Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня ___магістра___
(бакалавра, спеціаліста, магістра)

студентки _____Максимової Анжеліки Іванівни_____
(ПІБ)
академічної групи _____123м-19-1_____
(шифр)
спеціальності _____123 «Комп'ютерна інженерія»_____
(код і назва спеціальності)
за освітньо-професійною програмою ___«Комп'ютерна інженерія»___
(офіційна назва)
на тему «Комп'ютерна система відеоконтролю стрічкових конвеєрів вугільної шахти з детальним опрацюванням підсистеми проектування топології мережі»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Цвіркун Л.І.			
розділів:				
теоретичний розділ	проф. Цвіркун Л.І.			
синтез системи	доц. Ткаченко С.М.			
розроблення програмного забезпечення	ас. Бешта Л.В.			
експериментальний розділ	доц. Ткаченко С.М.			
Рецензент				
Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2020

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ __Гнатушенко В.В._
(підпис) (прізвище, ініціали)

«_ __»__ _____ 2020 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня ____магістр _____
(бакалавра, спеціаліста, магістра)

студенту _____Максимової А.І.____ **академічної групи** ____123м-19-1____
(прізвище та ініціали) (шифр)

спеціальності _____123 «Комп'ютерна інженерія»_____

за освітньою-професійною програмою _123 «Комп'ютерна інженерія» _
(офіційна назва)

на тему «Комп'ютерна система відеоконтролю стрічкових конвеєрів вугільної шахти з детальним опрацюванням підсистеми проектування топології мережі»,

затверджену наказом ректора НТУ «Дніпровська політехніка» від 22.10.2020 р. №888

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	21.09.2020
Теоретичний	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	30.10.2020
Синтез системи	Розробка комп'ютерної системи	12.11.2020
Розроблення програмного забезпечення	Розробка програмного забезпечення	26.11.2020
Експериментальний розділ	Проведення і обробка результатів експериментів	06.12.2020

Завдання видано _____ _проф. Цвіркун Л. І.
(підпис керівника) (прізвище, ініціали)

Дата видачі _07 вересня_2020 р._

Дата подання до екзаменаційної комісії ____10.12.2020 р.____

Прийнято до виконання _____ _Максимова А.І._
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 119 с., 25 рис., 3 табл., 19 пос., 1 додаток.

Об'єкт дослідження: комп’ютерна система відеоконтролю стрічкових конвеєрів вугільної шахти.

Мета: розробити комп’ютерну систему з програмним забезпеченням, яке спрощує процес проектування або модернізації топології CAN-мережі у вугільних шахтах.

У вступі показано актуальність, мету, завдання, об'єкт, предмет, ідею дослідження.

У розділі «Стан питання і постановка задачі» розглянута сфера застосування об'єкта дослідження. Проаналізовано існуючі алгоритми оптимізації топології мереж, особливості контролю стрічкових конвеєрів вугільних шахт. Сформульовано мету і завдання роботи.

У теоретичному розділі описана теорія графів для її застосування в даній роботі, проаналізовано підходи розробки моделей топології мережі із застосуванням дерева Штейнера, проаналізовано техніки оптимізації при трасуванні друкованих плат, обгрунтована структура бази даних.

У розділі «Синтез системи» проаналізовані складові компоненти системи, вибрано обладнання та розроблена структурна схема системи відеоконтролю.

У розділі «Розроблення програмного забезпечення» обоснована структура і методи реалізації програмного забезпечення, розроблено програмне забезпечення для моделювання топології CAN-мережі у вугільних шахтах.

В експериментальному розділі поставлена задача експерименту і обоснована методика його проведення та перевірена працездатність і ефективність розробки топології мережі із застосування дерева Штейнера.

ВУГІЛЬНА ШАХТА, ПРОЕКТУВАННЯ, CAN-МЕРЕЖА, ТОПОЛОГІЯ, ПРОГРАМА, МОДЕЛЮВАННЯ, ДЕРЕВО ШТЕЙНЕРА

ABSTRACT

Explanatory note: p.119, img. 23, tabl. 3, sources.19, appl. 1.

Object of research: computer video surveillance system of belt conveyors of coal mine lines.

Objective: to develop a computer system with software that simplifies the process of designing or upgrading the topology of the CAN network in coal mines.

The introduction shows the relevance, purpose, objectives, object, subject, idea of the study.

In the section "Status of the question and problem statement" the scope of the object of study is considered. The existing algorithms of optimization of network topology, features of control of belt conveyors of coal mines are analyzed. The purpose and tasks of work are formulated.

The theoretical section describes the theory of graphs for its application in this work, analyzes the approaches to developing network topology models using the Steiner tree, analyzes the optimization techniques for tracing printed circuit boards, substantiates the structure of the database.

In the section "Synthesis of the system" the components of the system are analyzed, the equipment is selected and the structural scheme of the video surveillance system is developed.

In the section "Software development" the structure and methods of software implementation are substantiated, the software for modeling of topology of CAN-network in coal mines is developed.

In the experimental section the task of experiment is set and the technique of its carrying out is proved and efficiency and efficiency of development of a network topology on application of a Steiner tree is checked.

COAL MINING, DESIGN, CAN-NETWORK, TOPOLOGY, PROGRAM, SIMULATION, STEINER TREE

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних;

ПК – персональний комп’ютер;

ПЗ – програмне забезпечення;

СУБД – система управління базами даних;

ЕОМ – электронно-обчислювальна машина;

ОС – операційна система

ВСТУП

У вугільних шахтах і рудниках для доставки корисної копалини з вибою, а також для його транспортування в даний час широко застосовуються стрічкові конвеєри. Відстань транспортування при цьому досягає декількох кілометрів.

Конвеєрний транспорт забезпечує переміщення гірських мас за допомогою конвеєрів у вугільних шахтах і рудниках. У широкому сенсі цей транспорт, який об'єднує конвеєри і допоміжне обладнання (наприклад, бункери, живильники і ін.), Технічні засоби керування виконанням робіт, а також технічного обслуговування і ремонту.

Сучасний рівень технологій і вимоги безпеки вимагають використання у вугільних шахтах системи контролю конвеєрного транспорту. Системи відеоконтролю вимагають для своєї роботи досить розвинені мережі, які відсутні в вугільних шахтах України. Відповідно для використання автоматизованої системи контролю стрічкових конвеєрів у вугільних шахтах необхідно спроектувати і виконати монтаж мережі, яка буде з'єднувати потрібні пристрої. В якості основи для мережі можна використовувати поширений в шахтах телефонний дріт. Для цього завдання підійде CAN-мережа, що доведено дослідженнями.

В даний час можливе проектування мереж із застосуванням спеціального програмного забезпечення, яке дозволяє проектувати швидше, точніше, простіше, оберігає користувача від помилок, надає йому корисну інформацію про проектованої мережі. Існують різні програми для роботи з топологиями мереж, але не існує програми, яка б виконувала роботу з CAN-мережами в вугільних шахтах. Відповідно, таку програму, яка враховує всі особливості і вимоги цієї конкретної задачі, необхідно розробити на підставі результатів даного дослідження.

Мета і завдання дослідження: *Метою роботи* є обґрунтування параметрів комп'ютерної системи відеоконтролю стрічкових конвеєрів

вугільної шахти з детальним опрацьовуванням підсистеми проектування топології мережі.

Для досягнення поставленої мети необхідно вирішити такі завдання:

− виконати аналіз існуючого обладнання на вугільніх шахтах и використовуваних систем відеоконтролю стрічкових конвеєрів вугільніх шахт;

− виконати аналіз підсістем обробки інформації в комп’ютерних системах контролю;

− вивчити особливості побудови CAN-мереж;

− розробити структуру комп’ютерних системи відеоконтролю стрічкових конвеєрів з проробки підсистеми проектування топології мережі;

− розробити комплекс програм підсистеми проектування топології мережі;

− провести експериментальні дослідження и обґрунтувати параметри комп'ютерної системи відеоконтролю.

Об'єкт дослідження: комп'ютерна система відеоконтролю стрічкових конвеєрів вугільної шахти з підсістемою проектування топології мережі.

Предмет дослідження: процес проектування топології CAN-мережі вугільної шахти.

Методи дослідження: теорія оптимізації топологічної структури, теорія графів, теорія моделювання.

Ідея роботи: проектування топології CAN-мережі для контролю стрічкових конвеєрів вугільної шахти набагато ефективніше і простіше виконувати в спеціальному програмному забезпеченні, яке враховує особливості і вимоги процесу.

Наукові положення:

1. Встановлено, що представлення структури мережі системи автоматизованого контролю у вигляді дерева Штейнера дозволяє зменшити сумарну довжину ліній зв'язку на більш ніж 30%.

Наукові результати:

1. Обґрунтовано використання дерева Штейнера для моделювання структури мережі системи контролю, яке на відміну від бінарного дерева, в якому кожен вузол має один вхід і два виходи, дозволяє розробляти ієрархічні структури мереж, характерні для реальних вугільних шахт.

Обгрунтованість і достовірність наукових положень, висновків і рекомендацій підтверджуються тим, що в роботі використані експериментальні підтвердження результатів теоретичних досліджень.

Практичне значення отриманих результатів полягає в розробці програмного забезпечення, що дозволяє моделювати топологію CAN-мережі.

1 СТАН ПИТАННЯ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Технології підземного видобутку вугілля та особливості його транспортування

Вугілля вважається найбільш незвичайною породою з двох причин. По-перше, він утворюється з органічного матеріалу - колись живої тканини, по-друге, на відміну від інших порід, він може горіти і виділяти тепло.

Вугілля був основним видом палива під час промислової революції і зіграв важливу роль у розвитку багатьох країн. Він складається з вуглецю (звідси його чорний колір) і горючих газів - водню, азоту і кисню. Частина вуглецю і водню утворюють вуглеводень, що становить також основу нафти і природного газу.

Велика частина покладів вугілля утворилася 360-286 млн. Років тому, причому його було так багато, що геологи назвали цей період кам'яновугільним. Джерелом вугільних родовищ були доісторичні тропічні ліси, що виростали в болотистій місцевості і відрізнялися від сучасних. Здебільшого вони складалися з гігантських деревовидних папоротей, а також з великих хвощів і ряду дрібніших рослин.

Вугілля - єдиний вид органічного палива, запасів якого в надрах України досить для задоволення потреб всіх секторів економіки протягом декількох століть. Разом з тим українські родовища характеризуються надзвичайно складними геологічними умовами експлуатації, а більшість вугледобувних підприємств - незначною виробничою потужністю і досить низьким рівнем технічного забезпечення. Видобуток кам'яного вугілля ведеться підземним способом, і лише невеликий обсяг бурого вугілля (~ 500 тис. Т / рік) видобувається відкритим способом.

Відмирають деревовидні папороті та інша рослинність обсипалися в болота. У болотяній воді перебувало дуже мало кисню, що прискорює процес розкладання бактеріями органічного матеріалу, тому повільно гниють дерева

перетворювалися в торф − перша стадія появи вугілля. В процесі торфообразованія виділявся метан, або болотний газ.

Торф, ущільнюючи, перетворювався в вугілля. З шару торфу товщиною 10-15 м утворюється тонкий (близько 1 м) пласт вугілля. Перший етап ущільнення проходив в древніх болотах в міру того, як з'являлися все нові верстви гниючої рослинності, під масою яких спресовуються нижні шари. У кам'яновугільний період відбувалося підняття земної кори, в результаті чого пісок і мулисті відкладення накопичувалися поверх листя рослин. Згодом шари грунту і торфу були поховані під морськими водами, а потім знову вийшли на поверхню. Утворювалися інші болота, де з'являлися нові відкладення торфу. Цей процес, званий циклічним опадонакопиченням, повторювався багато разів. У вугільних районах є ряд розташованих один над іншим пластів вугілля, розділених шарами осадових порід. Товщина цих пластів коливається від декількох міліметрів до багатьох метрів.

Спочатку підземний видобуток вугілля починалася з проходки підхідних тунелів або штолень, які бурились від поверхневих виходів родовищ. Однак на глибину проходки цих шахт накладалися певні обмеження через недосконаліть засобів транспортування вугілля до поверхні і збільшенням ризику займання метану від свічок і іншого відкритого полум'я.

Збільшення потреби у вугіллі під час промислової революції стимулювало розвиток методів проходки шахтного стовбура для доступу до більш глибоких пластів. До середини двадцятого століття кількість вугілля, видобутого підземним способом у всьому світі, набагато перевищила кількість вугілля, видобутого тунельним способом.

Вугільна шахта складається з трьох головних складових частин: зона видобутку вугілля; зона транспортування вугілля до основи шахтного стовбура або похилої виробки і зона підйому вугілля до поверхні (на підйомнику або транспортером). Видобуток включає в себе і підготовчі

роботи, які необхідні, щоб отримати доступ до майбутніх зонам видобутку в шахті. Тому ця частина роботи найнебезпечніша.

При підготовці шахти до експлуатації найпростішим засобом доступу до вугільного пласту є розробка родовища від його виходу на поверхню. Ця технологія все ще залишається широко поширеною в областях, де місцевість над шахтою гориста, а пласти вугілля відносно плоскі.

Конкретний метод, яким проводиться видобуток вугілля в пласті, в даному випадку не має значення, важливо те, що доступ до пласту дешевий і вимагає мінімальної підготовки. Штольні також широко застосовують там, де технологія гірничої справи розвинена слабо, а вугілля, видобуте в штольні, може покрити витрати на підготовку родовища.

Іншими шляхами доступу до родовища є похилі виробки (бремсберги) і вертикальні шахтні стовбури. Зазвичай вибір залежить від глибини розробляється вугільного пласта. Чим глибше залягає пласт, тим дорожче буде коштувати ступінчастий бремсберг, за яким могли б рухатися транспортні машини або протягнуть стрічковий конвеєр.

Проходка шахтного стовбура, яка проводиться вертикально від поверхні, - довга і дорога робота. Вона вимагає більш тривалого часу для підготовки родовища (від початку проходки ствола і будівництва надземних споруд до видобутку першого вугілля). Якщо пласти залягають глибоко, як в більшості європейських країн і в Китаї, шахтні стовбури часто доводиться проходити через водоносні шари, що лежать над вугільними пластами. У таких випадках, щоб вода не потрапила в шахту, доводиться застосовувати спеціальні методи, наприклад, заморожування або цементацию. Потім шахтний ствол викладають зсередини сталевими кільцями або заливають стінки цементом, щоб забезпечити надійний, довгостроковий затвор.

Похилі виробки зазвичай використовуються, щоб дістатися до пластів, занадто глибоких для видобутку відкритим способом, але все-таки лежать близько до поверхні. У деяких місцях їх розробляють відкритим способом, а в інших необхідні підземні розробки.

В останньому випадку часто застосовують похилі виробки, щоб забезпечити зручний доступ в шахту для обладнання і поставити стрічковий конвеєр для вивезення добутого вугілля з шахти.

Похилі виробки відрізняються від штолень тим, що їх зазвичай проходять в скельній породі, а не по вугіллю (якщо тільки сам пласт не схилили вниз під постійним кутом). Самі вироблення проходяться під одним і тим же кутом на всьому протязі, для зручності під'їзду обладнання та роботи стрічкового конвеєра. Нововведення, що з'явилося після 1970-х рр., Складалося у використанні стрічкових конвеєрів для доставки вийнятого вугілля з глибоких шахт. Цей метод має багато переваг, в сенсі продуктивності і надійності, перед традиційним, з використанням підйомника в шахтному стовбурі.

Підземне видобування вугілля використовує два основні методи і безліч їх варіацій для різних умов і різних видів робіт. При камерно-стовповому способі проходиться мережу правильно розташованих горизонтальних виробок (штреків), між якими часто надовго залишають стовпи (цілики) для підтримки покрівлі виробки. Навпаки, при розробці лавами досягається повна виїмка великих ділянок вугільного пласта, в результаті чого породи покрівлі виробки обрушаются в порожнечу.

Камерно-стовпова розробка - найстаріша система при видобутку вугілля підземним способом і перша система, що використовувала ідею правильно розташованих циліків, які оберігають шахтарів від обвалів породи. Назва "камерно-стовпова" походить від стовпів, або циліків, що утворюють правильну сітку опор, які залишають як природну опору для покрівлі виробки. Зараз ця техніка перетворилася в високопродуктивний механізований метод, на який в деяких країнах припадає значна частина загального обсягу підземного видобутку.

Зазвичай камерно-стовпової метод використовується в дрібно залягають пластах, де тиск на опори з боку вищерозміщених породи не надто велике.

Ця система має два вирішальних переваги перед лавової розробкою: гнучкість і безпеку, що забезпечується самим принципом цього методу.

Головний же недолік - те, що витягується тільки частина запасів вугілля в родовищі (точна кількість залежить від таких факторів, як глибина залягання і потужність пласта). Виїмка до 60% вугілля цілком можлива. Можна, втім, проводити виїмку вугілля з самих ціликів як другу стадію розробки, тоді можна добути і 90% вугілля.

Ця система розробки годиться для будь-якого рівня технічної оснащеності, від найбільш трудомістких способів (наприклад, "кошикові розробка", більшість стадій якої, включно з транспортуванням вугілля, - ручні) до високомеханізованих методів. Вугілля може вилучатись із забою за допомогою яких вибухів, або машин для безперервної виїмки. Механізоване перевезення вугілля забезпечується або автотранспортом, або стрічковим конвеєром. Щоб підтримати покрівлю штреку і місця перетинів штреків, де відкритий простір більше, використовується анкерне кріплення і металеві або дерев'яні опори.

Комбайн безперервної дії, який включає в себе ріжучу робочу частину і систему завантаження вугілля, змонтовані на гусеничному ходу, важить зазвичай від 50 до 100 т, в залежності від висоти виробки, для якої він призначений, проектної потужності і необхідної ширини зарубки. Деякі обладнані машиною для постановки анкерного кріплення, яка зміцнює покрівлю виробки одночасно з виїмкою вугілля. В цьому випадку комбайн і машини для постановки анкерного кріплення працюють по черзі.

Локомотиви, що використовуються для транспортування вугілля під землею, можуть бути на електричній тязі (з подачею електроенергії по складеному кабелю або на акумуляторних батареях) або на дизельній тязі. Остання забезпечує більшу гнучкість. Вугілля навантажується з заднього відсіку комбайна безперервної дії в транспортну машину, яка відвозить вантаж - зазвичай від 5 до 20 т - на невелику відстань до бункерного завантажувального пристрою головного стрічкового конвеєра. У бункері

може бути встановлена дробарка для подрібнення негабаритних шматків вугілля або породи, які могли б заблокувати углеспуск або пошкодити стрічку конвеєра при подальшому транспортуванні.

Альтернативою автомобільного транспорту є безперервна система відкатки - гнучкий секційний конвеєр, окремі частини якого змонтовані на гусеничному ходу. По ньому вугілля прямо з комбайна йде в трубу. Це дає переваги з точки зору як безпеки персоналу, так і продуктивності праці; з тих же причин такі конвеєри зараз закладаються і в проекти відкатних виробок при лавовом способі видобутку.

Штреки мають ширину до 6,0 м і висоту, рівну, як правило, товщині пласта. Розміри стовпів залежать від глибини залягання пласта. Якщо пласт тонкий і залягає неглибоко, типовий стовп буде квадратним, 15,0 м у висоту і зі стороною 21,0 м.

Розробка лавами (інакше: довгими вибоями) зазвичай вважається винаходом двадцятого століття, однак, сама концепція була, мабуть, розроблена понад 200 років тому. Підштовхнув її впровадження в практику той факт, що на зміну переважно ручним способам розробки в 1950-і рр. прийшло значне підвищення рівня механізації. При сьогоднішньому рівні механізації розробка лавами стала високопродуктивним методом видобутку, що вимагає дуже невеликого числа робочих.

У виїмки довгими вибоями одне головна перевага перед камерно-стовпової розробкою: вона дозволяє провести повну виїмку групи камер за один прохід і в цілому вийняти більший відсоток від загальної кількості вугілля в поклади. Однак метод не так гнучкий, тому загальні придатні до розробки запаси вугілля повинні бути більше, і необхідна впевненість в тому, що попит на вугілля в найближчим часом не впаде, оскільки капітальні вкладення в підготовку і обладнання сучасного довгого забою складають іноді більше 20 млн. Американських доларів.

У минулому в окремих шахтах часто розроблялося одночасно кілька довгих вибоїв (наприклад, в деяких польських шахтах - більше десяти

вибоїв). Зараз же намітилася тенденція до зосередження виробничих потужностей в невеликому числі вибоїв, що працюють в інтенсивному режимі. Перевага тут у скороченні трудовитрат і в спрощенні підземної інфраструктури (а значить, і зниження витрат на її підтримку).

При розробці лавами породу покрівлі навмисно обрушают, коли пласт повністю вироблений; залишають тільки головні під'їзні вироблення, які зміцнюють колонами. Міцність покрівлі довгого забою забезпечується дво- або четирехстержневимі гідравлічними опорами, які приймають на себе вагу порід покрівлі і частково перерозподіляють навантаження на ще не вироблений забій і колони по сторонам групи камер. Таким чином люди і техніка в забої виявляються захищені від небезпеки обвалу позаду опор. Вугілля рубають за допомогою виїмкової комбайна на електричній тязі, зазвичай обладнаного двома врубочнимі ножовими барабанами. Зазвичай за один прохід із забою виймається смуга вугілля до 1,1 м завтовшки. Виїмкову комбайн на ходу вантажить вугілля на захищений щитками конвеєр, який після кожного заходу подається вперед за рахунок послідовного руху опор забою.

У забої вийнятий вугілля вантажиться на стрічковий конвеєр для відправки на поверхню. У міру того, як забій поглиблюється, конвеєр також доводиться періодично подовжувати; в разі ж відпрацювання зворотним ходом конвеєр, навпаки, коротшає. Приклад розташування конвеєрного транспорту в шахті показаний на рисунку 1.1.

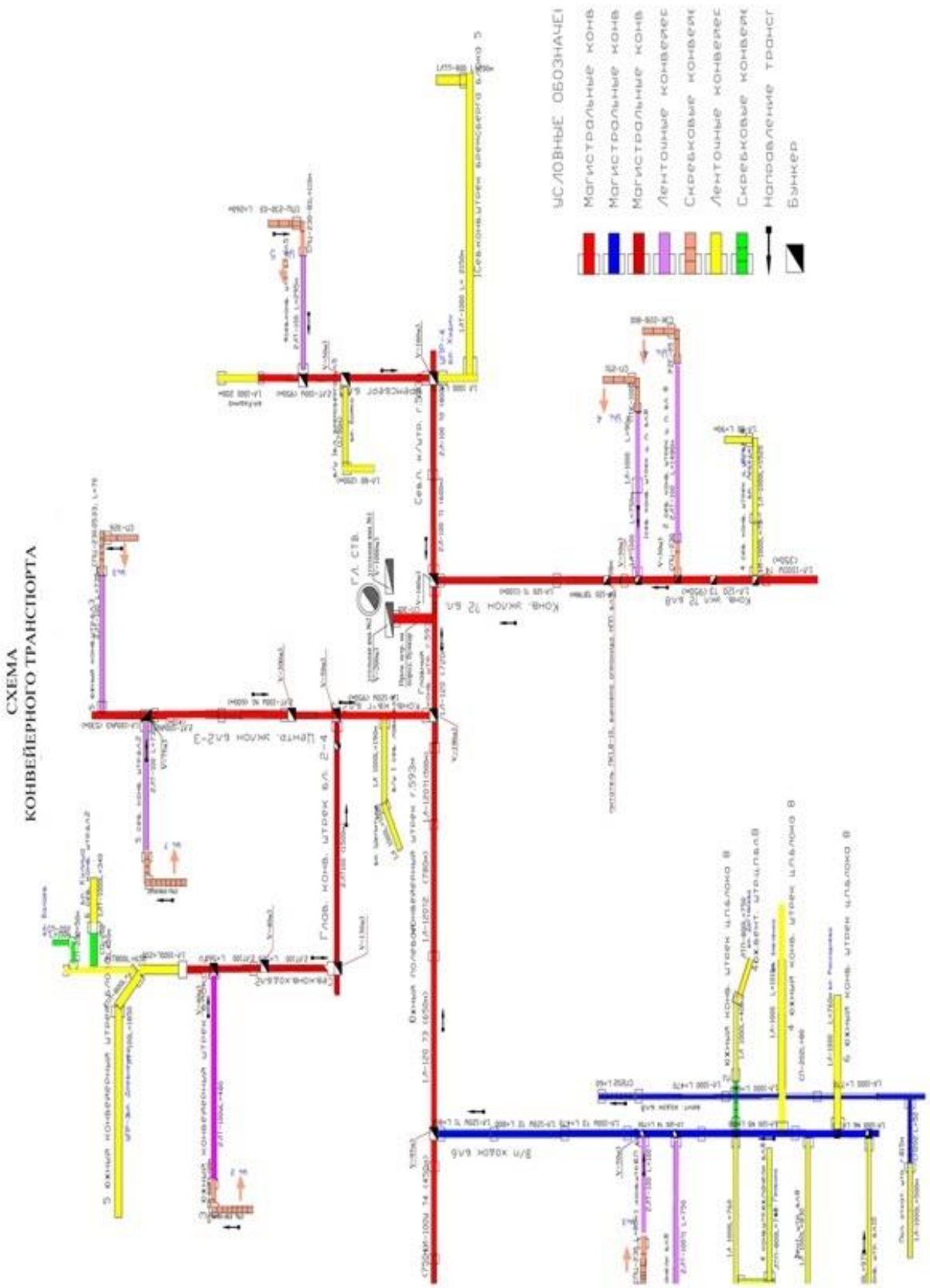


Рисунок 1.1 − Приклад розташування конвеєрного транспорту в шахті

За останні 40 років суттєво збільшилися як довжина забою, так і глибина групи камер (тобто блоку породи, через який йде забій). Наприклад, в США середня довжина забою збільшилася від 150 м в 1980 р до 227 м в 1993 р У Німеччині в 1990-х рр. середня довжина такого забою становить 270 м, і плануються забої понад 300 м. У Великобританії і Польщі зараз роблять забої до 300 м. Глибина групи камер в основному визначається геологічними умовами, наприклад, зрушеннями гірської породи, або межами шахти, але зараз вона, при сприятливих умовах, стійко тримається на рівні понад 2,5 км. У США обговорюється можливість збільшення глибини групи камер до 6,7 км.

Промисловим стандартом стає вироблення зворотним ходом, хоча вона і вимагає великих фінансових затрат на проходку штреків до кінця кожної групи камер перед початком виїмки. По можливості, штреки зараз намагаються робити всередині пласта за допомогою комбайнів для безперервної виїмки; при цьому замість застосовувалися раніше для кріплення сталевих арок і розкосів використовують анкерне кріплення. Цим досягається активна підтримка вищерозміщеної породи замість пасивної реакції на рухи породних мас. Її застосування, однак, обмежена міцністю покрівлі, складеними твердими породами.

