Пояснювальна записка
до дипломного проекту
Молнара Ервіна Ервіновича
на тему: «Цифрова система передачі даних з завадостійким кодуванням»

Київ – 2019 рік

ЗМІСТ

ВСТУП

Життя сучасного суспільства тісно пов’язане з обміном інформації. На сьогодні передавання інформації (зв’язок) є одним із найбільш дійових важелів управління економікою як усього світу, так і окремих країн.

Темпи зростання економічної, соціальної і політичної діяльності суспільства пропорційні темпам зростання сумарного трафіка в каналах обміну інформацією. Якщо з 1997 по 2003 рр. сумарний трафік зріс з 0,8 до 1,0 Тбіт/с, то за останні 5 років він досягнув 9,0 Тбіт/с. Такими можливостями в темпах зростання обсягів надання телекомунікаційних послуг в обміні інформацією галузь зв’язку зобов’язана трьом фундаментальним відкриттям другої половини XX століття:

– закону Мура, який в 70-ті роки передбачив подвоєння продуктивності інтегральних схем кожні 18 місяців зменшенням їхньої вартості більш ніж у два рази за цей самий час;

– теоретичне обґрунтування можливості цифровізації будь-яких видів аналогових сигналів (теорема В.А. Котельникова);

– впровадження на телекомунікаційних мережах волоконнооптичних технологій, які забезпечують більшу пропускну здатність. Значна частина в загальному обміні інформацією належить трафіку даних, під яким насамперед розуміють трафік, що визначається обсягом обміну інформацією в цифровому вигляді.

Актуальність теми

Проблеми надійного й ефективного передавання інформації пов’язані з використанням ефективних кодів. Розвинена в роботах В.А. Котельникова і К.Є. Шеннона теорія передавання повідомлень оцінила граничні можливості і тим самим ініціювала виклик дослідникам, які прагнуть наблизитися до цих границь.

Мета і задачі дослідження

Метою дипломної роботи є дослідження та розробка цифрової системи передачі даних з завадостійким кодуванням.

Об’єкт і предмет дослідження

Об’єктом дослідження є система для передачі даних, за допомогою якої можна передавати дані на відстань до 1 км. Предметом дослідження є кодування даних, за допомогою якого контролюється цілісність даних.

Наукова новизна одержаних результатів

Новизна даної роботи полягає у аналізі існуючих рішень і створенню діючої системи для передачі даних.

Практичне значення наукових результатів

Результати дослідження можна використовувати при створенні систем, які повинні бути максимально надійні. Створена модель демонструє цю надійність.

Структура роботи

У розділі 1 представлено постановку задачі.

У розділі 2 представлено дослідження різних систем передачі даних та їх переваг і недоліків.

У розділі 3 наведені існуючи рішення щодо завадостійкої передачі даних.

У розділі 4 запропоновано модель цифрової системи передачі даних.

У розділі 5 представлено цифрову систему передачі даних з завадостійким кодуванням.

1. ПОСТАНОВКА ЗАДАЧІ

Метою дипломної роботи є дослідження та розробка цифрової системи передачі даних з завадостійким кодуванням. Для досягнення даної цілі було вирішено наступні завдання:

1. Дослідження проблеми завадостійкої передачі даних.
2. Огляд існуючих рішень для передачі даних.
3. Розробка цифрової системи яка здатна передавати дані використовуючи завадостійке кодування.

2. ПЕРЕДАЧА ДАНИХ

2.1. Фізичні середовища передачі даних

Лінії зв'язку відрізняються фізичним середовищем, яке використовується для передачі інформації. Фізичне середовище передачі даних може являти собою набір провідників, по яких передаються сигнали. На основі таких провідників будуються повітряні або кабельні лінії зв'язку (рисунок 2.1). Як середовище також використовується земна атмосфера або космічний простір, через яке поширюються інформаційні сигнали. У першому випадку говорять про провідні середовища, а в другому - про бездротові (рисунок 2.1).

Рисунок 2.1 – Типи середовищ передачі даних

В сучасних телекомунікаційних системах інформація передається за допомогою електричного струму або напруги, радіосигналів або світлових сигналів - всі ці фізичні процеси представляють собою коливання електромагнітного поля різної частоти.

2.2. Повітряні лінії зв’язку

Повітряні лінії зв'язку являють собою проводи без будь-яких ізоляторів або екранів, прокладені між стовпами і висять у повітрі. Ще в недалекому минулому такі лінії зв'язку були основними для передачі телефонних або телеграфних сигналів. Сьогодні провідні лінії зв'язку швидко витісняються кабельними. Але подекуди вони все ще збереглися і при відсутності інших можливостей продовжують використовуватися, зокрема, і для передачі комп'ютерних даних. Швидкісні якості і перешкодозахищеність цих ліній залишають бажати кращого.

2.3. Кабельні лінії зв’язку

Кабельні лінії мають досить складну конструкцію. Кабель складається з провідників, укладених в кілька шарів ізоляції: електричної, електромагнітної, механічної і, можливо, кліматичної. Крім того, кабель може бути оснащений роз'ємами, що дозволяють швидко виконувати приєднання до нього різного обладнання. У комп'ютерних (і телекомунікаційних) мережах застосовуються три основні типи кабелю: кабелі на основі скручених пар мідних проводів - неекранована кручена пара (Unshielded Twisted Pair, UTP) і екранована кручена пара (Shielded Twisted Pair, STP), коаксіальні кабелі з мідною жилою, волоконнооптичні кабелі. Перші два типи кабелів називають також мідними кабелями.

2.4. Радіоканали

Радіоканали наземного і супутникового зв'язку утворюються за допомогою передавача і приймача радіохвиль. Існує велика різноманітність типів радіоканалів, що відрізняються як використовуваним частотним діапазоном, так і дальністю каналу. Діапазони широкомовного радіо (довгих, середніх і коротких

хвиль), звані також АМ-діапазонами, або діапазонами амплітудної модуляції (Amplitude Modulation, AM), забезпечують телекомунікацію при невисокій швидкості передачі даних. Більш швидкісними є канали, які використовують діапазони дуже високих частот (Very High Frequency, VHF), для яких застосовується частотна модуляція (Frequency Modulation, FM). Для передачі даних також використовуються діапазони ультрависоких частот (Ultra High Frequency, UHF), звані ще діапазонами мікрохвиль (понад 300 МГц). При частоті понад 30 МГц сигнали вже не відображаються іоносферою Землі і для стійкого зв'язку потрібна наявність прямої видимості між передавачем і приймачем. Тому зазначені частоти використовуються в супутникових або радіорелейних каналах або в таких локальних або мобільних мережах, в яких ця умова виконується.

2.5. Основні характеристики ліній зв’язку

Основною характеристикою ліній зв'язку є коефіцієнт ослаблення [дБ / км], який показує в децибелах наскільки зменшується абсолютний рівень потужності сигналу при проходженні одного кілометра шляху.

У радіоліній коефіцієнт ослаблення залежить від напрямку та величини пройденого сигналом шляху. Для направляючих систем коефіцієнт ослаблення (в певному частотному діапазоні) можна вважати постійним.

Мінімальними значеннями коефіцієнта ослаблення володіють світловоди, для них [дБ/км]=0,2, для хвилеводів [дБ/км]=2-3 і для мідних кабелів [дБ/км]=1-10.

Важливою характеристикою є також смуга пропускання, під нею розуміють діапазон частот електричних коливань, що проходять через лінію зв'язку з мінімальним ослабленням в [Гц]. Мінімальною смугою пропускання володіють повітряні лінії зв'язку, у них 100-150 кГц. Максимальна смуга пропускання є гідністю світловодів: 100-1000 ТГц (террагерц).

Еквівалентом смуги пропускання для цифрових систем і ліній телекомунікацій часто виступає пропускна здатність у бітах в секунду [біт / с]. Вона визначає максимальну швидкість передачі цифрового сигналу.

Величина пропускної спроможності в 2 Мбіт / с умовно ділить лінії на вузькосмугові (наприклад, симетричні кабелі) і широкосмугові (наприклад, коаксіальні кабелі і світловоди).

Іншою найважливішою характеристикою ліній зв'язку є чутливість до шумів і відповідно високий темп помилок при передачі цифрових сигналів. Найбільшою чутливістю до шумів і зовнішніх впливів володіють симетричні пари. У коаксіального кабелю і скловолокна чутливість до шумів і зовнішніх впливів на порядок менше, що пояснюється більш вдалою конструкцією. У радіоліній темп помилок визначається умовами передачі: погодою, наявністю перешкод у вигляді листя і будівель. Основні характеристики ліній зв'язку наведені в таблиці нижче (таблиця 2.1).

Таблиця 2.1 Порівняння характеристик ліній зв’язку.

Тип лінії	Смуга пропускання	Період повтору помилки в бітах
Симетрична пара в телефонному кабелі	1 МГц	10^5
Коаксіальний кабель	1 ГГц	10^7 - 10^9
Мікрохвильовий радіозв’язок	100 ГГц	10^9
Супутниковий зв’язок	100 ГГц	10^9
Оптоволокно	75 ТГц	10^{11} - 10^{13}

3. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

3.1. Завадостійке кодування

Завадостійке кодування - кодування, призначене для передачі даних по каналах з перешкодами, що забезпечує виправлення можливих помилок передачі внаслідок перешкод.

Для виявлення помилок використовують коди виявлення помилок, для виправлення - перешкодостійкі коди.

3.2. Класифікація завадостійких кодів

У безперервних кодах передана інформаційна послідовність не поділяється на блоки. Надлишкові елементи розміщуються в певному порядку між інформаційними.

Рівномірні блокові коди діляться на роздільні та нероздільні. У роздільних кодах елементи інформаційної та перевірочної частин кодової комбінації завжди стоять на певних місцях. У нероздільних кодах поділ на інформаційні та перевірочні розряди відсутня.

Роздільні коди, в свою чергу, діляться на систематичні (лінійні) і несистематичні (нелінійні). Систематичними кодами називаються блокові роздільні (n, k) -коди, в яких перевірочні елементи являють собою лінійні комбінації інформаційних, несистематичні коди такою властивістю не володіють.

Завадостійке кодування передбачає введення в передане повідомлення, поряд з інформаційними, так званих перевірочних розрядів, які формуються в пристроях захисту від помилок (кодерах - на передавальному кінці, декодерах - на приймальному). Надмірність дозволяє відрізнити дозволену і заборонену (перекручену за рахунок помилок) комбінації при прийомі, інакше одна дозволена комбінація переходила б до іншої.

Перешкодостійкий код характеризується трійкою чисел (n, k, d0), де n- загальне число розрядів в переданому повідомленні, включаючи перевірочні (r), k = nr - число інформаційних розрядів, d0- мінімальна кодова відстань між дозволеними кодовими комбінаціями, яке визначається як мінімальне число розрізняються біт в цих комбінаціях. Число виявлених (tj і (або) виправляються (t) помилок (розрядів) пов'язане з параметром d0 співвідношенням.

Іноді використовуються додаткові показники надмірності, похідні від наведених вище характеристик n, k: R = r / n- відносна надмірність, v = k / n - відносна швидкість передачі.

Рисунок 3.1 – Надлишкові коди

Існуючі перешкодостійкі коди можна розділити на ряд груп (рисунок 3.1), тільки частина з яких використовується для виявлення помилок в переданих

по мережі пакетах. У групі систематичних (лінійних) кодів загальною властивістю є те, що будь-яка дозволена комбінація може бути отримана в результаті лінійних операцій над лінійно-незалежними векторами (рисунок 3.2). Це сприяє спрощенню апаратної і програмної реалізації даних кодів, підвищує швидкість виконання необхідних операцій.

Рисунок 3.2 – Перешкодостійкі коди

3.3. Інтерфейси передачі даних

Розглянемо такі інтерфейси:

— CAN.

— RS-485.

— Струмова петля.

— RS-232.

— RS-422.

3.3.1. Інтерфейс CAN

CAN є послідовною шиною, що підтримує одночасну роботу багатьох ведучих пристроїв. Це означає, що всі вузли CAN-мережі мають можливість передавати дані і декілька вузлів одночасно можуть давати запит на шину.

На ринку CAN присутній у двох версіях: версія 2.0A задає 11-бітову ідентифікацію повідомлень (тобто в системі може бути 2048 учасників), версія 2.0B — 29-бітову (536 млн учасників). Слід відзначити, що версія 2.0B, яку часто називають FullCAN, поступово витісняє версію 2.0A, яку називають, також, BasicCAN.

Магістраль CAN використовує двопровідникову звиту пару і працює з максимальною швидкістю трансляції 1 Мбіт/с на довжині до 40 м. Із збільшенням відстані зменшується максимальна швидкість трансляції (напр. 250 кбіт/с до 250 м). На рисунку 3.3 показано рівні сигналів.

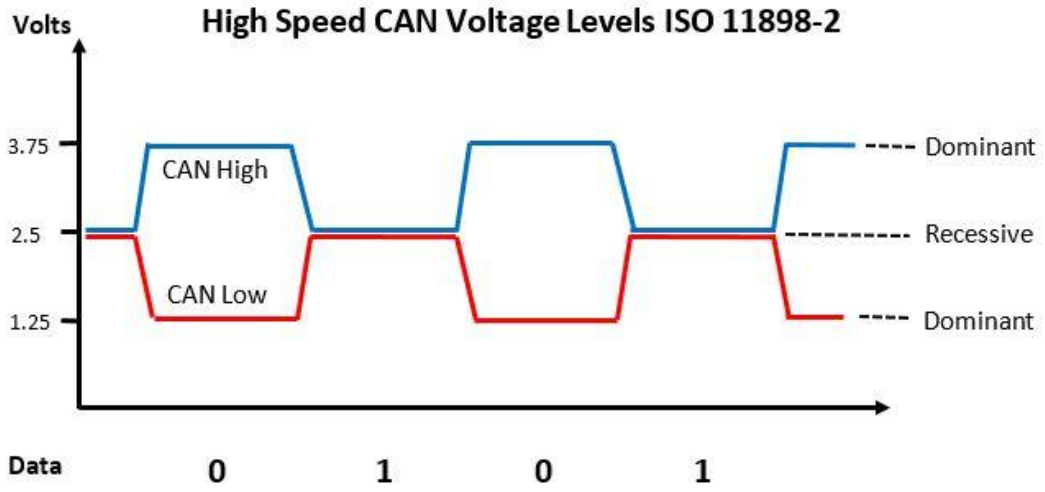

Рисунок 3.3 – Рівні сигналів на лініях CAN

3.3.2. Інтерфейс RS-485

RS-485 забезпечує передачу даних з швидкістю до 10 Мбіт/с. Максимальна дальність залежить від швидкості: при швидкості 10 Мбіт/с максимальна довжина лінії – 120 м, при швидкості 100 кбіт/с – 1200 м.

Кількість пристроїв, що підключаються до однієї лінії інтерфейсу, залежить від типу застосованих в пристрої приймачів. Один передавач розрахований на управління 32 стандартними приймачами. Випускаються приймачі з вхідним опором ½, ¼, 1/8 від стандартного. При використанні таких приймачів загальне число пристроїв може бути збільшене відповідно: 64, 128 або 256.

