Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2020 р

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «**МОДЕЛЬ КОРИГУВАЛЬНОГО КАСКАДНОГО КОДУ В КАНАЛАХ ПЕРЕДАЧІ ДАНИХ СИСТЕМИ УПРАВЛІННЯ»**

Захищено на засіданні
Атестаційної комісії №
протокол № __ від __.12.2021 р.
Оцінка _____ / _____
Голова Атестаційної комісії

(підпис) (прізвище та ініціали)

Виконав:

студент 6 курсу, групи КУ– 61
за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології.
Шаров Владислав Олегович___

Керівник:

к.т.н., доцент кафедри теоретичної та прикладної системотехніки
Бердніков Анатолій Георгійович_____

Рецензент:

Харків – 2021

АНОТАЦІЯ

Дипломна робота складається зі вступу, п’ятьох розділів, висновків, списку використаної літератури і 3 додатків. Загальний обсяг роботи становить 97 сторінок, з них 78 сторінок основного тексту з 17 таблицями, 54 рисунками, 15 найменуваннями переліку використаної джерел на 2 сторінках і 3 додатки на 19 сторінках.

Мета роботи - розробка моделі завадостійкого каскадного коду, який зможе використовуватись автоматизованих системах управління технологічним процесом.

Проведено аналіз сучасних існуючих способів завадостійкої передачі даних, статистики помилок. Використаний класичний підхід у боротьбі за завадостійкість, а саме використання завадостійких кодів. Задля вирішення проблеми використовується каскадний код на базі систематичного коду.

Зроблено вибір програмної платформи і середи розробки, опираючись на сучасні вимоги до програмних моделей. Реалізовано алгоритм роботи комп'ютерної моделі завадостійкого каскадного коду.

Об’єктом дослідження являється алгоритм завадостійкого кодування інформації каскадними кодами, дозволяючий підвищувати достовірність повідомлень каналів передачі даних у локальних обчислювальних мережах.

Предметом дослідження є передача інформації по дискретним каналам зв’язку.

Мета роботи: підвищення завадостійкості у дискретних каналах передачі даних шляхом створення моделі блочного систематичного каскадного коду для передачі даних дискретним каналам у автоматизованих системах управління технологічним процесом.

Завдання дослідження: Дослідити процес підвищення вірності передачі даних у системі управління за використанням завадостійких кодів.

КЛЮЧОВІ СЛОВА: завадостійкість, кодер, декодер, каскадні коди, систематичні, боротьба за завадостійкість, канал передачі даних, протокол АТМ.

ABSTRACT

Thesis consists of an introduction, five chapters, conclusions, list of references and 3 appendices. The total volume of the work is 97 pages, including 78 pages of the main text with 17 tables, 54 figures, 15 names of the list of used sources on 2 pages and 3 appendices on 19 pages.

The purpose of the work is to develop a model of noise-tolerant cascade code that can be used by automated process control systems.

The analysis of modern existing methods of noise-tolerant data transmission, error statistics is carried out. The classic approach in the fight for noise immunity is used, namely the use of noise-tolerant codes. To solve the problem, a cascading code based on systematic code is used.

The choice of software platform and development environment is made, relying on modern requirements to software models. The algorithm of operation of the computer model of noise-tolerant cascade code is realized.

The object of the study is an algorithm for noise-tolerant encoding of information by cascading codes, which allows to increase the reliability of messages of data transmission channels in local area networks.

The subject of the study is the transmission of information through discrete communication channels.

Purpose: to increase noise immunity in discrete data transmission channels by creating a model of block systematic cascade code for data transmission to discrete channels in automated process control systems.

Research objectives: To investigate the process of increasing the accuracy of data transmission in the control system using noise-tolerant codes.

KEY WORDS: noise immunity, encoder, decoder, cascade codes, systematic, noise immunity struggle, data transmission channel, ATM protocol.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АТМ – (Asynchronous Transfer Mode) – протокол асинхронний режим передачі;

ISDN – (Integrated Services Digital Network) – система, яка забезпечує передачу трафіку;

СПДП – система передачі дискретних повідомлень;

ПЗП – пристрій захисту від помилок

ППС – пристрій перетворення сигналу

РДК – розширений дискретний канал

ПП – пороговий пристрій

ВСТУП

У сучасні часи, комунікація між різними людьми являється основним стовпом, на якому тримається суспільство. Інформаційні технології, натомість, допомагають людині з даним питанням, розширюючи потенціал комунікації у сотні разів. Чітка, достовірна, об'єктивна і своєчасна інформація дедалі набуває все більшого значення для функціонування нашого суспільства.

Але при передачі інформації по складовим глобальних світових мереж виникають багато перепон, через які процес передачі інформаційних сигналів ускладнюється, а у деяких випадках стає взагалі неможливим. З цієї причини вдосконалюються технології, винаходяться нові методи, які дозволяють долати ці перепони, або їх мінімізувати настільки, щоб вони не заважали передавати інформацію.

Отже, можна виділити актуальність проблеми перепоностійкої (завадостійкої) передачі інформації, а тому і дослідження, які проводяться по даній темі також є актуальними, і саме на цій темі буде присвячена дана робота.

Перепони при передачі інформації також бувають різними. У сучасних системах передачі даних основною перепоною виступають помилки, які трапляються у момент передачі інформації від джерела до отримувача. Помилок може бути мало, може бути багато, вони можуть бути різного типу – що у купі робить процес передачі по системам досить складним.

Для підтримки вірної передачі даних використовуються багато методів, які дозволяють виключати або мінімізувати помилки. В залежності від систем і потреб, обираються і розробляються ті методи, які по встановленим критеріям будуть максимально ефективні у певній системі за певних обставин. Для систем, що використовують для зв'язку канали передачі без черг, також існують низка способів боротьби з помилками, але дана ніша також підлягає вивченню та покращенню.

На практиці, якщо при передачі повідомлення трапляється помилка, означає, що дане повідомлення необхідно перезапитати, а це значить, що до того моменту, поки повідомлення повторно не надійде, буде образована черга з інших повідомлень, тобто час на передачу повідомлення буде збільшено, а це вже означає зменшення пропускної здатності. З цієї причини, зараз багато уваги сконцентровано на методах підвищення завадостійкості, в тому числі і тих методах, які зможуть виправляти помилки на отримуючих пристроях, уникаючи необхідності робити перезапит, тому і збільшуючи пропускну здатність.

Для маленьких та не потужних каналів, які використовуються у локальних вираховувальних мережах, або у локальних системах управління різними процесами, в силу їх простоти та відносно невеликої ресурсної бази, необхідні такі ж прості й не затратні по ресурсам методи, які дозволять передавати необхідну інформацію без помилок. Оскільки канали у таких мережах можуть використовуватись для різних процесів, методи передачі, а тому також і контролю над помилками також мають бути універсальними.

Тому для різних типів трафіку, який проходить по каналам передачі даних, висуваються різні вимоги, важливим стає пошук компромісу, який буде задовольняти усім потребам, залишаючись оптимальним при різних форматах даних, що дозволить використовувати меншу кількість каналів, а тому зможе зекономити велику кількість ресурсів.

З цього можно зробити висновок, що розробка моделей, які зможуть адаптивно працювати з різними алгоритмами та протоколами, дозволять контролювати помилки, або будуть боротись з помилками, чим зможуть підвищити достовірність інформації і пропускну здатність каналів є актуальною.

Розділ 1
ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ, ПОСТАНОВКА ЗАДАЧІ

1.1 Джерело дискретних повідомлень, його характеристики

Для того, щоб розпочати роботу, розглянемо середу і предмет роботи, для яких проходить дослідження по темі.

Виділимо наступні поняття: *інформація, повідомлення, сигнал.*

Під *інформацією* розуміються відомості, які являються об’єктом передачі, розподілення, збереження, перетворення або, безпосередньо, використання.

Повідомлення – форма уявлення інформації.

Одні й ті ж самі відомості можуть бути представлені у різних формах. Наприклад, відомості про момент початку наступу може бути передано телефоном або телеграфом, або трьома зеленими ракетами. У першому випадку ми маємо справу з інформацією, поданою у безперервному вигляді (безперервне повідомлення). Вважатимемо, що це повідомлення виробляється *джерелом безперервних повідомлень*. У другому та третьому випадку - з інформацією, поданою в дискретному вигляді (дискретне повідомлення). Це повідомлення виробляється *джерелом дискретних повідомлень*.

Основна відмінність дискретного та безперервного джерел полягає в наступному. Багато різних повідомлень, що виробляються дискретним джерелом завжди кінцеве. Тому, на кінцевому відрізку часу кількість символів дискретного джерела також є кінцевою. У той же час, кількість можливих різних значень звукового тиску (або напруги в телефонній лінії), виміряне під час розмови, навіть на кінцевому відрізку часу, буде нескінечним.

Інформація, що міститься в повідомленні, передається від джерела повідомлень до одержувача *каналом передачі дискретних повідомлень*

Повідомлення надходить від джерела дискретних повідомлень, який характеризується *алфавітом* повідомлень, що передаються $A\{a_1, a_2, \dots, a_i\}$.

Алфавіт є сукупністю всіх можливих (різних) повідомлень (знаків) даного джерела.

Об'єм алфавіту – кількість різних символів алфавіту *K*. Кожне повідомлення алфавіту з'являється певною вірогідністю.

Вірогідність видачі символу (повідомлення) $a_i - p(a_i)$.

Один біт - це кількість інформації, яка переносить один символ джерела дискретних повідомлень у разі, коли алфавіт джерела складається з двох рівновірогідних символів.

Особливо необхідно відмітити *достовірність інформації*. Це – властивість інформації на виході системи відповідати інформації, що надійшла на її вхід.

Для каналів передачі дискретних повідомлень вводять аналогічну характеристику – *швидкість передачі інформації* по каналу *R*. Вона визначається кількістю біт, що передаються в секунду. Максимально можливе значення швидкості передачі по каналу називається *пропускною здатністю* каналу і позначається *C*.

Сигнали – форма повідомлення для передачі по каналу зв'язку

Вкрай важливим показником при передачі сигналів є *час розповсюдження сигналу по лінії зв'язку*:

$$t_p = \frac{D_{max}}{V_c}, \qquad (1.1)$$

де D_{max}– довжина каналу, V_c– швидкість розповсюдження сигналу у середі.

Для оцінки часових характеристик припустимо, що сигнал передається від станції ***А*** до станції ***Б***. На станції ***Б*** сигнал обробляється на протязі часу $t_{\text{оброб}}$ і до станції ***А*** відправляється квитанція про прийом. Тоді загальний час очікування відповіді на станції ***А*** буде дорівнювати

$$T_{очік} = 2 * t_р + t_{оброб}, \quad (1.2)$$

Будь-яка система зв'язку забезпечує передачу саме *сигналів*, а не повідомлень. Тому повідомлення, що надходить від джерела, попередньо має бути перетворено на *сигнал певної природи* (електричний, оптичний…), який є його переносником у цій системі зв'язку.

Види сигналів. Розрізняють чотири види сигналів: безперервний безперервний час, безперервний дискретний час, дискретний безперервний час і дискретний дискретний час.

Безперервні сигнали безперервного часу називають скорочено безперервними (аналоговими) сигналами. Вони можуть змінюватися в довільні моменти, приймаючи будь-які значення безперервного безлічі можливих значень (рис.1.1). До таких сигналів відноситься і відома всім синусоїда.

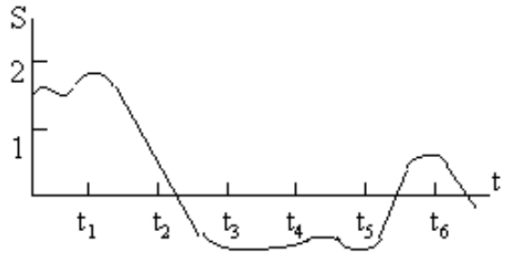

Рисунок 1.1 – Безперервний сигнал безперервного часу

Безперервні сигнали дискретного часу можуть набувати довільних значень, але змінюватися лише певні, наперед задані (дискретні) моменти t_1, t_2, t_3, ... (рис.1.2).

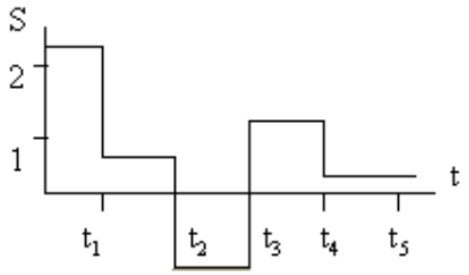

Рисунок 1.2 – Безперервний сигнал безперервного часу

Дискретні сигнали безперервного часу відрізняються тим, що можуть змінюватися в довільні моменти, та їх величини приймають лише дозволені (дискретні) значення (рис.1.3).

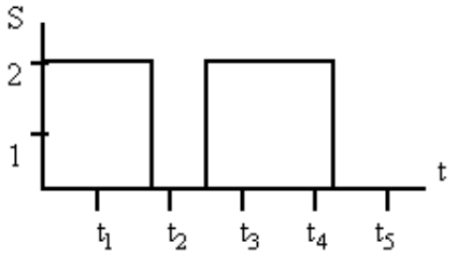

Рисунок 1.3 – Дискретні сигнали безперервного часу

*Дискретні сигнали дискретного час*у (скорочено дискретні) (рис.1.4) в дискретні моменти часу можуть набувати лише дозволені (дискретні) значення.

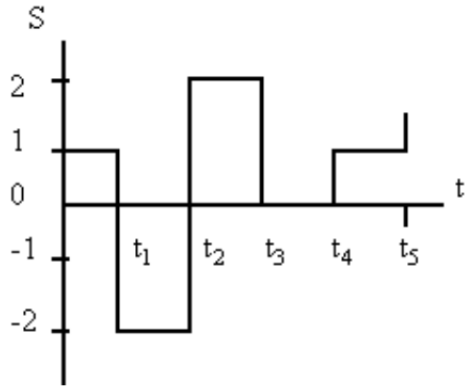

Рисунок 1.4 – Дискретні сигнали дискретного часу

Структурна схема системи передачі дискретних повідомлень (ПДП) зображена на рис.1.5. Джерело та отримувач повідомлень разом перетворювачем повідомлення до складу ПДП не входять.[1]

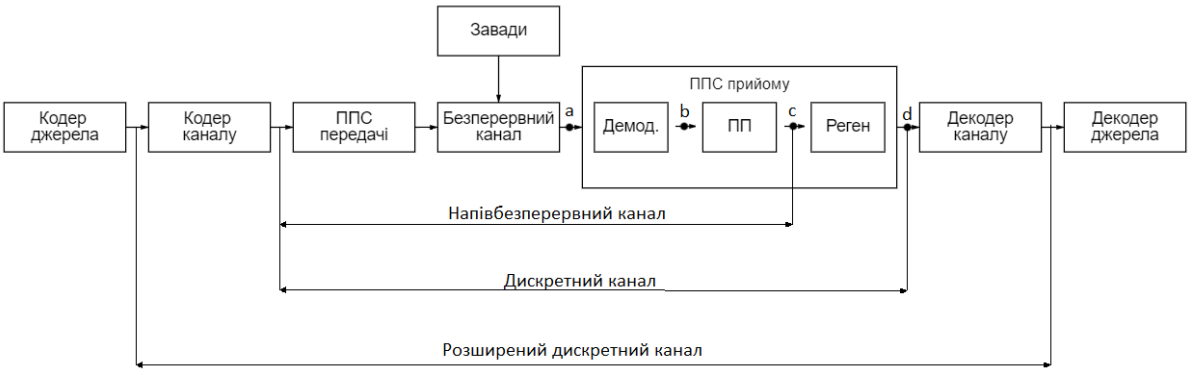

Рисунок 1.5 – Структурна схема системи передачі дискретних повідомлень

Кодер джерела. Повідомлення, що надходить від джерела повідомлень, у ряді випадків містять надлишковість(надмірність). Це пов'язано з тим, що символи , які входять у повідомлення, можуть бути статистично пов'язаними. Це дозволяє частину повідомлення не передавати, відновлюючи його на прийомі за допомогою відомого статистичного зв'язку.

Надмірність призводить до того, що за заданий проміжок часу буде передано менше повідомлень, і, отже, менш ефективно використовуватиметься канал передачі дискретних повідомлень. Завдання усунення надмірності передачі у СПДП виконує кодер джерела.

Кодер каналу. З метою підвищення вірності передачі використовується надлишкове кодування, що дозволяє на прийомі виявляти чи навіть виправляти помилки. Детальніше це питання буде розглядатись у наступному розділі.

У процесі кодування здійснюється перетворення вихідної кодової комбінації на іншу кодову комбінацію з надмірністю. На приймальному кінці

декодер каналу здійснює зворотне перетворення (декодування), в результаті якого отримуємо комбінацію вихідного коду. Часто кодер та декодер каналу називають *пристроями захисту від помилок* (ПЗП).

Пристрій перетворення сигналу. З метою узгодження кодера каналу та декодера каналу з безперервним каналом зв'язку, використовуються на передачі та прийомі пристрої перетворення сигналів (ППС). В окремому випадку це модулятор та демодулятор.

Розглянемо детальніше канали.

Безперервний канал. Це канал зв'язку, призначений передачі безперервних (аналогових) сигналів. Наприклад, абонентська телефонна лінія, тощо.

Дискретний канал. Спільно з каналом зв'язку, ППС утворюють дискретний канал, тобто канал, призначений передачі лише дискретних сигналів (цифрових сигналів даних).

Розрізняють *синхронні* та *асинхронні* дискретні канали.

У *синхронних дискретних каналах* введення кожного одиничного елемента проводиться в певні моменти, часу і вони призначені для передачі тільки ізохронних сигналів.

Асинхронним каналом можна передавати будь-які сигнали - ізохронні, анізохронні.

Розширений канал. Дискретний канал разом із кодером і декодером каналу (ПЗП) називається розширеним дискретним каналом (РДК).

У техніці передачі РДК називають каналом передачі.

Напівбезперервний канал (*дискретний канал безперервного часу*).

У системі ПДП іноді виділяють дискретний канал безперервного часу.

Для визначення виходу даного каналу необхідно детальніше розглянути ППС прийому. Він складається з демодулятора, порогового пристрою (ПП) та регенератора. Вихід ПП одночасно є виходом дискретного каналу безперервного часу.

Якщо на виході дискретного каналу маємо сигнал, що є дискретною функцією дискретного часу, то на виході напівбезперервного каналу сигнал є дискретною функцією безперервного часу. (Він же канал постійного струму).

1.2 Мережі передачі даних

Для передачі даних (обміну даними, інформацією, повідомленнями), тобто фізичного переносу даних (цифрового бітового потоку) у вигляді сигналів від точки до точки, або від точки до декількох приладів електрозв’язку по каналу передачі даних, як правило, використовують мідні (електричні) дроти, волоконно-оптичні системи лінії зв’язку, безпровідні канали передачі даних. Тому, в нашому випадку, для дослідження по темі необхідно сформувати систему, на основі якої буде базуватись робота.

Мережа передачі даних — сукупність трьох і більше кінцевих пристроїв (терміналів) зв'язку, об'єднаних каналами передачі даних і пристроями, що комутують (вузлами мережі), що забезпечують обмін повідомленнями між усіма кінцевими пристроями.

Існують такі види мереж передачі:

- Телефонні мережі — мережі, у яких кінцевими пристроями є прості перетворювачі сигналу між електричним та видимим/чутним.
- Комп'ютерні мережі — це мережі, кінцевими пристроями яких є комп'ютери.

Універсальними можливо ті системи, які використовують асинхронне тимчасове мультиплексування даних. Тут розуміється здатність мережі передавати трафік різного типу: чутливий до затримок (наприклад, голосовий) та еластичний, тобто допускає затримки в широких межах(наприклад, трафік електронної пошти або перегляд веб-сторінок).[11]

В сучасних системах управління широко застосовуються інтегральні цифрові мережі (Integrated Services Digital Network - ISDN), які забезпечують передачу всіх видів дискретної інформації в стандартному форматі.

Як приклади мереж з інтегрованим обслуговуванням можна привести універсальні протоколи ATM (Asynchronous Transfer Mode - асинхронний

режим передачі) або Broadband ISDN (B-ISDN), які забезпечують передачу будь-якого типу трафіку, як комп'ютерного, так і мультимедійного.

Технологія ATM використовується на багатьох магістралях найбільших операторів зв'язку і підтримує всі види трафіку, може використовуватися безпосередньо прикладним рівнем протоколів і працювати автономно без протоколів TCP / IP і TCP / UDP, однак наявність досить складного власного протоколу маршрутизації призвело до того, що в сучасних комп'ютерних мережах для об'єднання мереж використовуються, як правило, протоколи TCP / IP і TCP / UDP, але ATM залишається однією з технологій, на основі якої працюють багато локальних мереж, утворюють складну складену мережу, вимоги до яких, як приклад, дозволять нам сформувати вимоги до моделі.

Але при передачі інформації по безперервним каналам зв’язку, автоматизованих системах управління, або в цілому по локальним мережам, як було показано на рисунку (1.5), перепоною стають завади, які мають велике значення у теорії інформації і кодування, а також значно ускладнюють процес передачі повідомлень. Розглянемо детальніше дане питання.

1.3 Причини виникнення помилок

Хоча у роботі і будуть розглядатись саме дискретні канали, необхідно спочатку з’ясувати питання щодо завад, які трапляються у безперервних каналах, адже саме вони є однією з основних перепоною для достовірної передачі інформації.

Будь-який технічний засіб може викликати помилки: ситуації, коли двійкова одиниця під впливом ряду факторів може перетворюватися на нуль і навпаки. Контроль цих помилок є найважливішим завданням, яке має бути реалізовано програмно-технічними засобами. Причинами виникнення помилок є *загасання сигналу*, *шум* та *перешкоди*.[2]

Згасання сигналу - це зменшення значення струму, напруги або потужності сигналу, коли частина енергії сигналу розсіюється у вигляді втрат у міру його поширення в лінії зв'язку, що знижує чіткість прийому.

Шум - це заважаючі сигнали, які поєднуються з сигналом, призначеним для передачі та прийому, і спотворюють його. Білий шум (тепловий шум) – статистичний однорідний шум. Він є результатом руху заряджених частинок у фізичному середовищі передачі. Білий шум впливає на всі електричні прилади та проявляється незалежно від фізичного середовища. Усунути тепловий шум неможливо, тому найвища продуктивність систем передачі даних (СПД) обмежена саме його величиною.

