViz), а також інструмент 3D симуляції (Gazebo);

– активна і відкрита спільнота. Спільнота розробників робототехніки з академічного світу і промисловості, була відносно закрита до останнього часу. Але зараз ми бачимо активну, а головне відкриту співпрацю всіх учасників. У центрі цієї зміни – програмна платформа з відкритим вихідним кодом. У випадку з ROS існує більше 5000 пакетів, які були розроблені і викладені в загальний доступ. Опис цих пакетів, інструкцій та іншої корисної інформації – перевищує 18 000 Wiki сторінок;

– власна екосистема. Навколо ROS сформована власна екосистема (подібна до екосистем платформ Android та Apple). В ній існують розробники апаратних платформ, розробники програмних модулів, ентузіасти і компанії виробники промислового устаткування, єдине місце поширення та зберігання готових модулів, тисячі сторінок документації. Всі учасники взаємодіють і працюють в рамках єдиної платформи [29].

3.3.1 Особливості ROS

Фреймворк ROS – це набір утиліт, бібліотек, угод, який спрощує розробку складного програмного забезпечення для різних робототехнічних платформ. ROS працює поверх класичної операційної системи, забезпечуючи служби мета-операційної системи робота: апаратну абстракцію різних пристроїв робота, низькорівневий контроль цих пристроїв, реалізацію часто використовуваних функцій та завдань, передачу повідомлень між процесами, і управління пакетами [30].

Як і інші ОС, ROS складається з двох частин: безпосередньо ядра фреймворку `ros`, і `ros-pkg`, набору підтримуваних спільнотою пакетів, які реалізують різні функції робототехніки: позиціонування, планування, сприйняття, моделювання та інше [30].

ROS випускається відповідно до умов BSD-ліцензії і є відкритим вихідним кодом. Крім того, ROS безкоштовний для використання не тільки в дослідницьких, а й в комерційних цілях. Пакети з `ros-pkg` поширюються на умовах різних відкритих ліцензій і автори самі вирішують яку ліцензію використовувати [30].

Основними концепціями ROS є вузли (nodes), повідомлення (messages), теми (topics) та сервіси (services). Архітектурно всі обчислювальні задачі виконуються у вузлах ROS, які обмінюються між собою інформацією за допомогою повідомлень. Ці повідомлення вузлами публікуються в темах, які поділяють ці повідомлення на групи інтересів. Коли деякому вузлу необхідно отримувати повідомлення з певними даними, цей вузол підписується на певну тему (рис. 3.2). Для реалізації синхронної передачі повідомлень, яка необхідна в певних випадках, ROS визначає сервіси – механізм, який працює за принципом питання-відповідь [31].

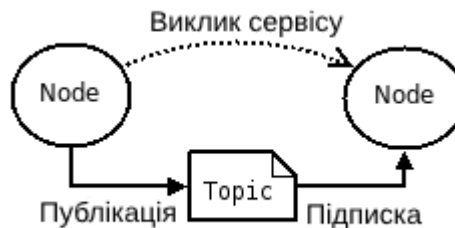


Рисунок 3.2 – Схема обміну повідомленнями між вузлами

3.4 Вибір бібліотеки для обробки відеопотоку

В якості бібліотеки для обробки відеокadrів було обрано OpenCV, оскільки розроблювана система працює в режимі реального часу, а продуктивність бібліотеки OpenCV перевищує продуктивність інших бібліотек комп'ютерного зору та обробки зображень (VXL, LTI) [32].

OpenCV – це високооптимізована бібліотека, орієнтована на додатки реального часу, алгоритмів обробки зображень, комп'ютерного зору та чисельних алгоритмів загального призначення з відкритим вихідним кодом [33].

Бібліотеку було розроблено компанією Intel та випущено у 2006 році. OpenCV поширюється під ліцензією BSD. Бібліотеку можна вільно використовувати як в академічних, так і в комерційних цілях. На сьогодні OpenCV широко використовується дослідницькими групами, компаніями та державними органами [33].

Вихідний код OpenCV написано мовою C++. Бібліотека також розроблюється для Python, Matlab, Ruby, Java, Lua та інших мов програмування. OpenCV підходить для роботи на Linux, Windows, Mac OS та Android [33].

У бібліотеці налічується понад дві з половиною тисячі оптимізованих алгоритмів, серед яких є як класичні, так і сучасні алгоритми машинного навчання та комп'ютерного зору. Даний набір алгоритмів може використовуватись для виявлення та розпізнавання облич, ідентифікації об'єктів, відстеження переміщень камери, відстежування об'єктів, що рухаються, класифікації дій людини на відео, отримання тривимірних моделей об'єктів, створення тривимірних хмар точок зі стереокамер, зшивання зображень разом для отримання зображення всієї сцени з високою роздільною здатністю, знаходження схожих зображень з баз даних зображень, видалення ефекту червоних очей з фотографій, зроблених за допомогою спалаху, слідкування за рухами очей та іншого [33].

Основні модулі OpenCV:

- `opencv_core` – відповідає за математичні обчислення;
- `opencv_imgproc` – забезпечує обробку зображень;
- `opencv_video` – відповідає за аналіз руху та відстеження об'єктів;
- `opencv_gpu` – забезпечує прискорення функцій бібліотеки за рахунок CUDA [33].

3.5 Висновки за розділом 3

Визначено функції, які повинна виконувати програмна система. Створено діаграму прецедентів для моделювання функціональних вимог до роботи системи.

Проаналізовано, порівняно та обрано основну мову програмування C++ для реалізації програми. В якості фреймворку для розробки програмного забезпечення роботів обрано ROS. Бібліотекою для обробки відеокадрів обрано OpenCV.

4 РОЗРОБКА ПРОГРАМИ

4.1 Опис реалізації структури програмного забезпечення

Систему керування УФ лампою робота-дезінфектора реалізовано на базі ROS, тому в основі структури програмного забезпечення знаходиться розподілена модульна архітектура.

Програмний проект складається з наступних модулів:

- camera_360 – вузол, що відповідає за обробку зображення;
- human_detection – вузол, що відповідає за детектування людей на зображенні;
- lamp_relay – вузол, що відповідає за вмикання/вимикання УФ лампи.

На рис. 4.1 приведена діаграма розгортання.

Всі вузли працюють на встановленому на роботі-дезінфекторі мікрокомп'ютері Jetson Nano.

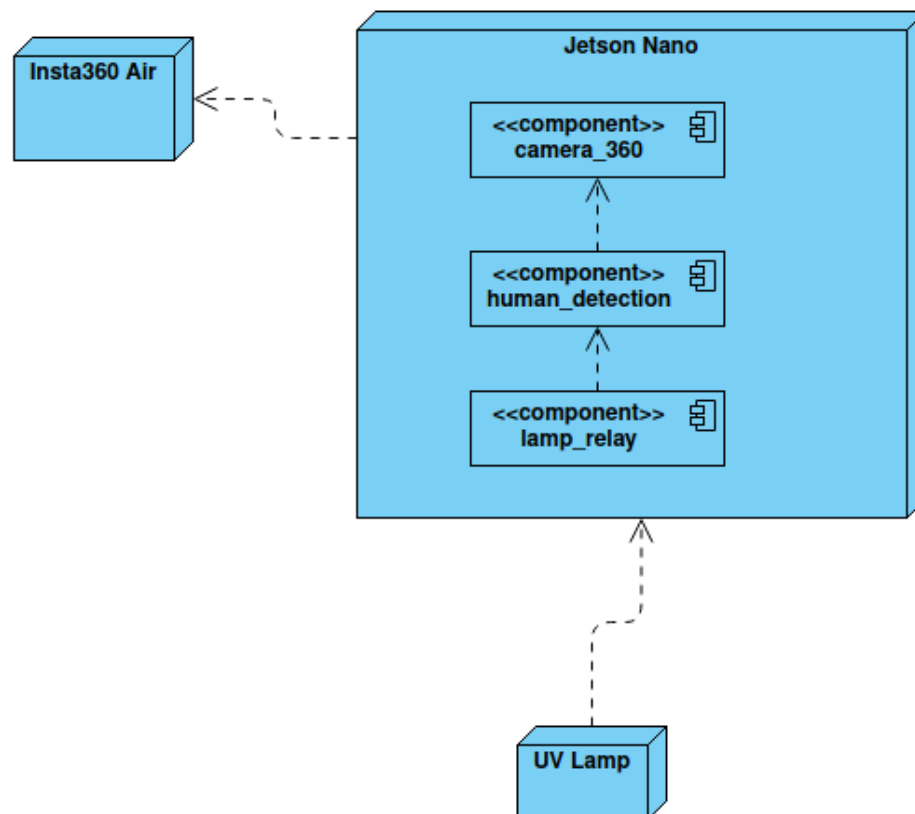


Рисунок 4.1 – Діаграма розгортання

До складу модулю camera_360 входить клас VideoCapturer, який відповідає за захоплення відеокадру і його обробку, а також декілька допоміжних класів.

До складу модулю human_detection входить клас PtzHumanFollower, який відповідає за детектування людини, а також декілька допоміжних класів.

4.2 Опис розроблених модулів

Для реалізації взаємодії між окремими вузлами системи використано модель Публікація-підписка (Publish–subscribe). Класи `ros::Publisher` і `ros::Subscriber` дозволяють публікувати повідомлення у певну тему (topic) та підписуватись на отримання нових повідомлень за певною темою відповідно. Для цього в класі VideoCapturer створюється екземпляр класу `ros::Publisher`, за допомогою якого публікується повідомлення з обробленим відеокадром. А в класі PtzHumanFollower створюються екземпляри класів `ros::Subscriber` та `ros::Publisher`, за допомогою яких отримується повідомлення з вже обробленим кадром та публікується повідомлення з інформацією щодо знаходження людей на ньому відповідно. Нижче наведено підписку та публікацію повідомлення в класі PtzHumanFollower:

```
ros::Publisher pub_hd;
ros::Subscriber sub_cam_360;
ros::NodeHandle nh;
//підписка
sub_cam_360 = nh.subscribe("camera_360/image_360", 1,
&PtzHumanFollower::cameraCallback, this);
//публікація
pub_hd.publish(msg_output);
```

На рис. 4.2 приведено схему взаємодії між вузлами системи. Для цього спочатку вузол camera_360 захоплює камеру, отримує необроблений відеокادر, обробляє його та опубліковує повідомлення з обробленим зображенням в темі camera_360/image, далі вузол human_detection отримує повідомлення з теми camera_360/image, в якому знаходиться оброблений кадр, встановлює чи є в кадрі людина та опубліковує повідомлення з даною інформацією в темі person_detection. Потім вузол lamp_relay отримує повідомлення з теми person_detection і якщо людину було детектовано, то лампа вмикається. Даний цикл продовжується доти, доки система працює.

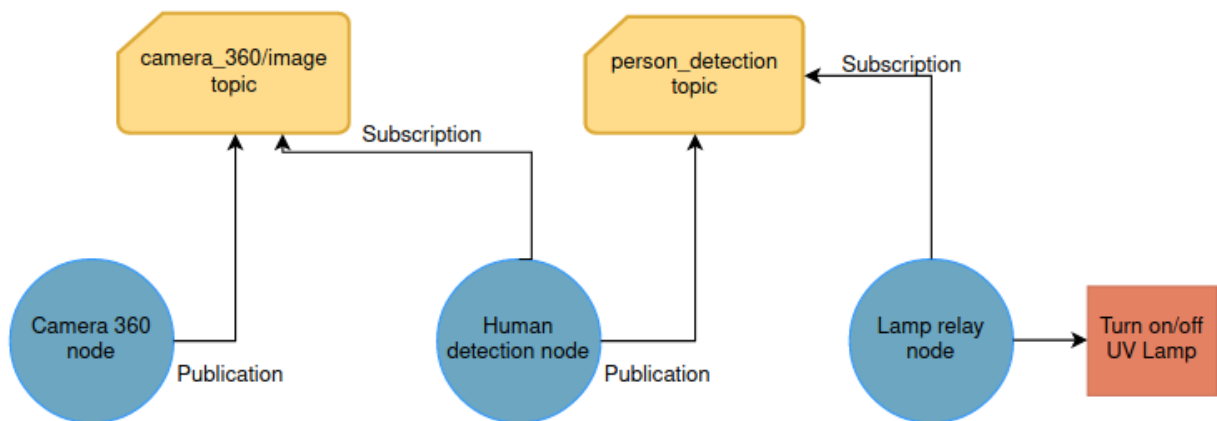


Рисунок 4.2 – Схема взаємодії між вузлами

Клас VideoCapturer реалізує логіку захвату та обробки зображення та включає наступні атрибути:

- cv::Rect _leftRect – ліва частина необробленого кадру;
- cv::Rect _rightRect – права частина необробленого кадру;
- cv::Size – розмір кадру;
- int _frameWidth – ширина кадру;
- int _frameHeight – висота кадру;
- cv::Mat _map_x – матриця значень x для обробки зображення;

- `cv::Mat _map_u` – матриця значень `u` для обробки зображення;
- `cv::VideoCapture videoCapture` – механізм для захвату відео з камери;
- `cv::Mat frame` – оброблений кадр;
- `cv::Mat capturedFrame` – необроблений кадр;
- `ros::Publisher cam_pub` – публікатор, за допомогою якого повідомлення з обробленими кадрами публікуються в темі;

Реалізовано наступні методи класу:

- `void VideoCapturer::init()` – метод, який ініціалізує поля об'єкту `VideoCapturer`;
- `void VideoCapturer::releaseCam()` – метод, який звільняє камеру;
- `void VideoCapturer::loop()` – метод, який запускає нескінченний цикл;
- `void VideoCapturer::setRects()` – метод, який ділить зображення на ліву і праву частини;
- `void VideoCapturer::setCurrentFrame()` – метод, який склеює ліву та праву частини кадру;
- `void VideoCapturer::onFrameReceive()` – метод, який публікує повідомлення з обробленим кадром;
- `VideoCapturer::VideoCapturer(int index, ros::NodeHandle *nh, QObject *parent)` – конструктор класу.