У стандарті RS-485 для передачі і прийому даних часто використовується одна і та ж вита пара дротів. Передача даних здійснюється за допомогою диференціальних сигналів. Різниця напруги однієї полярності між провідниками означає логічну одиницю, різниця іншої полярності – нуль (рисунок 3.4).

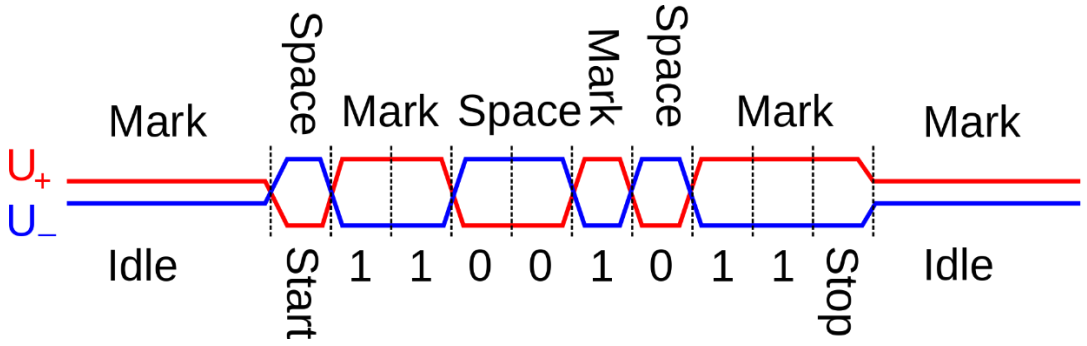

Рисунок 3.4 – Рівні сигналів на лініях RS-485

3.3.3. Інтерфейс струмова петля

Струмова петля – спосіб передачі інформації за допомогою вимірюваних значень сили електричного струму. В даний час такий спосіб більш поширений в інженерній практиці, ніж використання для цієї мети напруги. Для завдання вимірюваних значень струму використовується, як правило, кероване джерело струму. По виду передачі інформації розрізняються аналогова струмова петля і цифрова струмова петля.

Стандарт цифрової струмової петлі використовує відсутність струму як значення SPACE (низький рівень, логічний нуль) і наявність сигналу - як значення MARK (високий рівень, логічна одиниця). Відсутність сигналу протягом тривалого часу інтерпретується як стан BREAK (обрив лінії). Дані передаються старт-стопним методом, формат посилки збігається с RS-232, наприклад 8-N-1: 8 біт, без паритету, 1 стоп-біт.

Струмова петля (рисунок 3.5) може використовуватися на значних відстанях (до кількох кілометрів). Для захисту устаткування застосовується гальванічна розв'язка на оптоелектронних приладах, наприклад оптронах.

Через неідеальної джерела струму, максимально допустима довжина лінії (і максимальний опір лінії) залежить від напруги, від якого живиться джерело струму. Наприклад при типовому напрузі живлення 12 вольт опір не повинно перевищувати 600 Ом.

Джерело струму може розташовуватися в приймальному або передавальному кінці струмового петлі. Вузол з джерелом струму називають активним. Залежно від конструкції як передавач, так і приймач, можуть бути або активними (живити струмовий петлю), так і пасивними (харчуватися від струмового петлі).

Для комп'ютерів сімейства ДВК за замовчуванням приймається, що передавач - активний, приймач - пасивний.

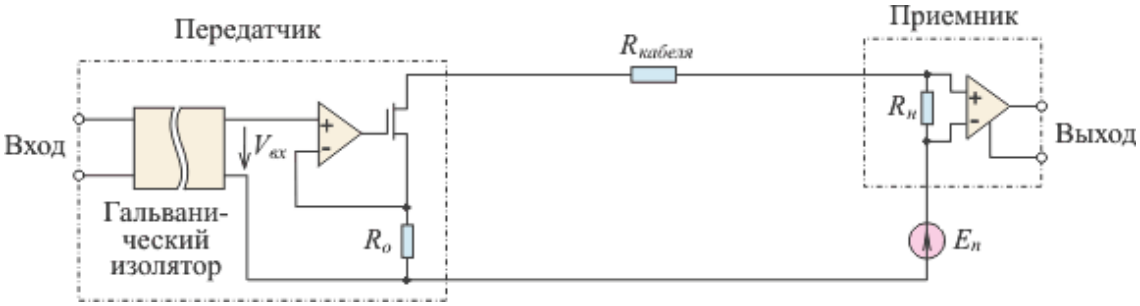

Рисунок 3.5 – Інтерфейс струмова петля

3.3.4. Інтерфейс RS-232

RS-232 – стандарт інтерфейсу обміну даними між двома пристроями шляхом послідовної передачі даних (асинхронний зв'язок або синхронний зв'язок), знаходить використання у послідовних портах комп'ютерів та інших пристроях.

В стандарті RS-232 дані передаються у вигляді часової послідовності бітів.

Стандарт RS-232 визначає рівні напруги, що відповідають логічним рівням «0» і «1». Допустимий діапазон значень напруги для логічної «1» складає від -3 до -15 В, для логічного «0» — від 3 до 15 В відносно загальної землі («GND») (рисунок 3.6). Значення напруги від −3 до 3 В є забороненими, це зроблено для покращення завадостійкості передачі .

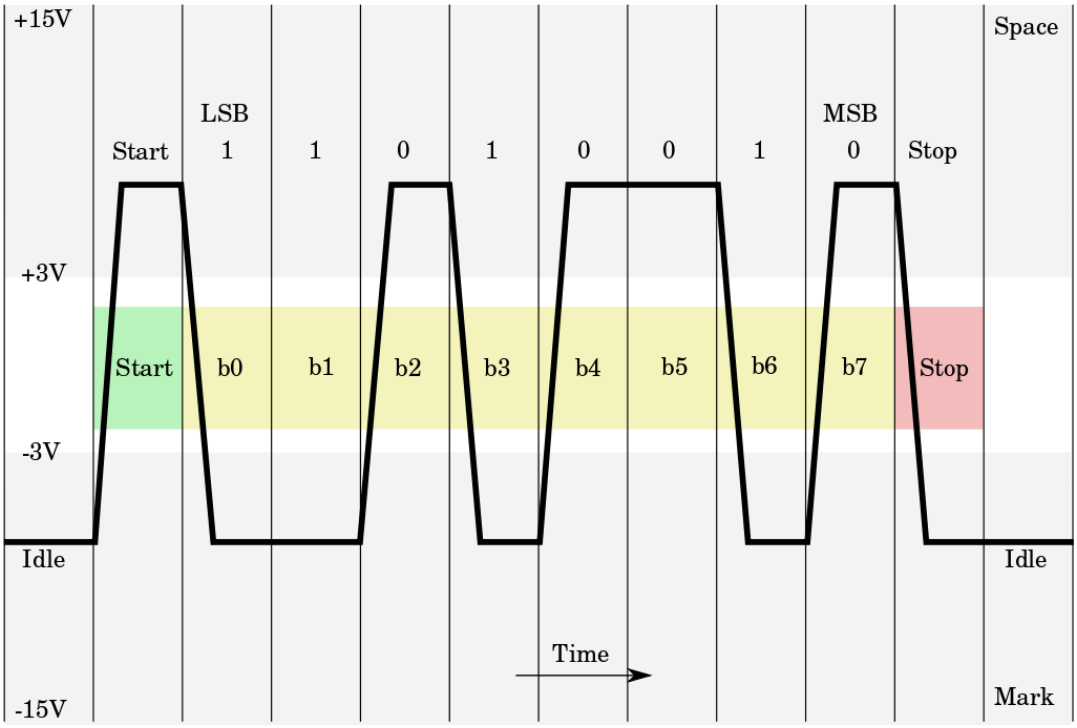

Рисунок 3.6 – Осцилограма рівнів напруги при передачі символу «K» (0x4B в ASCII). 1 старт-біт, 8 біт даних і 1 стоп-біт.

3.3.5. Інтерфейс RS-422

RS-422 – американський стандарт, його міжнародний еквівалент — рекомендація ITU-T V.11 (він же X.27). Цей технічний стандарт забезпечує збалансовану або диференціальну однонаправлену передачу даних без реверсу по термінованих або нетермінованих лініях, з можливістю з'єднання «точка-точка» або для багатоабонентської доставки повідомлень.

На відміну від RS-485, який забезпечує багатоточкову топологію, RS-422/V.11 не дозволяє мати декількох відправників, однак дозволяє мати декількох отримувачів повідомлень (рисунок 3.7).

Відзнакою цього стандарту є швидкість передачі даних до 10 мегабіт за секунду для кабелю довжиною 12 метрів. Хоча специфікація стандарту не встановлює верхню межу, в ній наведено діаграму затухання сигналу зі збільшенням довжини кабеля. Діаграма закінчується на 10 Мбіт/с.

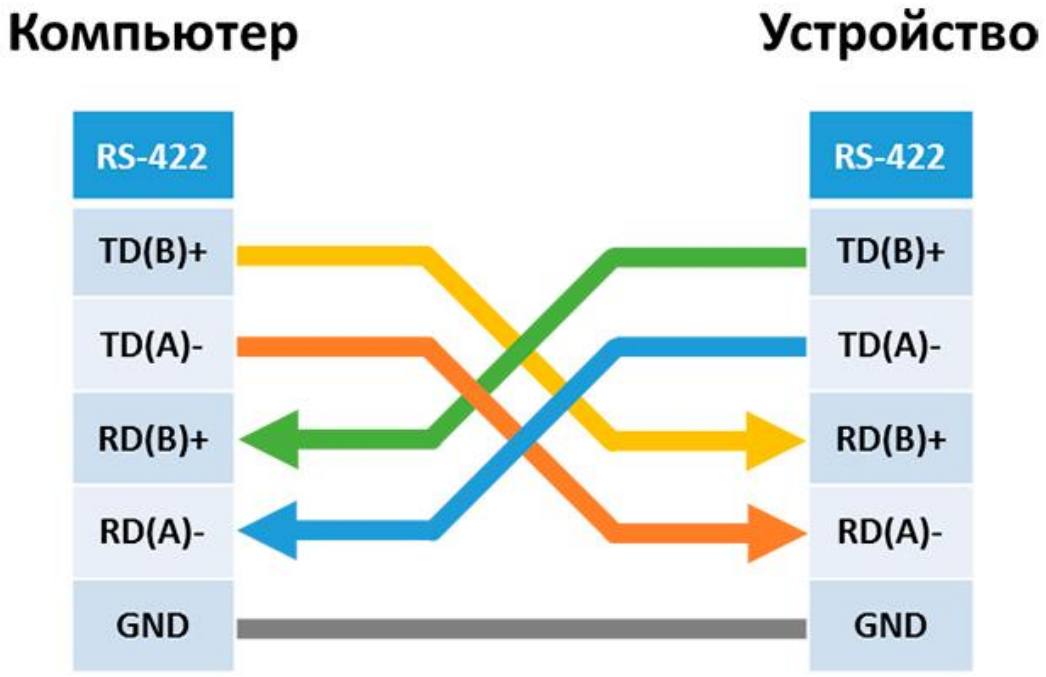

Рисунок 3.7 – Схема підключення по RS-422.

3.3.6. Порівняння RS-232, RS-422 та RS-485

В таблиці 3.1 порівняно характеристики RS-232, RS-422 та RS-485.

Таблиця 3.1 Порівняння характеристик RS-232, RS-422 та RS-485.

Назва	RS-232	RS-422	RS-485
Тип передачі	Повний дуплекс	Повний дуплекс	Напівдуплекс (2 дроти), повний дуплекс (4 дроти)
Максимальна відстань	15 метрів при 9600 біт/с	1200 метрів при 9600 біт/с	1200 метрів при 9600 біт/с
Задіяні контакти	TxD, RxD, RTS, CTS, DTR, DSR, DCD, GND	TxA, TxB, RxA, RxB, GND	DataA, DataB, GND
Топологія	Точка-точка	Точка-точка	Багатоточкова
Максимальна кількість пристроїв	1	1 (10 пристроїв в режимі прийому)	32 (з повторювачами більше, зазвичай до 256)

4. МОДЕЛЬ ЦИФРОВОЇ ПЕРЕДАЧІ ДАНИХ

4.1. Лінія зв’язку

Для даної системи вибрано кабельну лінію зв’язку – неекрановану кручену пару, оскільки вона має непогані характеристики та невисоку ціну.

4.2. Протокол передачі даних

Для нашої системи вибрано протокол RS485, оскільки він забезпечує достатню дальність (до 1200м), підтримує достатню кількість пристроїв (до 256) та має непогану завадостійкість.

4.3. Мікроконтролер

Для даної системи вибрано мікроконтролери лінійки STM32F1, а саме STM32F103C8T6 (рисунок 4.1, таблиця 4.1).

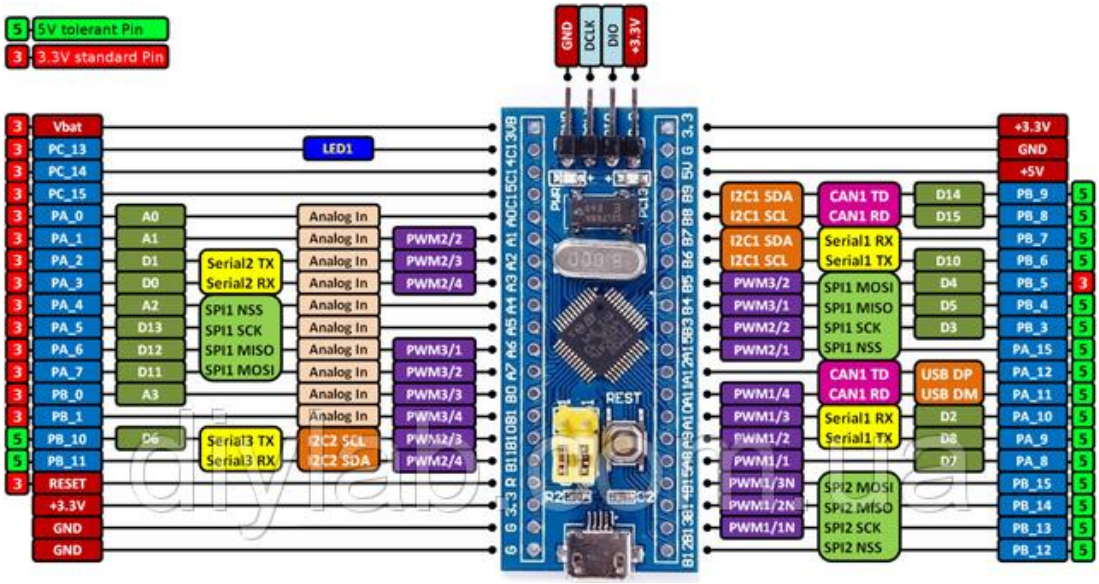

Рисунок 4.1 – Плата з STM32F103C8T6.

