МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА МОБІЛЬНОГО ДОДАТКА СИСТЕМИ ВІДЕОСПОСТЕРЕЖЕННЯ НА БАЗІ NODEMCU»**

Виконав: студент 4 курсу, групи 6.1219-1пі

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми програмна інженерія
(назва освітньої програми)

О.О. Антонов
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.

(підпис)

« 07 » 02 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Антонову Олексію Олександровичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проєкту) Розробка мобільного додатка системи відеоспостереження на базі NodeMcu

керівник роботи (проєкту) Мухін Віталій Вікторович, к.т.н., доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Постановка задачі, аналіз предметної області.
2. Основні теоретичні відомості.
3. Розробка мобільного додатку та управління камерою на базі NodeMcu.

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

презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 07.02.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	10.02.2023	
2.	Збір вихідних даних.	03.03.2023	
3.	Обробка методичних та теоретичних	24.03.2023	
	джерел.		
4.	Розробка першого розділу.	07.04.2023	
5.	Розробка другого розділу.	21.04.2023	
6.	Розробка третього розділу	15.05.2023	
7.	Оформлення та нормоконтроль		
	кваліфікаційної роботи.	01.06.2023	
8.	Захист кваліфікаційної роботи.	21.06.2023	

Студент _____ О.О. Антонов
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Мухін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка мобільного додатка системи відеоспостереження на базі NodeMcu»: 64 с., 62 рис., 24 джерела, 2 додатки.

АРДУЇНО, АНДРОЇД, ВІДЕОСПОСТЕРЕЖЕННЯ, ІНТЕРФЕЙС, МОБІЛЬНИЙ ДОДАТОК, СМАРТФОН.

Об’єкт дослідження: існуючі мобільні додатки організації відеоспостереження.

Мета роботи: розробити мобільний додаток системи відеоспостереження за камерами на основі NodeMcu.

Метод дослідження: аналітичний, порівняльний.

Відеоспостереження є системою, яка передає інформацію з вебкамер або ір-камер на обмежену кількість моніторів чи записувальних пристроїв. У відмінність від телевізійного мовлення, відеоспостереження передає зображення на обмежену кількість приймачів, що означає, що воно функціонує в закритому режимі і вимагає певних прав доступу для підключення. Зараз системи спостереження стали невід’ємною частиною охорони як публічних комерційних будівель, так і приватних територій. У кваліфікаційній роботі порівнюються існуючі мобільні додатки організації відеоспостереження та розробляється свій додаток для керування відеоспостереженням за камерою на основі плати NodeMcu.

SUMMARY

Bachelor’s qualification paper «Development of a Mobile Application for a Video Surveillance System Based on NodeMcu»: 64 pages, 62 figures, 24 references, 2 supplements.

ARDUINO, ANDROID, VIDEO SURVEILLANCE, INTERFACE, MOBILE APP, SMARTPHONE.

The object of the study is existing mobile applications of the video surveillance organization.

The purpose of the work is to develop a mobile application of a video surveillance system for cameras based on NodeMcu.

Research method: analytical, comparative.

Video surveillance is a system that transmits information from video cameras or television cameras to a limited number of monitors or recording devices. Unlike television broadcasting, video surveillance transmits images to a limited number of receivers, which means that it functions in closed mode and requires certain access rights to connect. Nowadays, surveillance systems have become an integral part of the protection of both public commercial buildings and private areas. The qualifying work compares the existing mobile applications of the video surveillance organization and develops its own application for controlling the video surveillance of the camera based on the NodeMcu board.

ЗМІСТ

Завдання на кваліфікаційну роботу 2

Реферат 4

Summary 5

Вступ 7

1 Аналіз предметної області та вимоги 8

1.1 Теоретичний аналіз аналогів 8

1.2 Призначення та цілі створення 10

1.3 Вимоги 10

1.4 Програмні засоби 11

1.5 Діаграма прецентентів 13

1.6 Діаграма послідовності 14

1.7 Діаграма компонентів 15

2 Налаштування, характеристика і опис необхідних елементів arduino 17

2.1 Характеристика та опис плати nodemcu 17

2.2 Характеристика та опис камери для плати nodemcu 21

2.3 Налаштування підключення плати 22

2.4 Підключення плати до камери 30

Реалізація 31

3.1 Розробка додатку управління камерою відеоспостереження 31

3.2 Опис та взаємодія зі скомпільованим додатком в телефоні 48

Висновки 55

Перелік посилань 56

Додаток А Скетч для підключення 58

Додаток Б Код android-додатку 59

ВСТУП

У наш час неможливо уявити навколишній світ без впливу на нього технологічної революції. Вона торкнулася як спеціалізованих, так і повсякденних сфер життя людей. Майже у кожному пристрої зараз вбудована камера, мікрофон чи ще будь-який датчик для отримання та обробки інформації з навколишнього середовища. Це можуть бути як і звичні нам смартфони, так і датчики руху, або пожежної безпеки, веб-камери, або навіть дослідницькі спеціалізовані дрони й інше обладнання. Всюди знайшлося місце для імплементації технологій.

Не обійшлося і у сфері персональної, або і корпоративної безпеки. Веб-камери, датчики руху, освітлення, температури – все це допомагає контролювати безпеку приміщення від широкого спектру загроз – від пограбування до появи цвілі. Стосовно персональної безпеки – використання різноманітного спектру камер допомагає контролювати стан безпеки приватної території, або ж навіть шпигувати за сусідами.

Але недостатньо просто купити та змонтувати комплект камер на найбільш вдалі для цього місця – необхідно якось і організувати підключення відео з них у певний хаб для використання їх в якості відеоспостереження а не просто обманки.

У кваліфікаційній роботі розглянуті вимоги до додатків системи відеоспостереження та розроблено свій аналог мобільного додатку на мові програмування Java з дотриманням основних вимог. Додаток має мати основний функціонал, необхідний користувачу, а також простий інтерфейс для підключення та використання у побуті для пересічного юзера.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГИ

1.1 Теоретичний аналіз аналогів

Для початку давайте розглянемо популярних конкурентів в сфері надання послуг відеоспостереження або віддаленого доступу. Зараз доволі багато компаній пропонують свої рішення як на комерційній основі так і на частково комерційній, як наприклад:

- AlfredCamera з більш ніж 10 млн завантажень в Google Play [1];
- популярний мобільний додаток з керування ip або web камерами ZoomOn [2];
- система відеоспостереження Faceter [3];
- додаток для спостереження за ip камерами Little Stars [4];

Порівняємо обрані рішення:

AlfredCamera [1]

Розробка – Alfred Systems Inc.

Плюси:

- є реєстрація через профіль Google, що додає простоти в користуванні;
- моживість керувати детекторами руху або звуку, якщо вони є на камері;
- маштабування зображення камери;
- функція фільтру слабого освітлення для місць з недостатком світла або вночі;
- можливість з додатку використати динаміки камери або інших підключених пристроїв;

- приємний, простий інтерфейс;
- підключення через qr-код або за допомогою однакового профілю на пристроях у заздалегідь завантаженому додатку AlfredCamera;
- мультиплатформа.

Мінуси:

- сервіси Google можуть збирати деякі метадані;
- є платні тарифи, реклама.

ZoomOn [2]

Розробка – Master App Solutions.

Плюси:

- проста реєстрація;
- біль-менш зрозумілий інтерфейс;
- підключення через qr-код або згенерований тимчасовий ключ;
- функція фільтру слабого освітлення для місць з недостатком світла або вночі;
- можливість з додатку використати динаміки камери або інших підключених пристроїв;
- мультиплатформа.

Мінуси:

- безкоштовний функціонал лише 15 хвилин, потім треба купувати;
- кожен раз необхідно вводити код при вході у профіль з email, що доволі муторно.

Faceter [3]

Розробка – Faceter South Africa (PTY) LTD.

Плюси:

- простий та зручний інтерфейс;

- підключення камер по qr-коду або посиланням;
- мультиплатформа;
- можливість запису відео.

Мінуси:

- 7 днів пробний період, потім підписка.

Little Stars [4]

Розробка – JXL.

Плюси:

- простий інтерфейс;
- підключення ip-камер по qr-коду або посиланням;
- безкоштовний додаток;
- можливість запису відео.

Мінуси:

- підключення лише у локальній мережі;
- відсутність профілю або чи авторизації.

1.2 Призначення та цілі створення

Функціональне призначення – у результаті аналізу та згідно поставленої задачі результатом кваліфікаційної роботи має бути розроблений мобільний додаток системи відеоспостереження камерами на основі плати NodeMCU.

Експлуатаційне призначення системи – у результаті реалізації буде отримано мобільний додаток для спостереження. З використанням такого стеку технологій як: Arduino, Java, Android. Ця робота, може стати гарною структурою для майбутньої розробки комерційного проекту. Або ж прикладом для розробників схожої системи.

Мета створення системи – розробити мобільний додаток відеодоступу за камерами на основі плати NodeMCU. Та на основі цього отримати знання та навички роботи з Arduino та Android.

1.3 Вимоги

Додаток не потребує спеціальних типів користувачів, лише звичайного юзера. Тож розглянемо основні вимоги від додатку для користувача системою відеоспостереження:

- швидка реєстрація свого профілю, або навіть його відсутність та коннект по згенерованому тимчасовому коду, або тимчасовому посиланню;
- можливість підключення однієї або декількох камер, можливо різних виробників та типів.

Серед технічних вимог які застосовуються до майбутнього веб додатку, можна виділити:

- простота використання;
- захищеність;
- швидкість передачі сигналу.

1.4 Програмні засоби

Із теми роботи, повністю зрозуміло, що нам буде необхідно попрацювати як з Android і мовою програмування Kotlin для створення мобільного додатку, так і з Arduino й відповідно мовою Arduino C для створення скетчу управління камерою. Тож, давайте розглянемо ці технології ближче.

Android – платформа і операційна система, що базується на ядрі Linux. Створена корпорацією Google для планшетів та смартфонів. Хоч Android і заснований на Linux, він стоїть дещо осторонь свого батька. Основним елементом цієї ОС є реалізація Dalvik віртуальної машини Java. Все програмне забезпечення та застосування засновані на цій реалізації [5].

Arduino – це апаратна платформа, призначена для любителів конструювання, що включає в себе плату мікроконтролера з введенням/виведенням і середовище розробки на основі мови програмування Processing/Wiring, що є спрощеною версією C/C++. Ардуїно може використовуватися для створення інтерактивних автономних об'єктів або підключатися до комп'ютерних програм (наприклад, Max/MSP, Adobe Flash, Pure Data, Processing). Інформація про плату (схема друкованої плати, специфікації компонентів, програмне забезпечення) доступна для громадськості і може бути використана тими, хто бажає створювати власні плати самостійно [6].

Arduino C – версія мови програмування C/C++, яка використовується для програмування мікроконтролерів Arduino. Ця мова надає зручний високорівневий інтерфейс для взаємодії з апаратними можливостями Arduino, зокрема для початківців та непрофесійних користувачів. Вона включає в себе набір бібліотек і функцій, які спрощують взаємодію з пінами введення/виведення, а також з різноманітними апаратними пристроями, такими як дисплеї, датчики та мотори. Основними функціями Arduino C є «setup()», яка виконується один раз при запуску або перезавантаженні плати Arduino, і використовується для ініціалізації змінних, налаштування режимів пінів та конфігурації необхідних параметрів. Другою важливою функцією є «loop()», яка виконується в безкінечному циклі і дозволяє програмі виконувати основну логіку, включаючи повторювані операції та зчитування вхідних даних [7].

Java – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009

року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано [8].

Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розроблення об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування [8].

1.5 Діаграма прецентентів

Діаграма прецедентів демонструє взаємодію між прецендентами і акторами [9]. На рисунку 1.1 зображена діаграма прецендентів мобільного додатку.

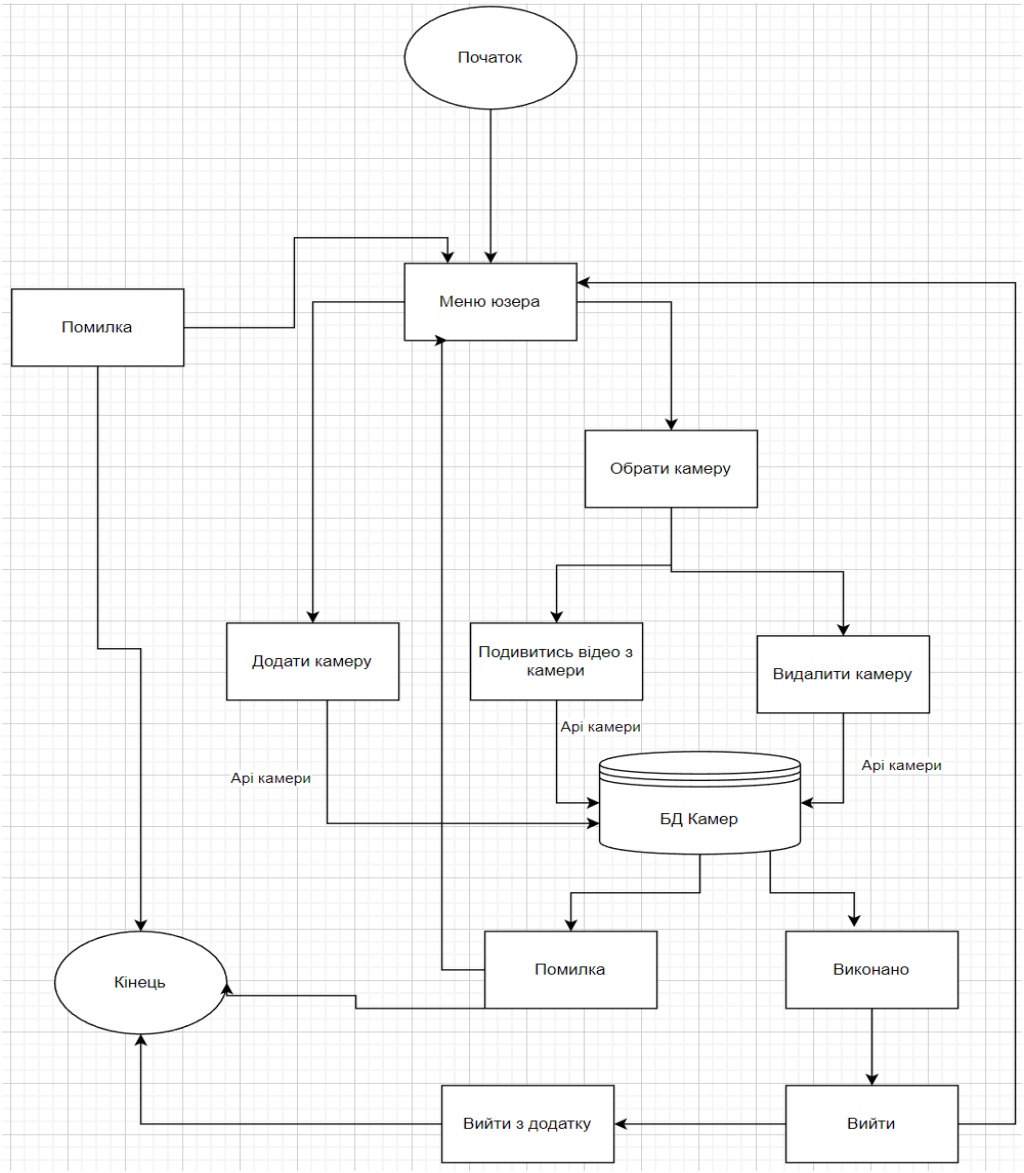


Рисунок 1.1 – Діаграма прецендентів

Вона використовує 2 основних елементи [9]:

- Actor (учасник) – множина логічно пов’язаних ролей, виконуваних при взаємодії з прецедентами або сутностями (система, підсистема або клас). Учасником може бути людина, роль людини в системі чи інша система, підсистема або клас, які представляють щось поза сутністю [9];
- Use case (прецедент) – опис окремого аспекту поведінки системи з точки зору користувача. Прецедент не показує, «як» досягається певний результат, а тільки «що» саме виконується [9];

1.6 Діаграма послідовності

На рисунку 1.2 зображена діаграма послідовності на прикладі використання юзером додатку для перегляду відео з камер.

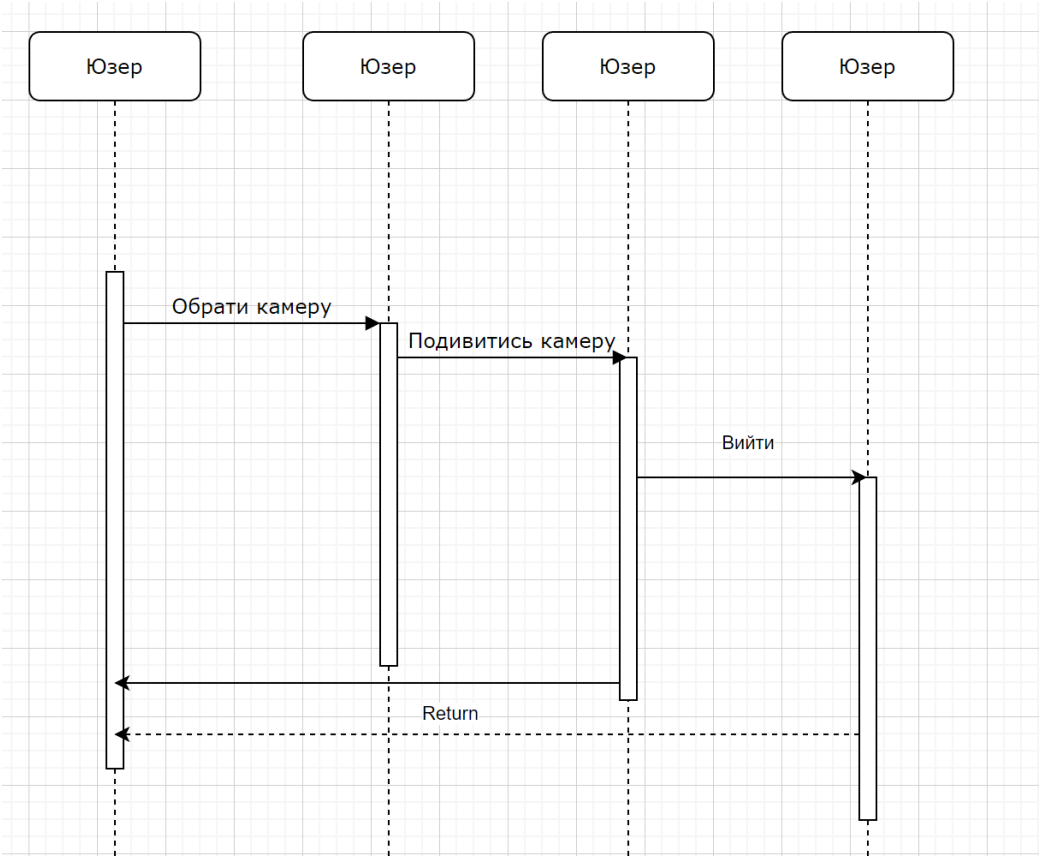


Рисунок 1.2 – Діаграма послідовності

Ця діаграма використовується для уточнення діаграм прецедентів – описує поведінкові аспекти системи. Діаграма послідовності відображає взаємодію об'єктів в динаміці, в часі. При цьому інформація набуває вигляду повідомлень, а взаємодія об'єктів передбачає обмін цими повідомленнями в рамках сценарію [9].

1.7 Діаграма компонентів

Компонентна діаграма UML – це один із типів діаграм, які можна знайти в діаграмах UML. Він здатний допомогти користувачам зрозуміти структуру конкретної системи [10].

На рисунку 1.3 зображена діаграма компонентів мобільного додатку.

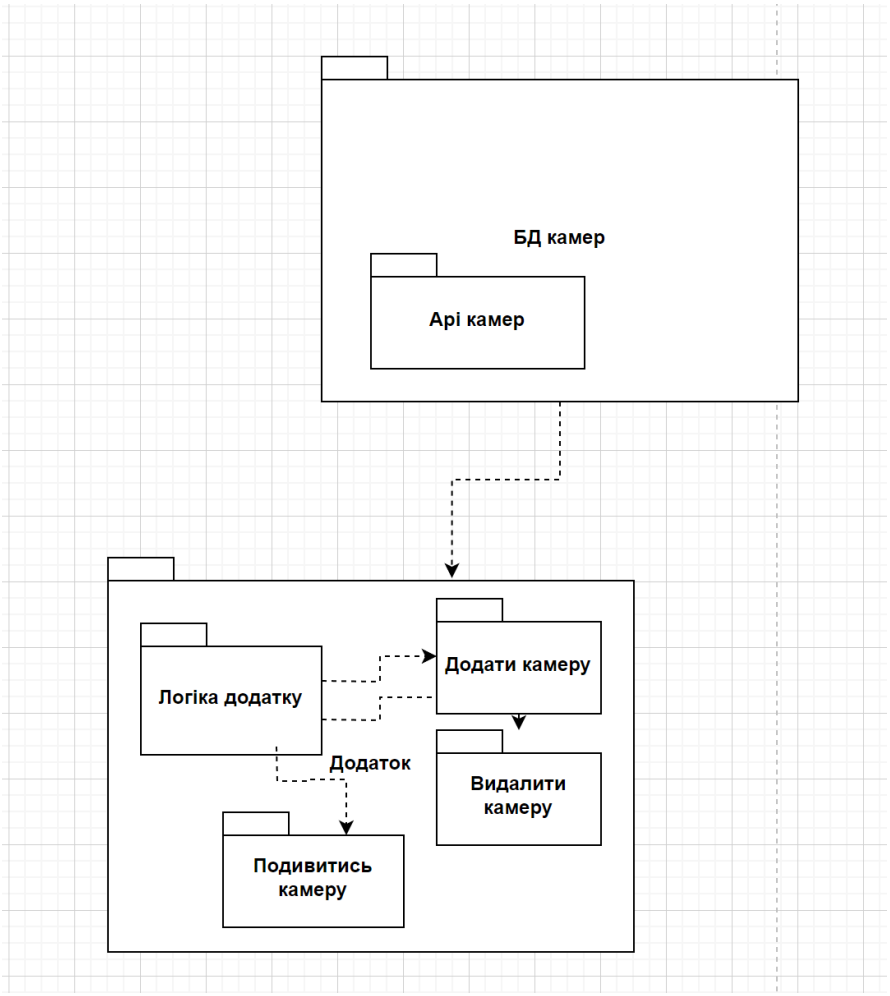


Рисунок 1.3 – Діаграма компонентів

Діаграми компонентів UML надають концептуальну картину взаємодії між різними системами. Можуть бути присутні як аспекти логічного, так і фізичного моделювання. Крім того, компоненти автономні. Це модульний системний елемент в UML, який можна замінити на альтернативні. Вони містять конструкції будь-якої складності та є самодостатніми. Лише через інтерфейси вкладені частини взаємодіють з іншими компонентами. Крім того, компоненти мають свої інтерфейси, але вони також можуть отримати доступ до операцій і служб інших компонентів за допомогою їхніх інтерфейсів. На

діаграмі компонентів інтерфейси також показують зв’язки та залежності в архітектурі програмного забезпечення [10].