Перешкода – небажана енергія, що призводить до спотворення сигналу під час обробки, зберігання та передачі інформації. Виділяють джерела перешкод, у яких відбувається формування цієї небажаної енергії, приймачі перешкод, які отримують спотворений сигнал, канал зв'язку між ними.

Перешкоди можуть виникати поза апаратними засобами, що приймають сигнал, так і всередині них. У першому випадку говорять про зовнішні перешкоди, у другому — про внутрішні. Прикладом зовнішніх джерел перешкод можуть бути грозові розряди, джерела потужного радіовипромінювання, електрозварювальне обладнання, електрифікований транспорт. Як приклади внутрішніх перешкод можна вказати електромагнітні збурення, пов'язані з роботою окремих внутрішніх пристроїв, наприклад, конденсаторів. Як зовнішніх, так і внутрішніх джерел перешкод може бути кілька. Окремі пристрої можуть бути і приймачами перешкод, та їх джерелами інших пристроїв. Приймачами перешкод вважаються пристрої та вхідні електричні кола апаратури каналів зв'язку.

Розрізняють корисний сигнал $S_{\mathrm{c}}(\mathrm{t})$, тобто. сигнал, що несе інформацію, яку слід передати, та сигнали перешкод $S_{\mathrm{n}}(\mathrm{t})$, що генеруються джерелами перешкод. Вказівка параметра t означає, що корисний сигнал та сигнал перешкод залежать від конкретного моменту часу.

Результуючий сигнал $\alpha(t)$ виходить як сумарна величина корисного сигналу і сигналу перешкод. Якщо $\alpha(t)$ може бути представлений у вигляді $\alpha(t) = S_{\mathrm{c}}(\mathrm{t}) + S_{\mathrm{n}}(\mathrm{t})$, то перешкода називається адитивною. Якщо ж $\alpha(t)$ представляється у вигляді $\alpha(t) = S_{\mathrm{c}}(\mathrm{t}) * S_{\mathrm{n}}(\mathrm{t})$, то перешкода називається мультиплікативною.

Якщо сигнал $S_{\mathrm{n}}(\mathrm{t})$ є цілком певною функціональною залежністю, знаючи яку можна абсолютно точно відновити корисний сигнал на основі результуючого, такі перешкоди називаються *регулярними*. Якщо ж відновити вихідний сигнал неможливо, то говорять про *нерегулярні перешкоди*. У цьому випадку відбувається порушення взаємно-однозначної відповідності між вихідним та отриманим сигналами.

За виглядом сигналу завад $S_{\mathrm{n}}(\mathrm{t})$ виділяють два типи завад[2]:

1. *Шумові завади*. В цьому випадку сигнал $S_{\mathrm{n}}(\mathrm{t})$ є безперервною випадковою функцією часу. Зазвичай перешкоди подібного типу є результатом накладання перешкод від багатьох окремих джерел, причому величина енергетичних викидів може перевищувати енергію сигналу в 3-4 рази. Шумові перешкоди $S_{\mathrm{n}}(\mathrm{t})$ найчастіше підпорядковуються нормальному закону розподілу.
2. *Імпульсні завади* (*хаотичні*). У цьому випадку сигнал $S_{\mathrm{n}}(\mathrm{t})$ є послідовністю окремих енергетичних викидів (імпульсів) з випадковими параметрами: амплітудою, моментом появи, тривалістю.

На фізичному рівні помилки трапляються наступним чином. При розгляді дискретних сигналів можна виділити декілька причин помилок: наприклад, краєві спотворення (коли зміщається фронт імпульсу, приклад на рисунку 1.6), можуть утворитися з причини певної затримки t у підсилювальних генераторах, котрі по мірі проходження сигналу по каналу відновлюють (підсилюють) сигнал для відновлення форми імпульсів.

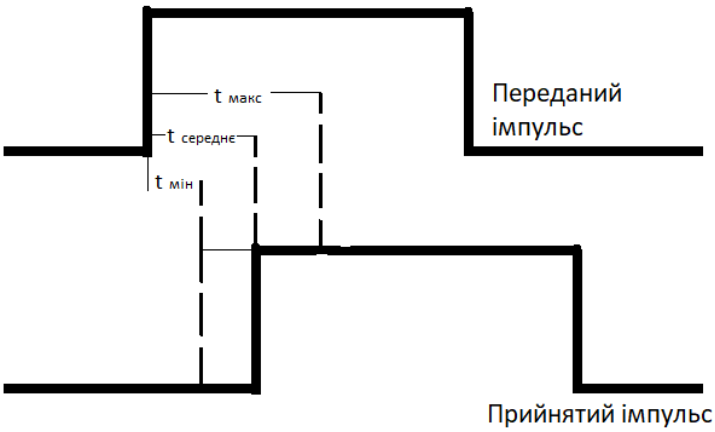

Рисунок 1.6 – Приклад крайового спотворення.

де $t_{макс}$ *та* $t_{мін}$ – межі можливого зміщення фронтів прийнятого імпульсу.

При переході до дискретного повідомлення, це буде означати, що передаваєма комбінація «1110» буде спотворена на «0111» (рис 1.7)

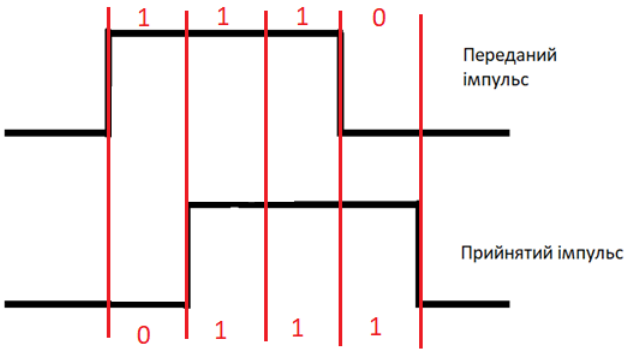

Рисунок 1.7 – Приклад крайового спотворення у дискретному форматі.

Або дроблення імпульсів (різкі зниження рівню або полярності усередені імпульсів, приклад на рисунку 1.8). Вони є результатом нестабільності коефіцієнта передачі канала зв'яку, імпульсних завад та інших. Причинами нестабільності коефіцієнту передачі у дротових лініях зв'язку є: комутація, метеорологічні умови, перенавантаження групових підсилювачів, погані контакти, переключення у апаратурі.

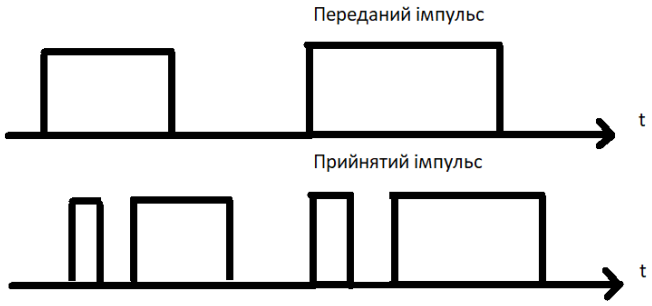

Рисунок 1.8 – Приклад дроблення імпульсів.

При переході до дискретного повідомлення, це буде означати, що передаваєма комбінація «11110000111110» буде спотворена на «01011100101111» (рис 1.9)

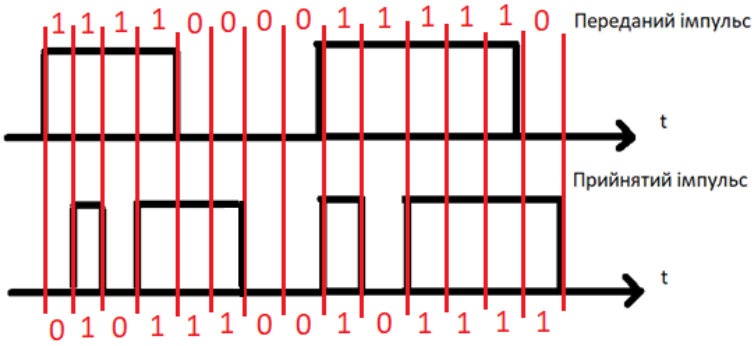

Рисунок 1.9 – Приклад дроблення імпульсів у дискретному форматі.

Помилки у комбінації можуть бути *одноразовими* (однократними), і *багаторазовими*: *дворазовими* (двократними), *трьохразовими* (трьохкратними) і так далі. Багаторазові помилки можуть бути *поодинокими* та *груповими*, де під груповими маються на увазі ті помилки, які трапились поспіль у сусідніх розрядах, а також змішаними, де мають місце одразу поодинокі та групові.

З цього можна зробити висновок, що помилки при передачі даних – розповсюджене явище, з яким необхідно боротись. Але до того, як підібрати методи боротьби, необхідно проаналізувати саме з якими помилками необхідно боротись.

1.4 Статистика помилок, методи боротьби з помилками

У таблиці 1.1 наведені приблизні значення *вірогідностей перекручувань елементарних двійкових символів* p_0 для різних типів каналів зв'язку. Для аналізу буде використовуватись імовірнісний критерій, оскільки його доцільно використовувати для оцінки завадостійкості дискретних каналів зв'язку при відомих імовірнісних характеристиках сигналу і перешкод.[3]

Таблиця 1.1

Вірогідності перекручувань елементарних двійкових символів

Тип каналу зв'язку	Вірогідність p_0
Повітряні проводні канали	$10^{-2} - 10^{-3}$
Кабельні і хвилеводні канали	$10^{-4} - 10^{-6}$
Радіорелейні канали	$10^{-3} - 10^{-4}$
УКХ канали прямої видимості	$10^{-3} - 10^{-4}$
Короткохвильові канали	$10^{-1} - 10^{-2}$

Дана робота базується на використанні кабельних та хвильових каналів, але, оскільки вірогідність $p_0 = 10^{-4}$ притаманна до більшості типів каналів, розглянемо кратність помилок саме для такої вірогідності. [3]

У каналах без пам'яті, тобто у каналах, у яких вірогідність появи символу на виході з каналу залежить тільки від символу на вході до каналу, дуже легко вирахувати вірогідність отримання будь-якої послідовності символів на виході, при заданій послідовності на вході, дуже легко можна підрахувати вірогідність перекручування n розрядів.[8]

$$\mathrm{P}_t = C_{\mathrm{n}}^{t} \cdot \mathrm{p}_0^{\mathrm{t}} \cdot (1 - p_0)^{\mathrm{n-t}}, \qquad (1.3)$$

де а $\mathrm{P_t}$ - це ймовірність помилки t-тої кратності в кодової комбінації, C_n^j – число поєднань n по t.

З причини того, що ми навели вірогідності перекручувань і знаємо їх, можна тепер проаналізувати саме у яких символах(розрядах) найчастіше трапляються помилки. Дана інформація зможе прояснити: кратність яких помилок є найбільш небезпечною для достовірності інформації. У джерелі [3] приводиться наступна статистика:

Таблиця 1.2

Статистика помилок залежно від довжини блоку

n	Кратність помилки (μ_n^t) у відсотках						$p_{пп}$
	1	2	3	4	5	6	
4	78	21	1	-	-	-	0,0016
9	56	32	9	3	-	-	0,00274
16	42	37	12	6	1	-	0,004
30	40	29	13	8	5	2	0,0063

де n – це загальна кількість розрядів у повідомленні, $p_{пп}$ – вірогідність помилкового прийому блоку даних розміром з *n (сума всіх вірогідностей помилок для* $\forall$ *розрядів у комбінації за формулою* (1.3), μ_n^t– помилка *t* при блоці розміром з *n*.

З цього можна зробити висновок, що абсолютна більшість помилок припадає саме на помилки 1,2-кратні, у середньому $> 80\%$.

Зведемо до графіку, який буде вказувати на вірогідність перекручувань, залежно від *n* та $p_{пп}$, де *p(μ)*- вірогідність помилки кратності *t.*

$$p(\mu) = p_{пп} * \mu_n^t \,, \tag{1.4}$$

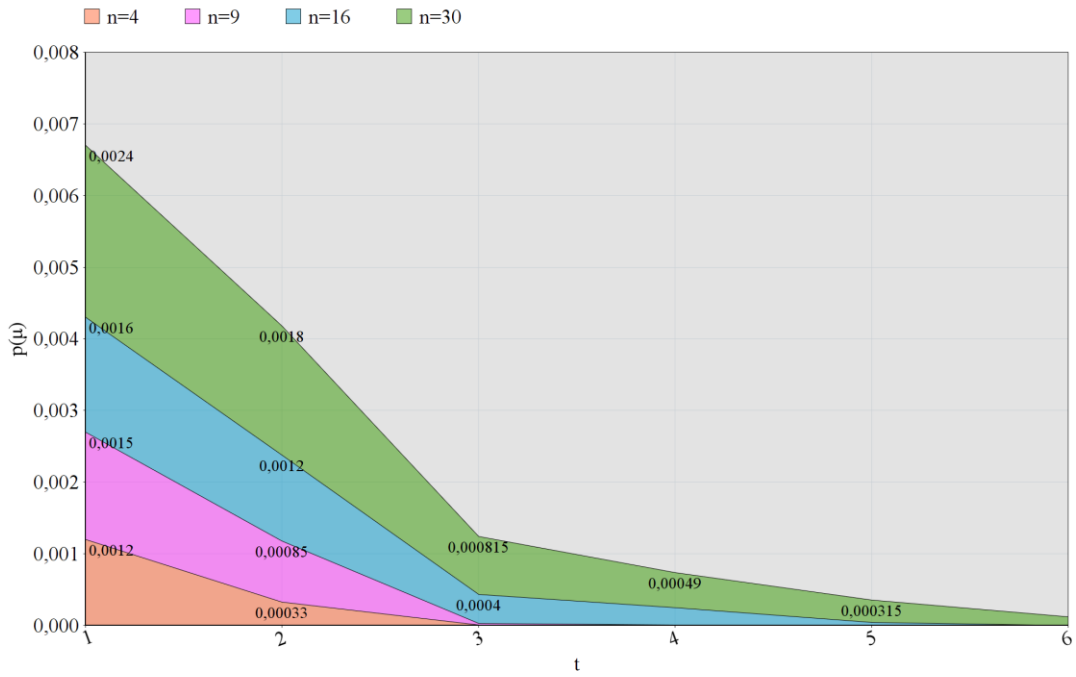

Рисунок 1.10 – Вірогідність перекручувань залежно від кратності помилки та вірогідності помилкового прийому

Як можна побачити з даного графіку, помилки кратності більше 2 настільки маловірогідні, що їх вірогідність, у рамках даної роботи, можна не враховувати, адже ураховуючи те, що дані були наведені для каналів зв'язку з $p_0 = 10^{-4}$, а назараз, згідно таблиці 1.1, сучасні канали зв'язку можуть гарантувати $p_0 = 10^{-6}$, не кажучи про канали, які можуть гарантувати $p_0 = від\ 10^{-8} до\ 10^{-10}$ [4], якщо мова іде про оптоволоконні канали передачі, вірогідність помилки кратності більше настільки маловірогідна, ($\approx$ 0,0000004 [4]) що боротьба з нею не цілесообразна.

Але, у даній роботі розглядаються ті канали, які мають $p_0 = від\ 10^{-5}\ до\ 10^{-6}$, тому що одними з основних типів каналів зв'язку, по яким передається більшість трафіку є кабельні [5,с.90] і хвилеводні канали [4].

Тому проблематика боротьби з однократними та двократними помилками є актуальною, а також потребує аналізу для покращення перешкодостійкості.

Також необхідно з’ясувати на якому діапазоні n необхідно концентруватись. З наступного рисунку (рис 1.11) видно, що p_{nn} зростає, залежно від n, у геометричній прогресії.

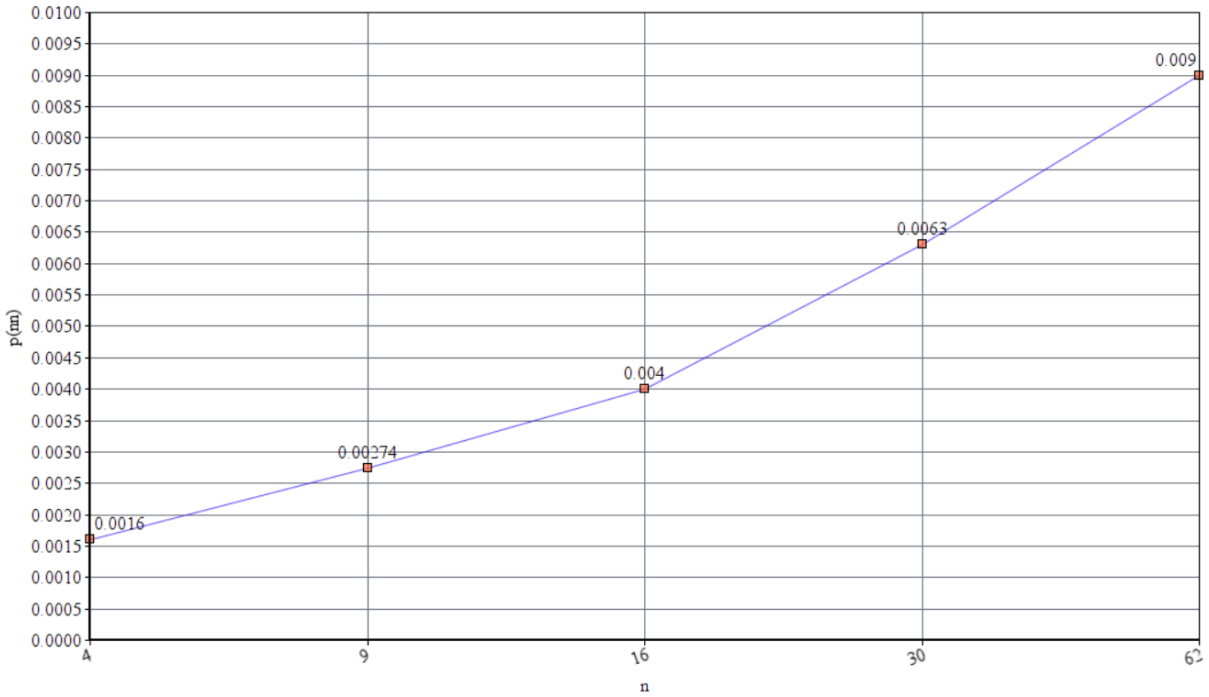

Рисунок 1.11 – Залежність $p_{пп}$ від n

З цього робиться висновок, що зручніше (з точки зору вірогідності помилкового прийому) буде оперувати саме невеликими блоками (кадрами) даних.

Існують багато методів боротьби з помилками при передачі даних. У розширеному дискретному каналі (дискретний канал + кодер + декодер каналу) виділяють наступні основні методи підвищення достовірності [7] :

1. Міри експлуатаційного і профілактичного характеру:
- Підвищення стабільності роботи генераторного приладдя,
- Резервування електроживлення,
- Виявлення і зміна приладдя з відмовами,
- Підвищення кваліфікації обслуговуючого персоналу.
2. Міри по підвищенню перешкодостійкості при передачі одиничних елементів:

- Підвищення відношення сигнал-завада (підвищення амплітуди і т.п.)
- Використання більш завадостійких методів модуляції,
- Покращення методів обробки,
- Вибір оптимальних сигналів
  (Хоча це не завжди можливо)

3. Використання зворотнього зв’язку.
4. Додавання надлишковості до коду, завадостійке кодування.

Безумовно, кожний із методів має місце у специфічних ситуаціях. Часто використовують одразу усі методи боротьби з помилками, але метою роботи буде фокусування саме на одному методі, для створення моделі, яка буде допомагати боротись за завадостійкість.

Міри експлуатаційного і профілактичного характеру мають велике значення при передачі повідомлень, але, як правило, вони потребують багато часу і значну кількість фінансів, адже заміна і подальша експлуатація приладдя займає більшу частину фінансових витримок на будь-якому великому проекті, а оскільки передача даних по дискретним каналам зв’язку – проект світового масштабу, економія ресурсів стає вкрай важливою.

Міри по підвищенню перешкодостійкості при передачі одиничних елементів мають схожу специфіку до мір експлуатаційного і профілактичного характеру у тому сенсі, хоча вони і дають досягти легшої демодуляції, та меншої вірогідність перекручувань, але при цьому є істотні недоліки: необхідність використання більш потужних модуляторів, якісніших каналів передачі, що також істотно потребує більших фінансових потужностей.

Використання зворотнього зв’язку. Серед плюсів: не треба ускладнювати систему та використовувати більш якісне обладнання; недоліки: суттєве збільшення витрати часу.

Використання завадостійких кодів. Серед плюсів: виправлення або виявлення перекручувань, нема необхідності використання більш якісних каналів зв’язку; недоліки: зниження інформативності повідомлень.

Зважуючи ці методи, темою даної роботи був обраний шлях боротьби за завадостійкість методом використання завадостійкого кодування.

Необхідно зазначити, що більшість локальних мереж використовують саме кабельні канали передачі зв’язку, тобто проблематика боротьби з однократними та двократними помилками потребує дослідження, задля створення моделі, яка може з ними боротись.

Модель, яка досліджується у дипломній роботі, має мати змогу адаптивно працювати, в тому числі, й з протоколом ATM, що буде означати, що розмірності її передаваємих кодових комбінацій має бути гнучкою і не перевищувати 48 байт[6], тобто, у бітовому еквіваленті, має бути кратною до 8.

Як вже ми дослідили, для невеликих і фіксованих кадрів, найбільш ймовірними є саме одноразові і дворазові помилки. На основі цих даних можна сформувати наступну задачу: необхідно проаналізувати та виконати дослідження методів підвищення завадостійкості у каналах передачі даних шляхом використання завадостійкого кодування.

Висновки до розділу 1

У розділі була розглянута предметна область джерел дискретних повідомлень, їх характеристики, наведені формули, на основі яких можна буде аналізувати роботу моделі у подальшому. Під час розглядання перепон, які стають на шляху достовірної передачі даних, виявлення їх причини, встановлено, що акцент необхідно робити саме на дискретних каналах зв’язку, складовою яких є безперервні канали, де самі помилки, безпосередньо, трапляються. Завдяки пошуку та аналізу статистики у каналах передачі даних, встановлено, що необхідно боротись з однократними та двократними помилками, а помилки більшої кратності є настільки маловірогідними, що було прийняте рішення їх можна не враховувати. Для конкретизації предметної області введено мережу, для якої має адаптивно працювати подальша модель, як приклад був використаний протокол ATM.