Таблиця 4.1 Основні характеристики мікроконтролера STM32F103C8T6.

Максимальна тактова частота	72 МГц
Flash пам’ять	64 або 128 кб
SRAM пам’ять	20 кб
Напруга живлення	2.0 – 3.6 В
Кварцевий резонатор	4 – 16 МГц
АЦП	16 каналів, 12біт
DMA	7 каналів
Входи / виходи	26/37/51/80
Відладка	SWD, JTAG
Таймери	7
I2C	2
USART	3
SPI	2
CAN	1
USB	1
Унікальний ID	96 біт
Підтримка низького споживання	Так

4.4. Модуль UART-RS485

Даний модуль (рисунок 4.2, рисунок 4.3) використовується для підключення мікроконтролерів по інтерфейсу RS-485 (рисунок 4.4). Напруга живлення: 5 В.

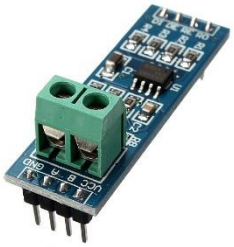

Рисунок 4.2 – модуль UART-RS485.

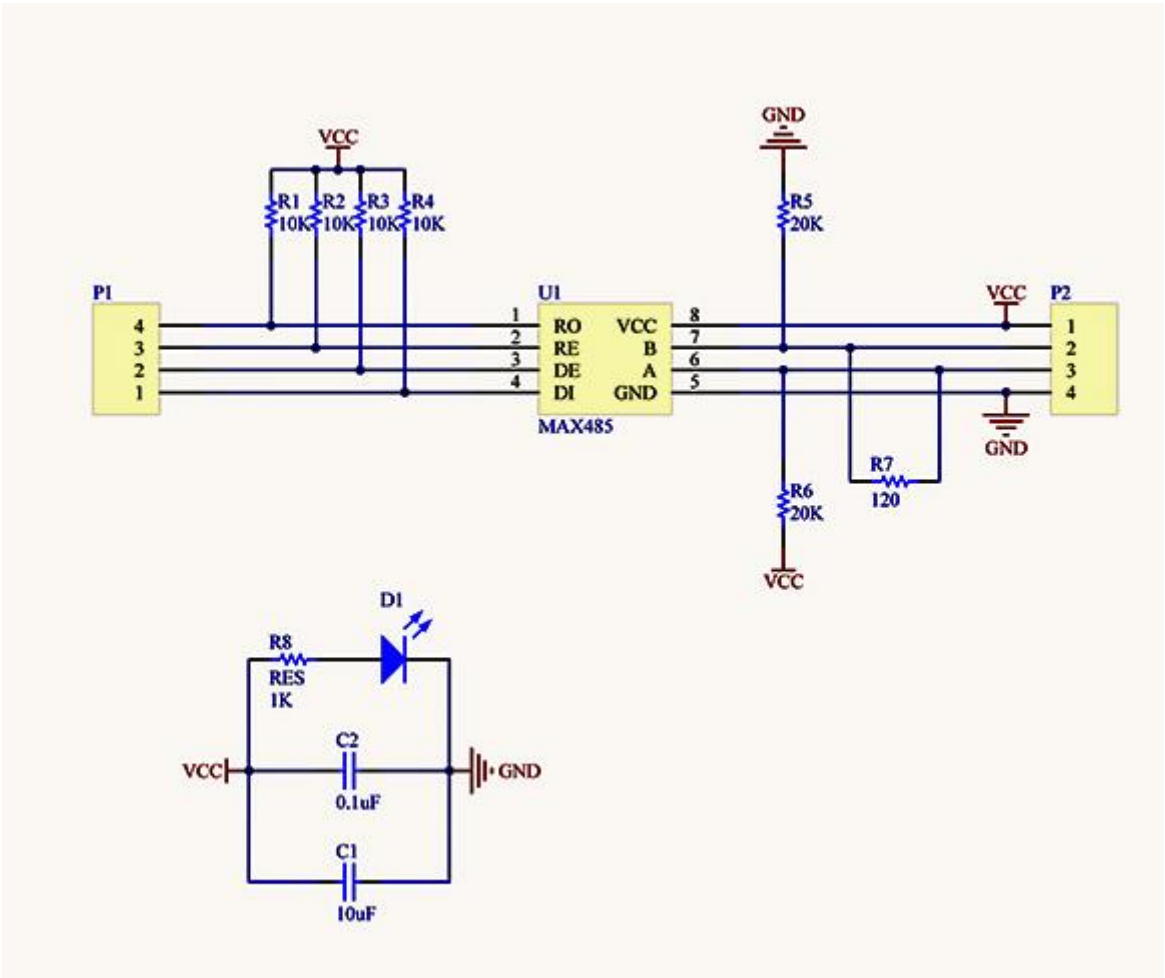

Рисунок 4.3 – схема модуля UART-RS485.

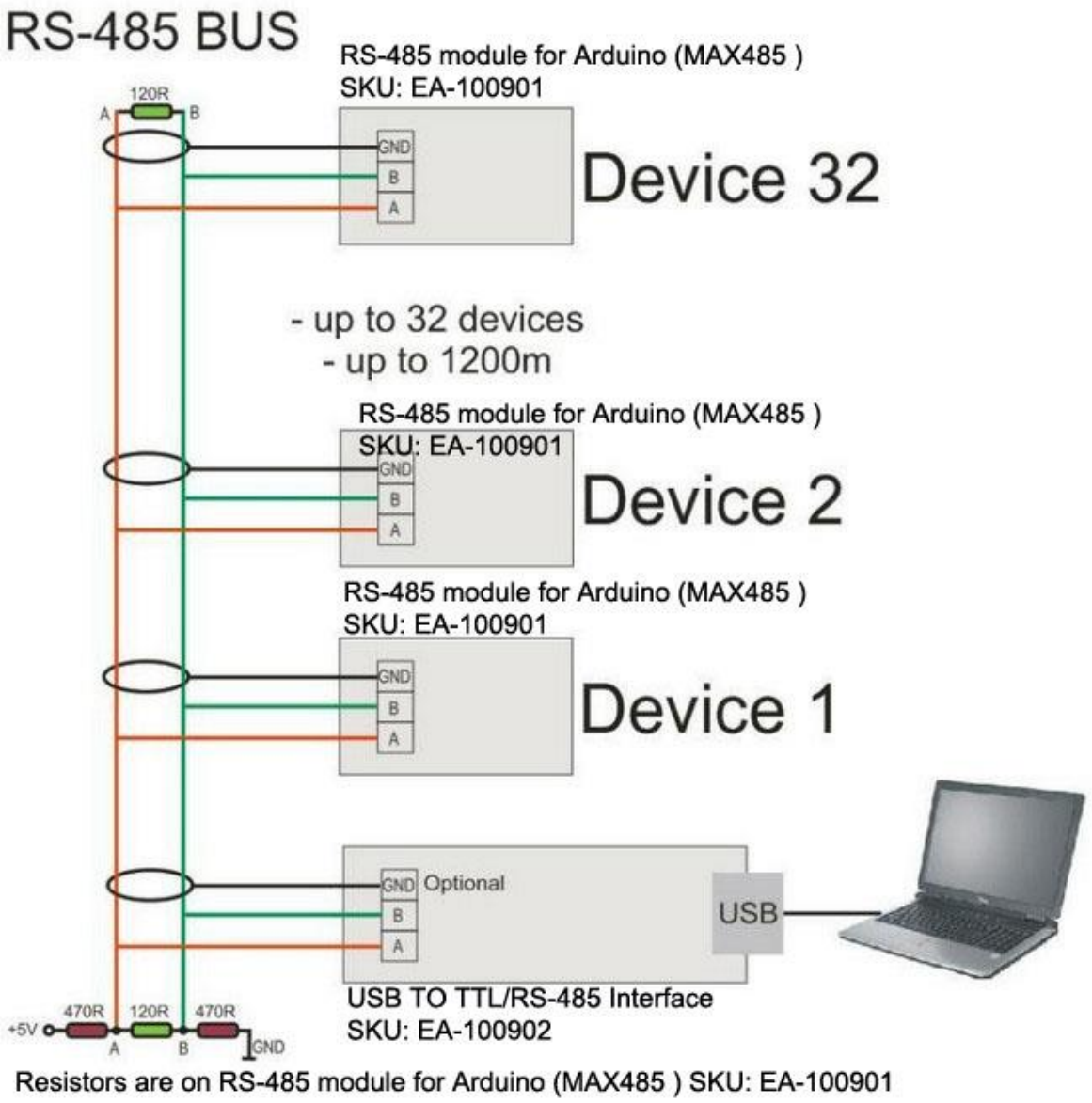

Рисунок 4.4 – приклад топології RS485.

4.5. Клавіатура

Для проекту використано мембранну матричну клавіатуру 3x4 (рисунок 4.5).

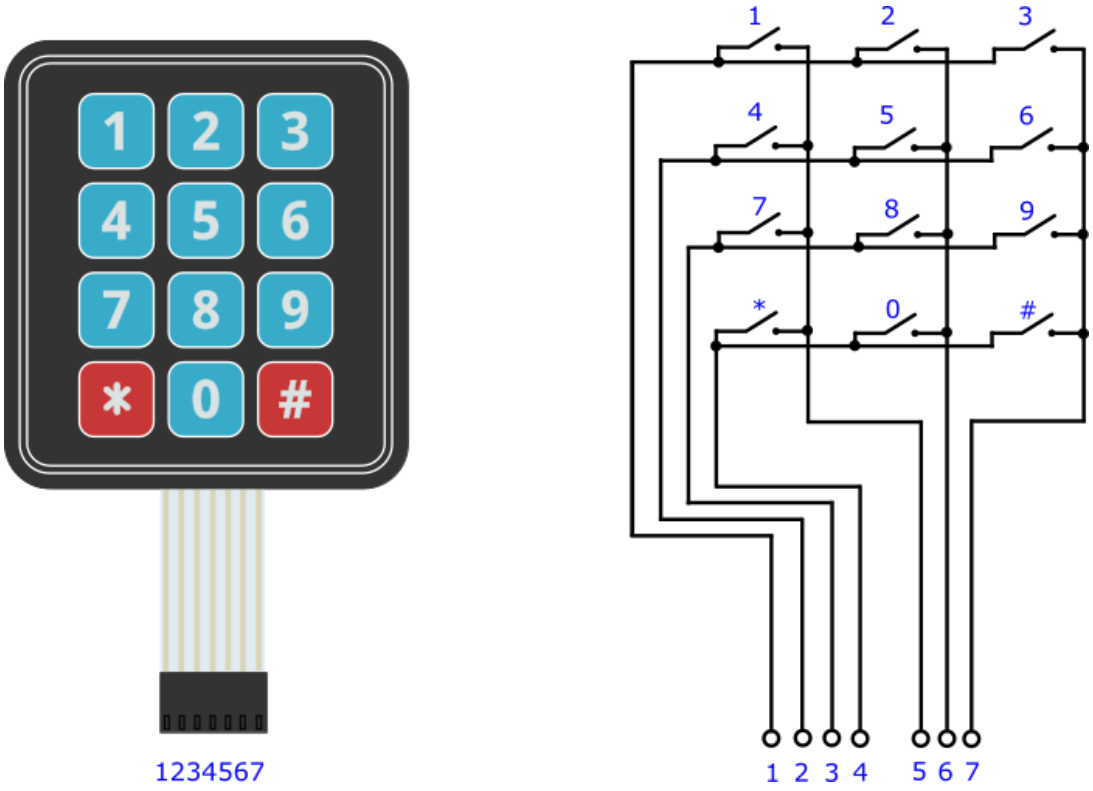

Рисунок 4.5 – Матрична клавіатура 3x4.

4.6. Дисплей

Для відображення інформації в якості дисплею використано чотирьохрозрядний семисегментний індикатор на базі контролера TM1637 (рисунок 4.6, таблиця 4.2).

Таблиця 4.2 основні характеристики індикатора.

Напруга живлення	5В
Колір індикації	Червоний
Контролер	TM1637
Інтерфейс	Двопровідний (DIO/CLK)
Розміри плати	42x23мм
Розмір цифри	0.36”

Рисунок 4.6 – Індикатор на TM1637.

4.7. Звуковий сигнал

Для створення звукового сигналу використано модуль п'єзодинаміка (рисунок 4.7) на 3.3В. Основою даного модулю є пасивний п’єзодинамік, завдяки чому на ньому можна відтворювати різні тональності.

Рисунок 4.7 – Модуль п'єзодинаміка.

4.8. Модуль USB-RS485

Для прослуховування лінії RS-485 використано модуль USB-RS485 (рисунок 4.8). Цей модуль в диспетчері пристроїв відображається як COM-порт. Для відображення даних можна використовувати будь-яку програму для роботи с COM-портами. Наприклад, Termite або Putty.

Рисунок 4.8 – Модуль USB-RS485.

4.9. Схема системи

Система складається з трьох модулів: Модуль А, модуль В, модуль С (рисунки 4.9, 4.10, 4.11). Модулі під’єднані між собою за допомогою крученої пари по інтерфейсу RS-485. Напруга живлення всієї системи 5В. На платі STM32F103C8T6 є лінійний стабілізатор напруги на 3.3В, оскільки для живлення контролера необхідно максимум 3.6В. Живлення індикатору здійснюється від 5В.

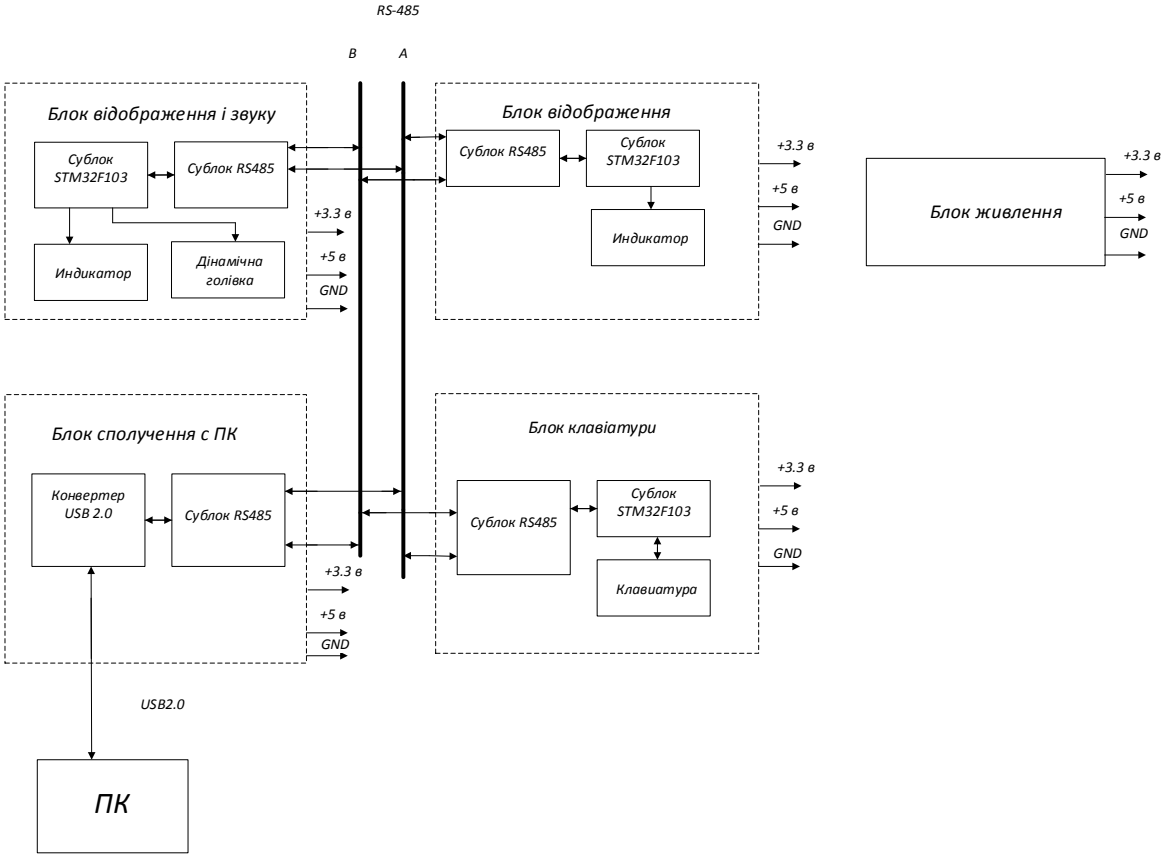

Рисунок 4.9 – Функціональна схема.

