Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2021 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація шифрованого трафіку через аналоговий тракт”

Виконав здобувач вищої освіти
ІІ курсу, групи _____
ОПП «Комп’ютера інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Майданик О.О.
«____» _____ 2021 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Минайленко Р.М
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *магістр*

Галузь знань *12 "Інформаційні технології"*

Спеціальність *123 "Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерна інженерія"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
"____" _____20___року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Майданику Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація шифрованого трафіку через аналоговий тракт*

2. Керівник роботи *Минайленко Роман Миколайович кандидат техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу №42-13 від 02.08.2021 року

3. Строк подання роботи до захисту *22.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою розробки є програмна реалізація шифрованого трафіку через аналоговий тракт*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Наукова новизна
7. Економічна ефективність розробленої програми.
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Діаграма процесів	*1 аркуш*
Показники економічної ефективності	*1 аркуш*

6. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В., к.т.н., доцент	25.10.2021	12.11.2021
Охорона праці	Оришака О.В., к.т.н., доцент	04.11.2021	20.11.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	28.11.2021 р.	

Дата видачі завдання
«__»_____20 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____20 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Майданик О.О. Дослідження та програмна реалізація шифрованого трафіку через аналоговий тракт. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній магістерській роботі розроблено програмне забезпечення, яке призначено для системи шифрованого трафіку через аналоговий тракт.

Метою розробки є дослідження та програмна реалізація системи шифрованого трафіку через аналоговий тракт.

Об'єктом дослідження є процес реалізації шифрованого трафіку через аналоговий тракт.

Предметом дослідження є методи реалізації шифрованого трафіку через аналоговий тракт.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи шифрованого трафіку через аналоговий тракт.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на мікроконтролерах сімейства STM32.

Програму розроблено в середовищі STM32CubeIDE 1.1.0.

Ключові слова: комп'ютерна інженерія, шифрований канал передачі даних, математичний більярд.

ABSTRACT

Maidanyk O.O. Research and software implementation of encrypted traffic through the analog path. 123 Computer Engineering. Central Ukrainian National Technical University. Kropivnitsky 2021

In this master's thesis, software designed for the system of encrypted traffic through the analog path.

The purpose of the development is the research and program implementation of the system of encrypted traffic through the analog path.

The object of the study is the process of encrypted traffic through the analog path.

The subject of the study is the methods of encrypted traffic through the analog path.

Methods of research are based on methods of encrypted traffic through the analog path.

The result of the work is the software implementation of the system of encrypted traffic through the analog path.

In the process of working on a software model an analysis of existing hardware and software was performed. All components of the software developed are fully described.

User-friendly user interface is developed. These are instructions for working with software.

The program can be used on microcontrollers of the STM32 family.

The program was developed in the STM32CubeIDE 1.1.0 environment.

Keywords: computer engineering, encrypted data channel, mathematical billiards.

ЗМІСТ

					ВКРМ-123.21.0012.00.00.ПЗ		
Вим	Арк.	№ докум.	Підп.	Дата			
Розроб.		Майданик О.О.			Дослідж ення т а програмна реалізація шифрованого т рафіку через аналоговий т ракт	Літ.	Аркуш 1 / Аркушів 102
Перев.		Минайленко Р.М				М	
Н.контр.		Гермак В.С.				ЦНТУ КІ-20М	
Затв.		Смірнов О.А.					

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ГВЧ – генератор випадкових чисел.

ГПВЧ – генератор псевдовипадкових чисел.

КБ – кібербезпека.

ОС – операційна система.

ПВЧ – псевдовипадкове число.

ХБС – хаотичний більярд Сіная.

MAC (англ. Message Authentication Code) - спеціальний набір символів, що додається до відповідного повідомлення для забезпечення цілісності повідомлення та аутентифікації джерела даних.

DES (англ. Data Encryption Standard) - це симетричний алгоритм шифрування певних даних.

AES (англ. Advanced Encryption Standard) - симетричний алгоритм блочного шифрування.

ARM Cortex - це група 32-бітних і 64-бітних процесорних ядер.

USB (англ. Universal Serial Bus) - універсальна послідовна шина, призначена для з'єднання комп'ютерів і периферійних пристроїв.

UART (англ. universal asynchronous receiver/transmitter) - універсальний асинхронний приймач/передавач.

SPI (англ. Serial Peripheral Interface) - послідовний периферійний інтерфейс.

I2C (англ. Inter-Integrated Circuit) - послідовна шина даних для зв'язку інтегральних схем.

XOR (англ. eXclusive OR) - виключна диз'юнкція або ж додавання за модулем два.

SWD (англ. Serial Wire Debug) – інтерфейс для завантаження програми до мікроконтролера з послідовною шиною.

ВСТУП

Актуальність теми. Більшість організацій дуже цінують переваги електронного документообігу та в зв'язку з цим активно нарощують потужність зберіганнях даних. Великі обсяги даних, які зберігаються вимагають спеціальних заходів кібербезпеки. В наш час інформація - дуже цінний товар, тому багато хто використовує всі можливі способи, щоб заволодіти нею. Встановлення підслуховування, датчиків руху, прихованих відеокамер в квартирах, офісах, автомобілях і телефонах - це давно не рідкість, а буденне явище. Однак багато хто ставиться дуже безтурботно до інформаційної безпеки - власної і бізнесу - і навіть не допускають можливості стеження за собою. Зусилля по захисту щодо усунення можливих витоків інформації докладають лише тоді, коли важливі відомості вже потрапили до рук конкурентів, зловмисників або були оприлюднені, що часто спричиняє до великих втрат, а іноді навіть стає причиною краху підприємства.

Мета й завдання дослідження. Метою даної роботи є дослідження та програмна реалізація шифрованого трафіку через аналоговий тракт.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих реалізацій трафіку через аналоговий тракт.

– Дослідження методів реалізації трафіку через аналоговий тракт.

– Програмна реалізація шифрованого трафіку через аналоговий тракт.

Об'єктом дослідження є процес реалізації алгоритму шифрованого трафіку на основі математичного більярду Сіная.

Предметом дослідження є методи реалізації шифрованого трафіку за допомогою хаотичного більярду Сіная.

Методи дослідження базуються на методах розробки програмного забезпечення, методах математичних обчислень, макетування пристрою.

Наукова новизна отриманих результатів. Полягає в тому, що:

- вперше застосовано для шифрування трафіку алгоритм шифрування побудований на засадах хаотичного (ентропійного) математичного більярду;

- досліджено та промодельовано швидкість шифрування та дешифрування, стійкість проти впливу зовнішніх факторів на реальних пристроях зокрема на ARM мікроконтролерах;

- алгоритм і ПЗ можуть бути використані в системах швидкої передачі повідомлень як по цифрових так і по аналогових каналах зв'язку.

Практична цінність отриманих результатів полягають в тому, що розроблена система дозволяє забезпечити шифрований трафік через аналоговий тракт.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, макетування пристрою відповідністю отриманих результатів окремими результатами, наведеними у науковій літературі.

Робота апробована на науково-технічної online конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» за підсумками проведення «Дня науки - 2020», основні положення магістерської роботи надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №11.

Також робота апробована на міжнародної науково-практичній on-line конференції "Проблеми енергоефективності та автоматизації в промисловості та сільському господарстві".

Головна перевага розробленого проекту – безпечна шифрована передача даних різними каналами зв'язку. Розроблена система шифрування має можливість адаптації до старих аналогових систем зв'язку.

Окремо розроблено керуючу програму для мікроконтролера, яка налагоджує радіообмін даними через радіомодулі загального призначення; апаратна частина яка передбачає обмін даними точка-точка; розроблені макети, які дозволяють вести обмін даними між комп'ютерними пристроям через радіомодулі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Сучасна КБ базується на різних засобах та методах. Один з методів захисту інформації є криптографія. Сучасна криптографія базується на різних математичних концепціях. Ці концепції використовують розділи математики – теорія чисел, теорія імовірності, теорія складності обчислень. Повідомлення або дані які передаються та зберігаються, шифруються і розшифровуються за допомогою складних алгоритмів.

У своїй більшості алгоритми шифрування відомі (симетричні, асиметричні, хеш-функії). При застосувані таких систем потрібно мати достатньо довгий ключ побудований на простих числах і пошук таких чисел є не тривіальною задачею.

Використання хаотичних систем у криптографії вивчається вже давно. Використання хаотичної криптографії добре зарекомендувала себе в шифруванні для зображень відео і аудіо. Вони характерезуються репродуктивністю та простотою реалізацією. У більшості випадків хаотичні алгоритми використовують для створення [2] генераторів псевдовипадкових чисел (ГПВЧ). У більшості випадків сучасні хаотичні більярди для побудови ГПВЧ.

Хаотичні більярди це двовимірні математичні системи на яких моделюють термодинамічні та квантові системи, вони є класом простих систем на яких вивчають природний хаос. У 1970 році академік Сінай розробив теорію хаотичних більярдів яка вивчається в теорії динамічних систем та теорії статистичної механіки.

У загальному вигляді ХБС представляє собою поле по якому рухається частинка з постійною швидкістю і відбивається від границь відповідно до закону відбиття кутів. Прийняті кути положення частинки в полі більярда можна

розглядати як випадкові значення які можна використовувати для побудови нового ГПВЧ. ХБС вважають першим класом хаотичним більярдом які також називають дисперсними.

У нашому випадку при опуклих сторонах поверхні рух кульки (математичної точки) викликає розбіжність траєкторії і таким чином зміщення початкової координати руху створює нову траєкторію або генерує новий хаотичний ряд.

Ідея застосування хаотичного більярду Сіная (ХБС) полягає в тому, що такі більярди мають ентерпійну характеристику траєкторії руху кульок неможливо відтворити навіть знаючи декілька попередніх координат. ХБС став популярним останні десять років. Такі більярди можуть складатися (мати геометричний вигляд різного виду):

– Опуклі криві.

– Увігнуті або лінійні.

– Прямокутний з внутрішнім колом.

Сінай показав в [3], що всі більярди із зовні опуклими межами є завжди сильно хаотичними. Такий більярд завжди хаотичний і траєкторії руху кульки всередині більярду завжди хаотичні. Сінай довів, що всі ХБС є аргодичними, такими що, перемішуються.

1.2 Область застосування

На основі ХБС [4] можна побудувати мобільні системи з шифрованою передачею даних які пов’язані з системами керування і управління технологічними даними наприклад: роботизовані мобільні системи, технологічні процеси різного напрямку, розумний дім. Такі системи захищеного каналу передачі даних повинні забезпечувати надійний захист та надійність від атак на елементи передачі даних. Система повинна забезпечувати [5] надійний

шифрований зв’язок точка-точка, що в свою чергу дозволяє в одному каналі зв’язку передавати одночасно декілька шифрованих потоків від різних об’єктів.

Сучасні комп’ютерні пристрої та операційни системи (ОС) оснащені системами криптозахисту [6] або такі системи можна придбати. Мікроелектронні системи оснащуються мікрозасобами вбудованими в архітектуру ядра мікроконтролера. При цьому ціна таких контролерів перевищує ціну стандартного мікроконтролера такого ж класу приблизно в два рази. Також не всі фірми розробники можуть використовувати ці засоби так як випускають мікроконтролери низької розрядності (8,16 біт).

Запропонований алгоритм [7] може бути застосований на відносно “слабких” мікроконтролерах які вбудовуються в системи управління різноманітними системами і вимагають захисту каналів передачі та прийому даних.

Актуальним залишається питання шифрування голосових даних. У більшості випадків [58] вона пов’язана з великим потоком даних для обробки і передачі. Передбачається, що така система може забезпечити шифрований зв’язок на цифрових мобільних пристроях.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень по профілю теми кваліфікаційної магістерської роботи

На сьогоднішній день існує багато пристроїв комунікації з різноманітними каналами зв’язку.

Взагалі суть всіх методів шифрування [9] заключається в перетворенні відкритих даних в зашифровану форму. За допомогою спеціальних алгоритмів кодування проводиться із використанням ключа шифрування. Зворотний процес декодування також вимагає ключа. Існують два можливі варіанти - симетричний та асиметричний. Симетричне кодування і декодування використовує один і той же ключ. Асиметричне кодування і декодування використовує один ключ, а для декодування - інший. На рисунку 2.1 зображено механізм шифрування даних.

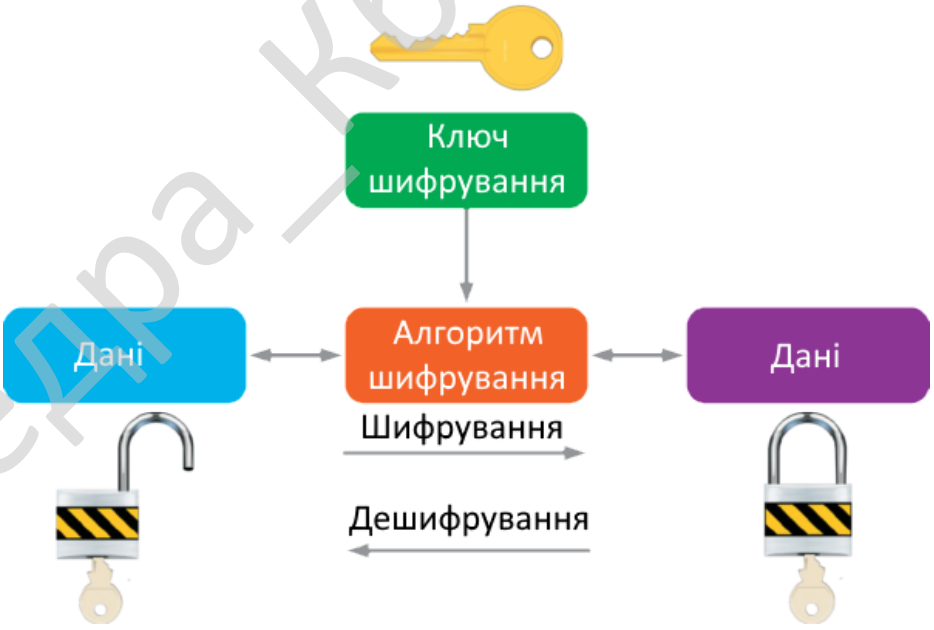


Рисунок 2.1 – Механізм шифрування даних

Функції хешування (згортки) [11] - це односторонні функцій, алгоритм яких перетворює з вихідного масиву даних на бітовий рядок заданої довжини. Вихідні дані часто називають ключем (повідомлення), а вихідні бітовим рядком «хеш-кодом», «хеш-сумою» або «зведенням повідомлень». Алгоритм перетворення який використовується повинен забезпечувати захист від можливостей відновлення вихідних даних за відомим хеш-кодом та від збігів хеш-кодів різних повідомлень. Тобто різні повідомлення після перетворень ні в якому разі не повинні бути однаковими. На рисунку 2.2 зображено роботу хеш функцій.

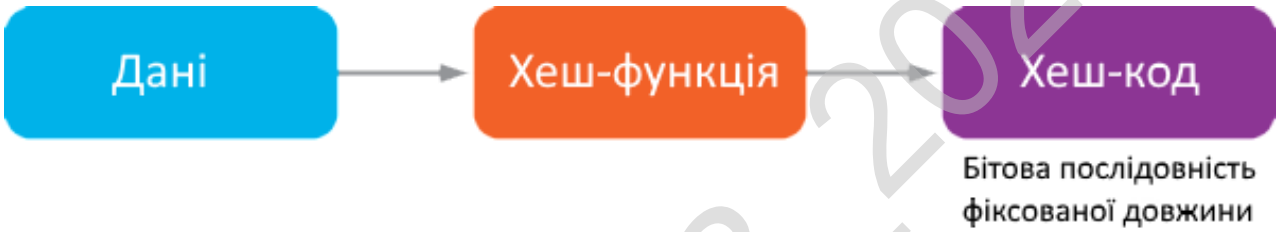


Рисунок 2.2 – Робота хеш функцій

Код аутентифікації повідомлень MAC (Message Authentication Code) [12]. Цей програмний механізм використовують для генерації хеш-коду як і звичайні хеш-функції. Але MAC для кодування використовує не тільки вихідне повідомлення, а ще й секретний ключ, який відомий тільки відправникові та одержувачеві. Таким чином одержувач даних може перевірити цілісність інформації для ідентифікації відправника. Без секретного ключа відправником сформується невірний хеш-код, що дає змогу одержувачу знайти невідповідність. На рисунку 2.3 зображено механізм роботи MAC алгоритмів.

Цифровий підпис. Механізм цифрових підписів дозволяє виконати аутентифікацію повідомлень. Тобто довести їх достовірність за допомогою цифрового підпису.

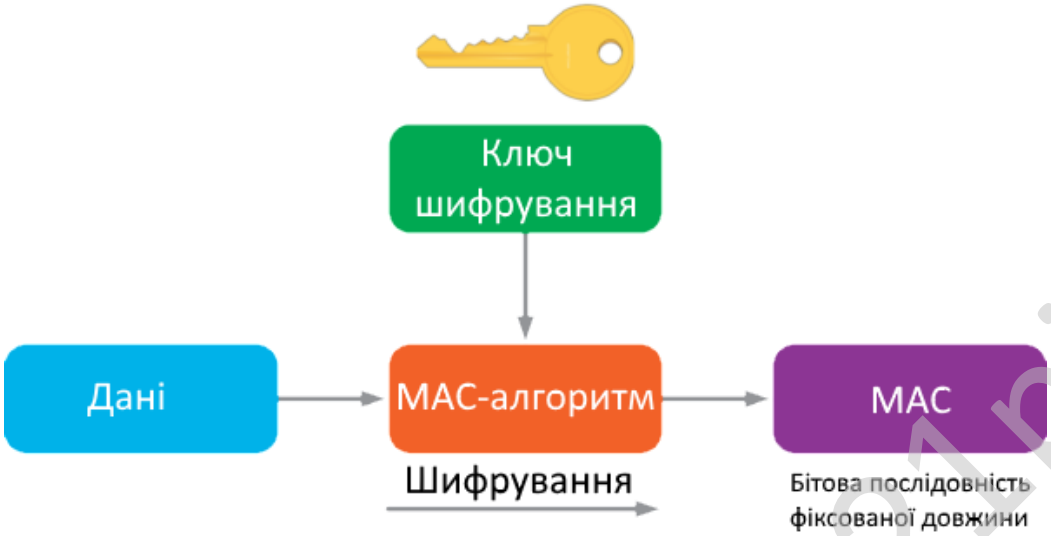


Рисунок 2.3 - Механізм роботи MAC – алгоритмів

Цифровий підпис працює за тим же принципом, як і звичайн ий підпис на папері - по підпису завжди розпізнається відправник. Для отримання цифрового підпису застосовується асиметричне шифрування [13]. Шифрування повідомлення використовує закритий ключ, а для розшифрування - відкритий. Закритий ключ відомий тільки відправнику, а доступ до відкритого ключа може мати більшість одержувачів даних. На рисунку 2.4 зображено механізм формування цифрових підписів.