1.2 Існуючі алгоритми оптимізації топології мереж

При організації мереж одним з основних завдань є розподіл потоків інформації по найкоротших шляхах. Під такими шляхами розуміють шляхи передачі інформації, найкоротші за часом передачі або протяжності, або шляху з мінімальними перешкодами, числом задіяних вузлів, вартістю і т.п. Таким чином, оптимізація шляхів може проводитися за різними техніко-економічними критеріями і вибрані шляхи повинні забезпечувати при заданих вимогах найбільш ефективне використання ліній і вузлів зв'язку. У зв'язку з цим має сенс проведення порівняльного аналізу існуючих на сьогоднішній день алгоритмів маршрутизації з метою вироблення закономірностей еволюції розвитку топології і методики управління інформаційними потоками.

Було виявлено, що використання адаптивної маршрутизації може зменшити середній час перебування пакета в мережі, дозволяє мінімізувати витрати на його доставку одержувачеві в мережах з різнорідним трафіком, збільшити загальну надійність мережі за рахунок можливості автоматичного вибору альтернативного маршруту на підставі даних про топологію мережі. Однак, дані переваги досягаються збільшенням навантаження на обчислювальні центри вузлів комутації, тому використання адаптивної маршрутизації обмежена розмірами автономної системи (домену). Очевидно, що при цьому виникає проблема розробки математичного забезпечення оптимізації процедур вибору маршруту з метою зниження навантаження на обчислювальні центри, а також щоб можна було скористатись багатоколійні маршрутизації.

На етапі вибору оптимального маршруту для відправки пакетів наступного вузла мережу будемо розглядати як зважений орієнтований граф. Вершини графа представляють маршрутизатори, а дуги, що з'єднують ці вершини, - фізичні лінії між вершинами. Кожній лінії зв'язку відповідає деякий інтегральне значення, представлене за допомогою «вартості» пересилки пакету по ній, яке може залежати як від фізичної довжини лінії,

тимчасових витрат при передачі даних, так і від фінансових витрат транспортування пакета. В даний час відомий ряд алгебраїчних методів, що дозволяють певною мірою описати цей процес з тим, щоб отримувати результати у формі, зручній для подальших досліджень. А для цього необхідно зробити зусилля по класифікаційному моніторингу відомих методів з метою оцінки їх прикладної значущості і обмежень їх використання.

1.2.1 Метод лінійного програмування

Визначення найкоротшого шляху методом лінійного програмування - задача відшукання екстремуму деякої адитивної функції, значення якої обмежені простором, описуваних системою рівнянь і нерівностей. Цей метод дозволяє оптимізувати шляхи по якомусь невідомому параметру гілок.

Процес передачі інформації в мережі буде оптимальним, якщо розподілити допустимі часи передачі інформації по окремим галузям Т-мережі так, щоб середній час передачі інформації по шляху було мінімальним. Для цього в якості вихідної умови покладемо загальний час, що витрачається на передачу інформаційних потоків по кожній колії, не перевищить встановленого для даного шляху допустимого часу. Процес проходження інформаційного потоку по і-му шляху мережі за умови, що трафики окремих гілок незалежні, може бути описаний таким лінійним алгебраїчним нерівністю:

$$Q_{i,1}t_1 + Q_{i,2}t_2 + \ldots + Q_{i,j}t_j + \ldots + Q_{i,n}t_n \le t_{\partial i}, \qquad (1.1)$$

где $Q_{i,j}$ – число пакетів, що передаються по і-му шляху в j-й гілці $i = 1, 2, \ldots, \theta$; $j = 1, 2, \ldots, n$; n – максимальне число гілок в мережі; θ – число можливих шляхів в мережі; t_j – час передачі одного пакета по j-й гілці; $t_{\partial i}$ – допустимий час проходження інформаційного потоку по і-му шляху.

Якщо в мережі можливі θ шляхів, то, отже, може бути отримана система лінійних алгебраїчних нерівностей:

$$\left.\begin{array}{l} Q_{1,1}t_1 + Q_{1,2}t_2 + \ldots + Q_{1,j}t_j + \ldots + Q_{1,n}t_n \le t_{\partial 1} \\ Q_{2,1}t_1 + Q_{2,2}t_2 + \ldots + Q_{2,j}t_j + \ldots + Q_{2,n}t_n \le t_{\partial 2} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ Q_{i,1}t_1 + Q_{i,2}t_2 + \ldots + Q_{i,j}t_j + \ldots + Q_{i,n}t_n \le t_{\partial i} \\ Q_{\theta,1}t_1 + Q_{\theta,2}t_2 + \ldots + Q_{\theta,j}t_j + \ldots + Q_{\theta,n}t_n \le t_{\partial\theta} \end{array}\right\} \quad (1.2)$$

Для вирішення системи лінійних нерівностей складається лінійна форма (цільова функція):

$$F_л = \frac{\sum\limits_i Q_{i,1}}{\sum\limits_i \sum\limits_j Q_{i,j}} t_1 + \frac{\sum\limits_i Q_{i,2}}{\sum\limits_i \sum\limits_j Q_{i,j}} t_2 + \ldots + \frac{\sum\limits_i Q_{i,j}}{\sum\limits_i \sum\limits_j Q_{i,j}} t_j + \ldots + \frac{\sum\limits_i Q_{i,n}}{\sum\limits_i \sum\limits_j Q_{i,j}} t_n \quad (1.3)$$

невідомих $t_j \ \ (j = 1,2,\ldots,n)$.

Серед всіляких невід'ємних розв'язків $(t_1^*, t_2^*, \ldots, t_n^*)$ системи лінійних нерівностей визначається таке рішення $(t_1^{(0)}, t_2^{(0)}, \ldots, t_n^{(0)})$, при якому лінійна функція приймає найменше можливе значення. Очевидно, лінійна функція описує час проходженні інформаційного потоку по обраному шляху мережі. Рішення розглянутої задачі лінійного програмування дозволяє оптимальним чином оцінити часовий ресурс трафіку передачі пакетів по гілках топології мережі при мінімізації середнього часу передачі пакету по шляху.

1.2.2 Методи динамічного програмування

Завданням відшукання найкоротших шляхів передачі інформації притаманні три основних властивості динамічного програмування: багатоходової вибір, адитивність і незалежність оптимального шляху від передісторії. Справді, дослідження процесу передачі повідомлень від вузла до вузла в мережі розпадається на окремі етапи (багатоходової вибір); довжина шляху, що складається з декількох гілок, дорівнює сумі довжин цих гілок (властивість адитивності), і, нарешті, найкоротший шлях передачі

повідомлень з будь-якого вузла не залежить від того, як повідомлення потрапило в цей вузол, а тільки від розташування цього вузла в мережі (властивість незалежності від передісторії). На практиці знайшли застосування: метод Флойда-Уоршелла, метод Беллмана-Форда, а також менш відомий матричний метод.

Метод Флойда-Уоршелла, заснований на поняттях про базисної лінії зв'язку і тернарной операції, застосовується для знаходження найкоротших відстаней між усіма вершинами зваженого орієнтованого графа. Переваги даного методу - простота реалізує алгоритму і можливість отримання маршрутної інформації відразу для всіх вузлів мережі, що робить його застосування доцільне при централізованих структурах управління інформаційними потоками. Говорячи про складність даного методу при програмної реалізації, варто відзначити, що три вкладених циклу містять операцію, виконувану за константне час, тобто алгоритм має кубічну складність. В даний час існують способи прискорення систем, що використовують даний метод, що дозволяють знизити складність. Йдеться про використання бітових масок (в моделі з проміжним зберіганням результатів обчислень в RAM) і спеціалізованих наборах мікропроцесорних команд, як SSE (Streaming SIMD Extensions), в яких перевага в продуктивності досягається в разі проведення однієї і тієї ж послідовності операцій над різними даними.

Метод Беллмана-Форда застосовується для пошуку найкоротшого шляху в зваженому графі. Основною перевагою є можливість розрахунку шляху в графі, в якому є ребра з негативним вагою. Для підрахунку найкоротшого шляху цим методом потрібно провести всього n-1 циклів, але на практиці цей алгоритм можна використовувати і для відстеження негативних циклів, провівши рівно n циклів. Якщо при виконанні останньої ітерації довжина найкоротшого шляху до будь-якої вершини строго зменшилася, то в графі є негативний цикл. Можна відстежувати зміни в графі

і, як тільки вони закінчаться, подальші ітерації будуть безглузді. Даний метод використовується в протоколі маршрутизації RIP (Routing Information Protocol) і його модифікаціях, причому деякі його модифіковані версії задіяні в невеликих мережах (не більше 15 робочих станцій), в яких не потрібно великих обчислювальних потужностей для розрахунку шляхів, а також - для мереж з майже незмінною структурою.

Матричний метод визначення найкоротших шляхів між усіма вузлами мережі було запропоновано Шімбелом і вдосконалений Оттерманом. Можливість повного топологічного аналізу мережі забезпечила широке практичне застосування цього методу при організації трафіку передачі даних в мережах.

При визначенні найкоротших шляхів матричних методом виконуються наступні операції:

1. По графу телекомунікаційної мережі (Т-мережу) складається матриця ваг і $S = \|s_{i,j}\|$

2. Матриця ваг перетворюється за правилами множення матриць, але з використанням спеціальних операцій Шімбела, в дисперсионную матрицю найкоротших шляхів між вузлами мережі, тобто $S^{\zeta} = \left\|\left(s_{i,j}\right)^{\xi}\right\| = D = \|\delta_{i,j}\|$

3. Дисперсійна матриця потім перетворюється на допоміжну матрицю $\Delta = \|d_{i,i}\| = D \cdot M$, де М - видозмінена матриця ваг S.

4. Значення і індекси матриці $\Delta = \|d_{i,j}\|$ послідовно розподіляються за дистанційними матрицями, що визначає довжину 1,2, ..., k-го шляхів, і маршрутним матрицями, що визначає проміжні вузли цих шляхів.

5. Виключаються петлі в найкоротших шляхах таким чином, щоб вага подальшої гілки на даному шляху топографій не дожжен перевершувати ваги попередньої. Очевидно, що цей крок алгоритму обмежує застосування цього

методу, оскільки перевірити зазначена умова можливо шляхом аналізу всіх шляхів проходження інформаційних потоків в мережі.

Однак, матричний метод дозволяє не тільки визначити величини найкоротших шляхів між усіма вузлами мережі, але також одночасно отримати довжини всіх можливих шляхів між кожною парою вузлів мережі. Це дає можливість використовувати матричний метод для відшукання в мережі обхідних шляхів. Обсяг обчислень при використанні матричного методу незначно залежить від структури мережі. Метод зручний для програмної реалізації.

1.2.3 Комбінаторний алгоритм топологічної оптимізації мережі передачі інформації

Розглянемо комбінаторний алгоритм вирішення задачі вибору оптимальної схеми з'єднання вузлів комутації пакетів вибору маршрутів і пропускної здатності ліній, який використовувався на різних етапах проектування і розвитку базової мережі системи «Сирена», а також інших великомасштабних мереж.

Комбінаторний підхід спирається на уявлення мережі передачі даних у вигляді кінцевого графа без петель і кратних ребер, вершиною якого відповідають вузлам мережі, а ребра - лініями зв'язку. Таке уявлення зручно для вивчення пропускної здатності мережі, тому що ці характеристики еквіваленти різних інваріанта графа, дослідження яких ведеться методами добре розробленої теорії графів.

Застосування теорії перерахування графів для вирішення завдання топологічної оптимізації довгий час вважалося безперспективним через необхідність дослідження значного числа можливих варіантів з'єднання ліній зв'язку. Наприклад, в мережі з 10 вузлів існує 245 варіантів розташування ліній зв'язків, включаючи безліч тривіальних випадків. Якщо припустити, що аналіз кожного варіанта становить 1 с (насправді ця цифра значно більше, тому що аналіз включає в себе перевірку надежностних показників, рішення задачі вибору пропускних спроможностей і розподілу потоків), то на дослідження всіх варіантів потрібно більш ніж 9 * 〚10〛 ^ 8 років.

Навіть для малих мереж з числом вузлів 6 і 7 потрібно відповідно 8,8 ч. І 24,2 діб.

Проте останнім часом комбінаторний підхід знаходить все більш широке застосування при вирішенні задачі топологічної оптимізації. Це пов'язано, по-перше, з підвищенням продуктивності ЕОМ, використовуваних для розрахунку комбінаторних задач; по-друге, з розробкою нових ефективних алгоритмів генерації графів із заданими властивостями і, нарешті, великий досвід розробки та експлуатації мереж передачі даних

дозволяє досить обґрунтовано формулювати вимоги до проектованої мережі, які істотно звужують клас можливих рішень задачі топологічної оптимізації.

Необхідність розробки комбінаторних алгоритмів викликана принаймні наступними причинами:

1. Великомасштабні мережі передачі даних зазвичай проектуються поетапно, причому розмірність мережі на перших етапах невелика, що дозволяє отримувати точне рішення задачі топологічної оптимізації за допомогою комбінаторних алгоритмів. У той же час відомі наближені алгоритми не дозволяють знаходити точне рішення навіть для мереж невеликої розмірності. При цьому кращі евристичні алгоритми за даними розробників дають похибку до 10%.

2. Наявність точного рішення дозволяє оцінити якість відомих і розроблених нових евристичних алгоритмів.

3. Комбінаторні алгоритми надають широкі можливості для вивчення властивостей оптимальних рішень і оптимізується функції, що в свою чергу створює передумови для розробки нових ефективних алгоритмів.

Розроблені алгоритми програмно реалізовані в пакеті прикладних програм синтезу топології мереж передачі даних, які включають:

- точний і наближений алгоритм конструктивного перерахування графів для вирішення завдання топологічного синтезу мережі мінімальної вартості при наявності обмежень на надійність. Зазначені алгоритми ефективно використовуються на етапі передпроектного дослідження мережі;

- комбінаторні алгоритму точного і наближеного рішення задачі синтезу топології, вибору пропускних спроможностей і маршрутів, які застосовуються на етапі технічного проектування і в процесі розвитку комп'ютерної мережі;

- алгоритм «насичення перетину»;

- алгоритми розв'язання задачі топологічного синтезу мереж з підвищеними вимогами до надійності (проектування мереж з кількістю незалежних шляхів між кожною парою вузлів великим двох);

- евристичні алгоритми задач синтезу топології мереж великої розмірності.

1.2.4 Графові алгоритми в виборі топології мережі

Особливе місце в математичному забезпеченні детермінованих процедур вибору трафіку передачі інформаційних потоків займають методи Флойда і Дейкстри, що базуються на принципі оптимальності.

Метод Дейкстри дозволяє знаходити найкоротші шляхи від однієї з вершин графа до всіх інших. Метод працює тільки для графів без ребер негативного ваги, хоча в даний час існують узагальнені методи для усунення даного недоліку (метод Дейкстри з потенціалами). Суть методу Дейкстри полягає у поетапному нарощуванні дерева найкоротших маршрутів від вихідного вузла. При цьому необхідно, щоб після додавання на кожному етапі лінії зв'язку і вузла знову утворений найкоротший маршрут був мінімально можливим за всіма кінцевим вузлам, ще не увійшли в дерево. В процесі побудови дерева найкоротших маршрутів обчислюються вектори ваг маршрутів і коригуються вектори початкових компонент найкоротших маршрутів. Складність методу Дейкстри залежить від способу знаходження вершини v, а також від способу зберігання безлічі невідвіданих вершин і способи оновлення міток. У графі G, n і m відповідно кількість вершин і ребер для пошуку вершини з мінімальною довжиною шляху до вершини v, проглядається все безліч n. Час роботи алгоритму мінімізації є $O\left(n^2+m\right)$. Для розряджених графів (для таких, для яких m<<n) при використанні спеціальних алгоритмів оптимізації швидкість роботи може скласти $O\left(n\log n+m\log n\right)$ або навіть $O\left(n\log n+m\right)$. Метод Дейкстри широко застосовується в мережевому програмуванні і технологіях, наприклад, його використовують в протоколі OSPF (Open Shortest Path First) для усунення кільцевих маршрутів.

Використання модифікованого алгоритму Дейкстри, як ефективного інструменту для розподілу вхідних інформаційних потоків в магістральних

IP-мережах з протоколом OSFP, дозволяє поліпшити робастної мережі до інформаційних перевантажень. При цьому можливо як критерій розподілу інформаційних потоків використовувати залишкову пропускну здатність каналу. Необхідно віднести до позитивних якостей протоколу відносну простоту практичної реалізації методу.

Метод Джонсона дозволяє знайти найкоротші шляхи між усіма парами вершин зваженого орієнтованого графа. Даний метод працює, якщо в графі містяться ребра з позитивним або негативним вагою, але відсутні цикли з негативним вагою. У методі Джонсона використовується метод Беллмана-Форда і метод Дейкстри, реалізовані у вигляді підпрограм. Ребра зберігаються у вигляді списків суміжних вершин.

Якщо в методі Дейкстри неубутна черга з пріоритетами реалізована у вигляді фібоначчійовой множини, то час роботи методу Джонсона дорівнює $O\left(n^2 \lg n + nm\right)$. При більш простий реалізації неубивающей черзі з пріоритетами час роботи стає рівним, але для розріджених графів ця величина в асимптотичному межі поводиться з точки зору нечіткого критерію логіки краще, ніж час роботи алгоритму Флойда-Уоршелла.

Висновки:

1. Для мереж з централізованим управлінням інформаційними потоками кращими є метод Флойда-Уоршелла і метод Джонсона, причому останній буде давати виграш по швидкості лише в разі топології мережі, описуваної великим розрідженим графом.

2. Серед однопутевих методів знаходження оптимального шляху можна відзначити метод Беллмана-Форда (протокол RIP) і метод Дейкстри (протокол OSFP). Обидва методи в даний час активно використовуються, але на різних рівнях мережі: метод Дейкстри в складі протоколу OSFP використовується для маршрутизації всередині автономних систем (протокол внутрішнього шлюзу), в той час як метод Беллмана-Форда (протокол RIP) використовується набагато рідше в невеликих мережах і для междоменной маршрутизації через обмеження на кількість вузлів в мережі. Але основною перевагою протоколу RIP як і раніше залишається простота конфігурування.

3. Матричний метод, як і інші багатоколійні методи, має перевагу перед однопутевимі, що полягає в наявності заздалегідь розрахованих альтернативних маршрутах, що, в кінцевому рахунку, призводить до збільшення надійності мережі і можливості перерозподілу навантаження між каналами зв'язку.

4. Визначення найкоротшого шляху може бути здійснено методами лінійного програмування, що дозволить оцінювати тимчасову інерційність мережі при мінімізації середнього часу передачі повідомлення.

1.3 Особливості контролю стрічкових конвеєрів вугільної шахти

Об'єкт розробки - система відеоконтролю стрічкових конвеєрів вугільної шахти з підсистемою зберігання інформації, яка призначена для безперервного автоматизованого збору інформації про стан технологічних об'єктів, а також для накопичення, обробки і відображення інформації.

Комп'ютерна мережа розробляється для умов видобутку вугілля на шахтах України і спрямована на контроль стану роботи шахти в процесі вуглевидобутку.

На шахтах України умови видобутку вугілля найбільш небезпечні в порівнянні з усіма вугледобувними країнами світу. Це обумовлено наявністю суфлярного виділення метану в видобувних забоях, вибухами газу та вугільного пилу, раптовими викидами газу, порід і ін.

На деяких шахтах введено автоматизовані системи контролю і управління технологічними процесами (АСУ ТП). Однак існуючі на шахтах локальні комп'ютерні мережі не пов'язані в єдину інформаційну систему, здатну відображати обстановку в вугільних лавах і забезпечувати можливість персоналу аналізувати інформацію, що надійшла і приймати рішення по управлінню всією технологічною системою шахти в безаварийном режимі.

Таким чином, для безаварійного функціонування технологічної системи шахти необхідно на будь-якому рівні управління своєчасно мати відповідну, достатню, достовірну інформацію, яка була б представлена в зручному для застосування вигляді.

Комп'ютерна мережа включає підсистему контролю стану обладнання та навколишнього середовища.

Для розробки системи пропонується створити єдину загальношахтного CAN-мережа, яка застосовує сучасні контролери і ЕОМ.

Загальні функції мережі:

- збір, накопичення, розподіл і збереження інформації про роботу обладнання в шахті, в тому числі і відеоінформації;

- введення інформації і управління базою даних;

- введення нормативно-довідкової інформації, плану попередження аварії та іншої інформації, необхідної для оперативного прийняття рішень;

- збір, накопичення і збереження виробничої інформації;

- передачу інформації персоналу, головним фахівцям і керівникам шахти;

- подання інформації на моніторах;

- інтелектуальна підтримка для прийняття управлінських рішень оперативно-диспетчерським персоналом і головними фахівцями.

Для забезпечення зв'язку і інформаційного обміну в середовищі підсистем і між ними повинні використовуватися стандартні інтерфейси і протоколи.

Для забезпечення необхідної надійності комп'ютерної мережі нульового і дільничного рівнів повинні бути передбачені апаратні і програмні засоби виявлення і локалізації відмов, а також кошти реконфігурації мережі

Технічні та програмні засоби автоматизації, які будуть використовуватися для побудови повинні допускати реалізацію обміну інформацією між дільничними і диспетчерськими рівнями на відстані до 15 км.

Комп'ютерна мережа на поверхні шахти необхідна для:

- з'єднання керівників шахти, керівників окремих ділянок і служб з гірським диспетчером, з комерційно-економічними службами і службами медико-психологічного відбору кадрів, підготовки і перепідготовки кадрів,

- комп'ютерного аналізу бази даних про стан технологічного циклу і його складових частин, що постійно поповнюється, дозволяє приймати правильні технологічні та управлінські рішення в оптимальному для кожної конкретної ситуації, а в разі необхідності включити систему аварійного оповіщення.

1.4 Мета і задачі дослідження

Цель и задачи исследования: Метою роботи є розробка комп'ютерної системи з програмним забезпеченням, яке спрощує і оптимізує процес проектування топології CAN-мережі в вугільних шахтах.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз існуючих способів подання і оптимізації топології;
- вивчити особливості побудови CAN-мереж;
- розробити модель топології мережі;
- розробити комп'ютерну систему контролю;
- визначити список функцій і вимог щодо розроблюваного програмного забезпечення;
- розробити програмне забезпечення.

2 ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

2.1 Теорія графов

Неформально граф можна розглядати як безліч точок і з'єднують ці точки ліній зі стрілками або без них.

З графами, самі того не помічаючи, ми стикаємося постійно. Наприклад, графом є схема ліній метрополітену. Точками на ній представлені станції, а лініями - шляхи руху поїздів. Досліджуючи свій родовід і зводячи її до далекого предка, ми будуємо так зване генеалогічне древо. І це древо - граф.

Графи служать зручним засобом опису зв'язків між об'єктами. Раніше вже були використані графи як спосіб наочного подання кінцевих бінарних відносин. Але граф використовують не тільки як ілюстрацію. Наприклад, розглядаючи граф, який зображає мережу доріг між населеними пунктами, можна визначити маршрут проїзду від пункту А до пункту Б. Якщо таких маршрутів виявиться кілька, хотілося б вибрати в певному сенсі оптимальний, наприклад найкоротший або найбезпечніший. Для вирішення завдання вибору потрібно проводити певні обчислення над графами. При вирішенні подібних завдань зручно використовувати алгебраїчну техніку, та й саме поняття графа необхідно формалізувати.

Методи теорії графів широко застосовуються в дискретної математики. Без них неможливо обійтися при аналізі і синтезі різних дискретних перетворювачів: функціональних блоків комп'ютерів, комплексів програм і т.д.

В даний час теорія графів охоплює великий матеріал і активно розвивається.

Графи є спосіб "візуалізації" зв'язків між певними об'єктами. Зв'язки ці можуть бути "спрямованими", як, наприклад, в генеалогічному древі, або "ненаправленими" (мережа доріг з двостороннім рухом). Відповідно до цього

в теорії графів виділяють два основних типи графів: орієнтовані (або спрямовані) і неорієнтовані.

Побудова математичного визначення графа здійснюється шляхом формалізації та "об'єктів", і "зв'язків" як елементів деяких (як правило, кінцевих) множин.

2.2 Трасування і розводка друкованих плат

Трасування з'єднань є, як правило, заключним етапом конструкторського проектування радіоелектронної апаратури та полягає у визначенні ліній, що з'єднують Еквіпотенціальна контакти елементів, і компонентів, складових проектоване пристрій.

Завдання трасування - одна з найбільш трудомістких в загальній проблемі автоматизації проектування радіоелектронної апаратури. Це пов'язано з кількома факторами, зокрема з різноманіттям способів конструктивно-технологічної реалізації з'єднань, для кожного з яких при алгоритмічній вирішенні задачі використовуються спеціальні критерії оптимізації і обмеження. З математичної точки зору трасування - найскладніші завдання вибору з величезної кількості варіантів оптимального рішення.

Одночасна оптимізація всіх з'єднань при трасуванні за рахунок перебору всіх варіантів в даний час неможлива. Тому розробляються в основному локально оптимальні методи трасування, коли траса оптимальна лише на даному етапі при наявності раніше проведених з'єднань.

Основне завдання трасування формулюється так: за заданою схемою з'єднань прокласти необхідні провідники на площині (платі, кристалі і т. Д.), Щоб реалізувати поставлені технічні з'єднання з урахуванням заздалегідь заданих обмежень. Основними є обмеження на ширину провідників і мінімальні відстані між ними.

Вихідною інформацією для рішення задачі трасування з'єднань зазвичай є список ланцюгів, параметри конструкції елементів і комутаційного поля, а також дані по розміщенню елементів. Критеріями трасування можуть бути відсоток реалізованих сполук, сумарна довжина провідників, кількість перетинів провідників, число монтажних верств, число міжшарових переходів, рівномірність розподілу провідників, мінімальна область трасування і т. Д. Часто ці критерії є взаємовиключними, тому оцінка якості трасування ведеться по домінуючому критерію при виконанні

обмежень за іншими критеріями або застосовують адитивну або мультипликативную форму оціночної функції

Відомі алгоритми трасування друкованих плат можна умовно розбити на три великі групи:

1. Хвильові алгоритми, засновані на ідеях Лі та розроблені Ю.Л. Зіманом і Г.Г. Рябовим. Дані алгоритми набули широкого поширення в існуючих САПР, оскільки вони дозволяють легко враховувати технологічну специфіку друкованого монтажу зі своєю сукупністю конструктивних обмежень. Ці алгоритми завжди гарантують побудова траси, якщо шлях для неї існує.

2. Ортогональні алгоритми, що володіють більшу швидкодію, ніж алгоритми першої групи. Реалізація їх на ЕОМ вимагає в 75-100 разів менше обчислень в порівнянні з хвильовими алгоритмами. Такі алгоритми застосовують при проектуванні друкованих плат з наскрізними металізованими отворами. Недоліки цієї групи алгоритмів пов'язані з отриманням великого числа переходів зі шару на шар, відсутністю 100% -ої гарантії проведення трас, великим числом паралельно йдуть провідників.

3. Алгоритми евристичного типу. Ці алгоритми частково засновані на евристичному прийомі пошуку шляху в лабіринті. При цьому кожне з'єднання проводиться по найкоротшому шляху, обходячи зустрічаються на шляху перешкоди.