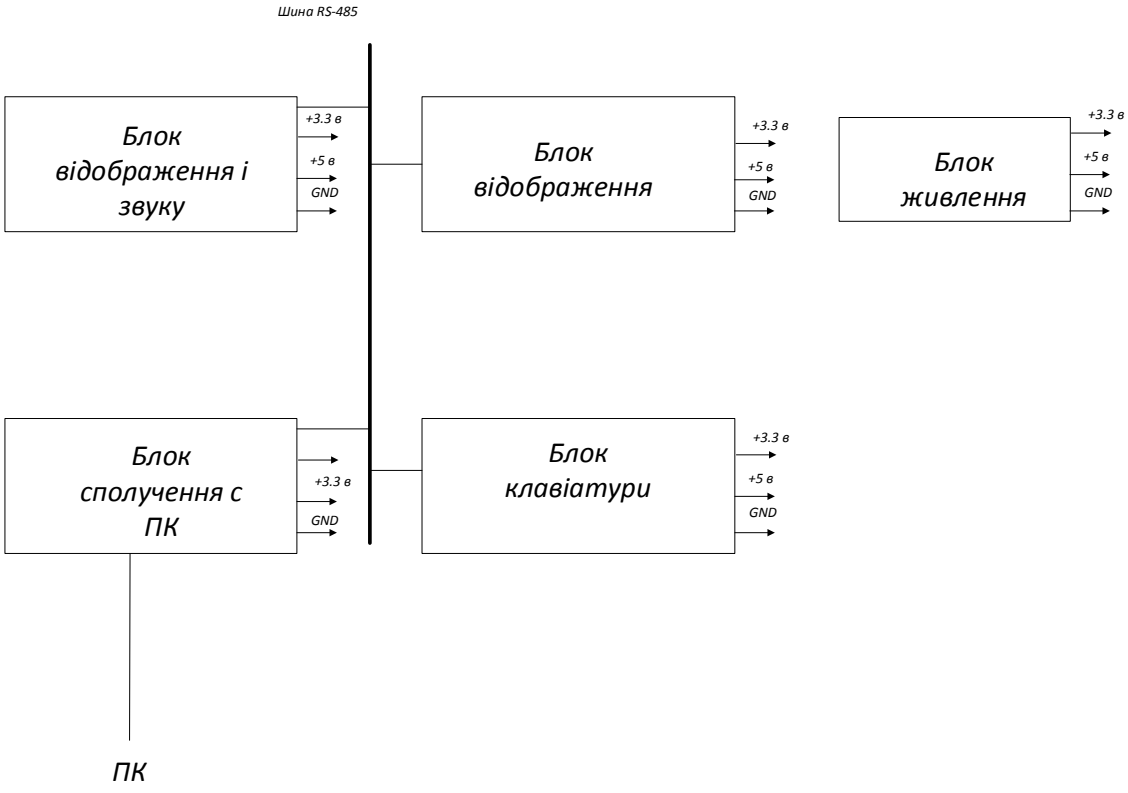

Рисунок 4.10 – Структурна схема.

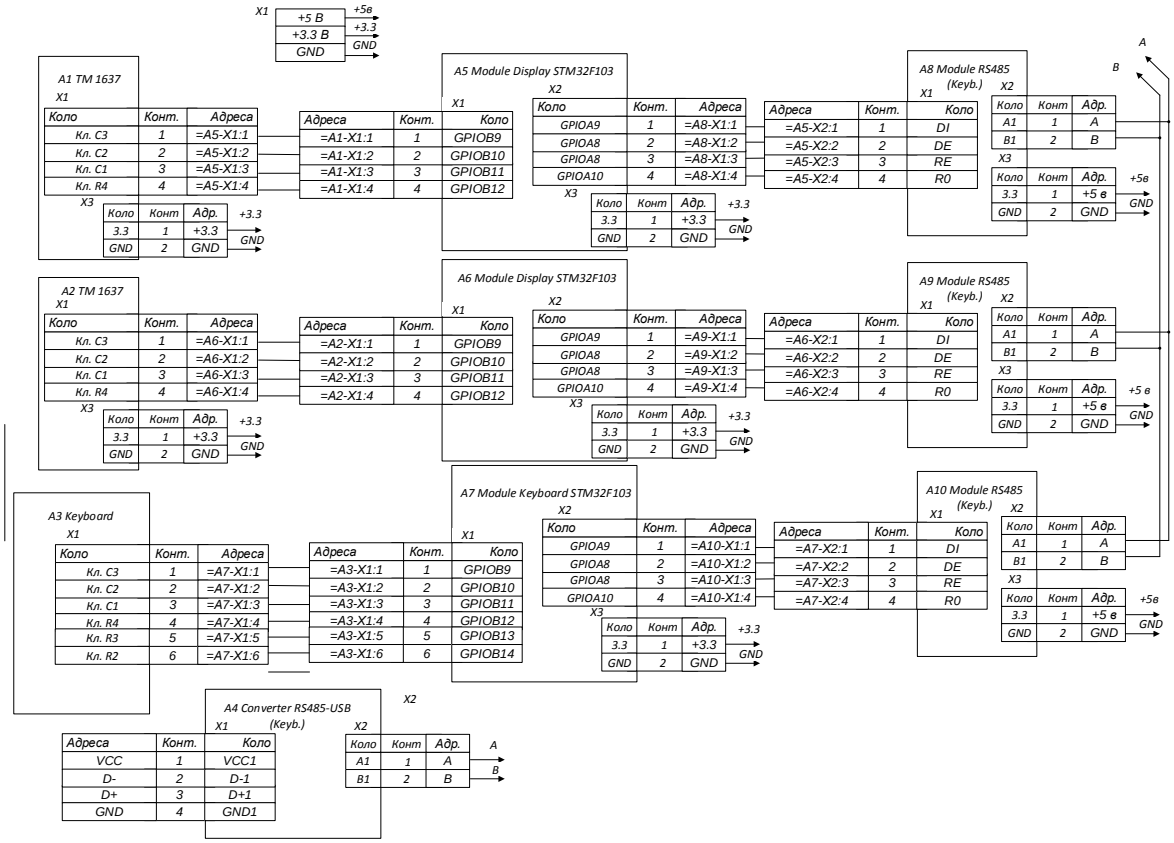

Рисунок 4.11 – Схема підключення.

4.9.1. Комплектація модулів

В таблиці нижче (таблиця 4.3) наведено комплектацію кожного з модулів.

Таблиця 4.3 Комплектація модулів.

Назва модуля	Модуль A	Модуль B	Модуль C
STM32f103C8T6	Так	Так	Так
Клавіатура	Так	Ні	Ні
Індикатор TM1637	Ні	Так	Так
П’єзодинамік	Ні	Ні	Так
UART-RS485	Так	Так	Так

5. СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1. Вибір технологій для розробки програмного забезпечення

Нижче наведено 4 основні середовища розробки для мікроконтролерів STM32. Для розробки даної системи використовувалося середовище Keil uVision 5.

5.1.1. IAR Embedded Workbench

IAR Embedded Workbench (рисунок 5.1) — інтегроване середовище розробки (IDE) випущене фірмою IAR Systems. Містить у собі зручний інтерфейс, оптимізовану CLIB/DLIB бібліотеку, підтримує різноманітні RTOS (Micrium uC/OS-II, OSEC ORTI), а також JTAG- адаптери різних фірм (OLIMEX, Phyton, ASHLING). IAR Embedded Workbench підтримує широкий спектр 8-, 16-, 32- розрядних мікроконтролерів — ARM, Actel, Infineon, NEC, Cypress, Atmel, Micronas, Analog Devices, ZiLOG, Microchip, Luminary Micro, Maxim, OKI, NXP, Samsung, STMicroelectronics, Texas Instruments, Renesas, Freescale Semiconductor, SiLabs і т.д. Кожній платформі відповідає своє середовище, наприклад платформі ARM відповідає IAR Embedded Workbench for ARM, платформі 8051 - IAR Embedded Workbench for 8051. В комплект IAR Embedded Workbench входять: C/C++ компілятор, транслятор мови асемблера, компонувальник, підпрограми для роботи з бібліотеками, редактор, менеджер проектів, C-SPY відладчик.

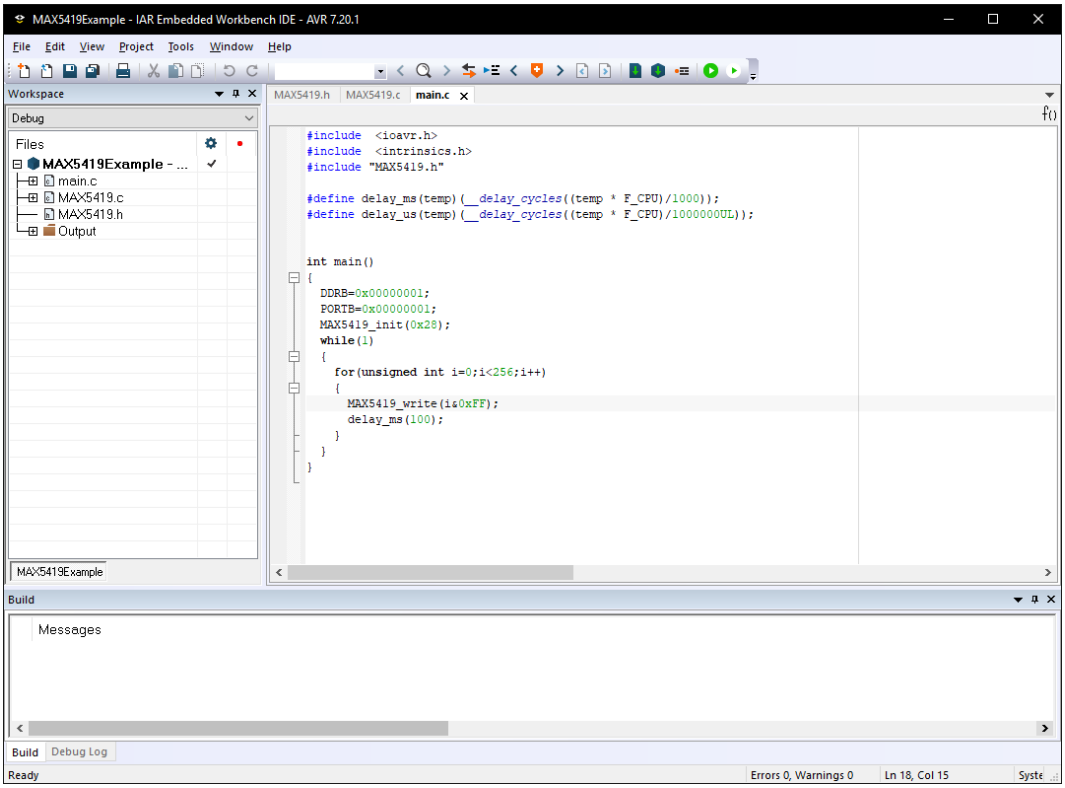

Рисунок 5.1 – Вигляд IAR Embedded Workbench.

5.1.2. Keil uVision

Keil uVision (рисунок 5.2) дозволяє працювати з проектами будь-якого ступеня складності, починаючи з введення і редагування вихідних текстів і закінчуючи внутрішньосхемною налагодженням коду і програмуванням ПЗУ мікроконтролера. Від розробника прихована велика частина другорядних функцій, що сильно розвантажує інтерфейс і робить управління інтуїтивно зрозумілим. Однак при зростанні складності реалізованих завдань, завжди можна задіяти весь потенціал модулів, що функціонують під управлінням єдиної оболонки. Серед основних програмних засобів Keil uVision можна відзначити:

1. Базу даних мікроконтролерів, що містить докладну інформацію про всі підтримуваних пристроях. Тут зберігаються їх конфігураційні дані і посилання на джерела інформації з додатковими технічними описами. При додаванні нового пристрою в проект все його унікальні опції встановлюються автоматично.

2. Менеджер проектів, службовець для об'єднання окремих текстів програмних модулів і файлів в групи, оброблювані за єдиними правилами. Подібна угруповання дозволяє набагато краще орієнтуватися серед безлічі файлів.

3. Вбудований редактор, який полегшує роботу з вихідним текстом за рахунок використання многооконного інтерфейсу, виділення синтаксичних елементів шрифтом і кольором. Існує опція настройки відповідно до смаків розробника. Редагування залишається доступним і під час налагодження програми, що дозволяє відразу виправляти помилки або відзначати проблемні ділянки коду.

4. Засоби автоматичної компіляції, ассемблирования і компонування проекту, які призначені для створення виконуваного (завантажувального) модуля програми. При цьому між файлами автоматично генеруються нові асемблерні і компиляторний зв'язку, які в подальшому дозволяють обробляти тільки ті файли, в яких відбулися зміни або файли, що знаходяться в залежності від змінених. Функція глобальної оптимізації проекту дозволяє досягти найкращого використання регістрів мікроконтролера шляхом неодноразової компіляції вихідного коду. Компілятори uVision працюють з текстами, написаними на Сі або асемблері для контролерів сімейств ARM, MSC51, C166 і багатьох інших. Крім того можливе використання компіляторів інших виробників.

5. Відладник-симулятор, відладжує роботу компільованою програми на віртуальній моделі мікропроцесора. Досить вірогідно моделюється робота ядра контролера і його периферійного обладнання: портів введення-виведення, таймерів, контролерів переривань. Для полегшення комплексної налагодження розроблюваного програмного забезпечення можливе підключення програмних моделей нестандартного обладнання.