Клас `PtzHumanFollower` реалізує логіку детектування людини на зображенні та включає наступні атрибути:

- `cv::Mat frame` – оброблений кадр;
- `std::vector<bbox_t> boxes` – детектовані на зображенні об'єкти;
- `bool personIsDetected{false}` – флаг, що визначає знайдено людей на зображенні (`true`) або не знайдено (`false`);
- `std_msgs::Bool msg_output` – повідомлення, що буде відправлено до теми;
- `ros::Publisher pub_hd` – публікатор, за допомогою якого повідомлення з інформацією щодо знаходження людей на зображенні публікуються в темі;

- `ros::Subscriber sub_cam_360` – підписник, за допомогою якого виконується підписка на отримання повідомлень з обробленим кадром з теми;
- `YoloDetector* yoloDetector` – YOLOv4-детектор;

Реалізовано наступні методи класу:

- `PtzHumanFollower::PtzHumanFollower()` – конструктор класу;
- `void PtzHumanFollower::cameraCallback(const std_msgs::UInt8MultiArray& msg)` – метод, за допомогою якого детектуються люди на зображенні та відправляється повідомлення у відповідну тему.

4.3 Висновки за розділом 4

Реалізовано програмне забезпечення керування УФ лампою робота-дезінфектора, описано розгортання програмного забезпечення, описано схему взаємодії між вузлами системи та принципи реалізації взаємодії між ними на основі моделі Публікація-підписка, описано основні реалізовані класи.

5 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для захисту людей від УФ опромінення, які знаходяться у приміщенні під час дезінфекції. Керування УФ лампою робота-дезінфектора реалізується автоматичним чином.

5.2 Умови виконання програми

Для роботи системи мають забезпечуватися наступні мінімальні вимоги:

- графічний процесор з 128 ядрами Cuda з розрахунковою продуктивністю в 472 GFLOPS;
- процесор з 4 ядрами з частотою 2,5 ГГц;
- обсяг оперативного запам'ятовуючого пристрою – 4 Гб;
- флеш-пам'ять – 16 Гб;
- пропускна здатність мережного інтерфейсу – 100 Мбіт/с.

5.3 Виконання програми

Програмне забезпечення складається з трьох вузлів: вузла, що відповідає за захват та обробку зображення, вузла, що відповідає за детектування людей на зображенні та вузла, який керує реле УФ лампи та вмикає або вимикає її.

Для того щоб розпочати роботу системи необхідно увімкнути робота, система почне працювати автоматично і буде працювати доти, доки робота не буде вимкнено.

Якщо система детектує людей у приміщенні, яке дезінфікується, то система вимкне УФ лампу (рис. 5.1).



Рисунок 5.1 – УФ лампа робота-дезінфектора вимкнена

Якщо система більше не розпізнає людей у приміщенні, яке дезінфікується, то система знов увімкне УФ лампу (рис. 5.2).



Рисунок 5.2 – УФ лампа робота-дезінфектора увімкнена

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

У роботі виконується дослідження та програмна реалізація системи управління роботом-дезінфектором на основі аналізу відеопотоку.

Розробка програмного додатку має призвести до створення системи, в межах якої можна буде керувати УФ лампою робота-дезінфектора. Користувачами будуть виступати інші системи робота-дезінфектора.

Заплановано здійснити продаж розроблюваної програмної системи за 200000 грн.

6.1 Планування розробки програмного виробу

Весь комплекс розробки програмного забезпечення можна поділити на етапи. Для кожного етапу вказується трудомісткість, кількість виконавців та тривалість робіт [34]. У розробці приймають участь програміст впродовж 2.5 місяців і консультант впродовж 0.5 місяця. Розробка програми починається 12 вересня 2020 року та має бути завершена до 26 листопада 2020 року.

Отримана інформація зведена в табл. 6.1, за даними якої складається зведений стрічковий графік планування розробки програмного продукту, який представляє собою таблицю, в першому стовпці якої розміщені в порядку збільшення термінів початку виконання всі види роботи, а навпроти – календарний період їх виконання.

Тривалість робіт розраховується за формулою (6.1):

$$T_{ц} = \frac{Q}{R}, \quad (6.1)$$

де $T_{ц}$ – довжина циклу, днів;

Q – трудомісткість, людино-днів;

R – кількість виконавців, люд.

Таблиця 6.1 – Характеристика робіт по розробці програми

Найменування роботи	Трудомісткість		Виконавці		Трива - лість, днів
	люди- но- днів	% до підсумк у	спеціаль- ність	кіль- кість	
1. Дослідження вимог до проектованого продукту	10	14.7	програміст	2	5
			консультант		
2. Вибір інструментів проектування, дослідження існуючих рішень	4	5.8	програміст	1	4
3. Проектування загальної структури програми	10	14.7	програміст	2	5
			консультант		
4. Розробка схеми функціонування програми	2	2.9	програміст	1	2
5. Створення програмного коду	24	35.3	програміст	1	24
6. Тестування та налагодження програмного продукту	14	20.6	програміст	1	14
7. Складання програмної документації	4	5.9	програміст	2	2
			консультант		
Всього	68	100.0			56

Необхідні початкові дані:

– час на розробку – 2.5 місяці;

– консультант працює – до 0.5 місяців (12 днів) із заробітною платою 10000 грн./місяць;

6.2 Визначення витрат на розробку програми

Для визначення витрат на розробку програми складається калькуляція кошторисної вартості робіт, яка включає наступні статті:

- основна заробітна плата;
- додаткова заробітна плата;
- відрахування на єдиний соціальний внесок;
- витрати на спеціальне обладнання;
- матеріали і комплектуючі вироби;
- накладні витрати;
- податки.

6.2.1 Розрахунок основної заробітної плати

Витрати по цій статті складаються з планового фонду заробітної плати усіх категорій працівників, зайнятих в розробці програмного продукту. Розрахунок заробітної плати ведеться на основі відомостей про трудомісткість (табл. 6.1). Результати розрахунків зведені в табл. 6.3.

Таблиця 6.3 – Розрахунок основної заробітної плати

Посада виконавця	Кіль- кість, осіб	Місячний оклад	Кіль- кість, міс. роботи	Премії та доплати, грн.	Середня місячна зарплата, грн.	Сумарна зарплата, грн.
Програміст	1	10000	2.5	—	10000	25000
Консультант	1	10000	0,5	—	10000	5000
Всього	2					30000

Розрахунок заробітної плати проводиться за формулою (6.2):

$$ЗП_{\text{осн}} = \sum_1^i (ЗП_{\text{мес}_i} \cdot T_{p_i}), \quad (6.2)$$

де $ЗП_{\text{мес}_i}$ – щомісячна заробітна плата i -того працівника, грн.;

T_{p_i} – час, витрачений на розробку програми i -тим працівником, міс. [34].

6.2.2 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати [34]. Розрахунок основної та додаткової заробітної плати наведений в табл. 6.4.

Таблиця 6.4 – Розрахунок додаткової заробітної плати

Посада виконавця	Додаткова заробітна плата, грн.	Сума основної та додаткової заробітної плати, грн.
Програміст	2500	27500
Консультант	500	5500
Всього	3000	30000

6.2.3 Відрахування на єдиний соціальний внесок

Відрахування на єдиний соціальний внесок визначають у відсотковому відношенні від суми основної і додаткової зарплат, з урахуванням премій і доплат. Премії і доплати дорівнюють 0. Відрахування на соціальне страхування складають 22% [34].

Розрахунок виконується за формулою (6.3):

$$ВІД = (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0.22, \quad (6.3)$$

де $ЗП_{\text{осн}}$ – основна заробітна плата, грн.;

$ЗП_{\text{дод}}$ – додаткова заробітна плата, грн. [34].

Отже, відрахування на єдиний соціальний внесок дорівнюють:

$$(30000 + 3000) \cdot 0,22 = 7260 \text{ грн.}$$

6.2.4 Визначення витрат на матеріали

Матеріали, які знадобилися для розробки даної програми, представлені в табл. 6.5.

Таблиця 6.5 – Витратні матеріали

Матеріал	Кількість	Вартість, грн.
Папір формату А4	1 пачка	60
Олівці	5 шт.	5
Тонер для принтера	1 шт.	160

Витрати на матеріали розраховуються за формулою (6.4):

$$M = \sum_{i=1}^n (C_i \cdot N_i \cdot (1 + K_{\text{т.з}}) - C_{i0} \cdot N_{i0}), \quad (6.4)$$

де M – витрати на матеріали, покупні напівфабрикати і комплектуючі вироби, грн.;

$K_{\text{т.з.}}$ – коефіцієнт, що враховує транспортно-заготівельні витрати (0,03 – 0,1);

C_i – ціна i -го найменування матеріалу, напівфабрикату, комплектуючого, грн.;

N_i – потреба в i -му матеріалі, напівфабрикаті, комплектуючому;

C_{i0} – ціна поворотних відходів i -го найменування матеріалу, грн.;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів, комплектуючих [34].

Розрахуємо витрати на матеріали:

$$M = (1+0,1) \cdot 60 + 5 \cdot (1+0,1) \cdot 5 + (1+0,1) \cdot 160 = 269,5 \text{ грн.}$$

Разом витрати на матеріали склали 269,5 грн.

6.2.5 Витрати на спеціальне устаткування

У цю статтю входять витрати на оплату машинного часу ЕОМ. Витрати на оплату машинного часу ЕОМ для написання і відладки цієї програми визначають по формулі (6.5):

$$C_{mo} = P_{екс} \cdot t_{mo}, \quad (6.5)$$

де C_{mo} – витрати на оплату машинного часу, грн.;

$P_{екс}$ – експлуатаційні витрати на одну годину машинного часу, грн.;

t_{mo} – машинний час ЕОМ для написання і відладки цього програмного продукту, год. [34].

Для розрахунку експлуатаційних витрат на одну годину машинного часу складемо кошторис витрат на зміст і експлуатацію цього устаткування.

Амортизаційні відрахування визначають по формулі (6.6):

$$A = \Phi_6 \frac{H_a}{100}, \quad (6.6)$$

де Φ_6 – балансова вартість обчислювальної техніки, грн.;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, % [34].

Балансова вартість обчислювальної техніки визначається як ціна обчислювальної техніки, помножена на коефіцієнт, що враховує транспортно-заготівельні витрати і монтаж обладнання ($K=0,05$).

Балансова вартість обчислювальної техніки становить $28000 \cdot (1 + 0,05) = 29400$ грн., а норма амортизаційних відрахувань – 25%.

Таким чином амортизаційні відрахування становлять:

$$A = 29400 \cdot 0,25 = 7350 \text{ грн.}$$

Стаття "Експлуатація обладнання" розраховується підсумовуванням витрат на силову електроенергію і допоміжні матеріали.

Витрати на електроенергію розраховуються за формулою (6.7):

$$C_{\text{э}} = N_n \cdot \Phi_{\text{эф}} \cdot K_{\text{зв}} \cdot K_{\text{зм}} \cdot C_{\text{э}}, \quad (6.7)$$

де N_n – номінальна потужність ЕОМ, кВт;

$\Phi_{\text{эф}}$ – річний ефективний фонд часу роботи ЕОМ, год.;

$K_{\text{зв}}$ – середній коефіцієнт завантаження за часом;

$K_{\text{зм}}$ – коефіцієнт завантаження за потужністю;

$C_{\text{э}}$ – ціна однієї кВт·год. електроенергії, грн.

Номінальна потужність ЕОМ = 0,3 кВт. $\Phi_{\text{эф}} = 1800$ год. Середній коефіцієнт завантаження за часом – 0,45. Коефіцієнт завантаження за потужністю – 0,7. Ціна однієї кВт·год електроенергії – 2,11 грн.

Таким чином:

$$C_{\text{э}} = 0,33 \cdot 1800 \cdot 0,45 \cdot 0,7 \cdot 2,11 = 722,17 \text{ грн.}$$

Зарплата обслуговуючого персоналу розраховується за формулою (6.8):

$$ЗП_{\text{обсл}} = ФЗП_{\text{р}} \cdot (1 + K_{\text{відр}}) \cdot \frac{t_{\text{обсл}}}{\Phi_{\text{еф.обсл}}}, \quad (6.8)$$

де $\Phi ЗП_r$ – річний фонд зарплати обслуговуючих працівників, грн.;

$K_{\text{відр}}$ – коефіцієнт відрахувань на соціальне страхування і до інших фондів, %;

$t_{\text{обсл}}$ – час впродовж року, необхідний на техобслуговування ЕОМ, год./рік;

$\Phi_{\text{еф.обсл}}$ – річний ефективний фонд часу обслуговуючого персоналу, год./рік [34].