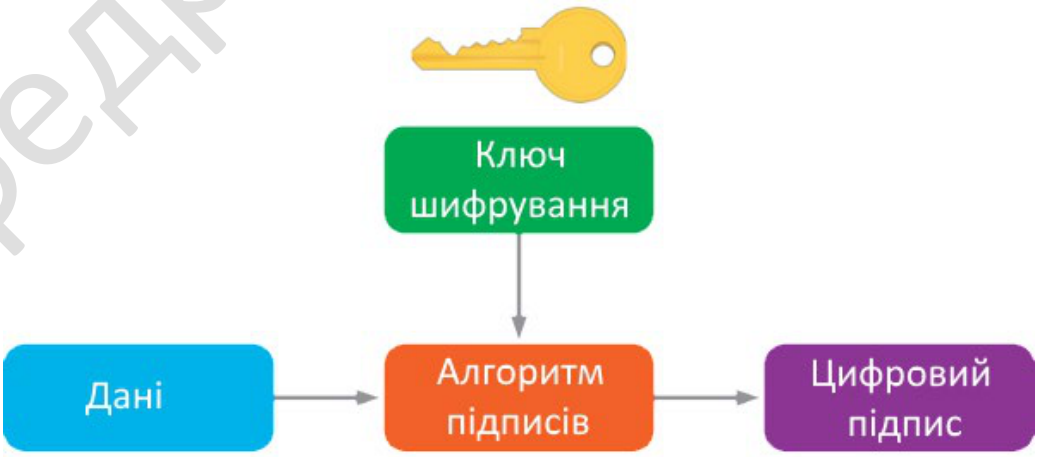


Рисунок 2.4 - Механізм формування цифрових підписів

Генерація випадкових чисел. Кодування даних має сенс, тільки у випаду коли ключ шифрування розгадати неможливо. Тому ключ шифрування повинен бути випадковим. Для такого шифрування застосовують генератори псевдо випадкових чисел. ГПВЧ може використовувати різні методи формування вихідної послідовності псевдовипадкових чисел, такі як природні джерела випадкових сигналів (наприклад, шум напруги на напівпровідниковому діоді) або класичні програмні. На рисунку 2.5 зображено приклад роботи генератора псевдовипадкових чисел.

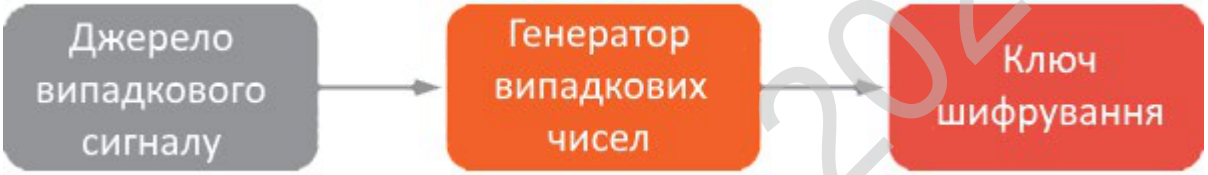


Рисунок 2.5 - Приклад роботи генератора псевдовипадкових чисел

В даній роботі буде розглянуто пристрої з безпровідним каналом зв’язку. Та можливі реалізації шифрування на мікроконтролерах. Тому слід розглянути платформу від одної з провідних компаній по виробництву напівпровідників та мікроконтролерів – STMicroelectronics. Дана компанія надає рішення в сфері 32-х бітних мікроконтролерів, в старших серіях яких є блоки апаратного шифрування. Компанія [14] STMicroelectronics створила - програмно-апаратну платформу відкритого середовища розробки STM32 ODE (STM32 Open Development Environment). STM32 ODE - середовище яке об'єднує в собі інструменти для роботи з апаратною частиною та набір програмних бібліотек. Набір бібліотек реалізований у вигляді єдиного комплексу STM32 Cube IDE. Середовищем розробки STM32 Cube IDE підтримується вся лінійка 32-х розрядних мікроконтролерів STM32 [15] та плати для розробників (Evaluation boards), стартові набори налагоджувальних плат (Discovery boards), базові плати (Nucleo

boards) і плати розширення для Nucleo. На рисунку 3.6 зображено плату Nucleo.

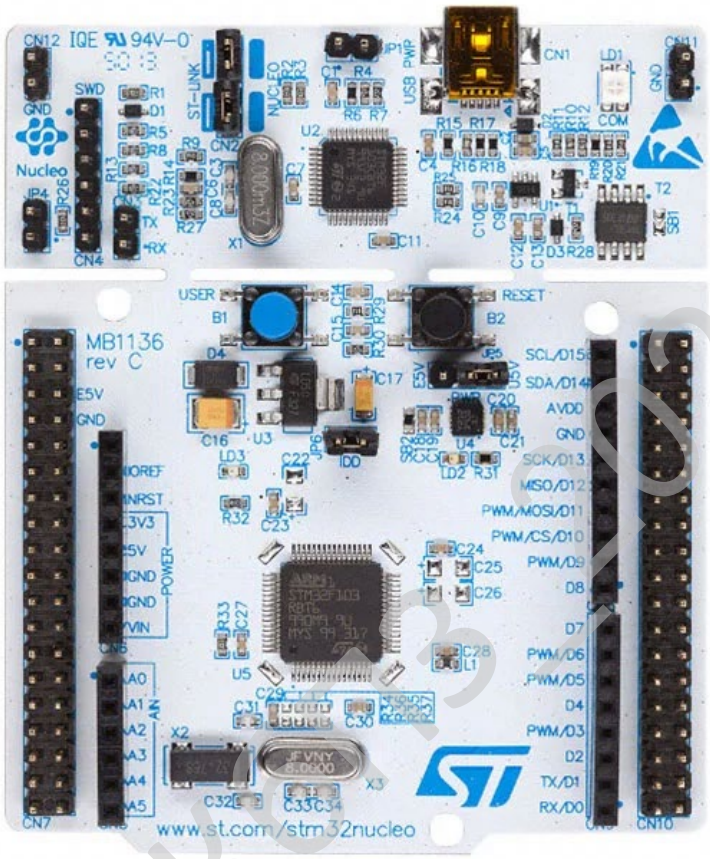


Рисунок 2.6 – Зображення налагоджувальної плати Nucleo

STM32 Cube IDE - програмна частина платформи STM32 ODE яка об'єднує програмне забезпечення кількох рівнів:

- Рівень драйверів які залежних від апаратної частини (бібліотеки периферії);

- Рівень незалежних апаратної від апаратної частини драйверів для мікроконтролерів (hardware abstraction layer) і драйвери для плат розширення (board support package, BSP);

- Програмне забезпечення проміжного рівня Middleware (USB, Bluetooth стеки протоколів та додаткові бібліотеки, наприклад, бібліотека шифрування);

- ПЗ прикладного рівня - стандартні приклади для плат розробників (налагоджувальних плат).

- Пакет розширення для додатків з шифруванням X-cube-cryptolib [16] які входять до складу STM32 Cube IDE. Пакет включає в себе ПЗ, яке працює на двох рівнях. На рівні додатків і на проміжному рівні. На рисунку 2.7 зображено положення бібліотеки X-cube-cryptolib в системі ПЗ від STMicroelectronics.

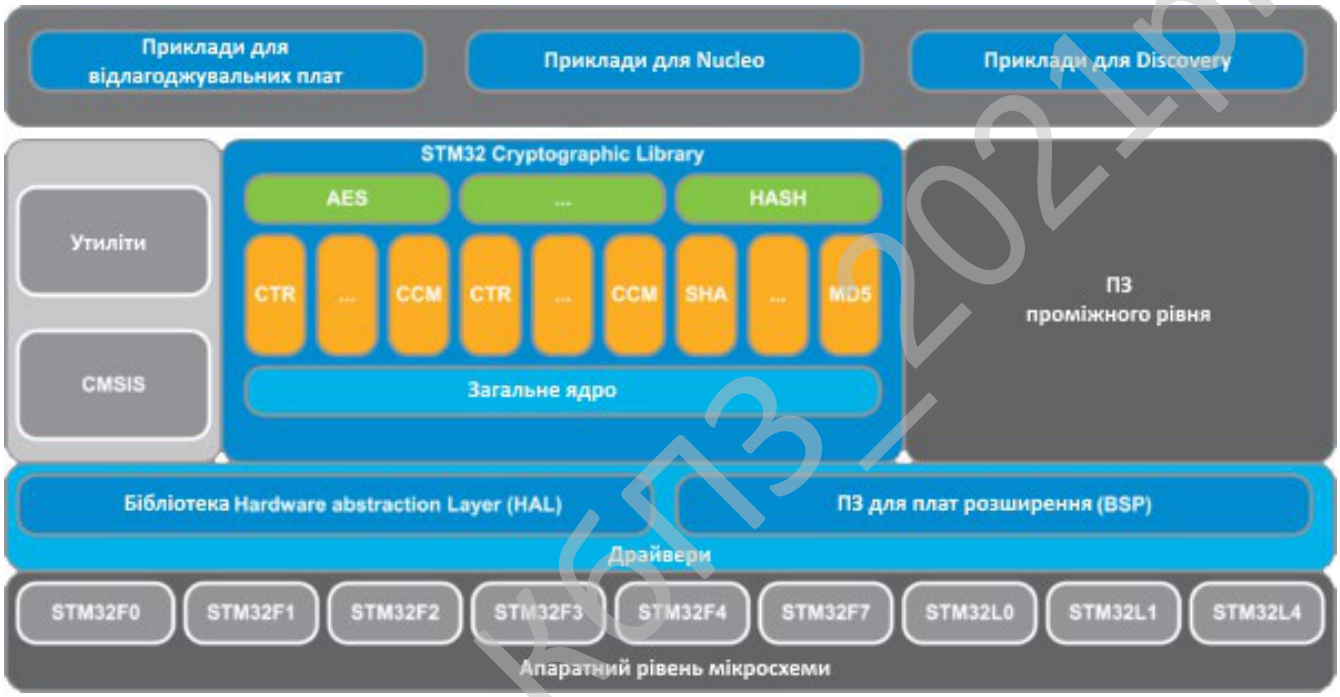


Рисунок 2.7 - Положення бібліотеки X-cube-cryptolib в системі ПЗ від STMicroelectronics

Для розробників пропонується 31 приклад ПЗ прикладного рівня для кожного алгоритму захисту. Розробникам доступні шаблони для середовищ розробки IAR, Keil ARM®, GCC які є найбільш популярними і шаблони для різних мікроконтролерів налагоджувальні плати, стартові набори, плати Nucleo.

Все ПЗ прикладного рівня представляється у вигляді відкритих файлів на мові програмування - С. Користувачі легко можуть застосувати частини коду в своїх проектах, змінюючи та доповнюючи їх, що дає можливість максимально

швидко налагодити роботу з бібліотекою шифрування, яка виконує свої функції на проміжному рівні.

Бібліотека шифрування STM32 crypto library - це набір скомпільованих файлів для всіх підтримуваних сімейств STM32 мікроконтролерів і інтегрованих середовищ розробки. Реалізації скомпільованих рішень будується за принципом модульності. Структура модульності дозволяє використовувати компілятору тільки ті модулі, які потрібні користувачеві, наприклад, AES CCM, AES CTR, HASH SHA, а інші, для економії ресурсів та пам'яті, не включати в проект. Такий підхід формування проекту дозволяє, при необхідності, додавати програмні модулі в будь-який час роботи над керуючою програмою МК.

Тому як б ібліотеки шифрування представлені у вигляді скомпільованих файлів бібліотек, то користувачі можуть лише включати їх в свої проекти, але не вносити зміни в них неможна. Через це кожна бібліотека поставлена в декількох варіантах для кожного з середовищ розробки (Keil ARM®, GCC, IAR) та з різними можливостями, оптимізація за обсягом пам'яті і оптимізація по швидкості.

STM32 crypto library як бібліотека шифрування знаходиться на проміжному рівні (Middlewares) та доступна як набір скомпільованих файлів об'єднаних в дві групи:

- Група апаратно незалежних версій блоків шифрування (Firmware implementation) - не використовують блоки шифрування які спеціально виконані апаратно на самому чіпі МК , тому здатні працювати з усіма мікроконтролерами STM32 від наймолодшої серії STM32F0 до найпродуктивніших STM32F7;

- Група бібліотек апаратно залежних прискорювачів (Hardware acceleration) - бібліотеки для МК STM32 з окремими периферійними вбудованими в чіп блоками шифрування. Інтегрованих модулі криптографії значно прискорюють роботу керуючих програм МК.

При встановленні X-cube-cryptolib з сайту ST Microelectronics [17] програміст отримує архів STM32 Cube Expansion-Crypto-V3.1.0. При розархівуванні створюється складна мережа вкладених директорій з файлами.

Файли бібліотеки шифрування для апаратно залежних блоків, знаходяться в папці AccHw_Crypto, а ті, що незалежні від апаратної частини - в директорії Fw_Crypto. Розташування файлів в цих папках однакова для кожної із серій STM32 (STM32F0, STM32F1, STM32F2) створена своя директорія. При цьому скомпільовані бібліотеки поставляються для кожної версії у кількох варіантах, що відрізняються сімейством МК який використовується, компілятором і оптимізатором.

Як висновок компанія STMicroelectronics пропонує розробникам пристроїв на основі МК STM32 безкоштовний програмний пакет X-cube-cryptolib для створення захищених додатків. Для роботи доступні скомпільовані файли бібліотек шифрування даних для різноманітних сімейств МК і середовищ розробки. Також користувачам доступні апаратно незалежні програмні модулі та апаратно залежні від вбудованих блоків прискорювачів варіанти реалізації. Бібліотеку шифрування можна використати з стандартним набором периферії мікроконтролера та вести обмін даними через будь-який канал зв’язку.

Серед безлічі видів каналів зв'язку найбільш часто виділяють дротові канали зв'язку (наприклад кабельні, повітряні, світловодні) та радіозв'язок (супутникові, тропосферні, та інші). В свою чергу прийнято класифікувати такі канали на основі характеристик вхідного і вихідного сигналів, а також за змінами характеристик сигналів від таких явищ в каналі як затримки і загасання сигналів.

За типом середовища поширення канали зв'язку діляться на акустичні, оптичні (інфрачервоні), дротові, радіоканали.

Класифікують канали зв'язку на:

- Безперервні (аналогові) у яких на вході і виході каналу - безперервні в часі сигнали.

- Дискретні (цифрові) мають дискретні сигнали на вході і виході каналу зв’язку.

- Безперервно-дискретні аналогово-цифрові на вході - безперервні (аналогові) сигнали, а на виході - дискретні (цифрові) сигнали.

- Дискретно-безперервні цифро-аналогові мають на вході каналу -

дискретні (цифрові) сигнали, а на виході - безперервні (аналогові) сигнали.

Існують канали [18] лінійні і нелінійні, просторово-часові та тимчасові. Часто використовується класифікація каналів зв'язку за діапазоном частот (шириною каналу).

Майбутня система повинна передбачати безперервно дискретний та цифровий канал зв’язку. Тому дана система шифрування може використовуватися в складі окремих пристроїв зв’язку, таких як – рації.

Зараз існує багато пристроїв передачі шифрованої інформації в частості звукової на основі алгоритму - Advanced Encryption Standard (AES) [19]. Дані пристрої досить схожі до головної ідеї індивідуального завдання. Тому слід розглянути сам алгоритм та приклад пристроїв на основі нього.

Advanced Encryption Standard (AES), відомий під назвою Rijndael - це симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128,192,256 біт). стандарт шифрування AES прийнятий американський урядом США для широкого використання. А ктивний аналіз алгоритму підтвердив доцільність як і його попередник, DES [20]. Державний інститут стандартів і технологій (англ. National Institute of Standards and Technology, NIST) США після п'ятилітньої підготовки 26 жовтня 2001 року опублікував попередню специфікацію алгоритму AES. 26 травня 2002 року оголошено стандартом шифрування - AES. Станом на 2009 рік алгоритм симетричного шифрування AES став одним із найпоширеніших.

В середині 90-х років постала велика потреба у новому стандарті шифрування. Використовуваний в ті часи стандарт DES з довжиною ключа кодування 56 біт, дозволяв застосувати тільки метод грубої сили для дешифрування даних. Вже наприкінці 90-х часто відбулись успішні злами даних. Більшою мірою архітектура DES орієнтована на реалізацію апаратно, а реалізація програмно не давала необхідної швидкодії на платформах із обмеженими ресурсами. Модифікація DES 3-DES була ще повільнішою хоч і мала достатню довжину ключа.

AES і запропонований алгоритм Рейменом і Дейцменом, [21] не одне і те ж. Алгоритмом Рейндола підтримується розмір блока та ключа в широкому діапазоні. Натомість алгоритм AES має фіксовану довжину у 128 біт, а розміри ключа можуть бути зі значеннями в 128, 192 та 256 біт. В той час як алгоритм Рейндол підтримує у діапазоні від 128 до 256 біт розмірність блоку та ключа із кроком 32 біт. AES оперує із масивом 4×4 байт через його фіксований розмір блоку, що називається станом (додаткові колонки мають версії алгоритму із більшим розміром блоку).

На основі алгоритму AES існують готові модулі зв'язку, які випускає фірма KENWOOD який називають The Secure Cryptographic Module (SCM) або ж Захищений криптографічний модуль - це апаратний криптографічний модуль, який надає підтверджені функції криптографічної безпеки. Апаратний криптографічний модуль зображено на рисунку 2.8.

Рисунок 2.8 - Апаратний криптографічний модуль

Динний модуль має контакти л керуючими сигналами та послідовні шини SPI та UART.

На малюнку 2.9 зображено логічну блок-схему SCM. СКМ включає: процесор (DSP), енергонезалежну пам'ять (NVM), годинник, схема виявлення несанкціонованих дій та роз’єм.

У криптографічному [22] модулі SCM ініціалізуючи 256-бітну послугу шифрування чи дешифрування AES або використовуючи послугу AES-OTAR з CBC-MAC або CMAC для підтвердження цілісності KMM модуль переходить у затверджений режим. Будь-які запити на послугу ініціалізації 128-бітного шифрування або шифрування DES або AES або DES - призведе до переходу модуля на не затверджений режим роботи, вихід із затвердженого режиму роботи. За винятком використання не затвердженого алгоритма інших функціональних відмінностей у модулі немає між затвердженим режимом і не затвердженим режимом.