Були розглянуті та проаналізовані методи, які сприяють боротьбі з перешкодами, і завдяки своїм характеристикам був обраний шлях боротьби методом використання завадостійкого кодування, бо він є гнучким, прогнозованим та маловитратним. В результаті, задача на подальшу роботу поставлена.

Розділ 2.
АНАЛІЗ МОЖЛИВОСТІ ЗАВАДОСТІЙКОГО КОДУВАННЯ ПРИ ПІДВИЩЕННІ ДОСТОВІРНОСТІ В КАНАЛАХ ПЕРЕДАЧІ ДАНИХ

2.1 Характеристики завадостійких кодів

Побудова завадостійких (перешкодостійких) кодів можлива завдяки введенню надлишковості в код, а саме при застосуванні для передачі інформації коду, у якого в використанні знаходяться не всі доступні комбінації, а тільки деякі з них – коригувальні (надлишкові) коди.

Надлишковість коду також розділяється на абсолютну та відносну. Абсолютна надлишковість – кількість додаткових розрядів, які вводяться: $k = n - m$, де n – загальна кількість елементів у комбінації (довжина),m – загальна кількість інформаційних розрядів у комбінації.

Відносна надлишковість – вказує на ту частину загальної кількості символів кодової комбінації, яку складаються інформаційні символи, також цим терміном означають швидкість передачі інформації.

$$k_{відн} = \frac{k}{n} = \frac{(n-m)}{n} = 1 - \frac{m}{n}, \qquad (2.1)$$

Властивості коригування надлишкових кодів є залежними від кодування (побудови) самих кодів і їх параметрів (числа розрядів, надмірність, тривалість символів та ін.). Розглянемо загальні принципи по використанню надлишковості.

На вхід кодуючого пристрою надійшла деяка послідовність m інформаційних двійковий розрядів. На виході їй буде відповідати послідовність з n двійкових символів, де $n>m$. Усього може бути 2^m різних послідовностей з 2^n , які будуть називатись *дозволеними* кодовими комбінаціями. Інші $2^n - 2^m$ з можливих вихідних послідовностей для передачі не використовуються – вони будуть називатись *забороненими*.

Помилка при передачі інформації зводиться до того, що деякі з правильних передаточних символів перекручуються на інші – неправильні. Будь-яка із 2^m дозволених комбінацій може бути перекручена на будь-яку іншу. Усього таких случав перекручувань може бути $2^m * 2^n$. А саме до цього списку входять:

1. 2^m випадків безпомилкової передачі.

2. $2^m * (2^m - 1)$ випадків переводу в інші дозволені комбінації, що буде відповідати невиявляємим помилкам.

3. $2^m * (2^n - 2^m)$ випадків переходу у недозволені комбінації, зо буде відповідати виявляємим помилкам. Частина виявляємих помилкових кодових комбінацій від загального числа можливих випадків передачі дорівнює [8]:

$$К_{вия} = \frac{2^m * (2^n - 2^m)}{2^m * 2^n} = 1 - \frac{2^{2m}}{2^n}, \qquad (2.2)$$

Для прикладу, розглянемо виявляючу здатність коду, комбінація котрого містить чотири надлишкових символ ($n = k + 4$). У такому випадку, число вихідних послідовностей складає 2^{k+4}, що буде переважати загальне число кодуємих вхідних послідовностей у два рази. За підмножину дозволених кодових комбінацій можна прийняти, наприклад, підмножину 2^k комбінацій, яка містить парну кількість одиниць (або нулів). При кодуванні до кожної послідовності з k інформаційних символів додається один символ (0 або 1), такий, щоб кількість одиниць у кодовій комбінації була парною. Перекручування будь-якої парної кількості символів переводить дану

дозволену комбінацію у підмножину заборонених комбінацій, що виявляється на приймаючій стороні по непарності кількості одиниць. Частина виявлених помилок складає:

$$К_{вия} = 1 - \frac{2^{m}}{2^{n}} = 1 - \frac{2^{m}}{2^{m+4}} = 0{,}97875$$

Згідно сучасних вимог до достовірності інформації дана частина виявлених помилок ($\approx 98\%$) не є задовільною, але необхідно враховувати, що цей результат отриманий без використання перешкодостійкого кодування, а просто за рахунок введення надмірності. Використання завадостійких коригуючих кодів можуть значно покращити результат.

Для оцінки ступеню відмінності між двома довільними комбінаціями вихідного коду використовується характеристика, яка отримала назву відстані між комбінаціями.

Найменша відстань між дозволеними кодовими комбінаціями d_{min} – це дуже важлива характеристика коду, так як саме вона характеризує його коригувальні здібності.

Нехай необхідно побудувати код, який виявляє всі помилки кратністю t_m і нижче. Побудувати такий код – значить із множини 2^{n} можливих обрати 2^{m} дозволених комбінацій таким чином, щоб кожна з них у сумі за модулем два з будь-яким вектором помилок вагою $w_l \leq t_m$ не давала би у результаті будь-якої дозволеної комбінації. Для цього необхідно, щоб мінімальна кодова відстань задовольняла умові [3]:

$$d_{min} \geq t + 1\,, \qquad (2.3)$$

де *вага кодової комбінації w* – дорівнює кількості ненульових елементів у кодовій комбінації.

Кодова відстань – це ступінь різності двох будь-яких кодових комбінацій. Для того, щоб підрахувати кодову відстань, необхідно підсумувати кількість одиниць у сумі цих комбінацій за модулем 2.

Існує ціла низка завадостійких кодів. Вони різняться за своїми функціями, особливостями структури та фізичними властивостями. Не будемо розглядати окремо кожен вид, але зазначимо, що майже кожен з цих кодів має праве на існування у сьогоднішніх реаліях. Блок-схема з видами завадостійких кодів на рис.2.1 :

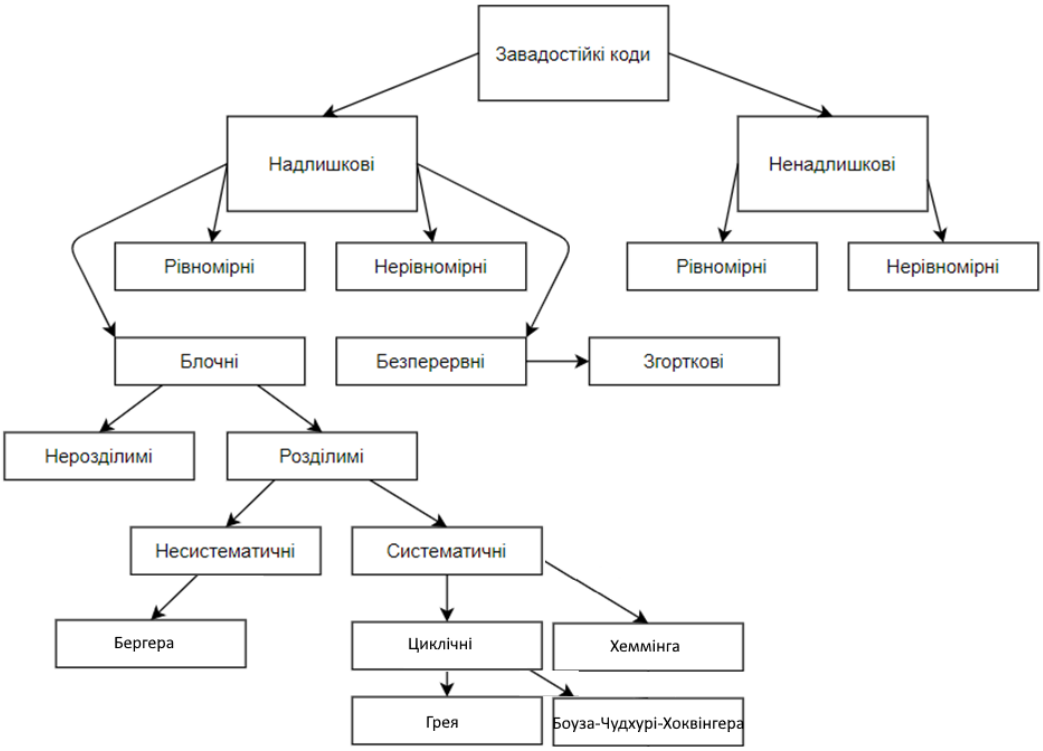

Рисунок 2.1 – Види завадостійких кодів

Згідно постановки задачі, необхідно розробити модель, що дозволить боротись із помилками кратності 1 та 2, тому надалі річ буде йти про надлишкові завадостійкі коди.

Згідно постановки задачі, а саме того, що код має мати однакову довжину для усіх повідомлень, було обрано *рівномірний код*, у якого n залишається незмінним для кожного повідомлення.

Завадостійкі коди розділяються на два великі класи: *блочні* та *безперервні* коди.

При кодуванні блочним кодом послідовність m елементів приймається за блок (повідомлення). Кожному блоку з m інформаційних символів у відповідність ставиться кодовий блок довжиною n. Код називається (n,m)-*кодом*, при цьому $n \geq m$. Кодовий блок у каналі зв'язку декодується незалежно від інших кодових блоків.

Відмінною особливістю безперервних кодів є те, що первинна послідовність символів, що несуть інформацію, безперервно перетворюється за певним законом в іншу послідовність, що містить надмірну кількість символів. Тут процеси кодування і декодування не вимагають поділу кодових символів на блоки.

З цих двох типів кодів було обрано блочні коди, адже до їх переваг можна віднести схожість процедур кодування та декодування, що спростить роботу моделі, а також блочні коди досить гнучкі у використанні, їх можна масштабувати, не змінюючи елементів у самій кодуючій (декодуючій) системі.

Блочні коди поділяють на розділимі та нерозділимі.

У розділимих кодах завжди можна виділити інформаційні символи, що містять інформацію, що передається, і контрольні (перевірочні) символи, які є надлишковими і служать виключно для корекції помилок.

Нерозділимі коди не мають чіткого поділу кодової комбінації на інформаційні та перевірочні символи.

Знову, з причини того, що чіткий поділ у повідомленні на інформаційні та перевірочні розряди облегшує роботу кодуючих та декодуючих елементів, у даній роботі будуть розглядатись розділимі коди.

Розділимі блокові коди розділяються на несистематичні і систематичні.

Несистематичні розділимі коди будуються таким чином, що перевірочні символи визначаються як сума підблоків довжини l, на які поділяється блок інформаційних символів.

Більшість відомих розділимих кодів складають систематичні коди. Значення перевірочних символів у систематичних кодах визначається в результаті проведення лінійних операцій над завчасно вибраними інформаційними символами. Для випадку двійкових кодів кожен перевірки символ вибирається таким, щоб його сума по модулю два з певними інформаційними символами дорівнювала нулю (перевірка на парність). Декодування зводиться до перевірки на парність певних груп символів.

Оскільки систематичні коди є простішими, аніж несистематичні, та дозволяють завчасно знати точне місцезнаходження по призначенню розрядів у комбінації, надалі у роботі будуть розглядатись саме вони.

2.2 Систематичні коди, матрична форма систематичних кодів

Ці коди відносяться до групи розділимих блокових кодів. Для систематичного коду сума елементів по модулю два двох дозволених комбінацій також дає дозволену комбінацію.

В теорії кодування широко використовується матричне подання кодів, з цієї причини розглядати процес кодування та декодування систематичних кодів ми також будемо з позиції матричного множення.[3]

Дозволені кодові комбінації мають задовольняти наступним вимогам:

1. Кодова відстань між будь-якими парами дозволених комбінацій не повинна бути меншою, ніж d_{min},

2. Кожна дозволена комбінація, як і будь-яка інша ненульова дозволена комбінація, повинна мати достатню кількість одиниць, не менш, ніж d_{min},

3. Усі дозволені комбінації мають бути лінійно незалежними, тобто жодна з них не може бути отримана шляхом сумування інших.

Усі дозволені комбінації можуть бути отримані із матриці, яка складається з n стовпчиків та m рядків.

$$P_{n,m} = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1m} & b_{11} & b_{12} & \cdots & b_{1p} \\ a_{21} & a_{22} & \cdots & a_{2m} & b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mm} & b_{m1} & b_{m2} & \cdots & b_{mp} \end{vmatrix}, \qquad (2.4)$$

У даній матриці перші m стовпчиків – інформаційні, а останні p – перевірочні. Таку матрицю $P_{n,m}$ називають *породжуючою*.

Породжуюча матриця $P_{n,m}$ може бути описана за допомогою двох під матриць: інформаційної – U_m та перевірочної – H_p.

$$U_m = \begin{vmatrix} a_{11} & a_{12} & \vdots & a_{1m} \\ a_{21} & a_{22} & \vdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \vdots & a_{mm} \end{vmatrix} H_p = \begin{vmatrix} b_{11} & b_{12} & \vdots & b_{1p} \\ b_{21} & b_{22} & \vdots & b_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \vdots & b_{mp} \end{vmatrix}, \qquad (2.5)$$

Для побудови образуючої матриці прийнято обирати інформаційну матрицю у вигляді квадратної одиничної матриці:

$$U_m = \begin{vmatrix} 1 & 0 & 0 & 0 & \vdots & 0 & 0 \\ 0 & 1 & 0 & 0 & \vdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \vdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \vdots & 1 & 0 \\ 0 & 0 & 0 & 0 & \vdots & 0 & 1 \end{vmatrix}, \qquad (2.6)$$

При цьому перевірочна підматриця H_p має будуватись відносно наступних вимог:

1. Кількість одиниць у рядку не має перевищувати $d_{min} - 1$.
2. Сума за модулем два двох будь-яких строк повинна мати не менш $d_{min} - 2$ одиниць.

Перевірочні символи формуються за рахунок лінійних операцій над інформаційними елементами. Для кожної кодової комбінації вони мають бути створені із k незалежних сум за модулем два.

Вибір інформаційних символів, які беруть участь у формуванні деякого перевірочного символу, залежить від вибору коду!

Перевірочні суми прийнято рахувати за допомогою перевірочної матриці H. Вона формується наступним чином: спочатку будується під матриця H^1, що є транспонованою по під матриці H_p

$$H^1 = \begin{vmatrix} b_{11} & b_{21} & \vdots & b_{m1} \\ b_{12} & b_{22} & \vdots & b_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ b_{1p} & b_{2p} & \vdots & b_{mp} \end{vmatrix}, \tag{2.7}$$

Після цього до неї дописується одинична матриця

$$P_{n,m} = \begin{vmatrix} b_{11} & b_{21} & \vdots & b_{m1} & 1 & 0 & 0 & \vdots & 0 & 0 \\ b_{12} & b_{22} & \vdots & b_{m2} & 0 & 1 & 0 & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{1p} & b_{2p} & \vdots & b_{mp} & 0 & 0 & 0 & \vdots & 0 & 1 \end{vmatrix}, \tag{2.8}$$

Алгоритм визначення перевірочних символів з інформаційних за допомогою матриці (2.8) наступний: позиції, що займаються одиницями в першому рядку підматриці H^1, визначають інформаційні розряди, які повинні брати участь у формуванні першого перевірочного розряду кодової комбінації. Позиції одиниць у другому рядку підматриці H^1 визначають інформаційні розряди, які беруть участь у формуванні другого перевірочного розряду і т. д.

Наведемо образуючу матрицю для коду (7,4). Дана розмірність обрана з причини її малогабаритності.

$$P_{7,4} = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{vmatrix}, \quad (2.9)$$

Тоді перевірочна підматриця буде:

$$H_p = \begin{vmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{vmatrix}, \quad (2.10)$$

Транспонована перевірочна підматриця буде:

$$H^1 = \begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{vmatrix}. \quad (2.11)$$

Додаючи до H^1 одиничну матрицю, отримуємо перевірочну матрицю:

$$H = \begin{vmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{vmatrix}, \quad (2.12)$$

Кодові комбінації повинні мати *k=n-m=7-4=3* перевірочних елементів.

Підматриця H^1 вказує на те, що перевірочні символи визначені наступними рівностями:

$b_1 = a_2 \oplus a_3 \oplus a_4$,

$b_2 = a_1 \oplus a_3 \oplus a_4$,

$b_3 = a_1 \oplus a_2 \oplus a_4$.

де $\oplus$ – сумування за модулем два.

Для формуємого повідомлення з комбінацією початковою $G(x) = [0\ 0\ 1\ 1]$ перевірочні символи будуть:

$b_1 = 0 \oplus 1 \oplus 1 = 0$,

$b_2 = 0 \oplus 1 \oplus 1 = 0,$

$b_3 = 0 \oplus 0 \oplus 1. = 0.$

Тоді синтезована комбінація $F(x)$ буде мати вигляд $F(x) =$ [0 0 1 1 0 0 1]

Перевірочна матриця *H* дуже зручна для визначення помилок у кодовій комбінації, а, отже, і виправлення їх. Перевірка кодових комбінацій при цьому виконується шляхом підсумовування по модулю два перевірочних символів кодових комбінацій і перевірочних символів, які були обчислені за прийнятими інформаційним символам. В результаті буде отримана сукупність контрольних рівностей, кожне з яких представляє суму по модулю два, одного з контрольних символів і певної кількості інформаційних елементів.

Склад контрольних рівностей легко визначається по перевірочній матриці *H*. До складу першої контрольної суми повинні входити символи, позиції яких зайняті одиницями в першому рядку матриці *H*. До складу другої контрольної суми повинні входити символи, позиції яких зайняті одиницями у другому рядку матриці *H* і т. д.

Для наведеного вище коду (7,4) рівності мають вигляд:

$S_1 = a_2 \oplus a_3 \oplus a_4 \oplus b_1,$

$S_2 = a_1 \oplus a_3 \oplus a_4 \oplus b_2,$

$S_3 = a_1 \oplus a_2 \oplus a_4 \oplus b_3.$

У результаті таких *k* перевірок буде отримане *k*-розрядне двійкове число (синдром), яке буде дорівнювати нулю за відсутності помилок, і буде відмінним від нуля при наявності помилок.

Якщо код призначений для виправлення помилок, то має бути заздалегідь визначена відповідність між видом синдрому і видом помилки, яку необхідно виправити.

Нехай в розглянутому нами прикладі з кодом (7,4) сталася помилка в першому розряді кодової комбінації (спотворений символ a_1). Утворення

помилки також уявляють у вигляді додавання до передаваємої комбінації вектору помилки $\theta(x)$. Тобто отримана комбінація $F'(x) = F(x) + \theta(x)$.

У даному випадку, $\theta(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1]$. Тоді перевірочні операції дадуть наступний результат:

$S_1 = 0, S_1 = 1, S_1 = 0.$

Таким чином, при помилці в першому розряді кодової комбінації буде отримано синдром 010. У разі помилки в другому розряді буде отримано синдром 101 і т.д.

Якщо для виправлення одноразових помилок в кодових комбінаціях отримати синдроми досить просто, то для виправлення дворазових, триразових і т.д. помилок, а також для виправлення пачок помилок побудова синдромів є досить складним завданням.

Окремо відмітимо метод (код), коли до комбінації додається лише один надлишковий символ. Вибирається надлишковий символ таким чином, щоб загальна кількість одиниць в кодової комбінації була парною. Перевірка кодової комбінації виконується шляхом підсумовування по модулю два всіх його символів. Надлишковість даного коду дорівнює $\frac{k}{n} = \frac{1}{n}$.

Така надлишковість, залежно від довжини повідомлення, є низькою, а перевага використання даного коду зростає паралельно зі зростанням довжини повідомлення, адже кількість додаткових розрядів зажди буде становити один розряд .

Код дозволяє виявляти одноразові помилки і всі помилки непарної кратності, так як тільки в цих випадках кількість одиниць в комбінації стане непарною. Не виявляються помилки парної кратності.

2.3 Коди Хеммінга

Коди Хеммінга є одними з простіших представників блочних систематичних кодів. Принцип їх формування аналогічний загальному принципу побудови систематичних кодів. Основною ідеєю кодів Хеммінга є

чітке використання певних розрядів у комбінації, задля досягнення можливості виправляти однократну помилку.

Код Хеммінга, що забезпечує виправлення всіх одноразових (однократних) помилок, повинен мати мінімальну кодове відстань d_{min} = 3. Довжина коду n вибирається з умови: $2^m \leq \frac{2^n}{1+n}$.

Код будується таким чином, щоб в результаті $\rho = n - k$ перевірок отримати ρ-розрядне двійкове число, що вказує номер спотвореної позиції в кодової комбінації. Для цього перевірочні символи повинні перебувати на тих номерах позицій, які висловлюються ступенем двійки $(2^0, 2^1 2^2, \ldots, 2^{k-1})$, (так як кожен з них входить тільки в одне з перевірочних рівнянь), таким чином, якщо нумерувати позиції зліва направо, то контрольні символи повинні знаходитися на першій, другій, четвертій і т.д. позиціях.

Результат першої перевірки дає цифру молодшого розряду синдрому в двійковому записі. Якщо результат цієї перевірки дасть 1, то один із символів перевіреної групи спотворений. Таким чином, першою перевіркою повинні бути охоплені символи з номерами, що містять в двійковій формі запису одиниці в першому розряді: 1, 3, 5, 7, 9 і т.д.

Результат другої перевірки дає цифру другого розряду синдрому. Отже, другою перевіркою повинні бути охоплені символи з номерами, що містять в двійковій формі запису одиниці в другому розряді: 2, 3, 6, 7, 10 і т.д.

Аналогічно, при третій перевірці повинні перевірятися символи, номера яких в двійковій формі запису містять одиниці в третьому розряді: 4, 5, 6, 7, 12 і т.д.

У таблиці 2.1 наведений загальний принцип розташування перевіряючих елементів і групи інформаційних символів, за які вони відповідають, тобто складають перевірочну матрицю.

Таблиця 2.1

Принцип розташування розрядів у кодах Хеммінга

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	…
$F(x)$	$П_1$	$П_2$	I_1	$П_3$	I_2	I_3	I_4	$П_4$	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	$П_5$	I_{12}	I_{13}	I_{14}	…
S_5																X	X	X	X	…
S_4								X	X	X	X	X	X	X	X					…
S_3				X	X	X	X					X	X	X	X					…
S_2		X	X			X	X			X	X			X	X			X	X	…
S_1	X		X		X		X		X		X		X		X		X		X	…

Де № - номер розряду у комбінації, $F(x)$ вказує на місця розрядів за призначенням (де $I_i -$ *інформаційні розряди,* $П_i -$ *перевірочні розряди*), зеленим кольором позначені перевірочні розряди, S_i – розряди , за які відповідає при перевірці і-й перевірочний розряд.