6. Додаткові утиліти, що полегшують виконання найбільш поширених завдань. Число і набір змінюється від версії до версії. Виділяють наступні з них:

• Source Browser, що містить базу даних програмних символів для швидкого пошуку;

• Find in Files, призначену для пошуку заданого коду у всіх файлах зазначеної папки або проекту;

• Tools Menu, що дозволяє використовувати утиліти сторонніх виробників;

• PC-Lint, що аналізує вихідний текст програми з виділенням потенційно небезпечних місць;

• Flash tool, програмуючу FLASH-пам'ять мікроконтролерів.

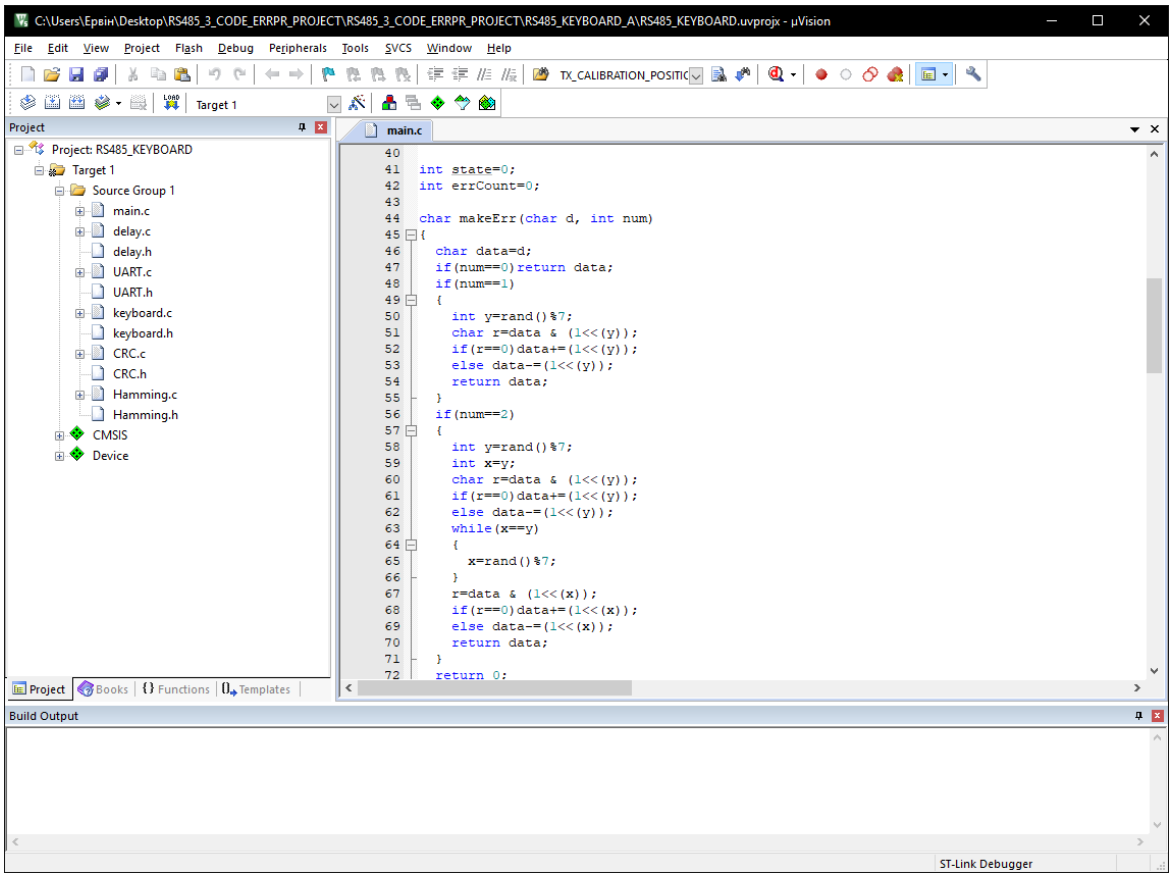

Рисунок 5.2 – Вигляд Keil uVision 5.

5.1.3. Eclipse з плагіном ARM і компілятором ARM-GCC

Eclipse (рисунок 5.3) це фреймворк для розробки модульних платформонезалежних застосунків із низкою особливостей:

1. Можливість розробки ПЗ на багатьох мовах програмування (рідною є Java);

платформонезалежна;

2. Модульна, призначена для подальшого розширення незалежним розробниками;
3. З відкритим сирцевим кодом;
4. Розробляється і підтримується фондом Eclipse, куди входять такі постачальники ПЗ, як IBM, Oracle, Borland.

Спочатку проект розроблявся в IBM як корпоративний стандарт IDE для розробки на багатьох мовах під платформи IBM. Потім проект було перейменовано на Eclipse і надано для подальшого розвитку спільноті.

Eclipse насамперед повноцінна Java IDE, націлена на групову розробку, має засоби роботи з системами контролю версій (підтримка CVS входить у поставку Eclipse, активно розвиваються кілька варіантів SVN модулів, існує підтримка VSS та інших). З огляду на безкоштовність, у багатьох організаціях Eclipse — корпоративний стандарт для розробки ПЗ на Java.

Друге призначення Eclipse — служити платформою для нових розширень. Такими стали C/C++ Development Tools (CDT), розроблювані інженерами QNX разом із IBM, засоби для підтримки інших мов різних розробників. Безліч розширень доповнює Eclipse менеджерами для роботи з базами даних, серверами застосунків та інших.

З версії 3.0 Eclipse став не монолітною IDE, яка підтримує розширення, а набором розширень. У основі лежать фреймворки OSGi, і SWT/JFace, на основі яких розроблений наступний шар — платформа і засоби розробки повноцінних клієнтських застосунків RCP (Rich Client Platform). Платформа RCP є базою для розробки різних RCP програм як торент-клієнт Azareus чи File Arranger. Наступний шар — платформа Eclipse, що є набором розширень RCP — редактори, панелі, перспективи, модуль CVS і модуль Java Development Tools (JDT).

Eclipse написана на Java, тому є платформонезалежним продуктом, крім бібліотеки графічного інтерфейсу SWT, яка розробляється окремо для більшості поширених платформ. Бібліотека SWT використовує графічні засоби платформи (ОС), що забезпечує швидкість і звичний зовнішній вигляд інтерфейсу користувача.

Відповідно до IDC, із Eclipse працюють 2,3 мільйона розробників.

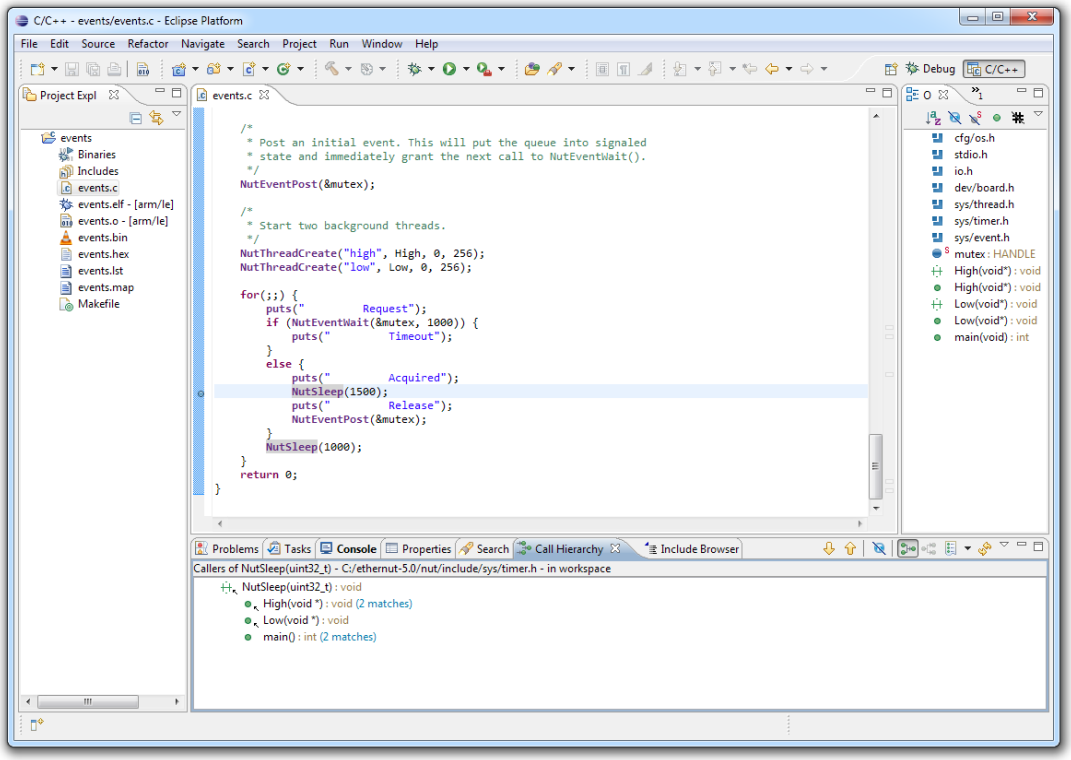

Рисунок 5.3 – Вигляд Eclipse.

5.1.4. CooCox IDE

Високоінтегроване програмне середовище, призначене для розробки коду мікроконтролерів архітектури ARM.

CooCox CoIDE (рисунок 5.4) є одним з найпростіших і швидких в плані установки, освоєння і налаштування рішень, що дозволяє навіть починаючим користувачам домагатися в ньому істотних результатів. Успішний старт перших проектів забезпечує майстер, що допомагає пройти через всі основні етапи розробки шляхом відповідей на прості запитання. Якісно зробити середовище

CooCox CoIDE дозволяє завантажувати вихідний код програми, редагувати його, проводити компіляцію (сторонніми засобами), прошивати контролер і проводити налагодження.

Програма заснована на базі Eclipse і має всі її переваги. Редактор коду включає в себе підсвічування синтаксису і спливаючі підказки. Присутні функції глобальної заміни змінної та пропозиції варіантів закінчення коду. Середовище підтримує мікроконтролери серії ST, а також ряд інших сімейств: Atmel, Holtek, Freescale, Nuvoton, NXP, Energy Micro, Texas Instruments і деякі інші. Список чіпів постійно збільшується з кожною версією програми. Вбудований відладчик ST-Link підтримує всі основні режими налагодження.

При створенні нового проекту пропонується вибір використовуваної мікросхеми і бібліотек. Можливий перегляд коротких характеристик кожного чіпа. CoIDE автоматично створює всю структуру проекту, а також підключає всі інші необхідні для роботи бібліотеки. Кожна з них містить кілька готових прикладів, які можна використовувати в проекті. Присутня функція поповнення бібліотек власними прикладами. При підключенні нових бібліотек до проекту враховуються всі залежності між ними.

Першим з недоліків CooCox CoIDE варто відзначити відсутність компілятора GCC, який потрібно завантажити і встановити окремо. А після цього в настройках CoIDE необхідно вказати правильний шлях до нього. Для серії ARM існує кілька варіантів компіляторів з різними наборами допоміжних засобів. За замовчуванням CooCox CoIDE розроблялася для взаємодії з ARM GCC.

Шляхи до файлів проекту жорстко прописуються в програмі. Просте переміщення папки з проектом призведе до того, що проект не збереться. Тому шляху необхідно редагувати вручну в файлах. Швидше за все, професійним розробникам програма здасться занадто простою, в ній також відсутня можливість тонких налаштувань.

Дане середовище розробки абсолютно безкоштовна і має відкритий код. Для отримання доступу до завантаження необхідно пройти просту процедуру

реєстрації. Також програму можна закачати через попередньо встановлений менеджер CoCenter, який в подальшому буде сповіщати про всі оновлення і дає можливість установки додаткових утиліт розробника. Серед них варто відзначити власну вбудовану операційну систему для роботи з мікроконтролерами CooCox CoOS, софт для програмування Flash-пам'яті CooCox CoFlash, а також інструмент, який спрощує конфігурування портів контролерів CooCox CoSmart. Встановлювати CoIDE рекомендується в каталог без російських букв, без пробілів, а найкраще в пропонований за замовчуванням варіант.

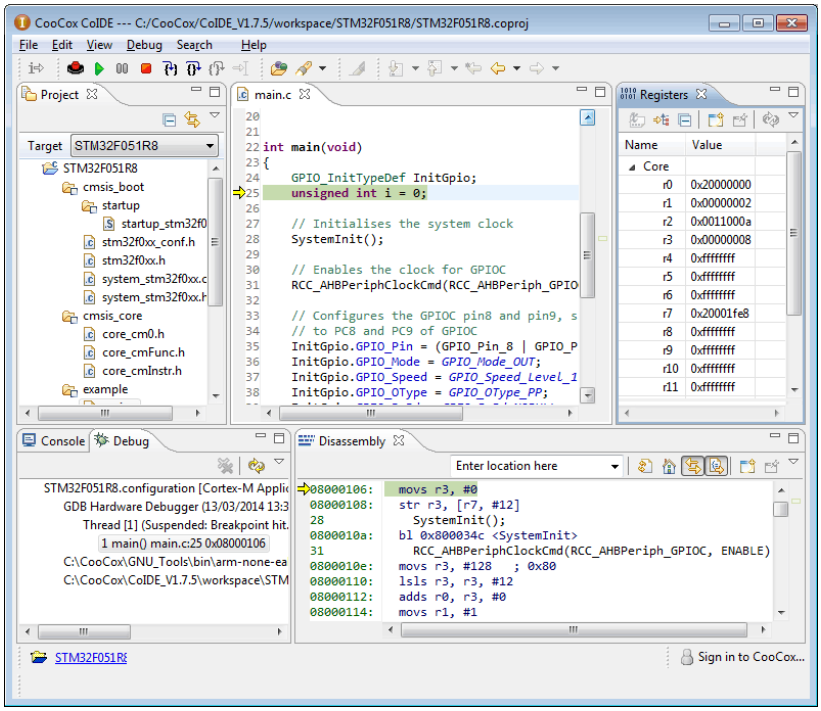

Рисунок 5.4 – Вигляд CooCox IDE.

5.1.5. STM32CubeMX

STM32CubeMX-STM32 (рисунок 5.5) – графічний генератор коду для мікроконтролерів STM32.

Генератори графічного коду дозволяють вам піти від рутинного програмування. Замість того, щоб створювати текстові файли самостійно, розробник використовує графічні інструменти, які автоматично формують

програмний код. Прикладом такого підходу є крос-платформний графічний редактор STM32CubeMX , що працює на всіх популярних ОС: Windows®, Linux і OS X®.