2 НАЛАШТУВАННЯ, ХАРАКТЕРИСТИКА І ОПИС НЕОБХІДНИХ ЕЛЕМЕНТІВ ARDUINO

2.1 Характеристика та опис плати NodeMCU

NodeMCU (Node MicroController Unit) – це середовище розробки, яке поєднує в собі відкритий програмний код та апаратне забезпечення і засноване на недорогому системно-на-кристалі (SoC) з назвою ESP8266. ESP8266, розроблений і виготовлений компанією Espressif Systems, містить в собі ключові компоненти комп'ютера: центральний процесор (CPU), оперативну пам'ять (RAM), мережу (WiFi), а також сучасну операційну систему та набір засобів розробки (SDK). Це робить його відмінним вибором для проектів Інтернету речей (IoT) будь-якого виду [11].

Проте, як чіп, ESP8266 також важко отримати доступ і використовувати. Вам потрібно паяти проводи з відповідною аналоговою напругою до його контактів, навіть для найпростіших завдань, таких як включення живлення або відправлення натискання клавіші "комп'ютеру" на чіпі. Ви також повинні програмувати його на низькорівневих машинних інструкціях, які можуть інтерпретуватися апаратним забезпеченням чіпа. Цей рівень інтеграції не є проблемою при використанні ESP8266 як вбудованого контролера чіпа в масово вироблених електронних пристроях. Проте це велике завдання для ентузіастів, хакерів або студентів, які хочуть експериментувати з ним у своїх власних проектах IoT [11].

Проект Arduino створив апаратну модель з відкритим кодом та програмний набір інструментів для свого універсального контролера IoT. Аналогічно до NodeMCU, апаратне забезпечення Arduino – це мікроконтролерна плата з USB-роз'ємом, світлодіодними індикаторами та стандартними контактами для даних. Вона також визначає стандартні інтерфейси для взаємодії з датчиками та іншими платами. Проте, на відміну

від NodeMCU, до плати Arduino можуть бути використані різні типи процесорних чіпів (зазвичай ARM або Intel x86) з чіпами пам'яті та різноманітними середовищами програмування. Існує також посилання на дизайн Arduino для чіпа ESP8266. Однак гнучкість Arduino також означає значні відмінності між різними виробниками. Наприклад, більшість плат Arduino не мають можливості WiFi, а деякі навіть мають послідовний порт для обміну даними замість USB-порту [11].

Отже, за результатом порівняння технічних специфікацій (див. рис. 2.1–2.2) у використанні в кваліфікаційній роботі було обрано плату LoLin NodeMCU [11]. Розглянемо її конструкцію більш детально.

	Official NodeMCU	NodeMCU Carrier Board	LoLin NodeMCU
Microcontroller	ESP-8266 32-bit	ESP-8266 32-bit	ESP-8266 32-bit
NodeMCU Model	Amica	**Amica**	Clone LoLin
NodeMCU Size	49mm x 26mm	49mm x 26mm	58mm x 32mm
Carrier Board Size	n/a	102mm x 51mm	n/a
Pin Spacing	**0.9" (22.86mm)**	**0.9" (22.86mm)**	1.1" (27.94mm)
Clock Speed	80 MHz	80 MHz	80 MHz
USB **to Serial**	CP2102	**CP2102**	CH340G
USB Connector	Micro USB	Micro USB	Micro USB
Operating Voltage	3.3V	3.3V	3.3V
Input Voltage	4.5V-10V	4.5V-10V	4.5V-10V
Flash Memory/SRAM	4 MB / 64 KB	4 MB / 64 KB	4 MB / 64 KB
Digital I/O Pins	11	11	11
Analog In Pins	1	1	1
ADC Range	0-3.3V	0-3.3V	0-3.3V
UART/SPI/I2C	1 / 1 / 1	1 / 1 / 1	1 / 1 / 1
WiFi Built-In	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n
Temperature Range	-40C - 125C	-40C - 125C	-40C - 125C
Product Link		NodeMCU	NodeMCU

Рисунок 2.1 – Технічні специфікації основних плат NodeMCU [11]

Живлення(Power) – на платі знаходиться чотири електричні роз'єми: роз'єм VIN та три роз'єми 3,3 В. Якщо у вас є стабілізоване джерело живлення 5 В, ви можете використати роз'єм VIN для живлення ESP8266 та його

периферійних пристроїв. Роз'єми 3,3 В є виходами вбудованого регулятора напруги і можуть бути використані для живлення зовнішніх компонентів [11].

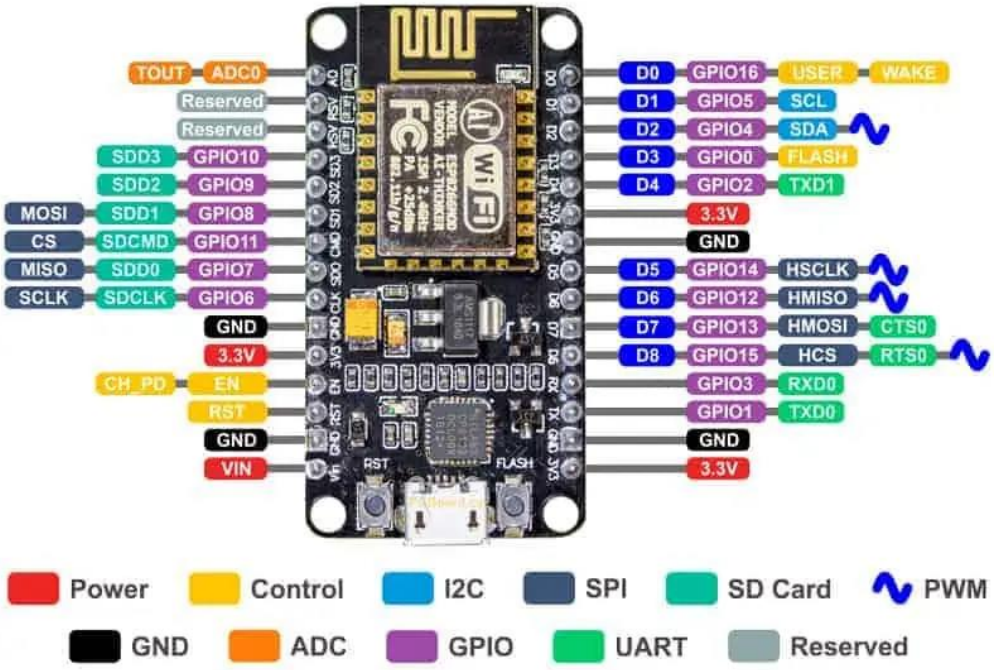


Рисунок 2.2 – Технічні специфікації плати LoLin NodeMCU [11]

З'єднання землі(GND) – з'єднання використовується для заземлення плати розвитку ESP8266 NodeMCU [11].

I2C-з'єднання – використовуються для підключення різноманітних сенсорів та периферійних пристроїв до вашого проекту через шину I2C. Підтримуються як режим мастера, так і режим підлеглого. I2C-комунікація може бути програмно керованою з максимальною швидкістю тактування 100 кГц. Важливо, щоб частота тактування I2C була вищою за найнижчу частоту тактування пристроїв-підлеглих [11].

GPIO-з'єднання – плата ESP8266 NodeMCU має 17 GPIO-з'єднань, які можуть бути програмно налаштовані для різних функцій, таких як I2C, I2S, UART, PWM, інфрачервона віддалена, LED-дисплей та призначення кнопок. Кожне GPIO-з'єднання може працювати з внутрішнім підтягуючим резистором до землі або до живлення, або в режимі високої імпедансності.

Якщо воно налаштоване як вхід для генерації переривань, то може реагувати на спадні або зростаючі фронти [11].

ADC-з'єднання – це з'єднання надає сигнал до вбудованого 10-бітного АЦП SAR NodeMCU. Цей АЦП дозволяє виконувати дві функції: моніторинг напруги живлення на позначці VDD3P3 та моніторинг вхідної напруги на позначці TOUT (але не одночасно) [11].

UART-з'єднання – плата ESP8266 NodeMCU має два UART-інтерфейси, UART0 та UART1, які дозволяють асинхронний зв'язок (RS232 і RS485) та підтримують швидкості передачі даних до 4,5 Мбіт/с. Для комунікації можна використовувати UART0 (позначки TXD0, RXD0, RST0 і CTS0), який підтримує контроль потоку. UART1 (позначка TXD1) підтримує тільки передачу даних і зазвичай використовується для журналювання подій [11].

SPI-з'єднання – ESP8266 має два інтерфейси SPI, SPI та HSPI, які можуть працювати в режимі підлеглого або мастра. Ці SPI-інтерфейси також підтримують такі функції [11]:

- 4 режими передачі SPI;
- швидкість тактування до 80 МГц, отримана від джерела тактування 80 МГц;
- буфер FIFO до 64 байтів.

SDIO-з'єднання – ESP8266 має безпечний цифровий інтерфейс введення-виведення (SDIO, Secure Digital Input/Output Interface), який використовується для прямого підключення SD-карт. Підтримуються SDIO v1.1 з шириною каналу 4 біти при частоті 25 МГц та SDIO v2.0 з шириною каналу 4 біти при частоті 50 МГц [11].

PWM-з'єднання – на платі присутні 4 канали ШИМ. Вихід ШИМ може бути програмно керованим для управління двигунами та світлодіодами. Діапазон частоти ШИМ може бути встановлений від 1000 мкс до 10000 мкс, що відповідає частоті від 100 Гц до 1 кГц [11].

Керувальні з'єднання(Recerved) – використовуються для управління ESP8266. Вони включають роз'єм EN для увімкнення та вимкнення чіпу, роз'єм RST для скидання чіпу та роз'єм WAKE для прокидання чіпу з режиму глибокого сну [11].

2.2 Характеристика та опис камери для плати NodeMCU

Для виконання нашої задачі необхідна камера, сумісна з платами Arduino NodeMCU, наприклад модель під назвою PTC06 [12]. Розглянемо її характеристики більш детально (див. рис. 2.3).

- Module size: 20mm x 28mm
- Module Weight: 3g
- Image sensor: CMOS 1/4 inch
- CMOS Pixels: 30M
- Pixel size: 5.6um*5.6um
- Manual focus adjustable from 5 to 15meters
- Output format: Standard JPEG
- White balance: Automatic
- Exposure: Automatic
- Gain: Automatic
- Shutter: Electronic rolling shutter
- SNR: 45DB
- Dynamic Range: 60DB
- Max analog gain: 16DB
- Frame speed: 640*480 30fps
- Scan mode: Progressive scan
- Viewing angle: 60 degrees
- Image size: VGA(640*480), QVGA(320*240), QQVGA(160*120)
- Baud rate: Default 38400
- Current draw: 75mA
- Operating voltage: DC +3-5V (the power pin is marked 3.3V but it goes into a 3-5V input regulator so you can power it with 3-5VDC)
- Communication: 3.3V TTL (Three wire TX, RX, GND)

Рисунок 2.3 – Технічні специфікації камери PTC06 [12]

Також, розглянемо виводи для підключення камери до плати (див. рис. 2.4):

Pin Description

Pin	Description
GND	Power Ground
RXD	Data Receive (3.3V TTL Level)
TXD	Data Transmit (3.3V TTL Level)
VCC	Power 5V DC
CVBS	Analog Video output

Рисунок 2.4 – Специфікації виводів камери PTC06 [13]

З'єднання землі(GND) – з'єднання використовується для заземлення [13].
RXD(Receive Data) – використовуєтья для прийому даних [13].
TXD(Transmit Data) – використовується для виводу даних [13].
VCC(Voltage Collector Collector) – вивод для енергозабезпечення [13].
CVBS(Color, Video, Blank, Sync) – аналоговий вивод сигналу [13].

Отже, ця камера сумісна з платою Lolin NodeMCU [11] та відповідає необхідним характеристикам для виконання задачі.

2.3 Налаштування підключення плати

Лише основні з плат Arduino не потребують додаткових драйверів та налаштуваннь для корректної роботи та запрограмування. Плата, використана у цій кваліфікаційній роботі не входить до списку основних, тож, для правильної роботи будуть необхідні додаткові маніпуляції.

Для початку, необхідно перейти на офіційний сайт Arduino [14] та обрати кнопку Software (див. рис. 2.5).

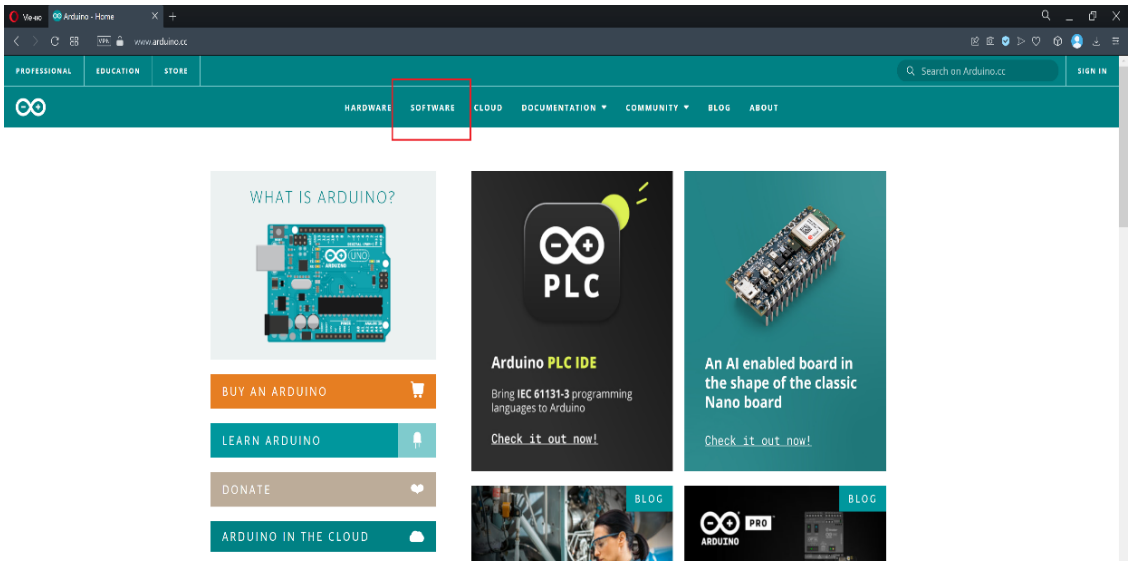


Рисунок 2.5 – Кнопка Software

Перейшовши на вкладку Software, необхідно обрати інсталятор Arduino IDE. Можна як .exe, так і портативною версією .zip (див. рис. 2.6):

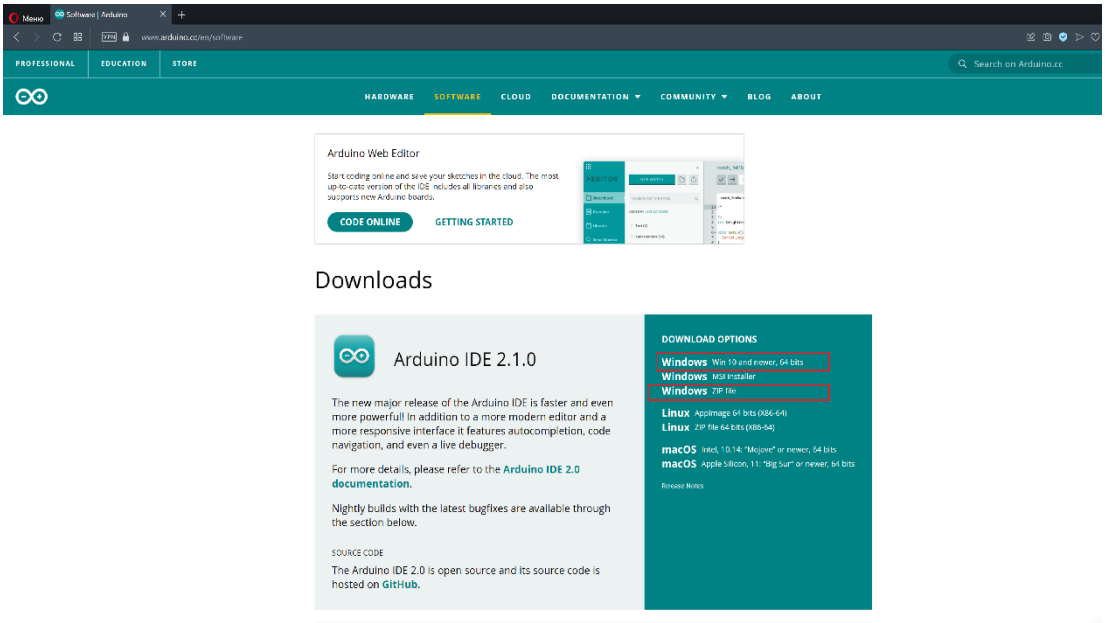


Рисунок 2.6 – Інсталятори Arduino IDE

Далі відмовляємося від донату натисткаючи кнопку Just download та починається процес завантаження (див. рис. 2.7).

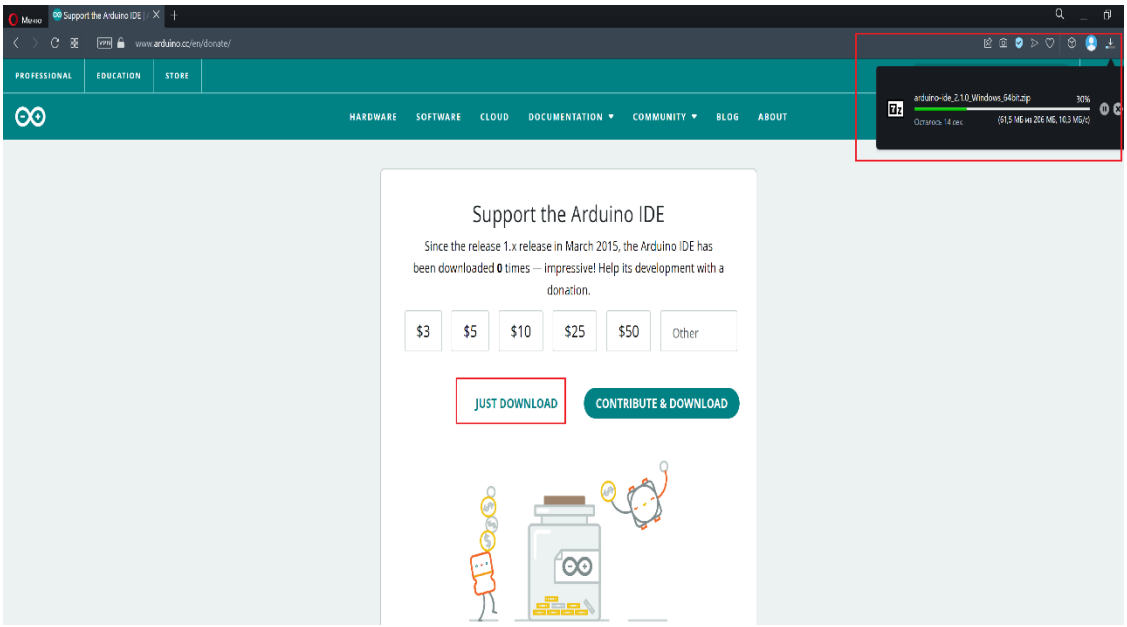


Рисунок 2.7 – Завантаження Arduino IDE

Після завантаження, у нашому випадку розпаковуємо портативний архів за допомогою або стокового архіватору, або WinRar [15] чи 7-zip [16] (див.рис. 2.8–2.10).