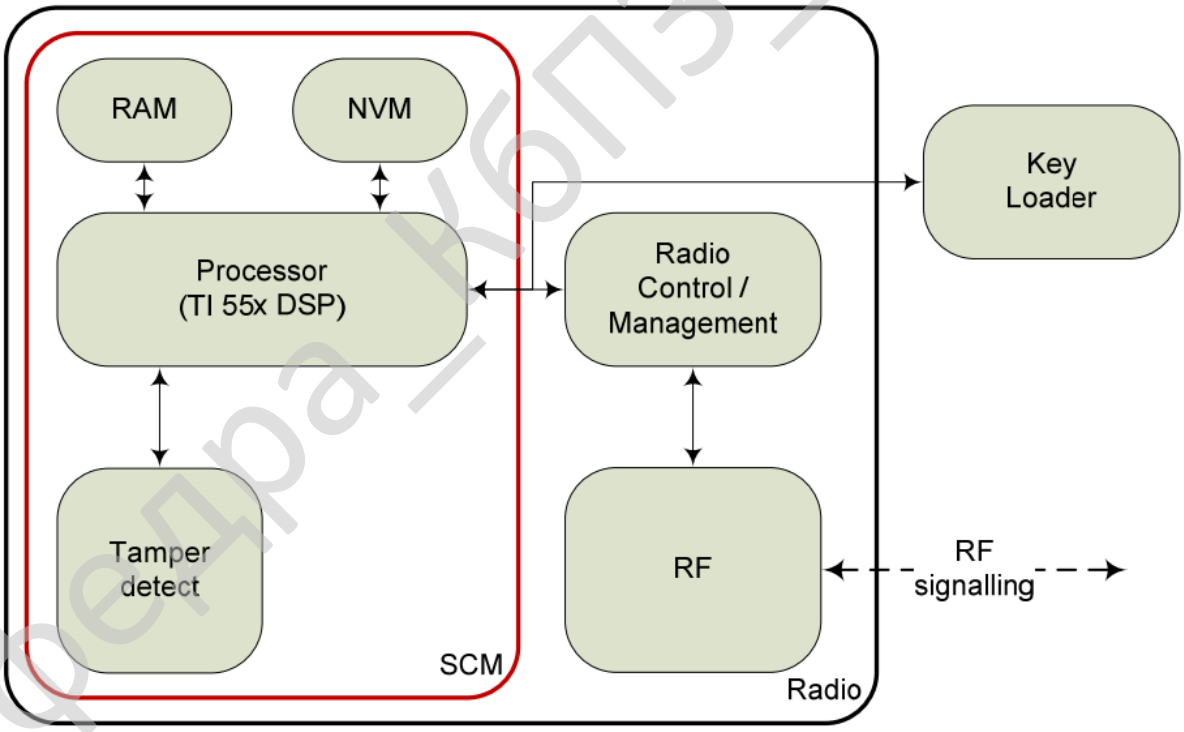


Рисунок 2.9 - Логічна блок-схему SCM

Конструкція модуля відповідає правилам безпеки таким як:

- Політика фізичної безпеки: всі компоненти модуля мають виробничий клас.

- СКМ пригнічує криптографічні операції в станах помилок і пригнічує вихід даних під час самотестів, нульованість та стани помилок.

- Інформація про стан не містить CSP або конфіденційних даних, які при неправильному використанні можуть призвести до компромісу модуля.

- Клавіші вводяться уповноваженими операторами в простому тексті, використовуючи сумісний завантажувач змінних ключів (розсилка вручну, електронний запис) або в зашифрованому вигляді за допомогою автоматизованих методів OTAR.

- Для затвердженого режиму роботи оператор повинен використовувати AES для безпечного зв'язку. DESне слід використовувати у затвердженому режимі роботи.

SCM поєднується з певним радіоприймачем: після виявлення недійсного радіоідентифікатора SCM обнулиться.

- Поведінка обнулення ключових подій SCM події залежить від прапора атрибута Infinite: коли прапор є.

- Набір, подія підробки обнулює всі клавіші AES та AES-OTAR; якщо його не встановлено, подія підробок нульових ключі, що зберігаються в оперативній пам'яті.

Виходячи з цього пристрою випливає великий мінус у вигляді високої ціни, через те що використовується спеціалізований мікроконтролер який призначений для обробки сигналів. Даний вид контролерів значно відрізняється за ціною від звичайних контролерів для вирішення широкого кругу задач. [23]

В статті [див. 24] запропоновані алгоритми моделювання більярду Сіная. Для моделі вибрано форму поля прямокутника з внутрішнім колом зображеного на рисунку 2.10

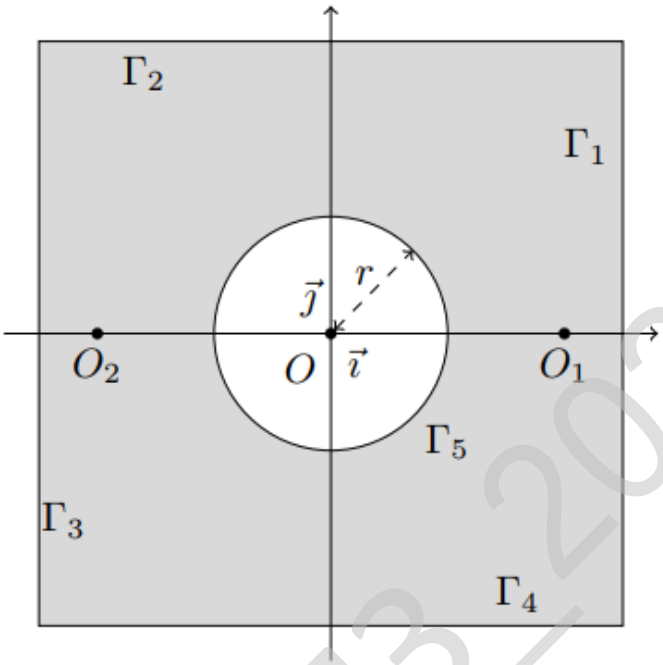


Рисунок 2.10 – Більярд Сіная з прямокутним полем та внутрішнім колом

У більярді, де рухається частинка з постійною швидкістю і відбивається від кордону відповідно до закону: "кут падіння дорівнює кут відображення". Кути та положення, прийнятої частинки можна розглядати як випадкові величини, що спонукали використовувати її при побудові нового ГПВЧ. Сінайський більярд - це перший клас хаотичного більярду, він є також дисперсійною системою. Круглий диск всередині більярд викликає розбіжні траєкторії.

Наведені приклади та алгоритми реалізації гарно ведуть себе при використанні потужних обчислювальних ресурсів. Їх застосування більше підходить до використання на ПК, що обмежує поширення системи для використання на мікроконтролерах. Кращим варіантом для побудови ГПВЧ на МБС є форма з опуклими сторонами без додаткового тіла всередині поля, що спрощує алгоритм розрахунку.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації коду програми, який повинен виконуватися на мікроконтролері було обрано середовище розробки від компанії [25] STMicroelectronics - STM32CubeIDE.

Середовище розробки STM32CubeIDE раніше називалося Atollic TrueSTUDIO та не мало в собі модуля для генерації проекту з частиною коду ініціалізації. Отже після придбання середовища розробки компанією STM воно було оновлене, та в його склад тепер входить модуль генерації проеку, який раніше був окремою програмою під назвою CubeMX.

STM32CubeIDE це вдосконалена платформа розробки C/C++ [26] з периферійною конфігурацією, генерацією коду, складанням коду та налагодженням для мікроконтролерів та мікропроцесорів STM32. Він заснований на рамках ECLIPSE ™/CDT та ланцюжку інструментів GCC для розробки та GDB для налагодження. Це дозволяє інтегрувати сотні існуючих плагінів, які доповнюють функції ECLIPSE ™ IDE.

STM32CubeIDE інтегрує всі функції STM32CubeMX, щоб запропонувати досвід роботи все в одному та заощадити час установки та розробки. Після вибору порожнього MCM або MPU STM32 або попередньо налаштованого мікроконтролера або мікропроцесора з вибору плати створюється проект і створюється код ініціалізації. У будь-який час під час розробки користувач може повернутися до ініціалізації та конфігурації периферійних пристроїв або середнього програмного забезпечення та відновити код ініціалізації, не впливаючи на код користувача.

STM32CubeIDE включає аналізатори збірки та стеку, які надають користувачеві корисну інформацію про стан проекту та вимоги до пам'яті.

STM32CubeIDE також включає в себе стандартні та вдосконалені функції налагодження, включаючи представлення основних процесорів, пам'яті та

периферійних регістрів процесора, а також живу змінну перегляду, інтерфейс переглядача послідовних проводів або аналізатор помилок.

Розглянемо створення проекту у середовищі розробки та загатний вигляд інтерфейсу. Першим кроком після інсталювання середовища розробки та відкриття вікна необхідно створити проект для мікроконтролера (рисунок 2.11).

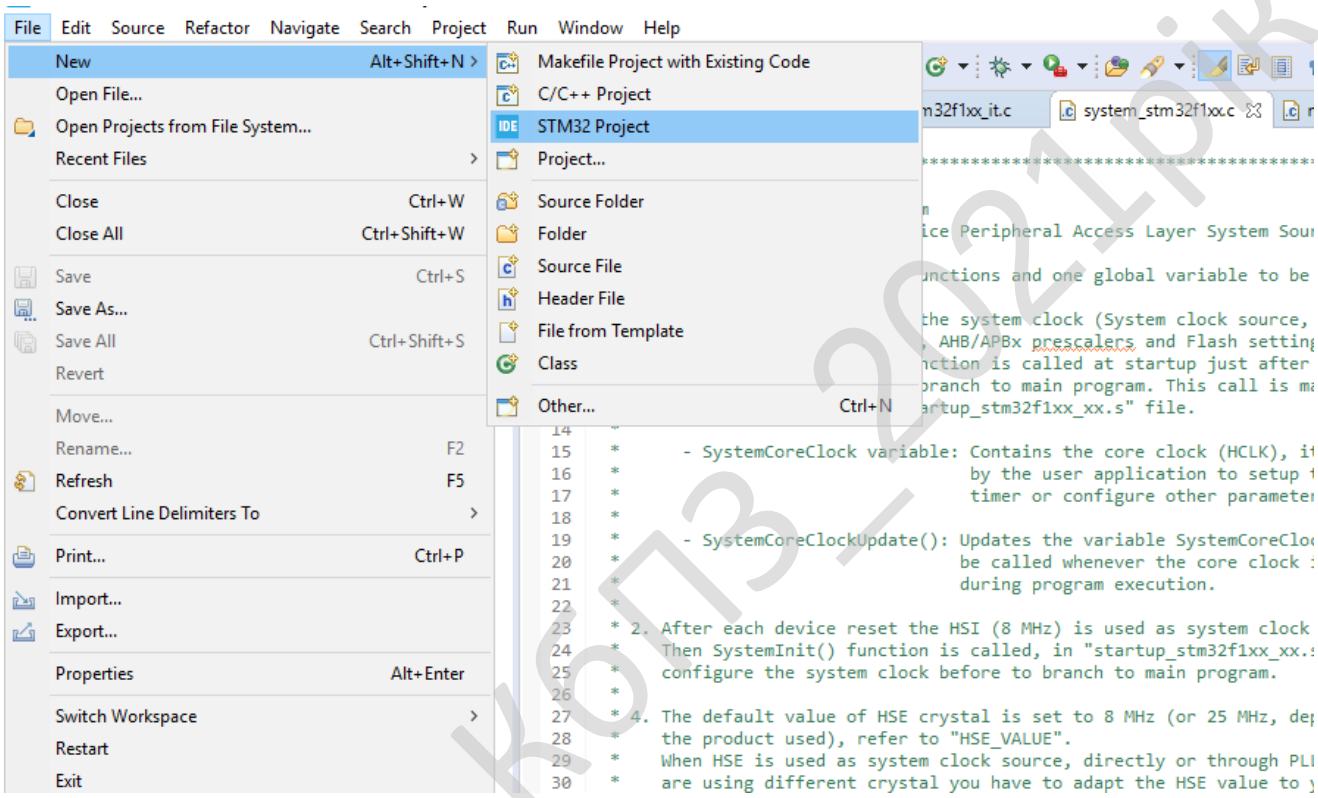


Рисунок 2.11 – Створення проекту

Далі потрібно вибрати конкретний мікроконтролер або одну із налагоджувальних плат. Також є можливість швидкого перегляду характеристик вибраного мікроконтролера у фільтрі зліва вікна. Дане вікно також дозволяє відкрити всю можливу документацію на даний мікроконтролер. Присутні і блок-діаграми. Також є перехід до можливих поставщиків даної мікросхеми або налагоджувальної плати для покупки. На рисунку 2.12 зображено вікно вибору мікроконтролера на який повинна створюватися програма.

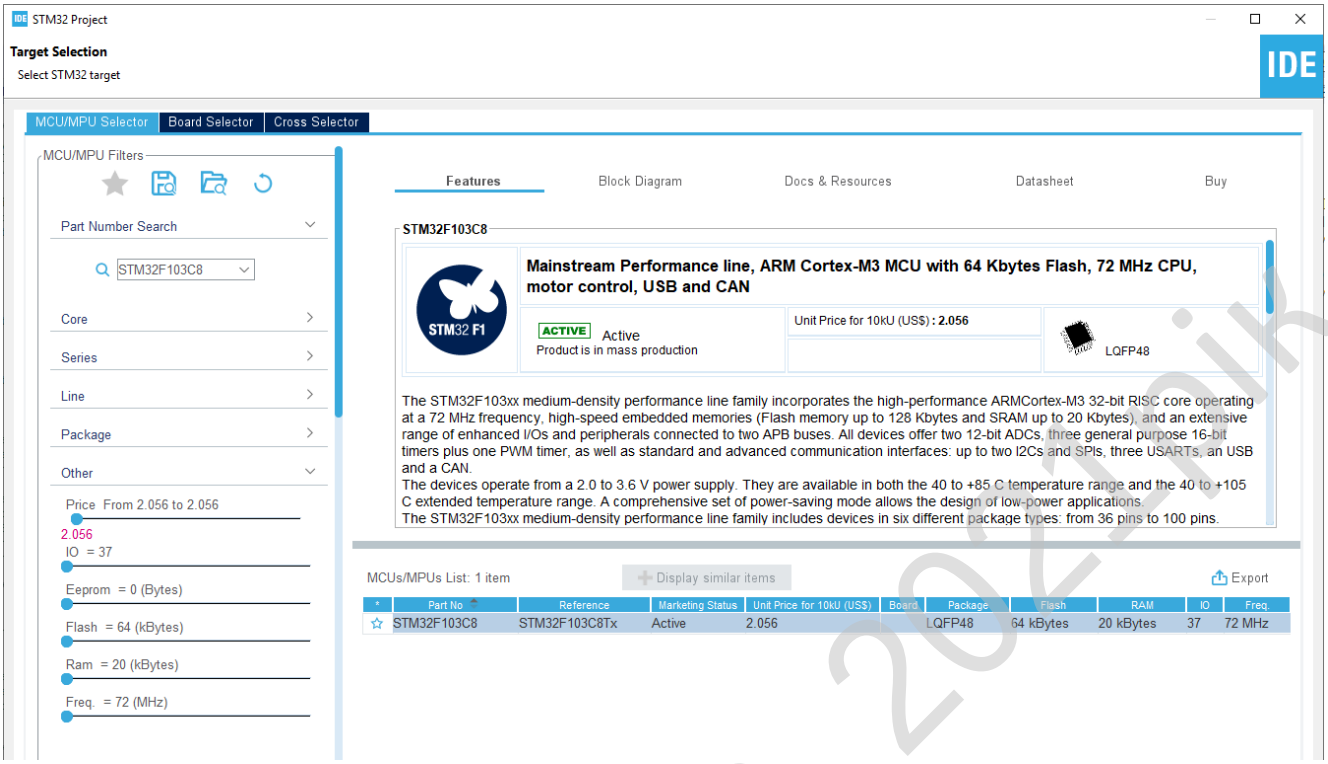


Рисунок 2.12 – Вікно вибору мікроконтролера

В наступному вікні слід вказати ім’я проекту та місце розташування на ПК. На рисунку 2.13 зображено діалогове вікно в якому потрібно вказати назву проекту.

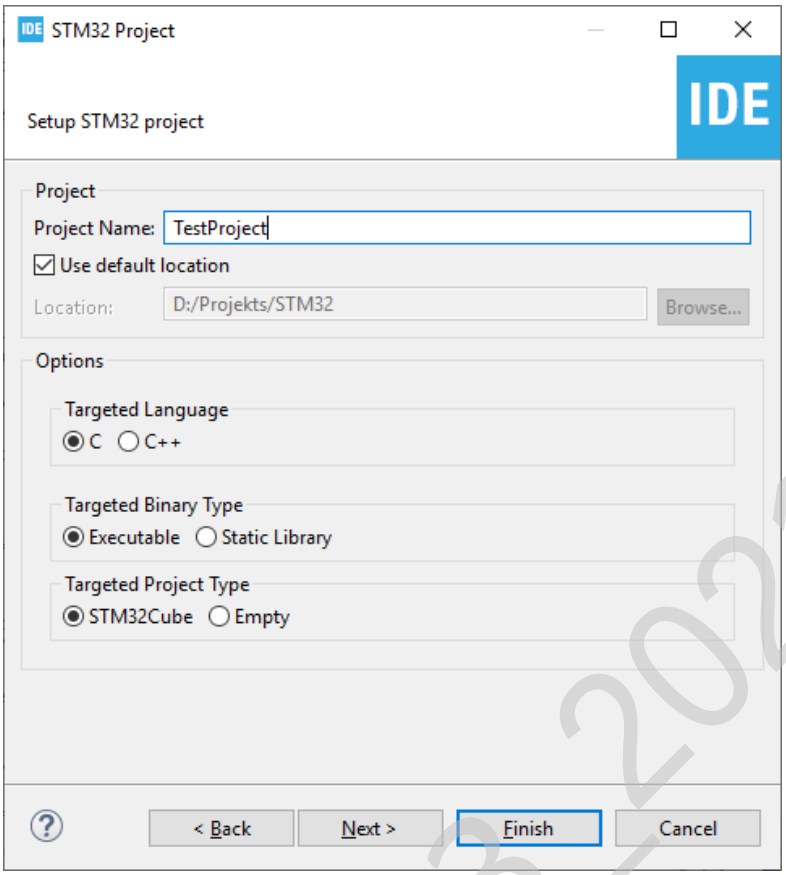


Рисунок 2.13 – Діалогове вікно створення проекту

Наступним вікном буде конфігуратор перефирії мікроконтролера. В даному вікні можливо перед налаштувати джерело тактування процесора та периферії. Також вибрати необхідну частоту джерела тактування та бо спеціальній блок схемі розподілити переддільники частоти для периферійних пристроїв мікроконтролера. Дане вікно також включає в себе зображення самої мікросхеми по периметру якої зображені контакти. Самі контактах підписані та являють собою кнопки з випадаючим списком в якому можна вибрати конкретну функцію яку буде виконувати дана ножка мікросхеми. На рисунку 2.14 зображено вікно конфігуратора.