Таким чином, перевірочні групи повинні мати вигляд:

$S_1 = K_1 \oplus K_3 \oplus K_5 \oplus K_7 \oplus K_9 \oplus$ ...,

$S_2 = K_2 \oplus K_3 \oplus K_6 \oplus K_7 \oplus K_{10} \oplus$ ...,

$S_3 = K_4 \oplus K_5 \oplus K_6 \oplus K_7 \oplus K_{12} \oplus$ ...,

$S_4 = K_8 \oplus K_9 \oplus K_{10} \oplus K_{11} \oplus K_{12} \oplus$ ....

Перевірочна матриця коду повинна мати n стовпців і k рядків. Кожен стовпець повинен складати двійкову комбінацію і вказувати номер відповідної позиції коду.

При n =15, m = 11, k = 4, H буде мати наступний вигляд:

$$H = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{vmatrix}, \quad (2.13)$$

Як можна побачити, побудова H подібна таблиці (2.1). Формування даного коду є дуже простим, чітким, що полегшує роботу з ним.

2.4 Циклічні коди

Циклічні коди отримали досить широке застосування завдяки їх ефективності при виявленні і виправленні помилок. Схеми кодуючих і декодуючих пристроїв для цих кодів надзвичайно прості і будуються на основі звичайних регістрів зсуву.

Назва кодів виникла від їх властивостей, які полягають в тому, що кожна нова кодова комбінація може бути отримана шляхом циклічної перестановки символів комбінації, відноситься до цього ж коду. Це означає, якщо, наприклад, комбінація $a_0, a_1, a_2, \cdots, a_{n-1}$ є дозволеною комбінацією циклічного коду, то комбінація $a_{n-2}, a_0, a_1, a_2, \cdots, a_{n-2}$ також належить даному коду.

Циклічні коди зручно розглядати, представляючи комбінацію двійкового коду не у вигляді послідовностей нулів і одиниць, а у вигляді полінома від фіктивної змінної x, а саме (2.14) [2]:

$$G(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0, \qquad (2.14)$$

де a_i - цифри даної системи рахунку (у нашому випадку двійкової системи)
Тобто комбінацію [1 0 1 1 0 1 1] уявляють наступним виглядом:

$$\sigma(x) = 1 * x^6 + 0 * x^5 + 1 * x^4 + 1 * x^3 + 0 * x^2 + 1 * x^1 + 1 * x^0, \quad (2.15)$$

Найбільша ступінь у розкладі (2.15) називається *ступенем поліному*. Подання кодових комбінацій в формі (2.15) дозволяє звести дії над комбінаціями до дії над многочленами. При цьому додавання двійкових многочленів зводиться до складання по модулю два коефіцієнтів при рівних ступенях змінної *x*; множення виконується за звичайним правилом множення ступеневих функцій, проте отримані при цьому коефіцієнти при рівних ступенях змінної *x* додаються по модулю два; ділення здійснюється за

правилами ділення ступеневих функцій, при цьому операції віднімання замінюються операціями підсумовування по модулю два.

Подання комбінацій в формах (2.14) і (2.15) зручне ще й тим, що згадана циклічна перестановка є результатом простого множення даного полінома на *x*. Дійсно, якщо одна з кодових комбінацій виражається поліномом

$V(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$, то нова комбінація, яка утворена циклічним зсувом $x * V(x) = a_0x + a_1x^2 + a_2x^3 + \cdots + a_{n-2}x^{n-1}$ також належить коду.

Хоча у останньому члені необхідно замінити x^n на 1. Таким чином, нова комбінація буде: $V_1(x) = a_{n-1} + a_0x + a_1x^2 + \cdots + a_{n-2}x^{n-1}$.

Наприклад, циклічний зсув кодової комбінації $G(x) = [1\ 0\ 1\ 0\ 1\ 0\ 1]$ може бути отримано шляхом множення поліному (2.15) на *x*: $G(x) * x = x^7 + x^5 + x^3 + x$.

Змінивши x^7 ,отримуємо поліном: $G_1(x) * x = x^5 + x^3 + x + 1$.

що дорівнює комбінації $G_1(x) = [0\ 1\ 0\ 1\ 0\ 1\ 1]$.

Для побудови образуючої матриці $P_{n,m}$ циклічного коду достатньо обрати початкову *n*-розрядну комбінацію $V_1(x)$. Циклічним зсувом можливо отримати *(n – 1)* різних комбінацій, з яких *m* комбінацій можуть бути обрані як використовуємі.

Підраховуючи рядки образуючої матриці у всіх можливих комбінаціях, можливо отримати інші кодові комбінації. Можна показати, що кодові комбінації, які отримуються з деякої комбінації $V_1(x)$ циклічним зсувом, задовольняють умовам, які висуваються до совокупності початкових комбінацій.

Циклічний зсув комбінації з одиницею у старшому *n*-му розряді рівнозначний множенню відповідного багаточлена на *x* з одночасним вираховуванням із результату багаточлена (*x*-1) або ($x^n + 1$), адже операції проводяться за модулем два. Тобто, якщо за початковий взяти деякий

поліном *P(x)*, процес отримання базових поліномів можна отримати у наступному вигляді:

$U_1(x) = P(x)$,

$U_2(x) = P(x)x - C_2(xn + 1)$,

$U_3(x) = P(x)x^2 - C_3(x^n + 1)$,

……

$U_n(x) = P(x)x^{n-1} - C_n(x^n + 1)$,

де $C_2, C_3, \cdots, C_n$- коефіцієнти, які приймають значення 1 при $P(x) * x^i \geq (x^n - -1)$, і значення 0 при $P(x) * x^i < (x^n - 1)$.

При такому способі утворення базових поліномів поліном $P(x)$ називають *образуючим*.

Якщо прийняти умову, за якої $P(x)$ є дільником двочлена $(x^n + 1)$, то базові комбінації, а з ними усі дозволені комбінації коду, отримують властивість ділимості на $P(x)$. З цього слідує, що належність кодової комбінації до групи дозволених, легко перевіряється діленням її полінома на образуючий поліном $P(x)$. Якщо залишок від ділення дорівнює нулю, то комбінація є *дозволеною*.

Дану властивість циклічного коду використовують для виявлення чи виправлення помилок. Тобто, якщо під впливом завад дозволена комбінація змінюється на заборонену, помилка може бути виявлена за наявності залишку при діленні комбінації на образуючий поліном $P(x)$.

Таким чином, образуючий поліном $P(x)$ повинен задовільняти основній вимозі – має бути дільником двочлена $(x^n + 1)$. Вибір $P(x)$ беззаперечно визначає циклічний код і його корегуючі властивості.

Циклічний (*n, m*)-код може бути отриманим шляхом множення його *m*-значного коду, який виражений у вигляді поліному (m - 1), на деякий образуючий поліном $P(x)$ у ступені (*n* – *m*).

Можливий і інший спосіб отримання циклічного коду. Для цього кодова комбінація простого *m*-значного коду *G(x)* помножається на одночлен x^{n-m},

а потім ділиться на образуючий поліном $P(x)$ у ступені (n – m). В результаті множення *G(x)* на x^{n-m} ступінь кожного одночлена, який входить до *G(x)*, збільшиться на (n – m). При діленні добутку $x^{n-m}G(x)$ на образуючий поліном $P(x)$ утворюється залишок *Q(x)*, того ж ступеню, як *G(x)*.

Результат множення та ділення можливо уявити у наступному вигляді:

$$\frac{x^{n-m}G(x)}{P(x)} = Q(x) + \frac{R(x)}{P(x)}, \tag{2.16}$$

де *R(x)* – залишок від ділення $x^{n-m}G(x)$ на $P(x)$.

Оскільки залишок *Q(x)* має таку ж ступінь, як і кодова комбінація *G(x)*, *Q(x)* також є дозволеною комбінацією простого m-значного коду.

Помноживши дві частини рівності (2.16) на $P(x)$, а також зробивши певні перестанови у формулі, отримуємо:

$$F(x) = Q(x)P(x) = x^{n-m}G(x) + R(x), \tag{2.17}$$

У правій частині виразу (2.17) знак «мінус» перед $R(x)$ замінено знаком «плюс», адже віднімання за модулем два зводиться до підсумовування.

Таким чином, кодова комбінація циклічного коду (n, m) може бути отриманою двома різними шляхами:

1. Шляхом множення простої кодової комбінації на ступеню (m-1) на одночлен x^{n-m} і додавання до цього добутку залишку, який було отримано від множення на образуючий поліном $P(x)$ ступеню(n-m).
2. Шляхом множення простої кодової комбінації ступеню (m-1) на образуючий поліном $P(x)$ ступеню(n-m).

При першому способі кодування перші m символів отриманої кодової комбінації збігаються з відповідними символами вихідної кодової комбінації простого коду.

При другому способі в отриманої кодової комбінації інформаційні символи не завжди збігаються з символами вихідної комбінації. Таким чином, можна легко реалізувати безліч кодових слів.

У зв'язку з цим на практиці зазвичай використовується перший спосіб отримання циклічного коду.

Матричне уявлення циклічних кодів.[3] Для формування рядків образуючої матриці по першим способом утворення циклічного коду приймають комбінації простого *m*-розрядного коду *G(x),* які містять одиницю хоча б у одному розряді. Дані комбінації помножують на x^{n-m}, вираховується залишок $R(x)$ від ділення отриманого добутку $x^{n-m}G(x)$ на образуючий поліном і записується відповідний рядок матриці у вигляді суми добутку $x^{n-m}G(x)$ і $R(x)$. При цьому образуюча матриця $P_{n,m}$ представляється двома під матрицями – інформаційною U_m та додатковою

$$H_p: P_{n,m} = \left|U_m, H_p\right|, \tag{2.18}$$

Інформаційна підматриця U_m – квадратна одинична матриця розмірністю $m \times m$. Додаткова підматриця H_p розмірністю $k \times m$ і формується залишками $R(x)$.

Образуюча матриця дозволяє отримати m базових комбінацій коду. Інші комбінації утворюються підсумовуванням за модулем два рядків добутку цієї матриці у всіх можливих поєднаннях.

Щоб виявити місце помилки у повідомленні, необхідно проаналізувати остачу ділення комбінації на похідний многочлен. Оскільки циклічні коди задаються перевірочними і породжуючи ми матрицями, а вигляд даних матриць задається образуючим багаточленом *P(x)*, кожен стовпчик канонічної форми перевірочної матриці можливо виявити шляхом знаходження залишку від ділення одночлену на $x^i (0 \le i \le n-1)$ на багаточлен *P(x)*. Якщо *n=7*, $P(x) = x^3 + x + 1$ проаналізуємо будування перевірочної матриці.

$$h_1 = x^0 : (x^3 + x + 1) \rightarrow R(x) = x^0 \rightarrow \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix},$$

$$h_2 = x^1 : (x^3 + x + 1) \rightarrow R(x) = x^1 \rightarrow \begin{vmatrix} 0 \\ 1 \\ 0 \end{vmatrix},$$

$$h_3 = x^2 : (x^3 + x + 1) \rightarrow R(x) = x^2 \rightarrow \begin{vmatrix} 0 \\ 0 \\ 1 \end{vmatrix},$$

$$h_4 = x^3 : (x^3 + x + 1) \rightarrow R(x) = x + 1 \rightarrow \begin{vmatrix} 1 \\ 1 \\ 0 \end{vmatrix},$$

$$h_5 = x^4 : (x^3 + x + 1) \rightarrow R(x) = x^2 + x \rightarrow \begin{vmatrix} 0 \\ 1 \\ 1 \end{vmatrix},$$

$$h_6 = x^5 : (x^3 + x + 1) \rightarrow R(x) = x^2 + x + 1 \rightarrow \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix},$$

$$h_7 = x^6 : (x^3 + x + 1) \rightarrow R(x) = x^2 + 1 \rightarrow \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix}.$$

Це означає, що для кожної позиції помилки синдром помилки буде унікальним, що дозволить її виправити.

Тобто, перевірочна матриця H(7,4) буде мати вигляд

$$H(7,4) = |h_1\, h_2\, h_3\, h_4\, h_5\, h_6\, h_7\ | = \begin{vmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{vmatrix}, \qquad (2.19)$$

2.5 Каскадні коди, (турбокоди)

Каскадними кодами (або турбокодами) називають ті коди, які складаються з двох або більше, менших за розміром і більш простих кодів. Тобто головна ідея полягає у тому, щоб одноразово закодувати повідомлення у кодері 1, а після це закодоване повідомлення кодується поверх кодером 2 і

так далі. При декодуванні процес декодування виконується у зворотному порядку. Можливість провести процес декодування повторно робить каскадні коди дуже зручними. Однією з ідей також є те, що другий декодер зберігає у собі адреси помилок, які зостались від першого, а при зверненні до першого декодера, у ньому зберігаються адреси помилок другого, що значить, що ці декодери пов'язані і збірні. Але більша достовірність інформації також свідчить про те, що при ланцюговому кодуванні і декодуванні більш ніж одним збірним кодуючим або декодуючим елементом, також буде використовуватись значно більше часу, тому фінальна потужність буде знаходитись від залежності від кількості і складності ітерацій.

Для розуміння припустимо, що в нас є комбінація з довжиною mM та основою коду q. Після цього вона розбивається на M блоків по m символів. Кожен блок M розглядається як символ коду за основи q^m. В такому разі послідовність з M таких символів можливо закодувати (N,M) кодом над полем GF(q^m). N = M+K, K — число q^m-х символів. Кожний із N символів кодується (n,m) кодом з основою q. Код (n,m) – внутрішній, а код (N+M) – зовнішній, див. рис.2.2. [8]

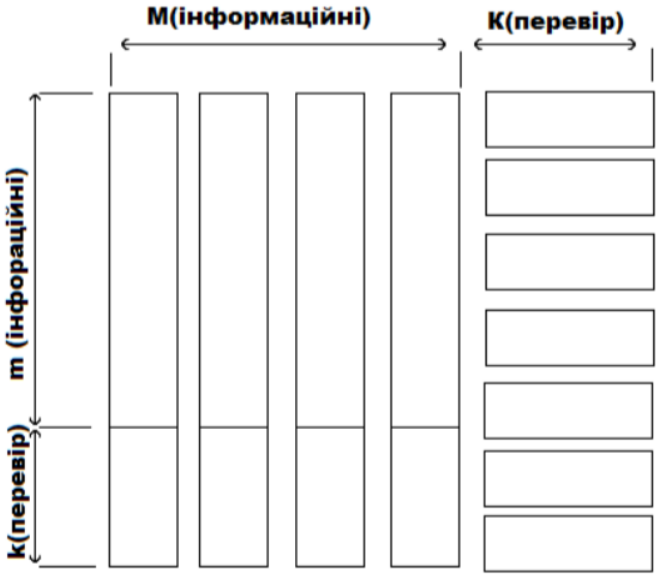

Рисунок 2.2 – Загальна схема каскадного коду.

Висновки до розділу 2

В розділі проаналізовані можливості завадостійкого кодування при підвищенні достовірності в каналах передачі даних. В результаті аналізу вибрані задовольняючі завданню коди, а саме блочні систематичні коди - а також були пояснені принципи їх побудови, кодування та декодування. Згідно постановці задачі обирались саме ті типи завадостійких кодів, який дозволить ефективно розробляти модель, і було обрано за основу використання саме систематичних кодів, де складовою частиною, як приклад, будуть виступати коди циклічні та Хеммінга, які, задля уніфікації, будуть формуватись матричним способом. Задля задовільнення вимог, висунута пропозиція по використанню каскадного типу кодування, яке зможе залишити процес кодування та декодування простим, а також зможе розширити можливості базового коду (коду першого ступеню) – коду, який тільки може виправляти однократні помилки. Під розширенням можливостей мається на увазі те, що каскадний код що дозволить не тільки виправляти однократні помилки, а також додасть можливість виявляти помилки парної кратності.

Розділ 3.

РОЗРОБКА І ДОСЛІДЖЕННЯ МОДЕЛІ

3.1 Модель з використанням каскадного кодування

У основі ідеї полягає використання каскадного коду, другим ступенем якого буде використання біту перевірки на парність. На прикладі циклічного коду і коду Хеммінга можна навести окремий приклад використання такого коду, але дана модель може застосовуватись і до інших кодів.

Перевагами даного підходу будуть: загальна простота коду; швидкодійність, бо на формування такого каскаду не будуть витрачатись великі ресурси, які могли б бути задіяні при формуванні другого каскаду

повноцінними кодами; зручність у трансляції по каналам передачі, адже основні розмірності блокових кодів, що мають змогу виправляти однократні помилки є наступними (7,4),(15,11)(31,26), тобто для утворення повноцінно рівномірної байтової структури (1 байт = 8 біт, тобто n має бути кратною до 8) завжди не вистачає одного біту; і, що саме важливе, використання такого підходу надасть змогу боротись з усіма основними видами помилок, які трапляються у каналах передачі даних, тобто з однократними та двократними помилками.

Як вже було відмічено, циклічні коди, коди Хеммінга здатні виявляти і виправляти однократні помилки, а при помилках більшої кратності навіть виявленню помилка вже не підлягає, що погано відображається на достовірності інформації.

Але при додаванні до коду каскаду, а саме біту перевірки (ϑ) на парність – з'являється можливість виявити двократні помилки і надіслати запит до відправника інформації про повторну передачу закодованого повідомлення. Тобто каскадна комбінація $J(x)$ буде мати вигляд:

$$J(x) = F(x) + \vartheta\,, \tag{3.1}$$

що вигляді породжуючої таблиці (для коду Хеммінга(16,11)) буде мати наступний вигляд (таблиця 3.1):

Таблиця 3.1

Розташування розрядів за призначенням у каскадному коді з кодом Хеммінга

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$J(x)$	$П_1$	$П_2$	I_1	$П_3$	I_2	I_3	I_4	$П_4$	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	$П_5$
S_5	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
S_4								X	X	X	X	X	X	X	X	
S_3				X	X	X	X					X	X	X	X	
S_2		X	X			X	X			X	X			X	X	
S_1	X		X		X		X		X		X		X		X	

Де $П_5 = \vartheta$ – біт перевірки на парність, а червоним кольором помічено його місце розташування у комбінації, $J(x)$ вказує на місця розрядів за призначенням (де $I_i -$ *інформаційні розряди,* $П_i -$ *перевірочні розряди*), зеленим кольором позначені перевірочні розряди, S_i – розряди , за які відповідає при перевірці і-й перевірочний розряд.

У вигляді породжуючих рівнянь система виглядає наступним чином:

$$П_1 = I_1 \oplus I_2 \oplus I_4 \oplus I_5 \oplus I_7 \oplus I_9 \oplus I_{11},$$

$$П_2 = I_1 \oplus I_3 \oplus I_4 \oplus I_6 \oplus I_7 \oplus I_{10} \oplus I_{11},$$

$$П_3 = I_2 \oplus I_3 \oplus I_4 \oplus I_8 \oplus I_9 \oplus I_{10} \oplus I_{11},$$

$$П_4 = I_5 \oplus I_6 \oplus I_7 \oplus I_8 \oplus I_9 \oplus I_{10} \oplus I_{11},$$

$$П_5 = П_1 \oplus П_2 \oplus I_1 \oplus П_3 \oplus I_2 \oplus I_3 \oplus I_4 \oplus П_4 \oplus I_5 \oplus I_6 \oplus I_7 \oplus I_8 \oplus I_9 \oplus I_{10} \oplus I_{11}.$$

У вигляді таблиці (для циклічного коду(16,11)) дана схема має наступний вигляд (таблиця 3.2):

Таблиця 3.2

Розташування розрядів за призначенням у каскадному коді з циклічним кодом

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$J(x)$	$П_1$	$П_2$	$П_3$	$П_4$	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	$П_5$
ϑ	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Де $П_i = R(x), П_5 = \vartheta$, а $I_i = G(x)x^4$

3.2 Алгоритм використання систематичних кодів з додаванням каскаду перевірки на парність

Схема кодування каскадним кодом (з додаванням біту перевірки на парність) виглядає наступним простим способом:

Рисунок 3.1 – Загальна схема кодування і декодування каскадним кодом

Від джерела приймається інформаційна комбінація, яка кодується кодом першого ступеню, кодування першого ступеню розглядалось у розділах 2.2, 2.3, 2.4 (код Хеммінга, циклічний код, тощо). Після кодування першого ступеню йде кодування другого ступеню, тобто до передаваємої комбінації $F(x)$ додається біт перевірки на парність (ϑ), в результаті отримується каскадна комбінація :$J(x) = F(x) + \vartheta$

У вигляді блок-схеми це виглядає наступним чином:

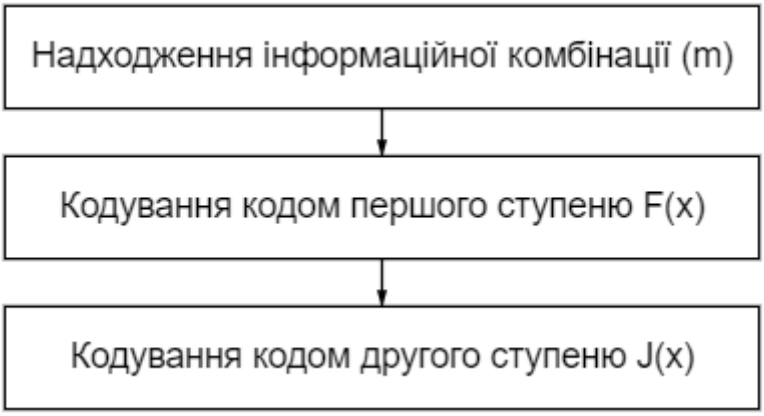

Рисунок 3.2 – Загальна схема кодування

На прикладі коду Хеммінга (16,11), $J(x)$ виглядає наступним чином:

П1	П2	І1	П3	І2	І3	І4	П4	І5	І6	І7	І8	І9	І10	І11	П5
1	1	0	1	0	0	0	1	0	0	0	0	0	0	1	1

Рисунок 3.3 – Додавання до коду Хеммінга (15,11) біту перевірки на парність

Де $П_5$ – біт перевірки на парність

Оскільки циклічний код(16,11) має подібну вихідну структуру до коду Хеммінга, у циклічного коду додавання біту парності має таку ж саму схему, тобто $J(x)$ буде мати вигляд:

І1	І2	І3	І4	І5	І6	І7	І8	І9	І10	І11	П1	П2	П3	П4	П5
1	1	0	1	0	0	0	1	0	0	0	0	0	0	1	1

Рисунок 3.4 – Додавання до циклічного коду (15,11) біту перевірки на парність

Саме такі комбінації і пересилаються по каналам передачі даних.