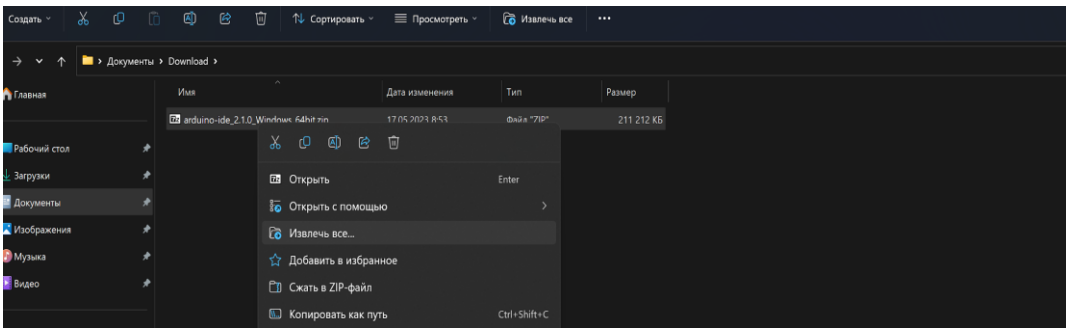


Рисунок 2.8 – Розпакування архіву Arduino IDE

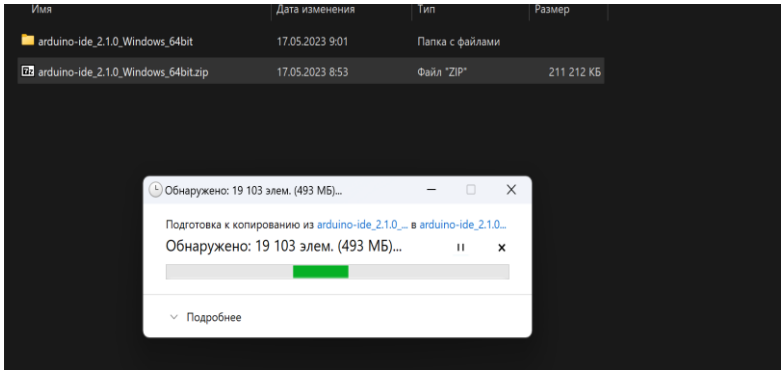


Рисунок 2.9 – Процес розпакування Arduino IDE

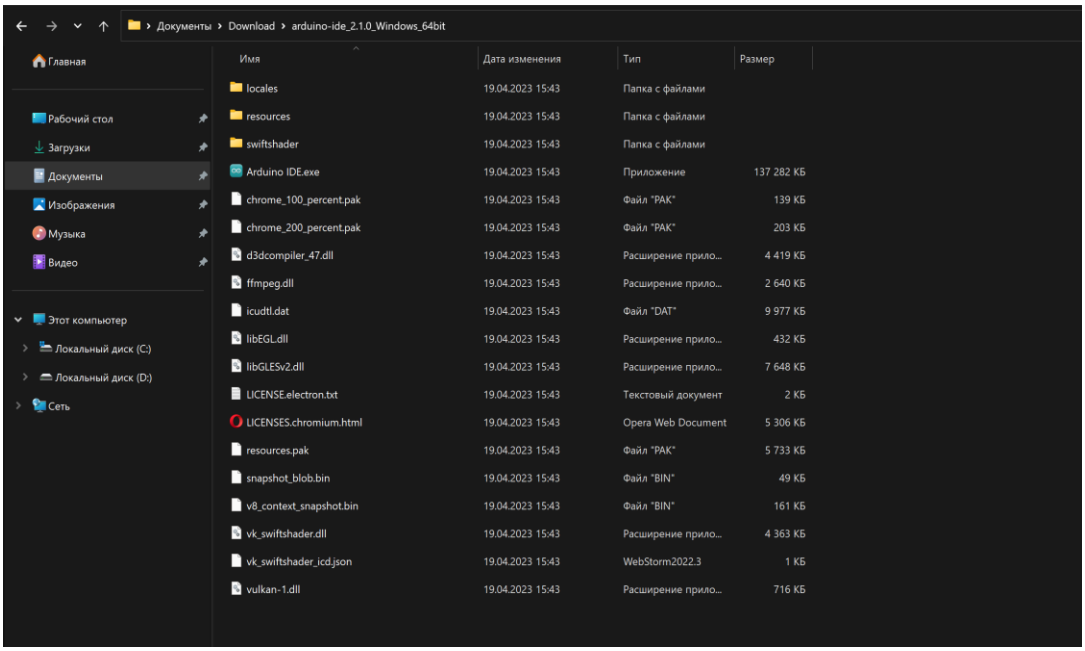


Рисунок 2.10 – Результат маніпуляцій

Натискаємо на файл Arduino IDE.exe, погоджуємось на можливі доп. завантаження, видаємо права, якщо попросить (див. рис. 2.11).

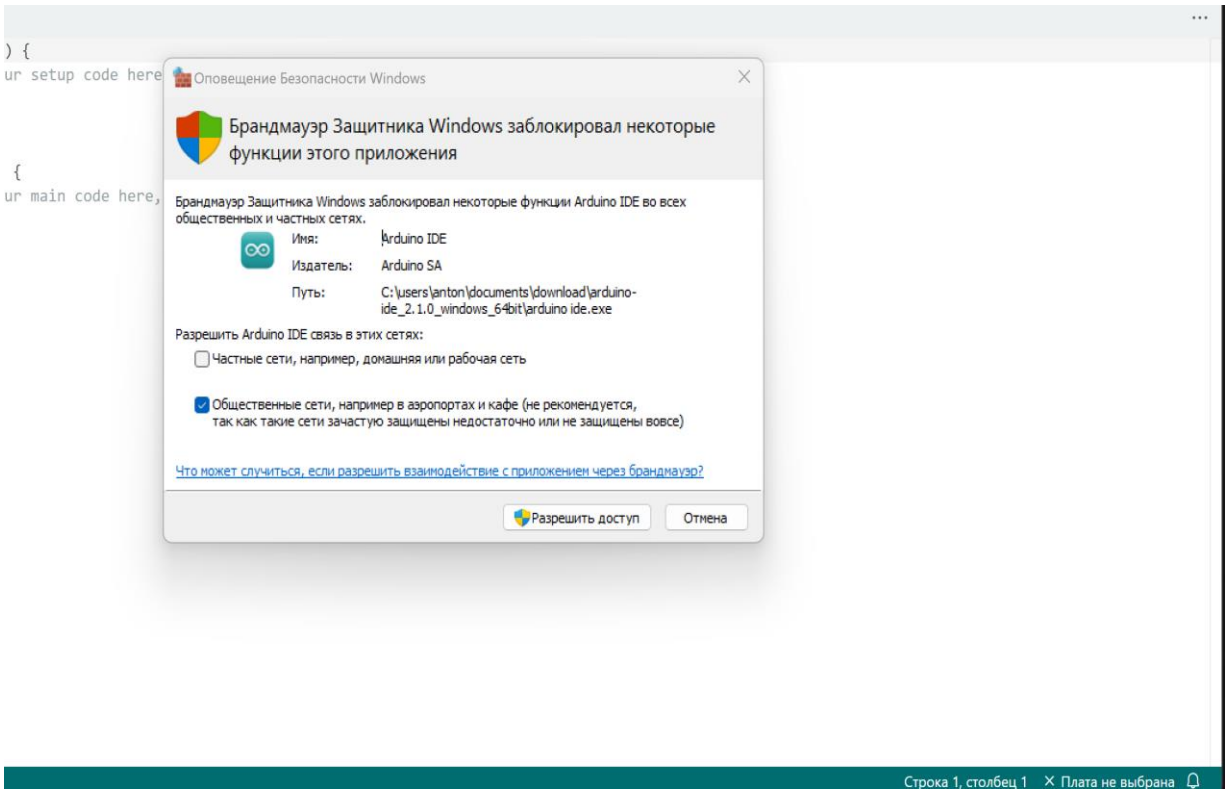


Рисунок 2.11 – Можливі додаткові права

Далі обираємо вкладку File та шукаємо кнопку Preferences (див. рис. 2.12).

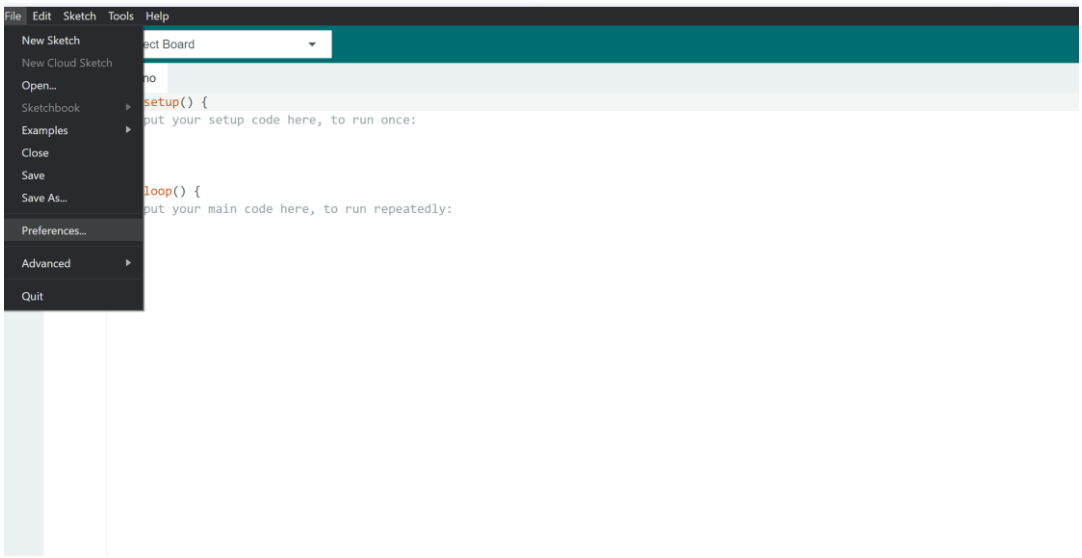


Рисунок 2.12 – Вкладка File, кнопка Preferences

Обираємо більш звичну нам мову, а також прописуємо до Additional boards manager URL додатковий драйвер для конкретно нашої плати NodeMCU 8266 з сайту esp8266 Arduino Core [17] (див.рис. 2.13–2.14).

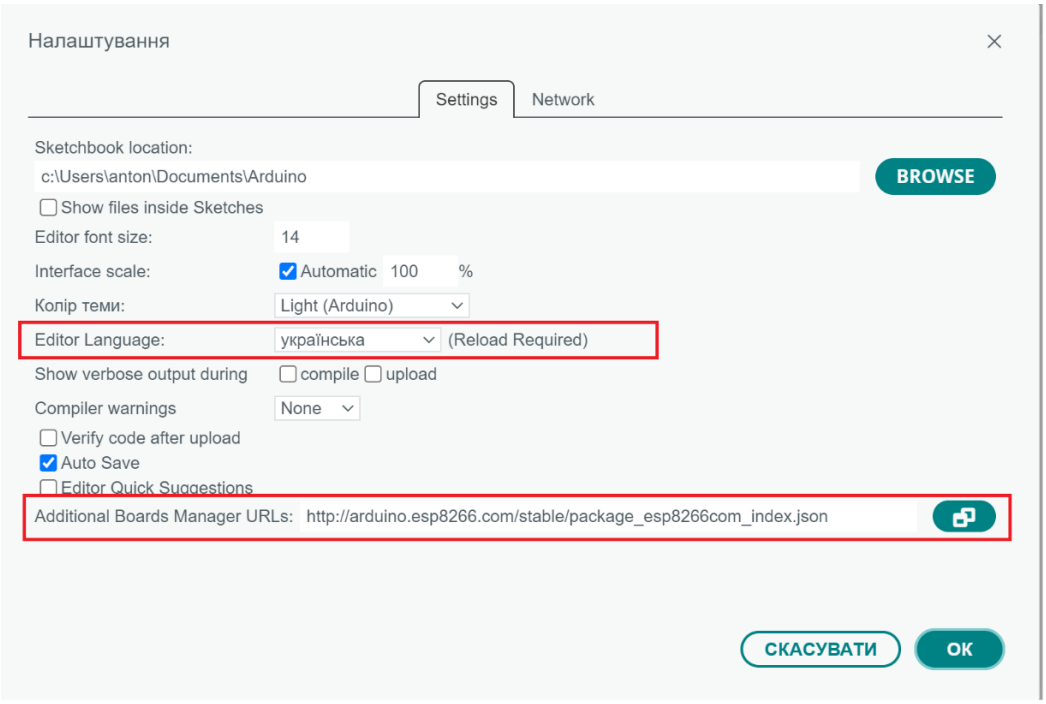


Рисунок 2.13 – Розділ Preferences та необхідні зміни в ньому

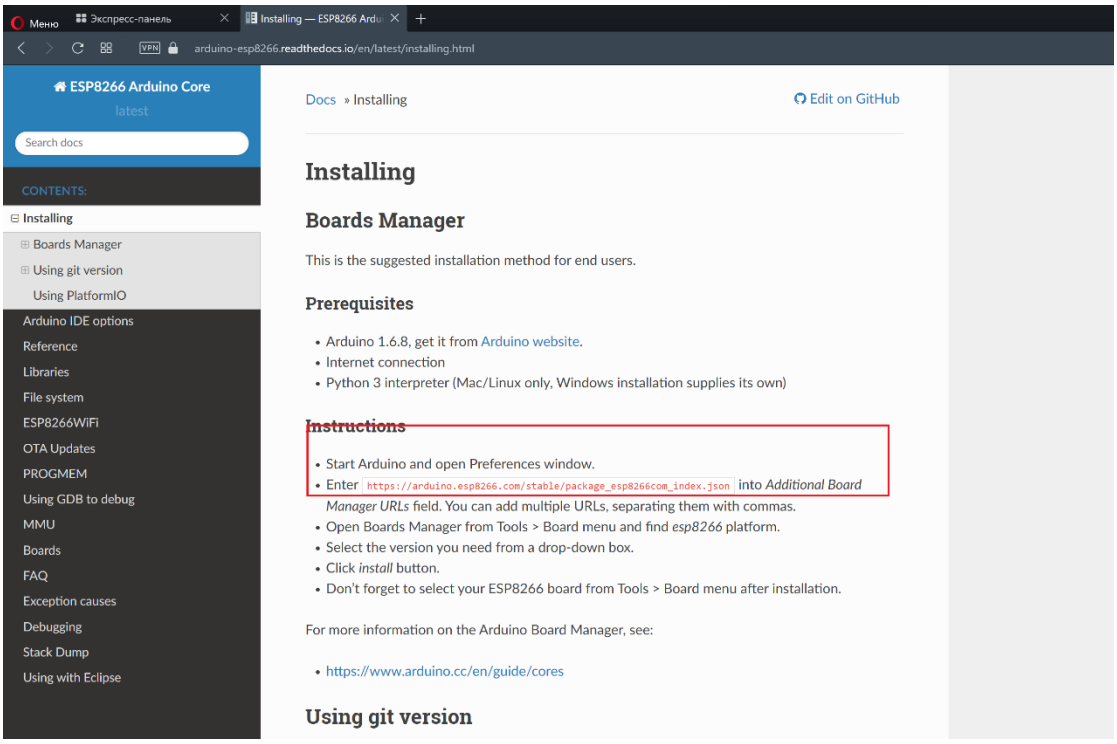


Рисунок 2.14 – Інструкції та необхідний json для плати

Далі вертаємось до початкового екрану взаємодії з Arduino IDE та шукаємо на вкладці Board вкладку Tools а у ній esp8266 (див. рис. 2.15). В нашому випадку, серед запропонованих драйверів обираємо для плати NodeMCU 1.0(ESP-12E Module).

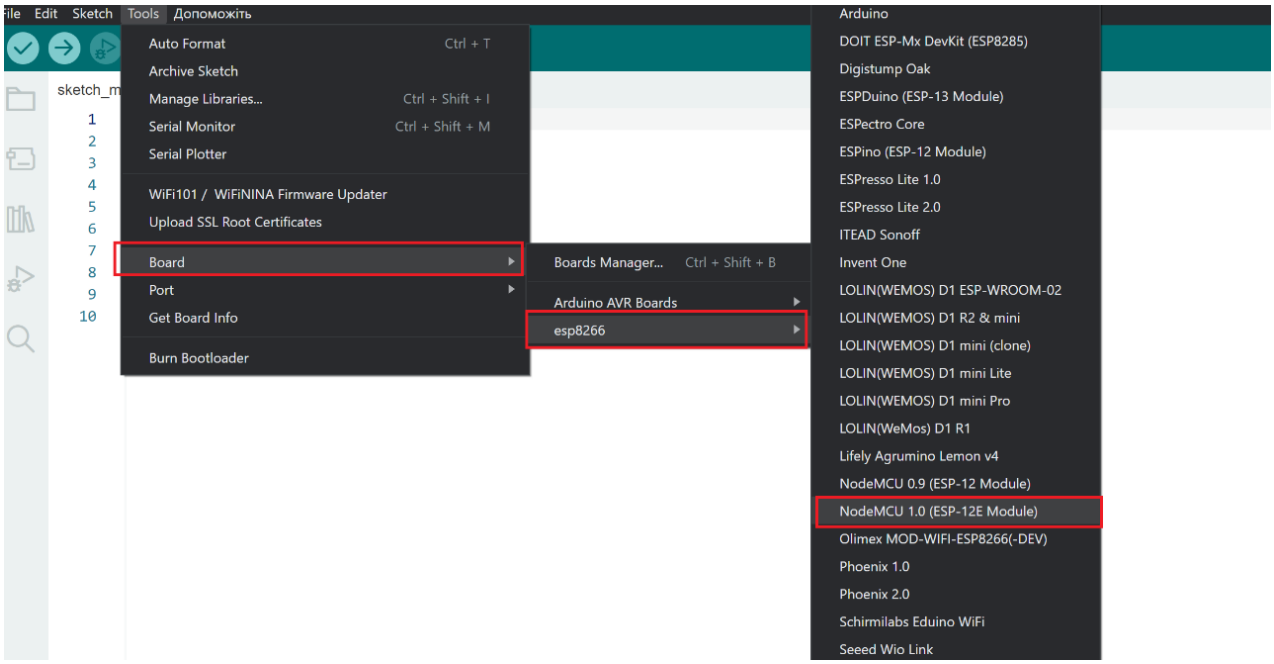


Рисунок 2.15 – Вибір правильного драйверу для плати серед запропонованих

Стосовно налаштувань IDE майже все. Тепер необхідно завантажити драйвер для взаємодії com-порта нашого пристрою з платами nodemsu. У нашому випадку, нам необхідний драйвер під назвою CH341SER для Windows. Завантажимо такий з сайту NanjingQinhengMicroelectronics [18] (див. рис. 2.16).

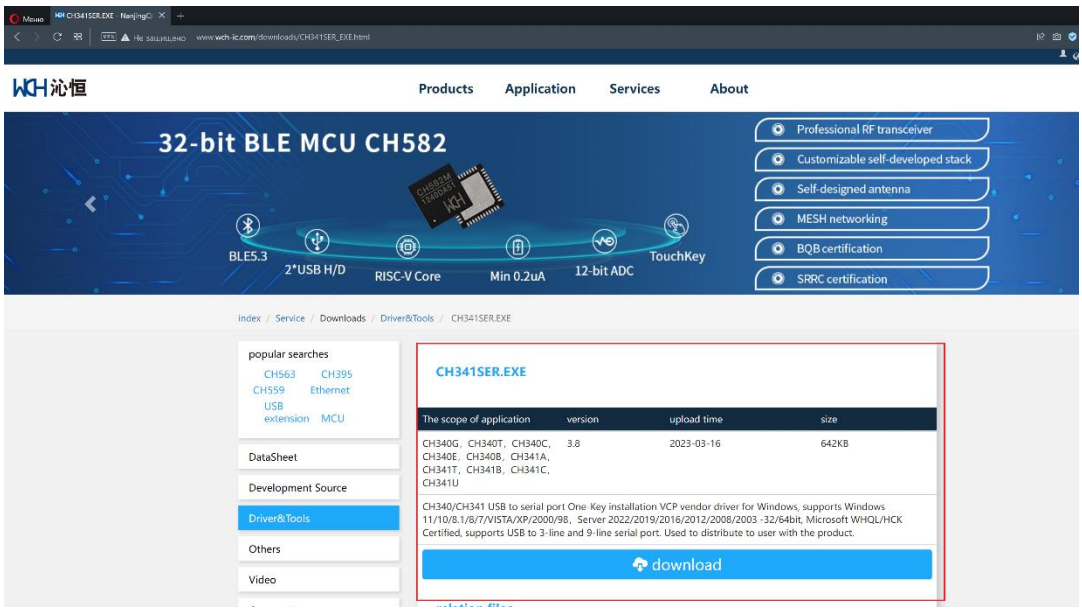


Рисунок 2.16 – Завантаження драйверу для взаємодії com-порта з NodeMCU

Ми завантажили файл під назвою CH341SER.EXE, зараз просто запускаємо його (див. рис. 2.17).

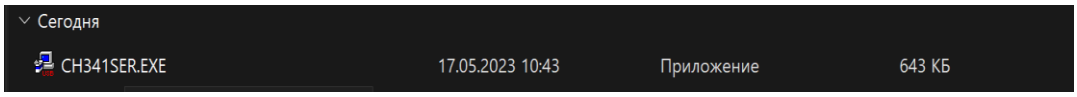


Рисунок 2.17 – Завантажений файл

Потім натискаємо на кнопку Install та отримуємо приблизно такий результат (див. рис. 2.18).

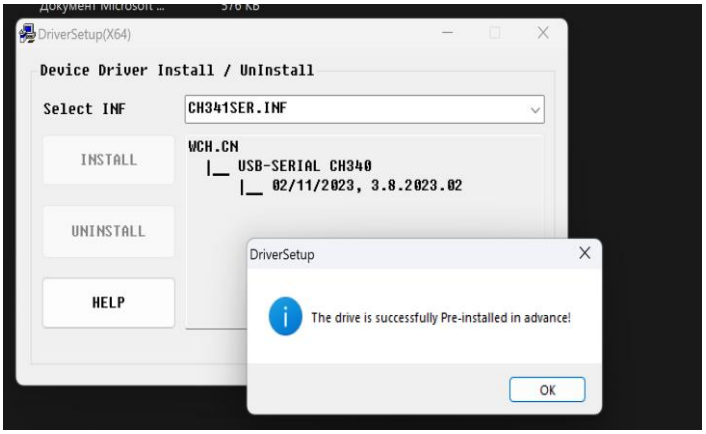


Рисунок 2.18 – Встановлення драйверу для com-порта

Після цього можемо підключати плату esp8266 через usb до нашого пристрою. Повертаємось в Arduino IDE та завершуємо налаштування. Ми вже обрали необхідний драйвер для плати, тепер натискаємо на вкладку Select board та обираємо той що з’явився після підключення нашої плати. У даному випадку, це COM6 (див. рис. 2.19). Підтвержуємо свій вибір.

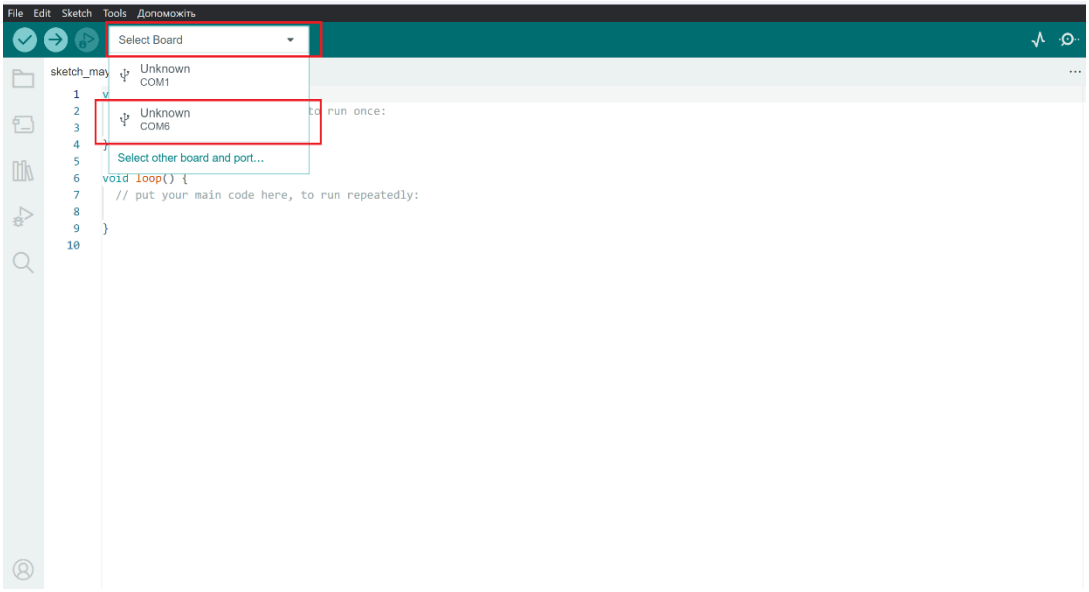


Рисунок 2.19 – Встановлення драйверу для com-порта

Отже, після всіх маніпуляцій, ми підготували плату до роботи і можемо приступати до розробки. З розробленим скетчем для плати детально можна ознайомитись у додатку А.