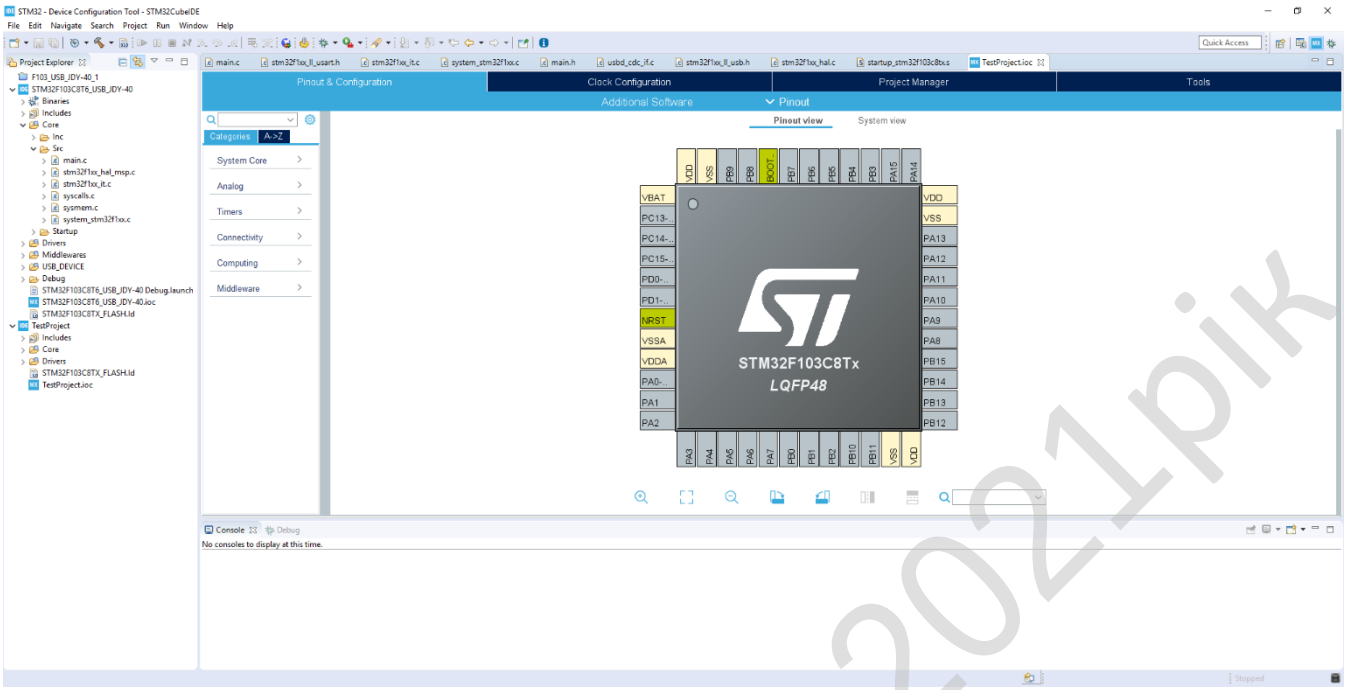


Рисунок 2.14 – Вікно конфігуратора

Потім після натискання кнопки Debug створиться проект. Вікно середовища розробки виглядає досить стандартно тому і не викликає труднощів у освоєнні інтерфейсу. Загальний вигляд вікна зі створеним проектом зображено на рисунку 2.15.

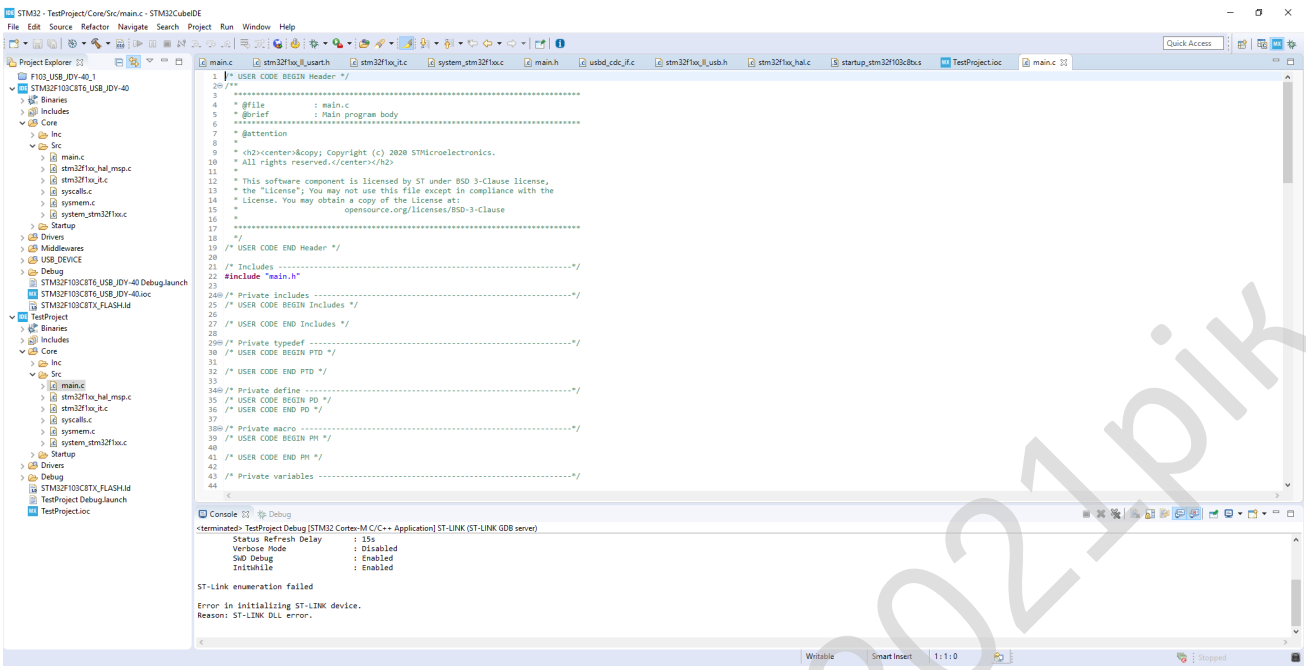


Рисунок 2.15 – Вікно зі створеним проектом

Далі для роботи необхідний програматор. Найкращим варіантом буде обрати програматор який називається - ST-LINK V2. Це внутрішньосхемний програматор відладчик для мікроконтролерів серії STM8 і STM32. Відладчик підключається до налагоджувальних плат за допомогою стандартного SWD (Serial Wire Debug) інтерфейсу (мікроконтролери на базі ядра STM32) або за допомогою SWIM (Single Wire Interface Module) [27] інтерфейсу (для мікроконтролерів сімейства STM8).

SWIM інтерфейс підтримує роботу з 1.65-5.5В платами і роботу в low-speed (9.7 Кб / с) і high-speed (12.8 Кб/с) режимах.

Стандартний 4-контактний SWIM роз'єм. Розташування контактів в точності відповідає оригінальному программатору STLINK / V2

SWD інтерфейс для роботи з ARM CORTEX платами по 3 -х провідному з'єднанню підтримується в діапазоні живлячих напруг 1.65-3.6В. Позначення контактів підключення вказані на корпусі програматора.

Живлення здійснюється за допомогою стандартного USB роз'єму.

Програматор повністю підтримує оновлення прошивки через фірмові утиліти оновлення фірми ST. На рисунку 2.16 зображено програматор STLINK V2.

Рисунок 2.16 - Програматор STLINK V2

Для роботи з самим програматором та можливістю запису програми на контролер необхідно встановити утиліту яка називається [28] ST-LINK utility. Через дану утиліту можна обновити прошивку самого програматора, зчитати та записати програму з мікроконтролера встановити біт захисту від зчитування програми. Дана функція дуже корисна для комерційних проектів. Після встановлення біту захисту пам'ять стає не доступною для зчитування. Але якщо виконати спробу читання програми, то вся пам'ять мікроконтролера очиститься. Дана можливість дає спокійно експериментувати під час створення пристрою не зіпсувавши мікросхему але при наявності вже готового пристрою така спроба

читання пам’яті виведе його із строю. На рисунку 2.17 зображено вікно утиліти [29].

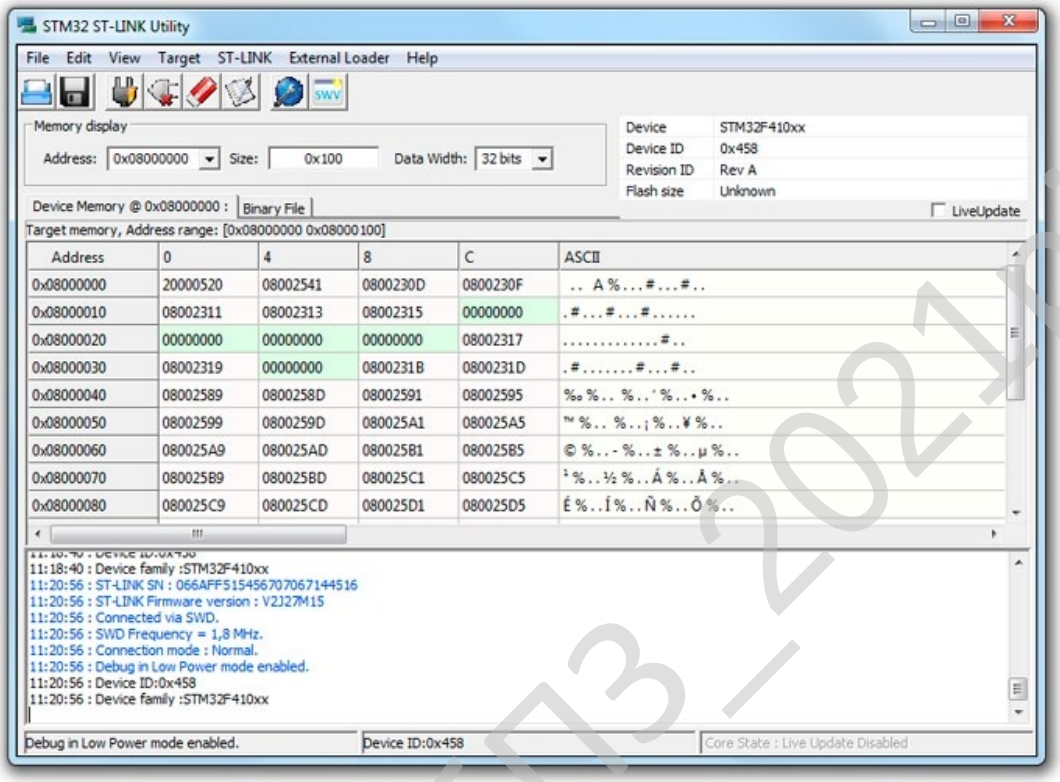


Рисунок 2.17 – Зображення вікна ST-LINK utility

Для подальшого відлагодження пристрою застосовано логічний аналізатор. Логічний аналізатор - електронний прилад, який може записувати і відображати послідовності цифрових сигналів. Він використовується для тестування і налагодження цифрових електронних схем, наприклад, при проектуванні компонентів комп'ютерів і керуючих електронних пристроїв. На відміну від осцилографів, логічні аналізатори мають значно більше входів (зазвичай від 16 до кількох сотень), але при цьому часто здатні показувати лише два рівня сигналу («0») і («1»).

Для роботи був обраний логічний аналізатор Saleae Logic 8.

Saleae Logic - компактний аналізатор на 8 цифрових входів. Дозволяє відстежувати і аналізувати складні протоколи при взаємодії різних датчиків,

виконавчих пристроїв з мікроконтролерами. Зв'язок з комп'ютером за допомогою роз'єму USB. Володіє точною швидкістю обробки даних, захоплення відбувається на частоті до 24 МГц.

Даний логічний аналізатор працює з безкоштовною програмою від фірми виробника. Вікно програми для роботи з аналізатором зображено га рисунку 2.18

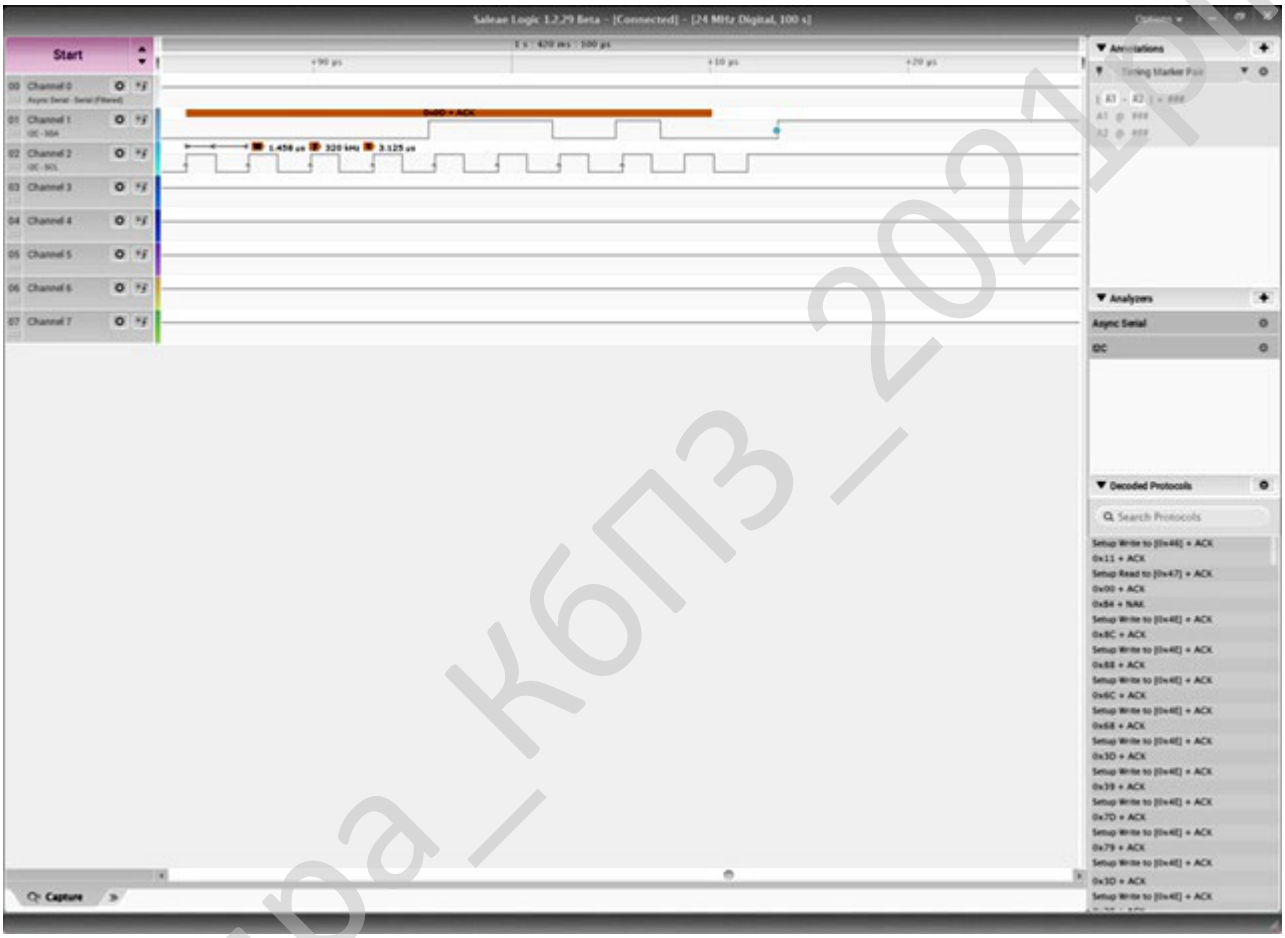


Рисунок 2.18 – Вікно програми Saleae Logic

Із рисунку 2.18 видно логічні рівні напруги та перехопленні дані з послідовної шини UART. Сам логічний аналізатор від’єднується до ПК по USB. Після чого програма автоматично розпізнає яка саме модель аналізатора під’єднана. На рисунку 2.19 зображений логічний аналізатор Saleae Logic 8. [30]

Даний логічний аналізатор в індивідуальній роботі допоможе відслідкувати логічні рівні які йдуть від мікроконтролера до модуля зв’язку по

радіоканалу по шині UART. Також паралельно буде видно логічні рівні контактів керування радіо модуля. Це дасть розуміння повної картини роботи схеми при налагодженні пристрою.

Рисунок 2.19 - Логічний аналізатор Saleae Logic 8

Також для налагодження роботи через USB з ПК буде використовуватися безкоштовна програма універсального терміналу - Terminal 1.9b.

Програма Terminal 1.9b [31] є монітором COM порту персонального комп'ютера. За допомогою програми можна легко відправляти і приймати дані через COM порт комп'ютера по протоколу RS232. Серед достоїнств Terminal гнучке налаштування програми під різні режими роботи. Інтерфейс програми простий і зрозумілий. Вікно термінальної програми зображено на рисунку 2.20.

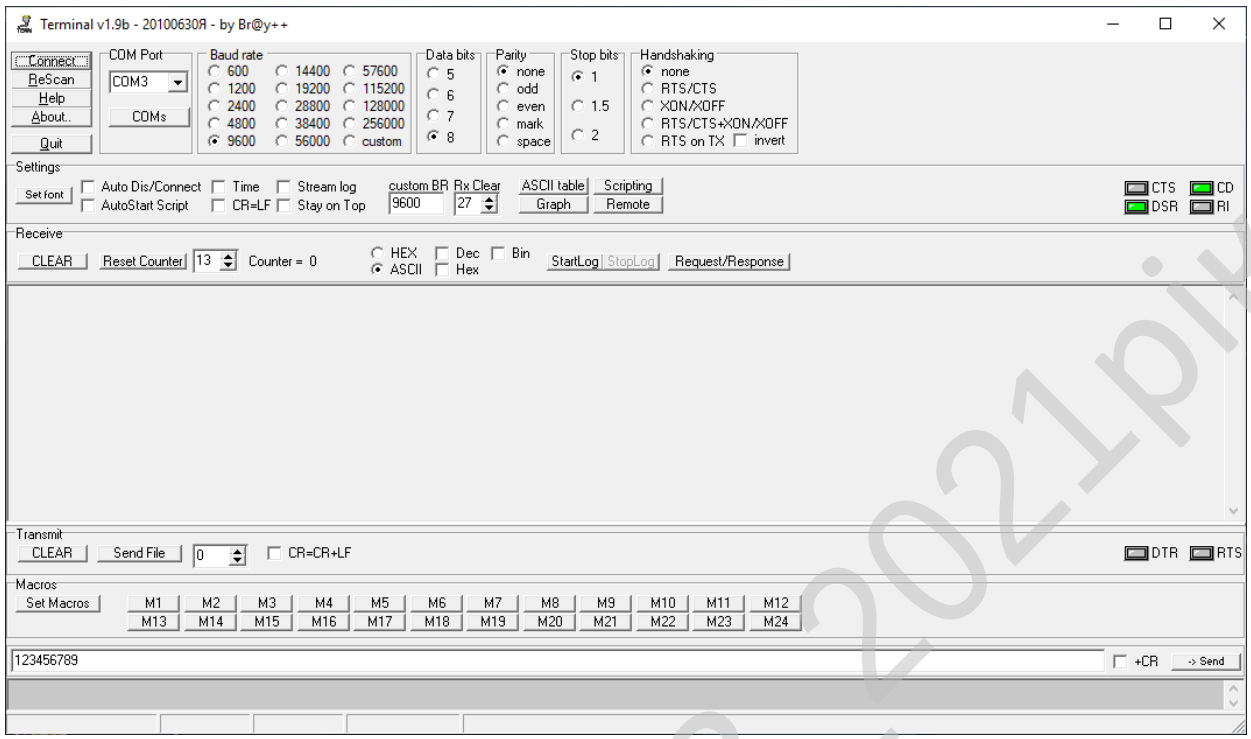


Рисунок 2.20 – Інтерфейс програми Terminal 1.9b

Даного набору засобів для розробки системи більш ніж достатньо для написання програми та налагодження схеми макету пристрою.

2.3 Розгорнута постановка завдання

Наступним кроком необхідно зібрати макетні плати для розробки та дослідження алгоритму на основі ХБС.

Далі необхідно розробити структурну та функціональну схему. Виходячи зі структурної та функціональних схем будується діаграма процесів. Далі розробляється математична модель алгоритму ХБС. Математична модель підтверджується на одній із мов програмування з виводом графічної інформації.

Після підтвердження роботи алгоритму необхідно підібрати компоненти для апаратної реалізації випробувального стенду (макетних плат). Розробка

макетних плат на основі підібраних компонентів. Розробка блок схеми керуючої програми для мікроконтролера.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Сучасному розвитку техніки та зв′язку притаманні дві особливості:

- цифрова (дискретна) форма подання всіх сигналів – незалежно від виду інформації, що передається цими сигналами (мова, текст, дані чи зображення);

- інтеграція обслуговування - повністю реалізується за умови застосування зв′язку тільки цифрової техніки.