Машину обслуговує один адміністратор з основною заробітною платою 5000 грн./міс. Річний фонд заробітної плати становить відповідно 60000 грн. На одну ЕОМ адміністратор витрачає по 1 годині на місяць, що в рік становить 12 годин. Час, необхідний на технічне обслуговування ЕОМ протягом року, дорівнює 1750 год.

Таким чином:

$$ЗП_{\text{обсл}} = 60000 \cdot (1 + 0,22) \cdot 12 / 1750 = 501,9 \text{ грн.}$$

Сума витрат по статті "Поточний ремонт устаткування" дорівнює 3% від балансової вартості устаткування, що дорівнює $29400 \cdot 0,03 = 882$ грн.

Інші витрати складають 4% від суми попередніх:

$$(7350 + 722,17 + 501,9 + 882) \cdot 0,04 = 378,2 \text{ грн.}$$

Зведемо отримані дані щодо кошторису витрат на утримання та експлуатацію устаткування в табл. 6.6.

Експлуатаційні витрати на одну годину машинного періоду складають за формулою (6.9):

$$P_{\text{екс}} = S_{\text{ктс}} / \Phi_{\text{эф.ктс}} = 9834,27 / 1800 = 5,46 \text{ грн./год.} \quad (6.9)$$

Розробка і відладка програмного продукту займає 304 години машинного часу (38 днів по 8 робочих годин), тобто $t_{mo} = 304$ год. Витрати на оплату машинного часу склали:

$$C_{mo} = 5,46 \cdot 304 = 1659,84 \text{ грн.}$$

Таблиця 6.6 – Кошторис витрат на утримання та експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація устаткування	7350
Експлуатація устаткування	722,17
Зарплата обслуговуючого персоналу з відрахуваннями	501,9
Поточний ремонт устаткування	882
Інші витрати	378,2
Всього	9834,27

6.2.6 Накладні витрати

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і так далі), зміст і експлуатацію будинків. Накладні витрати включаються у вартість розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників. В даному випадку накладні витрати складають 20 % до основної заробітної плати розробників – 6000 грн.

Результати калькуляції кошторисної вартості робіт по розробці програми і ціна програмного продукту наведені у табл. 6.7.

Таблиця 6.7 – Калькуляція кошторисної вартості робіт по розробці програми і ціна програмного продукту

№	Найменування статей витрат	Сума, грн.	Питома вага до підсумку, %
1	Основна заробітна плата	30000	62,2
2	Додаткова заробітна плата	3000	6,22
3	Соціальні відрахування	7260	15,06
4	Витрати на матеріали	269,5	0,55
5	Витрати на оплату машинного часу	1659,84	3,44
6	Накладні витрати	6000	12,45
7	Разом витрати	48189,34	100
8	Прибуток (30%)	14456,8	—
9	Всього	62646,14	—
10	Податок на додану вартість (20%)	12528,2	—
11	Підсумок	75174,34	—

6.3 Розрахунок капітальних вкладень

Обсяг капітальних вкладень споживача програми K , пов'язаних з її розробкою / купівлею / і впровадженням, визначають за формулою (6.10):

$$K = K_{\text{пп}} + K_{\text{пф}}, \quad (6.10)$$

де $K_{\text{пп}}$ – виробничі витрати, грн. ;

$K_{\text{пф}}$ – капітальні вкладення в виробничі фонди, необхідні для впровадження програми, грн. [34].

Виробничі витрати включають витрати, пов'язані з розробкою і налагодженням програмного, математичного та інформаційного забезпечення, проектуванням системи. Виробничі витрати визначають за формулою (6.11):

$$K_{пп} = \frac{S_{pn}}{n} + K_{пр.пп}, \quad (6.11)$$

де S_{pn} – сумарні витрати на розробку або придбання програмного продукту, грн;

n – кількість споживачів даної програми;

$K_{пр.пп}$ – інші виробничі витрати, грн. [34].

Капітальні вкладення в виробничі фонди включають витрати на придбання, установку, монтаж і налагодження комплексу технічних засобів (КТЗ) з урахуванням можливого використання КТЗ як для вирішення завдань для яких призначена розробляється програма, так і для завдань, що вирішуються за допомогою інших програм. Питомі капітальні вкладення в виробничі фонди, припадаючі на частку даного програмного продукту визначають за формулою (6.12):

$$K_{пф} = \frac{K_{ктс} \cdot T_{ктс}}{\Phi_{эф.ктс}}, \quad (6.12)$$

де $K_{ктс}$ – витрати на придбання, установку, монтаж і налагодження КТЗ, грн .;

$T_{ктс}$ – машинний час КТЗ, потрібне даному споживачеві програми для завдань, що вирішуються за допомогою даної програми, машино–год / рік;

$\Phi_{эф.ктс}$ – річний ефективний фонд часу роботи КТЗ, машино–год / рік [34].

$$K = \frac{75174,34}{1} = 75174,34 \text{ грн.}$$

6.4 Оцінка економічної ефективності

Витрати на розробку програми складають 75174,34 гривень. Очікується продати розроблювану програмну систему за 200000 грн.

Прибутком будуть вважатися кошти, отримані від продажу системи, тобто прибуток $\Pi = 200000$ грн.

Коефіцієнт ефективності вкладених коштів в програму E і термін їх окупності $T_{ок}$ розраховують за формулами (6.13), (6.14):

$$E = \frac{\Pi}{K}, \quad (6.13)$$

$$T_{ок} = \frac{K}{\Pi} \quad (6.14)$$

Таким чином за формулами (6.13), (6.14):

$$E = 200000 / 75174,34 = 2,66,$$

$$T_{ок} = 75174,34 / 200000 = 0,37 \text{ року.}$$

З урахуванням ризику 50 % надходження від продажів складуть 100000 гривень, що вище вартості програмного забезпечення, в якій вже була врахована рентабельність 30%.

Таким чином, можна зробити висновок про економічну доцільність розробки даного програмного забезпечення.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

У ході виконання дипломної кваліфікаційної роботи магістра досліджується та програмно реалізується система управління роботом-дезінфектором на основі аналізу відеопотоку. Під час проведення дослідження та розробки програмного забезпечення основні роботи було виконано за допомогою персонального комп'ютера (ПК) та мікрокомп'ютера у приміщенні обчислювального центру (ОЦ).

На основі загальних вимог з техніки безпеки при роботі з електронним обладнанням, а також специфічних вимог, встановлених міжнародними, внутрішніми стандартами, державними стандартами, санітарними нормами і правилами, санітарними нормами, державними будівельними нормами та іншими нормативними документами, для даної області робіт розроблено комплекс заходів, призначений для забезпечення безпечної і комфортної роботи та експлуатації проектного автоматизованого робочого місця.

На операторів ЕОМ, програмістів та інших робітників, що пов'язані з обладнанням ОЦ, у процесі роботи, за кваліфікацією ГОСТ 12.0.003 – 74 «ГОСТ 12.0.003–74 ССБТ. «Опасные и вредные производственные факторы. Классификация», впливають наступні небезпечні та шкідливі виробничі чинники:

- вплив електромагнітних полів від монітору комп'ютера, що призводить до підвищеного стомлення, зниження працездатності та профзахворювання;
- розумове та емоційне перевантаження внаслідок недотримання змін режимів праці та відпочинку, що викликає зниження працездатності та стомлення;
- недостатня освітленість робочої зони при недотриманні норм освітлення, викликає погіршення зору, підвищує кількість помилок;
- невідповідність значень параметрів мікроклімату, що знижує працездатність, погіршує самопочуття, може призвести до профзахворювань;

- можливість загоряння внаслідок порушень правил пожежної безпеки, короткого замикання електропроводки тощо;
- ураження електричним струмом у результаті порушення правил електробезпеки, несправності електрообладнання, відсутності захисного заземлення;
- непоінформованість персоналу у відповідній послідовності дій в умовах надзвичайної ситуації, що призводить до травмування та до інших непередбачених негативних наслідків.

Ці проблеми можуть вирішуватися комплексно як з позицій ергономічних вимог до умов праці (наприклад, покращення робочих місць), так і з позицій суворого підтримання регламенту режимів праці і відпочинку, цілеспрямованою профорієнтацією тощо.

7.2 Заходи із забезпечення безпеки

При роботі у ОЦ головним небезпечним чинником є вплив на персонал електричного струму.

Відповідно до ПУЕ «Правила улаштування електроустановок» [35], приміщення ОЦ за ступенем небезпеки ураження електричним струмом відноситься до 1-ї категорії приміщень без підвищеної небезпеки, оскільки існує можливість торкання корпусів апаратури й батареї центрального опалення.

Заходи по забезпеченню електробезпеки розроблено у відповідності до ДСТУ Б В.2.5-82:2016 «Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом» [36]. За способом захисту людини від ураження електричним струмом – відеотермінали, ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ відповідають першому класу захисту та заземлені відповідно до Державного нормативного акту про охорону праці 0.00-1.32-01 «Правила устройства электроустановок. электрооборудование специальных установок».

Для забезпечення електробезпеки у ОЦ використано наступні технічні заходи та засоби електробезпеки:

- ізоляція струмопровідних частин;
- недоступність струмопровідних частин;
- захисне відключення;
- ізоляція струмопровідних частин;
- захисне заземлення;
- блокування, занулення [35];
- захисне розділення електричних мереж ДСТУ ІЕС 61140:2005 «Захист проти ураження електричним струмом. Загальні аспекти щодо установок та обладнання (ІЕС 61140:2001, IDT)»;
- компенсація ємнісних струмів замикання на землю;
- вирівнювання потенціалів тощо [37].

У приміщенні ОЦ передбачено використання тільки трьохполюсних розеток, в яких один з полюсів підключено до шини захисного заземлення. З цим полюсом електрично з'єднані металеві частини обчислювальних машин, які можуть виявитись під напругою.

Кожен блок живлення комп'ютера або периферійного пристрою має мережевий фільтр. Конденсатори цього фільтру призначені для шунтування високочастотних перешкод живильної мережі на землю через дріт «земля» трьохполюсної вилки та розетки. Цей дріт може бути з'єднаний з заземлюючим контуром і з «нулем» силової мережі. При відсутності гальванічного зв'язку заземлюючого дроту з шиною захисного заземлення, на корпусі електронного пристрою може з'явитися напруга порядку 110 В, внаслідок поділення напруги мережі 220 В конденсаторами мережевого фільтру, що працює у даному випадку як ємнісний подільник напруги [37].

При з'єднанні двох електронних засобів (наприклад, комп'ютера та принтера) інтерфейсним кабелем, загальний дріт інтерфейсів послідовних та паралельних портів зв'язаний зі «схемною землею» і корпусом електронного

засобу. Тому, для усунення проблеми різності потенціалів, електронні засоби надійно заземлені через окремий дріт на загальний контур.

При підключенні електронних засобів до мережі живлення, виключено застосування мережевих дротів з двохполюсною вилкою для підключення до трьохполюсної розетки. В іншому випадку може виникнути різність потенціалів, що призведе до виходу електронної апаратури зі строю.

Відповідно до ПУЕ опір захисного заземлення у приміщенні ОЦ не перевищує 4 Ом [36].

Електрична ізоляція забезпечується конструкціями пристроїв і способами їхнього підключення. Відповідно до ПУЕ в електроустановках до 1000 В мінімальне значення опору ізоляції силових і освітлювальних електропроводок на ділянці між суміжними запобіжниками або за останніми запобіжниками між будь-яким проведенням і землею становить не менш 0,5 МОм.

Накопичена статична електрика, зокрема, на екрані монітора притягує пил, бруд та інші частини присутні в повітрі. Причому електризується не тільки екран, а і повітря на робочому місці, а також одяг користувача, особливо якщо він із синтетичного або шерстяного матеріалу [38].

В якості заходів захисту запропоновано:

- кілька разів на протязі робочого дня мити руки і обличчя водою, а після закінчення роботи вимити руки й лице з милом;
- щоденно протирати екран монітора, клавіатуру, комп'ютерну мишу антистатичною серветкою;
- щоденно в приміщенні з персональними комп'ютерами проводити вологе прибирання;
- встановити нейтралізатори статичної електрики;
- підтримувати у приміщенні вологість повітря зазначену в нормативних документах;
- користувачу персонального комп'ютера бажано носити одяг з природних волокон, в тому числі і шкарпетки.

З метою визначення стану електрообладнання та виявлення дефектів проводяться профілактичні випробування відносно до ПУЕ та будівельних норм і правил.

Профілактичні випробування включають: контроль ізоляції, контроль з'єднання дротів, вимірювання опору заземлюючих пристроїв, перевірку спрацювання лінії захисту і запобіжних пристроїв.

7.3 Заходи з виробничої санітарії і гігієни праці

Виходячи зі специфіки роботи у обчислювальних центрах ОЦ можна відмітити, що всі види робіт в них не пов'язані з великими енергозатратами робітників, а також виділенням шкідливих речовин у процесі роботи, а основним видом відходів є папір, що повністю утилізується і не забруднює навколишнє середовище. Таким чином можна вважати, що за ступенем тяжкості роботи у ОЦ відносяться до категорії «легка робота».