2.4 Підключення плати до камери

Для реалізації проекту необхідно підключити нашу камеру до плати. Для підключення будь-яких елементів Arduino між собою необхідно створити схему підключення згідно документації. Отже, на основі інформації з розділів 2.1 та 2.2 створимо схему підключення плати LoLin NodeMCU [11] до камери PTC 06 [12] (див. рис. 2.20).

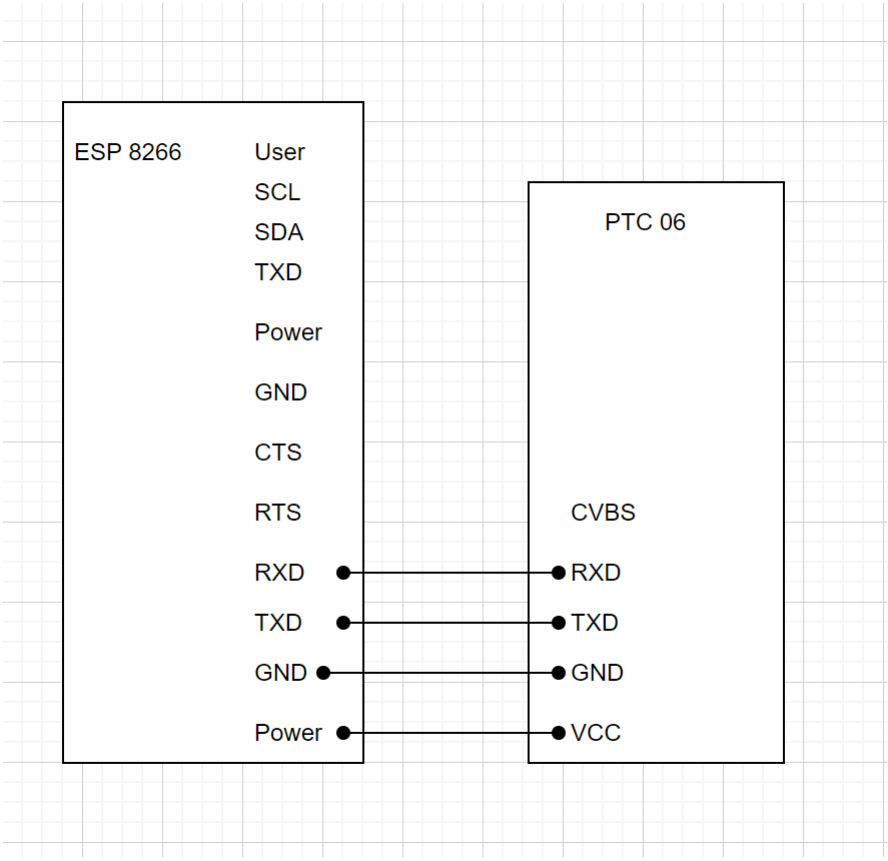


Рисунок 2.20 – Схема підключення плати до камери

Поєднаємо вказані піни камери та плати згідно схеми підключення. Якщо все було зроблено вірно, то можемо приступати до розробки мобільного додатку.

3 РЕАЛІЗАЦІЯ

3.1 Розробка додатку управління камерою відеоспостереження

Додаток буде створений на мові програмування Java, з підключенням по локальній мережі або через пряме посилання на відео в live-режимі за допомогою компонента VideoView, а також маючий змогу сканувати локальну мережу на наявність локальних api камер, підключених до неї для прямого підключення до них без необхідності вводу посилання. Повинна бути можливість створювати, зберігати, редагувати, видаляти посилання у додатку.

Для початку, пропишемо необхідні бібліотеки в наш проект у файл build.gradle (див. рис. 3.1).

```
plugins {
    id 'com.android.application'
}

android {
    namespace 'com.example.ipcamera'
    compileSdk 33

    defaultConfig {
        applicationId "com.example.ipcamera"
        minSdk 24
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.5.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'

    implementation 'com.google.code.gson:gson:2.10.1'
```

Рисунок 3.1 – Вміст файлу build.gradle

Далі пропишемо глобальні кольори для всього додатку у файлі themes.xml (див. рис. 3.2).

Рисунок 3.2 – Вміст файлу themes.xml

Маємо на увазі, що додаток бере свої кольори та стилі з файлів colors.xml, strings.xml, styles.xml (див. рис. 3.3).

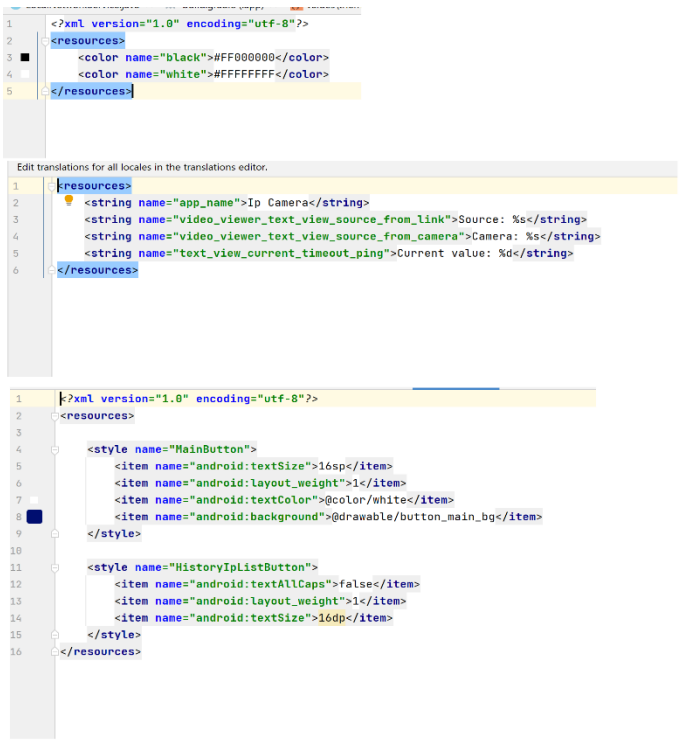


Рисунок 3.3 – Вміст файлів colors.xml, strings.xml, styles.xml відповідно

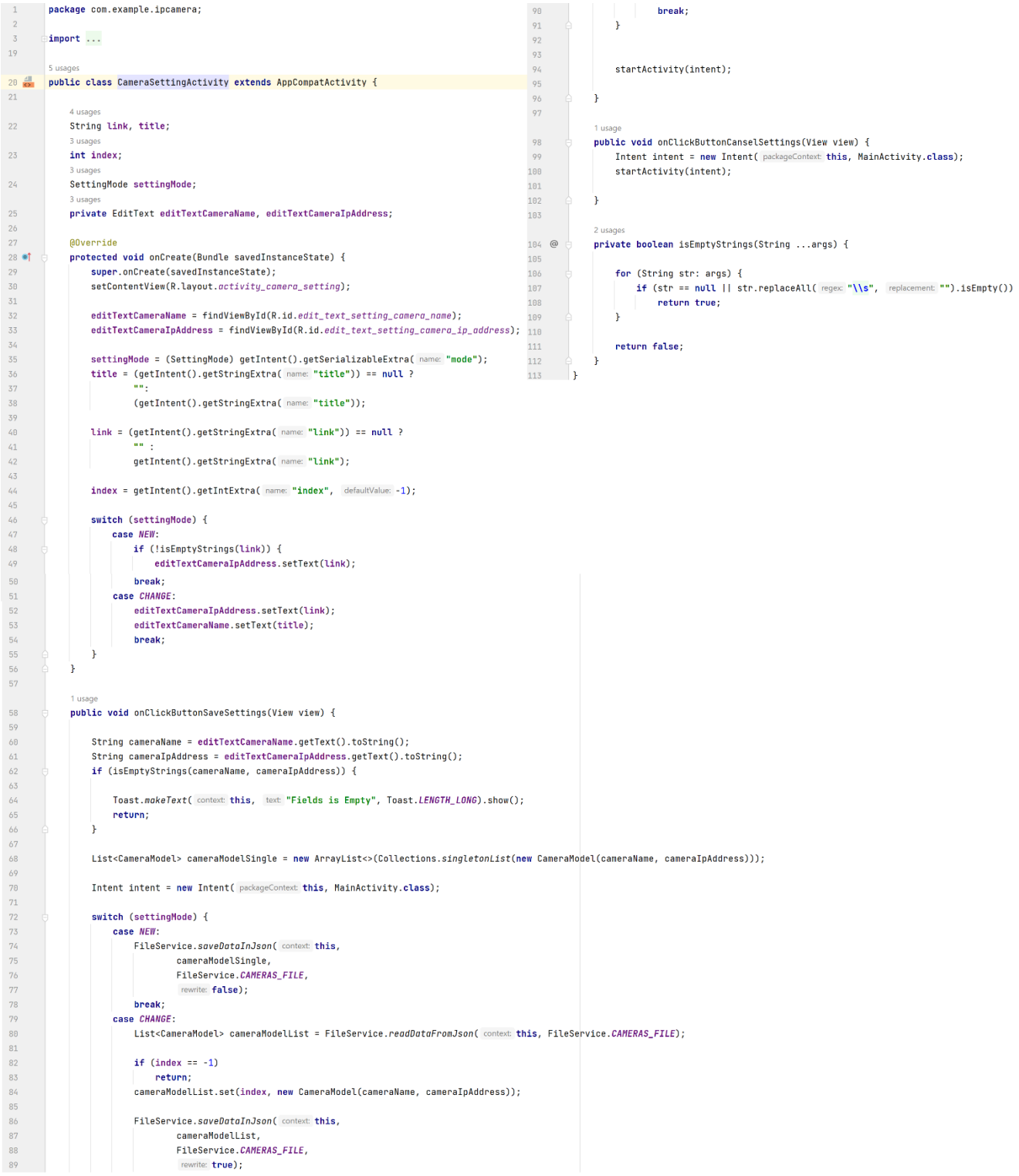
Пропишемо назву та необхідні опції/допуски від пристрою для додатка у файлі AndroidManifest.xml (див. рис. 3.4).

```
    android:maxSdkVersion="32"
    tools:ignore="ScopedStorage" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="Ip Camera"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.IpCamera"
    tools:targetApi="31"
    android:usesCleartextTraffic="true">
    <activity
        android:name=".LocalIpActivity"
        android:exported="false"
        android:screenOrientation="portrait"/>
    <activity
        android:name=".CameraViewerActivity"
        android:exported="false"
        android:screenOrientation="portrait" />
    <activity
        android:name=".CameraSettingActivity"
        android:exported="false"
        android:screenOrientation="portrait" />
    <activity
        android:name=".MainActivity"
        android:exported="true"
        android:screenOrientation="portrait">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
```

Рисунок 3.4 – Вміст файлу AndroidManifest.xml

Пропишемо дизайн кожного Activity нашого додатку. У нас буде 7 Activity, створимо кожному по одному. У данному випадку було створено файли з назвами activity_camera_setting.xml, activity_camera_viewer.xml, activity_local_ip.xml, activity_main.xml, button_setting_my_cameras.xml, fragment_history.xml, fragment_my_cameras.xml, з якими можна детально ознайомитись у додатку Б.

Створимо 6 Activity/Fragment, в яких прописана основна логіка для кожного екрану. У нашому випадку було створено 6 файлів: CameraSettingActivity, CameraVieverActivity, HistoryFragment, LocalIpActivity, MainActivity, MyCamerasFragment (див. рис. 3.5–3.14).



```
package com.example.ipcamera;

import ...

5 usages
public class CameraSettingActivity extends AppCompatActivity {

    4 usages
    String link, title;
    3 usages
    int index;
    3 usages
    SettingMode settingMode;
    3 usages
    private EditText editTextCameraName, editTextCameraIpAddress;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_camera_setting);

        editTextCameraName = findViewById(R.id.edit_text_setting_camera_name);
        editTextCameraIpAddress = findViewById(R.id.edit_text_setting_camera_ip_address);

        settingMode = (SettingMode) getIntent().getSerializableExtra( name: "mode");
        title = (getIntent().getStringExtra( name: "title")) == null ?
                "":
                (getIntent().getStringExtra( name: "title"));

        link = (getIntent().getStringExtra( name: "link")) == null ?
                "" :
                getIntent().getStringExtra( name: "link");

        index = getIntent().getIntExtra( name: "index",  defaultValue: -1);

        switch (settingMode) {
            case NEW:
                if (!isEmptyStrings(link)) {
                    editTextCameraIpAddress.setText(link);
                }
                break;
            case CHANGE:
                editTextCameraIpAddress.setText(link);
                editTextCameraName.setText(title);
                break;
        }
    }

    1 usage
    public void onClickButtonSaveSettings(View view) {

        String cameraName = editTextCameraName.getText().toString();
        String cameraIpAddress = editTextCameraIpAddress.getText().toString();
        if (isEmptyStrings(cameraName, cameraIpAddress)) {

            Toast.makeText( context: this,  text: "Fields is Empty", Toast.LENGTH_LONG).show();
            return;
        }

        List<CameraModel> cameraModelSingle = new ArrayList<>(Collections.singletonList(new CameraModel(cameraName, cameraIpAddress)));

        Intent intent = new Intent( packageContext: this, MainActivity.class);

        switch (settingMode) {
            case NEW:
                FileService.saveDataInJson( context: this,
                        cameraModelSingle,
                        FileService.CAMERAS_FILE,
                         rewrite: false);
                break;
            case CHANGE:
                List<CameraModel> cameraModelList = FileService.readDataFromJson( context: this, FileService.CAMERAS_FILE);

                if (index == -1)
                    return;
                cameraModelList.set(index, new CameraModel(cameraName, cameraIpAddress));

                FileService.saveDataInJson( context: this,
                        cameraModelList,
                        FileService.CAMERAS_FILE,
                         rewrite: true);
                break;
        }

        startActivity(intent);

    }

    1 usage
    public void onClickButtonCanselSettings(View view) {
        Intent intent = new Intent( packageContext: this, MainActivity.class);
        startActivity(intent);

    }

    2 usages
    private boolean isEmptyStrings(String ...args) {

        for (String str: args) {
            if (str == null || str.replaceAll( regex: "\\s",  replacement: "").isEmpty())
                return true;
        }

        return false;
    }
}
```

Рисунок 3.5 – Вміст файлу CameraSettingActivity

```java
package com.example.ipcamera;

import ...

5 usages
public class CameraViewerActivity extends AppCompatActivity {

    3 usages
    private OpenVideoActivityMode openMode;
    2 usages
    private Button buttonSaveSource, buttonBack;
    1 usage
    private Button buttonPlay, buttonStop, buttonFullScreen;
    4 usages
    private TextView textViewSource;
    12 usages
    private VideoView videoView;
    4 usages
    String ipAddress;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_camera_viewer);

        init();

        openMode = (OpenVideoActivityMode) getIntent().getSerializableExtra( name: "open_mode");

        String title = getIntent().getStringExtra( name: "camera_title");
        ipAddress = getIntent().getStringExtra( name: "camera_address");

        switch (openMode) {
            case FROM_CONNECT:
            case FROM_SCANNED_LIST:

                String source = String.format("Source: %s", ipAddress);
                textViewSource.setText(source);

                break;

            case FROM_MY_CAMERAS:

                String cameraName = String.format("Camera: %s", title);
                textViewSource.setText(cameraName);
                buttonSaveSource.setVisibility(View.GONE);

                break;
        }

        if (getResources().getConfiguration().orientation == Configuration.ORIENTATION_LANDSCAPE) {
            setFullScreenProperty();
        }

        videoView.setVideoURI(Uri.parse(ipAddress));
        videoView.start();
    }

    1 usage
    private void init() {
        buttonSaveSource = findViewById(R.id.button_save_camera_from_video_viewer);
        buttonBack = findViewById(R.id.button_back_from_video_viewer);

        buttonPlay = findViewById(R.id.button_play_video);
        buttonStop = findViewById(R.id.button_stop_video);
        buttonFullScreen = findViewById(R.id.button_full_screen_video);

        textViewSource = findViewById(R.id.text_view_video_source);
        videoView = findViewById(R.id.video_view_translate);


    }
```

Рисунок 3.6 – Вміст файлу CameraVieverActivity

```
87      private void setFullScreenProperty() {
88          LinearLayout linearLayoutMediaButton = findViewById(R.id.linear_layout_media_button);
89          LinearLayout linearLayoutFunctionalButton = findViewById(R.id.linear_layout_functional_button);
90          RelativeLayout relativeLayoutContainer = findViewById(R.id.relative_layout_container_video_viewer);
91          ImageButton imageButtonExitFullScreenMode = findViewById(R.id.image_button_exit_full_screen);
92
93          ViewGroup.MarginLayoutParams relativeLayoutParams = (ViewGroup.MarginLayoutParams) relativeLayoutContainer.getLayoutParams()
94          relativeLayoutParams.setMargins( left: 0,  top: 0,  right: 0,  bottom: 0);
95          relativeLayoutContainer.setLayoutParams(relativeLayoutParams);
96
97
98          ViewGroup.LayoutParams videoViewLayoutParams = videoView.getLayoutParams();
99          videoViewLayoutParams.height = ViewGroup.LayoutParams.MATCH_PARENT;
100         videoView.setLayoutParams(videoViewLayoutParams);
101
102         ViewGroup.MarginLayoutParams VideoViewMarginLayoutParams = (ViewGroup.MarginLayoutParams) videoView.getLayoutParams();
103         VideoViewMarginLayoutParams.setMargins( left: 0,  top: 0,  right: 0,  bottom: 0);
104         videoView.setLayoutParams(VideoViewMarginLayoutParams);
105
106         textViewSource.setVisibility(View.GONE);
107         linearLayoutMediaButton.setVisibility(View.GONE);
108         linearLayoutFunctionalButton.setVisibility(View.GONE);
109         imageButtonExitFullScreenMode.setVisibility(View.VISIBLE);
110     }
111
        3 usages
112 @   public void onClickVideoControlButtons(View view) {
113
114         String buttonTag = (String) view.getTag();
115
116         switch (buttonTag) {
117
118             case "play":
119                 videoView.start();
120                 break;
121
122             case "stop":
123                 videoView.pause();
124                 break;
125
126             case "full_screen":
127                 videoView.stopPlayback();
128
129                 setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
130                 break;
131         }
132     }
133
        1 usage
134     @SuppressLint("SourceLockedOrientationActivity")
135     public void onClickExitFullScreenMode(View view) {
136         setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
137     }
138
        1 usage
139     public void OnClickButtonSave(View view) {
140         videoView.stopPlayback();
141
142         Intent intent = new Intent( packageContext: this, CameraSettingActivity.class);
143         intent.putExtra( name: "mode", SettingMode.NEW);
144         intent.putExtra( name: "link", this.ipAddress);
145         startActivity(intent);
146     }
147
        1 usage
148     public void onClickButtonBack(View view) {
149
150         videoView.stopPlayback();
151
152         Intent intent;
153
154         if (openMode == OpenVideoActivityMode.FROM_SCANNED_LIST)
155             intent = new Intent( packageContext: this, LocalIpActivity.class);
156         else
157             intent = new Intent( packageContext: this, MainActivity.class);
158
159         startActivity(intent);
160     }
```

Рисунок 3.7 – Продовження вмісту файлу CameraVieverActivity

```java
package com.example.ipcamera;

import ...

3 usages
public class HistoryFragment extends Fragment {

    5 usages
    private ArrayAdapter<String> adapterListView;
    3 usages
    private View rootView;
    6 usages
    private ListView listView;
    3 usages
    private TextView textViewHistoryEmpty;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_history, container, attachToRoot: false);
    }

    9 usages
    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        rootView = view;

        listView = rootView.findViewById(R.id.list_view_history);
        textViewHistoryEmpty = rootView.findViewById(R.id.text_view_history_info);

        adapterListView = new ArrayAdapter<>(getActivity(), android.R.layout.simple_list_item_1, GlobalData.connectionHistoryList);


        if (GlobalData.connectionHistoryList.size() == 0)
            return;


        textViewHistoryEmpty.setVisibility(View.INVISIBLE);
                TextView textViewLink = requireActivity().findViewById(R.id.edit_text_link);

                if (textViewLink == null)
                    return;

                textViewLink.setText((String)listView.getItemAtPosition(position));
            }
        });

    }
    1 usage
    public void addItemToList(String link) {

        GlobalData.connectionHistoryList.add(link);

        if (listView == null)
            return;

        adapterListView.notifyDataSetChanged();
        textViewHistoryEmpty.setVisibility(View.INVISIBLE);
//        ArrayAdapter<String> adapter = new ArrayAdapter<>(getActivity(), android.R.layout.simple_list_item_1, GlobalData.connectionHistoryList);
//        listView.setAdapter(adapter);

    }
    1 usage
    public void clearAllData() {

        GlobalData.connectionHistoryList.clear();

        if (listView == null)
            return;

        adapterListView.clear();
        adapterListView.notifyDataSetChanged();
    }
}
```