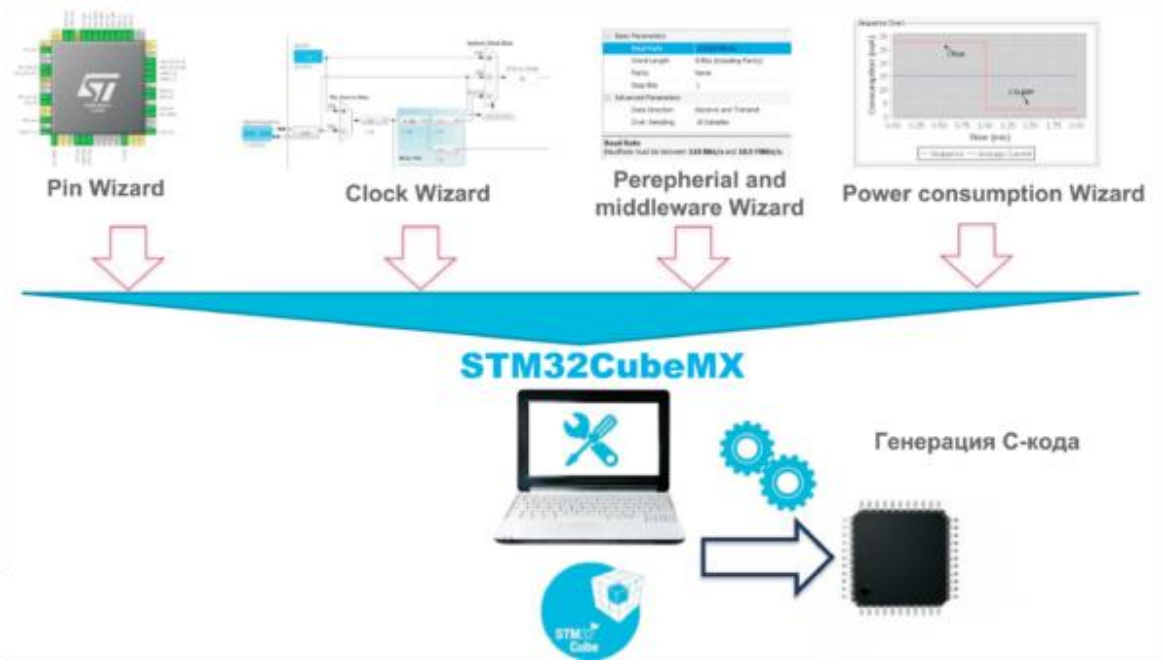

Рисунок 5.5 – Вигляд STM32CubeMX.

STM32CubeMX дозволяє:

- створювати і редагувати проекти для STM32 мікроконтролерів, а потім генерувати С-код для конкретних інтегрованих середовищ: PCK™ EWARM, Keil™ MDK-ARM, Atollic® TrueSTUDIO і AC6 System Workbench (SW4STM32);
- Пошук оптимального мікроконтролера або налагоджувальної плати для задоволення конкретної програми. Фільтрація проводиться в різних областях, як і в разі з Санкт MCU Finder;
- Візуалізуйте мікроконтроллер висновки з майстром PIN-коду з можливістю формувати CSV файл таблиці для плат tracers;
- Візуалізуйте налаштування cloclock з годинником майстра, включаючи глобальні сигнали годинника і периферійних сигналів годинника;

- візуалізації периферійних пристроїв і середнього рівня програмного забезпечення, файлової системи, протокол стеки, операційна система, і так далі з Perepherial і середній Wizardware. Різні бібліотеки від ST: HAL або LL можуть бути використані для створення с-коду;
- оцінити рівень споживання батареї та терміну служби акумулятора з набором параметрів мікроконтролера за допомогою утиліти «майстер енергоспоживання»;
- автоматично оновлювати як STM32CubeMX сам, так і STM32Cube програмних бібліотек він використовує для різних STM32 сімей;
- для переміщення проекту з одного мікроконтролера до іншої моделі.

STM32CubeMX є частиною платформи STMCube™. На додаток до STM32CubeMX, STMCube™ включає в себе окремі власні набори бібліотек (рис. 8). Наприклад, набір бібліотек STM32CubeF2 призначений для роботи з STM32F2 сімейними мікроконтролерами, і STM32CubeF4 Kit об'єднує бібліотеки для STM32F4.

У свою чергу, індивідуальні пакети STMCube включають в себе:

- для забезпечення мікроконтролера встановлено незалежні бібліотеки HAL;
- бібліотек середнього рівня. Наприклад, найсучасніші STM32CubeF7 пакет програмного забезпечення включає в себе наступні бібліотеки та стеки: CMSIS-RTOS на основі FreeRTOS, LwIP/IP стек/IP, FAT fatFs файлової системи з NAND Flash підтримка, StemWin-SEGGER Emwin графічний стек, повний Стек USB (хост і пристрій). Сенсорна бібліотека для сенсорних застосунків доступна для низки родин.
- прикладів і шаблонів проектів для різних середовищ і наборів налагодження (Discovery, Нуклео, оціночні плати).

На рисунку 5.6 показано як взаємодіють компоненти STM32CubeMX. У цьому прикладі користувач налаштував мікроконтролер STM32F429'IT за допомогою STM32CubeMX. Після візуального налаштування (висновки, такту і т. д.) закінчено, STM32CubeMX генерує с-код, який використовує бібліотеки з програмним пакетом STM32CubeF4. В результаті, користувач отримує завершений С-проект, створений для конкретного комплексного середовища розробки: PCK™ EWARM, Keil™ MDK-ARM, Atollic® TrueSTUDIO і AC6 System Workbench (SW4STM32). У цьому проекті вже включені всі необхідні бібліотеки та файли.

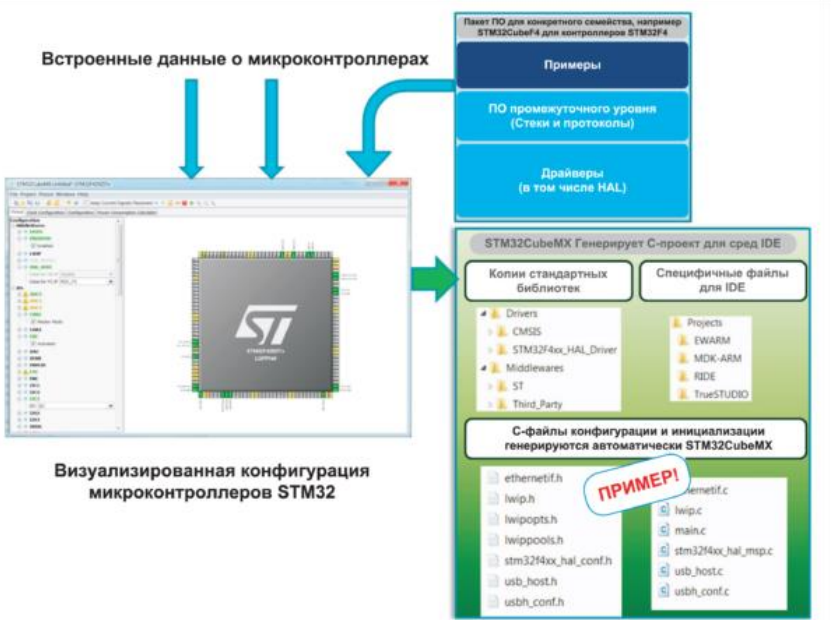

Рисунок 5.6 – Взаємодія компонентів STM32CubeMX.

5.2. Проектування та створення бібліотек

5.2.1. Налаштування середовища

Нижче наведено налаштування проекту в Manage Run-Time Environment (рисунок 5.7). Там необхідно активувати такі пункти: DSP, CORE, USART, GPIO, Framework.

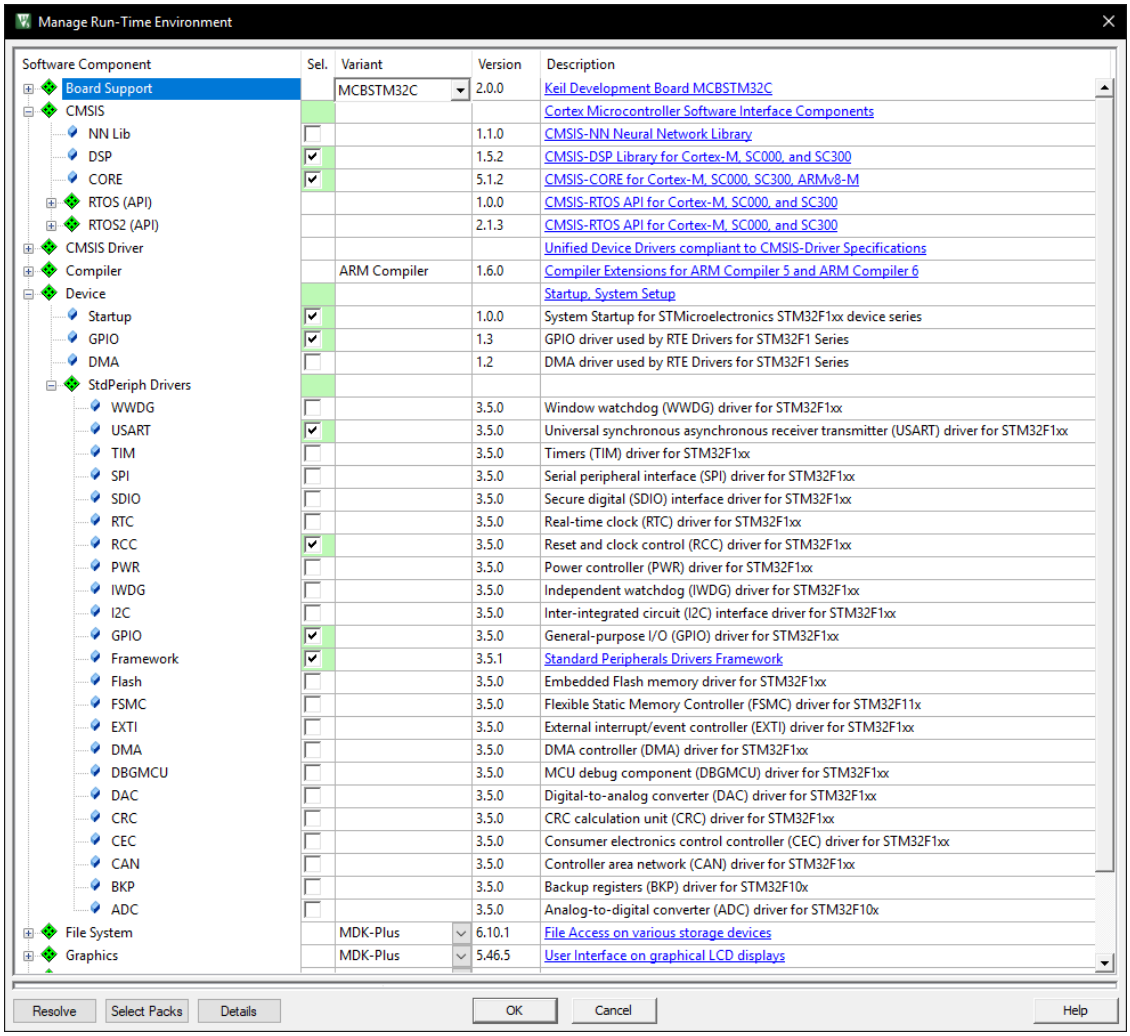

Рисунок 5.7 – Налаштування проекту в Manage Run-Time Environment.

5.2.2. Бібліотека delay

Дана бібліотека реалізує паузу в мілісекундах та мікросекундах. Вона базується на тактовій частоті контролера. У бібліотеки для цього є дві функції:

— void delay_us(unsigned long Val) (рисунок 5.8);

```
void delay_us( unsigned long Val)
{
  Val=Val*10;
  for( ; Val != 0; Val--)
      {
    __nop();
      }
}
```

Рисунок 5.8 – Функція delay_us.

— void delay_ms(unsigned long Val) (рисунок 5.9).

```
void delay_ms( unsigned long Val)
{
  Val=Val*10000;
  for( ; Val != 0; Val--)
      {
    __nop();
      }
}
```

Рисунок 5.9 – Функція delay_ms.

5.2.3. Бібліотека UART

Ця бібліотека реалізує передачу даних через модуль RS-485. Швидкість передачі 9600 бод, це значення можна змінити у файлі delay.h. У бібліотеці для цього є 3 функції:

— void UART_init(void);

Дана функція налаштовує модуль USART та ініціалізує вихід для керування напрямком передачі на модулі RS-485. Для цього здійснюються такі кроки: виходи RX та TX налаштовуються в режим альтернативних функцій, налаштовується пін на вихід для курування напрямком передачі, налаштовується модуль USART (вказується швидкість, стартовий біт, стоповий біт, та метод перевірки на помилку).

— void UART_send(char data[],unsigned char count) (рисунок 5.10);

Ця функція реалізує передачу даних через модуль RS-485.

```
void UART_send(char data[],unsigned char count)
{
  delay_ms(2);
  GPIO_SetBits(GPIOA, GPIO_Pin_8);
  delay_ms(2);
  for(unsigned char i=0;i<count;i++)
  {
    USART_SendData(USART1,data[i]);
    delay_ms(2);
  }
  delay_ms(2);
  GPIO_ResetBits(GPIOA, GPIO_Pin_8);
}
```

Рисунок 5.10 – Функція UART_send.

— void UART_dec(unsigned int val).

Дана функція реалізує передачу числа типу unsigned int (беззнакове ціле). Ця функція використовувалась для відладки.

5.2.4. Бібліотека keyboard

Ця бібліотека реалізує зчитування даних з клавіатури. Принцип роботи бібліотеки полягає в тому, що вона подає низький рівень сигналу на кожен рядок окремо, а на кожному стовпці вхід підтягнутий до плюсу, і коли кнопка натиснута, на вході буде низький рівень сигналу. В бібліотеці реалізовано дві функції:

— void keypad_init(unsigned int cols[numCol],unsigned int rows[numRow]);

В цій функції вмикається тактування порту до якого під’єднана клавіатура, налаштовуються входи та виходи, до яких під’єднана клавіатура. Як параметри в цю функцію передаються піни до яких під’єднані стовпці та рядки. Кількість рядків та стовпців можна змінити в файлі keyboard.h. Також можна змінити матрицю символів, виглядає вона наступним чином (рисунок 5.11):

```
char keypad[numRow][numCol]=
{
  '1','2','3',
  '4','5','6',
  '7','8','9',
  '*','0','#'
};
```

Рисунок 5.11 – Матриця символів клавіатури.

— char keypadGetKey(void) (рисунок 5.12).

Дана функція сканує клавіатуру, та повертає символ натиснутої кнопки при умові що кнопка натиснута та при попередньому скануванні вона не була натиснута.

```
char keypadGetKey(void)
{
  char res=KEYPAD_NO_PRESSED;
  for(int i=0;i<numCol;i++)
  {
    for(int k=0;k<numCol;k++)GPIO_SetBits(GPIOB, col[k]);
    GPIO_ResetBits(GPIOB, col[i]);
    for(int k=0;k<numRow;k++)if(!GPIO_ReadInputDataBit(GPIOB, row[k]))res=keypad[k][i];
  }
  if(res != oldKey) //if key is difference with old key
  {
    oldKey=res;
    return res;
  }
  return KEYPAD_NO_PRESSED;
}
```

Рисунок 5.12 – Функція сканування клавіатури.