Так як робота з обчислювальною технікою є малорухомою і створює велике навантаження на очі, кисті рук та опорно-руховий апарат, операторам для підтримки працездатності необхідно керуватися НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [38]. При роботі за дисплеєм у режимі праці і відпочинку передбачаються через кожні 40-45 хвилин 3-5 хвилинні перерви на відпочинок та гімнастичні вправи.

Виробниче середовище в приміщеннях ОЦ сполучає в собі раціональні архітектурно-планувальні рішення та найбільш оптимальні санітарно-гігієнічні умови.

У приміщенні ОЦ на робочих місцях з комп'ютерною технікою нормалізація параметрів повітряного середовища забезпечується згідно до вимог Державних санітарних норм 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [39], що наведено в таблиці 7.1.

Таблиця 7.1 Норми мікроклімату для приміщень з КТ

Пора року	Категорія робіт	Температура повітря, °С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	легка -1а	22-24	40-60	0,1
	легка -1б	21-23	40-60	0.1
Тепла	легка -1а	23-25	40-60	0.1
	легка -1б	22-24	40-60	0.2

Рівні позитивних і негативних іонів у повітрі приміщення ОЦ відповідають санітарно-гігієнічним нормам 2152-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень», що наведені в табл. 7.2 [40].

Таблиця 7.2 – Рівні іонізації повітря приміщень з КТ

Рівні іонізації повітря	Кількість іонів в 1 см ³ повітря	
	п+	п-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

Всі норми по освітленості визначені згідно Державних будівельних норм В.2.5-28-2006 «Природне і штучне освітлення» [36].

Природне освітлення здійснюється через віконні прорізи, при цьому коефіцієнт природної освітленості не нижче 1,5% за ДБН В.2.5-28:2018.

У тих випадках, коли одного природного освітлення недостатньо, використовується штучне освітлення. Штучне освітлення в приміщеннях здійснюється системою загального рівномірного освітлення, при цьому штучне освітлення застосовують не тільки в темний, але й у світлий час доби, найбільш

прийнятними для приміщень ОЦ є люмінесцентні лампи ЛБ (білого світла) потужністю 20, 40 або 80 Вт, які встановлюються в світильники типу ЛПО [36].

При застосуванні люмінесцентних ламп дотримуються наступних умов:

- температура навколишнього повітря не повинна бути нижче за 5 °С;
- напруга на освітлювальних приладах повинна бути не менше ніж 90% від номінальної [36].

Допускається застосування ламп розжарювання у світильниках місцевого освітлення. Загальне освітлення варто виконувати у виді суцільних чи переривчастих ліній світильників, розташованих збоку від робочих місць.

Розрахуємо систему загального рівномірного штучного освітлення в офісному (адміністративно-побутовому) приміщенні [37].

Визначимо розміри приміщення: довжина $A = 12$ м, ширина $B = 5$ м, висота $H = 4$ м.

Висота робочої поверхні складає $h = 0,7$ м.

Коефіцієнти відбиття поверхонь приміщення: колір стелі, стін – сірий, незначне пиловиділення, $c=50\%$; $ст=30\%$; $п=10\%$,

Розряд зорової роботи – III б.

Згідно з Державними будівельними нормами В.2.5-28-2006 «Природне і штучне освітлення. Норми проектування» рівень нормованого загального освітлення в приміщенні нормуються як $E_n = 200$ лк.

Обираємо світильник типу ЛВП, призначений для приміщень висотою до 6 м. $IP=65$ – ступінь захисту світильника, $[L/h] = 1,4$ – числове значення коефіцієнта світильника.

Коефіцієнт запасу k_z враховує зниження рівня освітленості з часом в результаті забруднення та старіння ламп, світильників і поверхонь приміщення, приймається в залежності від виробничих умов. Приймаємо $k_z = 1,6$.

Коефіцієнт нерівномірності (мінімального) освітлення z (відношення середньої освітленості до мінімальної освітленості), для люмінесцентних ламп низького тиску, як правило дорівнює $z = 1,1$.

Розраховуємо кількість рядів світильників у приміщенні за формулою (7.1):

$$N_p = \frac{B}{(H-h_p) \cdot [L/h]} = \frac{5}{(4-0,7) \cdot 1,4} = 1,1 \approx 2 \text{ ряди світильників.} \quad (7.1)$$

Визначимо максимально припустиму відстань між рядами світильників за формулою (7.2):

$$L_{max} = \frac{B}{N_p} \quad (7.2)$$

де B – ширина приміщення;

N_p – кількість рядів.

Розрахуємо:

$$L_{max} = \frac{5}{2} = 2,5 \text{ м.}$$

Висота підвісу світильника над робочою поверхнею обчислюється за формулою (7.3):

$$h = \frac{L_{max}}{[L/h]} \quad (7.3)$$

Розрахуємо:

$$h = \frac{2,5}{1,4} = 1,8 \text{ м.}$$

Знайдемо висоту звисання світильника від стелі за формулою (7.4):

$$h_3 = H - h_p - h, \quad (7.4)$$

де H – висота приміщення;

h_p – висота робочої поверхні;

h – висота підвісу світильника над робочою поверхнею.

Розрахуємо:

$$h_3 = 4 - 0,7 - 1,8 = 1,5 \text{ м.}$$

Чисельне значення індексу приміщення визначаємо за рівнянням (7.5):

$$i = \frac{AB}{h(A+B)} \quad (7.5)$$

де A – довжина приміщення, м;

B – ширина приміщення, м;

h – висота розміщення світильників над робочою поверхнею, м.

Розрахуємо:

$$i = \frac{12 \cdot 5}{1,8 \cdot (12 + 5)} = 1,96.$$

Значення коефіцієнта використання світлового потоку η вибирається в залежності від виду джерела світла, типу обраного світильника, коефіцієнтів відбиття поверхонь приміщення (ρ_c , $\rho_{ст}$, $\rho_{п}$) та індексу приміщення. В нашому випадку $\eta = 54\%$.

Визначаємо сумарний світловий потік освітлювальної установки у даному виробничому приміщенні за формулою (7.6):

$$\Phi_{\Sigma} = \frac{E_n \cdot S \cdot k_3 \cdot z}{\eta}, \quad (7.6)$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, лм;

E_n – нормоване значення освітленості, лк;

S – площа освітлюваної поверхні, м^2 ;

k_z – коефіцієнт запасу;

z – коефіцієнт нерівномірності (мінімальної) освітленості;

η – коефіцієнт використання світлового потоку.

Розрахуємо:

$$\Phi_{\Sigma} = \frac{200 \cdot 12 \cdot 5 \cdot 1,6 \cdot 1,1}{0,54} = 39111 \text{ лм.}$$

Визначаємо умовну загальну кількість світильників у приміщенні за формулою (7.7):

$$N^* = A \cdot \frac{B}{L_{\max}^2}. \quad (7.7)$$

Розрахуємо:

$$N^* = \frac{12 \cdot 5}{2,5 \cdot 2,5} = 9,6 \approx 10 \text{ шт.}$$

Розрахуємо світловий потік умовного джерела світла за формулою (7.8):

$$\Phi_{\text{л}} = \frac{\Phi}{N_{\text{л}}}, \text{ лм,} \quad (7.8)$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, лм;

$N_{\text{л}}$ – загальна кількість ламп у приміщенні, шт.

Загальна кількість ламп у приміщенні визначається за формулою (7.9):

$$N_{\text{л}} = N^* n, \quad (7.9)$$

де n – кількість ламп у світильнику (в ЛПП світильнику знаходиться 2 лампи).

Розрахуємо:

$$N_{\text{л}} = 10 \cdot 2 = 20 \text{ шт.}$$

Тоді,

$$\Phi_{\text{л}} = \frac{39111}{20} = 1955 \text{ лм.}$$

Обираємо тип стандартної лампи з найближчим значенням фактичного світлового потоку лампи $\Phi_{\text{л}}$ і знайти коефіцієнт m (співвідношення між розрахунковим світловим потоком лампи $\Phi_{\text{л}}^*$ та фактичним світловим потоком вибраної стандартної лампи $\Phi_{\text{л}}$):

Обираємо лампу ЛБ із $P_{\text{л}}=30$ Вт, $\Phi_{\text{л}} = 2180$ лм. Розрахунки виконаємо за формулою (7.10):

$$m = \frac{\Phi_{\text{л}}^*}{\Phi_{\text{л}}}. \quad (7.10)$$

Відповідно:

$$m = \frac{1955}{2180} = 0,9.$$

Визначаємо оптимальну (фактичну) кількість світильників у приміщенні за формулою (7.8).

Фактична кількість ламп у приміщенні розраховується за формулою (7.11):

$$N_{\text{фл}} = N_{\text{ф}} \cdot n \cdot m. \quad (7.11)$$

Розрахуємо:

$$N_{\text{фл}} = 10 \cdot 2 \cdot 0,9 = 18 \text{ шт.}$$

Виходячи з вимоги кратності загальної кількості світильників кількості рядів, для освітлення приміщення застосовуємо 10 світильників, по 5 у кожному ряді.

Визначаємо загальну розрахункову освітленість E_p у приміщенні, що створюється при застосуванні стандартних ламп, за формулою (7.12):

$$E_p = \frac{\Phi_{\text{л}} \cdot N_{\text{фл}}}{Sk_z} \quad (7.12)$$

Розрахуємо:

$$E_p = \frac{2180 \cdot 20 \cdot 0,54}{12 \cdot 5 \cdot 1,6 \cdot 1,1} = 223 \text{ лк.}$$

Тобто виконується умова $E_p = (-10\% \dots +20\%) \cdot E_n$, лк.

Розраховуємо загальну потужність освітлювальної установки за формулою (7.13):

$$P_{\Sigma} = N_{\text{фл}} \cdot P_{\text{л}}, \quad (7.13)$$

де $P_{\text{л}}$ – потужність вибраної стандартної лампи.

Розрахуємо:

$$P_{\Sigma} = 20 \cdot 30 = 600 \text{ Вт.}$$

Покриття підлоги матове з коефіцієнтом відбиття 0,3-0,5. Поверхня підлоги рівна, неслизька, з антистатичними властивостями.

Для підвищення продуктивності праці, а також зниження рівня стомлюваності при роботі з ЕОМ передбачені десятихвилинні перерви через кожну годину.

Для зниження нервово-емоційного напруження, втомлення зорового аналізатору, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії та запобігання втомі доцільно деякі з перерв використовувати для виконання комплексу фізичних вправ.

Для захисту від впливу шуму використано організаційні та технічні заходи відповідно до ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку [43].

Джерелами шуму при роботі з ПК є:

- вентилятор блока живлення мережі;
- вентилятор, розташований на процесорі;
- жорсткий диск;
- механічні сканери;
- швидкісні CD-ROM;
- пересувні механічні частини принтера.

До організаційних заходів відносять:

- раціональне розташування устаткування та робочих місць;
- постійний контроль режиму праці і відпочинку працівників;
- обмеження застосування обладнання та використання робочих місць, що не відповідають санітарно-гігієнічним вимогам.

Технічні заходи дають змогу значно зменшити вплив шуму на працівників і поділяються на заходи, що використовуються:

- в джерелі виникнення (конструктивні та технологічні);
 - на шляху розповсюдження (звукоізоляція, звукопоглинання, глушники шуму, звукоізоляційні укриття);
 - використання засобів колективного та індивідуального захисту
- Державних санітарних правил і норм 3.3.2.007-98 «Державних санітарних правил

і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Зниження рівня шуму в приміщенні можна здійснити за Державними санітарними нормами 3.3.6-037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвук» наступним чином:

- використанням блоків живлення ПК з вентиляторами на гумових підвісках;
- використанням ПК, в яких термодавачі (термодатчики) вмонтовані в блоці живлення та в критичних точках материнської плати (процесор, мікросхеми чіпсету), які дозволяють програмним шляхом регулювати як моменти ввімкнення вентиляторів, так і їх швидкість обертання;
- переведення жорсткого диска в режим сплячки (Standby), якщо комп'ютер не працює протягом визначеного часу;
- застосуванням приводів з одночасним зчитуванням декількох доріжок CD чи програми, що дозволяє зменшити швидкість;
- заміною матричних голчатих принтерів струменевими і лазерними принтерами, які забезпечують при роботі значно менший рівень звукового тиску;
- застосуванням принтерів колективного користування, що розташовуються на значній відстані від більшості робочих місць користувачів ПК;
- зменшенням шуму на шляху його поширення через розміщення звукоізолюючого відгородження у вигляді стін, перетинок, кабін;
- акустичною обробкою приміщень – зменшення енергії відбитих звукових хвиль шляхом збільшення площі звукопоглинання (розміщення на поверхнях приміщення облицювань, що поглинають звук, розташування в приміщеннях штучних поглиначів звуку) [44].

Рівні електромагнітного випромінювання та магнітних полів відповідають вимогам ГОСТ 12.1.006 «Система стандартів безпеки праці. Електромагнітні поля радіочастот. Допустимі рівні на робочих місцях і вимоги до проведення контролю» та Державних санітарних правил і норм 3.3.2.007-98

«Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Джерелами електромагнітних випромінювань є:

- блоки живлення від мережі (частота – 50 Гц);
- система кадрової розгортки (5 Гц-2 кГц);
- система рядкової розгортки (2-400 кГц);
- блок модуляції променя (5-10 МГц).