Схема при декодуванні виглядає дещо складніше, ніж схема кодування. На прикладі коду Хеммінга (16,11) та циклічного коду (16,11), з причини їх схожих властивостей, вона має наступний вигляд (рис 3.5):

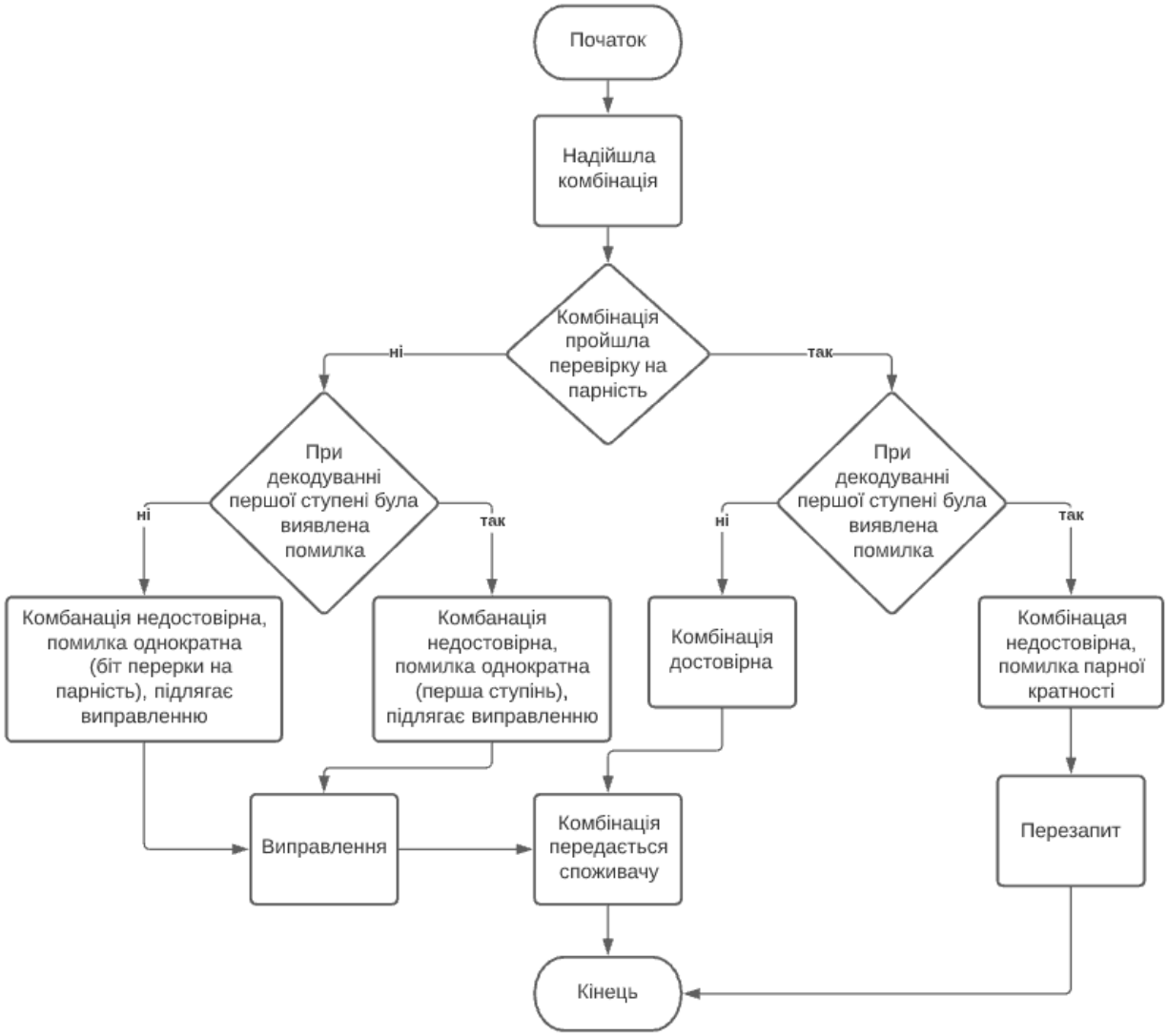

Рисунок 3.5 – Загальна схема декодування

де:

- «*Надійшла комбінація*» – до збірного декодеру надійшла комбінація $J'(x)$ з каналу передачі даних;

- «*Комбінація пройшла перевірку на парність*» – в результаті перевірки на парність (декодер другого ступеню), шляхом підсумовування усіх розрядів комбінації за модулем 2, не було виявлено помилки, що свідчить про

відсутність однократної помилки, яка могла би парність порушити. Де «так» – перевірка виявила помилку, «ні» – перевірка помилку не виявила;

- «*При декодуванні першої ступені була виявлена помилка*» – в результаті декодування коду першого ступеню була виявлена помилка. На прикладі коду Хеммінга мається на увазі випадок, де в результаті взаємодії отриманої комбінації з перевірочною матрицею, було отримано результат не $S_i = [0_1\ 0_2\ ...\ 0_i]$, що свідчить про наявність помилки і вказує на номер розряду, у якому виникла помилка. На прикладі циклічного коду мається на увазі випадок, де в результаті взаємодії комбінації з перевірочною матрицею (або діленні на породжуючий поліном *P(x)*), був виявлений залишок не $R_i(x) = [0_1\ 0_2\ ...\ 0_i]$, що свідчить про наявність помилки і унікальний синдром залишку вказує на номер розряду, у якому виникла помилка. Де «так» – перевірка виявила помилку, «ні» – перевірка помилку не виявила;

- «*Комбанація недостовірна, помилка однократна (біт перерки на парність), підлягає виправленню*» – з причини того, що отримана комбінація не пройшла перевірку на парність, що свідчить про наявність непарної помилки (однократної), а при декодуванні першого ступеню помилки виявлено не було, робиться висновок про помилку у розряді перевірки на парність $J'(x) \neq J(x)$, що не має значення для достовірності закодованого першим каскадом повідомлення.

- «*Комбанація недостовірна, помилка однократна (перша ступінь), підлягає виправленню*» – з причини того, що отримана комбінація не пройшла перевірку на парність, що свідчить про наявність непарної помилки (однократної), і при декодуванні першого ступеню була виявлена помилка $J'(x) \neq J(x)$, робиться висновок про помилку у комбінації першого ступеня.

- «*Виправлення*» – у випадку, коли помилка міститься у біті перевірки на парність, просто відкидання такого біту і декодування комбінації першого ступеню. У випадку, коли помилка міститься у комбінації першого ступеню, декодування комбінації і виправлення інформації, згідно алгоритмів коду;

- «*Комбінація достовірна*» – з причини того, що отримана комбінація пройшла перевірку на парність, а при декодуванні першого ступеню помилки виявлено не було, робиться висновок про відсутність помилки $J'(x) = J(x)$, комбінація декодується без виправлень;

- «*Комбінація передається споживачу*» – достовірна (виправлена) комбінація передається безпосередньо споживачу;

- «*Комбанація недостовірна, помилка парної кратності*» – з причини того, що отримана комбінація пройшла перевірку на парність, а при декодуванні першого ступеню була виявлена помилка, робиться висновок про помилку парної кратності $J'(x) \neq J(x)$, яка не підлягає виправленню .

- «*Перезапит*» – з причини неможливості виправити повідомлення, до відправника надсилається запит про повторне відправлення останнього повідомлення;

3.3 Дослідження підмоделі з використанням коду Хеммінга

До кодеру надійшла комбінація *G(x) = [11000000001]*. У кодовій комбінації розташовуватись комбінація буде наступним чином:

Таблиця 3.3

Фактичне розташування інформаційних розрядів у підмоделі з кодом Хеммінга

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$F(x)$	$П_1$	$П_2$	*1*	$П_3$	*1*	*0*	*0*	$П_4$	*0*	*0*	*0*	*0*	*0*	*0*	*1*

Після цього комбінація кодується за допомогою породжуючої матриці $P_{15,11}$, яка подібна і будується за тим же принципом, що й породжуюча матриця $P_{7,4}$, яка формувалась при розгляді принципу кодування систематичних кодів.

Згідно таблиці 3.1 процес кодування можна також уявити у вигляді наступної системи породжуючих рівнянь:

$П_1 = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1,$

$П_2 = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1,$

$П_3 = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1,$

$П_4 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1.$

В результаті, була отримана комбінація першого ступеню $F(x) = [1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1]$.

При додаванні біту перевірки на парність комбінація буде мати наступний вигляд:

$$\vartheta = П_5 = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1.$$

Таблиця 3.4

Фактичне розташування всіх розрядів у підмоделі з кодом Хеммінга з додаванням біту перевірки на парність

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$J(x)$	1	1	1	0	1	0	0	1	0	0	0	0	0	0	1	0

Тобто комбінація, що буде передаватись по дискретному каналу буде мати вигляд : $J(x) = [1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]$.

Як відомо з принципу роботи моделі, в результаті прийому кодової комбінації і її декодування можливо отримати чотири варіанти розвитку подій:

1. У першому варіанті, якщо вектор помилки буде «нульовим» $\theta(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$, тоді отримана комбінація. У даному випадку, біт перевірки на парність помилки не виявить і його можна відкинути. Відбувається декодування комбінації першого ступеню $F(x)$, в результаті якої $S_i = [0\ 0\ 0\ 0]$, тобто помилка відсутня $J'(x) = J(x)$. Тоді комбінація передається до споживача
2. У другому варіанті, якщо вектор помилки буде «ненульовим» $\theta(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1]$, тоді отримана комбінація буде мати вигляд $J'(x) = [1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1]$. У даному випадку, при перевірці на парність буде виявлена помилка,

біт перевірки на парність відкидається. Відбувається декодування комбінації першого ступеню $F(x)$, в результаті якої $S_i = [0\ 0\ 0\ 0]$, тобто помилка відсутня. Хоча отримана комбінація і не дорівнює відправленій $J'(x) \neq J(x)$, виходячи з умов моделі можна зробити висновок, що помилка трапилась у біті перевірки на парність, тому декодовану комбінацію можна передати до споживача.

3. У третьому варіанті, якщо вектор помилки буде «ненульовим» $\theta(x) = [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0]$, тоді отримана комбінація буде мати вигляд $J'(x) = [1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$. У даному випадку, при перевірці на парність буде виявлена помилка, біт перевірки на парність відкидається. Відбувається декодування комбінації першого ступеню $F(x)$, в результаті якої $S_i = [1\ 1\ 1\ 1]$, тобто помилка відбулась у останньому розряді. Хоча отримана комбінація і не дорівнює відправленій $J'(x) \neq J(x)$, згідно властивостей коду Хеммінга, помилка виправляється, тому декодовану та виправлену комбінацію можна передати до споживача.
4. У четвертому варіанті, якщо вектор помилки буде «ненульовим» $\theta(x) = [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0]$, тоді отримана комбінація буде мати вигляд $J'(x) = [1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0]$. У даному випадку, біт перевірки на парність помилки не виявить і його можна відкинути.. Відбувається декодування комбінації першого ступеню $F(x)$, в результаті якої $S_i = [0\ 0\ 1\ 1]$, тобто помилка відбулась у третьому розряді. Отримана комбінація не дорівнює відправленій $J'(x) \neq J(x)$. Виходячи з умов моделі можна зробити висновок, що трапилась парна помилка, тому декодовану комбінацію не можна передати до споживача і необхідно зробити перезапит.

3.4 Дослідження підмоделі з використанням циклічного коду

До кодеру надійшла комбінація *G(x) = [01101011011]*. У кодовій комбінації розташовуватись комбінація буде наступним чином:

Таблиця 3.5

Фактичне розташування інформаційних розрядів у підмоделі з циклічним кодом

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$F(x)$	0	1	1	0	1	0	1	1	0	1	1	$П_1$	$П_2$	$П_3$	$П_4$

У якості $P(x)$ нехай виступить наступний поліном: $P(x) = x^4 + x + 1$. В результаті утвориться залишок $R(x) = x^3 + x^2 + x$. Тоді комбінація першого ступеню буде виглядати наступним чином:

Таблиця 3.6

Фактичне розташування всіх розрядів у підмоделі з циклічним кодом

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$F(x)$	0	1	1	0	1	0	1	1	0	1	1	1	1	1	0

Додаючи до комбінації біт перевірки на парність отримуємо :

Таблиця 3.7

Фактичне розташування інформаційних розрядів у підмоделі з циклічним кодом та бітом перевірки на парність

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$J(x)$	0	1	1	0	1	0	1	1	0	1	1	1	1	1	0	0

Тобто комбінація, що буде передаватись по дискретному каналу буде мати вигляд : $J(x) = [0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0]$.

Як відомо з принципу роботи моделі, в результаті прийому кодової комбінації і її декодування можливо отримати чотири варіанти розвитку подій:

1. У першому варіанті, якщо вектор помилки буде «нульовим» $\theta(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$, тоді отримана комбінація. У даному випадку, біт перевірки на парність помилки не виявить і його можна відкинути. Відбувається

декодування комбінації першого ступеню $F(x)$, в результаті якої $R(x) = [0\ 0\ 0\ 0]$, тобто помилка відсутня $J'(x) = J(x)$. Тоді комбінація передається до споживача

2. У другому варіанті, якщо вектор помилки буде «ненульовим» $\theta(x) = [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1]$, тоді отримана комбінація буде мати вигляд $J'(x) = [0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1]$. У даному випадку, при перевірці на парність буде виявлена помилка, біт перевірки на парність відкидається. Відбувається декодування комбінації першого ступеню $F(x)$, в результаті якої $R(x) = [0\ 0\ 0\ 0]$, тобто помилка відсутня. Хоча отримана комбінація і не дорівнює відправленій $J'(x) \neq J(x)$, виходячи з умов моделі можна зробити висновок, що помилка трапилась у біті перевірки на парність, тому декодовану комбінацію можна передати до споживача.
3. У третьому варіанті, якщо вектор помилки буде «ненульовим» $\theta(x) = [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$, тоді отримана комбінація буде мати вигляд $J'(x) = [0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0]$. У даному випадку, при перевірці на парність буде виявлена помилка, біт перевірки на парність відкидається. Відбувається декодування комбінації першого ступеню $F(x)$, в результаті якої $R(x) = [0\ 0\ 1\ 1] = x + 1$, а даний синдром вказує, що помилка сталась у п’ятому розряді. Хоча отримана комбінація і не дорівнює відправленій $J'(x) \neq J(x)$, згідно властивостей циклічного коду, помилка виправляється, тому декодовану та виправлену комбінацію можна передати до споживача.
4. У четвертому варіанті, якщо вектор помилки буде «ненульовим» $\theta(x) = [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1]$, тоді отримана комбінація буде мати вигляд $J'(x) = [0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 1]$. У

даному випадку, біт перевірки на парність помилки не виявить і його можна відкинути. Відбувається декодування комбінації першого ступеню $F(x)$, в результаті якого $R(x) = [0\ 0\ 1\ 1] = x + 1$, а даний синдром вказує, що помилка сталась у п'ятому розряді. Отримана комбінація не дорівнює відправленій $J'(x) \neq J(x)$. Виходячи з умов моделі можна зробити висновок, що трапилась парна помилка, тому декодовану комбінацію не можна передати до споживача і необхідно зробити перезапит.

3.5 Дослідження показників моделі при використанні зворотнього зв'язку

Для того, щоб дослідити модель, по-перше, дослідимо як саме надлишковість буде впливати на кодову комбінацію.

Розглянемо такі зв'язані показники, як відношення надлишкових розрядів до інформаційних та відносну надлишковість (табл. 3.8). Чим менший перший показник і більше другий, тим краще буде для інформативності повідомлень.

Таблиця 3.8

Залежність кількості надлишкових розрядів та $k_{відн}$ від n

n	k	Кількість надлишкових розрядів до інформаційних у %	$k_{\text{відн}}$ (відносна надлишковість)
8	4	0,5	0,5
16	5	0,31	0,68
32	6	0,19	0,81
64	7	0,11	0,89

У вигляді графіку таблиця 3.8 виглядає наступним чином:

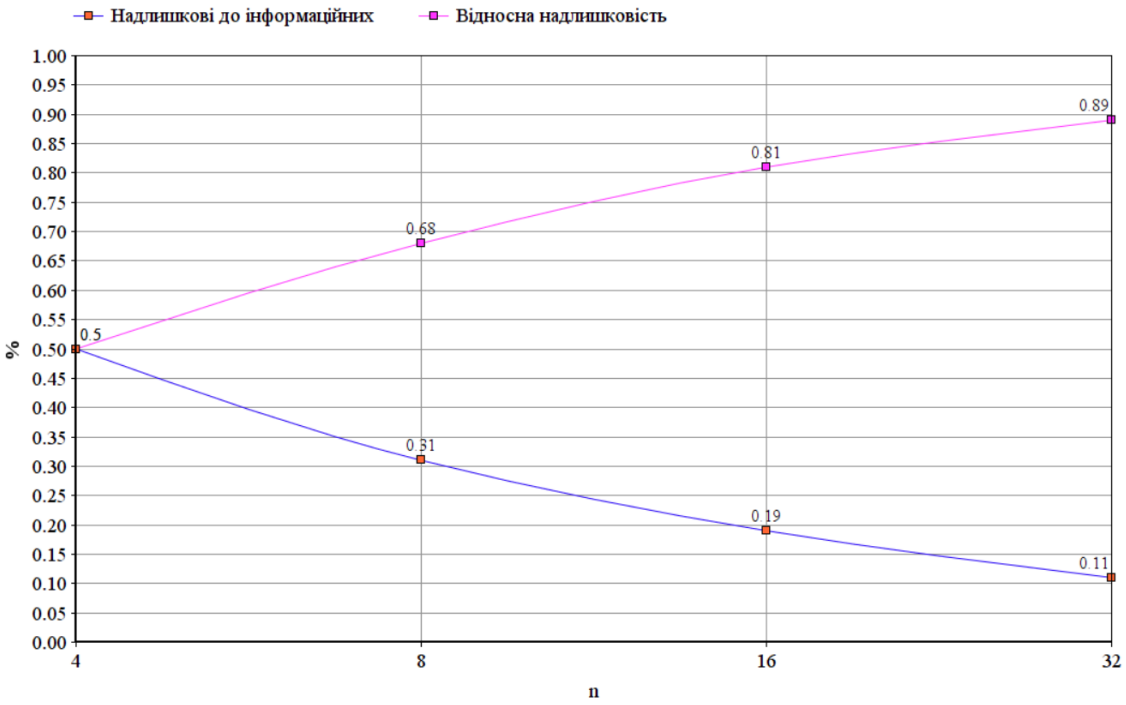

Рисунок 3.6 – залежність кількості надлишкових розрядів та $k_{відн}$ від n

З цього можна робити висновок, що використовувати модель буде дедалі більш доцільно, чим більше буде довжина повідомлення *n*.

Відомо [3], що для довжини блоку *n=16* $p_0 \approx 10^{-4}$. З умови моделі, якщо ураховувати те, що помилки кратності більше двох маловірогідні, вірогідність однократної помилки буде складати $p(\mu_{16}^1) \approx$ 0.000073, тоді $p(\mu_{16}^2) \approx 0.000027$.

Оскільки модель не виключає використання зворотнього зв'язку, адже багато мережевих протоколів підтримують зворотній зв'язок, але мінімізує його, необхідно з'ясувати, як саме це буде впливати на модель.

З формули 1.1 ми знаємо, що $t_p = \frac{D_{max}}{V_c}$. Для крученої пари швидкість поширення сигналу V_c серед кабелю (швидкість поширення електромагнітної хвилі) становить від 68 до 72% швидкості світла [12]. Прийнято називати цей параметр NVP – Nominal Velocity of Propagation – номінальна швидкість розповсюдження сигналу. Якщо перевести швидкість м/с розмірність нс/м, вийде відповідно 4.9 нс/м і 4.6 нс/м. За основу візьмемо швидкість 4.8 нс/м.

За умови, якщо $D_{max} = 1000 * 10^3 м$, то $t_{\text{р}} = \frac{1000*10^3}{4.8*10^9}$. Тоді час очікування буде : (формула 1.2)

$$\text{Т}_{очік} = 2 * \frac{1000*10^3}{4.8*10^9} + t_{\text{оброб}} = 0.0004166666666667 + t_{\text{оброб}},$$

де $t_{\text{оброб}}$ буде вже залежати від обладнання. Враховуючи сучасні вимоги до створення мереж [11], при передачі великих масивів даних, час на очікування часто стає суттєвою неприємною перепоною для достовірності і швидкості при передачі даних.

Це означає, що на 1000000 повідомлень 100 будуть помилковими. Серед 100 помилок 73 будуть однократними, а 27 двократними.

Якщо модель не використовувати і не виправляти помилки, тоді час очікування (у даному випадку затримка, яка буде заважати приймати наступні сигнали і спровокує чергу до моменту надходження відповіді від джерела) буде дорівнювати:

$$\text{Т}_{очік} = 100 * (2 * \frac{1000*10^3}{4.8*10^9} + t_{\text{оброб}}).$$

За умов використання моделі, 73 помилки будуть виправлятись одразу ж на приймаючому пристрої, а щодо останніх 27 помилок рішення буде приймати вже протокол верхнього рівня, і навіть якщо по всім помилкам буде прийнято рішення про перезапит, загальний час очікування (затримки) буде:

$$\text{Т}_{очік} = 27 * (2 * \frac{1000*10^3}{4.8*10^9} + t_{\text{оброб}}).$$

Якщо б модель не використовувалась, а виявлені помилки виявлялись на пристрої отримувача і перезапитувались, це зайняло б вчетверо більше часу, а пропускна здатність була б меншою.

У випадку повідомлення з n = 16, 31% надлишкових розрядів зекономили 73% часу при виправленні помилок.

Для мереж з використанням Wi-Fi, $p_{Wi-Fi}(\mu) \approx p_{\text{круч.пара}}(\mu)$, з швидкістю розповсюдження $V_{\text{с}} \approx 3 * 10^8$ [13],

$$T_{очік} = (2 * \frac{1000*10^3}{3*10^8} + t_{оброб}).$$

Хоча час очікування і збільшується, користь від використання моделі не знижується, знову йде економія 73% часу, де за умови сталої відстані ключову роль грають показники p_0 та V_c.