2.3 Розробка моделі топології мережі з використанням дерева Штейнера

Робота Штейнера була присвячена пошуку однієї точки, сума відстаней від якої до всіх точок заданої множини була б мінімальною. Однак ще в 1640 році вперше була поставлена задача, яка є окремим випадком обох описаних завдань - тієї, над якою працював Штейнер, і тієї, яка носить його ім'я: знайти точку P, сума відстаней від якої до кожної з трьох заданих точок мінімальна. Еванджеліста Торрічеллі і Бонавентура Кавальєрі незалежно один від одного вирішили це завдання. Торрічеллі і Кавальєрі довели, шануючи сумарне відстань мінімально, коли все зв'язані кути в точці P більше або дорівнюють 120 °.

Знаючи, що кути з вершинами в точці P повинні бути не менше 120 °, Торрічеллі і Кавальєрі придумали процедуру геометричної побудови для знаходження точки P (рисунок 2.1).

Найкоротша мережу для трьох точок A, B і C. На найдовшій стороні трикутника ABC будується рівносторонній трикутник ACX (зелений колір), і навколо нього описується коло (жовтий колір). На перетині її з відрізком BX знаходиться точка P, яка називається точкою Штейнера. Відрізки AP, BP і CP утворюють три сполучених кута по 120 ° і найкоротшу мережу, причому їх сумарна довжина дорівнює BX.

Найкоротша мережу для трьох точок A, B і C. На найдовшій стороні трикутника ABC будується рівносторонній трикутник ACX (зелений колір), і навколо нього описується коло (жовтий колір). На перетині її з відрізком BX знаходиться точка P, яка називається точкою Штейнера. Відрізки AP, BP і CP утворюють три сполучених кута по 120° і найкоротшу мережу, причому їх сумарна довжина дорівнює BX.

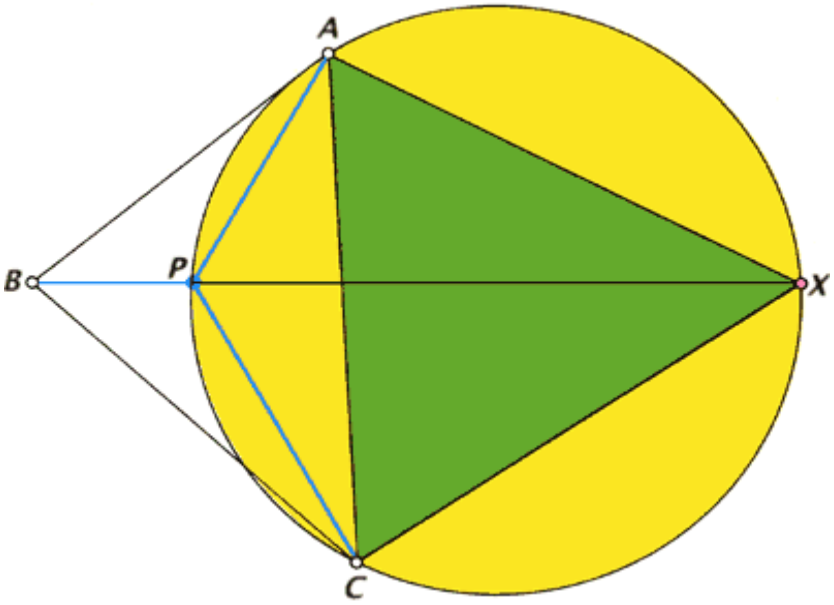


Рисунок 2.1 - Знаходження точки P

Найкоротша мережу для трьох точок A, B і C. На найдовшій стороні трикутника ABC будується рівносторонній трикутник ACX (зелений колір), і навколо нього описується коло (жовтий колір). На перетині її з відрізком BX знаходиться точка P, яка називається точкою Штейнера. Відрізки AP, BP і CP утворюють три сполучених кута по 120 ° і найкоротшу мережу, причому їх сумарна довжина дорівнює BX.

Завдання з трьома точками і завдання Штейнера для багатьох точок мають багато спільних властивостей. Їх рішення, що мають вид дерева, характерні тим, що при видаленні будь-якого відрізка з найкоротшою мережі ми повинні будемо виключити одну із заданих точок. Іншими словами, ми не можемо пройти по мережі з будь-якої заданої точки та повернутися в неї, без того щоб не пройти ті чи інші відрізки повторно. З цієї причини графічні рішення задачі з трьома точками і завдання з багатьма точками називаються деревами Штейнера. Відрізки прямих називаються ребрами, а точки, роль яких аналогічна точці P і які потрібно додати для побудови дерева, називаються точками Штейнера.

Завдання Штейнера для трьох точок дає також деяку інформацію про геометрії найкоротших дерев Штейнера.

По-перше, кожен кут дорівнює 120° або більше, а це означає, що кожна точка з'єднується з іншим деревом не більше ніж трьома ребрами.

По-друге, в кожній точці Штейнера сходяться рівно три ребра, утворюючи один з одним кути, в точності рівні 120 °.

По-третє, число ребер дерева завжди на одиницю менше сумарного числа заданих вихідних точок і точок Штейнера.

І нарешті, остання властивість: оскільки в кожній точці Штейнера сходяться рівно три ребра і принаймні одне ребро має стосуватися кожної із заданої множини точок, максимальне число точок Штейнера для будь-якого завдання на дві менше, ніж число заданих вихідних точок.

Виконаємо пошук мережі і розрахуємо сумарну довжину відрізків мережі для точок, які розташовані у вершинах рівностороннього трикутника, прямокутника і на «сходах» (рисунок 2.2).

У випадках a, d і g точки з'єднуються без додаткових проміжних точок і таке рішення називається мінімальним остовне деревом. Дерева Штейнера, отримані шляхом додавання вузлових точок, показані для випадків b, c, e, f, h і i. Тільки c, f і i є найкоротшими деревами Штейнера, або найкоротшими мережами. Числа під кожним рішенням вказують приблизну сумарну довжину відрізків мережі.

Завдання відшукання мінімального остовного дерева і найкоротшою мережі вирішувалися в застосуванні до планування топології телефонних мереж, трубопроводів і шосейних доріг. Рішення, наближені або точні, допомагають спланувати геометрію мережі і підрахувати необхідні кількості матеріалів. У більш складних формулюваннях завдання Штейнера можна враховувати такі фактори, як необхідність уникнення певних географічних властивостей місцевості, а також відшукувати найкоротші з'єднання між вузлами вже існуючих мереж.

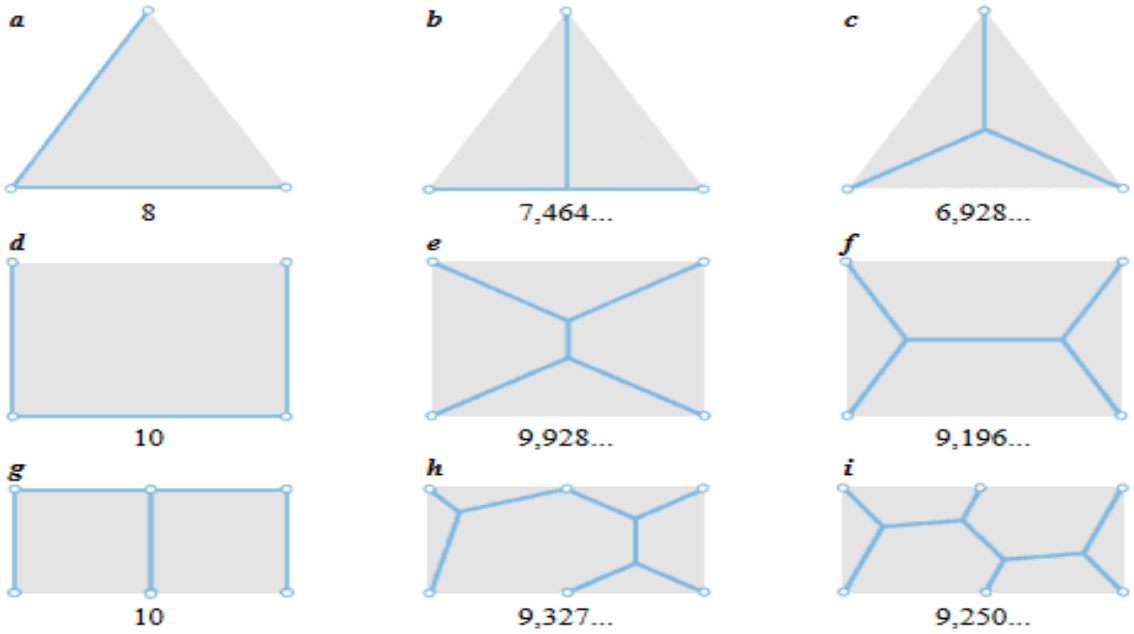


Рисунок 2.2 – Приклади з'єднання точок

Можливо, найбільш важливим практичним застосуванням завдання Штейнера є конструювання інтегральних електронних схем. Більш коротка мережа провідних ліній на інтегральній схемі вимагає меншого часу зарядки-розрядки в порівнянні з більш довгою мережею і підвищує, таким чином, швидкодія схеми. Однак завдання відшукання найкоротшою мережі на інтегральній схемі має іншу геометрію, так як провідники на ній зазвичай проходять лише в двох напрямках - горизонтальному і вертикальному (рисунок 2.3).

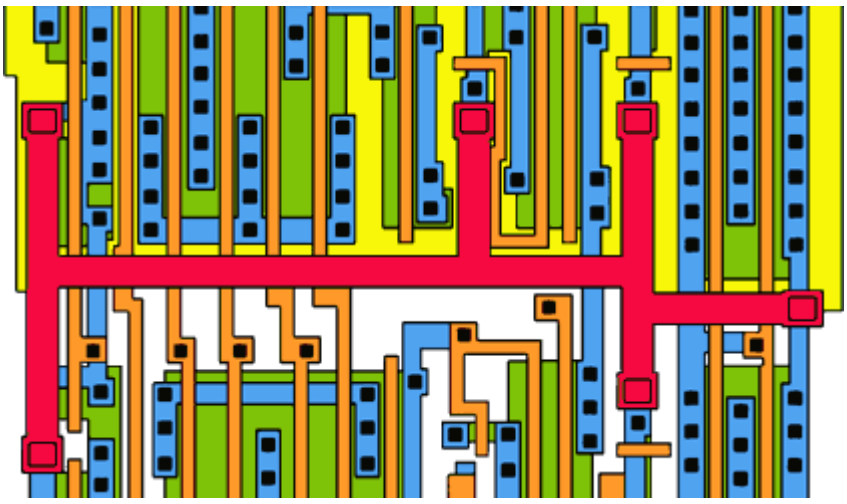


Рисунок 2.3 – Завдання Штейнера на інтегральної мікросхемі

Різновиди завдання про найкоротшою мережі застосовувалися при конструюванні електронних інтегральних схем, з тим щоб підвищити їх швидкодію. Найкоротша мережу з вертикальних і горизонтальних провідників, що пов'язують безліч висновків, виділена червоним кольором. Тут показані також провідники і висновки в більш глибоких шарах схеми.

Таке завдання, що отримала назву прямокутної завдання Штейнера, була вперше вивчена в 1965 році Морісом Хенань з Дослідницького центру ім. Томаса Уотсона корпорації IBM в Йорктаун-Хейтс (шт. Нью-Йорк). Як і в класичній задачі Штейнера, рішення для прямокутної її версії також містить точки Штейнера і вихідні точки, але ребра зустрічаються в них під кутом або 90 °, або 180 °. Хоча точки Штейнера можуть, здавалося б, лежати повсюдно в прямокутної задачі, так само як і в класичній задачі Штейнера, Хенан показав, що в найкоротшій прямокутної мережі на розташування точок Штейнера можна накласти певні обмеження. Через кожну вихідну точку проводяться горизонтальна і вертикальна прямі, і кожен перетин двох ліній дає можливе положення точки Штейнера. Щоб знайти найкоротшу мережу, алгоритм може розглянути всі підмножини можливих точок Штейнера. Однак у міру того, як число вихідних точок зростає, час рішення для кожного алгоритму, який здійснює повний перебір варіантів, росте експоненціально.

Таким чином при проектуванні топології CAN мережі можна застосувати подібний підхід. Додаючи додаткові точки і з'єднуючи кабелі під кутом 120 ° або близькому до нього, ми можемо досягти скорочення їх довжини, що призведе до зменшення їх вартості та зменшення величини падіння швидкості від відстані.

В якості центру з'єднання необхідно використовувати комутатори.

2.4 Обгрунтування структури бази даних

База даних системи повинна бути побудована в системі MySQL, тому що вона задовольняє пред'явлення вимог і найбільш поширена на хостингах і збірках серверів.

База даних являє собою 8 таблиць: A, B, C, D, E, Lines, Lines_with_turns, Ruler.

Використовується проста структура бази даних, тому що вона повністю задовольняє вимогам програми і не створює зайвих ускладнень.

Таблиці A, B, C, D, E містять інформацію про пристрої, які завдав користувач, і мають поля Name, X, Y. Name - ім'я з порядковим номером, X і Y - координати на карті.

Таблиця Ruler має такі ж поля, але використовується для роботи з масштабом.

Lines використовується для зберігання інформації про пристрої з'єднаних лініями. Має поля name_start (ім'я початкової точки), name_end (ім'я кінцевої точки), id (ідентифікаційний номер), distance (відстань між точками), has_turns (пряма лінія або ламана).

Lines_with_turns використовується для зберігання інформації, які з'єднані ламаними лініями. Має поля Start, End, Turn1, Turn2, Turn3, Turn4, Turn5, де перераховуються імена відповідних точок.

3 СИНТЕЗ СИСТЕМИ ВІДЕОКОНТРОЛЮ СТРІЧКОВИХ КОНВЕЄРІВ ВУГІЛЬНОЇ ШАХТИ

3.1 Вибір обладнання

У вугільних шахтах України широко використовується багатожильний неекранований телефонний кабель. Тому для економії часу і коштів систему контролю можна побудувати використовуючи цей кабель. У ділянках шахти, в яких кабель відсутній, але необхідний, необхідно його прокласти.

В даний час одним найбільш перспективних засобів побудови протяжних і високонадійних систем передачі інформації є польова шина CAN і створений на її основі протокол високого рівня CANopen. Цей тип мережі задовольняє вимогам умов шахти і є вибухобезпечним.

Для створення CAN мережі рекомендується використовувати неекрнірованние кабелі типу вита пара. Проізовдітелі обладнання з CAN інтерфейсом предьявляют для кабелів ще більш жорсткі вимоги.

Крім того, підземні умови можуть призвести до зниження якості зв'язку, наприклад, використання непаяною клемних з'єднань, скруток, близьке розташування до потужних пристроїв, що створює перешкоди.

Відповідно, необхідна перевірка можливості використання многожильного неекранованого телефонного кабелю в CAN мережі, який поширений в шахтах.

Як показали дослідження проведені в НГУ [16] максимальна протяжність лінії зв'язку в умовах шахти без ретрансляції, при якій достовірність передачі даних досягає 100%:

− 3500 м. На швидкості 10кбіт / с.

− 1900 м. На швидкості 20 кбіт / с.

− 1100 м. На швидкості 50 кбіт / с.

Крім того, тести з використанням протоколу CANopen і ретрансляцією пройдені успішно на всьому діапазоні швидкостей CAN шини від 10 кбіт / с до 1 Мбіт / с.

З цього був зроблений висновок, що в мережах з покадровой ретрансляцією можлива стабільна робота асинхронних сервісів протоколу CANopen.

Все вибране обладнання повинно підтримувати роботу з протоколом CANopen і з вибраним типом кабелю, а також підходити для умов шахти.

Відеокамери повинні бути розташовані в місцях, де потрібен нагляд і з'єднані кабелями.

Для здійснення контролю над обладнанням необхідні контролери відповідного типу.

Для з'єднання ділянок ліній, що ведуть до різного обладнання неоходімо використовувати комутатори.

У занадто довгих ділянках мережі буде спостерігатися зниження швидкості передачі даних і для виправлення цієї проблеми слід в таких місцях використовувати повторювачі.

Для організації роботи мережі необхідний сервер баз даних і web-сервер, які прийматимуть зображення з камер, обробляти їх, зберігати в базу даних і надавати диспетчеру доступ до них.

Також необхідно робоче місце диспетчера з комп'ютером, з якого буде здійснюватися доступ до интерфес роботи з відеокамерами.

3.2 Розробка структурної схеми системи

В результаті анализу вимог до технологичної схеми разміщения стрічкових конвеєрів (рисунок 2.1) була розроблена структурна схема комп'ютерної системи відеоконтролю вугільної шахти (рисунок 3.1).

Структура комплексу технічних засобів включає наземну та підземну частини.

Устаткування підземної частини системи складається:

- CAN-комутаторів;
- повторювачів;
- блоків керування конвеєрами з відеомодулями і відеокамерами.

Пропонується взаємодія між комутаторами даної системи виконати через іскробезпечне рішення на базі CAN-мережі.

3.3 Розробка функціональної схеми наземної частини системи

Функціональна схема наземної частини системи (рисунок 3.2) складається з сервера та контролера зв’язку, комутатора, веб-сервера і сервера бази даних встановлених на комп’ютері шахти і персональних комп’ютерів начальника участка, начальника зміни, механіка участка і диспетчерів.

Сервер зв’язку повинен виконувати запроси актуальних даних з блоків керуавння і відеокамер системи контролю. Зв'язок між сервером та персональними комп’ютерам шахтоуправління виконуєтьс за допомогою інтерфейсу Ethernet.

Зв'язок між сервером зв’язку та ПК шахтоуправління виконується за допомогою мережевого комутатора. До нього підключаються ПК диспетчерів та шахтоуправління.

На схемі зображено всі необхідні компоненти системи.

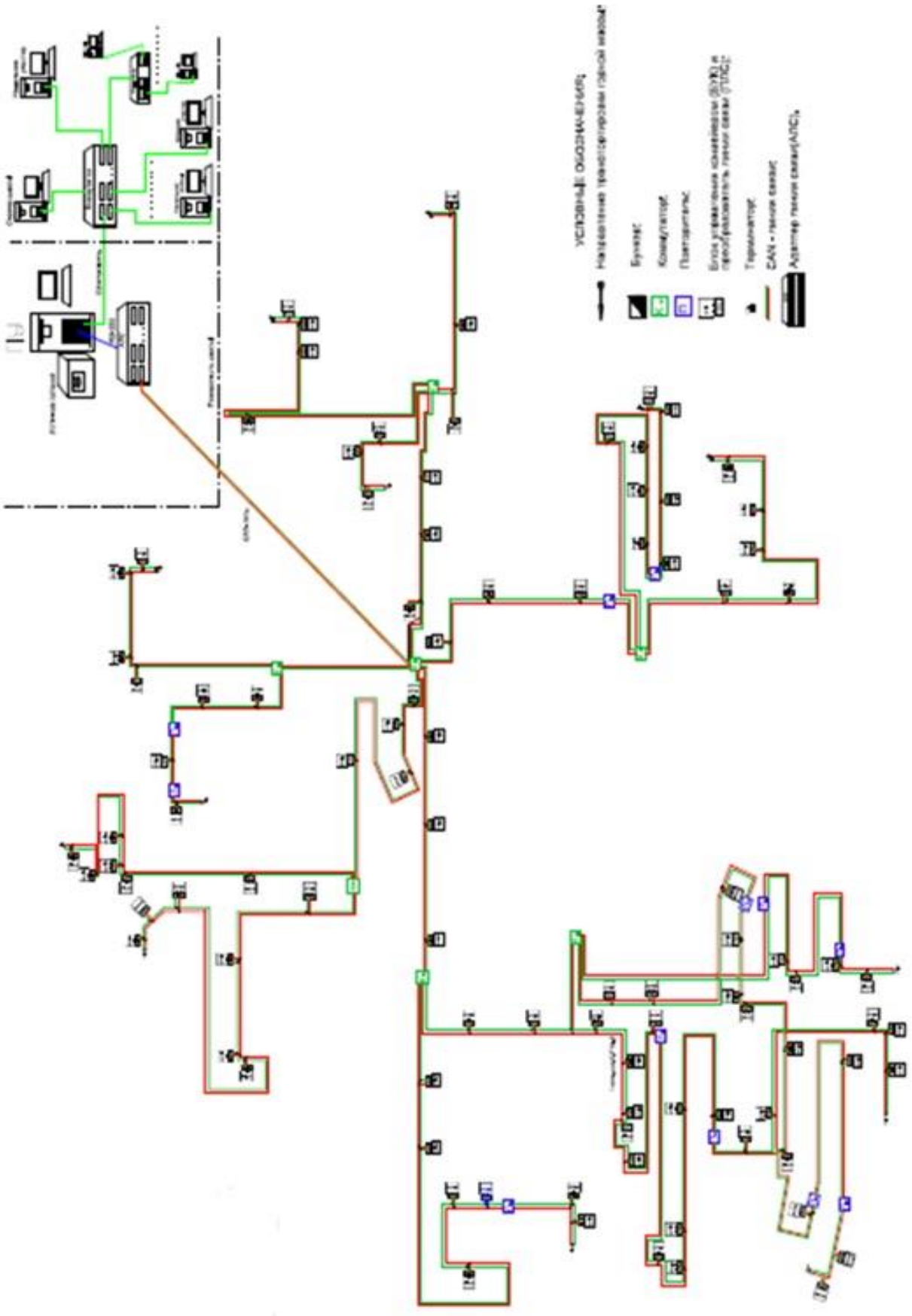


Рисунок 3.1 − Структурна схема комп'ютерної системи відеоконтролю

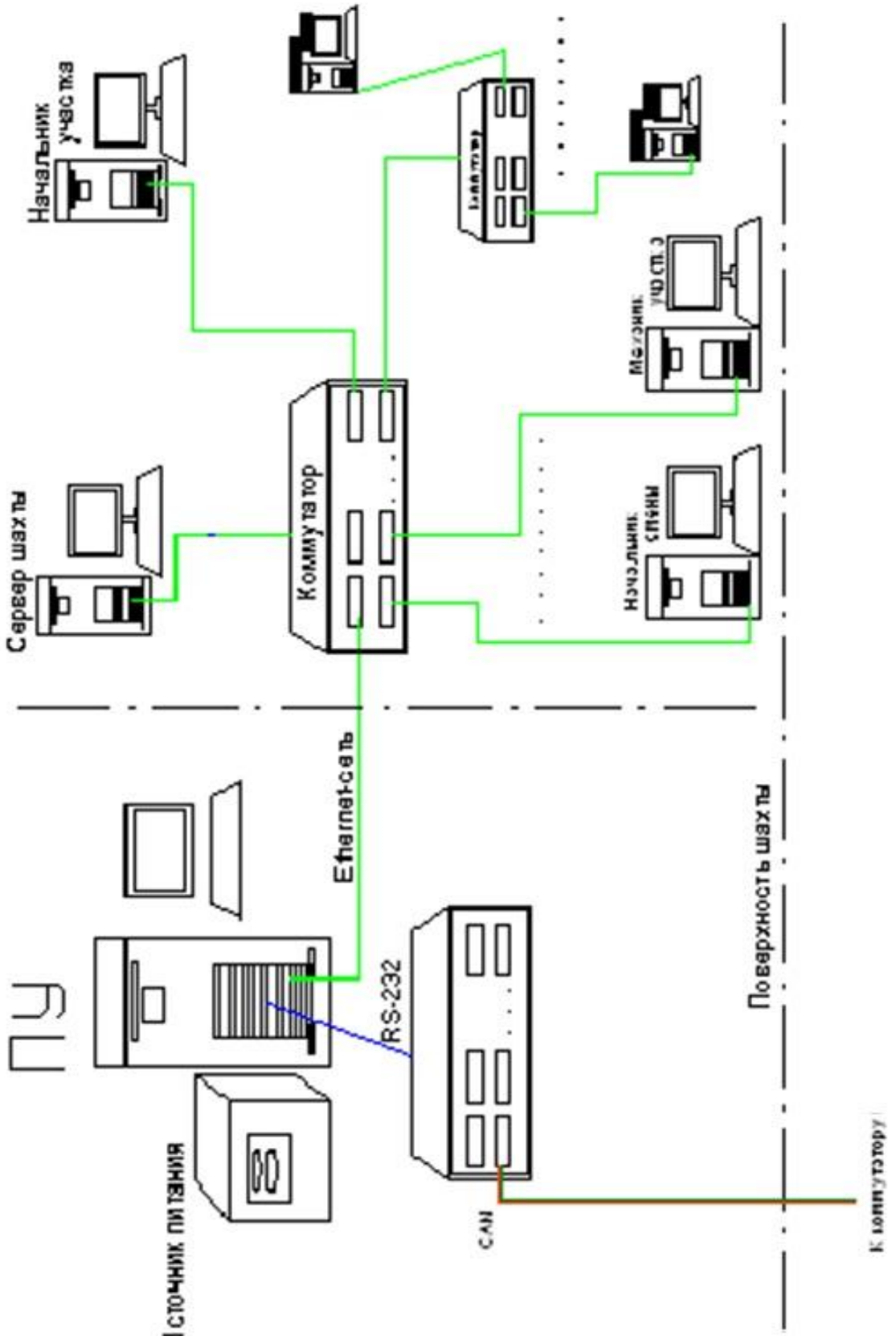


Рисунок 3.2 – Функціональна схема наземної частини системи

4 РАЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Постановка завдання по розробці підсистеми проектування топології CAN-мережі

Основні функції, які програма повинна виконувати:

- завантаження користувачем зображення плану шахти;
- нанесення на карту пристроїв (комутаторів, повторювачів, контролерів);
- з'єднання пристроїв лініями;
- введення масштабу;
- відображення довжин з'єднаних ділянок;
- експорт отриманого зображення з нанесеною топологією;
- експорт таблиці відстаней.

Програма повинна бути доступна користувачеві в веб-браузері і написана на відповідній мові програмування.

4.2 Призначення і сфера використання програмного забезпечення

Програма призначена для проектування топології мереж у вугільних шахтах. Вона надає проектувальнику необхідні для проектування інструменти. Її використання полегшує і прискорює роботу фахівця, дозволяє йому зберігати отриманий план, користуватися інформацією про довжину ліній, сортувати по довжині, експортувати дані в таблицю та інше.

4.3 Обгрунтування технічних характеристик

4.3.1 Обгрунтування структури і методів реалізації програмного забезпечення

Програма доступна користувачеві у вікні його веб-браузера. Програма має головну сторінку, де знаходяться кнопки інтерфейсу і поле для роботи з картою, а також ряд додаткових сторінок, в яких користувач виконує додаткові завдання (завантаження зображення, збереження роботи, висновок таблиць в файл).

Відповідно, за роботу з різними сторінками і різними подзадачами відповідають різні частини коду програми, що знаходяться в різних файлах.

Основні структурні завдання програми:

− інтерфейси різних сторінок програми;

− отрисовка ліній і точок;

− розрахунки відстаней і масштабу;

− занесення і висновок різної інформації в базу даних.

Більш детально складові частини програми описані в пункті 4.4.3.2.

4.3.2 Опис алгоритму і функціонування програми

Робота програми ґрунтується на тому, що в HTML форма з параметром input type = "image" являє собою кнопку, в яку можна завантажити зображення будь-якого розміру, а також дозволяє отримувати координати місця, в яке клікнув користувач. Відповідно всі зображення карти і є цією кнопкою.

Залежно від кнопок інтерфейсу, які натискає користувач, виконується відповідний програмний код. Наприклад, режими роботи з картою перемикаються і кліки користувача по карті встановлюють різні пристрої або з'єднують їх лініями. В інших випадках відкриваються сторінки з кнопками для імпорту, експорту та іншого.