Функції мереж зв′язку і кінцевих пристроїв перерозподіляються новітнім способом. Відбувається інтеграція [32] передачі інформації та систем комутації. Все більше з′являються багатофункціональні пристрої які дуже відмінні від телеграфного і телефонного апаратів. Кінцеві пристрої для візуалізації даних, придатні більше ніж для одного виду інформації. Мережа зв′язку дає змогу передавати різноманітну інформацію (текстову, мовну, дані і зображення,) через те саме з′єднання. Ці засоби сильно збільшили продуктивність роботи та економічну ефективність праці трьох важливих галузей промисловості - комп′ютерної індустрії (інформаційних технологій), побутової електроніки (індустрії розваг) і електрадіозв′язку, тому як об′єднання їх зусиль прискорило створення великої інформаційної інфраструктури. Глобальна інформаційна мережа, яка з′явилася в результаті поєднання засобів зв′язку та ЕОМ, що являє собою одну з найскладніших, створених людиною, кібернетичних систем. Система поєднує величезну кількість різних джерел та споживачів інформації. Споживачами можуть бути як найпростіші термінальні пристрої так і ПК, окремі користувачі, так і великі дата-центри чи підприємства, об′єкти, розосереджені на великих територіях та навіть у космосі. Слід уявляти інформаційну мережу як велику систему з великою кількістю користувачів із різним обладнання та засобами зв′язку, для системи керування і надання послуг. Роль зв′язку для

процесу інформатизації дуже значна, оскільки він повністю перекриває весь інформаційний процес від об′єктів спостереження та створення початкової інформації та її обробку в передавачі подальшу передачу даних і обробку у приймачі, до отримання інформації одержувачем в обробленому вигляді. Поняття інформаційної мережі та мережі електрозв′язку стали практично тотожними.

В даній роботі по розробці ПЗ захищеного каналу передачі даних Point to Point для мікроконтролерів необхідно забезпечити взаємодію двох однакових пристроїв по радіоканалу. Взаємодія пристроїв виконується через радіомодулі загального призначення. Дані радіомодулі повинні конфігуруватися з сторони мікроконтролера. В свою чергу мікроконтролер зв’язується з ПК через шину USB. При роботі двох пристроїв один виступає як приймач а інший як передавач.

При передача даних від однієї сторони в іншу виконується під’єднання пристроїв до ПК та з терміналу відправляється текстова інформація. Ця текстова інформація надходить до мікроконтролеру, який повинен прийняти дані в буфер та передати їх через іншу шину яка зв’язана з радіомодулем. Частина периферії яка відповідає за комунікацію через USB повинна бути конфігурована на роботу як комунікаційний пристрій (віртуальний COM порт).

При прийомі даних мікроконтролер повинний так само скласти дані в буфер від радіомодуля та передати їх по USB в сторону ПК.

Дана конфігурація дозволить налагодити комунікацію між двома пристроями. Таким чином на створених макетах можливо налагодити зв'язок точка-точка (Point to Point). Дана робота спрямована на вивчення в подальшому алгоритма шифрування на основі математичного більярду. Тобто робота по переддипломній практиці передбачає початок написання керуючої програми для мікроконтролера, яка дозволить в першу чергу налаштувати канал зв’язку та підготовку до подальшого впровадження алгоритму шифрування.

3.2 Розробка структурної схеми

Структурна схема складатися з трьох осовних компонентів:

– Мікроконтролер на який виконує перетворення та шифрування даних.

– Радіомодуль призначений для передачі даних на інший радіомодуль.

– ПК – виконую роль термінальної передачі даних.

Для роботи радіомодуля та можливості його конфігурації необхідно забезпечити сигнали команда-дані та дозволу роботи. Таких пристроїв потрібно два для забезпечення передачі даних по радіоканалу. Пристрої ідентичні та відрізняються один від одного напрямком передачі даних в даний момент часу. На рисунку 3.1 зображено структурну схему пристрою.

Рисунок 3.1 – Структурна схема пристроїв

Далі для кращого розуміння процесів взаємодії компонентів системи потрібно навести діаграму процесів.

3.3 Розробка функціональної схеми

Згідно із структурної схеми зображеної на рисунку 3.1. Функціональна схема має наступний вигляд (рисунок 3.2).

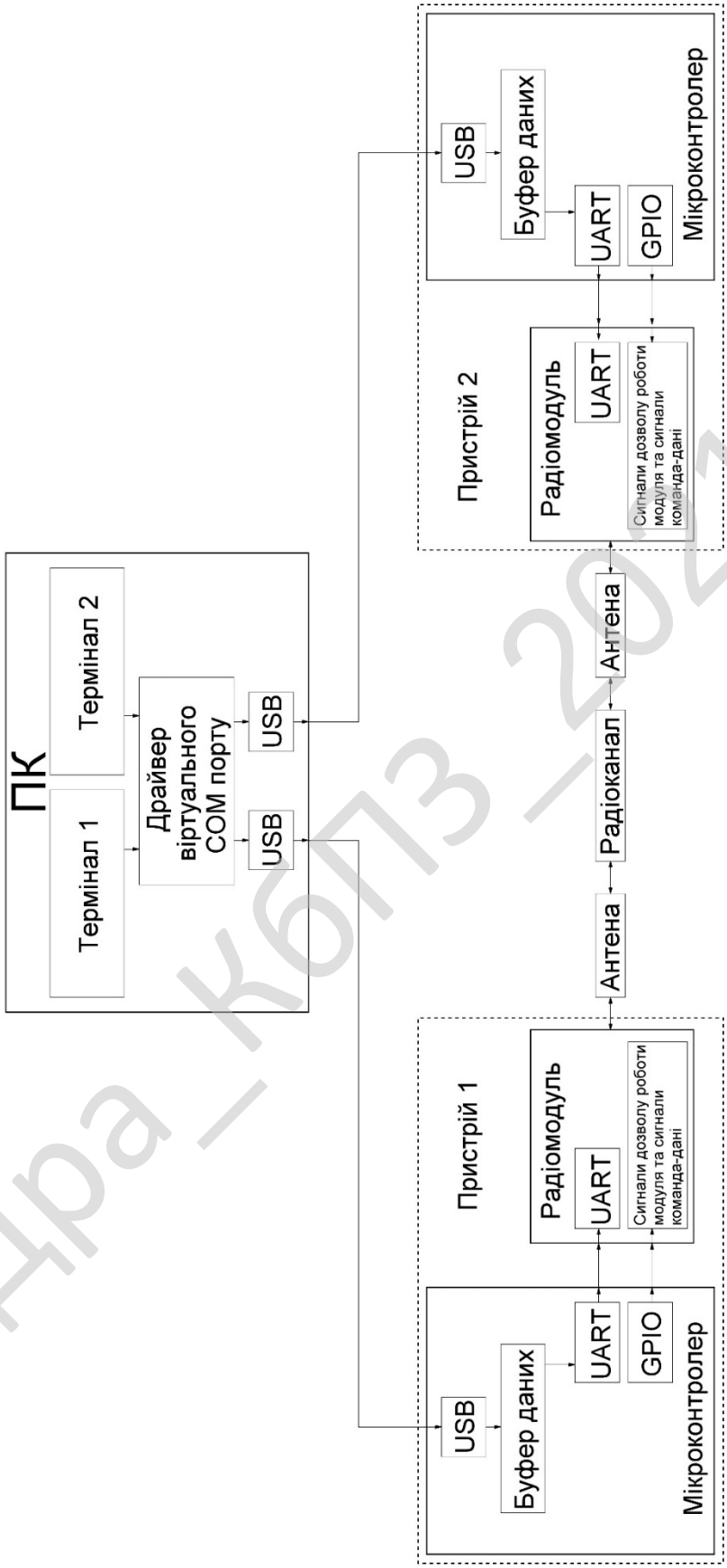


Рисунок 3.2 – Функціональна схема

На ПК згідно з встановленої ОС запускаються зовнішні віртуальні COM (USB) порти та довільні термінали введення виведення даних.Введена інформація передається на відповідний модуль, який встановлео на передачу. На цьому модулі виконується генерація псевдовипадкових чисел (ПВЧ) кодування та шифрування даних і передача через послідовний порт UART радіомодуля.

Другий аналогічний модуль виконує дії в зворотньому порядку. Приймає дані з радіоканалу виконує дешифрування і передає їх по шині USB до ПК на термінал.

3.4 Розробка діаграми процесів

Розробка діаграми процесів дозволяє спланувати всі можливі зв’язки та взаємодію компонентів системи. На рисунку 3.3 зображено діаграму процесів між пристроями системи.

Діаграма процесів розроблена на основі структурної та функціональних схем. Процеси генерації ряду ПВЧ і шифрування виконуються паралельно за допомогою [33] DMA.

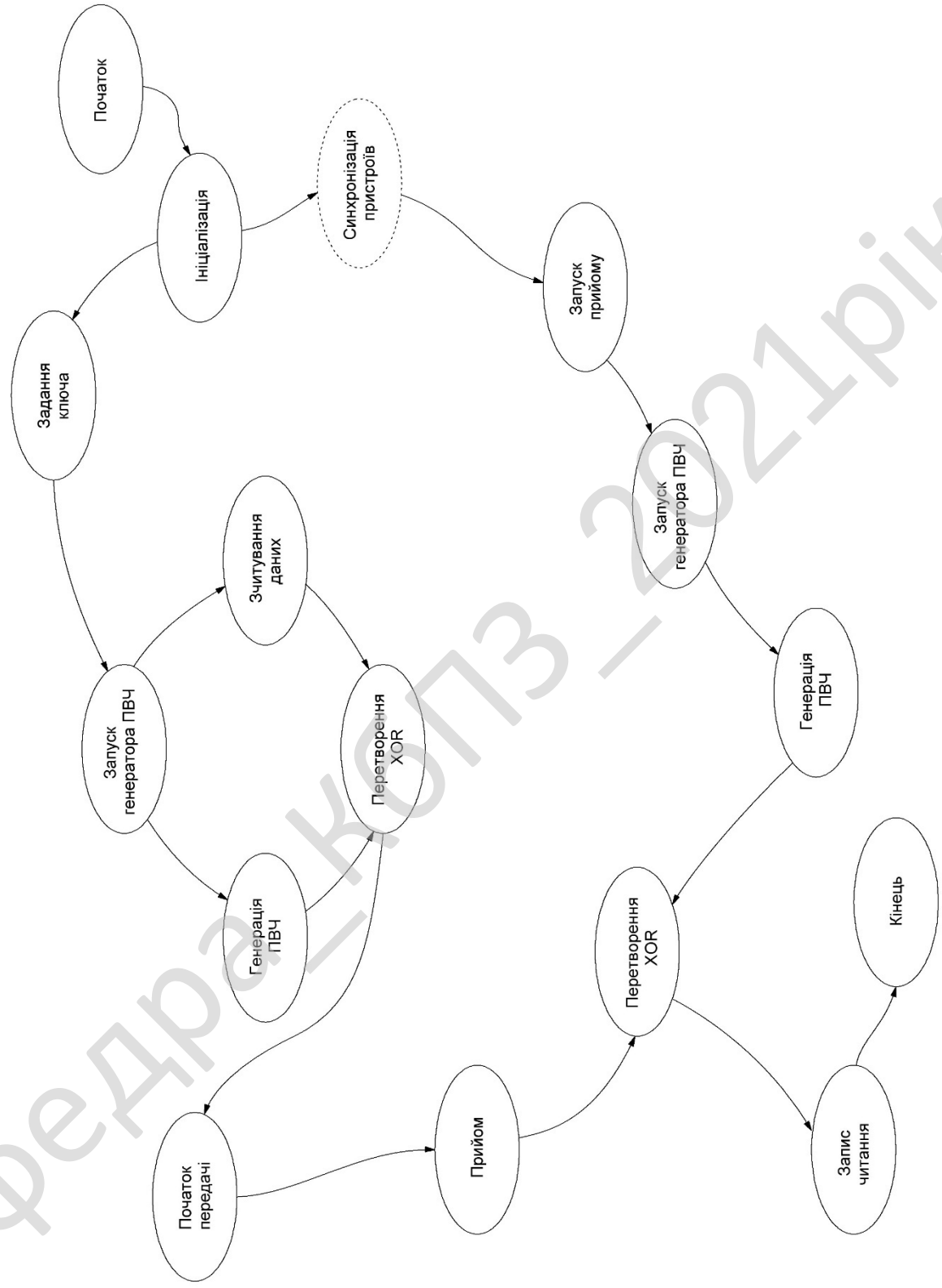


Рисунок 3.3 – Діаграма процесів

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

В даному розділі буде описано апаратну та програмну реалізації проекту.

Для побудови системи створено робочі макети пристроїв. Для створення макету обрано модуль на основі мікроконтролера [34] STM32F103C8T6.

Сімейство високопродуктивних лінійних ліній STM32F103xx включає високопродуктивне 32-розрядний ядро [35] RISC ARM ® Cortex ® -M3, що працює на частоті 72 МГц, високошвидкісну вбудовану пам'ять (флеш-пам'ять до 128 КБ і SRAM до 20 КБ) і широкий спектр розширених входів / виходів і периферійних пристроїв, підключених до двох шинам APB. Всі пристрої мають два 12-розрядних АЦП, три 16-розрядних таймера загального призначення і один ШІМ-таймер, а також стандартні і вдосконалені інтерфейси зв'язку: до двох I 2 C і SPI, три USART, USB і CAN. На рисунку 4.1 зображено модуль на основі мікроконтролера STM32F103C8T6.

Модуль має:

- Виводи портів A0-A12, B0-B1, B3-B15, C13-C15.

- Micro-USB через який можна живити плату. На платі присутній стабілізатор напруги на 3.3В. Живлення 3.3В або 5В можна подавати на відповідні виводи на платі.

- Кнопку RESET.

- Два перемички BOOT0 і BOOT1. Використовувати під час прошивки через UART.

- Два кварци 8МГц і 32768Гц. У мікроконтролера є множник частоти, тому на кварці 8 МГц можливо досягти максимальної частоти контролера 72Мгц.

- Два світлодіоди. PWR - сигналізує про подачу живлення. PC13 - підключений до виходу C13.

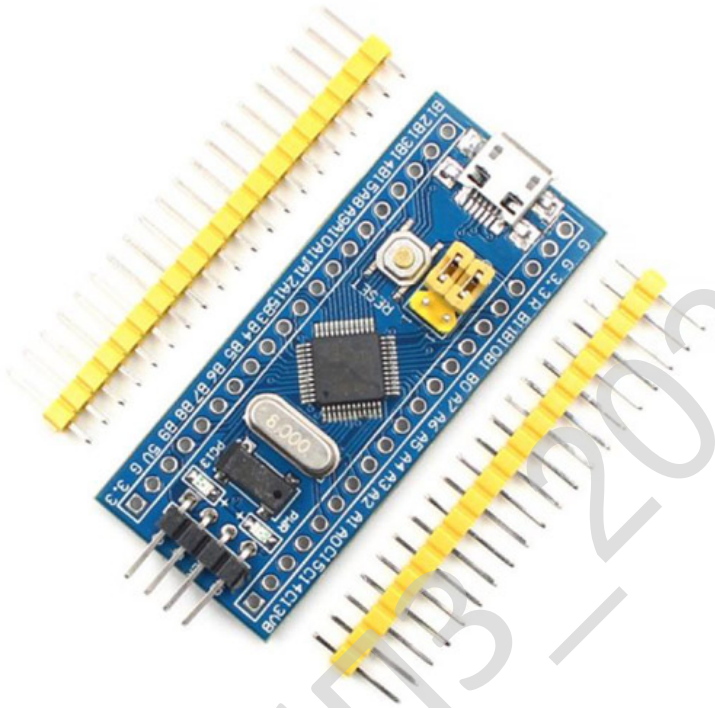


Рисунок 4.1 – Модуль на основі мікроконтролера STM32F103C8T6

Важливою частиною пристрою є радіомодуль. Вирішено використовувати радіомодуль JDY-40. Так як цей модуль має низьку ціну та відносно просте керування.

Модуль JDY-40 [36] дуже компактний і можна його живити від будь-якого літій-іонного акумулятора на 3,7 В, що дозволяє вбудувати його в будь-які моделі. Радіус дії радіомодуля до 120 метрів в прямій видимості, що для більшості пристроїв цілком достатньо, якщо потрібна велика дальність то можна застосувати зовнішню антену.

Для обміну даними з мікроконтролером модуля JDY-40 здійснюється по UART інтерфейсу з максимальною швидкістю 19200 біт. Для підключення

периферії до модулю є 8 портів введення -виведення. На рисунку 4.2 зображено модуль зв’язку JDY-40.

На основі модулів JDY-40 та макетної плати розроблено робочі макети пристроїв. На рисунку 4.3 зображено макети пристроїв для передачі даних по радіоканалу.

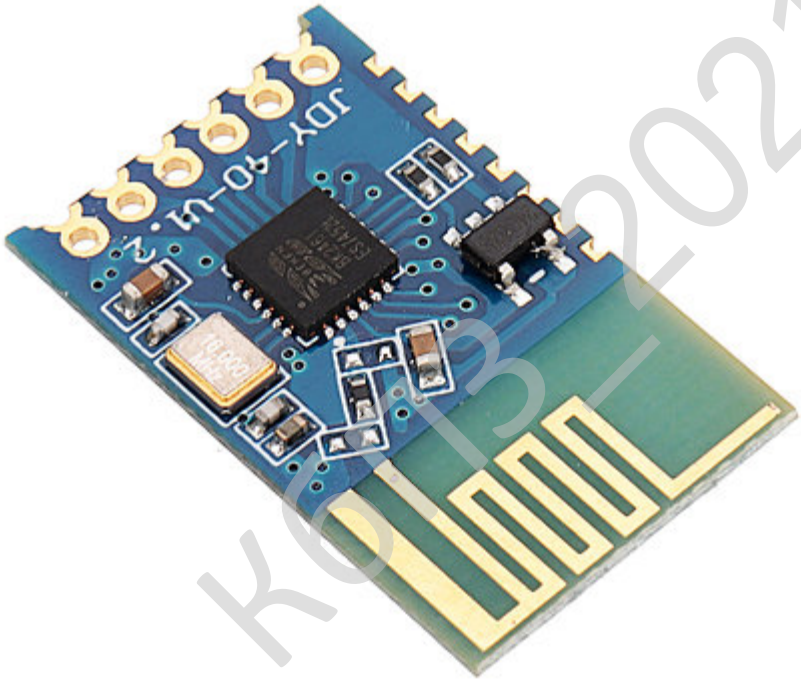


Рисунок 4.2 – Радіомодуль JDY-40

На рисунку 4.3 присутні програматори та логічний аналізатор за допомогою яких здійснювалася наладка схеми. Обидва пристрої підключені до ПК на якому відкрито два термінали. В кожному із терміналів необхідно обрати відповідний номер COM порту та відкрити його. Після відкриття COM порту можна передавати данні від ПК до мікроконтролера який в свою чергу передасть їх на радіомодуль.