3.6 Вибір програмного забезпечення

Для функціонування моделі, після формування усіх правил по її формуванню, необхідно обрати програмне забезпечення, яке буде дану модель реалізовувати для подальшого дослідження. У першу чергу необхідно обрати операційну систему. Існує три основні комп'ютерні операційні системи, кожна з них має особливо сильні та слабкі сторони – Linux, iOS, Windows. Але щоб бути впевненими у виборі оптимальної системи, необхідно звернутись до міжнародної статистики. Компанія Statcounter надає наступну інформацію (рисунок 3.7).

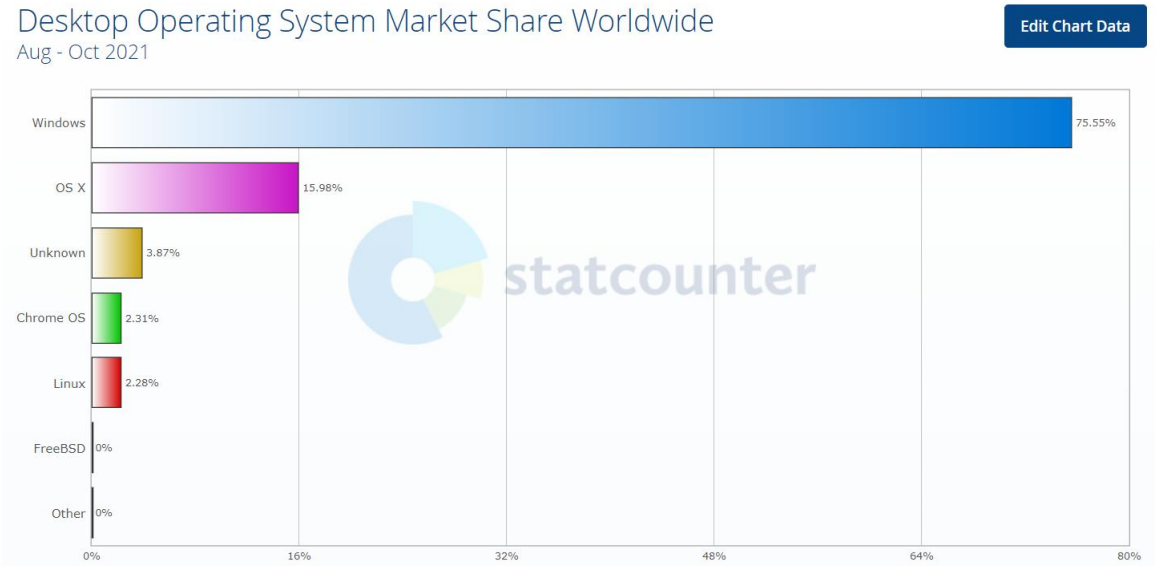

Рисунок 3.7 – Операційні системи для персональних комп'ютерів у світі.

Як видно, ОС Windows має найбільшу популярність. Виходячи з цього, прийнято рішення розробляти комп'ютерну модель для використання в ОС Windows.

Після того, як була обрана ОС, необхідно зробити вибір безпосередньо програми, яка буде дозволить реалізувати та проаналізувати алгоритм та функції моделі, надасть можливості для введення даних, тобто буде мати можливість ведення математичної статистики, заміру часу, а також реалізації роботи з поліномами та матрицями.

У джерелі [9] приводяться 67 математичних програм, серед яких є: SALOME, StatEx, FreeMat, Scilab та інші. Серед них було обрано програму Scilab версії 6.1.1 з тієї причини, що Scilab містить пакет наукових програм для числових розрахунків, які надають потужну відкриту середу для інженерних та наукових розрахунків. Також розглядалась така програма як MathCad_Prime_7.0, але не була обрана з причини її розміру 1.75 Гб, з наданням таких же самих можливостей для реалізації моделі як і Scilab, розмір якого 544 Мб.

Інтерфейс програми виглядає наступним чином:

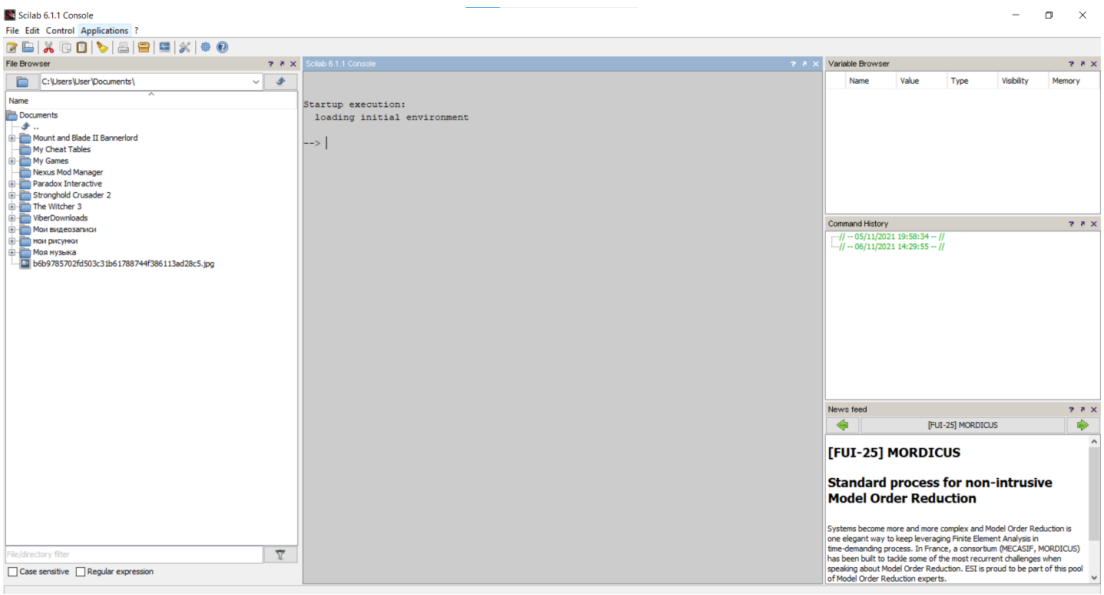

Рисунок 3.8 – Знімок екрану Scilab 6.1.1

Висновки до розділу 3

У розділі, згідно постановці задачі, за основи даних, які наводились у розділі 2, сформована модель по використанню каскадного коду з систематичних кодів, формуємих матричним способом, та додаванням до них біту перевірки на парність, що надає змогу виправляти однократні помилки, а також виявляти усі помилки парної кратності. Сформований алгоритм кодування та декодування, а також проаналізовано роботу алгоритму на базі двох під моделей із використання циклічного коду і коду Хеммінга. При дослідженні виявлено, що використання завадостійкого кодування по принципу моделі, економить до 73% часу при передачі дискретних повідомлень, враховуючи існування зворотнього зв’язку, що є суттєвим зменшенням часу.

Розділ 4.

ДОСЛІДЖЕННЯ ПРОГРАМИ МОДЕЛІ

4.1 Робота програми моделі при коді Хеммінга

Оскільки різні варіанти під моделей (для коду Хеммінга, для циклічного коду, тощо) розрізняються лише першим ступенем, тобто без урахування біту перевірки на парність, розпочнемо аналіз саме з перших ступеней коду, розмірність яких буде стартувати з (7,4), а потім буде зростати. За основу провідних програмних функцій, які будуть використовуватись для реалізації роботи моделі, було обрано програмне джерело від професора Калаваті П. [10]

Для початку, програмі були задані правила перевірки для даного коду, прописані правила побудови матриць.

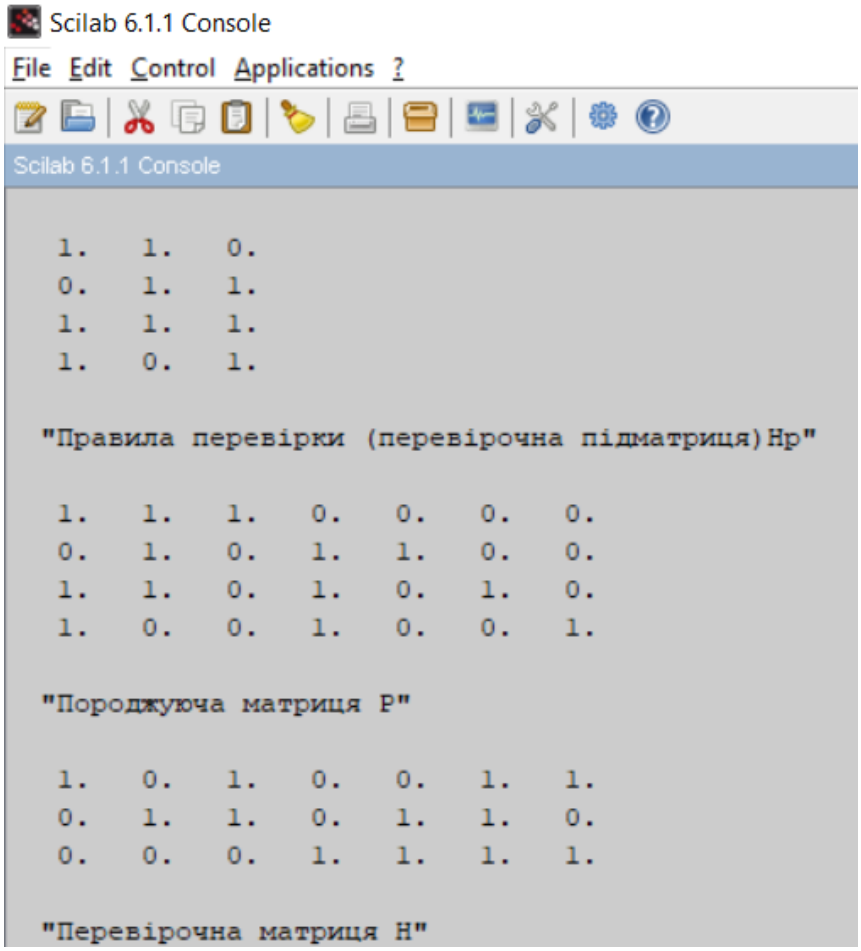

Рисунок 4.1 – Матриці H_p, P, H для коду Хеммінга

Після цього, була задана початкова вхідна комбінація *G(x)*. Вона була закодована.

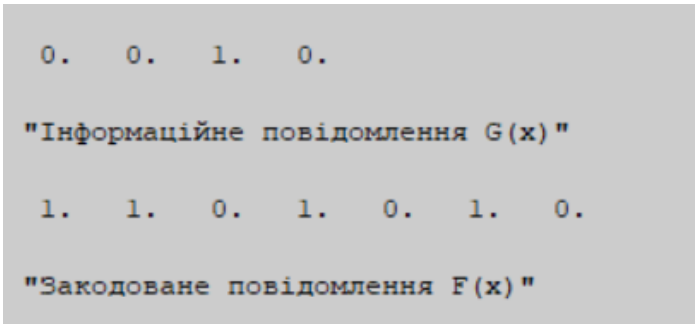

Рисунок 4.2 – Кодування комбінації для коду Хеммінга

Надалі було допущено, що при передачі трапилась помилка у деякому розряді комбінації, програма симулювала помилку. Після симуляції помилки,

кодова комбінація була декодована, було отримано $S \neq [0\ 0\ 0]$, що вказало на наявність помилки, а також вказало на номер помилкового розряду.

```
 1.   1.   0.   1.   0.   0.   0.

"Прийнята комбінація з урахуванням можливої помилки 6"

 1.   0.   1.

"Синдром"

 6.

"Номер помилкового розряду"
```

Рисунок 4.3 – Симуляція помилки для коду Хеммінга

Після виявлення номеру помилкового розряду, комбінація була виправлена та декодована.

```
 1.   1.   0.   1.   0.   1.   0.

"Отриманий код з помилкою виправлено"

 0.   0.   1.   0.

"Отримане повідомлення"
```

Рисунок 4.4 – Виправлення помилки та декодування комбінації для коду Хеммінга

Також був зафіксований час на виконання повної процедури кодування/декодування.

```
0.03125

"Час виконання"
```

Рисунок 4.5 – Час виконання процедури для коду Хеммінга

Час на виконання кожного разу був різний, але не відхилявся більше, ніж на 3-4% від результаті при першій спробі.

Усього було зроблено 5 спроб, результати зведені до таблиці. Також при симуляції допускався випадок, коли помилки не було (не було необхідності виправляти помилку). Також було додано час, за який програма кодувала і декодувала повідомлення, з урахуванням біту перевірки на парність (його додавання завжди становило майже одну і ту ж саму долю часу).

Таблиця 4.1

Час виконання програми для кодів Хеммінга з основою (7,4)

Номер спроби	Час виконання (7,4) у секундах	Час виконання (8,4) у секундах	Необхідність виправлення помилки
1.	0,03125	0,0321	+
2.	0,03151	0,03226	+
3.	0,03136	0,03218	+
4.	0,031	0,03176	-
5.	0,03116	0,03188	+
Середнє	0,03126	0,03203	

Де «+» - помилка була і виправлялась, «-» - помилки не було і вона не виправлялась.

Таке ж дослідження проводилось для кодів Хеммінга (16,11) та (32,26). Лістинг не приводиться, адже він подібний до коду (7,4) і має великий розмір.

Результати зведені до таблиць 4.2 та 4.3.

Таблиця 4.2

Час виконання програми для кодів Хеммінга з основою (15,11)

Номер спроби	Час виконання (15,11) у секундах	Час виконання (16,11) у секундах	Необхідність виправлення помилки
1.	0,0581	0,05895	+
2.	0,0579	0,0586	-
3.	0,05816	0,0588	+
4.	0,05806	0,05876	+
5.	0,05817	0,05863	+
Середнє	0,05807	0,05874	

Таблиця 4.3

Час виконання програми для кодів Хеммінга з основою (31,26)

Номер спроби	Час виконання (31,26) у секундах	Час виконання (32,26) у секундах	Необхідність виправлення помилки
1.	0,079	0,0874	+
2.	0,07933	0,0863	+
3.	0,0795	0,0878	+
4.	0,07896	0,07991	+
5.	0,07888	0,07988	-
Середнє	0,07913	0,08425	

При зведенні таблиць 4.1, 4.2 та 4.3 отримуємо:

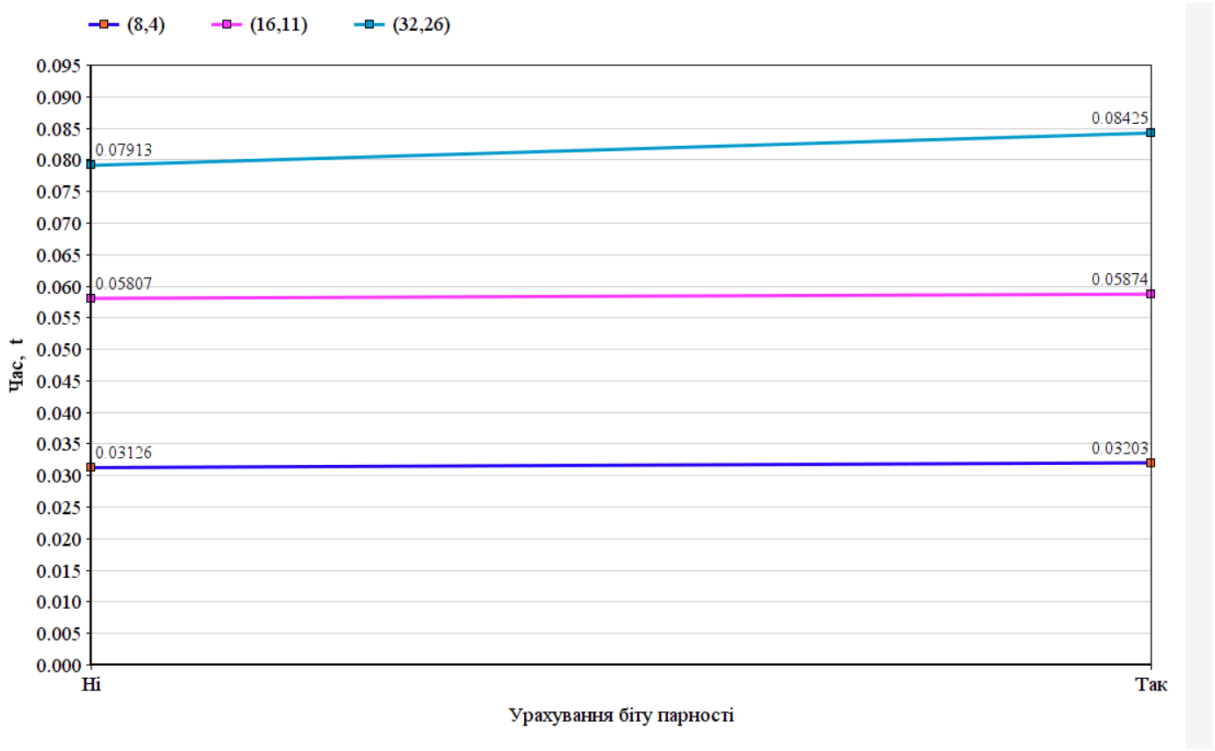

Рисунок 4.6 – Час виконання програми для коду Хеммінга при додаванні біту парності

З отриманого можна зробити висновок, що із додаванням біту перевірки на парність швидкодійність програми спадає не суттєво, що з урахуванням практичної корисності такого підходу є гарним результатом.

На рисунку 4.7 приводиться залежність часу виконання програми від розмірності.

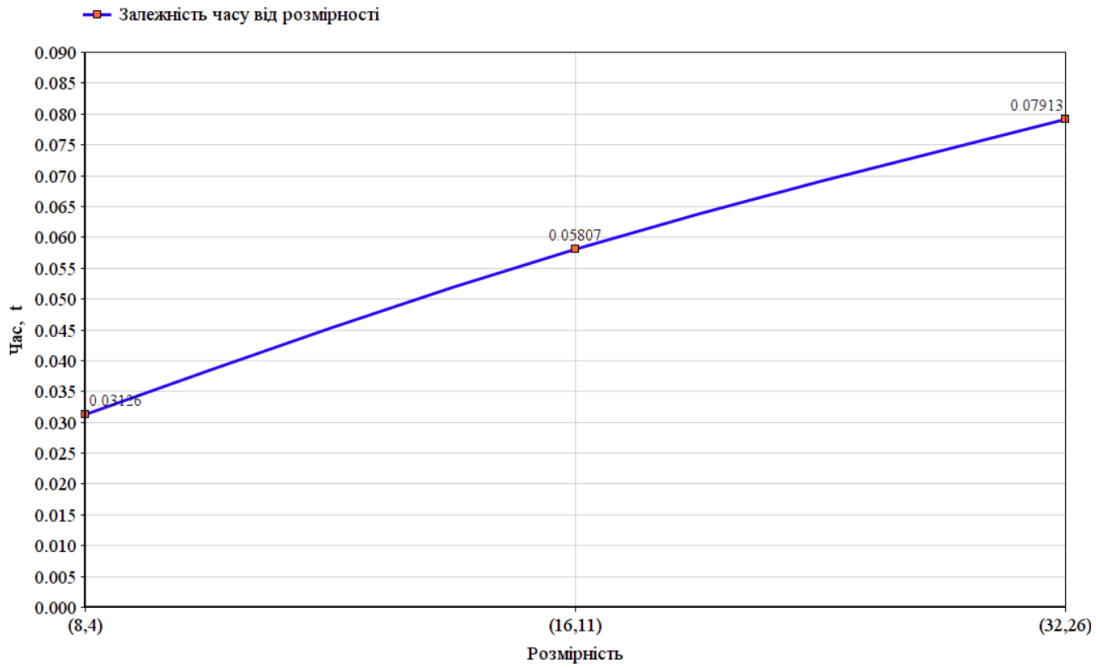

Рисунок 4.7 – Час виконання програми для коду Хеммінга при різних n

4.2 Робота програми моделі при циклічному коді

Для початку, програмі був заданий образуючий поліном та вхідний інформаційний поліном.

```
1 +x +x³

"Образуючий поліном P"
```

```
x² +x³

"Інформаційний поліном G"
```

Рисунок 4.8,4.9 – Образуючий поліном та інформаційний поліном для циклічного коду

Після цього, поліном був помножено на 10^3 та згенеровано поліном F.

```
1 +x⁵ +x⁶

"Інформаційний поліном G у ступені *x^3"

 1.   0.   0.   0.   0.   1.   1.

"Сгенерований поліном F"
```

Рисунок 4.10 – Кодування комбінації для циклічного коду

Надалі було допущено, що при передачі трапилась помилка у деякому розряді комбінації, програма симулювала помилку. Після симуляції помилки, кодова комбінація була декодована, було отримано $S \neq [0\ 0\ 0]$, що вказало на наявність помилки, а також вказало на номер помилкового розряду.

```
 0.   0.   0.   0.   0.   1.   1.

"Поліном F з урахування помилки 0"

 1.   0.   0.

"Синдром S"

 1.

"Номер розряду помилки"
```

Рисунок 4.11 – Симуляція помилки для коду Хеммінга

Після виявлення номеру помилкового розряду, комбінація була виправлена та декодована. Також був зафіксований час на виконання повної процедури кодування/декодування.

```
 1.   0.   0.   0.   0.   1.   1.

"Отримане повідомлення з помилкою виправлено"

 0.   0.   1.   1.

"Отримане повідомлення"

 0.   0.   1.   1.

"Надіслане повідомлення"

 0.046875

"Час виконання"
```

Рисунок 4.12 – Декодування комбінації для циклічного коду та час виконання процедури

Час на виконання кожного разу був різний, але не відхилявся більше, ніж на 4-5% від результаті при першій спробі.

Усього було зроблено 5 спроб, результати зведені до таблиці. Також при симуляції допускався випадок, коли помилки не було (не було необхідності виправляти помилку). Також було додано час, за який програма кодувала і декодувала повідомлення, з урахуванням біту перевірки на парність (його додавання завжди становило майже одну і ту ж саму долю часу).

Таблиця 4.4

Час виконання програми для циклічних кодів з основою (7,4)

Номер спроби	Час виконання (7,4) у секундах	Час виконання (8,4) у секундах	Необхідність виправлення помилки
1.	0,04688	0,04768	+
2.	0,04611	0,0468	-
3.	0,04697	0,0479	+
4.	0,04711	0,04769	+
5.	0,04656	0,04711	+
Середнє	0,04672	0,04743	

Де «+» - помилка була і виправлялась, «-» - помилки не було і вона не виправлялась.