Рисунок 3.8 – Вміст файлу HistoryFragment

```java
package com.example.ipcamera;

import ...

6 usages
public class LocalIpActivity extends AppCompatActivity {

    1 usage
    private ImageButton imageButtonSetPingTimeout;
    4 usages
    private ListView listView;
    3 usages
    private TextView textViewLocalIpInfo, textViewWaitingMassage, textViewCurrentValue;
    1 usage
    private Button buttonScanLocal, buttonBack, buttonSaveSetting;
    3 usages
    private RelativeLayout relativeLayoutSettingsBg;
    3 usages
    private EditText editTextSetTimeout;
    7 usages
    ArrayAdapter<String> arrayAdapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_local_ip);

        init();


        if (GlobalData.localIpAddressLIst != null) {
            textViewLocalIpInfo.setVisibility(View.GONE);
            arrayAdapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, GlobalData.localIpAddressLIst);
            listView.setAdapter(arrayAdapter);
        }


        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                Intent intent = new Intent( packageContext: LocalIpActivity.this, CameraViewerActivity.class);
                intent.putExtra( name: "open_mode", OpenVideoActivityMode.FROM_SCANNED_LIST);
                intent.putExtra( name: "camera_address", parent.getItemAtPosition(position).toString());

                startActivity(intent);
            }
        });
    }

    1 usage
    private void init() {
        imageButtonSetPingTimeout = findViewById(R.id.image_button_local_ip_settings);
        listView = findViewById(R.id.list_view_local_ip);
        textViewLocalIpInfo = findViewById(R.id.text_view_local_ip_info);
        textViewWaitingMassage = findViewById(R.id.text_view_scanning_waiting);
        buttonScanLocal = findViewById(R.id.button_scan_local);
        buttonBack = findViewById(R.id.button_back_from_local_ip);
        relativeLayoutSettingsBg = findViewById(R.id.relative_layout_background_setting_local_scan);
        editTextSetTimeout = findViewById(R.id.edit_text_set_timeout);
        buttonSaveSetting = findViewById(R.id.button_save_local_ip_settings);
        textViewCurrentValue = findViewById((R.id.text_view_current_ping_timeout));

        String value = String.format("Current value: %d", GlobalData.currentPingTimeout);

        textViewCurrentValue.setText(value);
    }

    1 usage
    private void scanLocalIp() {

        textViewWaitingMassage.setVisibility(View.VISIBLE);
        if (arrayAdapter != null) {
            arrayAdapter.clear();
            arrayAdapter.notifyDataSetChanged();
        }
        new Thread(new Runnable() {
            @Override
            public void run() {

                GlobalData.localIpAddressLIst = LocalNetworkService.getAllActiveLocalIp(GlobalData.currentPingTimeout);

                new Handler(Looper.getMainLooper()).post(new Runnable() {
                    @Override
                    public void run() {
                        textViewWaitingMassage.setVisibility(View.GONE);

                        arrayAdapter = new ArrayAdapter<>( context: LocalIpActivity.this, android.R.layout.simple_list_item_1, GlobalData.localIpAddressLIst);
                        listView.setAdapter(arrayAdapter);

                        textViewLocalIpInfo.setVisibility(View.GONE);
                    }
                });
            }
        }).start();

    }
```

Рисунок 3.9 – Вміст файлу LocalIpActivity

```
107     }
        1 usage
108     public void onClickOpenSettingsButton(View view) {
109         relativeLayoutSettingsBg.setVisibility(View.VISIBLE);
110     }
111
        1 usage
112     public void onClickCloseSettingRelativeLayout(View view) {
113         relativeLayoutSettingsBg.setVisibility(View.GONE);
114     }
115
        1 usage
116     public void onClickScanButton(View view) { scanLocalIp(); }
119
        1 usage
120     public void onClickBackButton(View view) {
121         Intent intent = new Intent( packageContext: this, MainActivity.class);
122         startActivity(intent);
123     }
124
        1 usage
125     public void onClickSaveNewTimeout(View view) {
126
127         String textValue = editTextSetTimeout.getText().toString();
128         int value = -1;
129
130         String errorMessage = "";
131
132         if (textValue.isEmpty())
133             errorMessage = "Field is empty!";
134         else {
135             try {
136                 value = Integer.parseInt(textValue);
137             } catch (Exception e) {
138                 errorMessage = "Incorrect format!";
139             }
140         }
141
142         if (value < 0) {
143             Toast.makeText( context: this, errorMessage, Toast.LENGTH_LONG).show();
144
145             return;
146         }
147
148         if (value < 10 || value > 1000) {
149             Toast.makeText( context: this,  text: "Incorrect diapason!", Toast.LENGTH_LONG).show();
150         } else  {
151             GlobalData.currentPingTimeout = value;
152
153             textViewCurrentValue.setText(
154                     String.format("Current value: %d", GlobalData.currentPingTimeout)
155             );
156
157             editTextSetTimeout.setText("");
158         }
159     }
160 }
```

Рисунок 3.10 – Продовження вмісту файлу LocalIpActivity

```java
package com.example.ipcamera;

import ...

7 usages
public class MainActivity extends AppCompatActivity {

    4 usages
    private HistoryFragment historyFragment;
    3 usages
    private MyCamerasFragment myCamerasFragment;
    2 usages
    private Button buttonHistory, buttonIpList;
    1 usage
    private ImageButton imageButtonMoreAction;
    4 usages
    private EditText editTextUserLink;
    3 usages
    private RelativeLayout relativeLayoutMoreActionBg;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);

        buttonHistory = findViewById(R.id.button_history);
        buttonIpList = findViewById(R.id.button_ip_cameras_list);
        editTextUserLink = findViewById(R.id.edit_text_link);
        relativeLayoutMoreActionBg = findViewById(R.id.relative_layout_background_more_action_panel);
        imageButtonMoreAction = findViewById(R.id.image_button_more_action);

        buttonIpList.setSelected(true);

        historyFragment = new HistoryFragment();
        myCamerasFragment = new MyCamerasFragment();

        setFragment(myCamerasFragment);

        GlobalData.connectionHistoryList = FileService.readData( context: this, FileService.HISTORY_FILE);

        if (GlobalData.connectionHistoryList == null) {
            GlobalData.connectionHistoryList = new ArrayList<>();
        }
    }

    2 usages
    public void onClickHistoryIpListButtons(View view) {
        if (view.isSelected())
            return;

        setSelectedButton(buttonHistory, buttonIpList);
        String tag = view.getTag().toString();

        switch(tag) {
            case "my_cameras":
                setFragment(myCamerasFragment);
                break;

            case "history":
                setFragment(historyFragment);
                break;

            default:
                break;
        }
    }

    1 usage
    private void setSelectedButton(View btnHistory, View btnIpLIst) {
        btnHistory.setSelected(!btnHistory.isSelected());
        btnIpLIst.setSelected(!btnIpLIst.isSelected());

    }

    1 usage
    public void onClickConnectButton(View view) {
        String link = editTextUserLink.getText().toString();
        if (link.replaceAll( regex: "\\s",  replacement: "").isEmpty()) {
            Toast.makeText( context: this,  text: "Field is Empty", Toast.LENGTH_LONG).show();
            return;
        }

        historyFragment.addItemToList(link);
        FileService.saveData( context: this, new ArrayList<>(Collections.singletonList(link)), FileService.HISTORY_FILE,  rewrite: false);

        Intent intent = new Intent( packageContext: this, CameraViewerActivity.class);
        intent.putExtra( name: "open_mode", OpenVideoActivityMode.FROM_CONNECT);
        intent.putExtra( name: "camera_address", editTextUserLink.getText().toString());

        startActivity(intent);

    }

```

Рисунок 3.11 – Вміст файлу MainActivity

```
114     public void onClickAddCameraButton(View view) {
115         Intent intent = new Intent( packageContext: this, CameraSettingActivity.class);
116         intent.putExtra( name: "mode", SettingMode.NEW);
117         intent.putExtra( name: "link", editTextUserLink.getText().toString());
118         startActivity(intent);
119     }
120
            3 usages
121     private void setFragment(Fragment fragment) {
122         @SuppressLint("CommitTransaction")
123         FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
124         ft.replace(R.id.frame_layout, fragment);
125         ft.addToBackStack( name: null);
126         ft.commit();
127
128     }
129
            1 usage
130     public void onClickMoreActionBg(View view) {
131         relativeLayoutMoreActionBg.setVisibility(View.GONE);
132     }
133
            3 usages
134 @   public void onClickMoreActionButtons(View view) {
135         String buttonTag = view.getTag().toString();
136         switch (buttonTag) {
137             case "local_ip":
138                 Intent intent = new Intent( packageContext: this, LocalIpActivity.class);
139                 startActivity(intent);
140                 break;
141
142             case "clear_history":
143                 showDeleteHistoryAlert();
144                 break;
145
146             case  "exit":
147                 finishAffinity();
148                 System.exit( status: 0);
149                 break;
150         }
151     }
            1 usage
152     public void onClickOpenMoreActionButton(View view) {
153         relativeLayoutMoreActionBg.setVisibility(View.VISIBLE);
154     }
155
            1 usage
156     private void showDeleteHistoryAlert() {
157         AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
158         builder.setTitle("Delete")
159                 .setMessage("Delete history?")
160                 .setCancelable(false)
161                 .setPositiveButton( text: "Yes", new DialogInterface.OnClickListener() {
162                     @Override
163                     public void onClick(DialogInterface dialog, int which) {
164                         historyFragment.clearAllData();
165                         FileService.cleanFile( context: MainActivity.this, FileService.HISTORY_FILE);
166                         dialog.cancel();
167                     }
168                 })
169                 .setNegativeButton( text: "No", new DialogInterface.OnClickListener() {
170                     @Override
171                     public void onClick(DialogInterface dialog, int which) { dialog.cancel(); }
174                 });
175
176         AlertDialog dialog = builder.create();
177         dialog.show();
178     }
179
180     @Override
181     public void onBackPressed() { moveTaskToBack( nonRoot: true); }
184 }
```

Рисунок 3.12 – Продовження вмісту файлу MainActivity

```java
package com.example.ipcamera;

import ...

3 usages
public class MyCamerasFragment
        extends Fragment
        implements MyAdapter.OnItemClickListener, MyAdapter.OnButtonClickListener, MyAdapter.OnItemLongClickListener {

    3 usages
    private View rootView;
    3 usages
    private MyAdapter adapter;
    3 usages
    private RecyclerView recyclerView;
    2 usages
    private TextView textViewMyCamerasEmpty;
    9 usages
    private List<CameraModel> camerasList;
    4 usages
    List<String> titleList;
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_my_cameras, container,  attachToRoot: false);
    }

    9 usages
    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        rootView = view;
        textViewMyCamerasEmpty = rootView.findViewById(R.id.text_view_my_cameras_info);
        camerasList = FileService.readDataFromJson(getContext(), FileService.CAMERAS_FILE);

        titleList = new ArrayList<>();

        for (CameraModel cm : camerasList)
            titleList.add(cm.getTitle());


        if (camerasList.size() > 0)
            textViewMyCamerasEmpty.setVisibility(View.INVISIBLE);

        recyclerView = rootView.findViewById(R.id.recycler_view_my_cameras);
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));


        adapter = new MyAdapter(titleList,  itemClickListener: this,  buttonClickListener: this,  itemLongClickListener: this);
        recyclerView.setAdapter(adapter);


    }

    1 usage
    @Override
    public void onItemClick(int position) {
        Intent intent = new Intent(getContext(), CameraViewerActivity.class);
        intent.putExtra( name: "open_mode", OpenVideoActivityMode.FROM_MY_CAMERAS);
        intent.putExtra( name: "camera_title", camerasList.get(position).getTitle());
        intent.putExtra( name: "camera_address", camerasList.get(position).getIpAddress());
        startActivity(intent);
    }

    1 usage
    @Override
    public void onButtonClick(int position) {

        Intent intent = new Intent(getContext(), CameraSettingActivity.class);
        intent.putExtra( name: "mode", SettingMode.CHANGE);
        intent.putExtra( name: "link", camerasList.get(position).getIpAddress());
        intent.putExtra( name: "title", camerasList.get(position).getTitle());
        intent.putExtra( name: "index", position);
        startActivity(intent);
    }
```

Рисунок 3.13 – Вміст файлу MyCamerasFragment

```
94      }
95
        1 usage
96      @Override
97      public void onItemLongClick(int position) { showDeleteItemAlert(position); }
100
        1 usage
101     private void showDeleteItemAlert(int index) {
102         AlertDialog.Builder builder = new AlertDialog.Builder(requireContext());
103         builder.setTitle("Delete")
104                 .setMessage("Delete item?")
105                 .setCancelable(false)
106                 .setPositiveButton( text: "Yes", new DialogInterface.OnClickListener() {
107                     @Override
108                     public void onClick(DialogInterface dialog, int which) {
109                         deleteItemByIndex(index);
110
111                         dialog.cancel();
112                     }
113                 })
114                 .setNegativeButton( text: "No", new DialogInterface.OnClickListener() {
115                     @Override
116                     public void onClick(DialogInterface dialog, int which) { dialog.cancel(); }
119                 });
120
121         AlertDialog dialog = builder.create();
122         dialog.show();
123     }
124
        1 usage
125     private void deleteItemByIndex(int index) {
126         camerasList.remove(index);
127         titleList.remove(index);
128         adapter.notifyItemRemoved(index);
129
130         FileService.saveDataInJson(getContext(), camerasList, FileService.CAMERAS_FILE,  rewrite: true);
131
132     }
133 }
```

Рисунок 3.14 – Продовження вмісту файлу MyCamerasFragment

Розробимо декілька допоміжних звичайних енамів для більш приємної подальшої взаємодії з кодом. У данному випадку було створено файли OpenVideoActivityMode і SettingMode, з якими детально можна ознайомитись у додатку Б.

Далі створимо адаптер MyAdapter для RecycleView для обробки подій, натискань кнопок та налаштувань різного опціоналу елементів, наприклад, для списку збережених посилань на камери (див. рис. 3.15–3.16).

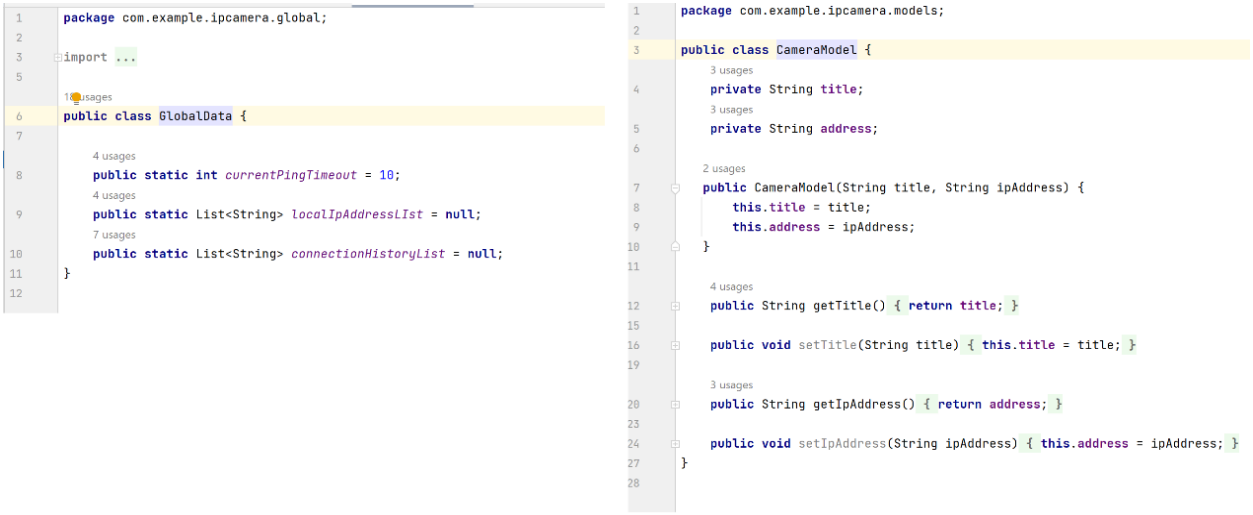
```java
package com.example.ipcamera.adapters;

import ...

7 usages
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {

    3 usages
    private List<String> dataList;
    4 usages
    private OnItemClickListener itemClickListener;
    3 usages
    private OnButtonClickListener buttonClickListener;
    2 usages
    private OnItemLongClickListener itemLongClickListener;

    3 usages  1 implementation
    public interface OnItemClickListener {
        1 usage  1 implementation
        void onItemClick(int position);
    }

    3 usages  1 implementation
    public interface OnButtonClickListener {
        1 usage  1 implementation
        void onButtonClick(int position);
    }

    3 usages  1 implementation
    public interface OnItemLongClickListener {
        1 usage  1 implementation
        void onItemLongClick(int position);
    }

    1 usage
    public MyAdapter(List<String> dataList,
                     OnItemClickListener itemClickListener,
                     OnButtonClickListener buttonClickListener,
                     OnItemLongClickListener itemLongClickListener) {
        this.dataList = dataList;
        this.itemClickListener = itemClickListener;
        this.buttonClickListener = buttonClickListener;
        this.itemLongClickListener = itemLongClickListener;
    }

    4 usages
    public static class ViewHolder extends RecyclerView.ViewHolder {
        2 usages
        public TextView textView;
        public Button button;

        1 usage
        public ViewHolder(View itemView) {
            super(itemView);
            textView = itemView.findViewById(R.id.text_view_item_my_cameras_list);
            button = itemView.findViewById(R.id.button_item_setting);
        }
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.button_setting_my_cameras, parent,  attachToRoot: false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, @SuppressLint("RecyclerView") int position) {
        String data = dataList.get(position);
        holder.textView.setText(data);

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int clickedPosition = holder.getAdapterPosition();
                if (itemClickListener != null) {
                    itemClickListener.onItemClick(clickedPosition);
                }
            }
        });
```

Рисунок 3.15 – Вміст файлу MyAdapter

```
78          });
79
80          holder.itemView.setOnLongClickListener(new View.OnLongClickListener() {
81              @Override
82              public boolean onLongClick(View v) {
83                  int clickedPosition = holder.getAdapterPosition();
84                  long id = holder.getItemId();
85
86
87                  if (itemClickListener != null) {
88                      itemLongClickListener.onItemLongClick(position);
89                  }
90                  return true;
91              }
92          });
93
94          holder.button.setOnClickListener(new View.OnClickListener() {
95              @Override
96              public void onClick(View v) {
97                  int clickedPosition = holder.getAdapterPosition();
98                  if (buttonClickListener != null) {
99                      buttonClickListener.onButtonClick(clickedPosition);
100                 }
101             }
102         });
103     }
104
105
106     @Override
107     public int getItemCount() { return dataList.size(); }
110 }
```

Рисунок 3.16 – Продовження вмісту файлу MyAdapter

Далі розробимо допоміжний клас GlobalData для більш простої взаємодії данних між різними Activity. А також розробимо у моделях сутність камери CameraModel для збереження списку камер (див. рис. 3.17).



```java
package com.example.ipcamera.global;

import ...

public class GlobalData {

    public static int currentPingTimeout = 10;
    public static List<String> localIpAddressList = null;
    public static List<String> connectionHistoryList = null;
}
```

```java
package com.example.ipcamera.models;

public class CameraModel {
    private String title;
    private String address;

    public CameraModel(String title, String ipAddress) {
        this.title = title;
        this.address = ipAddress;
    }

    public String getTitle() { return title; }

    public void setTitle(String title) { this.title = title; }

    public String getIpAddress() { return address; }

    public void setIpAddress(String ipAddress) { this.address = ipAddress; }
}
```

Рисунок 3.17 – Вміст файлів GlobalData та CameraModel відповідно

Далі розробимо сервіси. Нам необхідний сервіс для запису та отримання імформації з/у файл. Також, сервіс для серіалізації/десеріалізації json CameraModel(). І сервіс для сканування локальних ip адрес. Створимо файли FileService, JsonConverterService, LocalNetworkService та наповнимо їх необхідним функціоналом (див. рис. 3.18–3.20).

```java
package com.example.ipcamera.services;

import ...

20 usages
public class FileService {
    3 usages
    public static final String HISTORY_FILE = "history.txt";
    5 usages
    public static final String CAMERAS_FILE = "cameras.txt";

    3 usages
    static public void saveData(Context context, List<String> dataList, String fileName, boolean rewrite) {
        try (FileOutputStream fileOutputStream = context.openFileOutput(fileName, (rewrite) ? Context.MODE_PRIVATE : Context.MODE_APPEND)) {

            for (String data : dataList) {
                fileOutputStream.write((data + System.lineSeparator()).getBytes());
            }

        } catch (IOException e) {
            Toast.makeText(context, text: "File error", Toast.LENGTH_SHORT).show();
            e.printStackTrace();
        }
    }

    2 usages
    static public List<String> readData(Context context, String fileName) {
        try(FileInputStream fileInputStream = context.openFileInput(fileName)) {
            InputStreamReader streamReader = new InputStreamReader(fileInputStream);
            BufferedReader bufferedReader = new BufferedReader(streamReader);

            List<String> data = new ArrayList<>();
            String line = "";

            while ((line = bufferedReader.readLine()) != null) {
                if (!line.isEmpty())
                    data.add(line);
            }

            return data;

        } catch (IOException e) {
            e.printStackTrace();
        }

        return null;
    }

    3 usages
    public static void saveDataInJson(Context context, List<CameraModel> cameraModelList, String fileName, boolean rewrite

        List<String> jsonList = new ArrayList<>();

        for (CameraModel camera : cameraModelList) {
            jsonList.add(JsonConverterService.convertToJson(camera.getTitle(), camera.getIpAddress()));
        }

        saveData(context, jsonList, fileName, rewrite);
    }

    2 usages
    public static List<CameraModel> readDataFromJson(Context context, String fileName) {
        List<String> jsonLists = readData(context, fileName);

        List<CameraModel> camerasList = new ArrayList<>();

        if (jsonLists == null)
            return camerasList;

        for (String json : jsonLists) {
            camerasList.add(JsonConverterService.convertFromJson(json));
        }

        return camerasList;
    }

    1 usage
    public static void cleanFile(Context context, String fileName) {

        saveData(context, new ArrayList<String>(Collections.singletonList("")), fileName, rewrite: true);
```

Рисунок 3.18 – Вміст файлу FileService

```java
package com.example.ipcamera.services;

import ...

2 usages
public class JsonConverterService {
    1 usage
    public static String convertToJson(String title, String address) {
        JsonObject jsonObject = new JsonObject();
        jsonObject.addProperty( property: "title", title);
        jsonObject.addProperty( property: "address", address);

        Gson gson = new Gson();
        return gson.toJson(jsonObject);

    }

    1 usage
    public static CameraModel convertFromJson(String json) {
        Gson gson = new Gson();

        return gson.fromJson(json, CameraModel.class);
    }
}
```

Рисунок 3.19 – Вміст файлу JsonConventerService

```java
package com.example.ipcamera.services;

import ...

2 usages
public class LocalNetworkService {

    1 usage
    private static String getDeviceIPAddress() {
        try {
            Enumeration<NetworkInterface> interfaces = NetworkInterface.getNetworkInterfaces();
            while (interfaces.hasMoreElements()) {
                NetworkInterface networkInterface = interfaces.nextElement();
                Enumeration<InetAddress> addresses = networkInterface.getInetAddresses();
                while (addresses.hasMoreElements()) {
                    InetAddress address = addresses.nextElement();
                    if (!address.isLoopbackAddress() && address.isSiteLocalAddress() && Objects.requireNonNull(address.getHostAddress()).contains(".")) {
                        return address.getHostAddress();
                    }
                }
            }
        } catch (SocketException e) {
            e.printStackTrace();
        }
        return null;
    }

    1 usage
    public static List<String> getAllActiveLocalIp(int timeout) {
        List<String> ipAddressList = new ArrayList<>();

        try {
            String localIP = getDeviceIPAddress();

            if (localIP == null)
                throw new Exception("Local IP Addresses not found");

            String subnet = localIP.substring(0, localIP.lastIndexOf( ch: '.'));

            for (int i = 1; i <= 254; i++) {


                String ipAddress = subnet + "." + i;

                InetAddress inetAddress = InetAddress.getByName(ipAddress);

                if (!localIP.equals(inetAddress.toString().replace( target: "/",  replacement: "")) && inetAddress.isReachable(timeout)) {
                    ipAddressList.add(ipAddress);
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }

        return ipAddressList;
    }
}
```

Рисунок 3.20 – Вміст файлу LocalNetworkService

3.2 Опис та взаємодія зі скомпільованим додатком в телефоні

Якщо все було зроблено вірно, сміливо компілюємо наш додаток натиснувши на кнопку Run app. Після компіляції нам відкриється головне меню додатка (див. рис. 3.21).