Для зниження негативного впливу електромагнітного випромінювання здійснено:

- використання на робочому місці відеотерміналів, що відповідають сучасним вимогам захисту від випромінювань;
- не переобтяжувати приміщення значною кількістю робочих місць з відеотерміналами [43].

7.4. Заходи безпеки у надзвичайних ситуаціях

7.4.1 Заходи з пожежної безпеки

В приміщенні ОЦ джерелом загоряння можуть бути електронні схеми ЕОМ, прилади для технічного обслуговування, електроустаткування, кондиціонери, де за рахунок різних порушень відбувається перегрів тих чи інших елементів, електричні іскри, дуга, що можуть визвати загоряння горючих матеріалів, дерев'яні та пластикові стільці та столи, папір, перевантаження мережі живлення, людські чинники тощо.

Великою небезпекою для обчислювального центру є пожежа. Обчислювальний центр наповнений різноманітними електричними приладами, а пожежа основний фактор для їх знищення, що може призвести до великих матеріальних втрат. Так як в приміщенні обчислювального центру присутні всі три основні чинника, необхідні для виникнення пожежі: горючі речовини, окислення і джерела запалювання, то налагодження заходів пожежної безпеки в такому приміщенні є першочергово важливим.

Ймовірними причинами виникнення пожежу можуть бути несправність електрообладнання (кабелів, розеток), короткі замикання внаслідок виходу з ладу чи експлуатації несправного електроустаткування (ПЕОМ, периферійних пристроїв), порушення правил протипожежної безпеки тощо.

Для забезпечення пожежної безпеки проводиться пожежна профілактика, яка включає в себе комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, на запобігання пожежі, обмеження її поширення, а також на створення умов для успішного гасіння пожежі.

Експлуатація ліній електромережі практично повністю унеможлиблює виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, так як застосовуються дроти з важкогорючою і негорючою ізоляцією.

Для відводу теплоти від ПК у обчислювальному центрі постійно діє потужна система кондиціонування.

Обчислювальний центр відноситься до категорії «П-Па» згідно ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» [45], згідно ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [46], а клас можливої пожежі визначається, як «А».

Для ліквідації пожежі у початковій стадії її розвитку силами персоналу об'єктів застосовуються первинні засоби пожежогасіння.

Так як приміщення ОЦ має площу 60 м², тому відповідно до вимог п. 3.8 розділу «Типові норми належності вогнегасників» ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги» [44] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 3 штуки (з розрахунку один вогнегасник з величиною заряду вогнегасної речовини 3,5 кг і більше, на 20 м² площі приміщення). Відстань між вогнегасниками та місцями можливих загорянь не перевищує 15 м.

Під час прийняття на роботу і в процесі праці усі працівники обчислювального центру повинні проходити протипожежний інструктаж, перевірку знань з питань пожежної безпеки. Евакуаційні виходи і проходи утримуються постійно вільними та легкодоступними.

Відповідальний за протипожежний стан приміщень обчислювального центру повинен періодично (не рідше, ніж один раз на місяць) перевіряти працездатність пристрою, що забезпечує автоматичне вимкнення системи вентиляції у разі пожежі, а також вогнедимозатримувальних пристроїв. Під час ремонту слід не допускати руйнування або порушення цілісності негорючих діафрагм у фальш-підлоговому просторі [44].

7.4.2 Заходи з цивільного захисту. Порядок проведення аварійно-рятувальних та інших невідкладних робіт.

Проведення аварійно-рятувальних та інших невідкладних робіт (АРІНР) полягає у виконанні заходів, що передбачені чинним законодавством з питань ліквідації наслідків стихійного лиха, аварій і катастроф, епідемій і епізоотій, що створюють загрозу життю і здоров'ю населення. Для проведення АРІНР під час ліквідації наслідків стихійних лих, аварій, катастроф, епідемій, що створюють загрозу життю і здоров'ю населення створюються формування цивільної оборони. До проведення таких робіт залучаються: невоєнізовані формування цивільної оборони, а також медичні організації та військові частини і підрозділи.

Для проведення АРІНР необхідно:

- завчасно спланувати дії формувань цивільної оборони як при загрозі нападу так і під час проведення АРІНР;
- організувати з робітників, службовців та студентів формування цивільної оборони й підготувати їх для роботи в осередках ураження;
- оснастити формування цивільної оборони індивідуальними засобами захисту, приладами, майном та технікою;

- перевірити та уточнити плани цивільної оборони на навчаннях, що проводяться на об'єктах;
- вивести в найбільш короткі строки формування цивільної оборони, створені в містах, в заміську зону, розмістити їх в завчасно намічених районах та привести в готовність до проведення рятувальних робіт;
- організувати управління та керівництво формуваннями цивільної оборони при проведенні рятувальних робіт.

До заходів, які необхідні під час проведення рятувальних робіт можна віднести наступні: локалізація та гасіння пожеж, розвідка маршрутів для формувань цивільної оборони, пошук та рятування людей з-під завалів, а також зруйнованих захисних споруд, надання першої медичної допомоги та евакуація потерпілих, санобробка людей та знезараження їх одягу, знезараження місцевості, споруд і техніки. Під невідкладними роботами в осередках ураження розуміють:

- ліквідація або укріплення аварійних споруд;
- локалізація аварій;
- знешкодження боєприпасів, вогне- та вибухонебезпечних предметів;
- прокладання шляхів на заражених територіях і проїздів у завалах;
- відновлення та ремонт пошкоджених захисних споруд.

Рятувальні роботи в закладі (установі) поділяють на наступні два етапа:

- перший етап триває з моменту отримання сигналу про небезпеку до прибуття спеціалізованих формувань цивільної оборони;
- другий етап – від прибуття формувань цивільної оборони до виконання ними поставлених завдань (вивезення членів колективу в безпечну зону, евакуацію поранених у лікувальні заклади)

До складу невоєнізованих формувань входять такі групи: зв'язку, із забезпечення громадського порядку, протипожежної служби, медичної допомоги, протирадіаційного та протихімічного захисту.

Група зв'язку складається з керівника групи та помічників (черговий біля телефону та посильні). Завдання групи – оповіщення членів колективу про

загрозу виникнення надзвичайної ситуації, передача сигналу структурам цивільної оборони міста (району), підтримка засобів зв'язку у стані постійної готовності, забезпечення штабу цивільної оборони навчального закладу засобами зв'язку.

Група із забезпечення громадського порядку, керівник – працівник установи, що відповідає за її охорону. Завдання групи: забезпечення охорони закладу, підтримка порядку у випадках надзвичайних ситуацій та надання допомоги керівництву під час евакуації.

Група протипожежної служби: бере участь у розробці протипожежних профілактичних засобів, здійснює контроль за їх виконанням, забезпечує постійну готовність до використання засобів пожежогасіння, локалізує та гасить пожежу, надає допомогу у проведенні спеціальної обробки території.

Група медичної допомоги: організовується на базі медичного пункту закладу або установи. Завдання групи: організація і проведення санітарно-гігієнічних та профілактичних заходів, надання медичної допомоги ураженим, допомога при їх евакуації до лікарень, проведення часткової санітарної обробки уражених.

Група протирадіаційного та протихімічного захисту, організовує видачу засобів індивідуального захисту, здійснює контроль за радіаційною та хімічною ситуацією у закладі та на прилеглий території, за станом сховищ та укриттів, бере участь у заходах з ліквідації наслідків радіаційного та хімічного зараження.

При ліквідації наслідків НС виникає необхідність в організації рухомих пунктів харчування, речового і продовольчого забезпечення.

Підрозділи громадського харчування (пересувні пункти харчування ЦО) призначені для забезпечення гарячим харчуванням особового складу формувань в районах розміщення та при проведенні АРІНР, а також потерпілого населення в загонах першої допомоги [47].

ВИСНОВКИ

Розглянуто проблему автоматизації дезінфекції приміщень та забезпечення захисту людей від УФ опромінення, які знаходяться у приміщенні під час дезінфекції.

Виконано огляд існуючих рішень, які можуть використовуватися для дезінфекції приміщень, зокрема Model C від компанії UVD Robots, що є більш універсальним рішенням та може використовуватися у лікарнях, офісах, готелях, аеропортах, торговельних центрах, житлових будинках та інших приміщеннях, а також Робот TMI Rob Intelligent Disinfection Robot від компанії Shanghai TMI Robotics, що призначений для використання в медичних установах.

Проаналізовано нейромережеві методи, які можуть застосовуватись для розпізнавання людей на відеокadraх. Проаналізовано метод обробки відеокadру, що застосовується для приведення кадру до вигляду сферичної проекції. Розроблено систему для керування УФ лампою робота-дезінфектора.

Визначено функції, які повинна виконувати програма. Створено діаграму прецедентів для моделювання функціональних вимог до роботи системи. Обрано мову програмування C++ для реалізації програми. В якості фреймворку для розробки програмного забезпечення роботів обрано ROS. Бібліотекою для обробки зображень обрано OpenCV.

Реалізовано програмне забезпечення керування УФ лампою робота-дезінфектора, описано розгортання програмного забезпечення, описано схему взаємодії між вузлами системи та принципи реалізації взаємодії між ними на основі моделі Публікація-підписка, описано основні реалізовані класи.

Усі завдання, поставлені на роботу, виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. COVID-19 Pandemic Impact: A Surge in Proven Mobile Robotics Use Cases for Disinfection, Monitoring, Surveillance, and Delivery Will Propel the Market to US\$23 Billion by 2021 – [Electronic resource]. – Access mode: <https://www.abiresearch.com/press/covid-19-pandemic-impact-surge-proven-mobile-robotics-use-cases-disinfection-monitoring-surveillance-and-delivery-will-propel-market-us23-billion-2021/>
2. Методичні рекомендації щодо використання ультрафіолетового бактерицидного випромінювання – [Електронний ресурс]. – Режим доступу: <https://phc.org.ua/news/metodichni-rekomendacii-schodo-vikoristannya-ultrafioletovogo-baktericidnogo-viprominyuvannya>
3. Model C – [Electronic resource]. – Access mode: <https://www.uvd-robots.com/modelc>
4. TMiRob Series - Intelligent Sterilization Robot (ISR) – [Electronic resource]. – Access mode: <https://www.time-medical.com/intelligent-sterilization-robot>
5. 9 disinfection robots fighting the coronavirus – [Electronic resource]. – Access mode: <https://tectales.com/bionics-robotics/9-disinfection-robots-fighting-the-coronavirus.html>
6. TMiRobIntelligent Disinfection RobotProduct White Paper – [Electronic resource]. – Access mode: <http://frontierinstruments.com.au/wp-content/uploads/2020/03/TMI-disinfection-robot-whitepaper.pdf>
7. Нейронные сети, перцептрон – [Электронный ресурс]. – Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=Нейронные_сети,_перцептрон
8. Нейронная сеть – [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Нейронная_сеть
9. Глубокое обучение – [Электронный ресурс]. – Режим доступа: http://neerc.ifmo.ru/wiki/index.php?title=Глубокое_обучение

10. Neural Networks and Deep Learning – [Electronic resource]. – Access mode: <http://neuralnetworksanddeeplearning.com/about.html>
11. The neural network zoo – [Electronic resource]. – Access mode: <https://www.asimovinstitute.org/neural-network-zoo/>
12. Свёрточная нейронная сеть – [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Свёрточная_нейронная_сеть
13. Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество – [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/348000/>
14. A Beginner's Guide To Understanding Convolutional Neural Networks Learning – [Electronic resource]. – Access mode: <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/>
15. Сверточные нейронные сети – [Электронный ресурс]. – Режим доступа: http://neerc.ifmo.ru/wiki/index.php?title=Сверточные_нейронные_сети
16. Задача нахождения объектов на изображении – [Электронный ресурс]. – Режим доступа: http://neerc.ifmo.ru/wiki/index.php?title=Задача_нахождения_объектов_на_изображении
17. YOLOv4: Optimal Speed and Accuracy of Object Detection – [Electronic resource]. – Access mode: <https://arxiv.org/pdf/2004.10934.pdf>
18. You Only Look Once: Unified, Real-Time Object Detection – [Electronic resource]. – Access mode: <https://arxiv.org/pdf/1506.02640.pdf>
19. Распознавание человека по изображению лица нейросетевыми методами – [Электронный ресурс]. – Режим доступа: http://uiip.bas-net.by/structure/l_ori/starovoitov/Starovoitov_Publication_section/11_Starovoitov02_prep.pdf
20. Fisheye lens – [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Fisheye_lens
21. Converting a fisheye image into a panoramic, spherical or perspective projection – [Electronic resource]. – Access mode: <http://paulbourke.net/dome/fish2/>

22. Python (programming language) – [Electronic resource]. – Access mode: [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))
23. Знакомство с Python – [Электронный ресурс]. – Режим доступа: <https://younglinux.info/python/introductionpython.php>
24. C++ – [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/C%2B%2B>
25. Язык программирования C++ – [Электронный ресурс]. – Режим доступа: <https://metanit.com/cpp/tutorial/1.1.php>
26. Java – [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Java>
27. Язык программирования Java – [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/tutorial/1.1.php>
28. Python vs C++: Selecting the Right Tool for the Job – [Electronic resource]. – Access mode: <https://realpython.com/python-vs-cpp/>
29. Введение в Robot Operating System – [Электронный ресурс]. – Режим доступа: <http://docs.voltbro.ru/starting-ros/ros-about.html>
30. Обзор свободного фреймворка ROS - операционной системы для роботов – [Электронный ресурс]. – Режим доступа: <https://lvee.org/ru/abstracts/116>
31. ROS Concepts – [Electronic resource]. – Access mode: <http://wiki.ros.org/ROS/Concepts>
32. Сравнение библиотек компьютерного зрения для применения в приложении, использующем технологию распознавания плоских изображений – [Электронный ресурс]. – Режим доступа: <https://cyberleninka.ru/article/n/sravnenie-bibliotek-kompyuternogo-zreniya-dlya-primeneniya-v-prilozhenii-ispolzuyuschem-tehnologiyu-raspoznavaniya-ploskih>
33. OpenCV – [Электронный ресурс]. – Режим доступа: <https://uk.wikipedia.org/wiki/OpenCV>
34. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальностей 8.05010301 «Програмне забезпечення

автоматизованих систем» та 8.05010201 «Інформаційні технології проектування» для усіх форм навчання [Текст] / В.В. Остапенко, Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2015. – 22 с.