5.2.5. Бібліотека CRC

Дана бібліотека вираховує та перевіряє контрольну суму по принципу виключаючого або. В цій бібліотеці реалізовано дві функції:

— char calc_CRC(char buffer[],int count) (рисунок 5.13);

Ця функція приймає масив байт та кількість елементів у масиві, і повертає байт контрольної суми.

```
char calc_CRC(char buffer[],int count)
{
  char res=buffer[0];
  for(int i=1;i<count;i++)res=res^buffer[i];
  return res;
}
```

Рисунок 5.13 – Функція обчислення контрольної суми.

— char check_CRC(char buffer[],int count).

Ця функція приймає масив байт та кількість елементів у масиві, і повертає 0x00 якщо контрольна сума не правильна, або 0xFF якщо правильна.

5.2.6. Бібліотека protocol

Ця бібліотека реалізує простий протокол повідомлень між модулями. В ній є дві функції:

— void protocol_add(char data);

Дана функція додає отриманий символ по USART в буфер прийому протоколу.

— char protocol_check(char add)(рисунок 5.14).

Дана функція перевіряє буфер і повертає отриману команду, якщо така є в буфері.

```
char protocol_check(char add)
{
  if(time==0 && count==3 && buff[0]==add && check_CRC(buff,3))
  {
    count=0;
    return buff[1];
  }
  else if(time==0 && count!=0)
  {
    count=0;
    return 0;
  }
  if(time>0)time--;
  return 0;
}
```

Рисунок 5.14 – Функція перевірки буфера.

5.2.7. Бібліотека Hamming

Дана бібліотека реалізує кодування Хемінга з автовиправленням помилки в одному біті даних. В цій реалізації поле даних складає 4 біти. Тут реалізовано 3 функції:

— char calc_Ham(char data)(рисунок 5.15);

Ця функція вираховує закодовану посилку. На вході вона бере перші 4 біти із байту, і на виході повертає байт із закодованими даними.

```
char calc_Ham(char data)
{
  char dataBuf[4][7];
  char res=0;
  dataBuf[0][0]=0;
  dataBuf[0][1]=0;
  dataBuf[0][2]=data&(0x01);
  dataBuf[0][3]=0;
  dataBuf[0][4]=(data>>1)&(0x01);
  dataBuf[0][5]=(data>>2)&(0x01);
  dataBuf[0][6]=(data>>3)&(0x01);
  for(int i=0;i<7;i++)
  {
    for(int k=0;k<3;k++)
    {
      dataBuf[k+1][i]=((i+1)>>k)&(0x01);
    }
  }
  char val[3]={0,0,0};
  for(int k=0;k<3;k++)
  {
    for(int i=0;i<7;i++)
    {
      val[k]+=dataBuf[0][i]*dataBuf[k+1][i];
    }
  }
  res+=(val[0]%2);
  res+=((val[1]%2)<<1);
  res+=(dataBuf[0][2]<<2);
  res+=((val[2]%2)<<3);
  res+=(dataBuf[0][4]<<4);
  res+=(dataBuf[0][5]<<5);
  res+=(dataBuf[0][6]<<6);
  return res;
}
```

Рисунок 5.15 – Функція кодування.

— char check_Ham(char data)(рисунок 5.16);

Дана функція перевіряє закодований байт, якщо є помилка – повертає нуль, якщо ні – повертає розкодоване значення.

```
char check_Ham(char data)
{
  char dataBuf[4][7];
  dataBuf[0][0]=data&(0x01);
  dataBuf[0][1]=(data>>1)&(0x01);
  dataBuf[0][2]=(data>>2)&(0x01);
  dataBuf[0][3]=(data>>3)&(0x01);
  dataBuf[0][4]=(data>>4)&(0x01);
  dataBuf[0][5]=(data>>5)&(0x01);
  dataBuf[0][6]=(data>>6)&(0x01);
  for(int i=0;i<7;i++)
  {
    for(int k=0;k<3;k++)
    {
      dataBuf[k+1][i]=((i+1)>>k)&(0x01);
    }
  }
  char val[3]={0,0,0};
  for(int k=0;k<3;k++)
  {
    for(int i=0;i<7;i++)
    {
      val[k]+=(dataBuf[0][i]*dataBuf[k+1][i]);
    }
  }
  return (val[0]%2)+((val[1]%2)<<1)+((val[2]%2)<<2);
}
```

Рисунок 5.16 – Функція декодування.

— char fix_Ham(char d)(рисунок 5.17).

Ця функція виправляє помилку і повертає розкодоване значення. Якщо помилку виправити не вдалося – повертає 0.

```
char fix_Ham(char d)
{
  char data=d;
  char res=0;
  char val=check_Ham(data);
  if(val==0)
  {
    res+=((data>>2)&(0x01));
    res+=((data>>4)&(0x01))<<1;
    res+=((data>>5)&(0x01))<<2;
    res+=((data>>6)&(0x01))<<3;
    return res;
  }
  char r=data & (1<<(val-1));
  if(r==0)data+=(1<<(val-1));
  else data-=(1<<(val-1));
  val=check_Ham(data);
  if(val==0)
  {
    res+=((data>>2)&(0x01));
    res+=((data>>4)&(0x01))<<1;
    res+=((data>>5)&(0x01))<<2;
    res+=((data>>6)&(0x01))<<3;
    return res;
  }
  return 0;
}
```

Рисунок 5.17 – Функція виправлення помилки.

5.2.8. Бібліотека buzzer

Ця бібліотека реалізує керування звуковим сигналом. В ній є дві функції:

— void buzzer_init(int p);

Ця функція ініціалізує вихід для п’єзодинаміка.

— void buzzer_tone(int freq, int time)(рисунок 5.18).

Дана функція генерує тоновий сигнал заданої частоти на заданий період часу.

```
void buzzer_tone(int freq, int time)
{
  int timeNow=0;
  long long v=1000000/freq;
  int val=v;
  int t=time*1000;
  while(timeNow<t)
  {
    GPIO_SetBits(GPIOC, pin);
    delay_us(val);
    timeNow+=val;
    GPIO_ResetBits(GPIOC, pin);
    delay_us(val);
    timeNow+=val;
  }
}
```

Рисунок 5.18 – Функція генерування тонового сигналу.

5.2.9. Бібліотека tm1637

Ця бібліотека призначена для керування модулем семисегментного індикатора на базі контролера TM1637. В ній ралізовано такі функції:

— void TM1637_init(void);

Ця функція вмикає тактування порту та налаштовує піни для підключення дисплею.

— void TM1637_Generate_START(void);

Ця функція генерує команду «старт».

— void TM1637_Generate_STOP(void);

Ця функція генерує команду «стоп».

— void TM1637_WriteData(uint8_t data);

Ця функція передає байт даних.

— int8_t TM1637_coding(uint8_t DispData);

Дана функція кодує символ для відображення на індикаторі.

— void TM1637_coding_all(uint8_t DispData[]);

Дана функція кодує символи для відображення на індикаторі.

— void separate_Digit_to_digits(int16_t Digit,uint8_t SegArray[]) (рисунок 5.19);

Ця функція перетворює число на масив символів.

```
void separate_Digit_to_digits(int16_t Digit,uint8_t SegArray[])
{

  if((Digit > 9999)||(Digit < -999))return;
  if(Digit < 0)
  {
    SegArray[0] = INDEX_NEGATIVE_SIGN;
    Digit = (Digit & 0x7fff);
    SegArray[1] = Digit/100;
    Digit %= 100;
    SegArray[2] = Digit / 10;
    SegArray[3] = Digit % 10;
    if(BlankingFlag)
    {
      if(SegArray[1] == 0)
      {
        SegArray[1] = INDEX_BLANK;
        if(SegArray[2] == 0) SegArray[2] = INDEX_BLANK;
      }
    }
  }
  else
  {
    SegArray[0] = Digit/1000;
    Digit %= 1000;
    SegArray[1] = Digit/100;
    Digit %= 100;
    SegArray[2] = Digit / 10;
    SegArray[3] = Digit % 10;
    if(BlankingFlag)
    {
      if(SegArray[0] == 0)
      {
        SegArray[0] = INDEX_BLANK;
        if(SegArray[1] == 0)
        {
          SegArray[1] = INDEX_BLANK;
          if(SegArray[2] == 0) SegArray[2] = INDEX_BLANK;
        }
      }
    }
  }
  BlankingFlag = 1;
}
```

Рисунок 5.19 – Функція перетворення числа на масив символів.

— void TM1637_display_custom(uint8_t Seg_N,uint8_t DispData);

Ця функція виводить на дисплей кастомний символ.

— void TM1637_display(uint8_t Seg_N,int8_t DispData);

Дана функція відображає символ на дисплеї.

— void TM1637_display_all(uint16_t Digit);

Ця функція відображає число на дисплеї.

— void TM1637_display_all_custom(uint32_t Digit);

Ця функція відображає кастомні символи на дисплеї.

— void TM1637_brightness(uint8_t brightness);

Ця функція змінює яскравість дисплею.

— void TM1637_clearDisplay(void).

Ця функція очищає дисплей.

5.3.Проектування та створення основних програм

Нижче наведена схема компонентів (рисунок 5.20) та діаграма розгортання (рисунок 5.21).

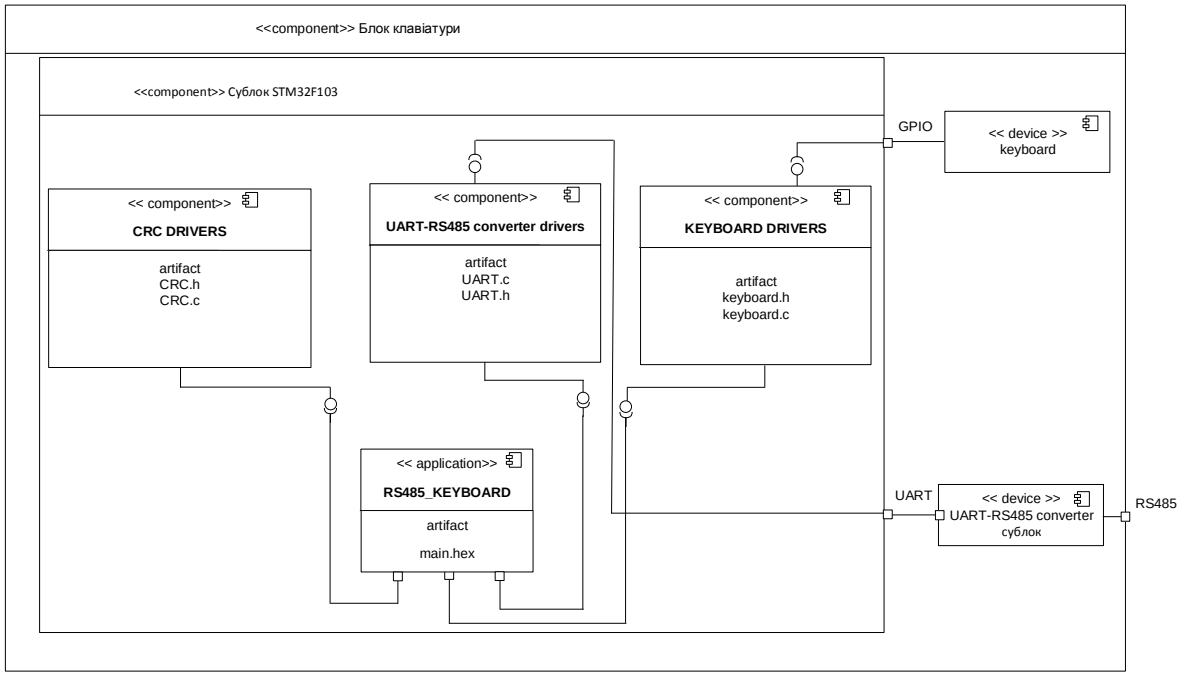

Рисунок 5.20 – Схема компонентів.

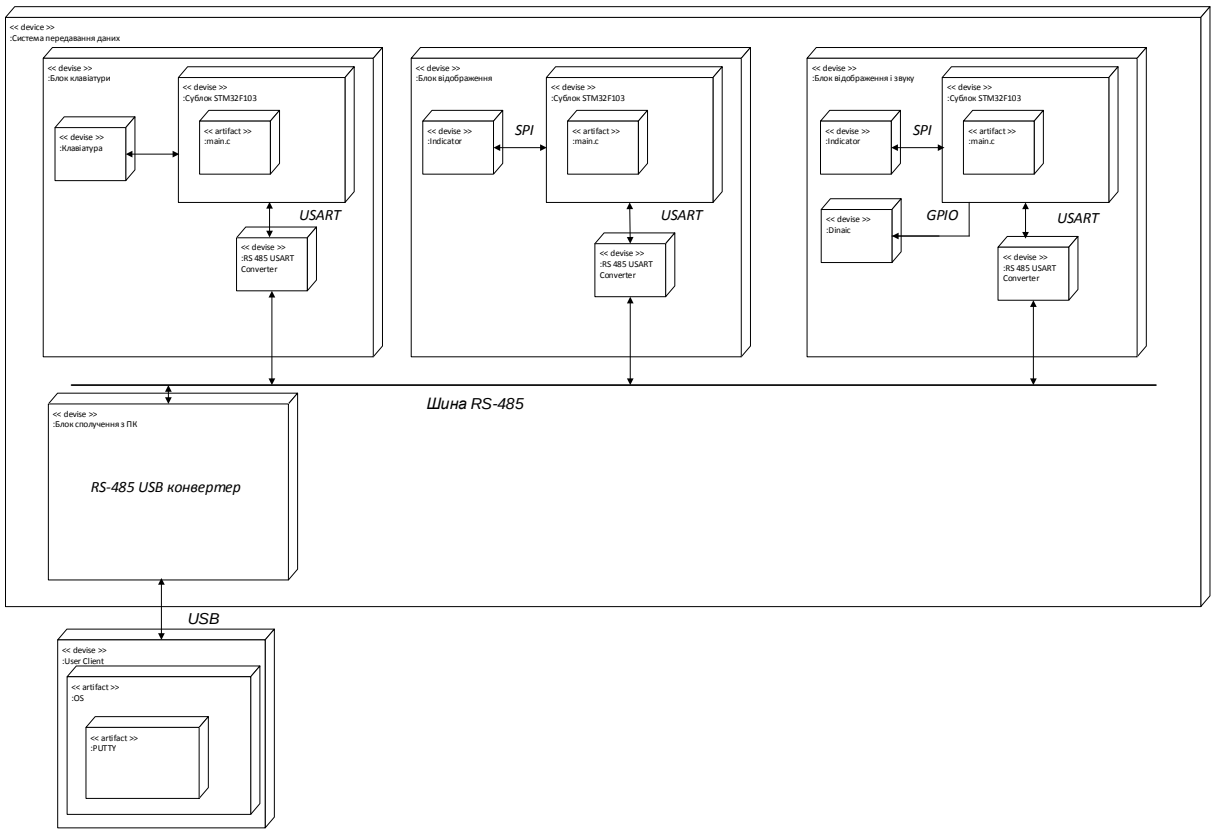

Рисунок 5.21 – Діаграма розгортання.