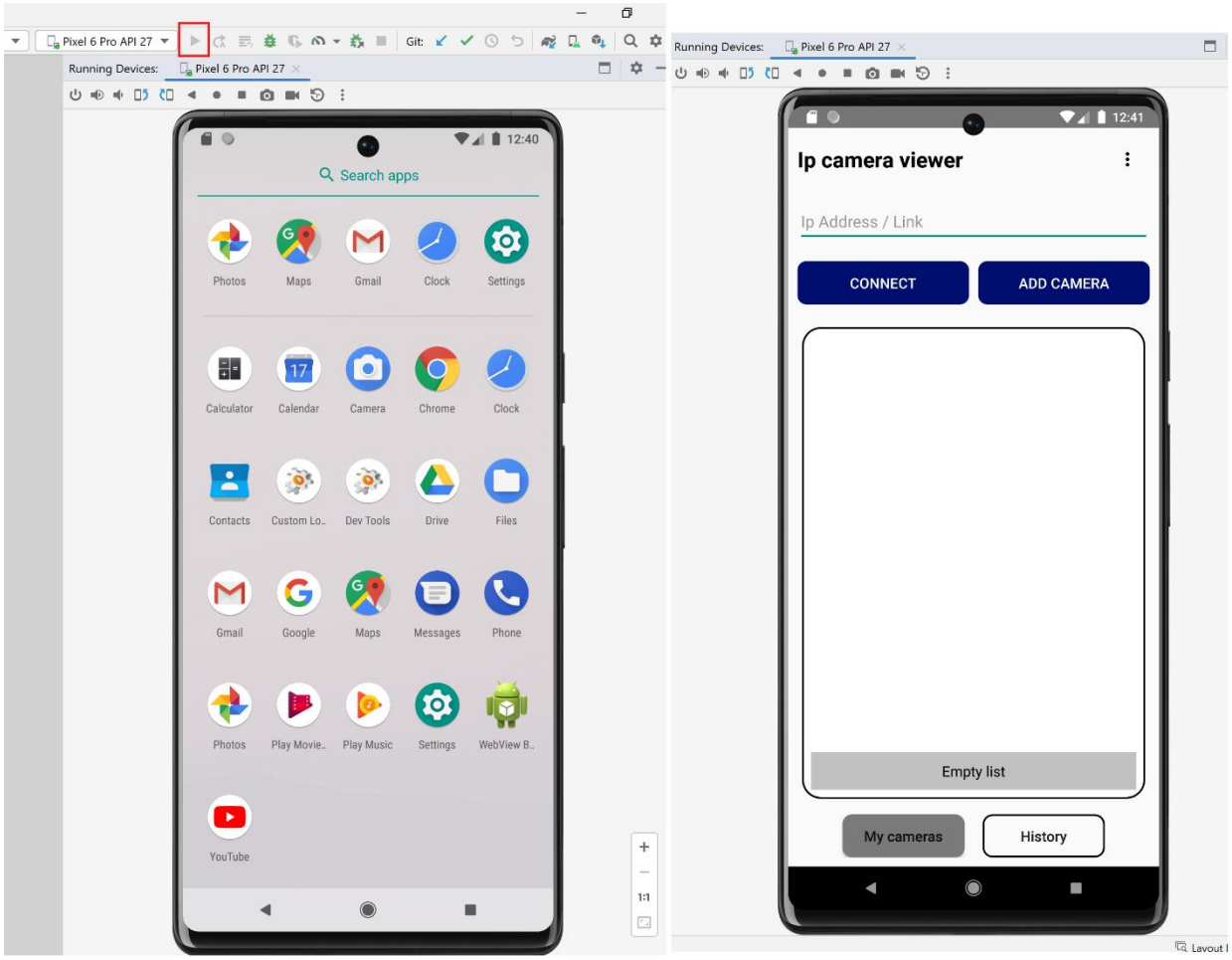


Рисунок 3.21 – Коректний результат компіляції додатку

Можна побачити назву, поле для вводу нового посилання, функціональні кнопки Connect та Add Camera, список збережених камер(наразі порожній), та кнопку History, яка перенесе нас до списку введених у додаток посилань. Розглянемо кнопку Connect. Спочатку введемо посилання на нашу камеру у поле Ip Address/Link і натиснемо Connect (див. рис. 3.22).

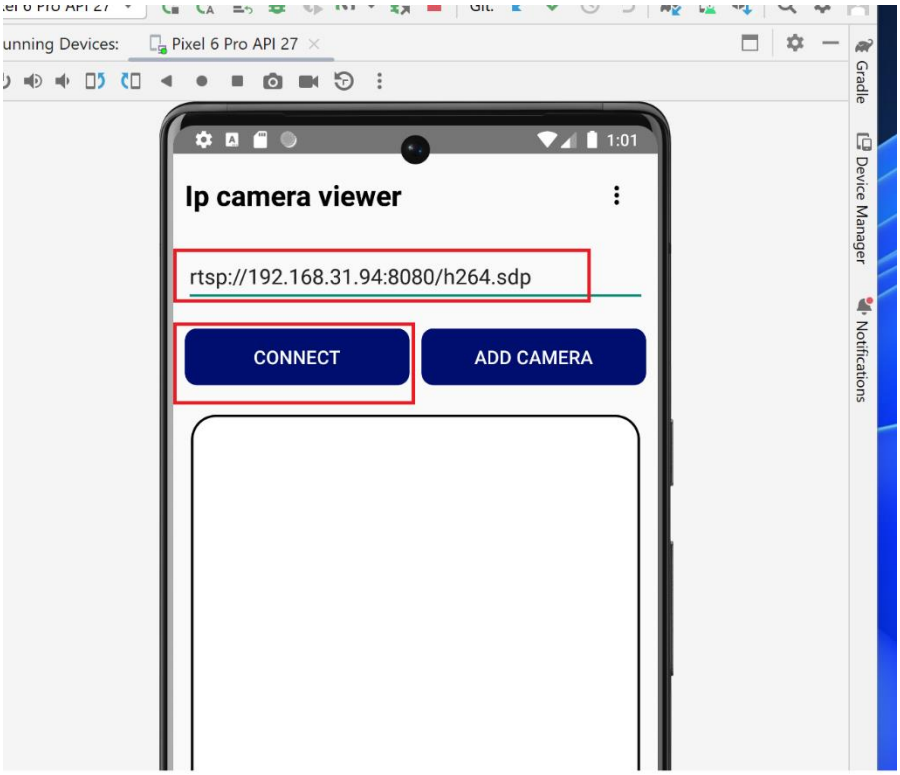


Рисунок 3.22 – Введення посилання на камеру в додаток

Далі нам відкриється екран з вікном відео та кнопками Play, Stop, Fullscreen, Save, Back. Наразі натиснемо на Fullscreen (див. рис. 3.23).

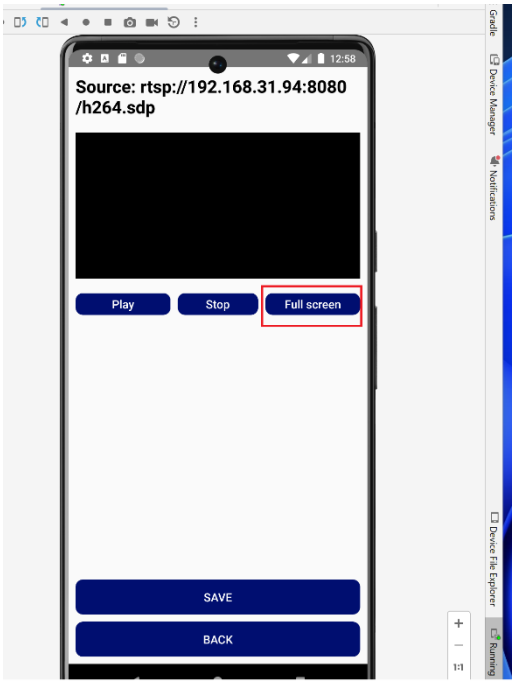


Рисунок 3.23 – Вікно перегляду та опцій відео

Через декілька секунд затримки з’явиться потокове відео з нашої камери. Можна натискати на символ, що виведе нас на попереднє вікно (див. рис. 3.24).

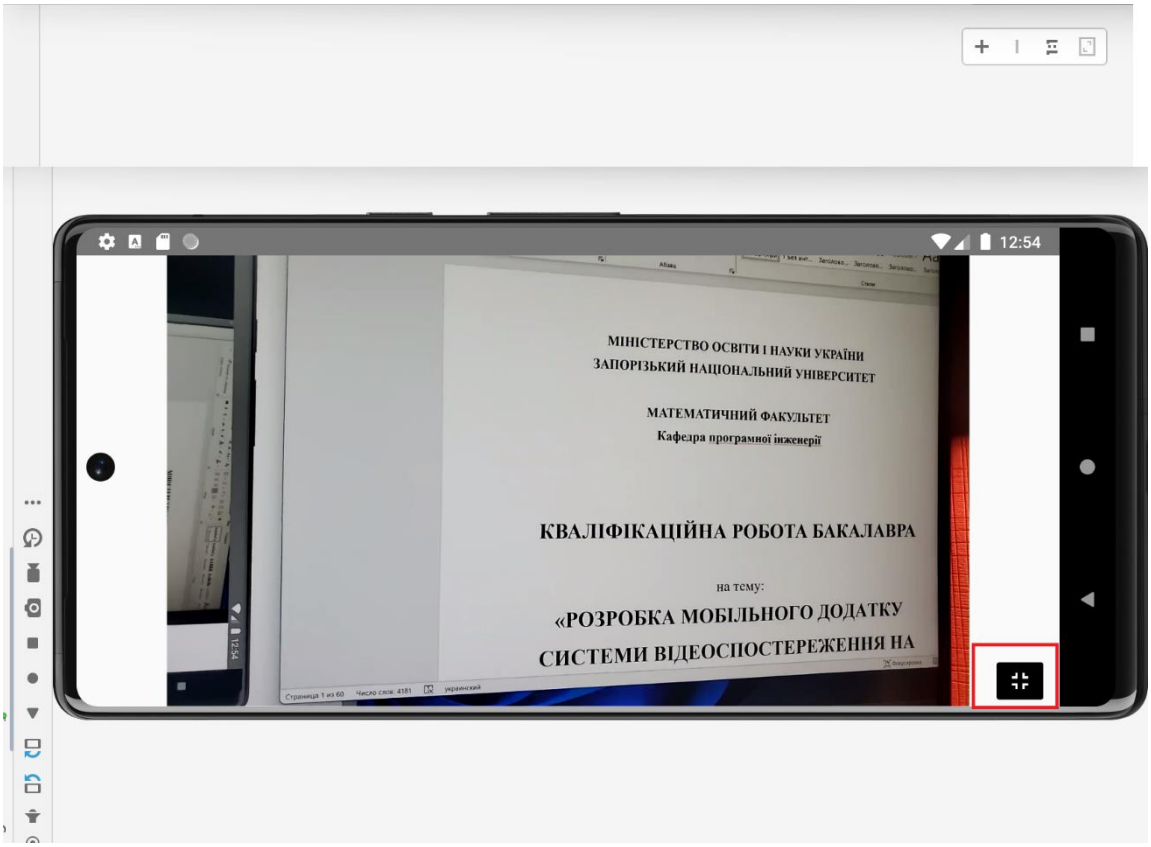


Рисунок 3.24 – Fullscreen режим відео

Можна використовувати як такі посилання, так і наприклад, посилання на потокове відео з мережі(публічні веб-камери, прямі трансляції передач). Є можливість навіть переробити його на аналог додатку перегляду онлайн ТБ.

Далі спробуємо зберегти посилання на камеру та дати йому назву. Натиснемо на кнопку Save та перейдемо у меню збереження назви камери та посилання. Дамо назву Camera 1 та натиснемо знову Save. Після цього посилання отримає назву Camera 1 та отримає шорткат на екрані головного меню (див.рис. 3.25). Кнопка Add Camera у головному меню має аналогічний функціонал:

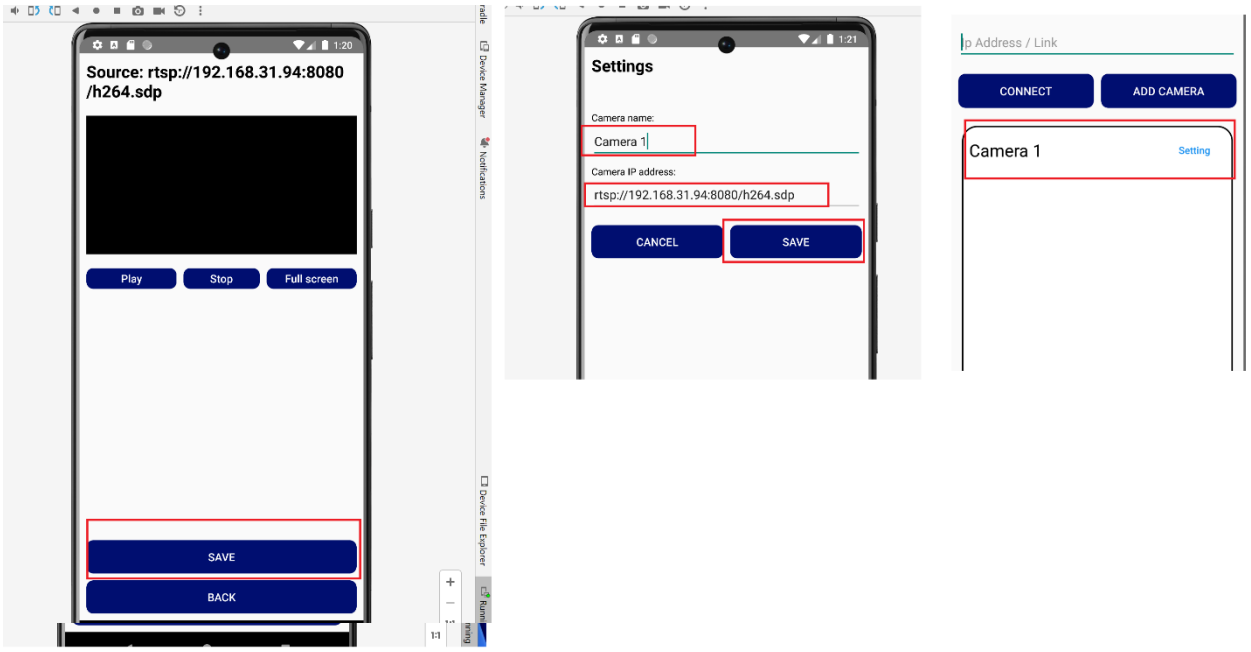


Рисунок 3.25 – Процес збереження посилання

У подальшому, при швидкому тапі на посилання буде відкриватися екран перегляду відео і опціоналу, а при довгому є можливість видалення збереженого посилання (див. рис. 3.26–3.27).

Рисунок 3.26 – Швидкий тап по збереженому посиланню Camera 1

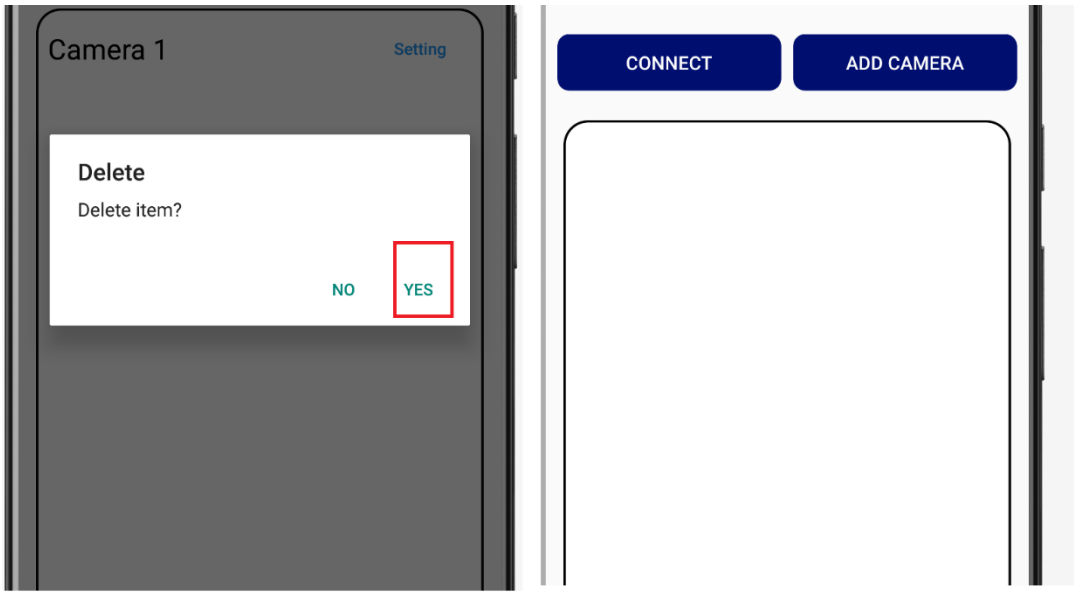


Рисунок 3.27 – Довгий тап для видалення збереженого посилання Camera 1

У додатку реалізована можливість сканування локальної мережі для можливості виявлення api підключених до неї камер. Для цього необхідно натиснути на символ у правому верхньому кутку та обрати кнопку Local Ip. Після цього можна налаштувати пінг сканування натиснувши на коліщатко у верхньому правому кутку, запустити сканування або повернутись у головне меню натиснувши кнопку Back (див. рис. 3.28).

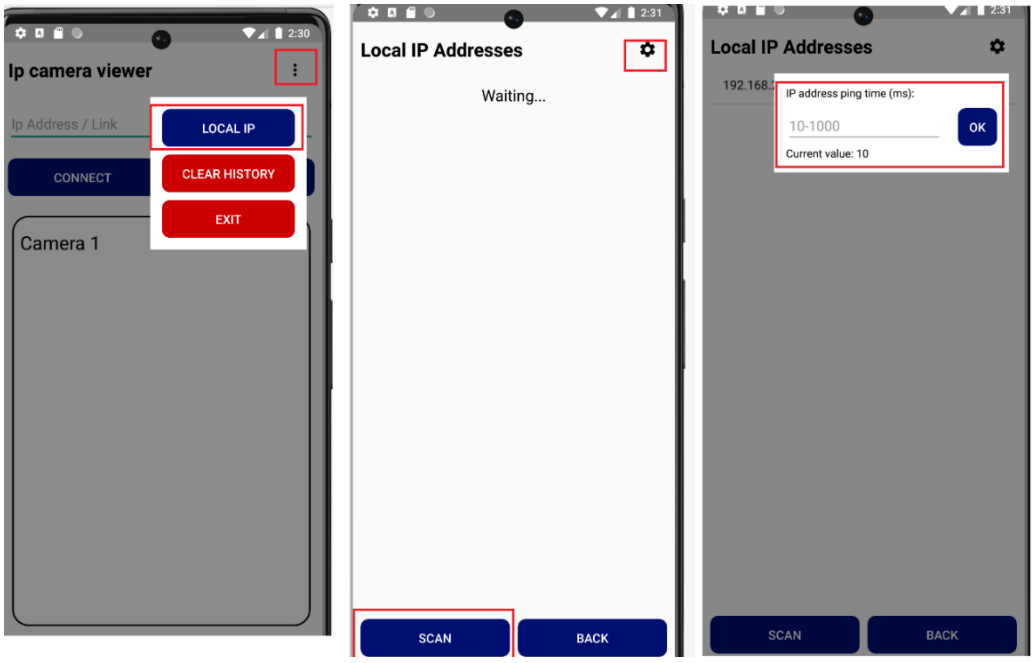


Рисунок 3.28 – Можливості взаємодії з скануванням локальної мережі

Також, у додатку присутня історія введених посилань. Для взаємодії з нею достатньо у головному меню натиснути на кнопку History. Нас перекине до списку з усіма введеними нами коли-небуть посиланнями (див. рис. 3.29).