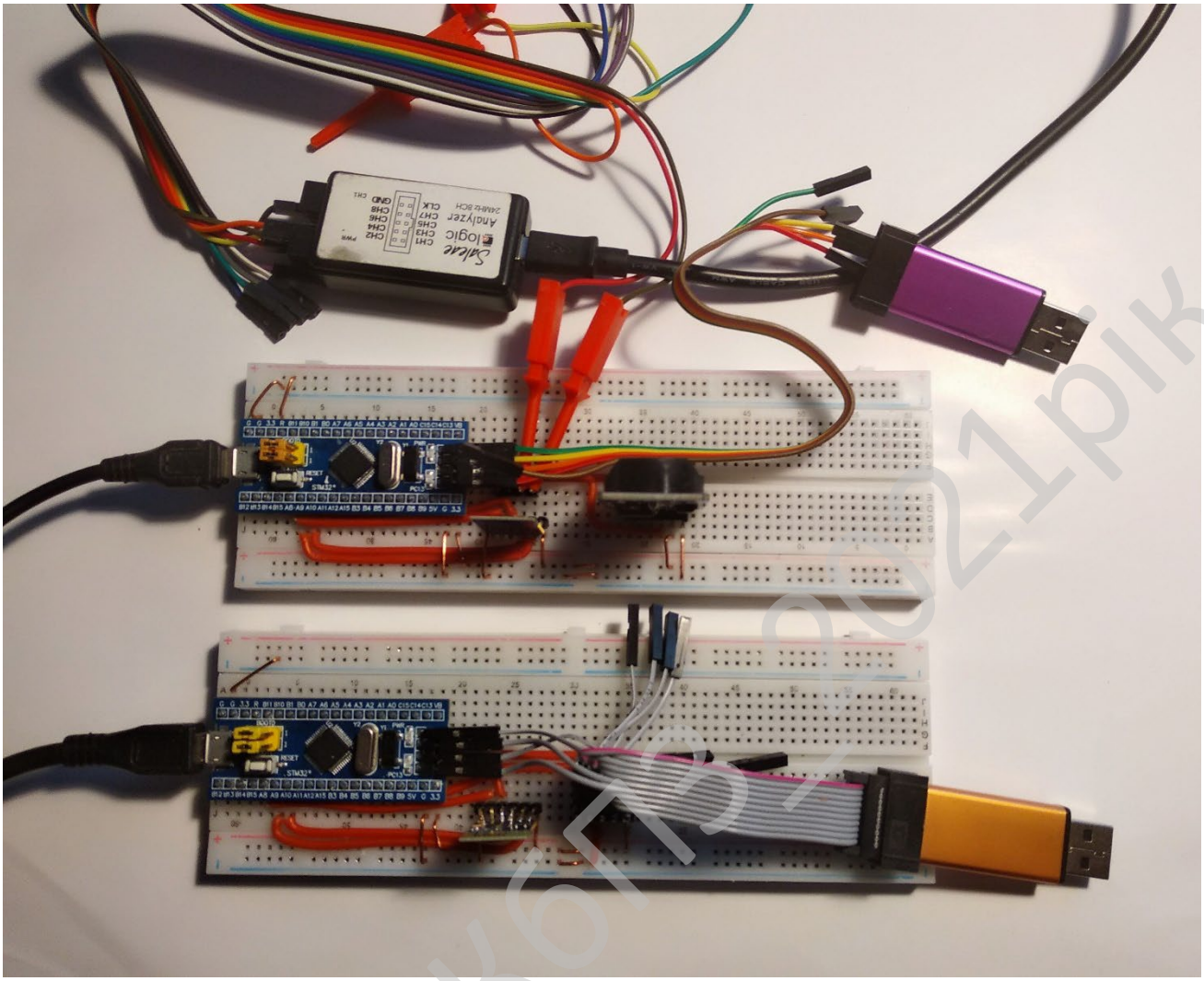


Рисунок 4.3 – Робочі макети пристроїв для передачі даних по радіоканалу

Далі буде розглянуто блок схему та опис алгоритму функціонування системи.

4.1 Блок-схеми та опис алгоритмів функціонування системи

Процес генерації випадкових чисел (ГВЧ) [37] є невід'ємною частиною багатьох криптографічних операцій..

В даний час існує велика кількість методів генерації послідовностей з різним ступенем випадковості. Однак на практиці більшість цих генераторів

виробляють послідовності, властивості яких не відповідають вимогам випадковості і є генераторами псевдовипадкових чисел(ГПВЧ).

Часто в числах, які згенеровані за допомогою таких ГПВЧ простежуються очевидні закономірності. Також, дуже часто генератори ГПВЧ є найбільш слабким місцем у системах шифрування. Справа в тому, що генератори програмного забезпечення повністю детерміновані. Вони використовують різні складні функції для обчислення псевдовипадкових чисел. Послідовності, отримані в результаті роботи таких генераторів, є передбачувані та відтворювані і не придатні для використання в криптографічних програмах.

Системи динамічного більярду виявили добре розвинену хаотична поведінку. Незважаючи на хороші характеристики, ці системи ще не застосовуються у криптографії. Головною причиною є складність вираження рівняння руху частинок в явній формі.

У 1976 році відомий математик Я. Г. Сінай довів [38], що поведінка більярдної кулі у динамічному більярді, яка визначається визначеним детермінованим рівнянням, та поведінка більярдної кулі, яке керується процесом Маркова першого порядку, нерозрізнимі. Оскільки марковський процес першого порядку є імовірносним процесом, який залежить тільки від попереднього зіткнення з перепоною, то він є як недетермінованим, так і непередбачуваним.

Програма моделювання руху кулі у більярді Сіная [39] базується на твердженнях, що рух кулі здійснюється без втрати швидкості(тертя відсутнє) та кут падіння дорівнює куту відбиття. Таким чином приймаємо, що рух кулі має швидкості по координатах Vx=cos(ά); Vy=sin(ά). Звідси Vx2 +Vy2=1. Приймаємо для побудови алгоритму руху кулі наступні вхідні параметри:

- напрямок руху кулі —V={Vx,Vy},

- початкове положення кулі у більярді — P={Px, Py}.

Алгоритм моделювання більярду Сіная наступний:

Програма моделювання руху кулі у більярді Сіная базується на твердженнях, що рух кулі здійснюється без втрати швидкості(тертя відсутнє) та

кут падіння дорівнює куту відбиття. Таким чином приймаємо, що рух кулі має швидкості по координатах Vx=cos(ά); Vy=sin(ά). Звідси Vx2 +Vy2=1. Приймаємо для побудови алгоритму руху кулі наступні вхідні параметри:

- напрямок руху кулі —V={Vx,Vy},

- початкове положення кулі у більярді — P={Px, Py}.

Для розробки алгоритму побудуємо математичну модель руху математичної точки в полі опуклого МБС.

Точка частинки рухається по більярду постійна швидкість v. Коли вона досягає кордону, зазнає пружного зіткнення з дзеркальним відображенням відповідно до закон відбиття, кут падіння дорівнює кут відображення відносно -! n нормального вектора в межі зіткнення кордону. Між двома зіткненнями частинка йде прямим шляхом. Спочатку частинка орієнтована під кутом $\theta_0 = (\overrightarrow{i}, \overrightarrow{v_0})$, де $\vec{i}$ одиничний вектор осі x.

З цього маємо $\overrightarrow{v_0} = \cos(\theta_0)\,\vec{i} + \sin(\theta_0)\vec{j}$

Після зіткнення маємо рівняння 4.1:

$$\left(\vec{i}, \overrightarrow{v_{new}}\right) = \left(\vec{i}, \overrightarrow{v_{old}}\right) + \left(\overrightarrow{v_{old}}, \vec{N}\right) + \left(\vec{N}, \overrightarrow{v_{old}}\right) \; mod\; 2\pi \qquad (4.1)$$

Після правила зіткнення маємо:

$$\left(\vec{N}, \overrightarrow{v_{new}}\right) = \left(-\overrightarrow{v_{old}}, \vec{N}\right)$$

Також є:

$$\left(-\overrightarrow{v_{old}}, \vec{N}\right) = \left(-\overrightarrow{v_{old}}, \overrightarrow{v_{old}}\right) + \left(\overrightarrow{v_{old}}, \vec{N}\right) \; mod\; 2\pi = \pi + \left(\overrightarrow{v_{old}}, \vec{N}\right) \; mod\; 2\pi$$

Тому рівняння 4.1 стає:

$$\left(\vec{i}, \overrightarrow{v_{new}}\right) = \left(\vec{i}, \overrightarrow{v_{old}}\right) + 2\left(\overrightarrow{v_{old}}, \vec{N}\right) + \pi \; mod\; 2\pi \qquad (4.2)$$

При $(n+1)$ зіткненні, де $n \geq 0$, ставимо $\theta_0 = \left(\vec{i}, \overrightarrow{v_{new}}\right)$ і отримуємо

$\theta_{n+1} = \left(\vec{i}, \overrightarrow{v_{new}}\right)$ і тому рівняння 2.2 стає:

$$\theta_{n+1} = \theta_n + 2\left(\overrightarrow{v_n}, \overrightarrow{N_{n+1}}\right) + \pi \; mod\; 2\pi \qquad (4.3)$$

$$\overrightarrow{v_n} = \cos(\theta_0)\,\vec{i} + \sin(\theta_0)\vec{j}$$

де $\overrightarrow{N_{n+1}}$ одиничний нормальний вектор на кордоні до (n + 1) зіткнення. Після геометричної форми маємо більярд:

$$\overrightarrow{N_{n+1}} = \begin{cases} -\frac{x_{n+1}}{|x_{n+1}|} & \text{Якщо } |x_{n+1}| = a \\ -\frac{y_{n+1}}{|y_{n+1}|} & \text{Якщо } |y_{n+1}| = a \\ -\frac{x_{n+1}\vec{i}+y_{n+1}\vec{j}}{\sqrt{x_{n+1}^2+Y_{n+1}^2}} & \text{Інакше} \end{cases} \quad (4.4)$$

Вважаємо $A_{n+1}(x_{n+1}, y_{n+1})$ точкою $(n+1)$ зіткнення, тому A_{n+1} належить до перетину траєкторія частинок з більярдною межею Г. Визначимо f, функцію переходу від (A_n, θ_n) до (A_{n+1}, θ_{n+1}), таких як:

$$(x_{n+1}, y_{n+1}, \theta_{n+1}) = f(x_n, y_n, \theta_n)$$

Генератор заснований на системі хаотичного більярду, тому створюються послідовності які успадковують хаос і непередбачуваність більярду. Ряд додаткових ітерацій витягується безпосередньо з пароля дозволяє генератору отримати вигоду у вигляді максимуму хаосу, який пропонує більярд.

Кути ініціалізації беруться використовуючи покажчик, який вказує на різні позиції до тих пір, поки є його загальне покриття. Дійсно, різниця між двома клавішами може спричинити різну орієнтацію на частинок і, таким чином, до утворених послідовностей.

Чутливість до невеликої зміни ключа є однією з основних властивостей ГПВЧ. Іншими словами, мала різниця в системі повинна викликати велику зміну псевдовипадкових послідовностей. Ця властивість робить генератор високо захищеним від статистичних та диференціальних атак, тому послідовність не може бути зламаною, навіть якщо між ними є невелика різниця ключа. Генератор заснований на двох динамічних системи суто хаотичного більярду

За побудованою математичною моделлю розроблено алгоритм

Алгоритм моделювання більярду Сіная наступний:

```
Begin
     Px := Value_Px; Py:=Value_Py;
     //Конкретні початкові координати початку руху кулі
     Vx := Value_Vx; Vy := Valye_Vy;
```

```
//Конкретний напрямок руху відповідно по координатах x та y
N := Value_N;
//Конкретна кількість ітерацій циклів обчислення точок перетину
for i:=1 to N do//Кількість зіткнень кулі
      x0  [0..3]:=[1,  -1,  -1,    1];//Коефіцієнти для векторів, які
      враховують квадрант знаходження
      y0[0..3]:= [1, 1, -1, -1];// кулі у площині більярду Сіная
      for  i:=0  to  3  do//Цикл знаходження координат точок перетину
      траєкторії руху кулі
            Cx := Px - x0[i]; Cy :=Py - y0[i];
            D := (Vx * Cx + Vy * Cy)2 — (Cx 2 + Cy2 - 1);
            if (D > 0) then
                  Begin
                        t0 := - (Vx  * Cx + Vy * Cy) - sqrt(D);
                        t1 := - (Vx  * Cx + Vy * Cy) + sqrt(D);
                        if (t0  > 0) and (t1  > 0) then
                              Begin
                                    t := min(t0 , t1);
                                    x0 := Vx * t + Px;
                                    y0 := Vy * t + Py;
                                    rx := x0  - x0[i];
                                     ry := y0 - y0[i];
                                    Vx* :=Vx * (ry2 - rx2) - 2 * Vy *
                                    rx *  ry;
                                    Vy := Vy (rx2 - ry2) — 2 * Vx * rx
                                    * ry;
                                    Px* := x0; Py :=y0;
                                    Vx := Vx* Vy:=Vy*;
                              End;
                  End;
      End;
End.
```

Будуємо блок схему алгоритму математичної моделі. На рисунку 4.4 зображено блок схему алгоритма математичної моделі більярда Сіная.

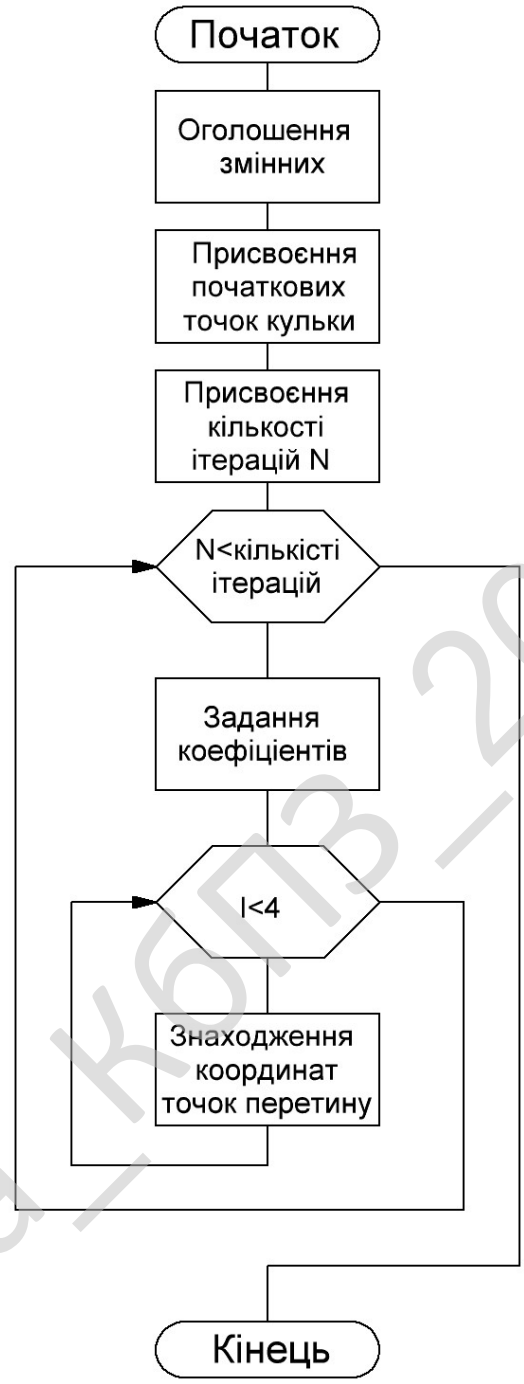


Рисунок 4.4 – Блок схема алгоритму програми

Реалізація математичної моделі виконується мовою Pyton в середовищі Visual Studio Communnity 2017 [40] з математичною бібліотекою Matplotlib [41]. (додаток А). На основі попередньої математичної моделі розроблено ПЗ для робочої моделі пристрою. На рисунку 4.5 зображено вивід графічної інформації

руху математичної точки в МБС при двох ітераціях.

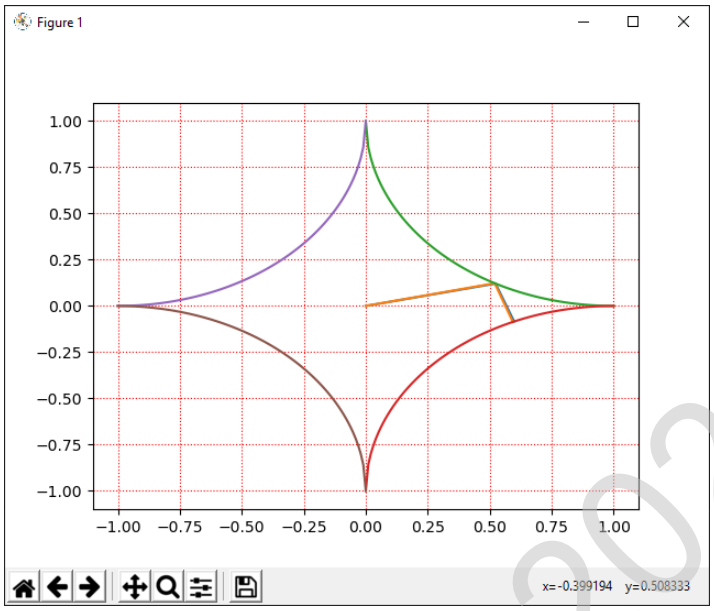


Рисунок 4.5 – Вивід графічної інформації руху математичної точки в МБС при двох ітераціях

З рисунку 4.5 видно, що математичні частинки починають свій рух від центру поля в бік початкового кута який задається на початку розрахунків. Далі частинка в точці перетину відбивається закону кутів та рухається до іншої точки перетину. Саме в цих точках перетину ї є число яке генерується як випадкове. Для кращої розуміння розглянем рисунок 4.6. На рисунку 4. 6 зображено вивід графічної інформації руху математичної точки в МБС при 12-ти ітераціях.

Згідно з малюнків 4.5 та 4.6 видно, що на полі рухаються дві математичні частинки. Це зроблено для порівняння їх розбіжності при невеликій різниці початкового кута. Початковий кут математичних частинок відрізняється всього лиш на 0,3 градуси. При цьому видно що така незначна різниця дає дуже велику розбіжність. Така висока розбіжність пояснюється високою хаотичністю даного математичного більярда. Тому при спробі проаналізувати трафік інформації з метою розшифровки не вдаватиметеся перехоплення. Також при використанні даного методу через канал зв’язку не передаються ніякої інформації про ключі.

Вони генеруються на кожному із пристроїв самостійно.