5.3.1. Модуль A

В модулі A використані такі бібліотеки: delay, UART, keyboard, CRC, Hamming.

Модуль має наступний функціонал:

1. Вибір адреси отримувача;

   Вибір адреси здійснюється таким чином: ‘*’+’1’ – вибір модуля B, ‘*’+’2’ – вибір модуля C.

2. Вибір кількості помилок;

   Вибір кількості помилок здійснюється таким чином: ‘#’+’0’ – нуль помилок, ‘#’+’1’ – одна помилка, ‘#’+’2’ – дві помилки.

3. Передача повідомлення по вибраному адресу з вибраною кількістю помилок.

Передача здійснюється таким чином: при натисканні кнопки від ‘1’ до ‘9’ передається повідомлення з командою номер якої відповідає номеру кнопки.

Для додавання помилки в код Хемінга реалізовано спеціальну функцію char makeErr(char d, int num) (рисунок 5.22).

```
char makeErr(char d, int num)
{
  char data=d;
  if(num==0)return data;
  if(num==1)
  {
    int y=rand()%7;
    char r=data & (1<<(y));
    if(r==0)data+=(1<<(y));
    else data-=(1<<(y));
    return data;
  }
  if(num==2)
  {
    int y=rand()%7;
    int x=y;
    char r=data & (1<<(y));
    if(r==0)data+=(1<<(y));
    else data-=(1<<(y));
    while(x==y)
    {
      x=rand()%7;
    }
    r=data & (1<<(x));
    if(r==0)data+=(1<<(x));
    else data-=(1<<(x));
    return data;
  }
  return 0;
}
```

Рисунок 5.22 – Функція додавання помилки в код Хемінга.

5.3.2. Модуль B

В модулі B використані такі бібліотеки: delay, UART, CRC, Hamming, tm1637, protocol.

Модуль має наступний функціонал:

1. Прийом повідомлень адресованих цьому модулю.
2. Автовиправлення помилки в одному біті даних.
3. Відображення помилки даних на дисплеї.
4. Відображення команди на дисплеї.

5.3.3. Модуль C

В модулі C використані такі бібліотеки: delay, UART, CRC, Hamming, tm1637, protocol, buzzer.

Модуль має наступний функціонал:

1. Прийом повідомлень адресованих цьому модулю.
2. Автовиправлення помилки в одному біті даних.
3. Відображення помилки даних на дисплеї.
4. Відображення команди на дисплеї.
5. Генерація тонового сигналу в залежності від команди.

5.4. Приклад роботи програми

Для перевірки програм зібрано макет з трьох модулів. При тестуванні модулів було перевірено всі функції, результати тестів наведені в таблиці нижче (таблиця 5.1):

Таблиця 5.1 Результати тестування.

Модуль A	Модуль B	Модуль C
Передача 0 помилок на B	Отримано успішно	Нічого не отримано
Передача 0 помилок на C	Нічого не отримано	Отримано успішне
Передача 1 помилка на B	Виправлено успішно	Нічого не отримано
Передача 1 помилка на C	Нічого не отримано	Виправлено успішно
Передача 2 помилки на B	Не виправлено	Нічого не отримано
Передача 2 помилки на C	Нічого не отримано	Не виправлено

Результати тестування показують що всі програми працюють коректно, автовиправлення успішно випрявляє помилкау в 1 біті даних (рисунки 5.23, 5.24, 5.25).

Рисунок 5.23 – Результат натискання кнопки ‘6’.

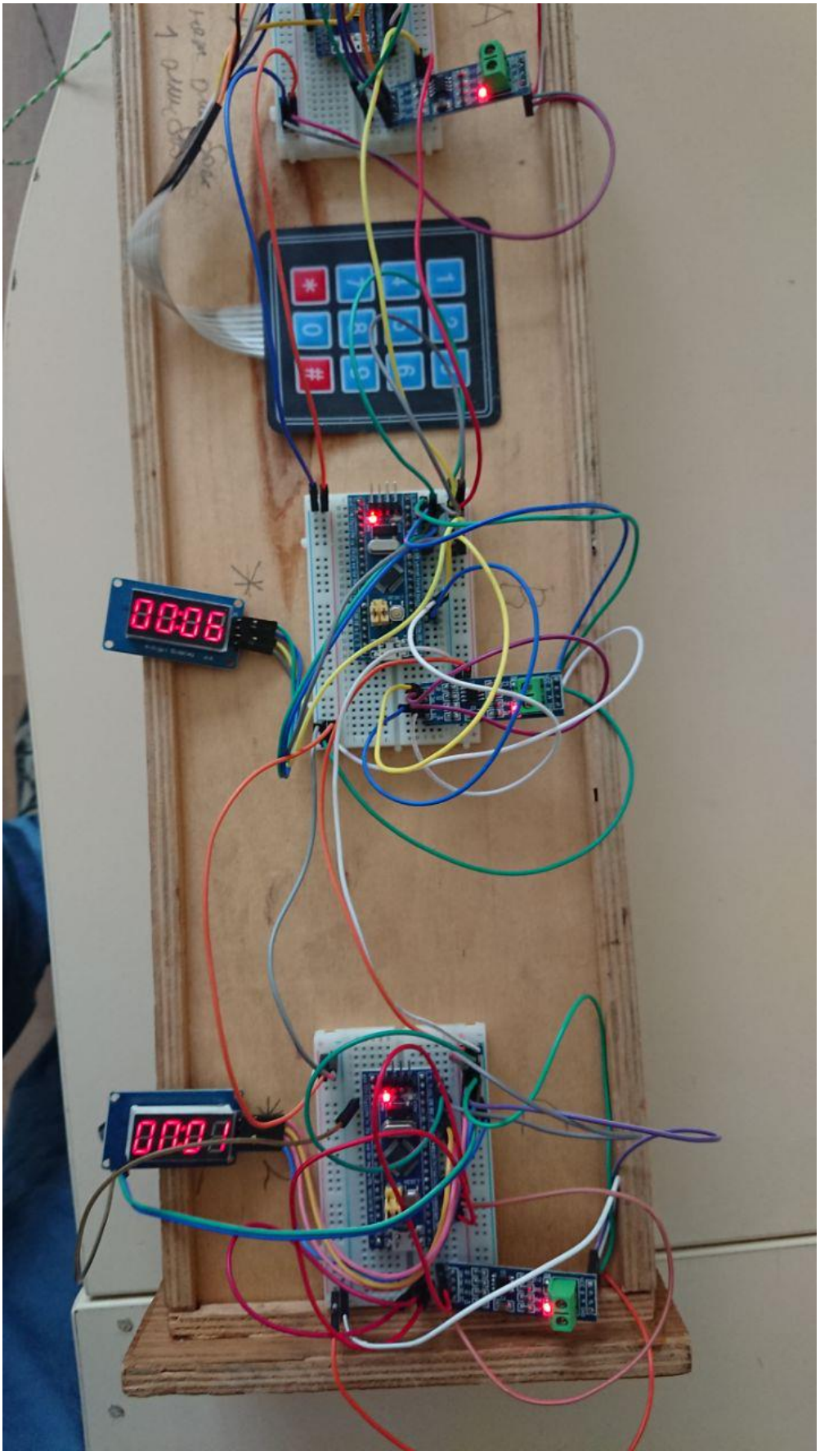

Рисунок 5.24 – Результат передачі команд на модулі B та C

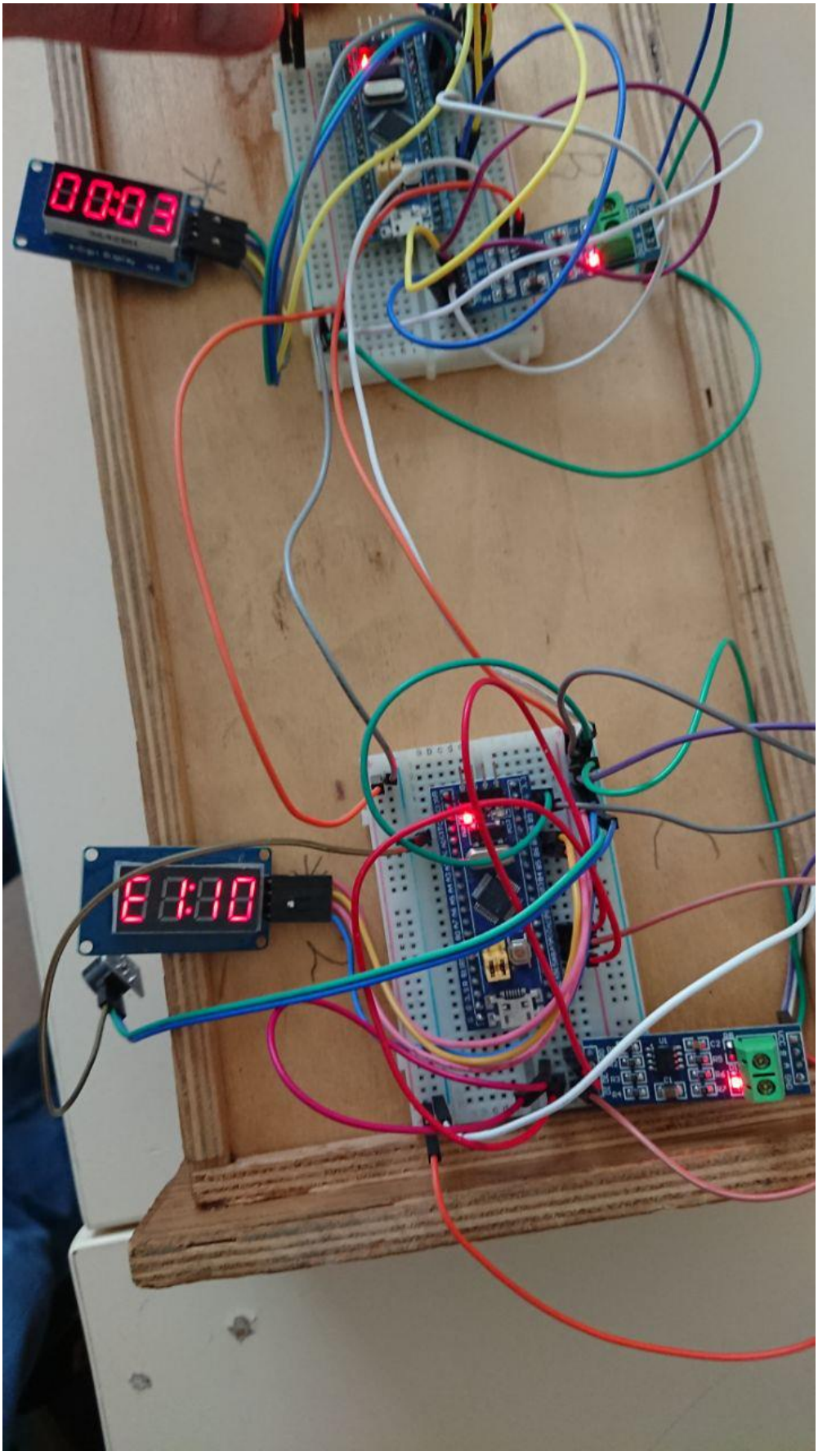

Рисунок 5.25 – Модуль С отримав повідомлення з помилкою.

ВИСНОВОК

В процесі роботи над дипломним проектом, що присвячений розробці прототипу цифрової системи передачі даних з завадостійким кодуванням було створено структурну та функціональну схему макету системи. Також розроблена схема з'єднань. На основі схеми з'єднань створений макет цифрової системи передачі даних з завадостійким кодуванням.

Розроблений прототип програм для програмування блоків макету, субблоки яких побудовані на основі мікроконтролера STM32F103C8.

У ході проектування програмного забезпечення було розроблено декілька бібліотек-драйверів, що призначені для окремих субблоків системи передачі даних, а саме:

- драйвер для передачі даних на UART-RS485;
- драйвер клавіатури;
- драйвер п'єзодинаміка;
- драйвер кодування;
- драйвер контрольної суми;
- драйвер індикатора.

Проведено тестування макету цифрової системи передачі даних. У ході тестування отримані позитивні результати роботи системи передавання даних.

Результати роботи можуть бути використані у якості лабораторної бази для вивчення предметів, що зв'язані із програмуванням мікроконтролерів і вбудованих систем.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Банкет В. Л. Завадостійке кодування в телекомунікаційних системах / В. Л. Банкет, П. В. Іващенко, М. О. Іщенко. – Одеса : ОНАЗ ім. О. С. Попова, 2011. – 100 с.

2. Банкет В. Л. Сверточные коды в системах передачи информации : учеб. пособ. / В. Л. Банкет. – Одесса : ОЭИС, 1986. – 57 с.

3. Кларк Дж. Кодирование с исправлением ошибок в системах цифровой связи / Кларк Дж. мл., Кейн Дж.; пер. с англ. – М.: Радио и связь, 1987. – 392 с.

4. Блок-схемы алгоритмов. ГОСТ [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: http://pro-prof.com/archives/1462.

5. Возможности Visual Studio [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://msdn.microsoft.com/ru-ru/library/bb386063(v=vs.110).aspx.

6. Скляр Б. Цифровая связь. Теоретические основы и практическое применение / Б. Скляр. [2-е изд., испр.] : пер. с англ. – М. : Изд. дом "Вильямс", 2003. – 1104 с.

7. Банкет В. Л. Цифровые методы передачи информации в спутниковых системах связи / В. Л. Банкет, П. В. Иващенко, А. Э. Геер. – Одесса : УГАС, 1996. – 180 с.

8. Кузьмин Н. В. Основы теории информации и кодирования / Н. В. Кузьмин, В. А. Кедрус. – К. : Вища школа, 1986. – 238 с.

9. Помехоустойчивость и эффективность систем передачи информации / А. Г. Зюко, А. И. Фалько, И. П. Панфилов и др. ; под ред. А. Г. Зюко. – М. : Радио и связь, 1985.– 282 с.

10. Теория передачи сигналов : учебник для вузов / А. Г. Зюко и др. – М. : Радио и связь, 1986. – 304 с.

11. Панфілов І. П. Теорія електричного зв’язку : [підручник для студентів вузів I та II рівнів акредитації] / І. П. Панфілов, В. Ю. Дирда, А. В. Капацін. – К. : Техніка, 1998. – 328 с.

12. Блейхут Р. Теория и практика кодов, контролирующих ошибки / Блейхут Р. ; пер. С англ. – М. : Мир, 1986. – 576 с.

13. Банкет В. Л. Сверточные коды в системах передачи информации : учеб. пособ. / В. Л. Банкет. – Одесса : ОЭИС, 1986. – 57 с.

14. Теория электрической связи : учебник для вузов / [А. Г. Зюко и др.] ; под ред. Д. Д. Кловского. – М. : Радио и связь, 1998. – 432 с.

15. Кларк Дж. Кодирование с исправлением ошибок в системах цифровой связи / Кларк Дж. мл., Кейн Дж. ; пер. с англ. – М. : Радио и связь, 1987. – 392 с.