35. Правила улаштування електроустановок [Текст] : ПУЕ-2017. – На заміну ПУЕ-86; чинний з 2017-08-21. – К.: Міненерговугілля України, 2017. – 617 с.

36. ДСТУ Б В.2.5-82:2016 Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом. - Київ, ДП "УкрНДНЦ", 2016. – 109 с.

37. Батлук В. А. Охорона праці: навчальний посібник. Третє видання, доповнене / В. А. Батлук, М. П. Кулик, Р. А. Яцюк. – Львів: Видавництво Львівської політехніки, 2011. - 388 с.

38. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. НПАОП 0.00-7.15-18. Державні санітарні правила і норми. – Харків: Форт, 2018 – 48 с.

39. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. – Харків: Форт, 2000 – 24 с.

40. Жидецький, В. Ц. Основи охорони праці: підручник / В. Ц. Жидецький. – 5-те вид., доп. – К. : Знання, 2014. – 373 с.

41. ДБН В.2.5-28:2018. Природне та штучне освітлення: посібник: проблеми природного і штучного освітлення. – К: Етін, 2019. – 180 с.

42. Методичні вказівки до лабораторного заняття «Розрахунок загального рівномірного штучного освітлення виробничих приміщень» з дисципліни «Цивільний захист і охорона праці в галузі»: для студентів усіх спеціальностей та форм навчання /Укл. : В.І. Шмирко, О.В. Коробко, Ю.І. Троян. – Запоріжжя: каф. ОПіНС. НУ «Запорізька політехніка», 2020. – 36с.

43. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку. – МОЗ України, 1999 – 24 с.

44. Березуцький В.В. Небезпечні виробничі ризики та надійність: навч. посібник / В.В. Березуцький, М.І. Адаменко; Харківський політехнічний ін-т, нац. техн. ун-т. – Харків: ФОП Панов А.М., 2016. – 385 с.

45. ДСТУ Б В.1.1-36:2016. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою [Текст] :– На заміну НАПБ Б.03.002-2007 ; чинний від 2017-01-01. – К. : Мінрегіонбуд України, 2016. – 66 с. – (Державний Стандарт України)

46. Пожежна безпека об'єктів будівництва. Загальні вимоги [Текст] : ДБН В.1.1-7:2016. – На заміну ДБН В.1.1.7-2002 ; чинний від 2017-06-01. – К. : Мінрегіон України, 2017. – 47 с. – (Державні будівельні норми).

47. Стеблюк М. І. Цивільна оборона та цивільний захист [Текст] : навч. посіб. для вузів / М. І. Стеблюк. – К.: Знання, 2013, - 487 с.

ДОДАТОК А
Технічне завдання

Вступ

Метою роботи є програмна реалізація системи управління УФ лампою робота-дезінфектора на основі аналізу відеопотоку для забезпечення захисту людей від УФ опромінення, що знаходяться у приміщенні, яке дезінфікується.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра за темою “Дослідження та програмна реалізація системи управління роботом-дезінфектором на основі аналізу відеопотоку. Research and software implementation of disinfection robot control system based on video flow analysis”, затверджене наказом № 301 від 3 листопада 2020 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Розробка призначена для реалізації системи, яка буде аналізувати відеопотік у режимі реального часу та визначати є в приміщенні, що дезінфікується, люди чи немає та в залежності від їх наявності вмикати або вимикати УФ лампу, що забезпечить захисту людей від УФ опромінення.

A.3 Вимоги до програми або програмного продукту

A.3.1 Вимоги до функціональних характеристик

Для забезпечення встановленої цілі система повинна виконувати наступні функції:

- оброблювати відеокадри;
- розпізнавати людей на відеокадрі;
- вмикати УФ лампу;

– вимикати УФ лампу.

A.3.2 Вимоги до надійності

Для забезпечення надійного функціонування, по-перше, має використовуватися мікрокомп'ютер, який відповідає мінімальним вимогам системи, а по-друге, має забезпечуватися надійне мережне з'єднання.

A.3.3 Умови експлуатації

Для забезпечення експлуатації програми ніяких особливих вмінь не вимагається.

A.3.4 Вимоги до складу та параметрів технічних засобів

Для роботи системи мають забезпечуватися наступні мінімальні вимоги:

- графічний процесор з 128 ядрами Cuda з розрахунковою продуктивністю 472 GFLOPS;
- процесор з 4 ядрами з частотою 2,5 ГГц;
- обсяг оперативного запам'ятовуючого пристрою – 4 Гб;
- флеш-пам'ять – 16 Гб;
- пропускна здатність мережного інтерфейсу – 100 Мбіт/с.

A.3.5 Вимоги до транспортування та збереження

Збереження програми має виконуватися на мікрокомп'ютері. Окрім того остання версія програми має зберігатися на переносних носіях для забезпечення накопичення версій.

A.3.6 Вимоги до програмної документації

Програмна документація розміщується в додатках пояснювальної записки на роботу і повинна включати технічне завдання та текст програми. В тексті програми приводиться фрагмент виконаної програмної реалізації.

A.3.7 Вимоги до маркування та упакування

Якщо програма зберігається на переносному носії, то він має обов'язково зберігатися в чохлі, на якому має бути зафіксовано назву роботи.

A.4 Стадії та етапи розробки

Розробка повинна бути проведена в таких стадіях:

- розробка технічного завдання;
- робоче проектування;
- розробка та програмування;
- розроблення документації;
- тестування;
- впровадження.

На стадії розробки технічного завдання повинні бути виконані перераховані нижче роботи:

- постановка завдання;
- визначення та уточнення вимог до технічних засобів;
- визначення вимог до програми;
- визначення стадій, етапів та термінів розробки програми і документації до неї;
- вибір мов програмування;
- погодження та утвердження технічного завдання.

На стадії робочого проектування виконується дизайн архітектури ПЗ.

На стадії розробки програми необхідно виконати роботу з програмування і розміщення програми.

На стадії розробки програмної документації повинна бути виконана розробка програмних документів до ПЗ.

На стадії тестування тестувальники повинні виконати пошук дефектів в програмному забезпеченні і порівняти описану в вимогах поведінку системи з реальною. При виявленні дефекту тестувальник має скласти звіт про помилку, який передається розробникам. Останні його повинні виправити, після чого тестування повторюється для того, щоб переконатися, що проблема була виправлена, і чи виправлення не стало причиною появи нових дефектів в продукті. Коли програма протестована і в ній більше не виявлено серйозних дефектів, розпочинається етап реалізації і передачі ПЗ кінцевим користувачам.

А.5 Порядок контролю та приймання

Дипломна кваліфікаційна робота магістра після виконання має бути прийнята керівником та консультантами усіх відповідних розділів, зазначених у завданні на роботу, після чого подається до захисту після виконання всіх відповідних процедур.

Програма, що розробляється, повинна реалізовувати функції, визначені технічним завданням. Основні результати мають бути представлені на презентації.

ДОДАТОК Б
Текст програми

Б.1 Текст файла videocapturer.h

```

#ifndef VIDEOCAPTURER_H
#define VIDEOCAPTURER_H

#include <QObject>
#include <QMutex>
#include <QEventLoop>
#include <QTimer>
#include <QElapsedTimer>
#include <string>

#include <opencv2/opencv.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/videoio.hpp>
#include "opencv2/imgcodecs.hpp"

#include "fpscounter.h"

#include <ros/ros.h>
#include <std_msgs/Int16.h>
#include <std_msgs/UInt8MultiArray.h>
#include <std_msgs/String.h>

class VideoCapturer : public QObject
{
    Q_OBJECT

    const int _INITIAL_WIDTH = 640;
    const int _INITIAL_HEIGHT = 640;
    const int _HORIZONTAL_STEP = 50;
    const int _VERTICAL_STEP = 10;

```

```

const int SMALL_WIDTH = 1472;
const double _MAX_SCALE = 1;
const double _MIN_SCALE = 0.1;
const double _SCALE_STEP = 0.1;

cv::Rect _leftRect, _rightRect;
int _currentWidth;
int _currentHeight;
double _scale;
int _positionX;
int _positionY;
int _frameWidth;
int _frameHeight;
int _frame_squeeze_level;
cv::Size _frameSize;

static constexpr auto DefaultApiPreference =
cv::CAP_DSHOW; //CAP_GSTREAMER

const std::string _pathToMap_x_1472 =
"/home/tima/tima/src/tima_camera360/config/xmap_insta360_1472x736_
b25.pgm";

const std::string _pathToMap_y_1472 =
"/home/tima/tima/src/tima_camera360/config/ymap_insta360_1472x360_
b25.pgm";

const std::string _pathToMap_x_3006 =
"/home/tima/tima/src/tima_camera360/config/xmap_insta360_video_full_
b60.pgm";

const std::string _pathToMap_y_3006 =
"/home/tima/tima/src/tima_camera360/config/ymap_insta360_video_full_
b60.pgm";

cv::Mat _map_x;

```



```

        connect(&timer, &QTimer::timeout, [this, index,
eventLoop]() {
            if (index != frameIndex) {
                eventLoop->quit();
            }
        });

        timer.setSingleShot(true);
        timer.start(0);

        eventLoop->exec();

        delete eventLoop;
    }

    frameMutex.lock();

    cv::Mat lastFrame;
    frame.copyTo(lastFrame);
    index = frameIndex;

    frameMutex.unlock();

    return lastFrame;
}

signals:

    void onFrame();

public slots:
    void releaseCam();

private:

```

```

    void init();
    void onFrameReceive();
    void onFullFrameReceive();
    void onAngleReceive();
    void onCommandReceive(const std_msgs::String& msg);
    void moveUp();
    void moveDown();
    void moveLeft();
    void moveRight();
    void zoomIn();
    void zoomOut();
    void centralize();

private slots:

    void loop();
    void fpsCounterUpdate(int fps);

protected:
    int getRectX();
    int getRectY();
    int getCurrentWidth();
    int getCurrentHeight();
    void setCurrentFrame();
    void setRects();

};

#endif // VIDEOCAPTURER_H

```

Б.2 Текст файла videocapturer.cpp

```

#include <QCoreApplication>
#include <QTimer>
#include <QDebug>

```

```

#include "videocapturer.h"

VideoCapturer::VideoCapturer(int index, ros::NodeHandle *nh,
QObject *parent)
    : QObject(parent),
    frameIndex(0),
    fpsCounter(10000)
{
    videoCapture = cv::VideoCapture(index);

    if (!videoCapture.isOpened()) {
        qFatal("Failed to open a video stream #%d", index);
    }

    angl_pub = nh-
>advertise<std_msgs::Int16>("camera_360/angle", 1, true);
    cam_pub = nh-
>advertise<std_msgs::UInt8MultiArray>("camera_360/image", 1,
true);

    cam_pub_full = nh-
>advertise<std_msgs::UInt8MultiArray>("camera_360/image_360", 1,
true);

    sub_rotation_cam = nh-
>subscribe("camera_360/camera_control", 100,
&VideoCapturer::onCommandReceive, this);

    init();
}

void VideoCapturer::init()
{
    _frameSize =
cv::Size(videoCapture.get(cv::CAP_PROP_FRAME_WIDTH),
videoCapture.get(cv::CAP_PROP_FRAME_HEIGHT));