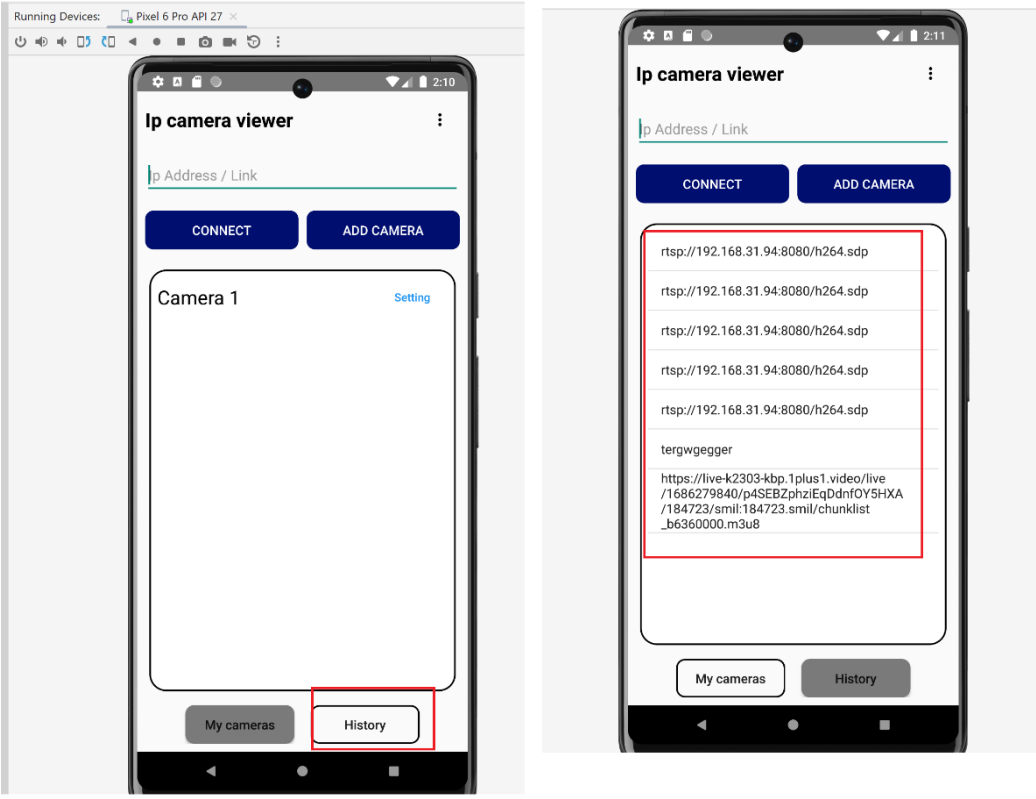


Рисунок 3.29 – Історія введених посилань у додатку

Після цього ми бачимо список з посиланнями. Тап по будь-якому з них перекине посилання у графу Ip Address / Link для можливості повторного підключення. Також, є можливість видалити історію, натиснувши на символ у верхньому правому кутку та обравши кнопку Clear History. Кнопка My Cameras поверне нас у головне меню з списком збережених нами посилань (див. рис. 3.30).

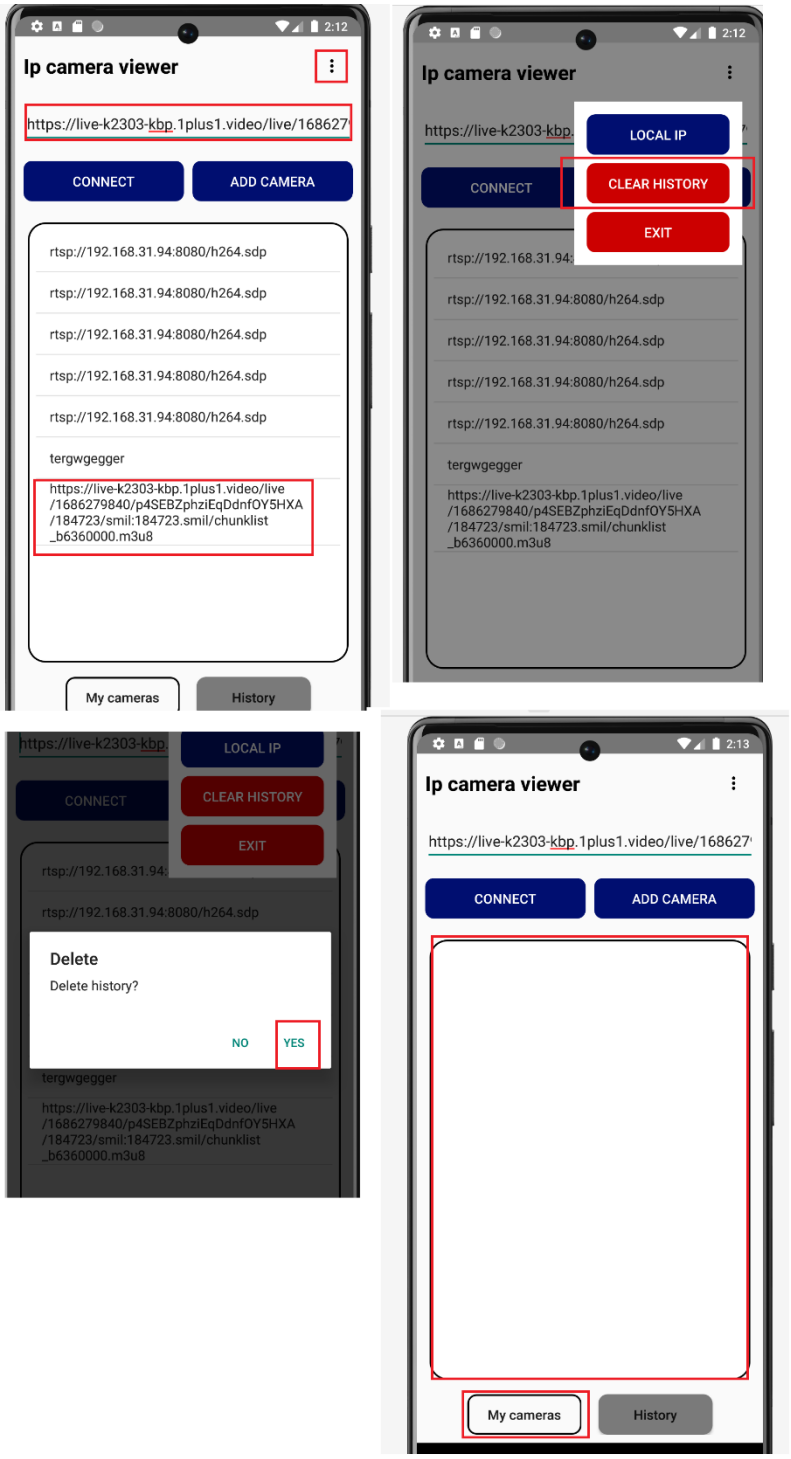


Рисунок 3.30 – Можливості взаємодії з історією посилань

ВИСНОВКИ

В ході кваліфікаційної роботи, згідно зібраним вимогам до майбутнього проекту, було спроектовано та розроблено мобільний додаток для організації системи відеоспостереження на основі NodeMCU з однієї або декількох підключених ip-камер до локальної мережі або за прямим посиланням на камеру(потокове відео). Реалізована можливість додавання, видалення, редагування, зберігання посилань на камери у додатку, а також сканування локальної мережі на наявність api камер, підключених до неї. Додаток створений на мобільній платформі Android, з використанням мови програмування Java та IDE Android Studio. Робота може бути використана у подальшому як приклад розробки схожих проектів або створення комерційного проекту з більшим функціоналом на його основі.

ПЕРЕЛІК ПОСИЛАНЬ

1. AlfredCamera. Official site. URL : https://alfred.camera (дата звернення 30.04.2023).

2. ZoomOn. Official site. URL : https://play.google.com/store/apps/details?id=cz.master.lois&hl=uk (дата звернення 30.04.2023).

3. Faceter. Official site. URL : https://faceter.cam (дата звернення 30.04.2023).

4. Little Stars. Official site. URL : https://play.google.com/store/apps/details?id=com.jxl.app.littlestars.project&hl=ru&gl=US (дата звернення 30.04.2023).

5. Android. Official site. URL : https://uk.wikipedia.org/wiki/Android (дата звернення 30.04.2023).

6. Arduino. Official site. URL : https://uk.wikipedia.org/wiki/Arduino (дата звернення 30.04.2023).

7. Arduino.ua. Official site. URL : https://doc.arduino.ua/ru/prog/ (дата звернення 30.04.2023).

8. Java. Official site. URL : https://uk.wikipedia.org/wiki/Java (дата звернення 30.04.2023).

9. Evegreen. Official site. URL : https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення 30.04.2023).

10. Mindonmap. Official site. URL : https://www.mindonmap.com/uk/blog/uml-component-diagram/ (дата звернення 30.04.2023).

11. Make-it.ca. Official site. URL : https://www.make-it.ca/nodemcu-details-specifications/ (дата звернення 30.04.2023).

12. Element14. Official site. URL : https://community.element14.com/products/devtools/product-pages/w/documents/22985/ttl-serial-jpeg-camera-with-ntsc-video (дата звернення 30.04.2023).

13. PTC06 Serial Camera Specification. Official site. URL : https://cdn-shop.adafruit.com/product-files/1386/1386+v2+datasheet.pdf (дата звернення 30.04.2023).

14. Arduino Developer Documentation. Official site. URL : www.arduino.cc (дата звернення 30.04.2023).

15. WinRar. Official site. URL : https://www.win-rar.com/start.html?&L=4 (дата звернення 30.04.2023).

16. 7-zip. Official site. URL : https://www.7-zip.org/download.html (дата звернення 30.04.2023).

17. ESP8266 Arduino Core. Official site. URL : https://arduino-esp8266.readthedocs.io/en/latest/installing.html (дата звернення 30.04.2023).

18. NanjingQinhengMicroelectronics. Official site. URL : http://www.wch-ic.com/downloads/CH341SER_EXE.html (дата звернення 30.04.2023).

19. Android Studio. Official site. URL : https://developer.android.com/studio (дата звернення 30.04.2023).

20. Блох Д. Java : ефективне програмування. Київ : Науковий Світ, 2022. 464 с.

21. Мартін Р. С. Чистий код. Київ : Фабула, 2019. 368 с.

22. Сьерра К., Бейтс Б. Head First. Київ : Фабула, 2022. 720 с.

23. Оакс С. Продуктивність Java : Повний посібник. Каліфорнія : O’Reilly, 2014. 426 с.

24. Манетті Л., Джуней Дж. Рецепти Java 17 : підхід «проблема-рішення». Нью-Йорк : Apress, 2022. 628 с.

ДОДАТОК А

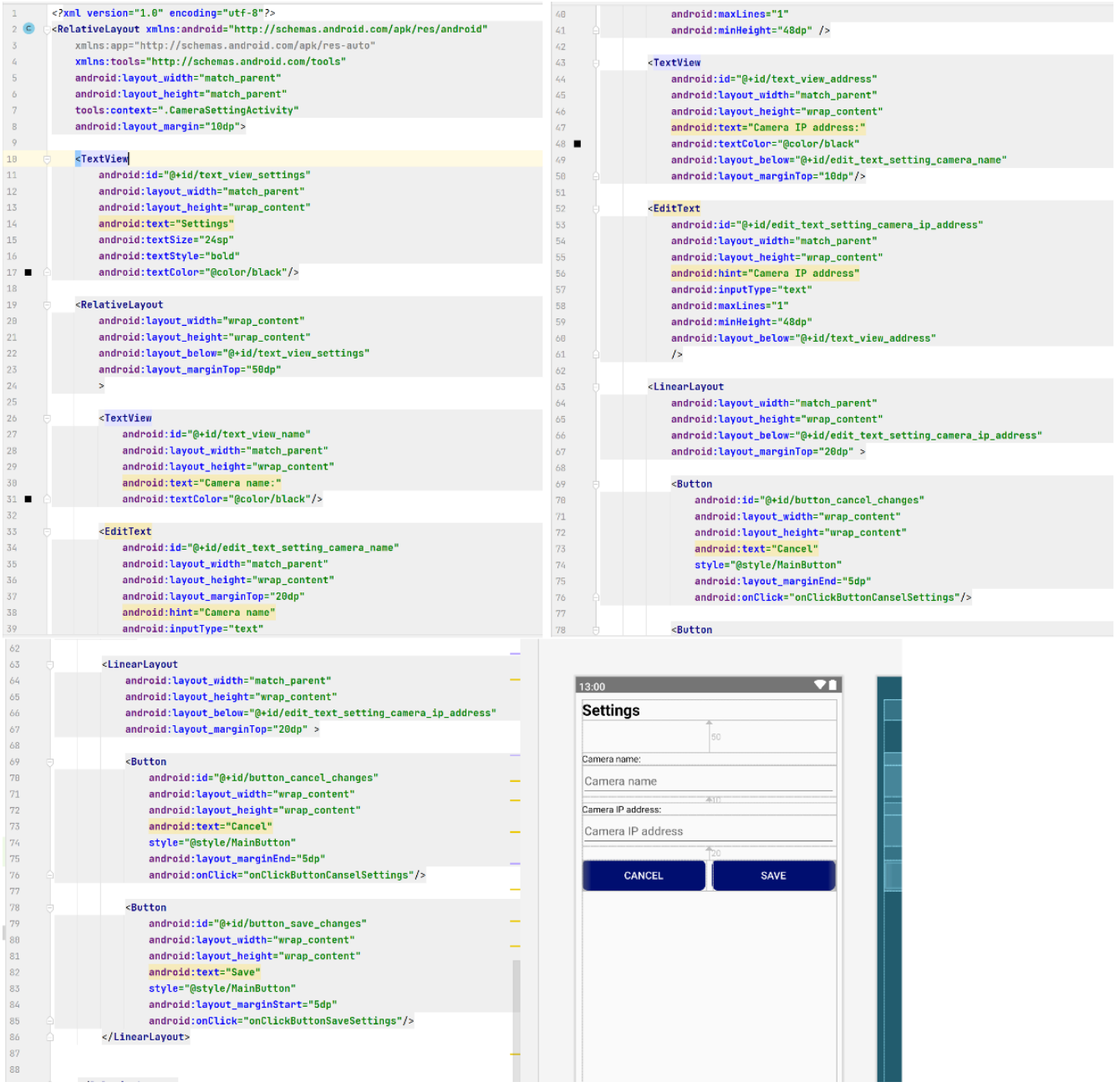
Скетч для підключення

```
const char *ssid = "xxx";
const char *password = "xxx";
WiFiClient client;
WiFiClient dclient;
File fh;   // SPIFFS file handle
char FTPserver[] = "xxx"; // name of FTP server
String fileName = "picture.jpg";
String  path = "/picture.jpg";
const boolean debug = false;  // true = messages about FTP on Serial (conflicts with camera on Serial)
char outBuf[128];
char outCount;
int counter = 0;   // for FTP filename
const int ftp_timeout = 10;   // timeout in seconds for FTP replies
const boolean PTC06 = false;
byte incomingbyte;
int a = 0x0000, //Read Starting address
    j = 0,
    k = 0,
    count = 0,
    sendcount = 0;
uint8_t MH, ML;
boolean EndFlag = 0;
const int cmd_delay = 1000;
const int capture_timeout = 60;   // timeout in seconds -- should be enough to read/save large picture (45 sec at baud 38400)
const int LED_PIN = 5;
const unsigned long main_interval = 60;  // seconds between pics
unsigned long main_timestamp;
boolean first = true;   // to take picture in first loop, not waiting for interval

void setup() {
  pinMode(LED_PIN, OUTPUT);

  WiFi.begin ( ssid, password );
  while ( WiFi.status() != WL_CONNECTED ) {  // blink LED while connecting to wifi
    delay (500);
    digitalWrite(LED_PIN, HIGH);
    delay(100);
    digitalWrite(LED_PIN, LOW);
  }

  if (PTC06) {
    Serial.begin(115200);
  } else
  {
    Serial.begin (38400);
  }
  SPIFFS.begin();

  delay(3000);   // camera needs time to start up
}

void loop() {

  yield();  // for wifi stack

  if (millis() - main_timestamp > main_interval * 1000L || first) {     // make sure first loop takes picture

    first = false;    // switch flag off after first picture
    picture();   // take picture and save to SPIFFS, LED will flash

    digitalWrite(LED_PIN, HIGH);                        // LED on during FTP
    if (!doFTP()) {                                   // send over FTP to cloud server, LED full on, try a second time if first fails
      delay(100);
      doFTP();
    }
    digitalWrite(LED_PIN, LOW);                         // LED off when FTP finished

    main_timestamp = millis();
    counter++;                      // for picture index
    if (counter == 10) counter = 0;
  }
}
```

Рисунок А.1 – Скетч для підключення плати

ДОДАТОК Б

Код Android-додатку



```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".CameraSettingActivity"
    android:layout_margin="10dp">

    <TextView
        android:id="@+id/text_view_settings"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Settings"
        android:textSize="24sp"
        android:textStyle="bold"
        android:textColor="@color/black"/>

    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/text_view_settings"
        android:layout_marginTop="50dp"
        >

        <TextView
            android:id="@+id/text_view_name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Camera name:"
            android:textColor="@color/black"/>

        <EditText
            android:id="@+id/edit_text_setting_camera_name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:hint="Camera name"
            android:inputType="text"
            android:maxLines="1"
            android:minHeight="48dp" />

        <TextView
            android:id="@+id/text_view_address"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Camera IP address:"
            android:textColor="@color/black"
            android:layout_below="@+id/edit_text_setting_camera_name"
            android:layout_marginTop="10dp"/>

        <EditText
            android:id="@+id/edit_text_setting_camera_ip_address"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Camera IP address"
            android:inputType="text"
            android:maxLines="1"
            android:minHeight="48dp"
            android:layout_below="@+id/text_view_address"
            />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@+id/edit_text_setting_camera_ip_address"
            android:layout_marginTop="20dp" >

            <Button
                android:id="@+id/button_cancel_changes"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Cancel"
                style="@style/MainButton"
                android:layout_marginEnd="5dp"
                android:onClick="onClickButtonCanselSettings"/>

            <Button
                android:id="@+id/button_save_changes"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Save"
                style="@style/MainButton"
                android:layout_marginStart="5dp"
                android:onClick="onClickButtonSaveSettings"/>
        </LinearLayout>
```


Рисунок Б.1 – Вміст файлу activity_camera_setting.xml

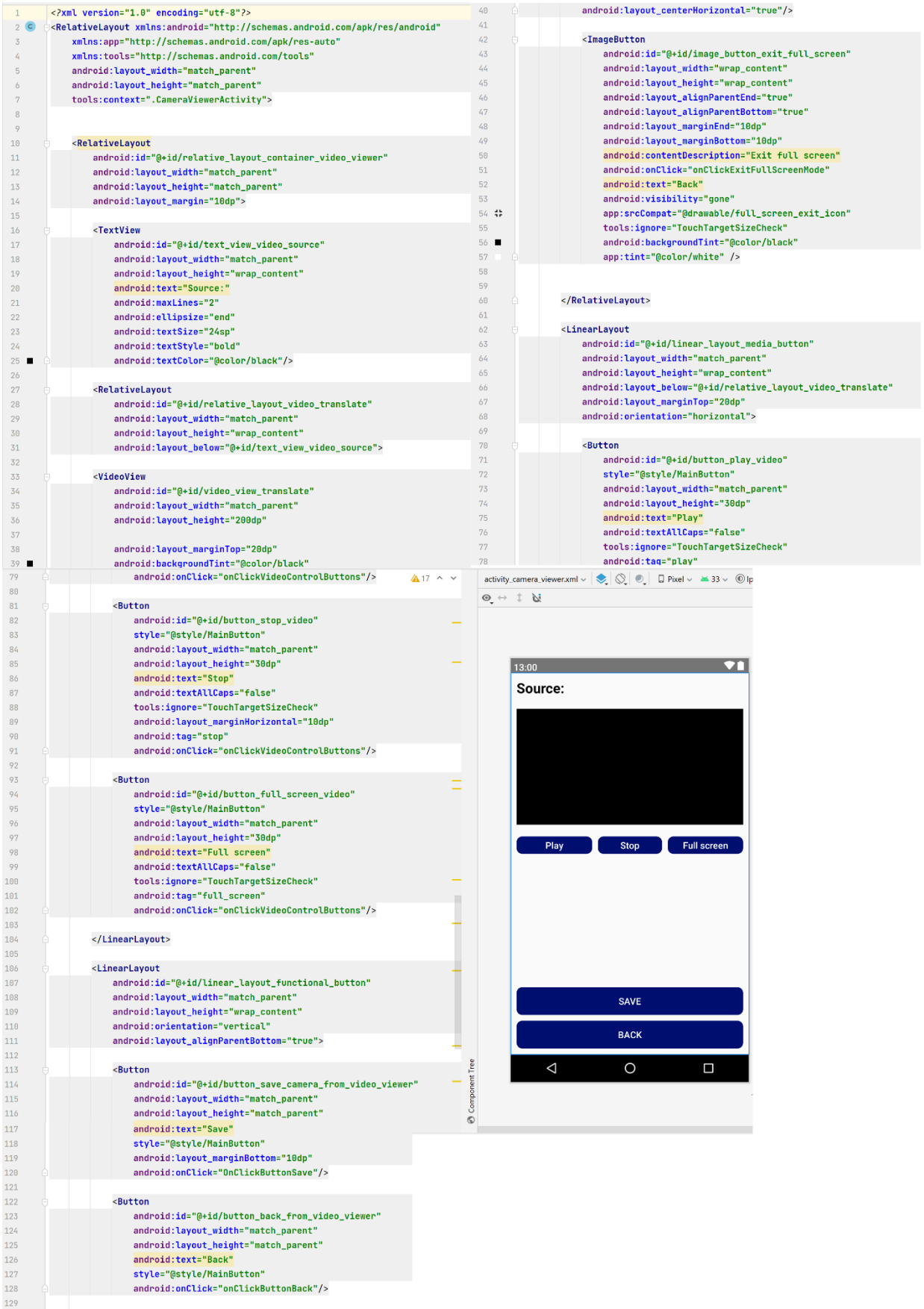
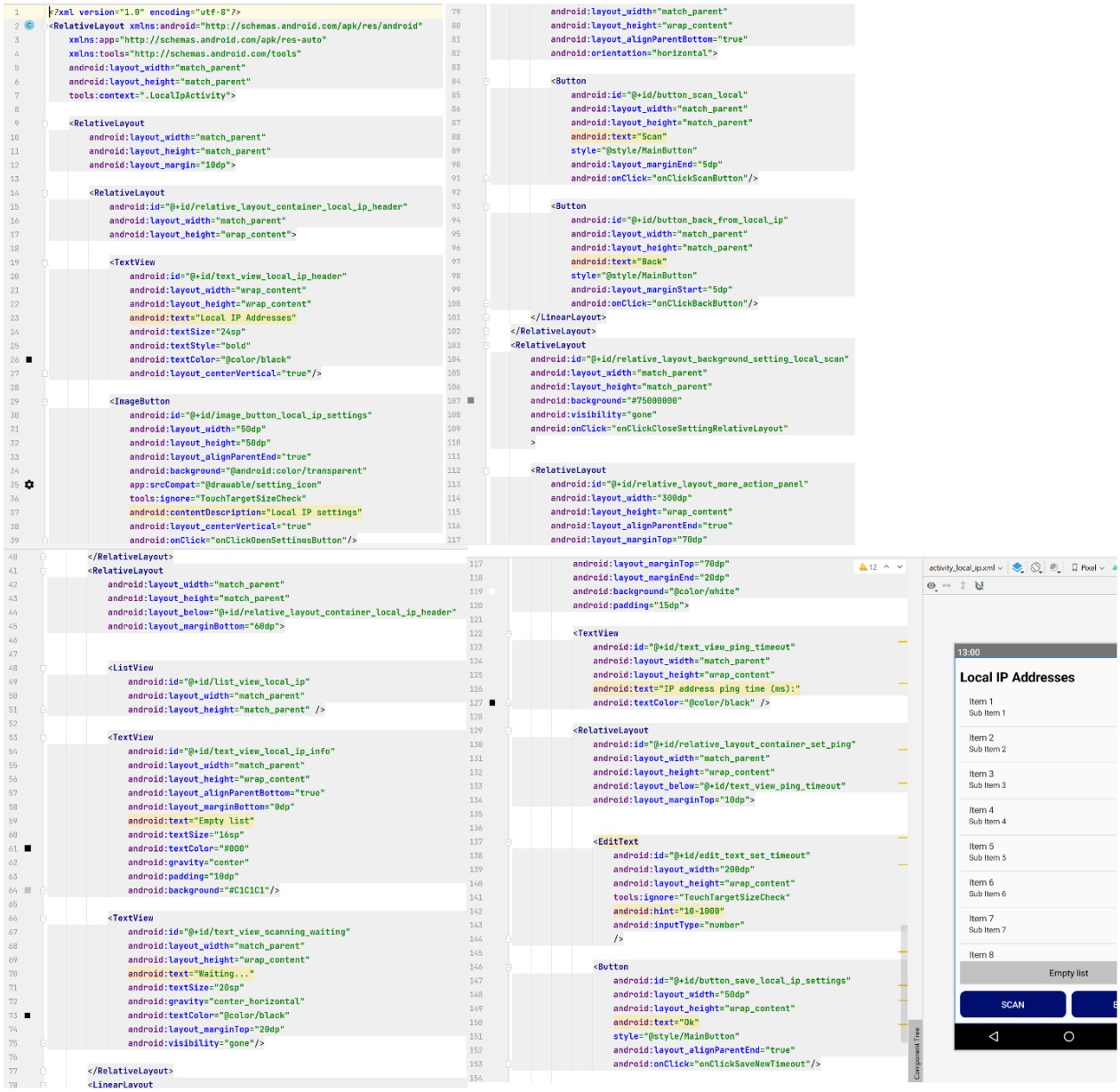