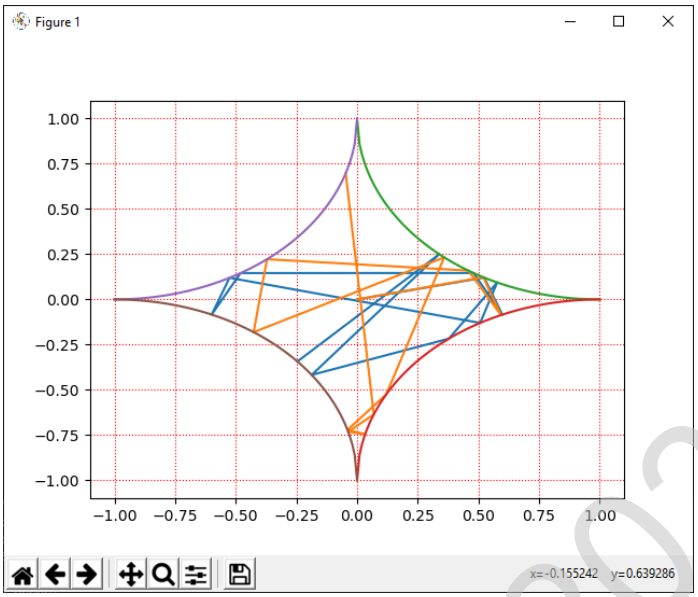


Рисунок 4.6 - Вивід графічної інформації руху математичної точки в МБС при 12-ти ітераціях

На рисунку 4.7 зображено вивід графічної інформації руху математичної точки в МБС при 200-та ітераціях

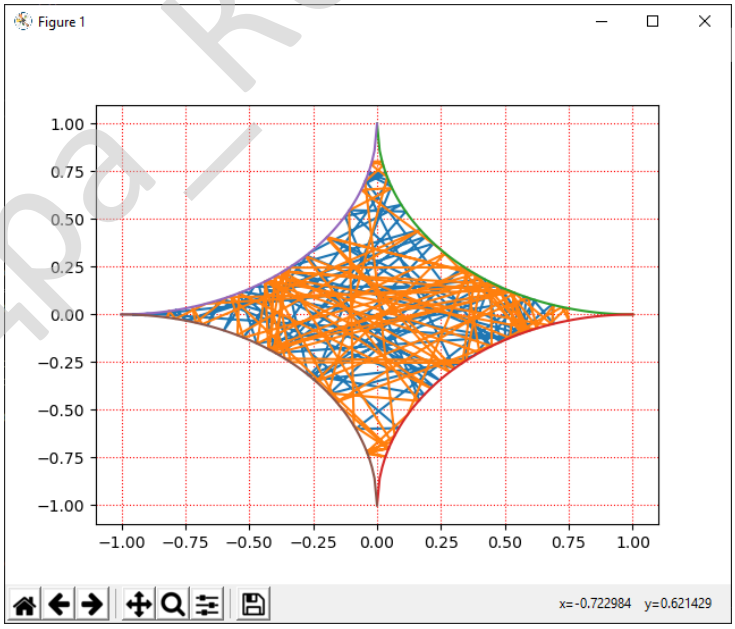


Рисунок 4.7 - Вивід графічної інформації руху математичної точки в МБС при 200-та ітераціях

Всі програми для мікроконтролерів починаються з ініціалізації. Спочатку треба ініціалізувати тактовий генератор мікроконтролера з вказаними джерелами тактового сигналу, множенням тактового сигналу для ядра та діленням для периферії та системних шин. Після цього ініціалізується потрібна для роботи периферія мікроконтролера. А саме:

- GPIO – порти вводу виводу.

- UART – послідовний асинхронний порт.

- USB – універсальна системна шина.

Зазвичай першими ініціалізують порти вводу виводу (GPIO) [42]. GPIO - general-purpose input output pin або ж порт вводу виводу. Порт вводу виводу - це контакт загального призначення. Може працювати як на вхід так і на вихід. Тобто має можливість або читати логічний стан контакта або навпаки задавати логічний рівень (рівень напруги). Також деякі групи контактів мають можливість працювати в аналоговому режимі при вимірах аналого-цифрового перетворювача (АЦП). Потім треба послідовну шину UART.

UART – це універсальна асинхронна послідовна шина [43]. Шина UART дуже гнучка та дозволяє підключати багато різних пристроїв (мікросхем модулів). Сама шина є повністю дуплексною. Тобто дозволяд передавати данні в обидві сторони як від ведучого пристрою так і від веденого пристрою.

Ініціалізація UART починається з тактування. Після ініціалізації тактування UART необхідно налаштувати контролер NVIC який відповідає за події та вектори переривання.

NVIC (Nested vectored interrupt controller) - модуль контролю переривань [44]. Він виконує наступні функції:

- Дозволяє або забороняє переривання.

- Назначає пріоритет переривань (від 0 до 15. 0 - максимальній пріоритет, 15 - мінімальний пріоритет).

- Автоматично зберігає дані при виконанні одиноких чи вкладених переривань.

Вмикаємо переривання UART. Блок схема функції ініціалізації UART зображена на рисунку 4.8

Рисунок 4.8 - Блок схема функції ініціалізації UART

Після ініціалізації UART необхідно ініціалізувати модуль USB.

0Універсальна послідовна шина (USB) - це найуспішніший шина в історії ПК [45], які використовуються для підключення таких пристроїв, як миша, ігрові приставки та джойстики, сканери, цифрові камери та принтери. USB також перейшла у побутові пристрої. Через її поширеність та зручність її і було обрано для виконання проекту. На рисунку 4.9 зображено блок схему функції ініціалізації USB.

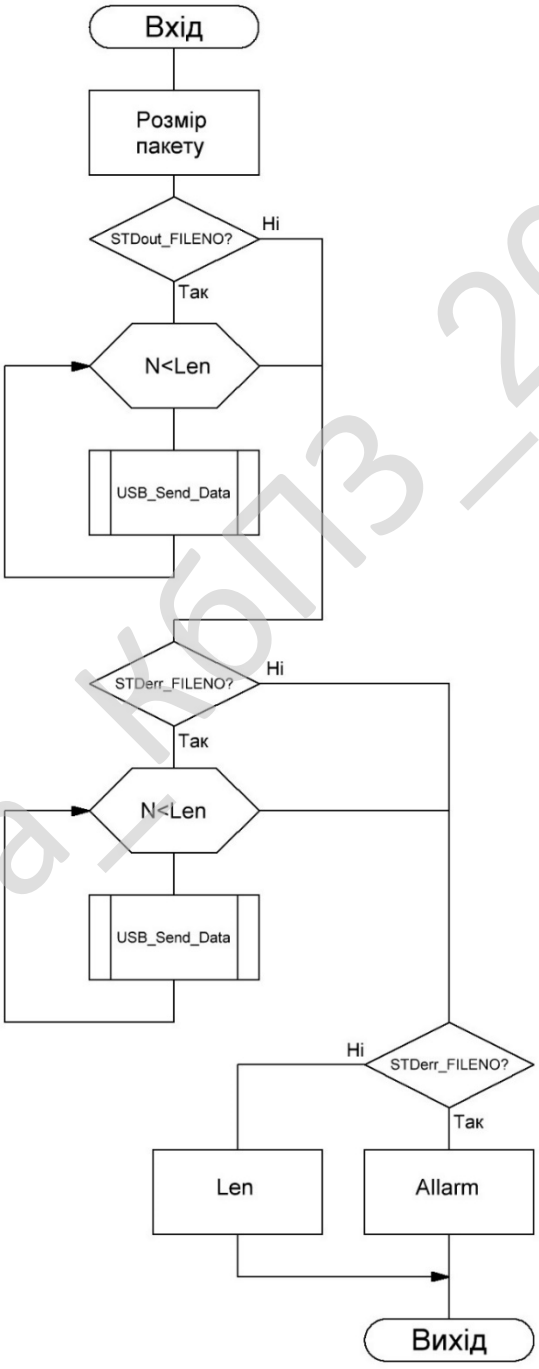


Рисунок 4.9 - Блок схема функції ініціалізації USB

Далі у основній програмі встановлюємо сигнали для ввімкнення радіомодуля у робочий режим. Встановлення сигналу для радіомодуля для дозволу передачі даних. Сигнал встановлюється на контактові GPIO який найзручніше використати. Тобто той контакт який є найближчим до модуля. Також перед встановленням сигналу команда дані (контакт SET) або ж просто дозволу передачі даних необхідно включити сам радіомодуть подачею логічного нуля на ніжку CS. При наявності логічної одиниці на контакті CS радіомодуль знаходиться в сплячому режимі. Це дає змогу керувати енергоспоживанням пристрою. Але першочергове призначення даного контакту – це керуванням декількома пристроями на одній шині. Тому як шина призначена для передачі даних між двома пристроями (ведучим та веденим) в один момент часу. Радіомодуль за допомогою контакту SET дозволяє налаштувати швидкість передачі даних та його адресу за допомогою якої декілька радіомодулів виявляють, що данні йдуть саме до нього. Таким чином можливо налаштувати модулі так, щоб вони не конфліктували між собою. Або ж навпаки відправляти дані з одного модуля на декілька інших. В даному випадку налаштування швидкості та адрес не важливе та залишене за замовченням. Після цього йде головний цикл програми. В головному циклі програми створено перевірку на наявність даних в буфері USB. Якщо буфер не пустий то зчитуємо його розмір. Розмір буфера даних які прийшли з шини USB не фіксованого розміру. В такому випадку необхідно визначити фіксовану величину інформації яка буде передаватися на радіомодуль. Це потрібно для запуску ГПВЧ з конкретною кількістю ітерацій (псевдовипадкових чисел). Якщо буфер перевищує розмір кадру його необхідно відправити по частинам. Після запуску ГПВЧ та розділенні даних з буфера необхідно виконати операцію виключного АБО (XOR) [46] між вхідною інформацією та псевдовипадковим числом. Після цього присвоюємо дані буферу UART.

Далі необхідно виконати обнулення буферів. На рисунку 4.8 зображено блок схему алгоритму програми.

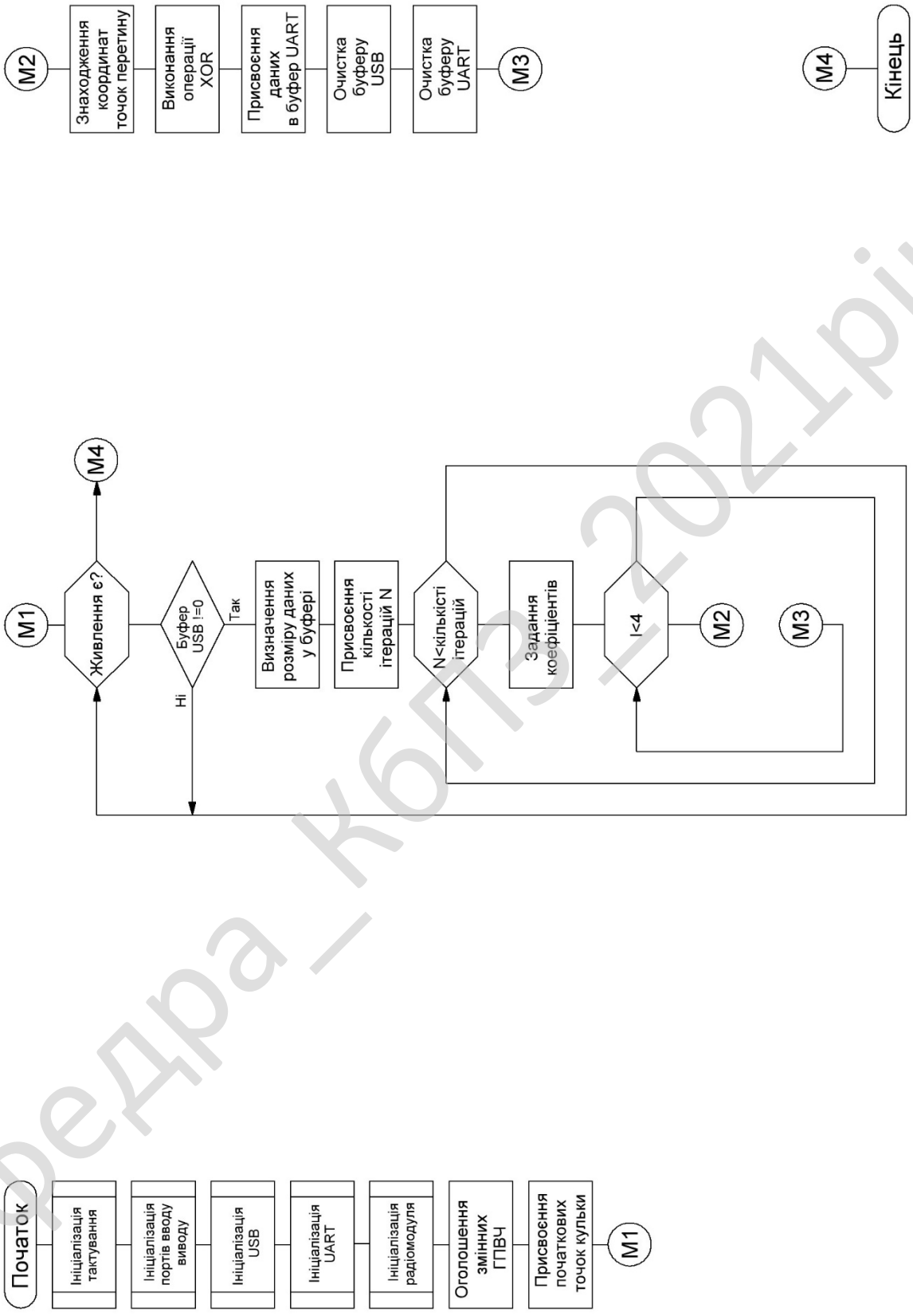


Рисунок 4.9 – Блок схема алгоритму програми

4.2 Захист розробленого програмного забезпечення

Сучасні розробники пристроїв як правило при розробці електронних приладів використовують мікроконтролери (МК). Сучасні МК стали потужні та доступні по ціні. Тому в більшості випадків використання звичайніх процесорів недоцільне. Але питання захисту програмного забезпечення мікроконтролера стоїть дуже гостро та передбачає спеціальні заходи по запобіганню несанкціонованого копіювання керуючої програми МК.

На рівні захисту інформації всі мікроконтролери можна поділити на дві групи – звичайні та захищенні. Захищенні мікроконтролери мають різні рівні доступу даних та захисту на рівні чіпа з використанням шифрування інформації [1]. Для подолання захисту таких мікроконтролерів треба використати складне та дорого вартісне обладнання, що часто стає економічно не доцільно.

На даний час у користувачів техніки популярний - Internet ofhings. Розумні машини, розетки, лампочки, телевізори, ваги, світильники. Раніше виробники роутерів та IP-камер стикалися з проблеми які тепер стають актуальними для майже будь-яких пристроїв.

Перед усіма розробниками електронної техніки дуже актуальним стає захист прав на інтелектуальну власність. Як найскладніша, так і найпростіші програми повинні мати однаковий рівень захищеності як інтелектуальна власність розробника. Популярні МК STM8, STM32, AVR та платформа для навчання Arduino використовується для створення різноманітних пристроїв автоматики та робототехніки. Тому необхідно розглядати методи захисту інтелектуальної власності розробника на прикладі модифікації програм для МК

Плагіат вихідного коду став серйозною загрозою для розвитку дрібномасштабних виробництв, а також порушення права інтелектуальної власності є загрозою для розвитку апаратної системи. Існує багато програмних рішень для порівняння вихідних кодів, але вони часто не реалістичні в цьому сценарії. Схема цифрового водяного маркування - одне з можливих рішень цієї

проблеми. Використовується нова техніка водяного маркування, щоб її можна було легко та надійно виявити спеціальними методами. Усі підходи використовують інформацію бічного каналу, отриману під час виконання підозрілого коду. Основний метод є пасивним, тобто попередня модифікація вихідного коду не потрібна. Він визначає, що використовуються ваги Хеммінга [47] у виконаних інструкціях підозрілого пристрою та використовує алгоритми відповідності рядків для порівнянь з реалізацією посилань. Інший метод вставляє додаткові фрагменти коду у вигляді водяного знака, який можна ідентифікувати при споживанні енергії виконаного вихідного коду. Запропоновані підходи є надійними щодо атак трансформації коду.

Програмне забезпечення плагіату та піратства є серйозною проблемою, яка, за оцінками, коштує невеликій промисловості мільярдами доларів на рік. Піратство програмного забезпечення - серйозна проблема, яка привертала увагу в минулому. Однак компанії, що працюють із вбудованими системами, також стикаються з серйозними проблемами щодо плагіату програмного забезпечення та програмного піратства. Якщо дизайнер підозрює, що його код використовується у вбудованому пристрої, важко визначити, належить код користувачеві або дизайнеру. Конструктор повинен перевірити код у пристрої, що підозрюється, і порівняти його з оригінальним кодом, щоб виявити плагіат. Однак механізм захисту пам’яті програм запобігає несанкціонованому доступу до пам’яті програми, що використовується в сучасних мікроконтролерах. Тож механізм захисту доводиться перемагати спочатку для перевірки програмного плагіату. Це робить тестування вбудованих пристроїв на програмний плагіат дуже складним, особливо якщо це потрібно робити автоматизовано. Серйозна загроза в цьому плані полягає в тому, що власникиінтелектуальної власності з незаконними копіями можуть призвести до величезних грошових втрат. Це не тільки впливає на виробників та споживачів оригінального обладнання, але й призводить до економічної шкоди. Вбудовані системи - це сучасні системи, які відіграють життєво важливу роль у розвитку багатьох товарів народного споживання. Якщо

не передбачено профілактичний технологічний захист, складні технології дозволяють здійснювати напади на обладнання та програмне забезпечення у вбудованих системах. Атаки варіюються від цільової модифікації до повного зворотного проектування та піратства продуктів. У запропонованій системі досліджуються способи нападів, а також заходи захисту, які вважаються для боротьби з піратством продукції технологічними засобами. Запропоновано нову техніку водяного маркування, яка використовує бічні канали як будівельний блок, який можна легко та надійно виявити методами, адаптованими з аналізу бічних каналів. Апаратним заходом є вбудовування унікального сигналу в бічний канал пристрою, який служить водяним знаком. Це змушує дизайнерів перевіряти водяні знаки в інтегрованій мікросхемі. Запропоновані водяні знаки реалізовані програмовано для витоку цифрового підпису. Запропонована методика вимагає переписування вихідного коду та додавання шістнадцяткових кодів у мовних перетвореннях. Система програмного забезпечення в реальному часі використовує різноманітні методи захисту коду. Більшість із цих методів вимагають певних модифікацій у вихідному коді та можуть бути неефективними для того, щоб мати різноманітні атаки. Виявлення обладнання є одним із ефективних методів порівняно з вищезгаданим програмним методом. Але все ж апаратні методи також мають недоліки, такі як дизайнеру потрібно модифікувати апаратне забезпечення процесора, що, як правило, непросто. Методи обфускування перетворюють програму з її початкової форми в нову форму, яку легко зрозуміти без зміни функції програми. Тут мета полягає в тому, щоб збити з пантелику зловмисника, і це збільшить труднощі зворотної інженерії. Недоліком методів обфускування є відсутність теоретичної основи, тому вона не може оцінити ефективність заходів кількісно.

Програмування знаку плагресиму в даний час пов'язане з рядками ваги Хеммінга. План можна скоротити трьома кроковими інструментами: По-перше, потік виконання першого програмування відображається на рядку ваг Хеммінга. Це визнається або шляхом вимірювання та оцінки використання енергії

виконання, як було виражено вище, або шляхом повторного введення в дію потоку виконання та фігурування ваг Хеммінга. Другий крок - записати потужний натяк на підозрілий гаджет, щоб додатково отримати рядок ваг Хеммінга, позначених як "t". Узгоджений дублікат рядка буде розпізнаний на третьому кроці, поглянувши на два рядки s і t. Збалансований дублікат є захопленим припущенням щодо порушення програмних прав на авторські права, а згодом їх легко відрізнити. Перевірка популяції бітового рядка часто потрібна в криптографії та інших програмах. DPA заснований на алгоритмі та його потужності, що споживається процесором. Щоб виявити водяний знак, слід перевірити значення потужності для різних значень. Комбінуючи різні функції з вхідними значеннями та клавішею з водяними знаками, користувач може перевірити та обчислити правильне припущення. Розрахунок водяного знака та його енергетичного сліду базується на тактовому циклі. Сліди повинні бути належним чином синхронізовані, щоб уникнути небажаного шуму при аналізі бічних каналів.