Таке ж дослідження проводилось для циклічних кодів (16,11) та (32,26). Лістинг не приводиться, адже він подібний до коду (7,4) і має великий розмір.

Результати зведені до таблиць 4.5 та 4.6.

Таблиця 4.5

Час виконання програми для циклічних кодів з основою (15,11)

Номер спроби	Час виконання (15,11) у секундах	Час виконання (16,11) у секундах	Необхідність виправлення помилки
1.	0,0697	0,07039	+
2.	0,0665	0,06999	-
3.	0,0695	0,07066	+
4.	0,06999	0,07099	+
5.	0,06969	0,07036	+
Середнє	0,06907	0,07047	

Таблиця 4.6

Час виконання програми для циклічних кодів з основою (31,26)

Номер спроби	Час виконання (31,26) у секундах	Час виконання (32,26) у секундах	Необхідність виправлення помилки
1.	0,08902	0,08977	+
2.	0,08911	0,08978	+
3.	0,08864	0,08911	-
4.	0,08903	0,0898	+
5.	0,08909	0,08983	+
Середнє	0,08897	0,08965	

При зведенні таблиць 4.4, 4.5 та 4.6 отримуємо:

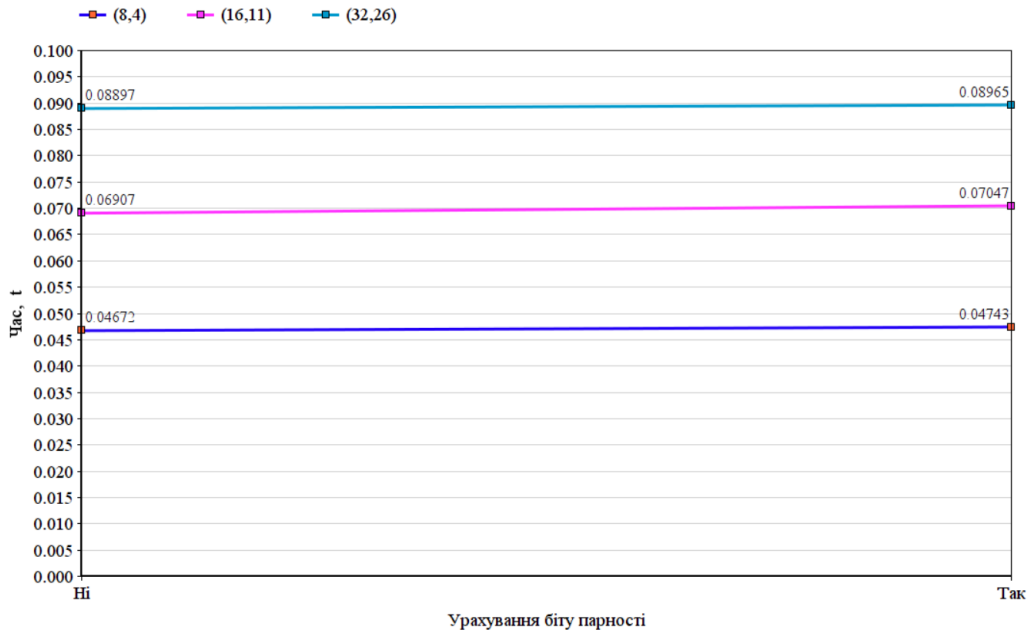

Рисунок 4.13 – Час виконання програми для циклічного коду при додаванні біту парності

З отриманого можна зробити висновок, що із додаванням біту перевірки на парність швидкодійність програми спадає не суттєво, що з урахуванням практичної корисності такого підходу є гарним результатом.

На рисунку 4.14 приводиться залежність часу виконання програми від розмірності.

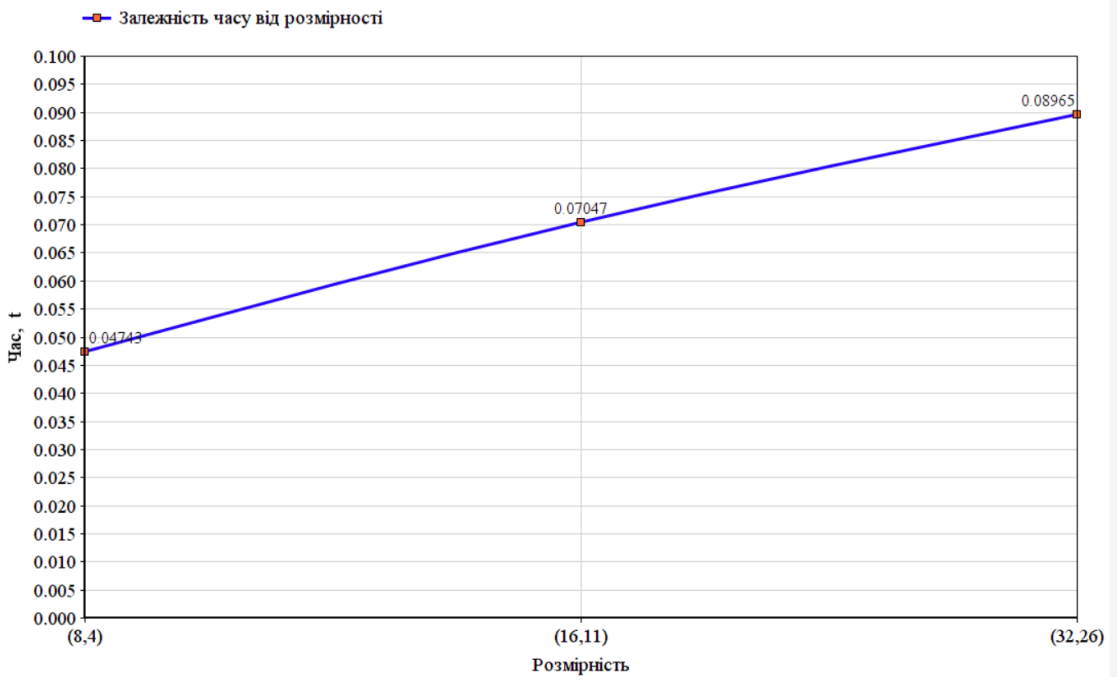

Рисунок 4.14 – Час виконання програми для коду Хеммінга при різних n.

4.3 Порівняльний аналіз показників моделі

Основний показник, яким можна оперувати у порівнянні двох під моделей – час виконання програми. На рис 4.15 приведені криві, які вказують на даний параметр. Необхідно відмітити, що циклічний код, з причини його дельш більшої складності, займає трохи більше часу на виконання, ніж код Хеммінга, але необхідно відмітити, що для розмірності (8,4) різниця між циклічним кодом та кодом Хеммінга становила 0,01617, для розмірності (16,11) 0,0124, для розмірності (32,26) 0,01052. Тобто є пряма тенденція на зменшення різниці часу.

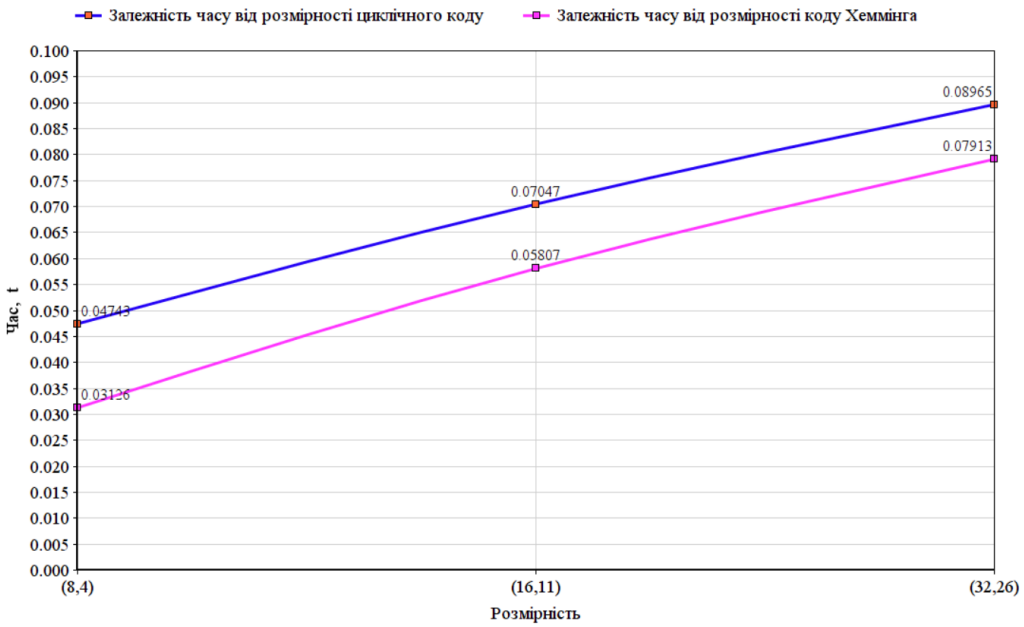

Рисунок 4.15 – Час виконання програми для коду Хеммінга та циклічного коду при різних n

Висновки до розділу 4

У розділі реалізовувалась і досліджувалась програма моделі. На основі програми Scilab написано код, який реалізовує роботу моделі, а також було проведене дослідження на швидкодійність серед двох підмоделей (на основі коду Хеммінга та циклічного коду), які формувались матричним методом, що дозволило та оцінити їх порівняти. Як результат, з’ясовано, що для

невеликих об’ємів блоків даних, циклічний код поступається у швидкодійності коду Хеммінга, але дана різниця знижається разом із зростання об’ємів блоків даних.

РОЗДІЛ 5.

РОЗРОБКА РЕКОМЕНДАЦІЙ ПО ВИКОРИСТАННЮ РОЗРОБЛЕНОЇ ПРОГРАМИ МОДЕЛІ В НАВЧАЛЬНОМУ ПРОЦЕСІ

5.1 Призначення комп'ютерної моделі каскадного коду

Комп'ютерна модель каскадного коду розроблена для автоматизованого кодування інформаційної комбінації каскадними кодами Хеммінга та циклічного коду, виправлення однократної (одноразової) помилки, виявлення двократної (дворазової).

5.2 Умови роботи комп'ютерної моделі каскадного коду

Розроблена модель може експлуатуватися на персональному комп'ютері (ПК). Для роботи знадобиться операційна система Windows. Для повноцінного використання моделі на ПК використовується дисплей, клавіатура і маніпулятор типу «миша».

5.3 Опис основних функцій моделі каскадного коду підмоделей для і їх першого ступеню

Для роботи з моделлю необхідно запустити програму Scilab 6.1.1 Desktop, після чого відкрити файл CyclicCodes, або HammingCodes в залежності від досліджуємої підмоделі. Повний лістинг і результати виконання програм не будуть надаватись у повному обсязі з причини їх великого об’єму.

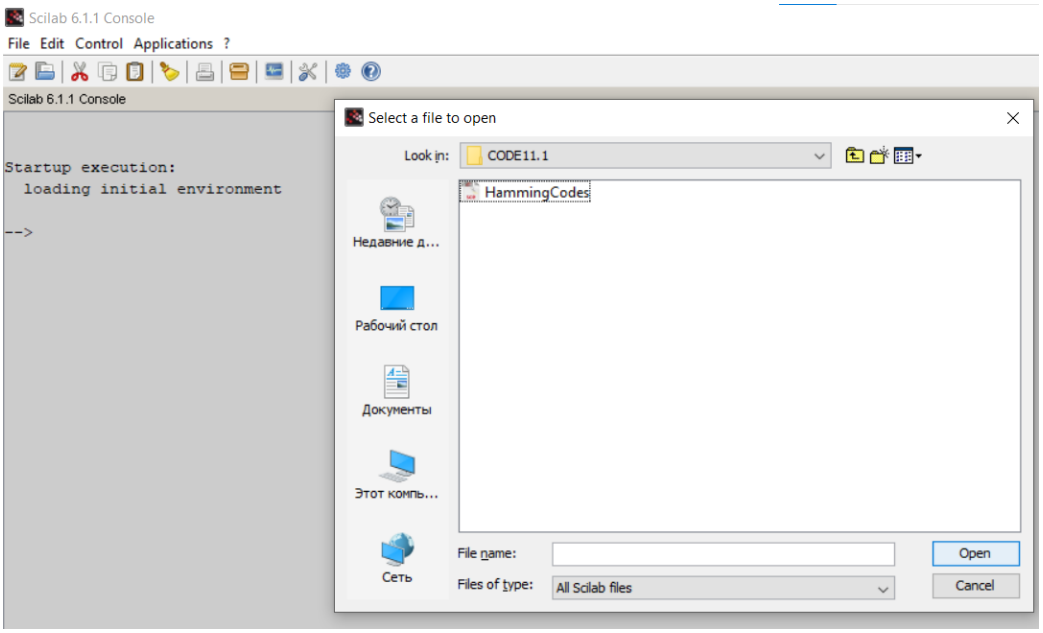

Рисунок 5.1 – Меню вибору файлу для роботи.

В разі дослідження коду Хеммінга (вибору файла HammingCodes), у відкрившомуся скрипті задаються параметри *n*, *k*, а також задається перевірочна матриця.

```
k = 4; //Information message matrix length
n = 7; //Coded word length

P = [1 1 0;0 1 1 ;1 1 1;1 0 1]//Parity Matrix
disp(P,'Правила перевірки (перевірочна підматриця)Hp')
```

Рисунок 5.2 – Введення головних параметрів для коду першого ступеню підмоделі з кодом Хеммінга.

На основі введених даних генерується інформаційна комбінація, після чого вона кодується.

```
All_M = [0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;
0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;
1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;
1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1]
RandMessage=modulo(round(16*rand()),16)+1//Get random number between 1 to 16
M=All_M(RandMessage,:)//Select a random row from 1 to 16 as Information Message

disp(M,'Інформаційне повідомлення G(x)')

C = M*G;//Generate code word
C = modulo(C,2);//Convert generated code into binary
disp(C,'Закодоване повідомлення F(x)')
```

Рисунок 5.3 – Кодування коду Хеммінга.

На основі отриманої комбінації симулюється можливість неправильного прийому (помилки).

```
ErrPos=modulo(round(8*rand()),8)//Get random number between 0 to 7

if ErrPos==0 then
    //Do nothing, as '0' means no error
else
    if R(ErrPos)==0 then
        R(ErrPos)=1//Invert bit at Erroneous Bit Position
    else
        R(ErrPos)=0//Invert bit at Erroneous Bit Position
    end
end
```

Рисунок 5.4 – Симуляція неправильного прийому для коду Хеммінга.

Після прийому починається процес отримання перевірочного синдрому *S*.

```
S=R*H' //Find Syndrome Matrix
S = modulo(S,2);//Convert Syndrome Matrix into binary
disp(S,'Синдром')
```

Рисунок 5.5 – Декодування коду Хеммінга

На основі отриманого *S* робиться висновок про стан повідомлення першого ступеню каскадного коду. Приймається рішення щодо коду першого ступеню, на основі якого буде прийматись загальне рішення про отриману комбінацію у рамках моделі.

```
if S==[0 0 0]  then //[0 0 0] indicates no error
    disp(R,'Код прийнятий без помилок')
    disp([R(3) R(5:7)],'Отримане повідомлення F(x)')//Extract and display  Message from code word
else
    //Find erroneous bit position
    //Here we find colomn within H matrix with pattern simmilar to Syndrome Matrix
    //The position number of that colomn is equivalent to erroneous bit position

    ErrPos=1//Initiallize erroneous bit position
    d=[H(:,ErrPos)]'//Transpose of first coloumn of H matrix
                    //(Transpose is used to convert colomn to row as syndrome is in row format)

        while ((d(1)<>S(1))|(d(2)<>S(2))|(d(3)<>S(3))) do //Check element wise inequallity for any element (OR condition)
            ErrPos=ErrPos+1//Increament erroneous bit position (Point to next colomn)
            d=[H(:,ErrPos)]'//Transpose of next coloumn of H matrix
        end

        disp(ErrPos,'Номер помилкового розряду')

        //Error correction
        if R(ErrPos)==0 then
                R(ErrPos)=1//Invert bit at Erroneous Bit Position
                disp(R,'Отриманий код з помилкою виправлено')
                disp([R(3) R(5:7)],'Отримане повідомлення')//Extract and display  Message from code word
```

Рисунок 5.6 – Прийняття рішення по коду Хеммінга

В разі дослідження циклічного коду (вибору файла CyclicCodes), у відкрившомуся скрипті задаються параметри *n*, *k*, а також задається образуючий поліном.

```
k = 4; //Information Message Length
n = 7; //Codeword Length
timer()
//Generator Polynomial
x=poly(0,'x');
GenPoly=1+x+x^3;
disp(GenPoly,'Образуючий поліном P');
```

Рисунок 5.7 – Введення головних параметрів для коду першого ступеню підмоделі з циклічним кодом.

Після йде формування довільного інформаційного поліному.

```
M=All_M(RandMessage,:)//Select a random row from Message Matrix All_M as Information Message
disp(M,'Information Message M')

//Message Polynomial
MesPoly=(M(1)*1) + (M(2)*(x^1)) + (M(3)*(x^2)) + (M(4)*(x^3));
disp(MesPoly,'Інформаційний поліном G');
```

Рисунок 5.8 – Кодування циклічного коду.

Надалі інформаційний поліном кодується першим ступенем, після чого також симулюється можливість неправильного (з помилкою) прийому.

```
//Generating Codeword Polynomial
p=(x^(n-k))*(MesPoly);//Step 1 - multiply MesPoly by x^(n-k), [x^(n-k)*u(x)]
[RemPoly,q]=pdiv(p,GenPoly);//Step 2 - divide above product by GenPoly, g(x) (Polynomial Division)
RemPoly=modulo(RemPoly,2);//Convert Remainder Polynomial to binary to get parity check polynomial, b(x)
//disp(RemPoly,'Remainder Polynomial b(x)');
CodePoly=RemPoly+(MesPoly*(x^(n-k)));//Step 3 - add (x^(n-k)*u(x)) and b(x) to get Codeword Polynomial
disp(CodePoly,'Інформаційний поліном G у ступені *x^3');

//Finding Coefficients of Codeword Polynomial
CodePolyCoeff=coeff(CodePoly);
    //Removal of - signs from Coefficients of Codeword Polynomial
    for i=1:length(CodePolyCoeff)
        if (CodePolyCoeff(i)==-1) then
            CodePolyCoeff(i)=1;
        end
    end
//disp(CodePolyCoeff,'Coefficients of Codeword Polynomial');

//Generating 7 bit Codeword from Coefficients of Codeword Polynomial
C=CodePolyCoeff;
if length(C)<7 then
    C(1,7)=0;//Assigning a value outside arrey dimension will automatically
            //pad additional zeros to resize the arrey / vector
end
disp(C,'Сгенерований поліном F');
```

Рисунок 5.9 – Симуляція неправильного прийому для циклічного коду.

Після цього йде декодування отриманої комбінації.

```
//Reception and Decoding-------------------------------------------------------------------

disp(R,'Поліном F з урахування помилки θ')

//Received Polynomial
RecPoly=(R(1)*1) + (R(2)*(x^1)) + (R(3)*(x^2)) + (R(4)*(x^3)) + (R(5)*(x^4)) + (R(6)*(x^5)) + (R(7)*(x^6));
//disp(RecPoly,'Received Polynomial u(x)');

//Syndrome Polynomial
[SynPoly,q]=pdiv(RecPoly,GenPoly);
SynPoly=modulo(SynPoly,2)
//disp(SynPoly,'Syndrome Polynomial')

//Finding Coefficients of Syndrome Polynomial
SynPolyCoeff=coeff(SynPoly);
    //Removal of - signs from Coefficients of Syndrome Polynomial
    for i=1:length(SynPolyCoeff)
        if (SynPolyCoeff(i)==-1) then
            SynPolyCoeff(i)=1;
        end
    end
//disp(SynPolyCoeff,'Coefficients of Syndrome Polynomial');

//Generating 3 bit Syndrome from Coefficients of Syndrome Polynomial
if length(SynPolyCoeff)<3 then
    SynPolyCoeff(1,3)=0;//Assigning a value outside arrey dimension will automatically
            //pad additional zeros to resize the arrey / vector
```

Рисунок 5.10 – Декодування циклічного коду.

На основі отриманого *S* робиться висновок про стан повідомлення першого ступеню каскадного коду. Приймається рішення щодо коду першого

ступеню, на основі якого буде прийматись загальне рішення про отриману комбінацію у рамках моделі.

Висновки до розділу 5

У п’ятому розділі розроблено посібник для користувача. Посібник дозволяє користувачу, який знає принципи використання програми Scilab, без складнощів взаємодіяти з моделлю. Розроблену модель можна використовувати в навчальному процесі.

Для перевірки працездатності розробленої комп'ютерної моделі каскадного коду та правдоподібності отриманих результатів, модель була протестована. Розроблена комп'ютерна модель тестувалася по 5 спроб для коду Хеммінга з k = 4, 11, 27, та по 5 спроб для циклічного коду з k = 4, 11, 27. числах, кожного разу вірно генеруючи комбінацію. Експерементально закодована та декодована комбінація повністю відповідає вимогам до моделі.

Програма та методика випробування моделі приведена в додатку В

ВИСНОВКИ ПО РОБОТІ

У основі роботи полягла розробка моделі каскадного коду для достовірної передачі по дискретним каналам зв’язку. На початку роботи, завдяки дослідженню предметної області, а саме розбору схеми будування дискретних каналів зв’язку, а також видів сигналів, з’ясовано, що помилки, які трапляються у безперервних каналах зв’язку – одна з основних перепон для достовірної передачі інформації. В результаті пошуку та аналізу статистики помилок у дискретних каналах зроблено висновок, що саме однократні (одноразові) та двократні (дворазові) помилки є основною проблемою, боротьба з якими і стала постановкою задачі по роботі.

В ході дослідження вирішено було боротись із помилками за поміччю використання завадостійкого кодування. Оскільки модель має адаптивно

працювати з різними системами, необхідним критерієм постала універсальність, тобто щоб модель могла працювати за різних розмірностей, при цьому не змінюючи алгоритму кодування та декодування. Серед багатьох видів завадостійких кодів, на основі даних розглянутих при постановці задачі, вирішено було використовувати саме каскадні коди, першим ступенем яких будуть систематичні коди, які будуть кодуватись матричним способом, а другим ступенем буде виступати код перевірки на парність.