Різних видів пристроїв відповідають різні імена і різні таблиці в базі даних.

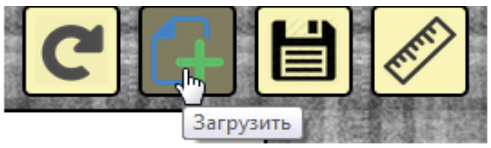
При нанесенні точок координати і імена точок заносяться до відповідних таблиць в базі даних. При їхньому з'єднанні імена початку і кінця і відстань між ними теж заносяться в свою таблицю. Теж саме відбувається і при з'єднанні ламаними і непрямими лініями.

При відображенні точок і ліній, а також їх збереження в зображення або таблицю відбувається читання потрібної інформації з бази даних.

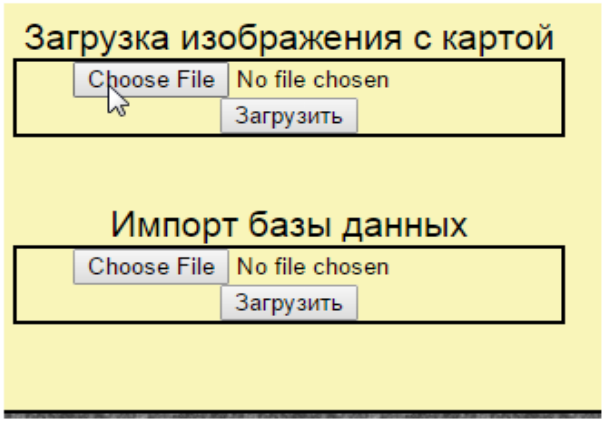
При імпорті, експорті, завантаження зображення і інших операціях так само виконується відповідний програмний код.

Приклади роботи деяких базових функцій програми: Завантаження плану шахти користувачем показана на рис. 4.1.

Шаг 1



Шаг 2



Шаг 3

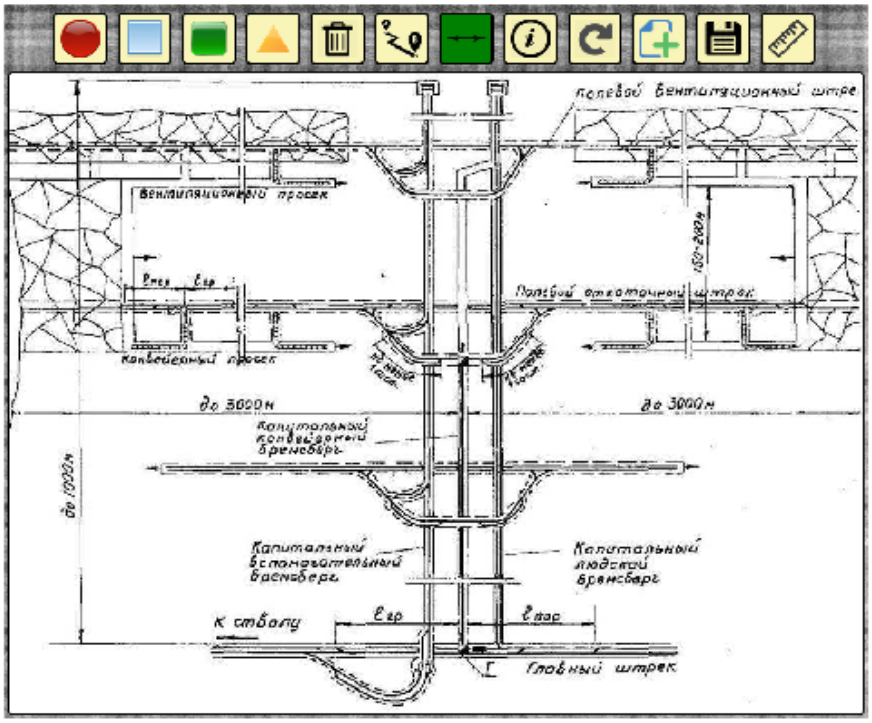


Рисунок 4.1 – Завантаження карти користувача

Нанесення і з'єднання точок – рисунок 4.2.

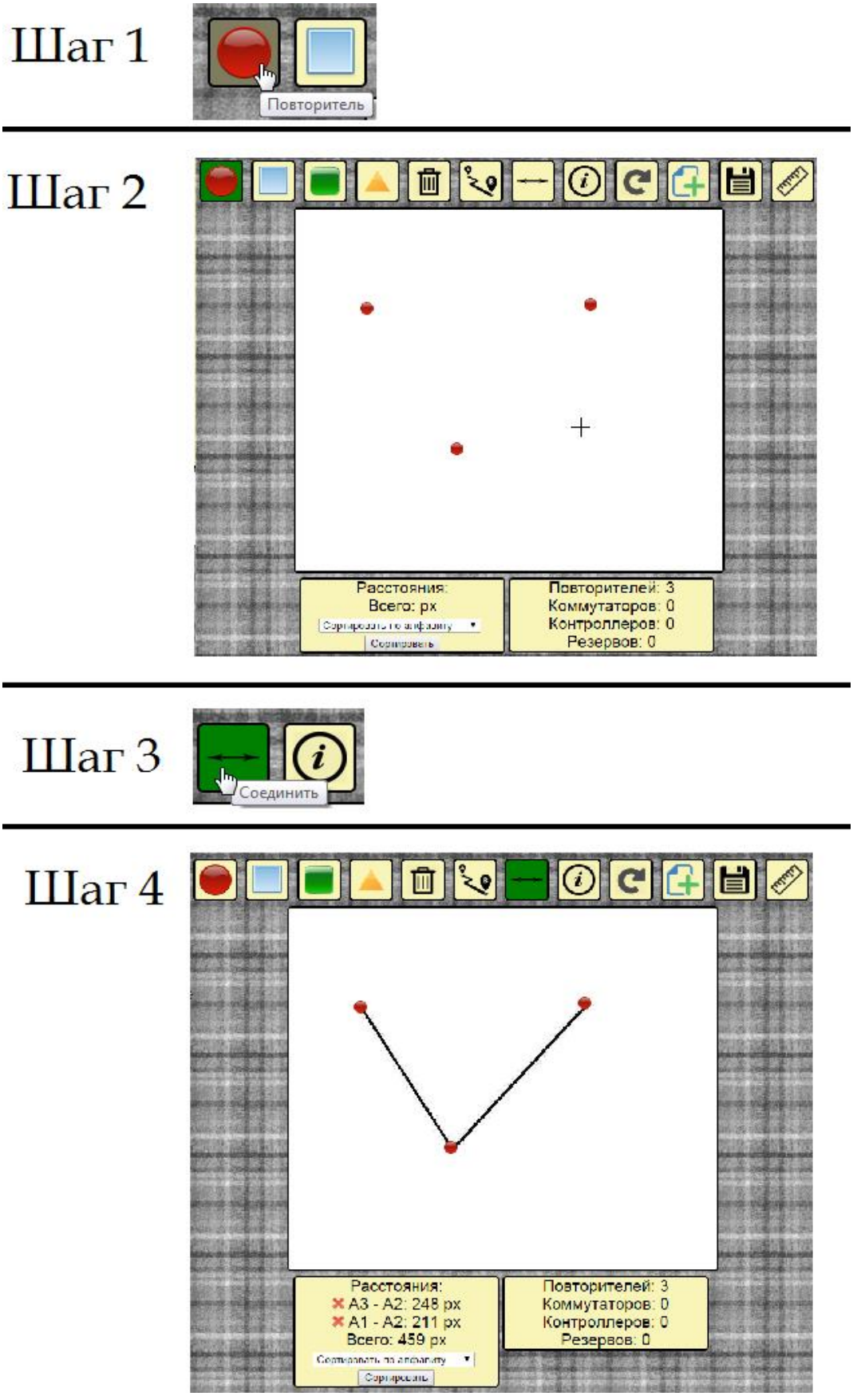


Рисунок 4.2 – Установка і з'єднання пристроїв

З'єднання точок ламаною лінією, рисунок – 4.3.

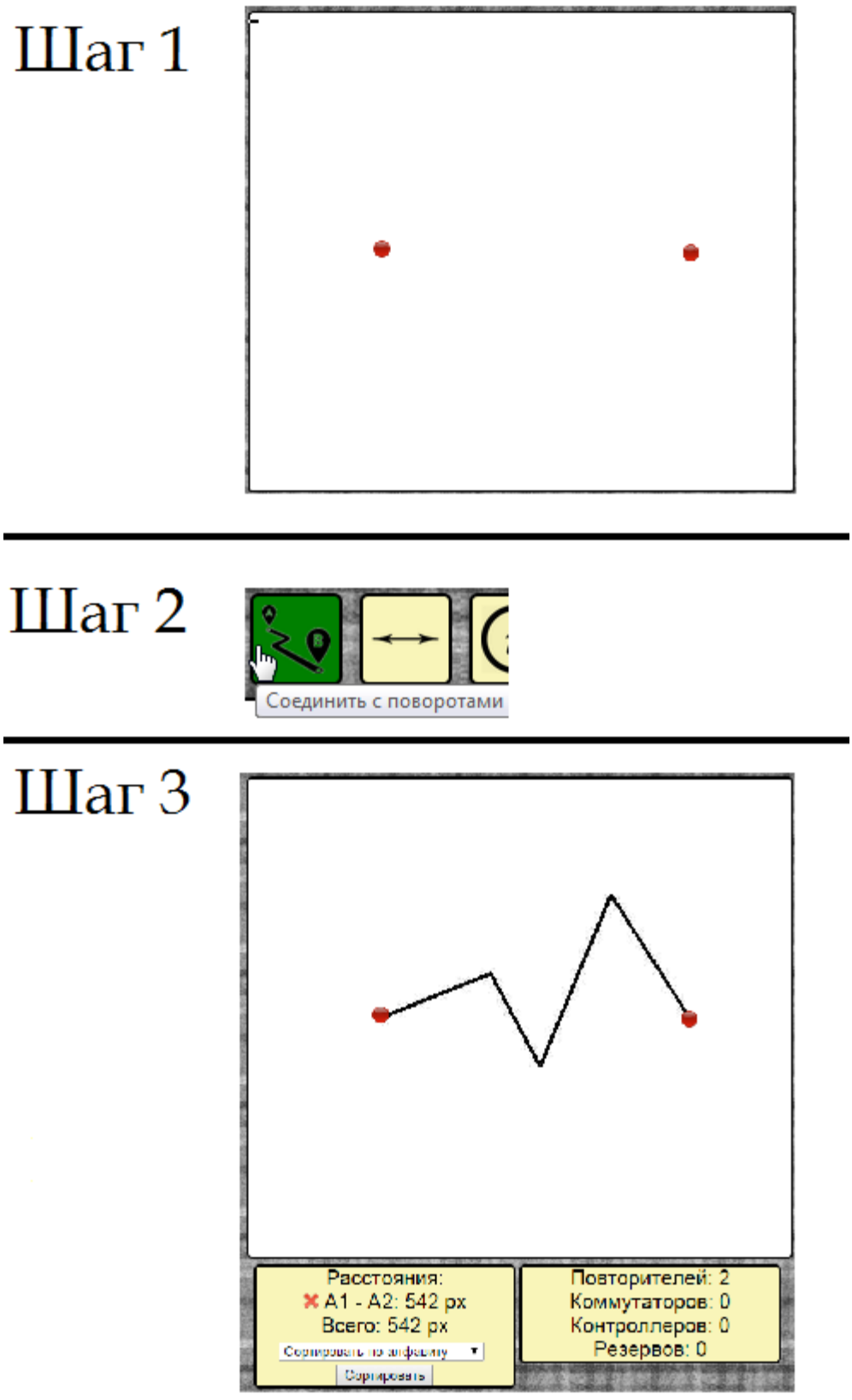


Рисунок 4.3 – З'єднання точок ламаною лінією

Видалення точки, рисунок − 4.4.

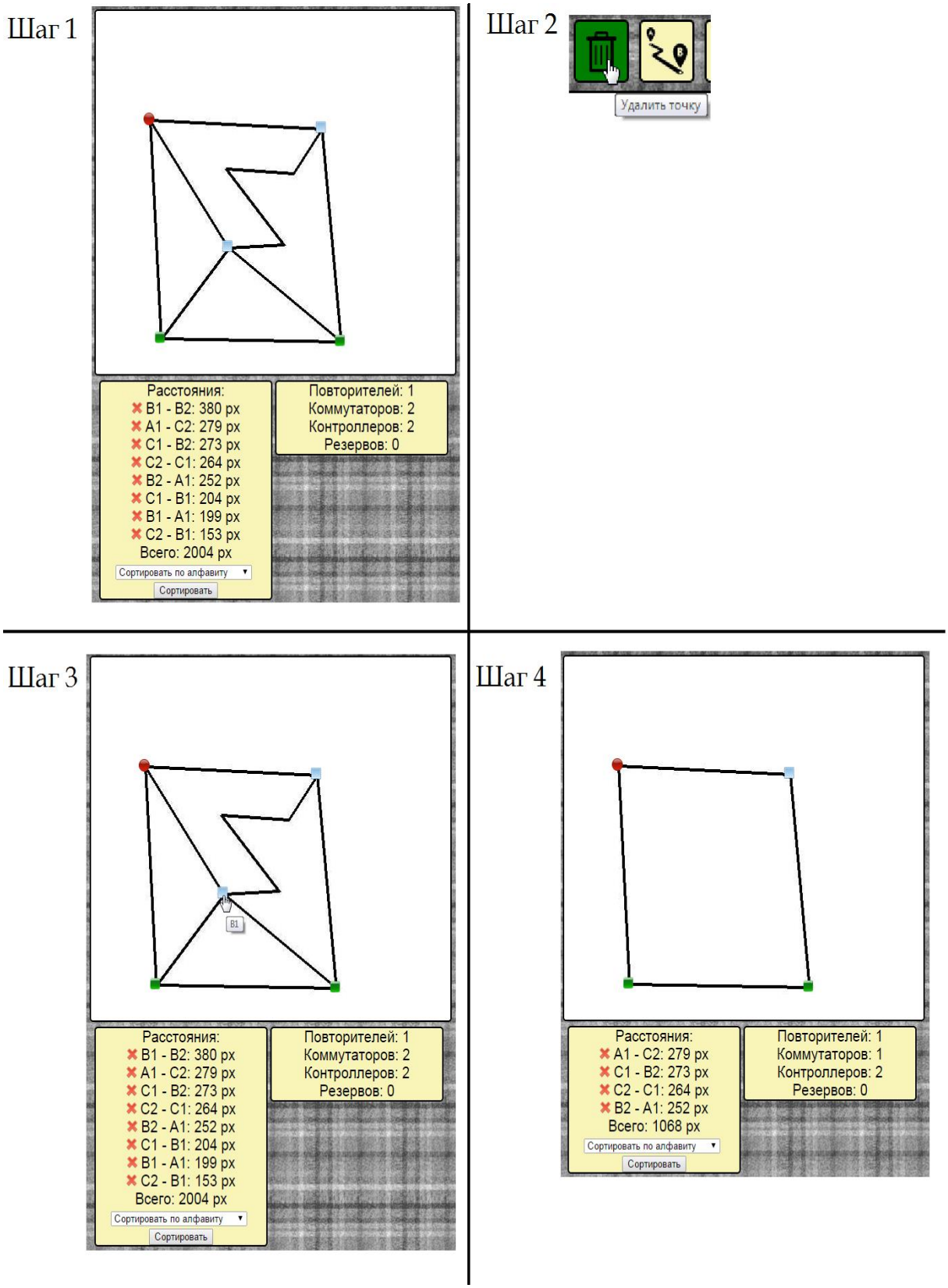


Рисунок 4.4 – Видалення точки

4.3.3 Опис і обґрунтування вибору методу організації вхідних та вихідних даних

Вхідні дані

Активний режим роботи з картою зберігається в txt-файл, щоб після оновлення сторінки режим не змінився. Використовується txt-файл, щоб зайвий раз не звертатися до бази даних.

Координати кліка зберігаються в базу даних, щоб була можливість в подальшому виводити їх на екран, виконувати обчислення, видаляти, експортувати в файл та інше.

Зображення карти зберігається в папці img для зручності роботи.

Вихідні дані

Кількість пристроїв підраховується з кількості записів у відповідній таблиці бази даних і виводиться на панелі інформації.

Довжини ділянок ліній розраховуються в програмі і заносяться в масив для можливості виведення їх на екран, виконання обчислень і видалення.

Таблиця зі списком пристроїв і відстаней між ними експортується програмою в .csv файл, щоб користувач міг їм користуватися не запускаючи програму.

Зображення з нанесеною картою зберігається програмою в jpg-файл, якщо це необхідно користувачу.

База даних експортується, що дозволяє зберегти стан поточної топології і завантажити його пізніше, щоб продовжити над ним роботу.

4.3.4 Вимоги до функціональних характеристик

Програма повинна реалізовувати наступні функції:

− завантаження користувачем карти в форматі jpg, gif, png .;

− додавання користувачем пристроїв на карту;

− з'єднання пристроїв лініями (прямими і ламаними);

− видалення пристроїв;

− установка масштабу користувачем;

− висновок відстаней між пристроями в пікселях (якщо користувач не поставив масштаб) і в метрах (якщо масштаб заданий);

− видалення всіх пристроїв і ліній одночасно;

− збереження карти із зазначеними пристроями та лініями в графічний файл;

− збереження таблиці з пристроями і відстанями в .sql файл;

− висновок текстових підказок при наведенні миші на кнопку елемента інтерфейсу;

− запис імен та координат точок в базу даних;

− висновок імені пристрою при наведенні миші на неї;

− відображення кількості кожного типу пристроїв.

4.4 Опис розробленої програми

4.4.1 Загальні відомості

4.4.1.1 Позначення і найменування програми

Програма має наступні атрибути:

- Адреса головного файлу - /main_coords.php
- Розмір файлу - 40225 байт
- Версія файлу - 1.0
- Версія продукту - 1.0
- початкового файлу - main_coords.php
- Назва продукту - «MineDraft»
- Мова – українська

4.4.1.2 Програмне забезпечення, необхідне для функціонування програми

Для використання програми необхідний будь-який з наступних веб-браузерів:

− Internet Explorer 7 і вище;

− Mozilla Firefox 3.5 і вище;

− Google Chrome 10.0 і вище.

Програма повинна бути розміщена на хостингу або локальному веб-сервері з підтримкою PHP 5.6, бібліотекою GD і MySQL.

4.4.1.3 Мови програмування, на яких написана програма

Програма написана на мові PHP 5.6.

4.4.2 Функціональне призначення

4.4.2.1 Типи вирішуваних завдань

Программа може вирішувати наступні види задач:

- завантаження зображення з планом шахти;
- нанесення на план пристроїв і кабелів;
- розрахунок довжин сегментів ліній і їх сортування;
- відображення кількості і типів нанесених пристроїв;
- збереження нанесеною топології в графічний файл;
- збереження таблиці довжин кабелів.

4.4.2.2 Функціональні обмеження

Дозволені формати для зображення плану шахти - jpg, gif, png.

Максимальна кількість пристроїв - 80.

4.4.3 Логічна структура

4.4.3.1 Алгоритм програми

Докладний алгоритм програми представлений на рисунках 4.5-4.7

Рисунок 4.5 – Схема алгоритму програми (частина 1)

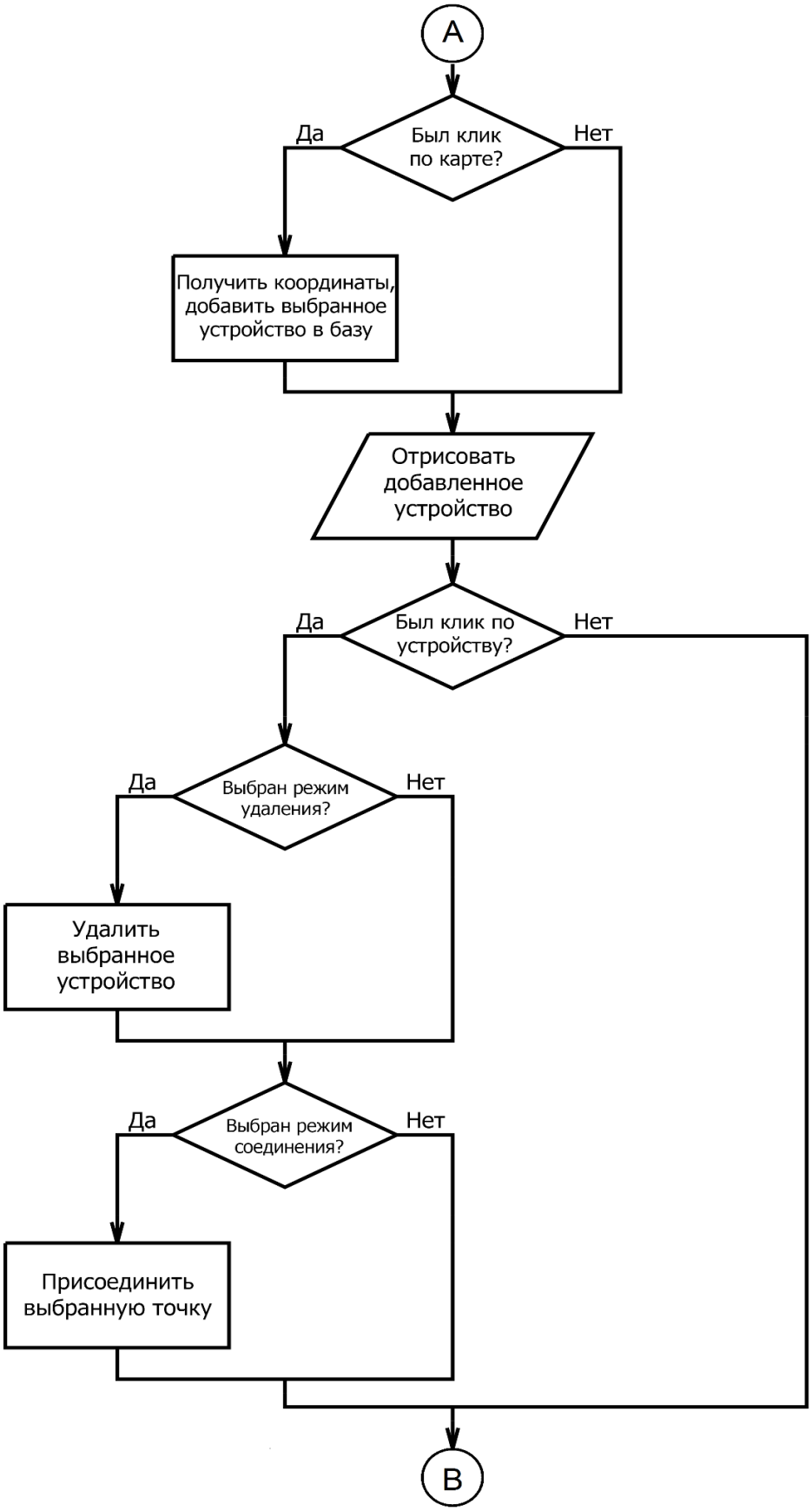


Рисунок 4.6 – Схема алгоритму програми (частина 2)

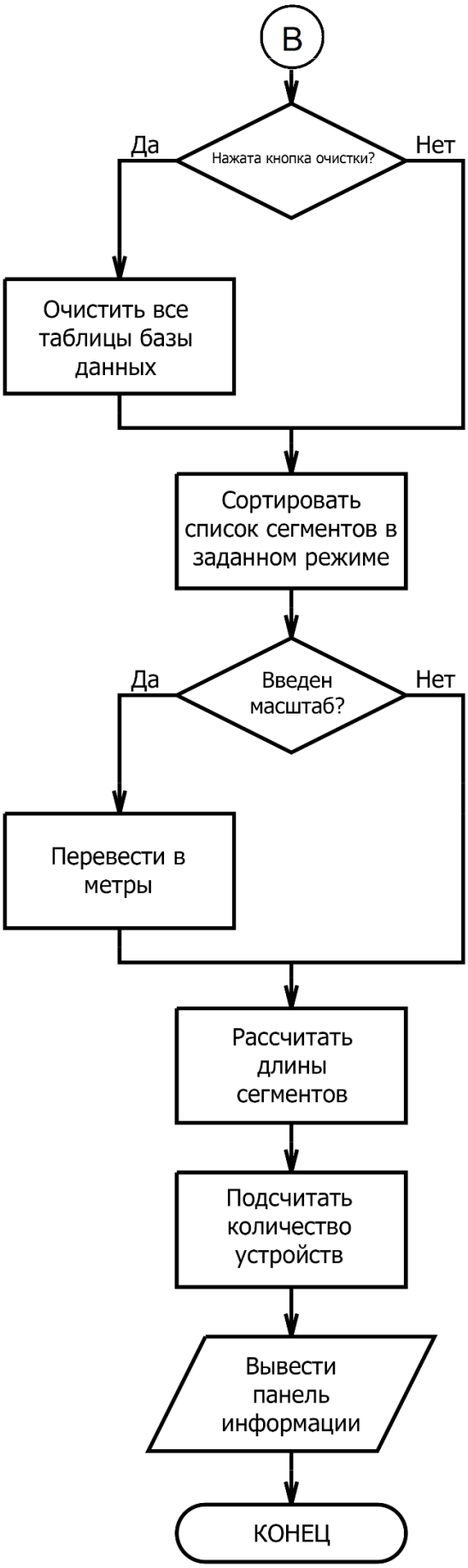


Рисунок 4.7 – Схема алгоритму программи (частина 3)

4.4.3.2 Структура програми з описом функцій складових частин

Програма складається з ряду файлів, які логічно розділені і виконують різні підзадачі:

− main_coords.php - верстка головної сторінки, відображення карти і елементів інтерфейсу, обробка дій користувача пов'язаних з його взаємодією з інтерфейсом програми;

− db.php - містить інформацію про логін та пароль бази даних, виконує підключення до неї;

− distance.php - отримує дані про назви точок початку і кінця ліній, які завдав користувач, і заносить їх в масиви для відтворення ліній на карті;

− distance_info.php - отримує дані про назви точок початку і кінця ліній, які завдав користувач, сортує відповідно до обраного режиму і заносить їх в масиви для обчислення довжини ліній;

− distance_with_turns.php - отримує дані про назви ламаних, непрямих ліній і заносить їх в масиви;

− export.php - експортує базу даних в файл;

− icon_coords.php - виконує отрисовку точок;

− icon_coords_save.php - виконує отрисовку точок при збереженні зображення в файл;

− import.php - імпортує базу даних з файлу;

− points.php - виконує отрисовку ліній;

− resize.php - зменшує зображення до потрібного розміру;

− ruler_distance.php − інтерфейс для введення масштабу;

− ruler_start.php − інтерфейс для введення першої точки;

− ruler_end.php − інтерфейс для введення другої точки;

− save.php − збереження зображення з нанесеною топологією в файл;

− save_links.html − сторінка з варіантами збереження;

− style.css − дані про форматування веб−сторінок;

− table_save.php − збереження бази даних в csv-файл;

− upload.php − сторінка для завантаження зображення або бази даних.