Рисунок Б.2 – Вміст файлу activity_camera_viewer.xml



```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".LocalIpActivity">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="10dp">

        <RelativeLayout
            android:id="@+id/relative_layout_container_local_ip_header"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TextView
                android:id="@+id/text_view_local_ip_header"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Local IP Addresses"
                android:textSize="24sp"
                android:textStyle="bold"
                android:textColor="@color/black"
                android:layout_centerVertical="true"/>

            <ImageButton
                android:id="@+id/image_button_local_ip_settings"
                android:layout_width="50dp"
                android:layout_height="50dp"
                android:layout_alignParentEnd="true"
                android:background="@android:color/transparent"
                app:srcCompat="@drawable/setting_icon"
                tools:ignore="TouchTargetSizeCheck"
                android:contentDescription="Local IP settings"
                android:layout_centerVertical="true"
                android:onClick="onClickOpenSettingsButton"/>
        </RelativeLayout>
        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_below="@+id/relative_layout_container_local_ip_header"
            android:layout_marginBottom="60dp">


            <ListView
                android:id="@+id/list_view_local_ip"
                android:layout_width="match_parent"
                android:layout_height="match_parent" />

            <TextView
                android:id="@+id/text_view_local_ip_info"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_alignParentBottom="true"
                android:layout_marginBottom="0dp"
                android:text="Empty list"
                android:textSize="16sp"
                android:textColor="#000"
                android:gravity="center"
                android:padding="10dp"
                android:background="#C1C1C1"/>

            <TextView
                android:id="@+id/text_view_scanning_waiting"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Waiting..."
                android:textSize="20sp"
                android:gravity="center_horizontal"
                android:textColor="@color/black"
                android:layout_marginTop="20dp"
                android:visibility="gone"/>

        </RelativeLayout>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:orientation="horizontal">

            <Button
                android:id="@+id/button_scan_local"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Scan"
                style="@style/MainButton"
                android:layout_marginEnd="5dp"
                android:onClick="onClickScanButton"/>

            <Button
                android:id="@+id/button_back_from_local_ip"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Back"
                style="@style/MainButton"
                android:layout_marginStart="5dp"
                android:onClick="onClickBackButton"/>
        </LinearLayout>
    </RelativeLayout>
    <RelativeLayout
        android:id="@+id/relative_layout_background_setting_local_scan"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#75000000"
        android:visibility="gone"
        android:onClick="onClickCloseSettingRelativeLayout"
        >

        <RelativeLayout
            android:id="@+id/relative_layout_more_action_panel"
            android:layout_width="300dp"
            android:layout_height="wrap_content"
            android:layout_alignParentEnd="true"
            android:layout_marginTop="70dp"
            android:layout_marginTop="70dp"
            android:layout_marginEnd="20dp"
            android:background="@color/white"
            android:padding="15dp">

            <TextView
                android:id="@+id/text_view_ping_timeout"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="IP address ping time (ms):"
                android:textColor="@color/black" />

            <RelativeLayout
                android:id="@+id/relative_layout_container_set_ping"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/text_view_ping_timeout"
                android:layout_marginTop="10dp">


                <EditText
                    android:id="@+id/edit_text_set_timeout"
                    android:layout_width="200dp"
                    android:layout_height="wrap_content"
                    tools:ignore="TouchTargetSizeCheck"
                    android:hint="10-1000"
                    android:inputType="number"
                    />

                <Button
                    android:id="@+id/button_save_local_ip_settings"
                    android:layout_width="50dp"
                    android:layout_height="wrap_content"
                    android:text="Ok"
                    style="@style/MainButton"
                    android:layout_alignParentEnd="true"
                    android:onClick="onClickSaveNewTimeout"/>
```


Рисунок Б.3 – Вміст файлу activity_local_ip.xml

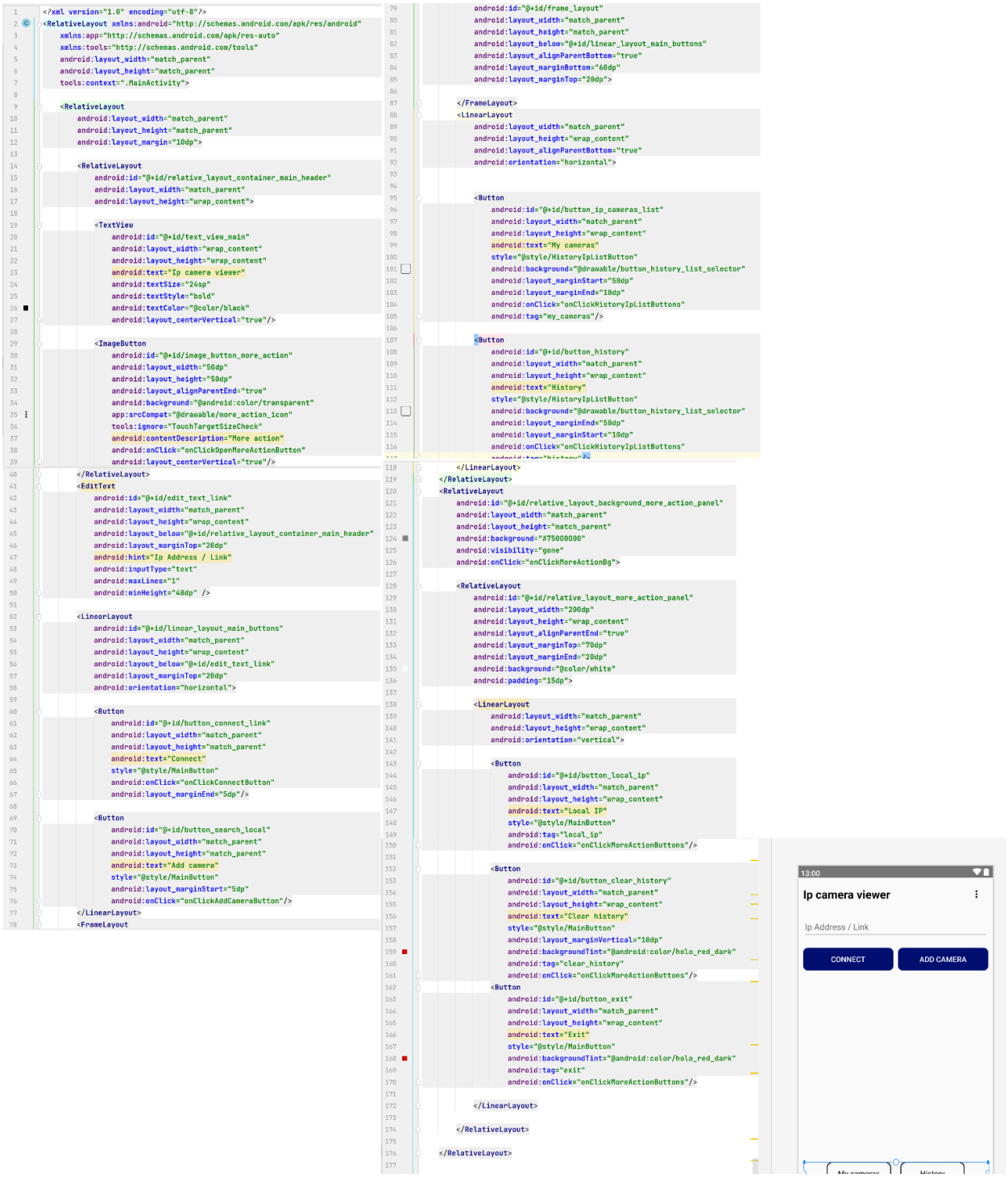


Рисунок Б.4 – Вміст файлу activity_main.xml

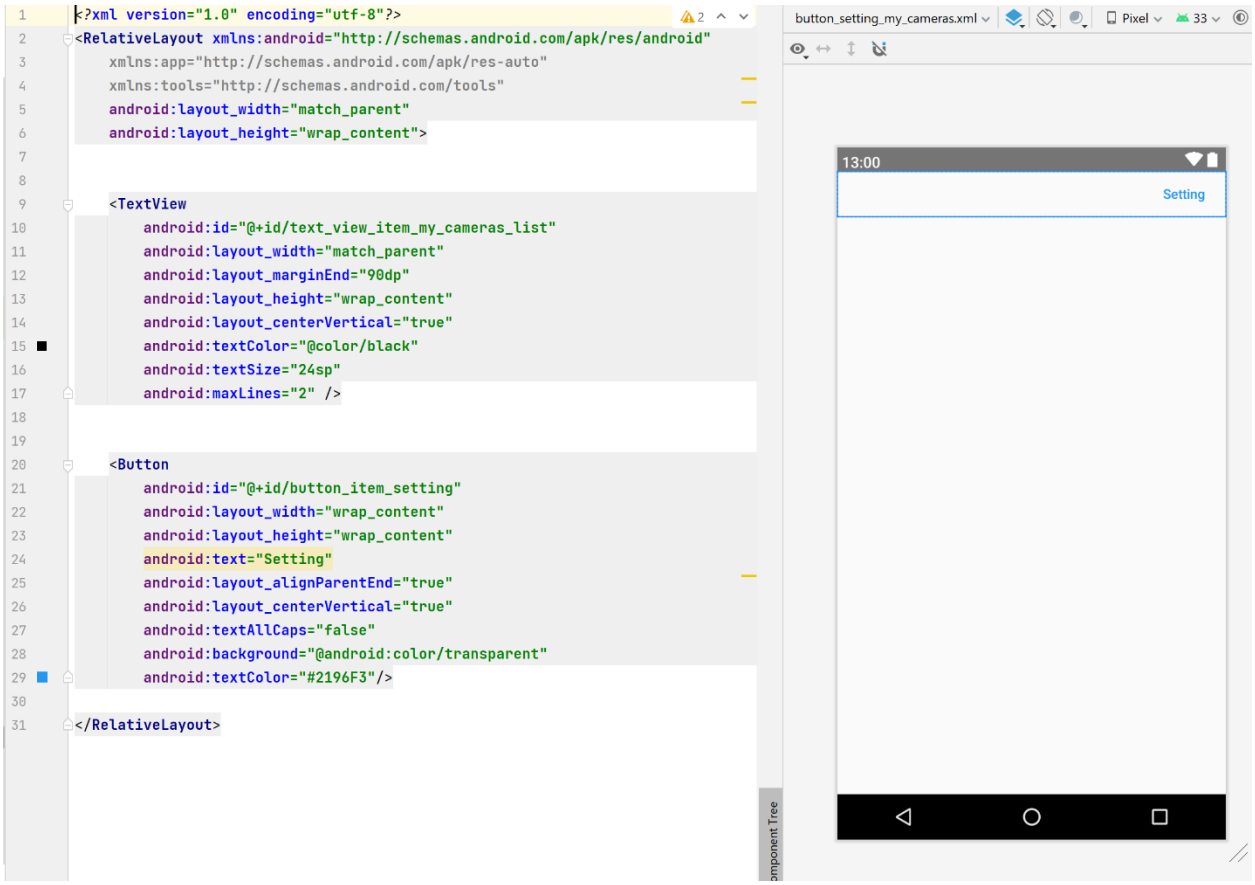


Рисунок Б.5 – Вміст файлу button_setting_my_cameras.xml

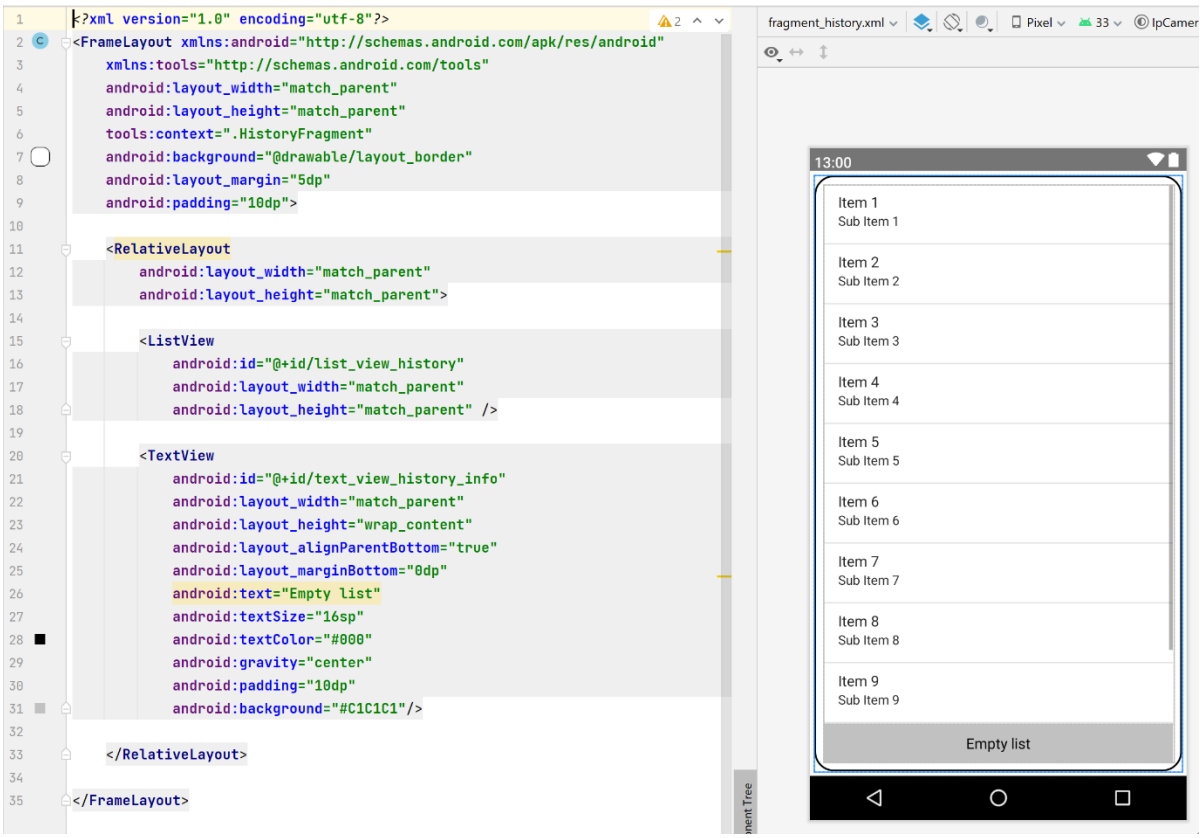
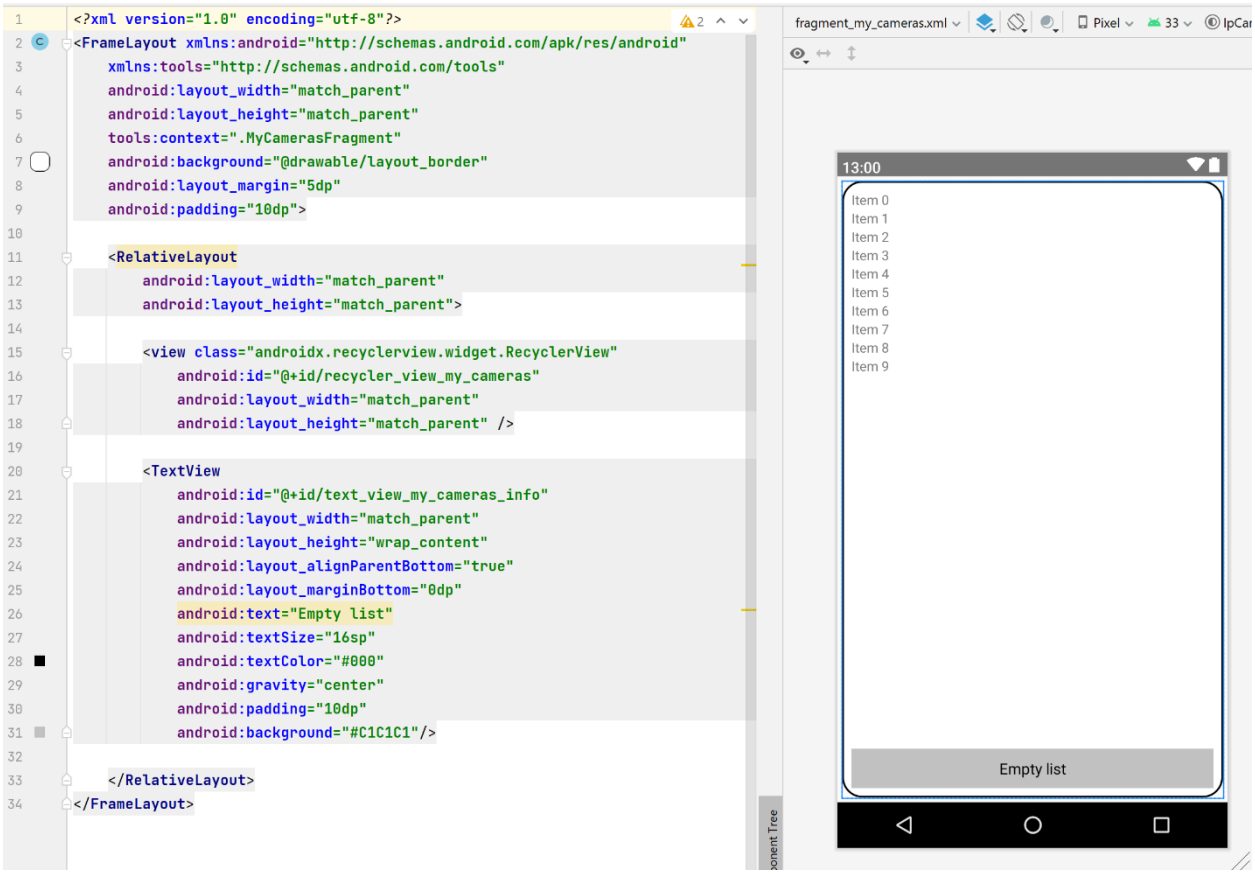


Рисунок Б.6 – Вміст файлу fragment_history.xml



```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MyCamerasFragment"
    android:background="@drawable/layout_border"
    android:layout_margin="5dp"
    android:padding="10dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <view class="androidx.recyclerview.widget.RecyclerView"
            android:id="@+id/recycler_view_my_cameras"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

        <TextView
            android:id="@+id/text_view_my_cameras_info"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_marginBottom="0dp"
            android:text="Empty list"
            android:textSize="16sp"
            android:textColor="#000"
            android:gravity="center"
            android:padding="10dp"
            android:background="#C1C1C1"/>

    </RelativeLayout>
</FrameLayout>
```


Рисунок Б.7 – Вміст файлу fragment_my_cameras.xml

```java
package com.example.ipcamera.enums;

public enum OpenVideoActivityMode {

    2 usages
    FROM_CONNECT, FROM_MY_CAMERAS, FROM_SCANNED_LIST;
}
```

```java
package com.example.ipcamera.enums;

public enum SettingMode {
    3 usages
    CHANGE, NEW;


}
```

Рисунок Б.8 – Вміст файлів OpenVideoActivityMode та SettingMode відповідно