На основі прийнятого рішення про схему коду, розроблена модель каскадного коду, яка у подальшому аналізовувалась. Для аналізу використовувались дві підмоделі (на основі циклічного коду та коду Хеммінга). В результаті аналізу з'ясовано, що використання каскадного кодування до 4 разів зменшує час на передачу і корекцію помилкових повідомлень, а також зменшує кількість перезапитів, що зменшує час очікування у чергах повідомлень, що збільшує пропускну здатність.

На основі двох підмоделей, була написана комп'ютерна програма, яка реалізовувалась за допомогою програми Scilab, що дозволило проаналізувати програмні показники, головним з яких був час виконання алгоритму. Було з'ясовано, що для невеликих розмірностей формування циклічного коду займає на 5-6% більше часу, ніж формування коду Хеммінга, але така тенденція зменшувалась по мірі зростання довжини блоку n.

Для написаної програми розроблена рекомендація по використанню програми, де описуються умови роботи з комп'ютерною моделлю і наводяться головні функції моделі, які реалізовують логіку роботи алгоритму.

З цього можна зробити висновок, що поставлене завдання було виконано, а модель може адаптивно працювати з різними мережами і протоколами.

Список використаних джерел

1. Основы передачи дискретних сообщений. [Електронний ресурс] // siblec.ru: информ.- 2009-2021 Банк лекций Siblec.Ru. Учебные материалы ОКСО 210000. Электронная техника, радиотехника и связь. Лекции для преподавателей. URL: https://siblec.ru/telekommunikatsii/osnovy-asinkhronnogo-rezhima-peredachi (дата звернення 17.10.2021)

2. Черпаков, И. В. Теоретические основы информатики : учебник и практикум для академического бакалавриата / [И. В. Черпаков.]— Москва : Издательство Юрайт, 2019. — 353 с

3. Информационные параметры позиционных кодов: / [Н.В. Захарченко, С.М. Горохов, А.В. Кочетков]. – Одесса: ОНАС им. А.С. Попова, 2018. – 212 с.

4. Интегрально-оптические волноводные дисперсионные элементы для ВОЛС:/[Д.т.н В.Ш. Берикашвили, Н.Т. Ключник, К.Н. Костенко, к.т.н. М.Я. Яковлев].- Москва, ЦНИТИ «Техномаш-ВОС»,01.02.2005.-16 с.

5. Протоколы Интернет: / [Семенов Ю.А].- Москва, Горячая линия-Телеком, 2001 - 1100 с.

6. Общие сведения о сети Integrated Services Digital Network: история создания, компоненты, инкапсуляция, использование. Типы пользовательского интерфейса, которые поддерживает технология. Адресация в сетях, стек протоколов. Подключение оборудования к сети. [Електронний ресурс] // siblec.ru: информ.- 2009-2021 Банк лекций Siblec.Ru. Учебные материалы ОКСО 210000. Электронная техника, радиотехника и связь. Лекции для преподавателей. URL: https://siblec.ru/telekommunikatsii/osnovy-asinkhronnogo-rezhima-peredachi (дата звернення 16.10.2021)

7. Основы передачи дискретных сообщений [Електронний ресурс] // siblec.ru: информ.- 2009-2021 Банк лекций Siblec.Ru. Учебные материалы ОКСО 210000. Электронная техника, радиотехника и связь. Лекции для преподавателей. URL: https://siblec.ru/telekommunikatsii/osnovy-asinkhronnogo-rezhima-peredachi (дата звернення 17.10.2021)

8. Основи теорії передачі інформації :/ [Ю. І. Лосев, С. І. Шматков ; за ред. Ю. І. Лосева]. - Харків : ХНУ ім. В. Н. Каразіна, 2013. - 290,

9. Рекомендуемые бесплатные программы для математического моделирование и анализа в Windows [Електронний ресурс]// URL: https://pro-spo.ru/po/2008-03-25-05-39-32 (дата звернення 06.11.2021)

10. Scilab Manual for Digital Communication by Prof Kalawati Patil Others Thakur College of Engineering & Technology1 Solutions provided by Mr Sanjay Rawat Others Mumbai University/Thakur College of Engg. & Tech. November 5, 2021 . Funded by a grant from the National Mission on Education through ICT. - 54 c.

11. Олифер Виктор, Олифер Наталья 0-54 Компьютерные сети. Принципы, технологии, протоколы: Юбилейное издание. — СПб.: Питер, 2020. — 1008 с.: ил. — (Серия «Учебник для вузов»). ISBN 978-5-4461-1426-9

12. ЗАДЕРЖКА РАСПРОСТРАНЕНИЯ СИГНАЛА В КАБЕЛЕ [Електронний ресурс] URL: https://www.icsgroup.ru/library/consult/detail.php?NUM=211 (дата звернення 13.11.2021)

13. Пропускная способность и скорость распространения сигнала в локальной сети [Електронний ресурс]] URL: https://skomplekt.com/propusknaya-sposobnost-i-skorost-rasprostranenya-signala-v-localnoy-seti/ (дата звернення 21.11.2021)

14. Искусство помехоустойчивого кодирования /[Морелос-Сарагоса Роберт] Москва: техносфера, 2005.- 320 с.

15. Помехоустойчивое кодирование. Методы и алгоритмы / Под. ред. чл.-кор. РАН Ю. Б. Зубарева Золотарёв В.В., Овечкин Г.В.2004 г. -126 стр.

ДОДАТКИ
Додаток А

(Сторінка 1 додатку А, буде додана при від скануванні)

(Сторінка 2 додатку А, буде додана при від скануванні)

Додаток Б

Технічне завдання
на розробку програмної моделі коригувального каскадного коду в каналах передачі даних системи управління

Назва розділу	Назва і зміст підрозділу
1. Введення	1)Назва: Модель коригувального каскадного коду в каналах передачі даних системи управління. 2)Область застосування: телекомунікаційні системи в автоматизованих системах управління технологічним процесом.
2. Підстава для розробки	1)Навчальний план за спеціальністю 151 – автоматизація та комп'ютерно-інтегровані технології. 2)Завдання на дипломну роботу: наказ про затвердження тем кваліфікаційних робіт від 10.09.2021 р. м. Харків №0210-05/1804 (Додаток А)
3.Призначення розробки	1)Мета розробки: підвищення завадостійкості у дискретних каналах передачі даних шляхом створення моделі каскадного коригувального коду для достовірної передачі повідомлень по дискретним каналам зв'язку в автоматизованих системах управління технологічним процесом. 2)Призначення виробу: дослідити процес підвищення вірності передачі даних у системі управління за використанням каскадних кодів. 3)Вихідні дані для розробки представити у розділі 2 програмної записки.
4. Вимоги до програмного виробу	1)Вимоги до функціональних характеристик: - генерувати початкову комбінацію, шляхом вводу користувачем будь-якого дозволеного числа, що являється вхідними даними; - генерувати систематичний код (n/m) на основі початкової комбінації; - генерувати каскадний код (n+1/m) - симулювати одноразову та дворазову помилки, продемонструвати хід пошуку помилок (декодувати). - програма має гарантовано виправляти одноразові помилки, а у разі їх відсутності не вносити змін до комбінації; 2)Вимоги до надійності : - програма має без помилок працювати на ОС Windows (7,8,10); - програма має бути сумісною з останньою версією середи Scilab; 3)Вимоги до умов експлуатації: -умови експлуатації програми співпадають з умовами експлуатації персональної електронно-вираховувальної машини, на якій буде працювати, а також сумісних з нею персональними комп'ютерами;

<table>
<tr><td></td><td colspan="2">-програма має бути розрахована на непрофесійного користувача;.
-для користування програмою користувачу необхідно пройти короткий курс навчання роботі з програмою;

4)Вимоги до складу і параметрів технічних засобів експлуатації:
- персональний комп’ютер у повній комплектації (системний блок, дисплей, миша, клавіатура).
- ноутбук на базі операційної системи Windows;
5)Вимоги до інформаційної та програмної сумісності:
- операційна система Windows (7,8,10);
- середа розробки Scilab 6.1.1 (2021).

6)Вимоги до маркування та упаковки: не потрібні.
7)Вимоги до транспортування і зберігання : не потрібні.
8)Спеціальні вимоги : не потрібні.</td></tr>
<tr><td>5. Вимоги до програмної документації.</td><td colspan="2">Програмною документацією щодо розроблюваного виробу вважати:
1) Справжнє технічне завдання на розробку моделі (представити у вигляді Б до пояснювальної записки до дипломної роботи);

2)Опис використаного програмного забезпечення (Представити у розділі 3 пояснювальної записки);

3) Програму та методику випробувань (представити у вигляді додатку В до пояснювальної записки до дипломної роботи);

4)Лістинг програми не приводити з причини великого об’єму.</td></tr>
<tr><td>6. Техніко-економічні показники</td><td colspan="2">1) Орієнтовна оцінка ефективності виконуваної роботи: (Представити у розділі 4 пояснювальної записки);

2) Терміни і можливі витрати грошових коштів на розробку програмного виробу: не потрібні ;

3) Порівняння економічних показників розробленого програмного вироби з вітчизняними і зарубіжними зразками або аналогами (Представити у розділі 4 пояснювальної записки);</td></tr>
<tr><td rowspan="4">7. Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>Від 01.09.21
до 07.09.21</td><td>Постановка задачі та перевірка актуальності роботи.</td></tr>
<tr><td>Від 07.09.21
до 21.09.21</td><td>Патентний пошук.</td></tr>
<tr><td>Від 21.09.21
до 01.10.21

Від 01.10.21
до 11.10.21</td><td>Розробка моделі

Дослідження моделі та її правки.</td></tr>
</table>

	Від 11.10.21 до 19.11.21 30.11.2021 Від 14.12.21 до 16.12.21	Звіт з науково-дослідницької практики Передзахист Представлення дипломної роботи комісії.
8. Порядок контролю і приймання	1) Перевірку ходу розробки програмного виробу Керівнику робіт виконувати 1 раз в 4 тижні. 2) Випробування програмного виробу провести відповідно до Програми і методики випробувань у вигляді додатку В; 3) Захист розробленого програмного виробу провести на засіданні атестаційної комісії. 4) Пояснювальну записку уявити на паперових носіях в одному примірнику, в електронному вигляді - на CD-диску в одному екземплярі.	

Виконавець
студент групи КУ-61
Шаров В. О.
_____

Замовник
к. т. н., доцент кафедри ТПС
Бердніков А.Г.
_____

Додаток В

Затверджую

_____

«_____»_____2020р.

Програма і методика випробувань виробу

« Комп’ютерна модель коригувального каскадного коду в каналах передачі даних системи управління»

1) **Об’єкт випробувань**

Об’єктом випробувань являється комп’ютерна модель систематичного каскадного коду.

2) **Мета випробувань**

Метою випробувань є:

кодування вихідної комбінації для кодів Хеммінга та циклічного коду до коду (n,m), та їх декодування;

підтвердження адекватності моделі.

3) **Вимоги до програми**

- Під час функціонування програма не повинна фатально порушувати роботу системи.
- Програма повинна видавати результат кодування до коду Хеммінга (n,m).
- Програма повинна видавати результат кодування до циклічного коду (n,m).
- Вимоги до складу і параметрів технічних засобів: Необхідний обсяг оперативної пам'яті – не менше 256 Мб, наявність миші, клавіатури.
- Вимоги до інформаційної та програмної сумісності: сумісність з ОС Windows, наявність Scilab версії не нижче 6.1.1

4) Вимоги до програмної документації

Склад програмної документації що пред’являється на випробуванні:

- Технічне завдання на програмний продукт
- Описання програми (Розділ 4 пояснювальної записки до дипломного проекту)
- Програма та методика випробувань
- Текст програми не пред’являється через великий об’єм

5) Засоби випробувань

Ноутбук Asus TUF F15.

1) Програма та методика випробувань

1) Перевірка складу та якості програмної документації проводиться візуально, на відповідність вимогам ГОСТ 19.101-77 «Види програм і програмних документів», ГОСТ 19.106-78 «Вимоги до програмних документів, виконаним друкованим способом».

2) Програма працює у відповідності з умовами експлуатації ОС Windows 7,8,10, на ПК будь-якого типу.

3) Тестування програмної моделі

Методика випробувань здійснюється за допомогою порядку проведень випробувань:

1. Запуск програми відбувається шляхом виконання скрипту програми .
2. Введення вхідних даних для моделювання, виконується шляхом написання у скрипті необхідних даних, введених з клавіатури .
3. Проводиться обробка даних (програма виробляє дії за рішенням завдання управління матеріальними ресурсами)
4. Виведення результатів обробки (в графічному вигляді рішення задачі представляється у вигляді формул та виразів, так само видаються проміжні розрахунки для кожного етапу функціонування моделі каскадного коду)

Для проведення випробувань пропонуються вихідні дані:

-для коду Хеммінга G(x) = [0010].

-для циклічного коду G(x) = [1100]. P(x) = $x^3 + x + 1$

Вміст тестових файлів і результати роботи програми:

Оскільки випробовуватись будуть дві моделі, відповідно, результатів по випробуванню також буде два.

Етап 1 (для циклічного коду) . Запуск програми.

Одразу після запуску на екрані з’являється загальний скрипт.

```
//Note: Details of scilab software version and OS version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and 64 bit versions)
//Program Title: Cyclic Codes (7,4)

clc;
clear;
k = 4; //Information Message Length
n = 7; //Codeword Length
timer()
//Generator Polynomial
x=poly(0,'x');
GenPoly=1+x+x^3;
disp(GenPoly,'Образуючий поліном P');

//Generating Random Message

//All_M = All 16 posibilities for Information Message Matrix
All_M = [0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;
0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;
1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;
1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1]

RandMessage=modulo(round(16*rand()),16)+1//Get random number between 1 to 16

M=All_M(RandMessage,:)//Select a random row from Message Matrix All_M as Information Message
disp(M,'Information Message M')

//Message Polynomial
MesPoly=(M(1)*1) + (M(2)*(x^1)) + (M(3)*(x^2)) + (M(4)*(x^3));
disp(MesPoly,'Інформаційний поліном G');

//Encoding----------------------------------------------------------------------------

//Generating Codeword Polynomial
p=(x^(n-k))*(MesPoly);//Step 1 - multiply MesPoly by x^(n-k), [x^(n-k)*u(x)]
[RemPoly,q]=pdiv(p,GenPoly);//Step 2 - divide above product by GenPoly, g(x) (Polynomial Division)
RemPoly=modulo(RemPoly,2);//Convert Remainder Polynomial to binary to get parity check polynomial, b(x)
//disp(RemPoly,'Remainder Polynomial b(x)');
CodePoly=RemPoly+(MesPoly*(x^(n-k)));//Step 3 - add (x^(n-k)*u(x)) and b(x) to get Codeword Polynomial
```

Рисунок В.1 – Початок головного скрипта для циклічного коду

Етап 2 (для циклічного коду) . Введення даних.

До програми вводяться необхідні вихідні дані.

```
 1 +x +x³

"Образуючий поліном P"
```

```
 x² +x³

"Інформаційний поліном G"
```

Рисунок В.2 – Введення даних для циклічного коду

Етап 3 (для циклічного коду) . Кодування.

На основі вихідних даних програма домножує комбінацію на 10^3 та генерує поліном F(x)

```
 1 +x⁵ +x⁶

"Інформаційний поліном G у ступені *x^3"

 1.    0.    0.    0.    0.    1.    1.

"Сгенерований поліном F"
```

Рисунок В.3 – Кодування циклічного коду

Етап 4 (для циклічного коду) . Симуляція помилки, декодування.

Надалі було допущено, що при передачі трапилась помилка у деякому розряді комбінації, програма симулювала помилку. Після симуляції помилки, кодова комбінація була декодована, було отримано $S \neq [0\ 0\ 0]$, що вказало на наявність помилки, а також вказало на номер помилкового розряду.

```
 0.   0.   0.   0.   0.   1.   1.

"Поліном F з урахування помилки 0"

 1.   0.   0.

"Синдром S"

 1.

"Номер розряду помилки"
```

Рисунок В.4 – Симуляція помилки, декодування циклічного коду

Після виявлення номеру помилкового розряду, комбінація була виправлена та декодована.

Надалі приводиться випробування для коду Хеммінга.

Етап 1 (для коду Хеммінга) . Запуск програми.

Одразу після запуску на екрані з’являється загальний скрипт.

```
//Note: Details of scilab software version and OS version used:
//Tested on OS: Windows 7 SP1, 64 bit and Windows XP SP3, 32 bit
//Scilab version: 5.4.1 (Tested on both 32 bit and 64 bit versions)
//Program Title: Hamming Codes (7,4)

clc;
clear;
timer()

k = 4; //Information message matrix length
n = 7; //Coded word length

P = [1 1 0;0 1 1 ;1 1 1;1 0 1]//Parity Matrix
disp(P,'Правила перевірки (перевірочна підматриця)Hp')

G = [P eye(k,k)]//Generator Matrix to create code word in P1P2P3D1D2D3D4 format
G(:,[3 4])=G(:,[4 3])//Swap colomn 3 and 4 of G to create code word in P1P2D1P3D2D3D4 format
disp(G,'Породжуюча матриця P')

H=[eye(n-k,n-k);P]'//Parity Check Matrix
H(:,[3 4])=H(:,[4 3])//Swap colomn 3 and 4 of H to satisfy GH'=0
disp(H,'Перевірочна матриця H')

//disp(modulo(G*H',2),'GH')//Check if the condition GH'=0 satisfy (for testing)

//M = [1 1 0 1]//Information Message Matrix for testing

//Generate random message
//All_M = All 16 posibilities for Information Message Matrix
All_M = [0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;
0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;
1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;
1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1]
RandMessage=modulo(round(16*rand()),16)+1//Get random number between 1 to 16
M=All_M(RandMessage,:)//Select a random row from 1 to 16 as Information Message

disp(M,'Інформаційне повідомлення G(x)')

C = M*G;//Generate code word
C = modulo(C,2);//Convert generated code into binary
```

Рисунок В.5 – Початок головного скрипта для коду Хеммінга

Етап 2 (для коду Хеммінга). Введення даних.

До програми вводяться необхідні вихідні дані.

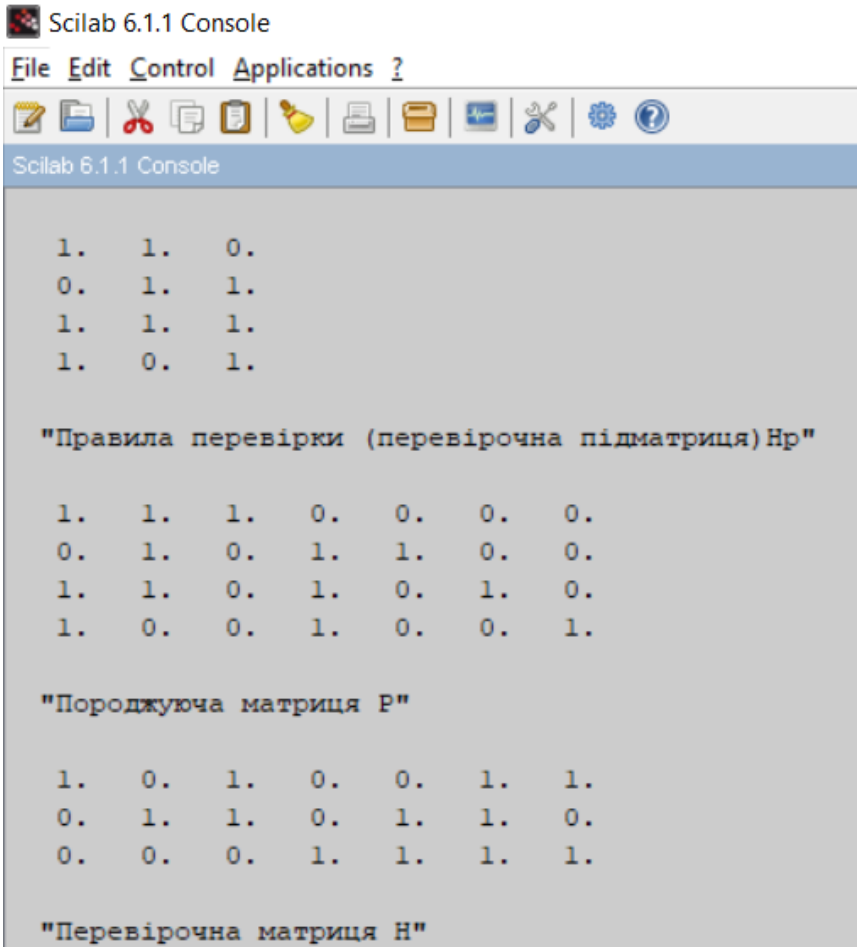

Рисунок В.6 – Введення даних для коду Хеммінга

Етап 3 (для коду Хеммінга) . Кодування.

На основі вихідних даних програма генерує поліном F(x)

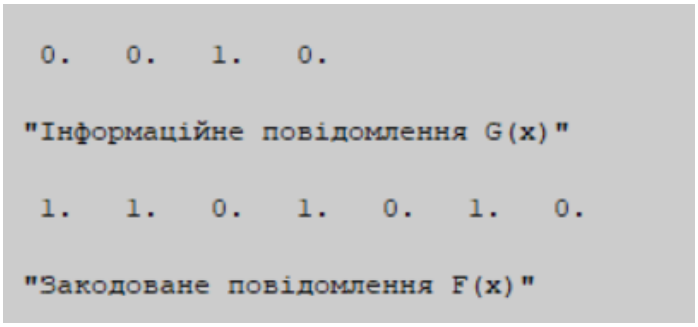

Рисунок В.7 – Кодування для коду Хеммінга

Етап 4 (для коду Хеммінга) . Симуляція помилки, декодування.

Надалі було допущено, що при передачі трапилась помилка у деякому розряді комбінації, програма симулювала помилку. Після симуляції помилки, кодова комбінація була декодована, було отримано $S \neq [0\ 0\ 0]$, що вказало на наявність помилки, а також вказало на номер помилкового розряду.

```
 1.   1.   0.   1.   0.   0.   0.

"Прийнята комбінація з урахуванням можливої помилки 0"

 1.   0.   1.

"Синдром"

 6.

"Номер помилкового розряду"
```

Рисунок В.8 – Симуляція помилки для коду Хеммінга

Після виявлення номеру помилкового розряду, комбінація була виправлена та декодована.

5. Висновок

Проведенні випробування повністю підтверджують працездатність моделі та її адекватність, коректність кодування та декодування даних.

Виконавець:

студент Шаров Владислав Олегович