Частина цих розроблених файлів представлені в Додатку А.

4.4.4 Використовувані технічні засоби

Програма повинна бути розміщена на хостингу або локальному веб-сервері з підтримкою PHP 5.6, бібліотекою GD і MySQL.

4.4.5 Виклик і завантаження

Програма викликається у вікні браузера. Доступ може бути здійснений за допомогою Інтернету або ж локально на комп'ютері користувача.

4.4.6 Вхідні дані

Вхідні дані представлені в таблиці 4.1.

Таблиця 4.1 – Вхідні дані

Вхідні дані	
Активний режим роботи з картою	Зберігається в tool_name.txt
Координати кліка	Зберігається в базу даних
Зображення карти	Зберігається в папці img

4.4.7 Вихідні дані

Вихідні дані представлені в таблиці 4.2.

Таблиця 4.2 – Вихідні дані

Вихідні дані	
Кількість пристроїв	Підраховується з кількості записів у відповідній таблиці бази даних
Довжини ділянок ліній	Розраховуються в програмі і заносяться в масив
Таблиця зі списком пристроїв і відстаней між ними	Експортується програмою в файл *.csv
Зображення з нанесеною картою	Зберігається програмою в .jpeg файл
Бази даних	Експортується в файл *.sql

4.4.8 Очікувані техніко-економічні результати від впровадження системи

Використання програми дозволяє проектувальнику нанести топологію мережі швидко і з високою точністю. Програма автоматично відображає кількість і тип використаних пристроїв, розраховує довжини сегментів і переводить їх в метри, якщо користувач вкаже масштаб. Точна інформація про довжину кабелів, кількість і тип пристроїв дозволить уникнути помилок при їх прокладанні і заощадить час. Крім того, програма дозволяє користувачеві наочно бачити найдовші ділянки мережі, які можуть привести до зниження швидкості передачі даних, і які вимагають установки повторювача. Також програма надає користувачеві можливість експортувати в файли зображення з нанесеною картою і таблицю зі списком сегментів і відстаней, що дозволяє зберегти результати роботи і користуватися ними в подальшому без використання програми.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Постановка завдання експерименту і обгрунтування методики

За допомогою експерименту необхідно перевірити спосіб оптимізації топології мережі за допомогою дерева Штейнера. Після проведення експерименту можна буде оцінити чи дає цей спосіб можливість зменшити довжину кабелів і наскільки ефективно. Для цього буде побудовано кілька варіантів з'єднань для трьох і чотирьох точок, розрахована довжина витрачених кабелів. Після збору всіх даних можна буде визначити найоптимальніший спосіб з'єднання точок. Побудова варіантів буде проводитися в розробленій програмі, підрахунок довжин ділянок і сумарної довжини буде проведено в ній же.

5.2 Проведення експерименту для проектування з’єднань

5.2.1 Трьох точок

На цьому етапі виконаємо з'єднання трьох точок чотирма різними способами з використанням розробленого ПО і підрахуємо плановану загальну довжину сегментів мережі для наступних сполук (рисункм 5.1-5.4):

- послідовного;
- з довільною додатковою точкою;
- з додатковою рівновіддаленою точкою;
- з використанням дерева Штейнера.

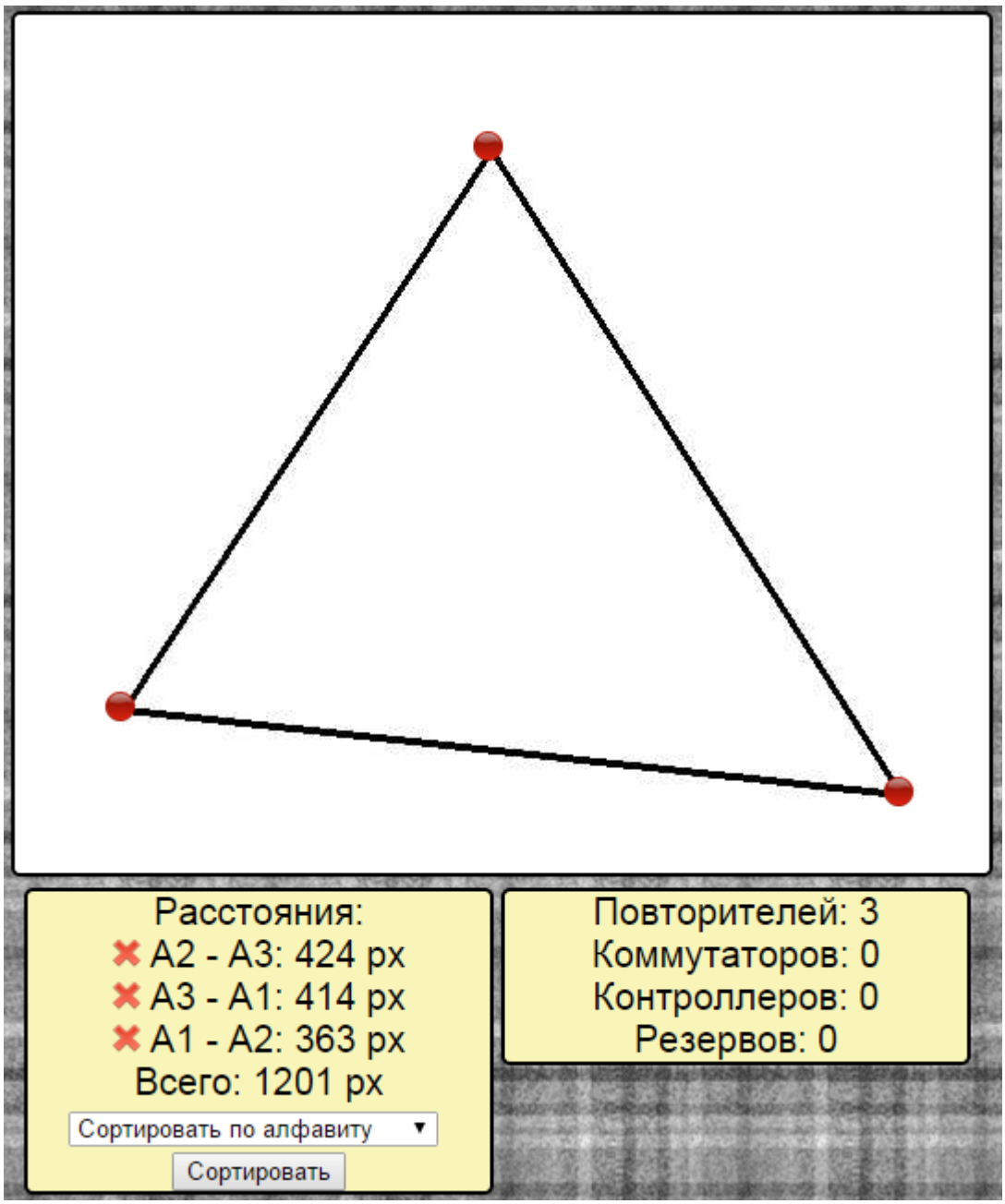


Рисунок 5.1 – Три точки мережі з'єднані послідовно у вигляді трикутника

На рисунку 5.1 точки мережі з'єднані послідовно у вигляді трикутника, сумарна довжина − 1201 піксель.

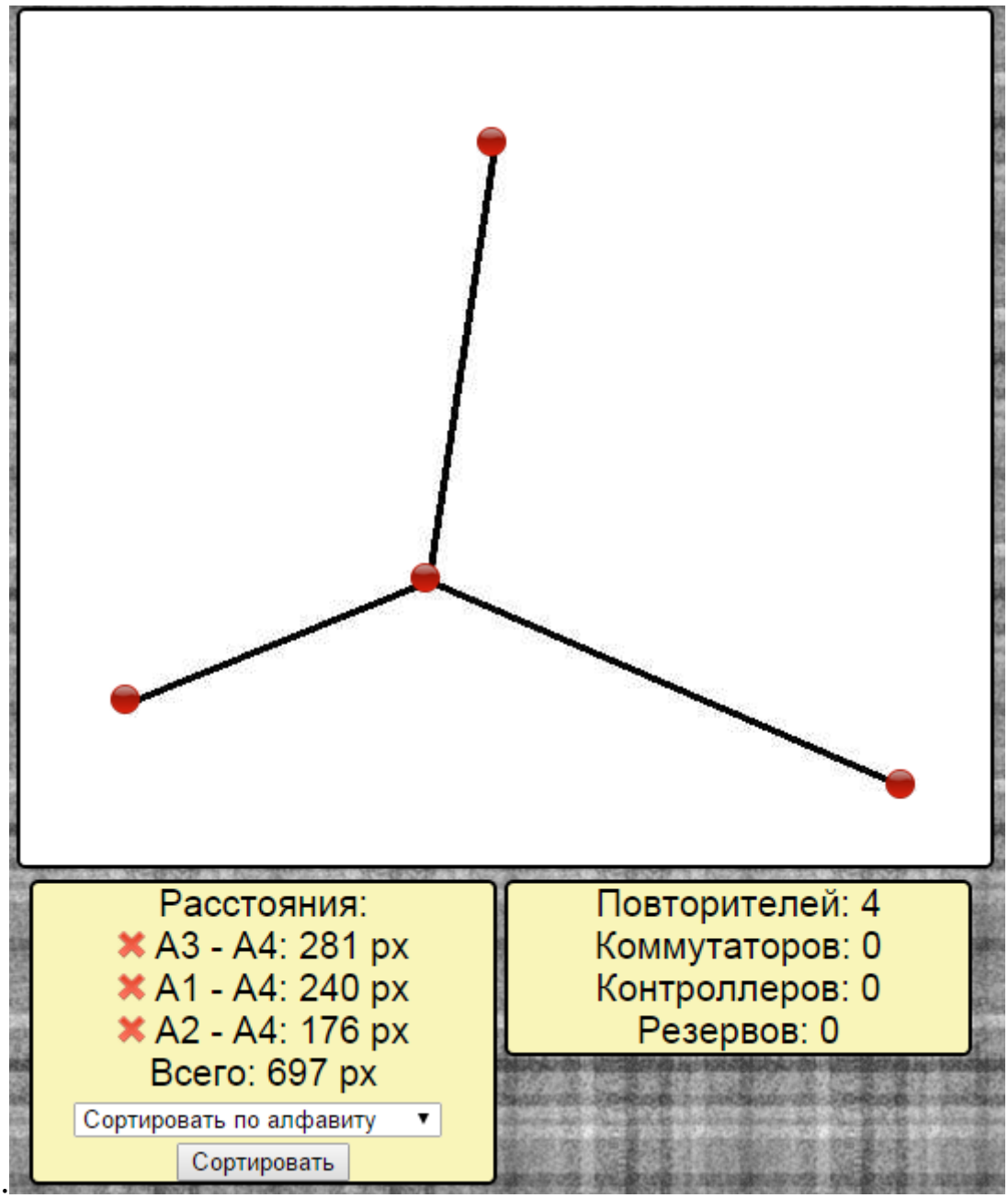


Рисунок 5.2 – Центральна точка розташована довільним чином від трьох інших

На рисунку 5.2 до трьох точок мережі додана четверта центральна, що розташована довільним чином, сумарна довжина − 697 пікселей.

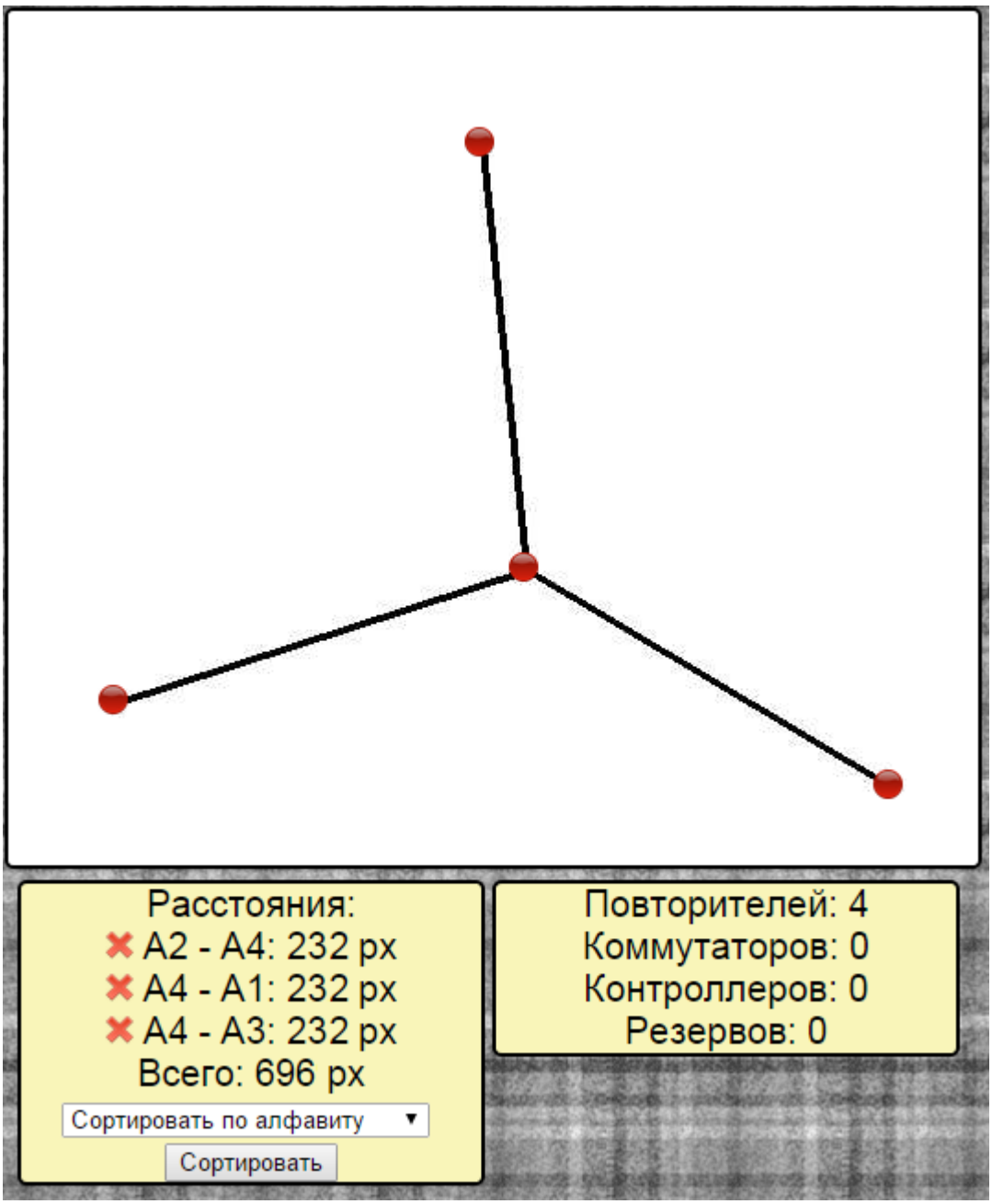


Рисунок 5.3 – Центральна точка розташована рівновіддалено від трьох інших точок

На рисунку 5.3 до трьох точок мережі додана четверта центральна, що розташована рівновіддалено (на 232 пікселя) від інших, сумарна довжина – 696 пікселей.

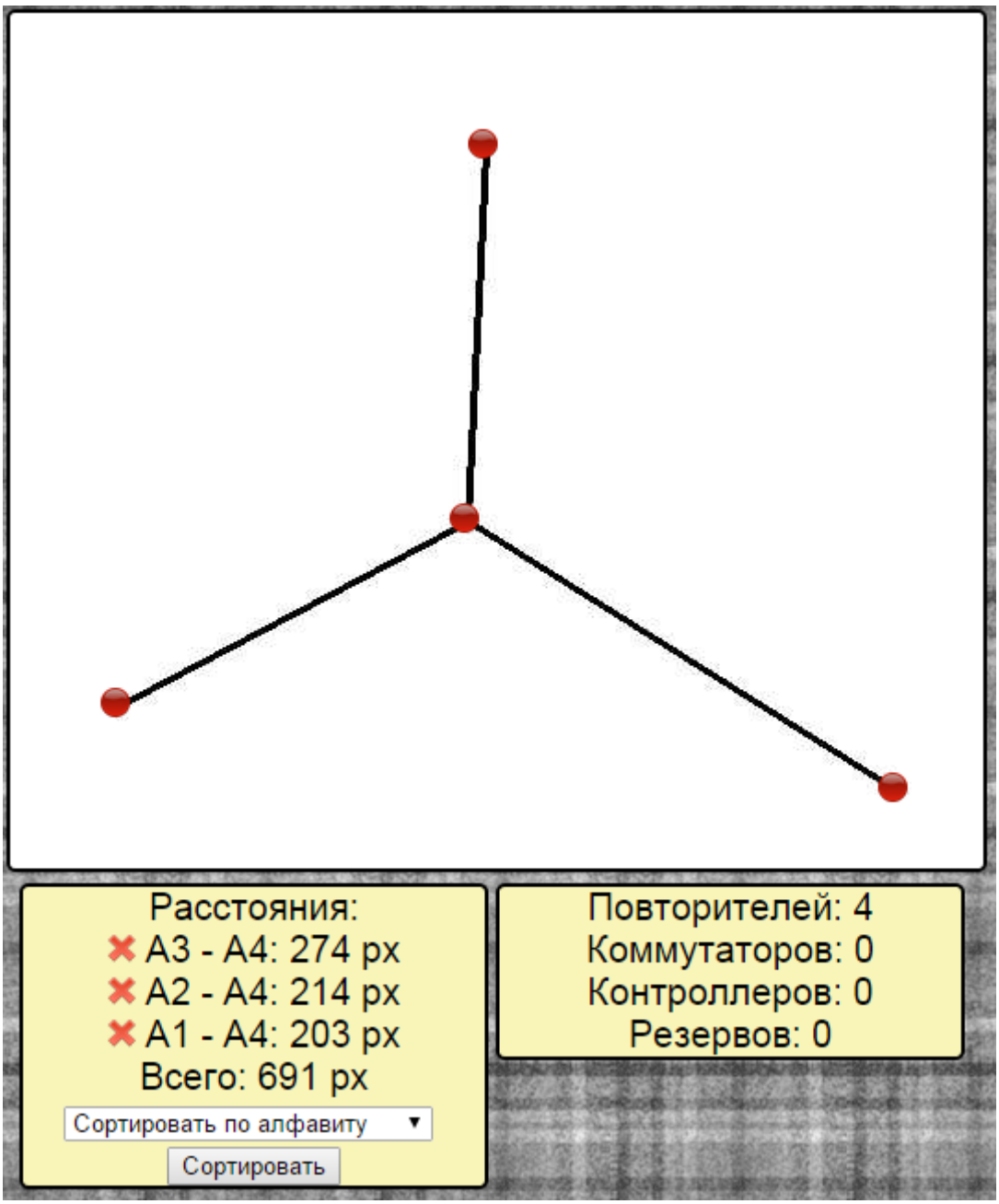


Рисунок 5.4 – Три точки з'єднані з використанням дерева Штейнера

На рисунку 5.4 до трьох точок мережі додана четверта центральна і виконано з’єднання з використанням дерева Штейнера, сумарна довжина – 691 піксель.

5.2.2 Чотирьох точок, які розташовані симетрично

На цьому етапі виконаємо з'єднання різними способами з використанням розробленого ПО чотирьох точок мережі, що розташовані симетрично, і підрахуємо плановану загальну довжину сегментів мережі для наступних сполук (рисункм 5.5-5.9):

- послідовного;
- з довільною додатковою точкою;
- з використанням дерева Штейнера, що розташоване горизонтально;
- з використанням дерева Штейнера, що розташоване вертикально.

Рисунок 5.5 – Чотири точки розташовані симетрично, початковий варіант

На рисунку 5.5 показано початкове положення чотирьох точок мережі,

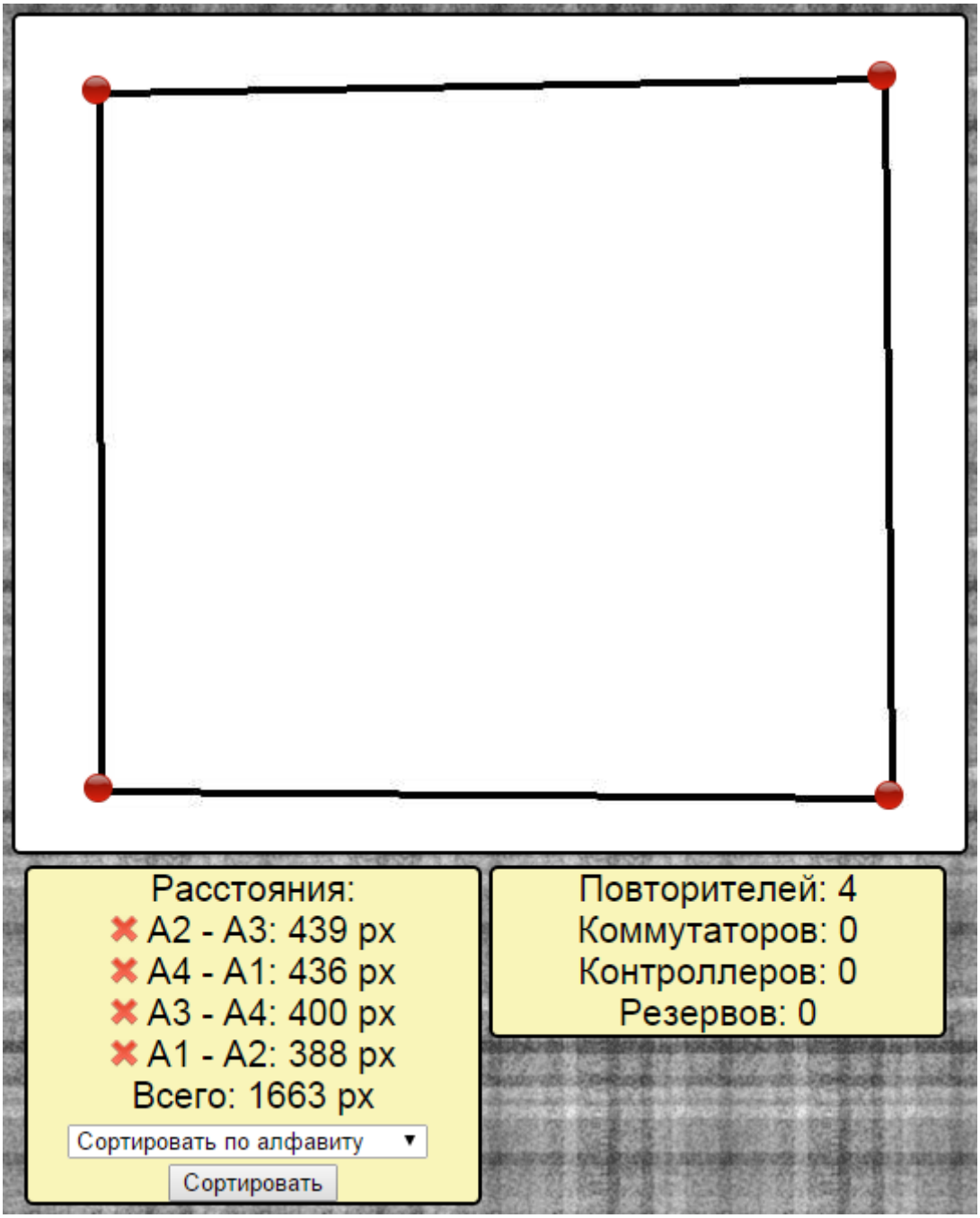


Рисунок 5.6 – Последовне з’єднання чотирьох точок

На рисунку 5.6 чотири точки мережі з'єднані послідовно у вигляді квадрата, сумарна довжина − 1663 пікселя.

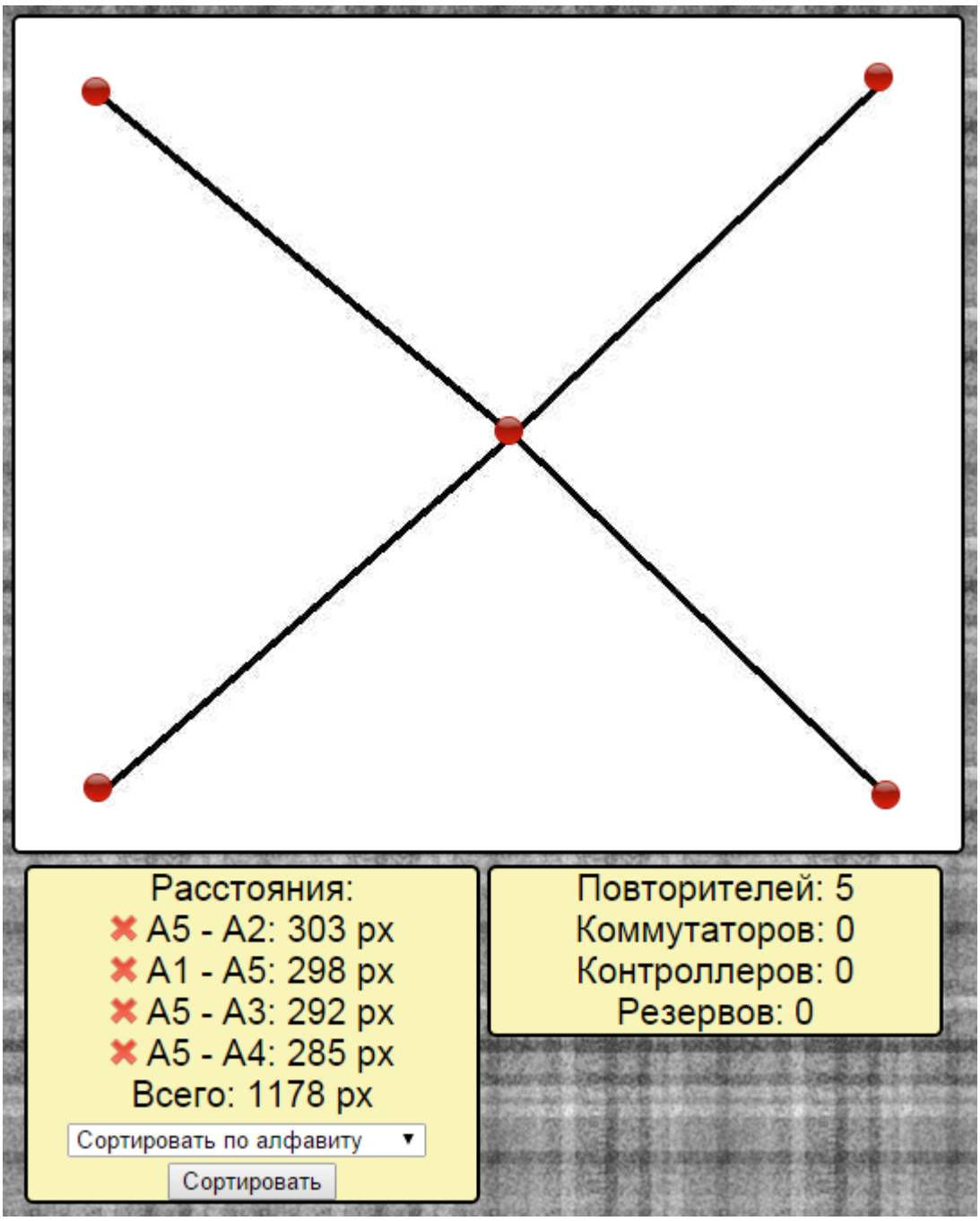


Рисунок 5.7 – Центральна точка мережі розташована довільним чином і з’єднана з чотирьма іншими

На рисунку 5.7 до чотирьох точок мережі під’єднана п’ята, що розташована приблизно в центрі, сумарна довжина − 1178 пікселей.