```

```

        _frameWidth = _frameSize.width;
        _frameHeight = _frameSize.height;

        if (_frameWidth == SMALL_WIDTH){
            _map_x = cv::imread(_pathToMap_x_1472,
cv::IMREAD_UNCHANGED);
            _map_x.convertTo(_map_x, CV_32FC1);

            _map_y = cv::imread(_pathToMap_y_1472,
cv::IMREAD_UNCHANGED);
            _map_y.convertTo(_map_y, CV_32FC1);
        }
        else{
            _map_x = cv::imread(_pathToMap_x_3006,
cv::IMREAD_UNCHANGED);
            _map_x.convertTo(_map_x, CV_32FC1);

            _map_y = cv::imread(_pathToMap_y_3006,
cv::IMREAD_UNCHANGED);
            _map_y.convertTo(_map_y, CV_32FC1);
        }

        _scale = _MAX_SCALE;
        _positionX = _frameWidth / 2;
        _positionY = _frameHeight / 2;

        setRects();

        auto app = QApplication::instance();
        connect(app, &QCoreApplication::aboutToQuit, this,
&VideoCapturer::releaseCam, Qt::BlockingQueuedConnection);

        connect(&fpsCounter, &FpsCounter::update,
            this, &VideoCapturer::fpsCounterUpdate);

```

```

        auto timer = new QTimer(this);

        connect(timer, &QTimer::timeout, this,
&VideoCapturer::loop);
        connect(app, &QCoreApplication::aboutToQuit, timer,
&QTimer::stop, Qt::BlockingQueuedConnection);

        timer->start(0);
    }

void VideoCapturer::releaseCam()
{
    ROS_INFO_STREAM("Release the camera");
    videoCapture.release();
}

void VideoCapturer::loop()
{
    videoCapture.read(capturedFrame);

    if(!capturedFrame.empty())
    {
        cv::remap(capturedFrame, capturedFrame, _map_x, _map_y,
cv::INTER_LINEAR, cv::BORDER_CONSTANT, cv::Scalar(0, 0, 0));
        setCurrentFrame();

        // publish the frame
        onFrameReceive();
        onFullFrameReceive();
        onAngleReceive();

        frameMutex.lock();
        capturedFrame.copyTo(frame);
        frameIndex++;
    }
}

```

```

        frameMutex.unlock();

        emit onFrame();
        fpsCounter.tick();
    }

}

void VideoCapturer::setRects()
{
    if (getRectX() < 0)
    {
        _rightRect = cv::Rect(0, getRectY(), getCurrentWidth() +
getRectX(), getCurrentHeight());
        _leftRect = cv::Rect(_frameWidth + getRectX(),
getRectY(), abs(getRectX()), getCurrentHeight());
        return;
    }

    if (getRectX() < _frameWidth - getCurrentWidth())
    {
        _rightRect = cv::Rect(getRectX(), getRectY(),
getCurrentWidth(), getCurrentHeight());
        _leftRect = cv::Rect();
        return;
    }

    if (getRectX() > _frameWidth - getCurrentWidth())
    {
        _rightRect = cv::Rect(0, getRectY(), getRectX() +
getCurrentWidth() - _frameWidth, getCurrentHeight());
        _leftRect = cv::Rect(getRectX(), getRectY(), _frameWidth
- getRectX(), getCurrentHeight());
        return;
    }
}

```

```

    }

    void VideoCapturer::setCurrentFrame()
    {
        this->setRects();
        capturedFrame.copyTo(_fullFrame);
        if (_leftRect.area())
        {
            cv::rectangle(_fullFrame, _leftRect, cv::Scalar(0, 0,
255), 8);
        }
        cv::rectangle(_fullFrame, _rightRect, cv::Scalar(0, 0,
255), 8);

        cv::resize(_fullFrame, _fullFrame, cv::Size(), 0.5,
0.5);

        if (!_leftRect.area())
            capturedFrame = capturedFrame(_rightRect);
        else
        {
            cv::hconcat(capturedFrame(_leftRect),
capturedFrame(_rightRect), capturedFrame);
        }
    }

    int VideoCapturer::getRectX()
    {
        return _positionX - getCurrentWidth() / 2;
    }

    int VideoCapturer::getRectY()
    {
        return _positionY - getCurrentHeight() / 2;
    }

```

```

int VideoCapturer::getCurrentWidth()
{
    return _INITIAL_WIDTH * _scale;
}

int VideoCapturer::getCurrentHeight()
{
    return _INITIAL_HEIGHT * _scale;
}

void VideoCapturer::onFrameReceive()
{
    std::vector<uchar> buff;
    std::vector<int> params;
    params.push_back(cv::IMWRITE_JPEG_QUALITY);
    params.push_back(_frame_squeeze_level);
    cv::imencode(".jpg", capturedFrame, buff, params);
    std_msgs::UInt8MultiArray msg;
    msg.data = buff;
    cam_pub.publish(msg);
}

```

Б.3 Текст файла ptzhumanfollower.h

```

#ifndef PTZHUMANFOLLOWER_H
#define PTZHUMANFOLLOWER_H

#include <opencv2/imgproc.hpp>

#include "yolodetector.h"
#include <vector>

#include <ros/ros.h>
#include <std_msgs/UInt8MultiArray.h>

```

```

#include <std_msgs/Bool.h>

class PtzHumanFollower
{
    // YOLO parameters
    static constexpr auto DetectorThreshold = 0.1;
    YoloDetector* yoloDetector;

    // cv params
    cv::Mat frame;
    std::vector<bbox_t> boxes;
    cv::Rect trackingRect;
    bool frameError = false;

    // ros params
    bool personIsDetected{false};
    std_msgs::Bool msg_output;
    ros::Publisher pub_hd;
    ros::Subscriber sub_cam_360;

public:
    explicit PtzHumanFollower();

private:
    void cameraCallback(const std_msgs::UInt8MultiArray&
msg);

};

#endif // PTZHUMANFOLLOWER_H

```

Б.4 Текст файла ptzhumanfollower.cpp

```
#include "ptzhumanfollower.h"

PtzHumanFollower::PtzHumanFollower()
{
    ros::NodeHandle nh;
    yoloDetector = new YoloDetector();
    yoloDetector->setThreshold(DetectorThreshold);

    pub_hd = nh.advertise<std_msgs::Bool>("person_detection", 1,
true);
    sub_cam_360 = nh.subscribe("camera_360/image_360", 1,
&PtzHumanFollower::cameraCallback, this);
}

void PtzHumanFollower::cameraCallback(const
std_msgs::UInt8MultiArray& msg)
{
    frame = cv::imdecode(msg.data, cv::IMREAD_COLOR);

    if (frame.empty())
    {
        if (frameError) return;

        ROS_ERROR("Couldn't decode input array");
        frameError = true;

        return;
    }

    frameError = false;
    unsigned int objectId = yoloDetector->getClassIds()["person"];
```

```

boxes = yoloDetector->detect(frame);
personIsDetected = false;

for (auto &box : boxes)
{
    int id = box.obj_id;

    if (id == yoloDetector->getClassIds()["person"])
    {
        personIsDetected = true;

        break;
    }
}

msg_output.data = personIsDetected;
pub_hd.publish(msg_output);
}

```

Б.5 Текст файла yolodetector.h

```

#ifndef YOLODETECTOR_H
#define YOLODETECTOR_H

#include <QObject>
#include <QStringList>
#include <QHash>
#include <QString>

#include "yolo_v2_class.hpp"

class YoloDetector : public QObject
{

```

```
Q_OBJECT
```

```
Detector detector;
QStringList classNames;
QHash<QString, int> classIds;
double threshold;
```

```
public:
```

```
static cv::Scalar colorHash(const QString &str);
```

```
explicit YoloDetector(const char *configFileName =
                      DARKNET_PATH "cfg/"
DARKNET_MODEL_NAME ".cfg",
                      const char *weightsFileName =
                      DARKNET_WEIGHTS_PATH
DARKNET_MODEL_NAME ".weights",
                      const char *namesFileName =
DARKNET_NAMES_PATH,
                      QObject *parent = nullptr);
```

```
inline QStringList
getClassNames() const
{
    return classNames;
}
```

```
inline QHash<QString, int>
getClassIds() const
```

```

    {
        return classIds;
    }

    inline double
    getThreshold() const
    {
        return threshold;
    }

    inline void
    setThreshold(double threshold)
    {
        this->threshold = threshold;
    }

    inline std::vector<bbox_t>
    detect(cv::Mat mat)
    {
        return detector.detect(mat, getThreshold());
    }

private:

    void loadClasses(const char *fileName);

};

#endif // YOLODETECTOR_H

```

Б.6 Текст файлу yolodetector.cpp

```

#include <QFile>
#include <QtGlobal>
#include <QTextStream>
#include <QDebug>

#include "yolodetector.h"

cv::Scalar YoloDetector::colorHash(const QString &str)
{
    int color = 0;

    for (int i = 0; i < str.size(); i++) {
        color ^= (str.at(i).unicode() << 8) & 0xFFFFFFFF;

        for (int j = 0; j < 8; j++) {
            if (color & 0x800000) {
                color = (color << 1) ^ 0x102110;
            } else {
                color <<= 1;
            }
        }
    }

    return cv::Scalar((color >> 16) & 0xFF, (color >> 8) &
0xFF, color & 0xFF);
}

YoloDetector::YoloDetector(const char *configFileName, const
char *weightsFileName,
                                const char *namesFileName, QObject
*parent)

```

```

        : QObject(parent),
          detector(configFileName, weightsFileName)
    {
        loadClasses(namesFileName);
    }

void YoloDetector::loadClasses(const char *fileName)
{
    classNames.clear();
    classIds.clear();

    QFile namesFile(fileName);

    if (!namesFile.open(QIODevice::ReadOnly |
QIODevice::Text)) {
        qFatal("Failed to load namesFile '%s'", fileName);
    }

    QTextStream textStream(&namesFile);
    int classId = 0;

    for (auto className = textStream.readLine();
        !className.isNull();
        className = textStream.readLine(), classId++) {

        classNames << className;
        classIds[className] = classId;
    }
}

```

Б.7 Текст файла ptzhightapplication.h

```

#ifndef PTZHIGHLIGHTAPPLICATION_H
#define PTZHIGHLIGHTAPPLICATION_H

```

```

#include <QObject>

#include "videocapturer.h"
#include "opencvmetatypes.h"
#include "threadcontainer.h"
#include "posixsignaladapter.h"
#include "roscopy.h"

class PtzHighlightApplication : public QObject
{
    Q_OBJECT

    OpenCvMetaTypes openCvMetaTypes;
    VideoCapturer *videoCapturer;
    ThreadContainer *videoCapturerContainer;

    ROSProxy rosProxy;
    PosixSignalAdapter signalAdapter;

    std::string camera_id;

public:
    explicit PtzHighlightApplication(int argc, char *argv[],
    QObject *parent = nullptr);
    ~PtzHighlightApplication();

private slots:

};

```

```
#endif // PTZHIGHLIGHTAPPLICATION_H
```

Б.8 Текст файла ptzhightapplication.cpp

```
#include <QCoreApplication>
#include <QDebug>
#include <QTimer>

#include "ptzhightapplication.h"

PtzHighlightApplication::PtzHighlightApplication(int argc,
char *argv[], QObject *parent)
    : QObject(parent),
      rosProxy(argc, argv, "camera_360_node")
{
    auto nh = rosProxy.getNodeHandle();
    ros::NodeHandle nh_private("~");

    nh_private.param<std::string>("camera_id", camera_id,
"/dev/camera360");
    const char * c = camera_id.c_str();

    videoCapturer = new VideoCapturer(c, nh);
    videoCapturerContainer = new
ThreadContainer(videoCapturer);
}

PtzHighlightApplication::~PtzHighlightApplication()
{
    ROS_INFO_STREAM("delete Camera_360_application");
}
```

```

        delete videoCapturer;
        delete videoCapturerContainer;
    }

```

Б.9 Текст файла `lamp_control.py`

```

import rospy
import Jetson.GPIO as GPIO
import threading
from std_msgs.msg import Bool

class LampControl:
    def __init__(self):
        self.main_power_pin = 32
        self.uv_lamp_pin = 21
        self.uv_recirculator_pin = 19
        self.main_relay_state = False
        self.uv_lamp_state = False
        self.uv_recirculator_state = False
        self.timer_state = False
        self.person_detection_counter = 0
        self.detection_control = Bool(True)

        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(self.uv_recirculator_pin, GPIO.OUT,
initial=GPIO.HIGH)
        GPIO.setup(self.uv_lamp_pin, GPIO.OUT,
initial=GPIO.HIGH)
        GPIO.setup(self.main_power_pin, GPIO.OUT)

        self.lamp_subscriber = rospy.Subscriber(
            "uv_lamp", Bool, self.uv_lamp)
        self.recirculator_subscriber = rospy.Subscriber(

```

```

        "uv_recirculator", Bool, self.uv_recirculator)
self.person_detection_subscriber = rospy.Subscriber(
    "person_detection", Bool, self.person_detection)

def uv_lamp(self, msg):
    rospy.loginfo(msg.data)
    if msg.data:
        if not self.main_relay_state:
            self.main_relay_control(True)

        if self.timer_state:
            self.timer_state = False

        GPIO.output(self.uv_lamp_pin, GPIO.LOW)
        self.uv_lamp_state = True
    else:
        if not self.uv_recirculator_state:
            self.main_relay_control(False)

        GPIO.output(self.uv_lamp_pin, GPIO.HIGH)
        self.uv_lamp_state = False

def uv_recirculator(self, msg):
    if msg.data:
        if not self.main_relay_state:
            self.main_relay_control(True)

        GPIO.output(self.uv_recirculator_pin, GPIO.LOW)
        self.uv_recirculator_state = True
    else:
        if not self.uv_lamp_state:
            self.main_relay_control(False)

        GPIO.output(self.uv_recirculator_pin, GPIO.HIGH)
        self.uv_recirculator_state = False