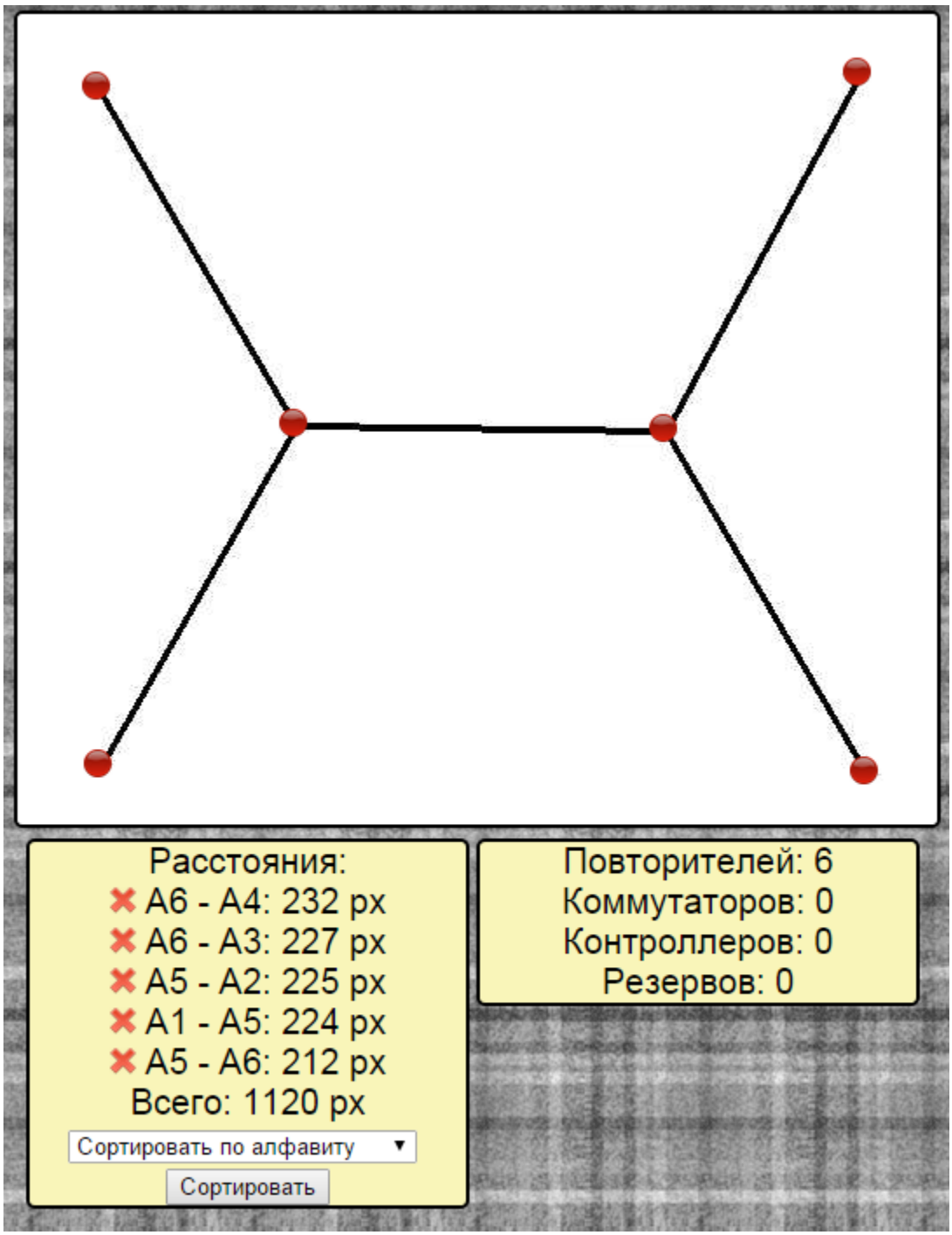


Рисунок 5.8 – З’єднання чотирьох точок мережі з використанням дерева Штейнера, що розташоване горизонтально

На рисунку 5.8 чотири точки мережі з’єднані з використанням дерева Штейнера, що розташоване горизонтально, сумарна довжина − 1178 пікселей.

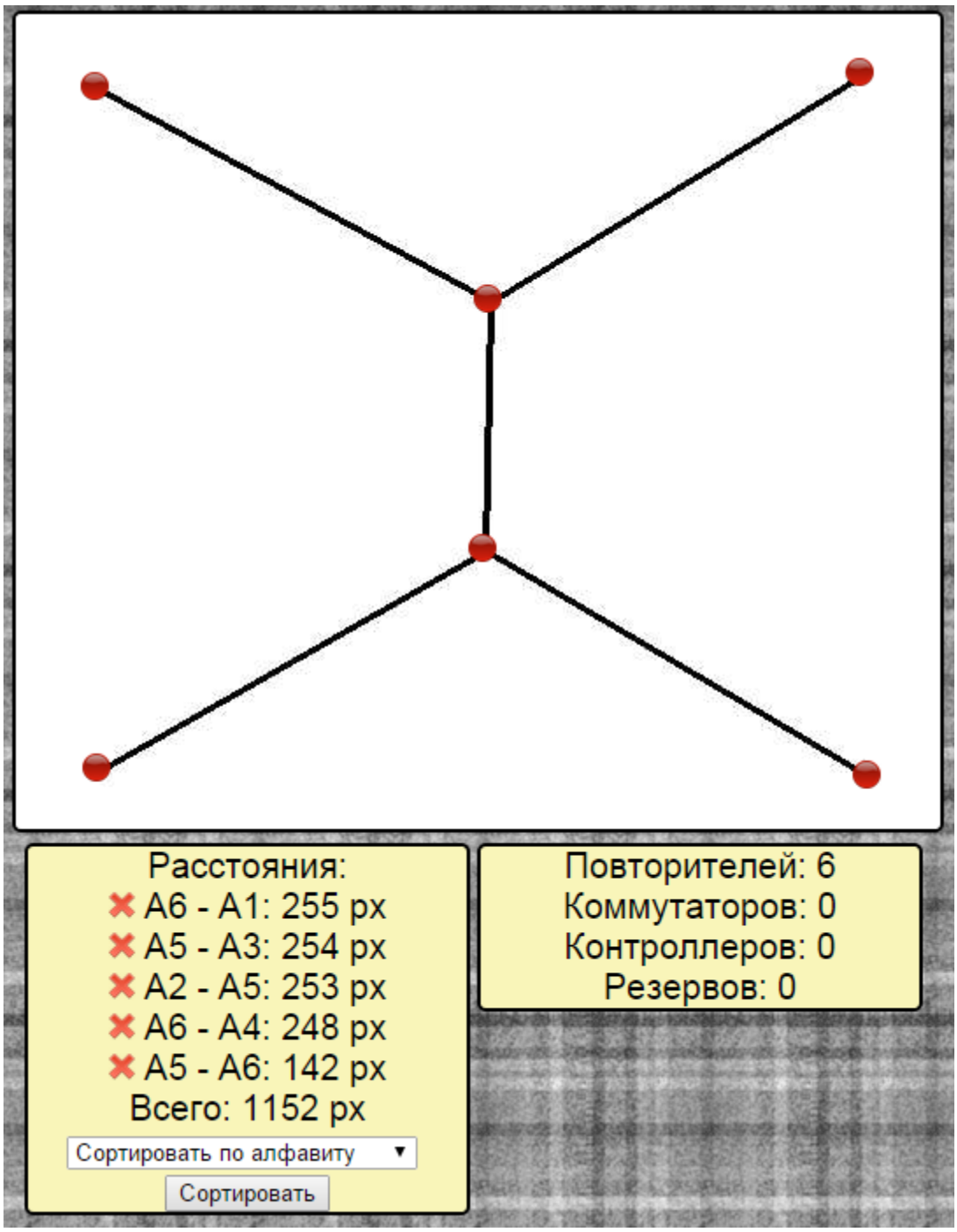


Рисунок 5.9 – З’єднання чотирьох точок мережі з використанням дерева Штейнера, що розташоване вертикально

На рисунку 5.9 чотири точки мережі з’єднані з використанням дерева Штейнера, що розташоване вертикально, сумарна довжина − 1152 пікселя.

5.2.3 Чотири точки, які розташовані несиметрично

На цьому етапі виконаємо з'єднання різними способами з використанням розробленого ПО чотирьох точок мережі, що розташовані несиметрично, і підрахуємо плановану загальну довжину сегментів мережі для наступних сполук (рисункм 5.10-5.12):

- послідовного;
- з довільною додатковою точкою, що розташована в центрі;
- з використанням дерева Штейнера.

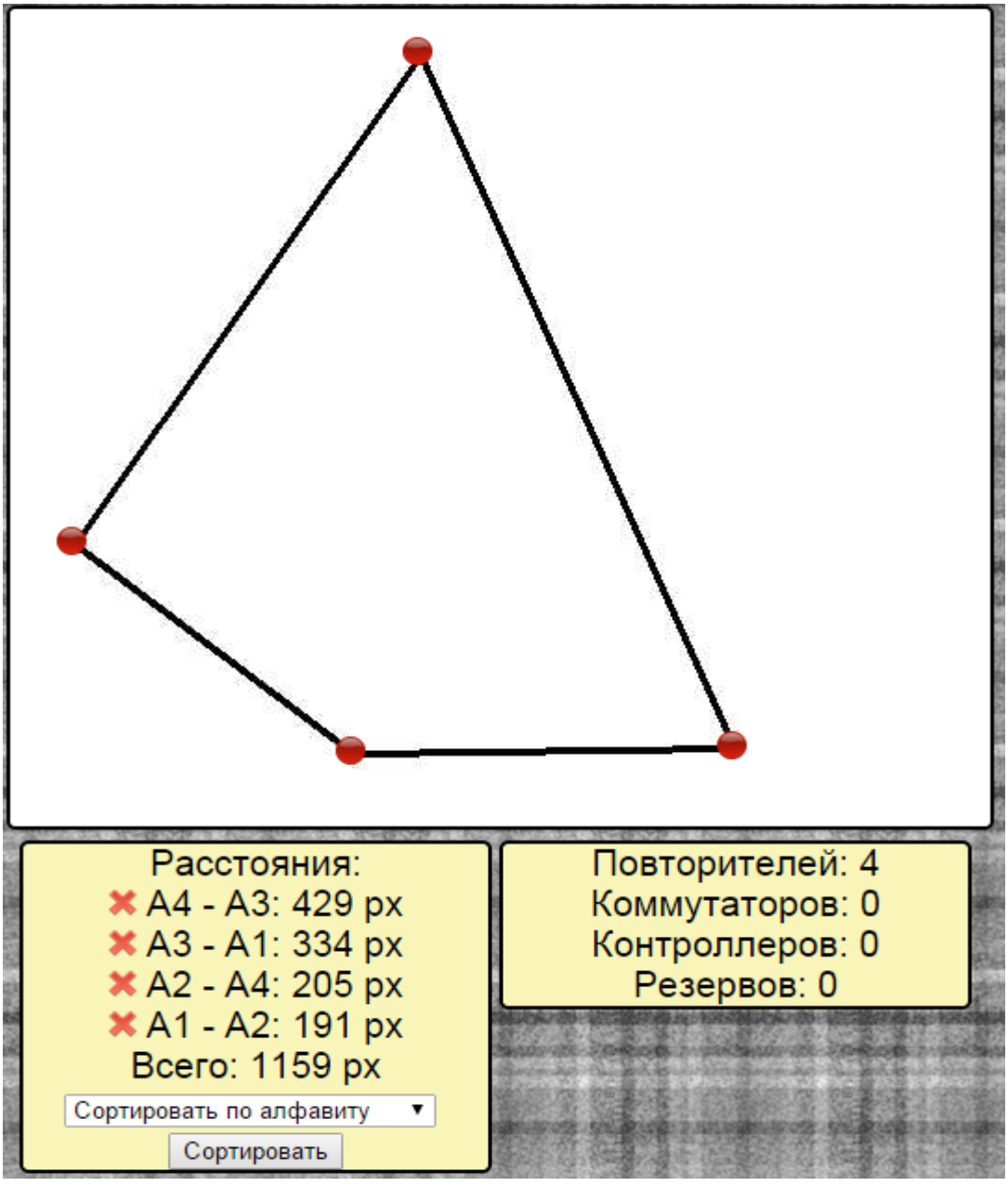


Рисунок 5.10 – Послідовно з'єднання чотирьох точок мережі, що розташовані несиметрично

На рисунку 5.10 чотири точки мережі, що розташовані несиметрично, з'єднані послідовно у вигляді чотирьохкутника, сумарна довжина − 1159 пікселей.

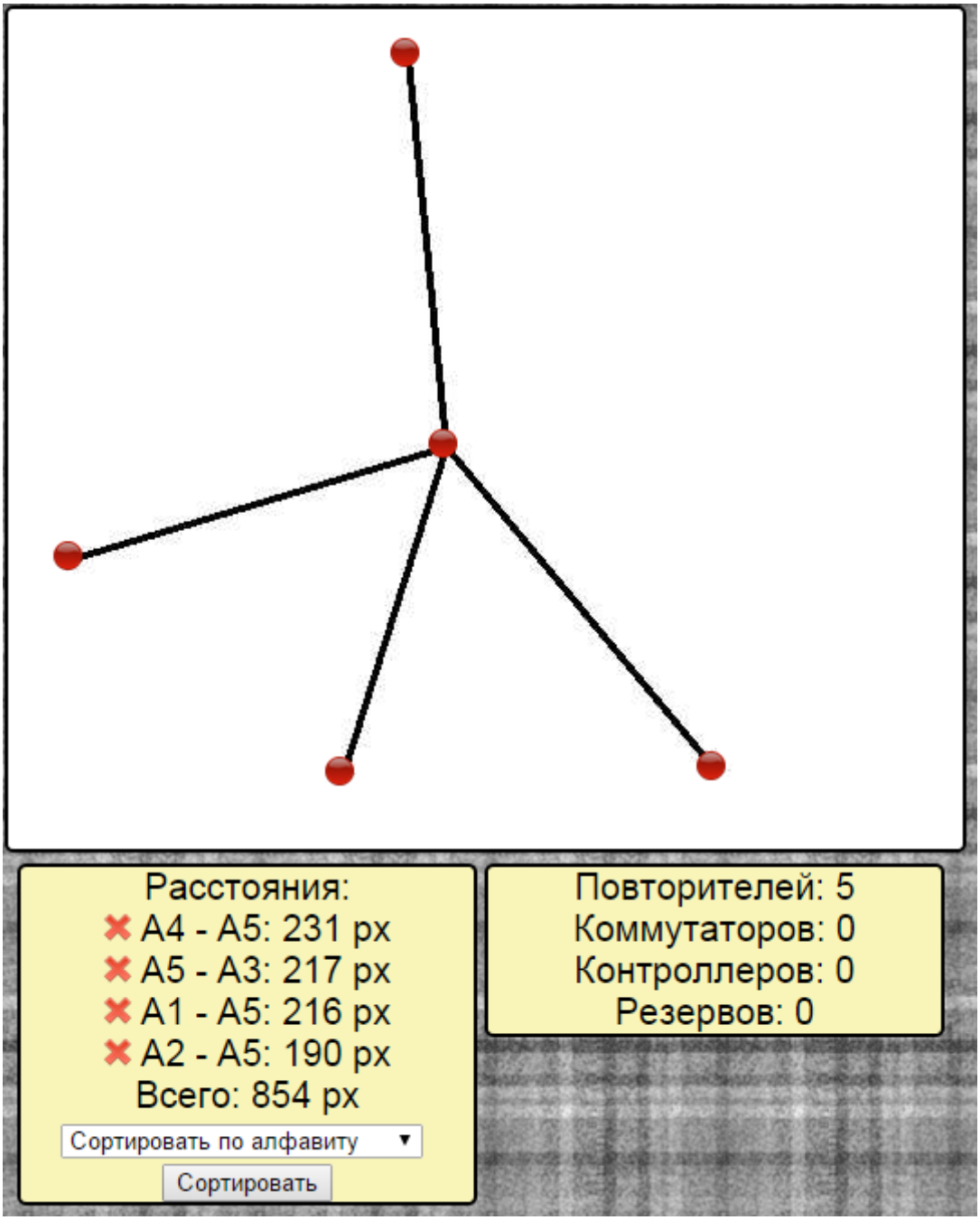


Рисунок 5.11 – Центральна точка мережі розташована довільним чином і з’єднана з чотирьма несисметричними іншими

На рисунку 5.11 центральна точка, що розташована довільним чином, з’єднана з чотирьма несисметричними іншими, сумарна довжина − 854 пікселей.

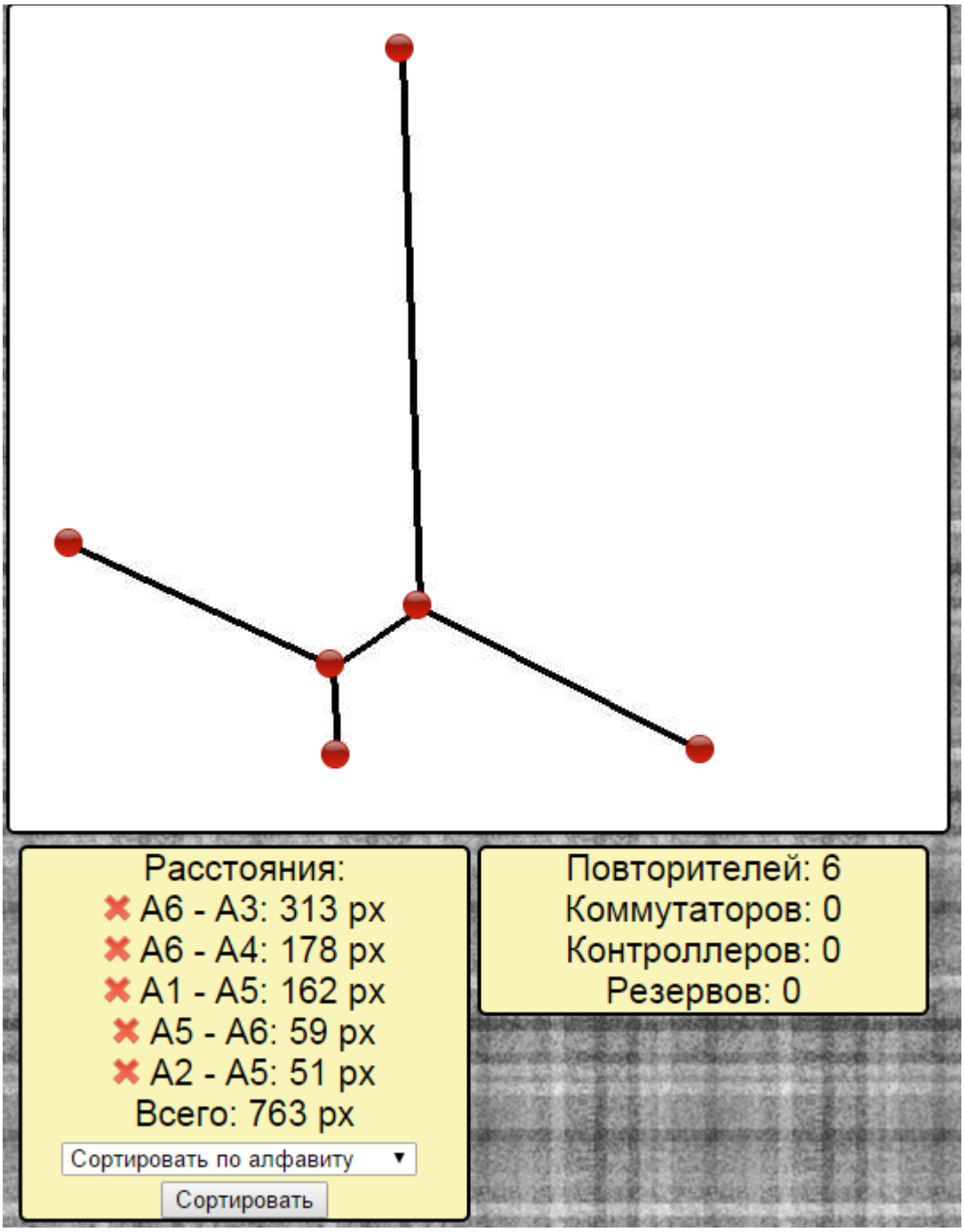


Рисунок 5.12 – З’єднання чотирьох точок мережі, що розташовані несиметрично, з використанням дерева Штейнера

На рисунку 5.12 показано з’єднання чотирьох точок мережі, що розташовані несиметрично, з використанням дерева Штейнера, сумарна довжина − 763 пікселя.

5.3 Оцінка результатів

В таблице таблица 5.1 наведено порівняння результатів експериментів, оранжевим кольором показані більш невдалі варианти (с найбільшою загальною довжиною сегментів мережі), а зеленим – кращі.

Таблиця 5.1 – Результати дослідження

Три точки		**%**	**Різниця, %**
Тип з’єднання:	**Довжина:**		
Послідовне	1201	100,00	0,00
З додатковою точкою	697	58,03	41,97
Дерево Штейнера	691	57,54	42,46

Чотири точки симетрично		**%**	**Різниця, %**
Тип з’єднання:			
Послідовне	1663	100,00	0
З додатковою точкою	1178	70,84	29,16
Дерево Штейнера	1120	67,35	32,65

Чотири точки несимметрично		**%**	**Різниця, %**
Тип з’єднання:			
Послідовне	1159	100,00	0
З додатковою точкою	854	73,68	26,32
Дерево Штейнера	763	65,83	34,17

Аналіз результатів досліджень показує, що найгіршим варіантом завжди є послідовне з'єднання, а найкращим − з'єднання у вигляді дерева Штейнера.

Ефективність і зменшення сумарної довжини ділянок відрізняються при різних початкових позиціях точок.

У першому експерименті с трьома точками при використанні дерева Штейнера вдалося досягти скорочення довжини майже в два рази (на 42,5%) у порівнянні з послідовним з'єднанням. У двох інших з чотирма точками − приблизно в півтора рази (на 32,7% і 34,2% відповідно).

З отриманих результатів видно, що використання додаткових точок, розташованих приблизно по центру між початковими теж помітно зменшує довжину мережі, але не так сильно, як при використанні дерев Штейнера.

Так для схеми з трьома точками − на 42,0%, для схеми з чотирма симетричними точками − на 29,2%, для схеми з чотирма несіммерічними точками − на 26,3%.

Останній висновок особливо важливий при розробці топології мереж для вугільних шахт тому, що при цьому діє велике обмеження за рахунок топології виробок на яку доводиться накладати топологію мережі.

ВИСНОВКИ

Атестаційна робота магістра є завершеною роботою в якій вирішена задача − обґрунтування параметрів комп'ютерної системи відеоконтролю з підсистемою моделювання тпологіі мережі.

Основні висновки і результати роботи полягають в наступному:

1. Проведен аналіз існуючого обладнання на вугільніх шахтах і використовуваних систем відеоконтролю стрічкових конвеєрів вугільніх шахт.
2. Вивчено особливості побудови CAN-мереж.
3. Досліджено методи контролю стрічкових конвеєрів, що показало можливість побудови комп’ютерної системи відеоконтролю технологічного процесу на основі локальної CAN-мережі.
4. Обґрунтовано параметри комп’ютерної системи відеоконтролю стрічкових конвеєрів вугільної шахти з опрацюванням підсистеми проектування топології мережі.
5. Розроблено програмне забезпечення підсистеми проектування топології мережі.

СПИСОК ПОСИЛАНЬ

1. Ткачев В.В., Поперечный Д.А., Надточий В.В., Гапон В.В., Цвиркун Л.И., Грулер Г. Исследование возможности применения полевой шины CAN протокола CANopen для создания систем передачи информации в шахтных условиях. – Сборник научных трудов НГУ. Выпуск №19, Том 2 – Днепропетровск, 2004. – с.50–59.
2. Вишневский В. М. Теоретические основы проектирования компьютерных сетей. – М.: Техносфера, 2003. – 512 с.: ил;
3. Овчинников В.Н. Организация передачи информации в автоматизированных системах управления. – М.: Энергия, 1974. − 128 с.: ил;
4. Левитин А. В. Алгоритмы: введение в разработку и анализ. — М.: «Вильямс», 2006. — с. 349 — 353;
5. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы: построение и анализ — 2-е изд. — М.: Вильямс, 2006. — с. 1296;
6. Bellman R. On a Routing Problem // Quarterly of Applied Mathematics. 1958. Vol 16, No. 1. C. 87-90, 1958;
7. Shimbel A. Structural Parameters of Communication networks, 1953, v.15, № 4;
8. Otterman J. Matrix Multiplication in Search for Alternate Routes, 1963, v.38, № 2;
9. Кузнецов Н. А., Фетисов В. Н. «Алгоритм Дейкстры с улучшенной робастностью для управления маршрутизацией в IP-сетях», Автоматика и телемеханика, № 2, 2008;
10. Дэвис Д., Барбер Д. Сети связи для вычислительных машин. – М.: Мир, 1976. – 680 с.
11. Нечипоренко В. И. Структурный анализ систем. – М.: Сов. радио, 1977. – 21

12. E. N. Gilbert and H. O. Pollak. Steiner Minimal Trees. In: SIAM Journal on Applied Mathematics, 1968, v. 16, No 1, pp. 1–29.
13. Z. A. Melzak. Companion to Concrete Mathematics. John Wiley & Sons, Inc., 1973.
14. Pawel Winter. An Algorithm for the Steiner Problem in the Euclidean Plane. In: Networks, 1985, v. 15, No 3, pp. 323–345.
15. Pawel Winter. Steiner Problem in Networks: A Survey. In: Networks, 1987, v. 17, No 2, pp. 129–167.
16. М. Гэри, Д. Джонсон. Вычислительные машины и труднорешаемые задачи. — Перев. с англ. М.: Мир, 1982.
17. 17. Gruhler G., Pivnjak G., Tkachov V., Tsvirkun L., Poperechnyy D. Very large hierarchical CANopen systems in mining // CAN Newsletter. – 2004. – № 4. – с. 48–54.
18. Цвиркун Л.И., Липовой Р.В. Топологическая оптимизация CAN-сети системы контроля конвейерных линий угольной шахты. – Збірник наукових праць НГУ. № 32 – Дніпропетровськ: РВК НГУ, 2009. – с. 141-146.
19. Цвіркун Л. І. Інформаційні технології при топологічній оптимізації моделі розподіленої системи передачі інформації / Системні технології. Регіональний міжвузівський збірник наукових праць. – Випуск 2 (67). – Дніпропетровськ, 2010 – С. 191-197.

Додаток А

Текст программи підсистеми проектування топології мережі

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПІДСИСТЕМИ ПРОЕКТУВАННЯ ТОПОЛОГІЇ МЕРЕЖІ

Текст программы

804.02070743.20002-01 12 01

Листів 35

2020

АНОТАЦІЯ

Даний документ містить вихідний код програми для проектування топології мережі.