```

```

def main_relay_control(self, command):
    if command:
        GPIO.output(self.main_power_pin, GPIO.HIGH)
        self.main_relay_state = True
    else:
        GPIO.output(self.main_power_pin, GPIO.LOW)
        self.main_relay_state = False

def person_detection(self, detection):
    if detection.data:
        self.person_detection_counter += 1
        if self.person_detection_counter > 3 and
self.uv_lamp_state:
            self.uv_lamp(Bool(data=False))
            self.person_detection_counter = 0
            if not self.timer_state:
                t = threading.Timer(60.0,
self.uv_lamp, [Bool(data=True)])
                t.start()
                self.timer_state = True
            elif self.person_detection_counter > 3:
                self.person_detection_counter = 0

if __name__ == '__main__':
    rospy.init_node('lamp_control_node')
    LampControl()
    rospy.spin()

```

ДОДАТОК В
Слайди презентації

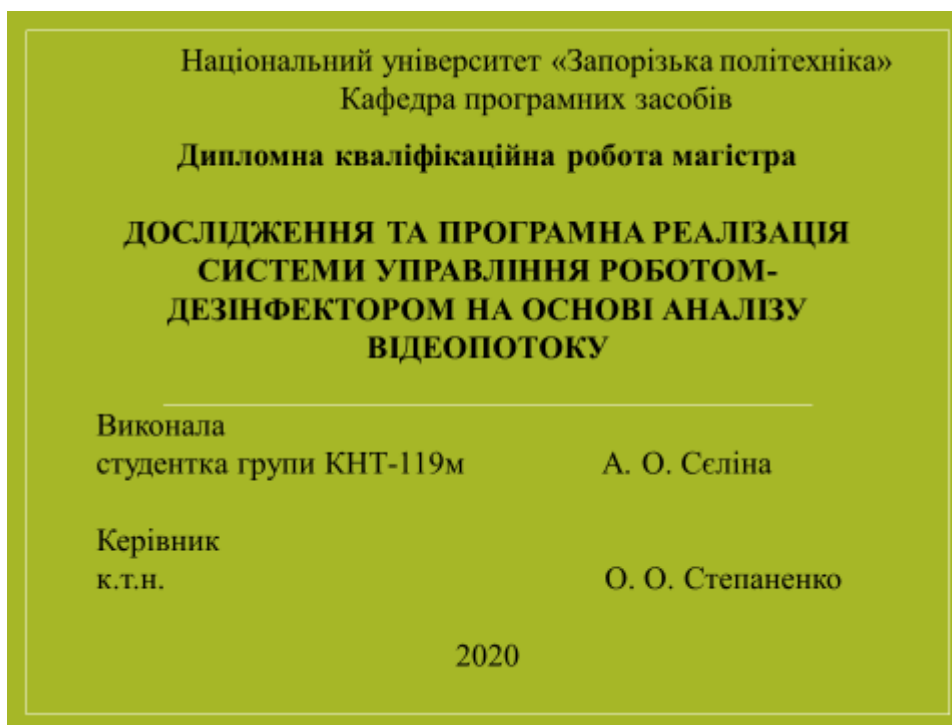


Рисунок В.1 – Слайд 1

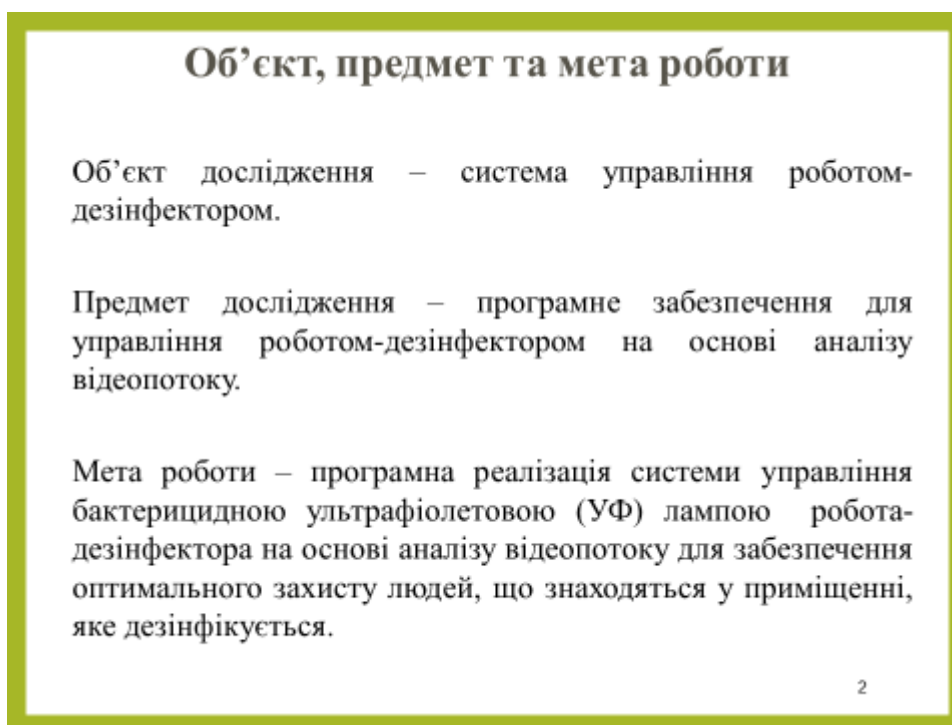


Рисунок В.2 – Слайд 2

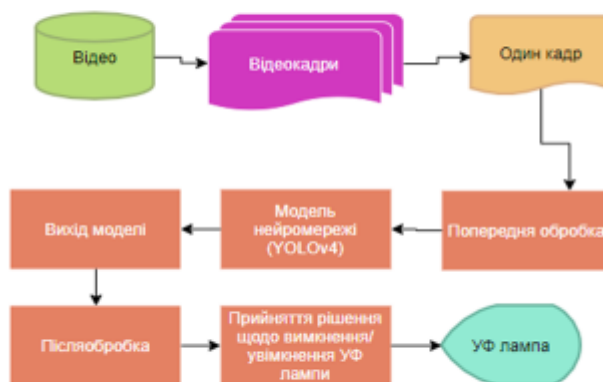
Завдання

- ✓ дослідження методів аналізу відеокадрів;
- ✓ проектування структури програмного забезпечення;
- ✓ розробка програмного забезпечення;
- ✓ тестування та відлагодження програми;
- ✓ економічне обґрунтування доцільності розробки та забезпечення засходів охорони праці та безпеки в надзвичайних ситуаціях.

3

Рисунок В.3 – Слайд 3

Система керування УФ лампою



4

Рисунок В.4 – Слайд 4



Рисунок В.5 – Слайд 5



Рисунок В.6 – Слайд 6

Порівняння мов програмування

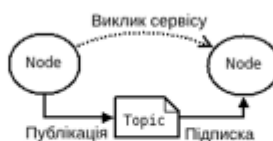
Критерій	Python	C++	Java
Об'єктно-орієнтована парадигма програмування	+	+	+
Безкоштовне рішення	+	+	+
Багатопотоковість	+	+	+
Робота з обраними інструментами (OpenCV, YOLOv4, Darknet)	+	+	+/-
Більш висока швидкість виконання	-	+	-

7

Рисунок В.7 – Слайд 7

Структура системи на базі ROS

Основними концепціями ROS є вузли (nodes), повідомлення (messages), теми (topics). Архітектурно всі обчислювальні задачі виконуються у вузлах ROS, які обмінюються між собою повідомленнями. Ці повідомлення вузлами публікуються в темах, які поділяють їх на групи інтересів. Коли деякому вузлу необхідно отримувати повідомлення з певними даними, цей вузол підписується на певну тему.



8

Рисунок В.8 – Слайд 8

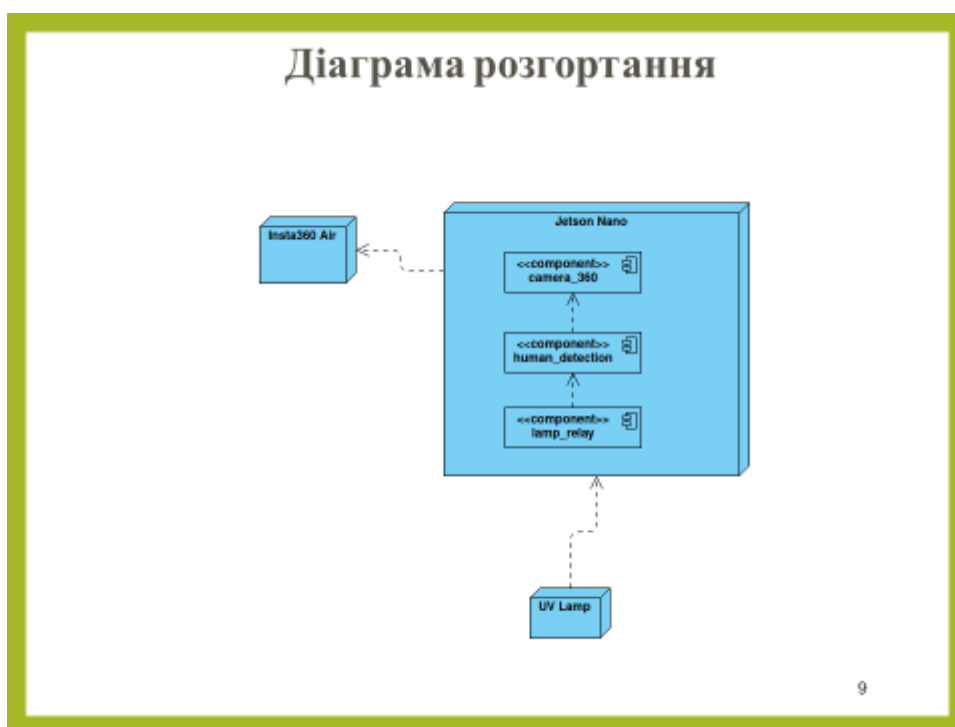


Рисунок В.9 – Слайд 9



Рисунок В.10 – Слайд 10



Рисунок В.11 – Слайд 11

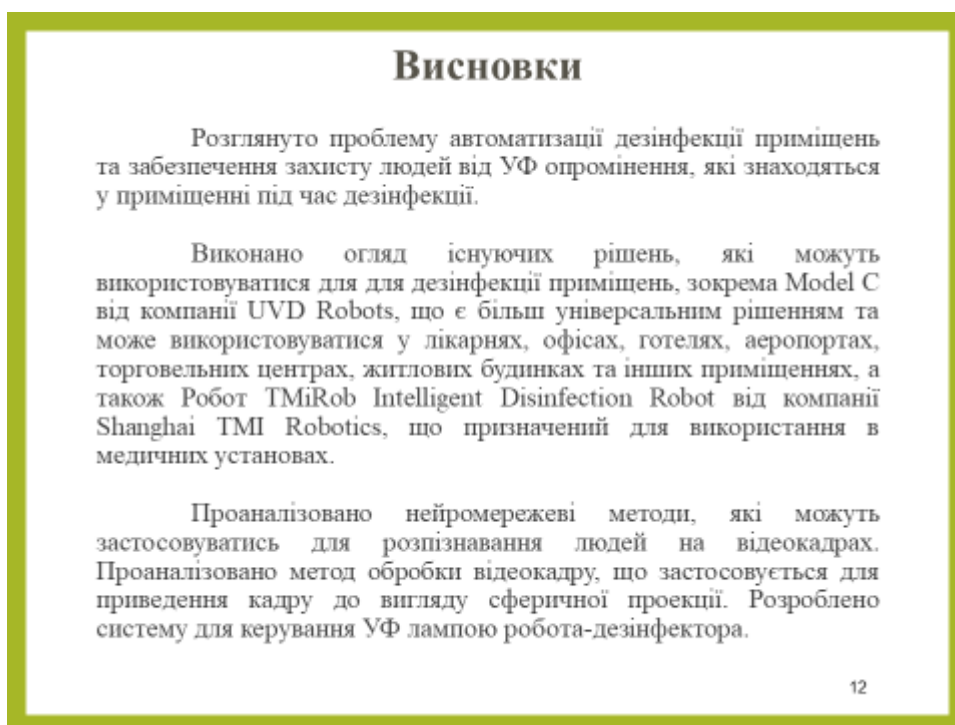


Рисунок В.12 – Слайд 12

Висновки

Визначено функції, які повинна виконувати програма. Створено діаграму прецедентів для моделювання функціональних вимог до роботи системи. Обрано мову програмування C++ для реалізації програми. В якості фреймворку для розробки програмного забезпечення роботів обрано ROS. Бібліотекою для обробки зображень обрано OpenCV.

Реалізовано програмне забезпечення керування УФ лампою робота-дезінфектора, описано розгортання програмного забезпечення, описано схему взаємодії між вузлами системи та принципи реалізації взаємодії між ними на основі моделі Публікація-підписка, описано основні реалізовані класи.

Усі завдання, поставлені на роботу, виконано.

13

Рисунок В.13 – Слайд 13

Дякую за увагу!

14

Рисунок В.14 – Слайд 14