Текст програми реалізований на мові PHP 5.6.

Середовище розробки та налагодження − PHPStorm 7.1.2.

ЗМІСТ

3

main_coords.php

```
<!DOCTYPE html>
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf8"/>
   <title>Coordinates</title>
   <link rel="stylesheet" type="text/css" href="style.css">
   <link rel="icon"
        type="image/png"
        href="pickaxe.png">
</head>
<body>
<?php
if (!empty($_FILES['file'])) {

   if ($_FILES['file']['error'] == 0) {
      $allowed_extensions = array('jpg');
      // check extension
      $file = explode(".", $_FILES['file']['name']);
      $extension = array_pop($file);
      if (in_array($extension, $allowed_extensions)) {

         $uploaddir = 'img/'; //директория файла, где img - папка
         $uploadfile = $uploaddir . "1.jpg";
         @move_uploaded_file($_FILES['file']['tmp_name'], $uploadfile); // переміщуємо з
тимчасового сховища за адресою з змінної $uploadfile
         include("resize.php");
         img_resize($uploadfile, "img/resized.jpg", 1500, 0);
      } else {
//          echo "Только расширение .jpg разрешено";
      }
   } else {
//       echo "Загрузка неудачна";
   }
}
//-----------------------------------MYSQL-----------------

$host = "mysql.hostinger.com.ua";
$username = "u410442243_coord";

$password = "123456";
$database_name = "u410442243_coord";

$connection = mysql_connect($host, $username, $password);
mysql_select_db($database_name); //вибір бази даних

//отримання координат клика на карту
$foo_x = $_POST['foo_x'];
$foo_y = $_POST['foo_y'];
//активний tool
@$tool_name = file_get_contents("tool_name.txt");
```

4

```
//-------------------------------------ТУЛБАР----------------------
?>

<div class="toolbar">

   <form action=" method=post>
      <input
         <?if ($tool_name == "repeater")
            echo "style='background-color: green;"'?>
         type="image" src="repeater.png" name="repeater" title="Повторитель"/>
      <input
         <?if ($tool_name == "switch")
            echo "style='background-color: green;"'?>
         type="image" src="switch.png" name="switch" title="Коммутатор"/>
      <input
         <?if ($tool_name == "controller")
            echo "style='background-color: green;"'?>
         type="image" src="controller.png" name="controller" title="Контроллер"/>
      <input
         <?if ($tool_name == "reserve")
            echo       "style='background-color:       green;"'?>type="image"       src="reserve.png"
name="reserve" title="Резерв"/>
      <input
               <input
         <?if ($tool_name == "turn")
            echo "style='background-color: green;"'?>
         type="image" src="turn.png" name="turn" title="Соединить с поворотами"/>
      <input
         <?if ($tool_name == "line")
            echo  "style='background-color:  green;"'?>type="image"  src="line.png"  name="line"
title="Соединить"/>
      <input type="image" src="info.png" name="info" title="Показать расстояния"/>
      <input type="image" src="reset.png" name="reset" title="Очистить"/>
      <input type="image" src="add_new.png" name="add_new" title="Загрузить"/>
      <input type="image" src="save.png" name="save" title="Сохранить"/>
      <input
         <?if ($tool_name == "ruler")
            echo "style='background-color: green;"'?>
         type="image" src="ruler.png" name="ruler" title="Ввести масштаб"/>
   </form>
</div>
<!--------------------------------------КАРТА--------------------------->
<div class="map_wrapper"
    style="<? $img = imagecreatefromjpeg("img/resized.jpg"); ?>width: <? echo imagesx($img);
?>px; height: <? echo imagesy($img); ?>px;<? imagedestroy($img); ?>">
<div class="map">
   <form action=" method=post>
      <input type="image" alt=' Finding coordinates of an image' src="points.php" name="foo"/>
```

```
<!--            <input type="image" alt=' Finding coordinates of an image' src="image001.jpg" name="foo"/>-->
   </form>

</div>
<!--</div>-->


<?
//---------------------------- Якщо натиснута кнопка тулбара ----------------------
$repeater = $_POST['repeater_x'];
$switch = $_POST['switch_x'];
$controller = $_POST['controller_x'];
$reserve = $_POST['reserve_x'];
$delete = $_POST['delete_x'];
$reset = $_POST['reset_x'];
$line = $_POST['line_x'];
$info = $_POST['info_x'];
$add_new = $_POST['add_new_x'];
$save = $_POST['save_x'];
$ruler = $_POST['ruler_x'];
$turn = $_POST['turn_x'];
if (isset($repeater)) {
   $tool_name = "repeater";
   file_put_contents('tool_name.txt', "$tool_name");
   $table_name = "A";
   file_put_contents('tool_table.txt', "$table_name");
   header("Location: " . $_SERVER["REQUEST_URI"] . "");
} elseif (isset($switch)) {
   $tool_name = "switch";
   file_put_contents('tool_name.txt', "$tool_name");
   $table_name = "B";
   file_put_contents('tool_table.txt', "$table_name");
   header("Location: " . $_SERVER["REQUEST_URI"] . "");
} elseif (isset($controller)) {
   $tool_name = "controller";
   file_put_contents('tool_name.txt', "$tool_name");
   $table_name = "C";
   file_put_contents('tool_table.txt', "$table_name");
   header("Location: " . $_SERVER["REQUEST_URI"] . "");
} elseif (isset($reserve)) {
   $tool_name = "reserve";
   file_put_contents('tool_name.txt', "$tool_name");
   $table_name = "D";
   file_put_contents('tool_table.txt', "$table_name");
   header("Location: " . $_SERVER["REQUEST_URI"] . "");
} elseif (isset($turn)) {
   $tool_name = "turn";
   } elseif (isset($info)) {
   @$info_state = file_get_contents("info_state.txt");
```

```
    if ($info_state == "on")
        $info_state = "off";
    else
        $info_state = "on";
    file_put_contents('info_state.txt', "$info_state");
    header("Location: " . $_SERVER["REQUEST_URI"] . "");
} elseif (isset($add_new)) {

    header('Location: http://borschik.16mb.com/coords_newest/coords/upload.php');

} elseif (isset($save)) {
    header('Location: http://borschik.16mb.com/coords_newest/coords/save_links.html');

} elseif (isset($ruler)) {

    header('Location: http://borschik.16mb.com/coords_newest/coords/ruler_start.php');
}
@$table_name = file_get_contents("tool_table.txt");


//----------------------------------------Якщо є клік по карті----------------------------
@$tool_name = file_get_contents("tool_name.txt");
if (($tool_name != "delete") && ($tool_name != "line") && ($tool_name != "turn")) {

    $submit = $_POST['foo_x'];
    if (isset($submit)) {

        //<!------------ ДОДАТОК ТОЧКИ----------->

        $query = "SELECT `Name` FROM `$table_name`";
        $result = mysql_query($query);
        while ($row = mysql_fetch_assoc($result)) {
            $input = $row['Name'];
        }

        if ($input != '') {
            $number = substr($input, 1) + 1; // видалити першу букву.
        } else {
            $number = 1;
        }
        $name = $table_name . $number;


        $query = "INSERT INTO $table_name (Name, X, Y) VALUES
('$name','$foo_x','$foo_y')";
        $result = mysql_query($query) or die(mysql_error());

        header("Location: " . $_SERVER["REQUEST_URI"] . "");

    }
}
```

```
if (isset($delete)) {
   $tool_name = "delete";
   file_put_contents('tool_name.txt', "$tool_name");
   header("Location: " . $_SERVER["REQUEST_URI"] . "");
}

@$tool_name = file_get_contents("tool_name.txt");
if ($tool_name == "delete") {

   $i = 0;
   while ($i < 20) {
      $icon_button_name = $A_Name[$i] . "_x";
      $icon_button = $_POST[$icon_button_name];
      if (isset($icon_button)) {
         $query = "DELETE FROM `A` WHERE `A`.`Name` = '$A_Name[$i]'";
         $result = mysql_query($query);
         $query = "DELETE FROM `Lines` WHERE `name_start` = '$A_Name[$i]'";
         $result = mysql_query($query);
         $query = "DELETE FROM `Lines` WHERE `name_end` = '$A_Name[$i]'";
         $result = mysql_query($query);
         $query = "DELETE FROM `Lines_with_turns` WHERE `Start` = '$A_Name[$i]'";
         $result = mysql_query($query);
         $query = "DELETE FROM `Lines_with_turns` WHERE `End` = '$A_Name[$i]'";
         $result = mysql_query($query);
         header("Location: " . $_SERVER["REQUEST_URI"] . ""); //оновити сторінку
      }
      $i++;
   }

   $i = 0;
   while ($i < 20) {
      $icon_button_name = $B_Name[$i] . "_x";
      $icon_button = $_POST[$icon_button_name];
      if (isset($icon_button)) {
         $query = "DELETE FROM `B` WHERE `B`.`Name` = '$B_Name[$i]'";
         $result = mysql_query($query);
         $query = "DELETE FROM `Lines` WHERE `name_start` = '$B_Name[$i]'";
         $result = mysql_query($query);
         $query = "DELETE FROM `Lines` WHERE `name_end` = '$B_Name[$i]'";
         $result = mysql_query($query);
         $query = "DELETE FROM `Lines_with_turns` WHERE `Start` = '$B_Name[$i]'";
         $result = mysql_query($query);
         $query = "DELETE FROM `Lines_with_turns` WHERE `End` = '$B_Name[$i]'";
         $result = mysql_query($query);
         header("Location: " . $_SERVER["REQUEST_URI"] . ""); //оновити сторінку
      }
      $i++;
   }

   $i = 0;
   while ($i < 20) {
```

```
        if (isset($icon_button)) {
            $query = "DELETE FROM `C` WHERE `C`.`Name` = '$C_Name[$i]'";
            $result = mysql_query($query);
            $query = "DELETE FROM `Lines` WHERE `name_start` = '$C_Name[$i]'";
            $result = mysql_query($query);
            $query = "DELETE FROM `Lines` WHERE `name_end` = '$C_Name[$i]'";
            $result = mysql_query($query);
            $query = "DELETE FROM `Lines_with_turns` WHERE `Start` = '$C_Name[$i]'";
            $result = mysql_query($query);
            $query = "DELETE FROM `Lines_with_turns` WHERE `End` = '$C_Name[$i]'";
            $result = mysql_query($query);
            header("Location: " . $_SERVER["REQUEST_URI"] . ""); //оновити сторінку
        }
        $i++;
    }

    $i = 0;
    while ($i < 20) {
        $icon_button_name = $D_Name[$i] . "_x";
        $icon_button = $_POST[$icon_button_name];
        if (isset($icon_button)) {
            $query = "DELETE FROM `D` WHERE `D`.`Name` = '$D_Name[$i]'";
            $result = mysql_query($query);
            $query = "DELETE FROM `Lines` WHERE `name_start` = '$D_Name[$i]'";
            $result = mysql_query($query);
            $query = "DELETE FROM `Lines` WHERE `name_end` = '$D_Name[$i]'";
            $result = mysql_query($query);
            $query = "DELETE FROM `Lines_with_turns` WHERE `Start` = '$D_Name[$i]'";
            $result = mysql_query($query);
            $query = "DELETE FROM `Lines_with_turns` WHERE `End` = '$D_Name[$i]'";
            $result = mysql_query($query);
            header("Location: " . $_SERVER["REQUEST_URI"] . ""); //оновити сторінку
        }
        $i++;
    }

}
?>
</div>
<?
//-------------------------------------ЯКЩО НАТИСНУТО  СКИНУТИ-------------------------
if (isset($reset)) {

    @unlink('ruler_distance.txt');
    $query = "DELETE FROM A";
    $result = mysql_query($query) or die(mysql_error());
    $query = "DELETE FROM B";
    $result = mysql_query($query) or die(mysql_error());
    $query = "DELETE FROM C";
    $result = mysql_query($query) or die(mysql_error());
        $query = "DELETE FROM E";
```

```
    $result = mysql_query($query) or die(mysql_error());
    $query = "DELETE FROM `Lines`";
    $result = mysql_query($query) or die(mysql_error());
    $query = "DELETE FROM `Lines_with_turns`";
    $result = mysql_query($query) or die(mysql_error());
    $query = "ALTER TABLE `Lines` AUTO_INCREMENT=0";
    $result = mysql_query($query) or die(mysql_error());
    $query = "DELETE FROM `Ruler`";
    $result = mysql_query($query) or die(mysql_error());
    header("Location: " . $_SERVER["REQUEST_URI"] . "");
}
?>

<?
//---------------- За натиснутої кнопки З'ЄДНАННЯ ЛІНІЙ ----------
if (isset($line)) {
    $tool_name = "line";
    file_put_contents('tool_name.txt', "$tool_name");
    header("Location: " . $_SERVER["REQUEST_URI"] . "");
}

@$tool_name = file_get_contents("tool_name.txt");
if ($tool_name == "line") {

    $query = "SELECT `name_start` FROM `Lines`";
    $result = mysql_query($query) or die(mysql_error());

    while ($row = mysql_fetch_assoc($result)) {
        if ($row['name_start'] != '')
            $name_start[] = $row['name_start'];
    }
    $query = "SELECT `name_end` FROM `Lines`";
    $result = mysql_query($query) or die(mysql_error());
    while ($row = mysql_fetch_assoc($result)) {
        if ($row['name_end'] != '')
            $name_end[] = $row['name_end'];
    }
    $i = 0;
    while ($i < 20) {
        $icon_button_name = $A_Name[$i] . "_x";
        $icon_button = $_POST[$icon_button_name];

        if (isset($icon_button)) {
            if ((count($name_start)) > (count($name_end))) {
                $query = "UPDATE `Lines` SET `name_end` = '$A_Name[$i]' WHERE `name_end`
= '' ";
                $result = mysql_query($query) or die(mysql_error());
            }
            }
            header("Location: " . $_SERVER["REQUEST_URI"] . ""); //оновити сторінку
        }
```

```
        $i++;
    }
    $i = 0;
    while ($i < 20) {
        $icon_button_name = $B_Name[$i] . "_x";
        $icon_button = $_POST[$icon_button_name];

        if (isset($icon_button)) {
            if ((count($name_start)) > (count($name_end))) {
                $query = "UPDATE `Lines` SET `name_end` = '$B_Name[$i]' WHERE `name_end` =
" ";
                $result = mysql_query($query) or die(mysql_error());
            } else {
                $query = "INSERT INTO `Lines` (`name_start`) VALUES ('$B_Name[$i]')";
                $result = mysql_query($query) or die(mysql_error());
            }
            header("Location: " . $_SERVER["REQUEST_URI"] . ""); //оновити сторінку
        }
        $i++;
    }
    $i = 0;
    while ($i < 20) {
        $icon_button_name = $C_Name[$i] . "_x";
        $icon_button = $_POST[$icon_button_name];

        if (isset($icon_button)) {
            if ((count($name_start)) > (count($name_end))) {
                $query = "UPDATE `Lines` SET `name_end` = '$C_Name[$i]' WHERE `name_end` =
" ";
                $result = mysql_query($query) or die(mysql_error());
            } else {
                $query = "INSERT INTO `Lines` (`name_start`) VALUES ('$C_Name[$i]')";
                $result = mysql_query($query) or die(mysql_error());
            }

            header("Location: " . $_SERVER["REQUEST_URI"] . ""); // оновити сторінку
        }       $i++;
    }
    $i = 0;
    while ($i < 20) {
        $icon_button_name = $D_Name[$i] . "_x";
        $icon_button = $_POST[$icon_button_name];

        if (isset($icon_button)) {
            if ((count($name_start)) > (count($name_end))) {
                $query = "UPDATE `Lines` SET `name_end` = '$D_Name[$i]' WHERE `name_end`
= " ";
` = " LIMIT 1 ";
                //$query = "INSERT INTO `Lines` (`name_end`) VALUES ('$D_Name[$i]')";
            } else {
                $query = "INSERT INTO `Lines` (`name_start`) VALUES ('$D_Name[$i]')";
```

11

```
            $result = mysql_query($query) or die(mysql_error());
        }


        header("Location: " . $_SERVER["REQUEST_URI"] . ""); // оновити сторінку
    }
    $i++;
  }

}
//-----------------РЕЖИМ ПОВОРОТА КАБЕЛЕЙ-------------------------------------
@$tool_name = file_get_contents("tool_name.txt");
if ($tool_name == "turn") {
  $query = "SELECT `Start` FROM `Lines_with_turns`";
  $result = mysql_query($query) or die(mysql_error());
  while ($row = mysql_fetch_assoc($result)) {
    if ($row['Start'] != '')
      $turn_start[] = $row['Start'];
  }
  $query = "SELECT `End` FROM `Lines_with_turns`";
  $result = mysql_query($query) or die(mysql_error());
  while ($row = mysql_fetch_assoc($result)) {
    if ($row['End'] != '')
      $turn_end[] = $row['End'];
  }
  //--------- ЯКЩО В РЕЖИМІ ПОВОРОТУ клікнути на ТОЧКУ ---------------------------------
  $i = 0;
  while ($i < 20) {
    $icon_button_name = $A_Name[$i] . "_x";
    $icon_button = $_POST[$icon_button_name];

    if (isset($icon_button)) {
      if ((count($turn_start)) > (count($turn_end))) {
        $query = "UPDATE `Lines` SET `name_end` = '$A_Name[$i]', `has_turns` = 'Yes'
WHERE `name_end` = '' ";
        $result = mysql_query($query) or die(mysql_error());
        $query = "UPDATE `Lines_with_turns` SET `End` = '$A_Name[$i]' WHERE `End` =
'' ";
        $result = mysql_query($query) or die(mysql_error());


      } else {
        $query = "INSERT INTO `Lines` (`name_start`,`has_turns`) VALUES
('$A_Name[$i]','Yes')";
        $result = mysql_query($query) or die(mysql_error());
      }

      header("Location: " . $_SERVER["REQUEST_URI"] . ""); //оновити сторінку
    }
    $i++;
```

12

```
    }
    $i = 0;
    while ($i < 20) {
        $icon_button_name = $B_Name[$i] . "_x";
        $icon_button = $_POST[$icon_button_name];
        if (isset($icon_button)) {
            if ((count($turn_start)) > (count($turn_end))) {
                $query = "UPDATE `Lines` SET `name_end` = '$B_Name[$i]', `has_turns` = 'Yes'
WHERE `name_end` = " ";
                $result = mysql_query($query) or die(mysql_error());
                $query = "UPDATE `Lines_with_turns` SET `End` = '$B_Name[$i]' WHERE `End` =
" ";
                $result = mysql_query($query) or die(mysql_error());
            } else {
                $query = "INSERT INTO `Lines` (`name_start`,`has_turns`) VALUES
('$B_Name[$i]','Yes')";
                $result = mysql_query($query) or die(mysql_error());
                $query = "INSERT INTO `Lines_with_turns` (`Start`) VALUES ('$B_Name[$i]')";
                $result = mysql_query($query) or die(mysql_error());
            }
            header("Location: " . $_SERVER["REQUEST_URI"] . ""); //оновити сторінку
        }
        $i++;
    }
    $i = 0;
    while ($i < 20) {
        $icon_button_name = $C_Name[$i] . "_x";
        $icon_button = $_POST[$icon_button_name];

        if (isset($icon_button)) {
            if ((count($turn_start)) > (count($turn_end))) {
                $query = "UPDATE `Lines` SET `name_end` = '$C_Name[$i]', `has_turns` = 'Yes'
WHERE `name_end` = " ";
                $result = mysql_query($query) or die(mysql_error());
                $query = "UPDATE `Lines_with_turns` SET `End` = '$C_Name[$i]' WHERE `End` =
" ";
                $result = mysql_query($query) or die(mysql_error());
            } else {
                $query = "INSERT INTO `Lines` (`name_start`,`has_turns`) VALUES
('$C_Name[$i]','Yes')";
                $result = mysql_query($query) or die(mysql_error());
                $query = "INSERT INTO `Lines_with_turns` (`Start`) VALUES ('$C_Name[$i]')";
                $result = mysql_query($query) or die(mysql_error());
            }
            header("Location: " . $_SERVER["REQUEST_URI"] . ""); //оновити сторінку
        }
        $i++;
    }

    $i = 0;
    while ($i < 20) {
```

```
    $icon_button = $_POST[$icon_button_name];
    if (isset($icon_button)) {
      if ((count($turn_start)) > (count($turn_end))) {
        $query = "UPDATE `Lines` SET `name_end` = '$D_Name[$i]', `has_turns` = 'Yes' WHERE `name_end` = '' ";
        $result = mysql_query($query) or die(mysql_error());
        $query = "UPDATE `Lines_with_turns` SET `End` = '$D_Name[$i]' WHERE `End` = '' ";
        $result = mysql_query($query) or die(mysql_error());
      } else {
        $query = "INSERT INTO `Lines` (`name_start`,`has_turns`) VALUES ('$D_Name[$i]','Yes')";
        $result = mysql_query($query) or die(mysql_error());
        $query = "INSERT INTO `Lines_with_turns` (`Start`) VALUES ('$D_Name[$i]')";
        $result = mysql_query($query) or die(mysql_error());
      }
      header("Location: " . $_SERVER["REQUEST_URI"] . ""); //оновити сторінку
    }
    $i++;
  }

  //------ЯКЩО НА КРОЦІ ПОВОРОТУ КЛІКНУТЬ НА КАРТУ------------------------------

      if (isset($foo_x)) {

    //------------- ДОДАВАННЯ ТОЧКИ--------------

    $query = "SELECT `Name` FROM `E`";
    $result = mysql_query($query);
    while ($row = mysql_fetch_assoc($result)) {
      $input = $row['Name'];
    }

    if ($input != '') {
      $number = substr($input, 1) + 1; // видалити першу букву.
    } else {
      $number = 1;
    }
    $name = "E" . $number;

    $query = "INSERT INTO E (Name, X, Y) VALUES ('$name','$foo_x','$foo_y')";
    $result = mysql_query($query) or die(mysql_error());
    $query = "SELECT * FROM `Lines_with_turns`";
    $result = mysql_query($query);
    while ($row = mysql_fetch_row($result)) {
      $lines_with_turns_table_list[] = $row;
    }
    $j = 0;
    while ($j < mysql_num_rows($result)) {
      $i = 1;
      while ($i < 6) {
```

```
== ")){
                $turn = "Turn" . $i;
                $query = "UPDATE `Lines_with_turns` SET `$turn` = '$name' WHERE `End`="";
                echo $query;
                echo $i;
                $result = mysql_query($query) or die(mysql_error());;
                break;
            }
            $i++;
        }
        $j++;
    }
    header("Location: " . $_SERVER["REQUEST_URI"] . "");
  }


}

@$info_state = file_get_contents("info_state.txt");
if ($info_state == "on") {


  if (file_exists("ruler_distance.txt")) {
    @$distance_ruler_user = file_get_contents("ruler_distance.txt");
    $units = "метров";

    $query = "SELECT `X` FROM `Ruler` WHERE `Name` = 'start'";
    $result = mysql_query($query) or die(mysql_error());
    while ($row = mysql_fetch_assoc($result)) {
      $ruler_start_X = $row['X'];
    }
    $query = "SELECT `Y` FROM `Ruler` WHERE `Name` = 'start'";
    $result = mysql_query($query) or die(mysql_error());
    while ($row = mysql_fetch_assoc($result)) {
      $ruler_start_Y = $row['Y'];
    }
    $query = "SELECT `X` FROM `Ruler` WHERE `Name` = 'end'";
    $result = mysql_query($query) or die(mysql_error());
    while ($row = mysql_fetch_assoc($result)) {
      $ruler_end_X = $row['X'];
    }
    $query = "SELECT `Y` FROM `Ruler` WHERE `Name` = 'end'";
    $result = mysql_query($query) or die(mysql_error());
    while ($row = mysql_fetch_assoc($result)) {
      $ruler_end_Y = $row['Y'];
    }
    $distance_ruler_X = abs($ruler_start_X - $ruler_end_X);
    $distance_ruler_Y = abs($ruler_start_Y - $ruler_end_Y);
    $distance_ruler_pixels = round(sqrt(pow($distance_ruler_X, 2) + pow($distance_ruler_Y,
2)));
    $multiplier = $distance_ruler_user / $distance_ruler_pixels;
```

```
} else {
   $multiplier = 1;
   $units = "px";
}
?>

</div>
<div class=info_wrapper>
<div class=info>
   <?
   echo "Расстояния:"; ?><br><?




   //------------------------------ПОЧАТОК СОРТУВАННЯ----------------------------
   if ($_POST['menu'] == "sort_abc") {
      $sort_mode = "sort_abc";
      file_put_contents("sort_mode.txt", "$sort_mode");

   } elseif ($_POST['menu'] == "increase") {
      $sort_mode = "increase";
      file_put_contents("sort_mode.txt", "$sort_mode");

   } elseif ($_POST['menu'] == "decrease") {
      $sort_mode = "decrease";
      file_put_contents("sort_mode.txt", "$sort_mode");
   }


   include 'distance_turns.php';
   //------------------------------КІНЕЦЬ СОРТУВАННЯ------------------------------
   $i = 0;
   while ($i < count($lines_with_turns_table_list_Y)) {
      if (($lines_with_turns_table_list_X[$i + 1] != 0) && ($lines_with_turns_table_list_X[$i]
!= 0)) {

         $distance_lines_with_turns_X = abs($lines_with_turns_table_list_X[$i] -
$lines_with_turns_table_list_X[$i + 1]);
         $distance_lines_with_turns_Y = abs($lines_with_turns_table_list_Y[$i] -
$lines_with_turns_table_list_Y[$i + 1]);


         $distance_lines_with_turns[] = round(sqrt(pow($distance_lines_with_turns_X, 2) +
pow($distance_lines_with_turns_Y, 2)));

      } elseif (($lines_with_turns_table_list_X[$i] == 0) &&
($lines_with_turns_table_list_X[$i - 1] != 0)) {
         $distance_lines_with_turns[] = '|';
      }
```

```
    }
    $i = 0;
    while ($i < count($distance_lines_with_turns)) {

        if ($distance_lines_with_turns[$i] != '|') {
            $a = $distance_lines_with_turns[$i] + $a;
        }
        else{
            $distance_final[]=$a * $multiplier;

            $a=0;
        }

        $i++;
    }
    $i=0;
    while ($i<count($turn_name_end)){
        $query = "UPDATE `Lines` SET `distance` = '$distance_final[$i]' WHERE name_start = '$turn_name_start[$i]' AND name_end= '$turn_name_end[$i]' AND `has_turns` = 'Yes'"; // запись в БД расстояния между точками
        $result = mysql_query($query) or die(mysql_error());
        $i++;
    }

    include 'distance_info.php';

    $i = 0;
    while ($i < count($name_end)) {
        $distance_X = abs($start_X[$i] - $end_X[$i]);
        $distance_Y = abs($start_Y[$i] - $end_Y[$i]);


        $distance[] = round(sqrt(pow($distance_X, 2) + pow($distance_Y, 2)) * $multiplier);

        $query = "UPDATE `Lines` SET `distance` = '$distance[$i]' WHERE name_start = '$name_start[$i]' AND name_end= '$name_end[$i]' AND `has_turns` = ""; // запись в БД расстояния между точками
        $result = mysql_query($query) or die(mysql_error());

        $i++;

    }
    $sort_content = file_get_contents("sort_mode.txt");
    if($sort_content == "sort_abc"){
        $query = "SELECT * FROM `Lines` ORDER BY `name_start` ASC ";
    }
    elseif($sort_content == "increase"){
        $query = "SELECT * FROM `Lines` ORDER BY `distance` ASC ";
    }
    elseif($sort_content == "decrease"){
```

17

```
        }
        else{
            $query = "SELECT * FROM `Lines` ORDER BY `id` ASC";
        }
        $result = mysql_query($query);
        while ($row = mysql_fetch_assoc($result)) {
            $distance_completed[]=$row['distance'];
        }

        $i = 0;
        while ($i < count($name_end)) {
            ?>
            <form method="POST">
                <span class="delete_line_image"><input type="image" src="delete_line.png" name="<?= $i ?>" title="Удалить <?echo $name_start[$i]?> - <?echo $name_end[$i]?>" value="Удалить"/></span>
            </form>
            <?
            $distance_sum = $distance_completed[$i] + $distance_sum;
            echo "$name_start[$i] - $name_end[$i]: $distance_completed[$i]" . " $units" ?><br> <?;

            if (isset($_POST[$i . '_x'])) {
                $query = "DELETE FROM `Lines` WHERE name_start = '$name_start[$i]' AND name_end= '$name_end[$i]'";
                $result = mysql_query($query) or die(mysql_error());
                $query = "DELETE FROM `Lines_with_turns` WHERE Start = '$name_start[$i]' AND End= '$name_end[$i]'";
                $result = mysql_query($query) or die(mysql_error());
                header("Location: " . $_SERVER["REQUEST_URI"] . ""); //оновити сторінку
            }

            $i++;
        }

        echo "Всего: $distance_sum" . " $units"; ?><br>
        <style>
            select {
                width: 200px; /* Ширина списка в пикселах */
            }
        </style>
        <form action="" method="post">
            <select name="menu">
                <option value="sort_abc">Сортировать по алфавиту</option>
                <option value="increase">Сортировать по возрастанию длины</option>
                <option value="decrease">Сортировать по убыванию длины</option>
            </select>
            </br><input type="submit" value="Сортировать"/>
        </form>
    </div>

    <div class=info> <?
```

18

```
//-------------------------------------КІЛЬКІСТЬ ВСЬОГО-------------------------
        $query = "SELECT COUNT( `Name` ) FROM `A`";
        $result = mysql_query($query) or die(mysql_error());

        $count_A = mysql_result($result, 0);

        echo "Повторителей: " . $count_A;

        $query = "SELECT COUNT( `Name` ) FROM `B`";
```

19

```
        $result = mysql_query($query) or die(mysql_error());

        $count_B = mysql_result($result, 0);

        echo "<br> Коммутаторов: " . $count_B;

        $query = "SELECT COUNT( `Name` ) FROM `C`";
        $result = mysql_query($query) or die(mysql_error());

        $count_C = mysql_result($result, 0);

        echo "<br> Контроллеров: " . $count_C;

        $query = "SELECT COUNT( `Name` ) FROM `D`";
        $result = mysql_query($query) or die(mysql_error());

        $count_D = mysql_result($result, 0);

        echo "<br> Резервов: " . $count_D;
        if (file_exists("ruler_distance.txt")) {
            echo "<br> Масштаб 1:" . round($multiplier, 2);
        }
        ?> </div> </div><?
}
?>
</body>
```

db.php

```
<?
$host = "mysql.hostinger.com.ua";
$username = "u410442243_coord";

$password = "123456";
$database_name = "u410442243_coord";

$connection = mysql_connect($host, $username, $password);
mysql_select_db($database_name); //вибір бази данихх
```

distance.php

```
<?php


include"db.php";

$query = "SELECT `name_start` FROM `Lines` WHERE `has_turns` = " ORDER BY `id` ASC";
$result = mysql_query($query) or die(mysql_error());

while($row = mysql_fetch_assoc($result)){
   if( $row['name_start']!='')
      $name_start[] = $row['name_start'];
}

$query = "SELECT `name_end` FROM `Lines` WHERE `has_turns` = " ORDER BY `id` ASC";
$result = mysql_query($query) or die(mysql_error());
while($row = mysql_fetch_assoc($result)){
   if( $row['name_end']!='')
      $name_end[] = $row['name_end'];
}

$i=0;
while($i<200){
   $query = "SELECT `X` FROM `A` WHERE `Name` = '$name_start[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $start_X[] = $row['X'];
   }
   $query = "SELECT `Y` FROM `A` WHERE `Name` = '$name_start[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $start_Y[] = $row['Y'];
   }

   $query = "SELECT `X` FROM `A` WHERE `Name` = '$name_end[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $end_X[] = $row['X'];
```

```
$query = "SELECT `Y` FROM `A` WHERE `Name` = '$name_end[$i]'";
$result = mysql_query($query) or die(mysql_error());
while($row = mysql_fetch_assoc($result)){
   $end_Y[] = $row['Y'];
}

$query = "SELECT `X` FROM `B` WHERE `Name` = '$name_start[$i]'";
$result = mysql_query($query) or die(mysql_error());
while($row = mysql_fetch_assoc($result)){
   $start_X[] = $row['X'];
}
$query = "SELECT `Y` FROM `B` WHERE `Name` = '$name_start[$i]'";
$result = mysql_query($query) or die(mysql_error());
while($row = mysql_fetch_assoc($result)){
   $start_Y[] = $row['Y'];
}

$query = "SELECT `X` FROM `B` WHERE `Name` = '$name_end[$i]'";
$result = mysql_query($query) or die(mysql_error());
while($row = mysql_fetch_assoc($result)){
   $end_X[] = $row['X'];
}
$query = "SELECT `Y` FROM `B` WHERE `Name` = '$name_end[$i]'";
$result = mysql_query($query) or die(mysql_error());
while($row = mysql_fetch_assoc($result)){
   $end_Y[] = $row['Y'];
}

$query = "SELECT `X` FROM `C` WHERE `Name` = '$name_start[$i]'";
$result = mysql_query($query) or die(mysql_error());
while($row = mysql_fetch_assoc($result)){
   $start_X[] = $row['X'];
}
$query = "SELECT `Y` FROM `C` WHERE `Name` = '$name_start[$i]'";
$result = mysql_query($query) or die(mysql_error());
while($row = mysql_fetch_assoc($result)){
   $start_Y[] = $row['Y'];
}
$query = "SELECT `X` FROM `C` WHERE `Name` = '$name_end[$i]'";
$result = mysql_query($query) or die(mysql_error());
while($row = mysql_fetch_assoc($result)){
   $end_X[] = $row['X'];
}
$query = "SELECT `Y` FROM `C` WHERE `Name` = '$name_end[$i]'";
$result = mysql_query($query) or die(mysql_error());
while($row = mysql_fetch_assoc($result)){
   $end_Y[] = $row['Y'];
}

$query = "SELECT `X` FROM `D` WHERE `Name` = '$name_start[$i]'";
$result = mysql_query($query) or die(mysql_error());
```

```
 while($row = mysql_fetch_assoc($result)){
        $start_X[] = $row['X'];
    }
    $query = "SELECT `Y` FROM `D` WHERE `Name` = '$name_start[$i]'";
    $result = mysql_query($query) or die(mysql_error());
    while($row = mysql_fetch_assoc($result)){
        $start_Y[] = $row['Y'];
    }


    $query = "SELECT `X` FROM `D` WHERE `Name` = '$name_end[$i]'";
    $result = mysql_query($query) or die(mysql_error());
    while($row = mysql_fetch_assoc($result)){
        $end_X[] = $row['X'];
    }
    $query = "SELECT `Y` FROM `D` WHERE `Name` = '$name_end[$i]'";
    $result = mysql_query($query) or die(mysql_error());
    while($row = mysql_fetch_assoc($result)){
        $end_Y[] = $row['Y'];
    }

    $i++;
}
```

distance_info.php

```
<?php
include"db.php";

$sort_content = file_get_contents("sort_mode.txt");
if($sort_content == "sort_abc"){
    $query = "SELECT * FROM `Lines` ORDER BY `name_start` ASC ";
}
elseif($sort_content == "increase"){
    $query = "SELECT * FROM `Lines` ORDER BY `distance` ASC ";
}
elseif($sort_content == "decrease"){
    $query = "SELECT * FROM `Lines` ORDER BY `distance` DESC ";
}
else{
    $query = "SELECT * FROM `Lines` ORDER BY `id` ASC";
}
$result = mysql_query($query) or die(mysql_error());
unset($name_start);
unset($name_end);
while($row = mysql_fetch_assoc($result)){
    if( $row['name_start']!='')
        $name_start[] = $row['name_start'];
}
$result = mysql_query($query) or die(mysql_error());
while($row = mysql_fetch_assoc($result)){
    if( $row['name_end']!='')
```

```
 $name_end[] = $row['name_end'];
}

$i=0;
while($i<200){

   $query = "SELECT `X` FROM `A` WHERE `Name` = '$name_start[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $start_X[] = $row['X'];
   }
   $query = "SELECT `Y` FROM `A` WHERE `Name` = '$name_start[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $start_Y[] = $row['Y'];
   }

   $query = "SELECT `X` FROM `A` WHERE `Name` = '$name_end[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $end_X[] = $row['X'];
   }
   $query = "SELECT `Y` FROM `A` WHERE `Name` = '$name_end[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $end_Y[] = $row['Y'];
   }
   $query = "SELECT `X` FROM `B` WHERE `Name` = '$name_start[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $start_X[] = $row['X'];
   }
   $query = "SELECT `Y` FROM `B` WHERE `Name` = '$name_start[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $start_Y[] = $row['Y'];
   }

   $query = "SELECT `X` FROM `B` WHERE `Name` = '$name_end[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $end_X[] = $row['X'];
   }
   $query = "SELECT `Y` FROM `B` WHERE `Name` = '$name_end[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $end_Y[] = $row['Y'];
   }

   $query = "SELECT `X` FROM `C` WHERE `Name` = '$name_start[$i]'";
   $result = mysql_query($query) or die(mysql_error());
```

```
 while($row = mysql_fetch_assoc($result)){

      $start_X[] = $row['X'];
   }
   $query = "SELECT `Y` FROM `C` WHERE `Name` = '$name_start[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $start_Y[] = $row['Y'];
   }

   $query = "SELECT `X` FROM `C` WHERE `Name` = '$name_end[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $end_X[] = $row['X'];
   }
   $query = "SELECT `Y` FROM `C` WHERE `Name` = '$name_end[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $end_Y[] = $row['Y'];
   }

   $query = "SELECT `X` FROM `D` WHERE `Name` = '$name_start[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $start_X[] = $row['X'];
   }
   $query = "SELECT `Y` FROM `D` WHERE `Name` = '$name_start[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $start_Y[] = $row['Y'];
   }

   $query = "SELECT `X` FROM `D` WHERE `Name` = '$name_end[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $end_X[] = $row['X'];
   }
   $query = "SELECT `Y` FROM `D` WHERE `Name` = '$name_end[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while($row = mysql_fetch_assoc($result)){
      $end_Y[] = $row['Y'];
   }
   $i++;
}
```

distance_turns.php

```
<?php
include "db.php";

$query = "SELECT * FROM `Lines_with_turns`";
```

```
result = mysql_query($query);
while ($row = mysql_fetch_row($result)) {
   $lines_with_turns_table_list[] = $row[0];
   $lines_with_turns_table_list[] = $row[1];
   $lines_with_turns_table_list[] = $row[2];
   $lines_with_turns_table_list[] = $row[3];
   $lines_with_turns_table_list[] = $row[4];
   $lines_with_turns_table_list[] = $row[5];
   $lines_with_turns_table_list[] = $row[6];
}
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $turn_name_start[]=$row['Start'];
}
$result = mysql_query($query);
   while ($row = mysql_fetch_assoc($result)) {
      $turn_name_end[]=$row['End'];
   }

$i = 0;

while ($i < 100) {
   $query = "SELECT `X` FROM `A` WHERE `Name` = '$lines_with_turns_table_list[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while ($row = mysql_fetch_assoc($result)) {
      $lines_with_turns_table_list_X[] = $row['X'];
      if (($i + 1) % 7 == 0)
         $lines_with_turns_table_list_X[] = 0;
   }
   $query = "SELECT `Y` FROM `A` WHERE `Name` = '$lines_with_turns_table_list[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while ($row = mysql_fetch_assoc($result)) {
      $lines_with_turns_table_list_Y[] = $row['Y'];
      if (($i + 1) % 7 == 0)
         $lines_with_turns_table_list_Y[] = 0;
   }
   $query = "SELECT `X` FROM `B` WHERE `Name` = '$lines_with_turns_table_list[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while ($row = mysql_fetch_assoc($result)) {
      $lines_with_turns_table_list_X[] = $row['X'];
      if (($i + 1) % 7 == 0)
         $lines_with_turns_table_list_X[] = 0;
   }
   $query = "SELECT `Y` FROM `B` WHERE `Name` = '$lines_with_turns_table_list[$i]'";
   $result = mysql_query($query) or die(mysql_error());
   while ($row = mysql_fetch_assoc($result)) {
      $lines_with_turns_table_list_Y[] = $row['Y'];
      if (($i + 1) % 7 == 0)
         $lines_with_turns_table_list_Y[] = 0;
   }
```

```
    $query = "SELECT `X` FROM `C` WHERE `Name` = '$lines_with_turns_table_list[$i]'";
    $result = mysql_query($query) or die(mysql_error());
    while ($row = mysql_fetch_assoc($result)) {
        $lines_with_turns_table_list_X[] = $row['X'];
        if (($i + 1) % 7 == 0)
            $lines_with_turns_table_list_X[] = 0;
    }
    $query = "SELECT `Y` FROM `C` WHERE `Name` = '$lines_with_turns_table_list[$i]'";
    $result = mysql_query($query) or die(mysql_error());
    while ($row = mysql_fetch_assoc($result)) {
        $lines_with_turns_table_list_Y[] = $row['Y'];
        if (($i + 1) % 7 == 0)
            $lines_with_turns_table_list_Y[] = 0;
    }
    $query = "SELECT `X` FROM `D` WHERE `Name` = '$lines_with_turns_table_list[$i]'";
    $result = mysql_query($query) or die(mysql_error());
    while ($row = mysql_fetch_assoc($result)) {
        $lines_with_turns_table_list_X[] = $row['X'];
        if (($i + 1) % 7 == 0)
            $lines_with_turns_table_list_X[] = 0;
    }
    $query = "SELECT `Y` FROM `D` WHERE `Name` = '$lines_with_turns_table_list[$i]'";
    $result = mysql_query($query) or die(mysql_error());
    while ($row = mysql_fetch_assoc($result)) {
        $lines_with_turns_table_list_Y[] = $row['Y'];
        if (($i + 1) % 7 == 0)
            $lines_with_turns_table_list_Y[] = 0;
    }

    $query = "SELECT `X` FROM `E` WHERE `Name` = '$lines_with_turns_table_list[$i]'";
    $result = mysql_query($query) or die(mysql_error());
    while ($row = mysql_fetch_assoc($result)) {
        $lines_with_turns_table_list_X[] = $row['X'];
    }
    $query = "SELECT `Y` FROM `E` WHERE `Name` = '$lines_with_turns_table_list[$i]'";
    $result = mysql_query($query) or die(mysql_error());
    while ($row = mysql_fetch_assoc($result)) {
        $lines_with_turns_table_list_Y[] = $row['Y'];
    }

    $i++;
}
```

export.php

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf8"/>
```

```
 <title>Coordinates</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <link rel="icon"
          type="image/png"
          href="pickaxe.png">
</head>
<body>
<div class="export_wrapper">

<?php
include'db.php';
$export = fopen(dirname(__FILE__).'/user_db/export.sql','w');
function export($sql) {
global $export;
fwrite($export,$sql);
ob_flush();
}
function trace($msg){
echo $msg.'<br>';
ob_flush();
}
mysql_query('set names utf8');

$res=mysql_query('show tables');

while($tbl=mysql_fetch_array($res)){
$table=$tbl[0];
$r=mysql_query('show create table `'
.mysql_real_escape_string($table).'`');
$struct=mysql_fetch_array($r);
$sql_struct[$table]=$struct[1].';';
}

export("set names utf8;\n");

foreach($sql_struct as $tbl_name=>$crt_str){
trace('Идет экспорт '.$tbl_name);
export("DROP TABLE IF EXISTS `".$tbl_name."`;\n");
export($crt_str."\n");
export("LOCK TABLES `".$tbl_name."` WRITE;\n");
mysql_query('LOCK TABLES `'.$tbl_name.'` READ');
$res=mysql_query('select * from `'.$tbl_name.'`');
$insert_str='insert into `'.$tbl_name.'` values ';
while($item=mysql_fetch_assoc($res)){
foreach($item as $k=>$v){
$item[$k]=mysql_real_escape_string($v);
}
export($insert_str.'("'.implode('","',$item).'");'."\n");
}
export("UNLOCK TABLES;\n");
mysql_query('UNLOCK TABLES');
```

```
}

export('-- end of export');
trace('<a href="/coords_newest/coords/user_db/export.sql">Скачать файл экспорта</a><br><a
href="main_coords.php">Вернуться на главную</a><br>');
//trace('База была успешно экспортирована - сохраните файл:
http://diplombackup.zz.mu/coords/export.sql');
?>
   </div>
```

icon_coords.php

```
<?php
//---------- Опрацювання точки (Повторювач)----------
$query = "SELECT `X` FROM `A`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $A_X[] = $row['X'];
}
$query = "SELECT `Y` FROM `A`";
$result = mysql_query($query);
```



```
while ($row = mysql_fetch_assoc($result)) {
   $A_Y[] = $row['Y'];
}

//----------отримання імен з БД----------
$query = "SELECT `Name` FROM `A`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $A_Name[] = $row['Name'];
}
//----------
$i = 0;
while ($i < 20) {
   if (($A_Y[$i] != 0) && ($A_X[$i] != 0)) {
      ?>
      <div style="position: absolute; top: <? echo $A_Y[$i] - 8 ?>px; left: <? echo $A_X[$i] - 8
?>px">
         <!--          <img class="dot" src='repeater.png'/>-->

         <form action=" method=post>
            <input class="dot" type="image" src="repeater.png" name="<? echo $A_Name[$i]
?>"title="<? echo $A_Name[$i]?>"/>
         </form>
      </div>
   <?
   }

   $i++;
}
```

```
//-------------------------------------Опрацювання точки (Комутатор)--------------------
$query = "SELECT `X` FROM `B`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $B_X[] = $row['X'];
}

$query = "SELECT `Y` FROM `B`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $B_Y[] = $row['Y'];
}
$query = "SELECT `Name` FROM `B`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $B_Name[] = $row['Name'];
}
$i = 0;
while ($i < 20) {
   if (($B_Y[$i] != 0) && ($B_X[$i] != 0)) {
      ?>
      <div style="position: absolute; top: <? echo $B_Y[$i] - 8 ?>px; left: <? echo $B_X[$i] - 8
?>px">
         <!--          <img class="dot" src='switch.png'/>-->
         <form action=" method=post>
            <input class="dot" type="image" src="switch.png" name="<? echo $B_Name[$i]
?>"title="<? echo $B_Name[$i]?>"/>
         </form>
      </div>
   <?
   }
   $i++;
}
//------------------------------------ Опрацювання точки (Контролер)-------------
$query = "SELECT `X` FROM `C`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $C_X[] = $row['X'];
}

$query = "SELECT `Y` FROM `C`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $C_Y[] = $row['Y'];
}
$query = "SELECT `Name` FROM `C`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $C_Name[] = $row['Name'];
}
```

29

```
$i = 0;
while ($i < 20) {
   if (($C_Y[$i] != 0) && ($C_X[$i] != 0)) {
      ?>
      <div style="position: absolute; top: <? echo $C_Y[$i] - 8 ?>px; left: <? echo $C_X[$i] - 8
?>px">
         <!--          <img class="dot" src='controller.png'/>-->
         <form action=" method=post>
            <input class="dot" type="image" src="controller.png" name="<? echo $C_Name[$i]
?>" title="<? echo $C_Name[$i]?>"/>
         </form>
      </div>
   <?
   }
   $i++;
}
//------------------------------------- Опрацювання точки (Резерв)--------------------
$query = "SELECT `X` FROM `D`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $D_X[] = $row['X'];
}

$query = "SELECT `Y` FROM `D`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $D_Y[] = $row['Y'];
}

$query = "SELECT `Name` FROM `D`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $D_Name[] = $row['Name'];
}

$i = 0;
while ($i < 20) {
   if (($D_Y[$i] != 0) && ($D_X[$i] != 0)) {
      ?>
      <div style="position: absolute; top: <? echo $D_Y[$i] - 8 ?>px; left: <? echo $D_X[$i] - 8
?>px">
         <form action=" method=post>
            <input class="dot" type="image" src="reserve.png" name="<? echo $D_Name[$i] ?>"
title="<? echo $D_Name[$i]?>"/>
         </form>
      </div>
   <?
   }
   $i++;
}
```

30

icon_coords_save.php

```
<?php
$ink_dot1  = imagecolorallocate($img,  255, 0, 0);
$ink_dot2  = imagecolorallocate($img,  0, 0, 255);
$ink_dot3  = imagecolorallocate($img,  0, 255, 0);
$ink_dot4  = imagecolorallocate($img,  255, 102, 0);

//------------------------------------ Опрацювання точки (Повторювач)----------
$query = "SELECT `X` FROM `A`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $A_X[] = $row['X'];
}
$query = "SELECT `Y` FROM `A`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $A_Y[] = $row['Y'];
}

//---------------------------отримання імен з БД--------------------
$query = "SELECT `Name` FROM `A`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $A_Name[] = $row['Name'];
}
//-------------------------------------------------------

$i = 0;
while ($i < 20) {
   if (($A_Y[$i] != 0) && ($A_X[$i] != 0)) {

      imagefilledarc($img,$A_X[$i],$A_Y[$i],16,16,0,360,$ink_dot1,IMG_ARC_PIE);
        }

   $i++;
}
//------------------------------------ Опрацювання точки (Комутатор)----------------------
$query = "SELECT `X` FROM `B`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $B_X[] = $row['X'];
}

$query = "SELECT `Y` FROM `B`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $B_Y[] = $row['Y'];
}
$query = "SELECT `Name` FROM `B`";
$result = mysql_query($query);
```

```
while ($row = mysql_fetch_assoc($result)) {
   $B_Name[] = $row['Name'];
}
$i = 0;
while ($i < 20) {
   if (($B_Y[$i] != 0) && ($B_X[$i] != 0)) {
      imagefilledrectangle($img,$B_X[$i]-8,$B_Y[$i]-8,$B_X[$i]+8,$B_Y[$i]+8,$ink_dot3);
   }
   $i++;
}
//------------------------------------ Опрацювання точки (Контролер)----------

$query = "SELECT `X` FROM `C`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $C_X[] = $row['X'];
}

$query = "SELECT `Y` FROM `C`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $C_Y[] = $row['Y'];
}
$query = "SELECT `Name` FROM `C`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $C_Name[] = $row['Name'];
}

$i = 0;
while ($i < 20) {
   if (($C_Y[$i] != 0) && ($C_X[$i] != 0)) {

      imagefilledrectangle($img,$C_X[$i]-8,$C_Y[$i]-8,$C_X[$i]+8,$C_Y[$i]+8,$ink_dot2);

   }
   $i++;
}
//------------------------------------ Опрацювання точки (Резерв)--------------------

$query = "SELECT `X` FROM `D`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $D_X[] = $row['X'];
}

$query = "SELECT `Y` FROM `D`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $D_Y[] = $row['Y'];
```

32

```
}

$query = "SELECT `Name` FROM `D`";
$result = mysql_query($query);
while ($row = mysql_fetch_assoc($result)) {
   $D_Name[] = $row['Name'];
}

$i = 0;
while ($i < 20) {
   if (($D_Y[$i] != 0) && ($D_X[$i] != 0)) {
      $triangle_coords=array($D_X[$i]+8,$D_Y[$i]+8,$D_X[$i]-
8,$D_Y[$i]+8,$D_X[$i],$D_Y[$i]-8);
      imagefilledpolygon ($img, $triangle_coords, 3, $ink_dot4);
   }
   $i++;
}
```

import.php

```
<!DOCTYPE html>
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf8"/>
   <title>Coordinates</title>
   <link rel="stylesheet" type="text/css" href="style.css">
   <link rel="icon"
      type="image/png"
      href="pickaxe.png">
</head>
<body>
<div class="import_wrapper">
<?php
$allowed_extensions = array('sql');
include 'db.php';
if (!empty($_FILES['file'])) {
   if ($_FILES['file']['error'] == 0) {
      // check extension
      $file = explode(".", $_FILES['file']['name']);
      $extension = array_pop($file);
      if (in_array($extension, $allowed_extensions)) {

         $uploaddir = 'user_db/'; //директория файла, где img - папка
         $uploadfile = $uploaddir . "export.sql"; //переменная, в которой будет
приблицительно такая инфа = /img/*загруженный файл*
         move_uploaded_file($_FILES['file']['tmp_name'], $uploadfile); //перемещаем из
временного хранилища по адресу из переменной $uploadfile
         $sqlfile = 'user_db/export.sql';
         if (!file_exists($sqlfile)) ;
         $open_file = fopen($sqlfile, "r");
         $buf = fread($open_file, filesize($sqlfile));
```

```
 fclose($open_file);

          $a = 0;

          while ($b = strpos($buf, ";", $a + 1)) {
             $i++;
             $a = substr($buf, $a + 1, $b - $a);
             mysql_query($a);
             $a = $b;
          }

//echo "Загружено таблиц:" . $i;
          echo "Загрузка завершена";
       } else {
          echo "Только расширение .sql разрешено";
       }
   } else {
      echo "Загрузка неудачна";
   }
}
else{
   echo"Загрузка неудачна, возможно вы ввели этот адрес вручную";
}

?>
   <br><a href='main_coords.php'>Вернуться на главную</a><br>
   </div>
```

points.php

```
<?
header ("Content-type: image/png");
include 'distance.php';
$i=0;

$img = imagecreatefromjpeg("img/resized.jpg");
$ink = imagecolorallocate($img, 1, 1, 1);

imageinterlace($img, true);
imagesetthickness($img, 4);

while($i<count($name_end))
{
imageline($img,$start_X[$i],$start_Y[$i],$end_X[$i],$end_Y[$i],$ink);
   $i++;
}
//------------- ОПРАЦЮВАННЯ ПОВОРОТНИХ ЛІНІЙ ----------------------------------
include 'distance_turns.php';
$i=0;
while($i<count($lines_with_turns_table_list_Y))
```

```
{
   if(($lines_with_turns_table_list_X[$i+1]!=0)&&($lines_with_turns_table_list_X[$i]!=0))
   {

imageline($img,$lines_with_turns_table_list_X[$i],$lines_with_turns_table_list_Y[$i],$lines_wi
th_turns_table_list_X[$i+1],$lines_with_turns_table_list_Y[$i+1],$ink);
   }
   $i++;

}
imagejpeg($img, NULL, 50);
imagejpeg($img, 'imagesave.jpg', 50);
imagedestroy($img);

}
```