Перетворення символів для шифрування є несумісним. Шифрування вимірюється блоком 16 С / 128 біт. Персонажі піддаються транспозиції, а метод заміщення використовується багатовимірним масивом. Для блоку, який створює перехідний результат відповідної довжини, буде створено одноразовий під ключ. Ітерації шифрування виконуються за допомогою комбінації попереднього текстового блоку та простого тексту з 8 бітами С / 64, що дає 192 біти, і це буде теперішній блок тексту, який створює новий 24-символьний текстовий блок. Ця ж методика застосовується, поки вона не дає блоку 256 біт, і цей біт буде виконуватися операція XOR з попередніми 256 бітами, відмінними від першого блоку, якщо в простому тексті більше 32 біт. Ті ж 256 біт над якими виконувалася операція XOR з останніми 256 бітами, крім першого блоку, якщо є більше 32 символів. В обох методах, якщо зловмисник знає внутрішні комбінації стану, легко розкрити водяний знак та використовувані інструкції. Система архітектури пропонується з двома різними рівнями аутентифікації. Перший рівень визначає зміни в мові збірки з водяними знаками та забезпечує надійність. Другий рівень

визначає важливість споживання електроенергії в коді, що впливає, для підтвердження права власності. На малюнку 4.5 показано основний рівень захисту для виявлення крадіжок програмного забезпечення з незначними змінами коду рівня складання. Водяний знак програмного забезпечення бічного каналу складається з кількох інструкцій, які вставляються в код рівня складання. Ці інструкції передбачають зміни в машинному циклі та енергоспоживанні процесора, а розкрадання може бути виявлено надійним перевіряючим методами, дуже схожими на класичні атаки бічних каналів. На рисунку 4.5 зображено код архітектури системи - водяний знак.[48]

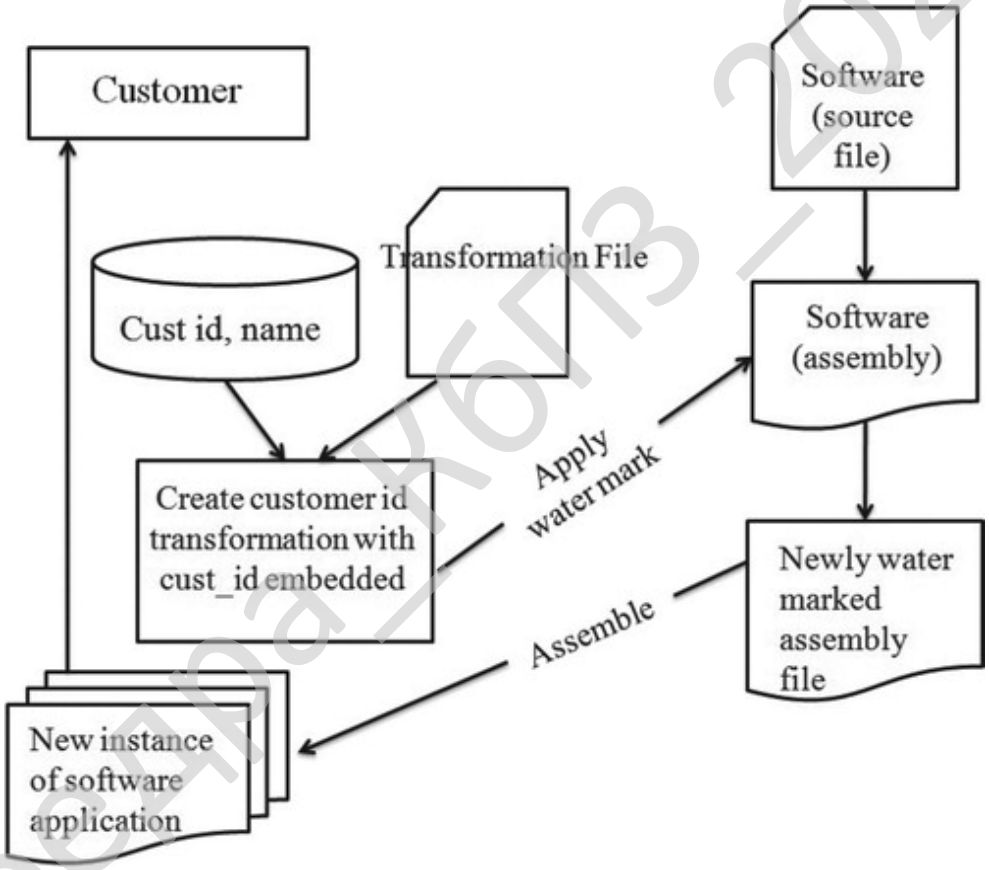


Рисунок 4.5 – Зображення коду архітектури системи - водяний знак

Від виробника мікроконтролерів STM32 передбачений захист від зчитування програми з пам’яті МК. Даний захист працює наступним чином. А саме блокую доступ до пам’яті програм вбудованому завантажувачу та при спробі

читання програми з flash пам’яті МК стирає всю програму яка була в ньому записана до читання та встановлення біту захисту. Даний біт захисту можна встановити як програмно так і за допомогою стандартної утиліти від ST.

Розглянемо встановлення захисту від читання за допомогою стандартної утиліти для завантаження програм у пам’ять МК. На рисунку 4.6 зображено вікно утиліти з вказаним меню Option Bytes в якому знаходиться конфігурація пам’яті та біт захисту.

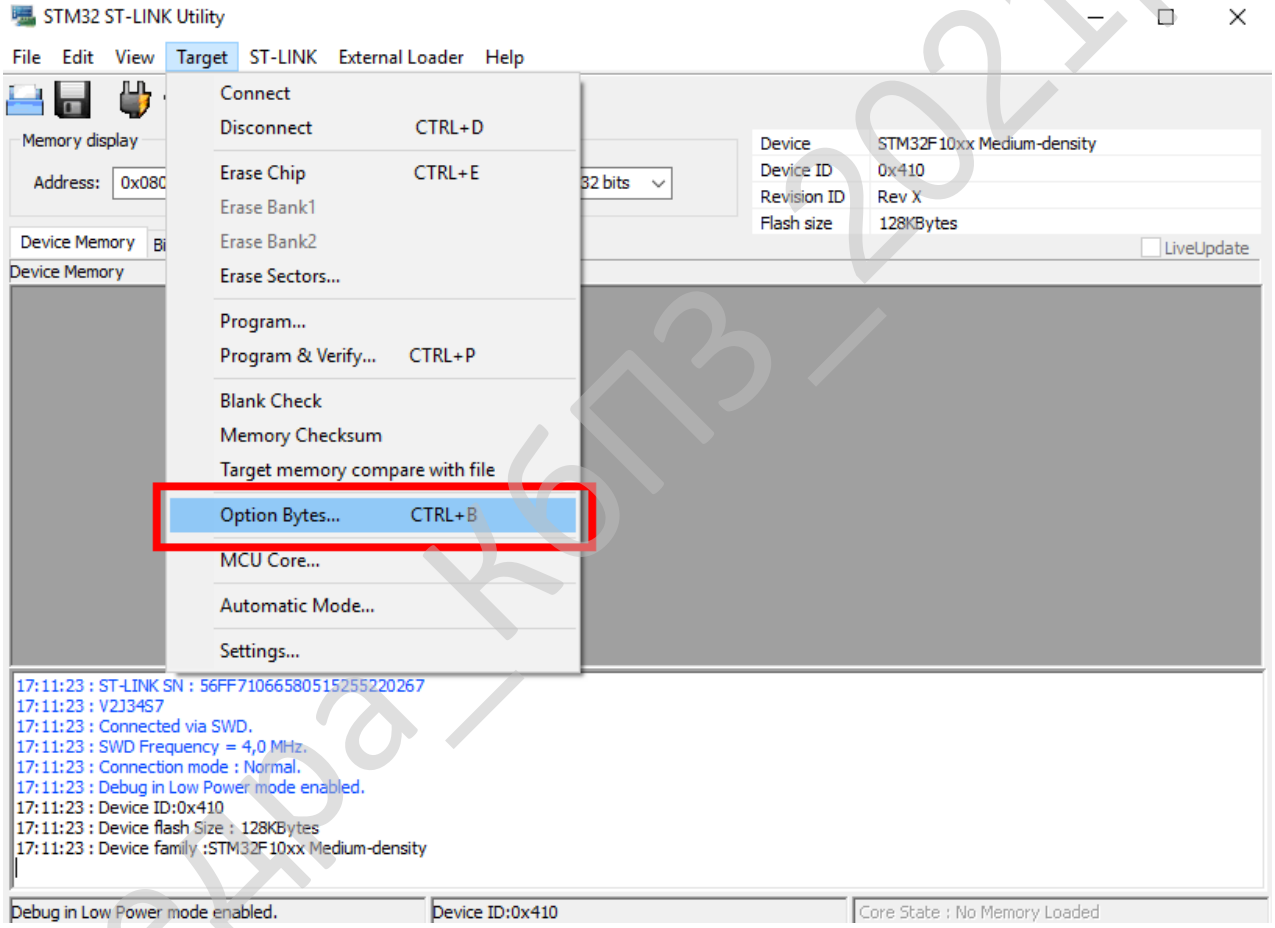


Рисунок 4.6 – Вікно утиліти ST-Link

Вікно Option Bytes доступне тільки при підключенні програматора з під’єднаним до нього мікроконтролером. На рисунку 4.6 внизу вікна видно текстове поле з виведеною інформацією про програматор, мікроконтролер та інтерфейс через який ведеться завантаження та читання програми. В даному

випадку це SWD. SWD [49] потребує підключення всього лиш двох сигнальних дротів та двох дротів живлення. Інтерфейс SWD ще зручний тим що дозволяє проводити внутрішньо схемне відлагодження програми пристрою. На рисунку 4.6 зображено вікно Option Bytes.

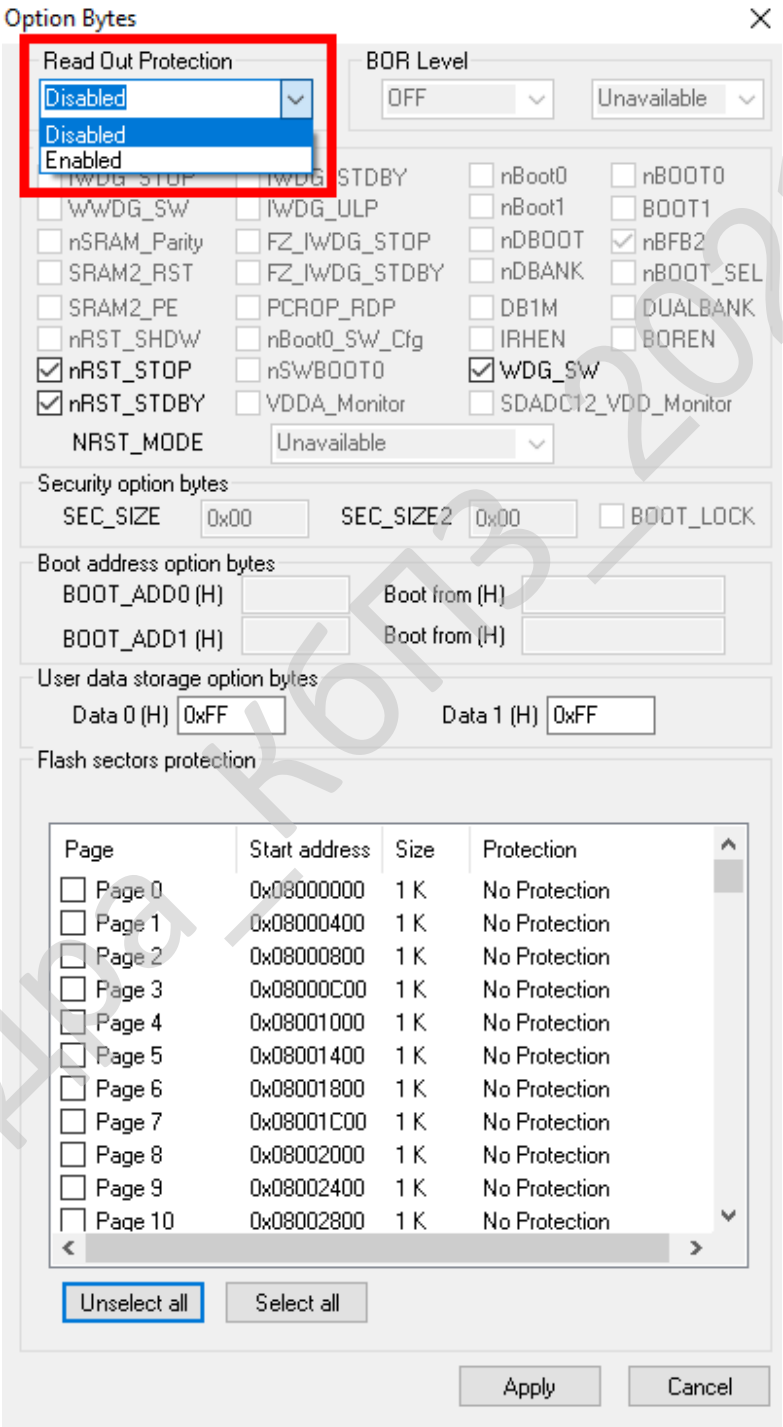


Рисунок 4.6 – Вікно Option Bytes

У вікні Option Bytes внизу вікна в таблиці необхідно встановити яку із сторінок пам’яті необхідно захищати від зчитування. Є можливість вибору одразу всіх сторінок за допомогою кнопки Select all. Потім у випадаючом у списку необхідно встановити пункт Enable. Далі натиснути кнопку Apply та виконати завантаження програми.

Розглянемо варіант захисту від читання за допомогою конфігурації біту захисту з програми користувача [50]. В наступному коді зазначені інструкції та бібліотека необхідні для захисту програми.

```
#include "main.h"
#include "stm32f10x_flash.h" // Хедер драйвер для флеша

#define NDEBUG // Макрос для дозволу чи заборони відлагодження

int main(void)
{
    #ifdef NDEBUG
        #warning "FLASH Read OUT Protection ON. DEBUG is OFF."
// Перевіряємо встановлений чи не встановлений захист
        if (FLASH_GetReadOutProtectionStatus() == RESET)
        {  // Якщо захисту немає
           FLASH_Unlock(); // Дозволяємо запис у пам’ять
           FLASH_ReadOutProtection(ENABLE);
// Захищаємо від читання програми
            FLASH_Lock(); // Забороняємо запис у пам’ять
        }
    #endif

    init(); // Ініціалізація

    while (1)
    {
        // Головний цикл програми
    }
}
```

Отже вибраний мікроконтролер гарно підходить для роботи в пристроях шифрованої передачі даних так як має можливість захисту програми від читання.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

При використанні промислового зразка розробленої системи потрібно мати комп'ютерний пристрій який може передавати цифровий сигнал.

Пристрій повинен бути оснащений засобами Bluetooth, Wi-Fi або радіомодулем. Апаратно програмний модуль як група, що складається з двох об'єктів які пов'язаних між собою або загальним пін кодом або вписаними у програмний код однаковими початковими даними:

– Початкова точка відліку координати по x, y та

– Напрямок куту напрямку руху.

Пристрій повинен бути оснащений модулем RTC який здійснює синхронізацію прийому та передачі пакетів.

Для захисту прошивки від зчитування кул-хакерами при завантаженні керуючої програми необхідно встановити біт захисту від зчитування. В разі спроби читання програми мікроконтролер очистить всю пам'ять програм. Після цього пристрій не буде працювати.

Перевірка здійснення шифрування та дешефрування виконується за допомогою зовнішнього ПК через який послідовно за таймером передаються повідомлення однакового змісту від пристрою до пристрою. Після чого повідомлення перевіряються на помилки.

Впровадження у виробництво можливе у декількох варіантах:

- Продаж запрограмованого мікроконтролера з встановленим бітом захисту від зчитування керуючої програми.

- У вигляді невеликого модуля який виступає UART – UART мостом.

- У вигляді модуля який має в собі радіомодуль з підключенням до схеми користувача по UART або USB інтерфейсу.

- У вигляді модуля з аналоговим входом та аналоговим виходом.

Розглянемо функціональну схему варіанта використання у вигляді запрограмованого мікроконтролера з захищеною пам’яттю програми. На рисунку 5.1 зображено функціональну схему МК який виступає мостом між різними інтерфейсами.

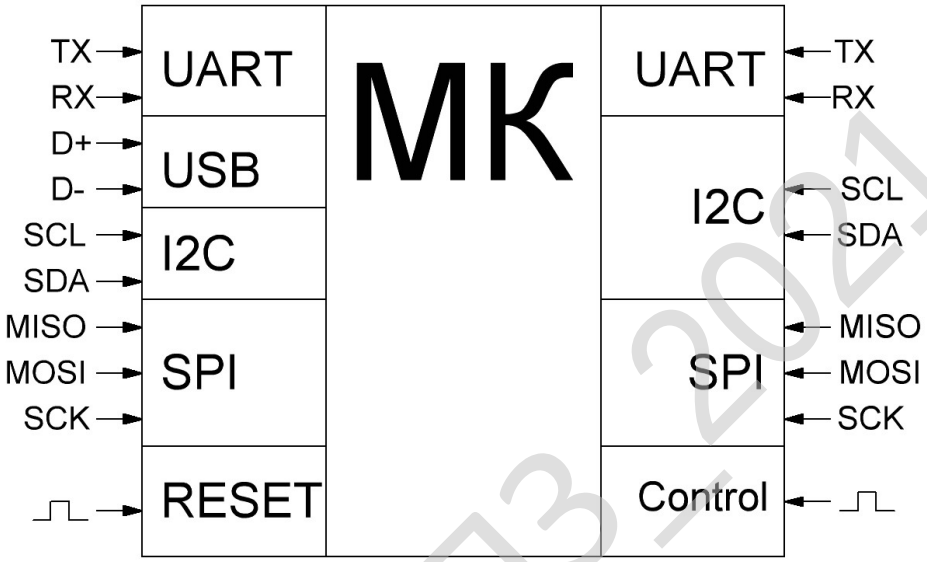


Рисунок 5.1 – Функціональна схема моста на основі МК

Такий міст повинен шифрувати та розшифровувати трафік даних який проходить через нього. З рисунку 5.1 видно що МК має декілька вхідних та декілька вихідних інтерфейсів передачі даних. Це дозволяє з легкістю адаптувати МК з алгоритмом шифрування у будь яку систему передачі даних. Також на схемі вказано сигнал контроля за допомогою якого користувач матиме змогу вибирати потрібні йому інтерфейси. Отже такий варіант виступає у вигляді моста між декількома інтерфейсами. Але якщо користувач потребує одразу в передачі даних через радіоканал то необхідно використати варіант модуля з МК радіомодуля. Такий варіант одразу передбачає передачу даних між двома пристроями через радіоканал, що на сьогоднішній день є дуже актуальним. Функціональна схема модуля МК з радіомодулем зображена на рисунку 5.2

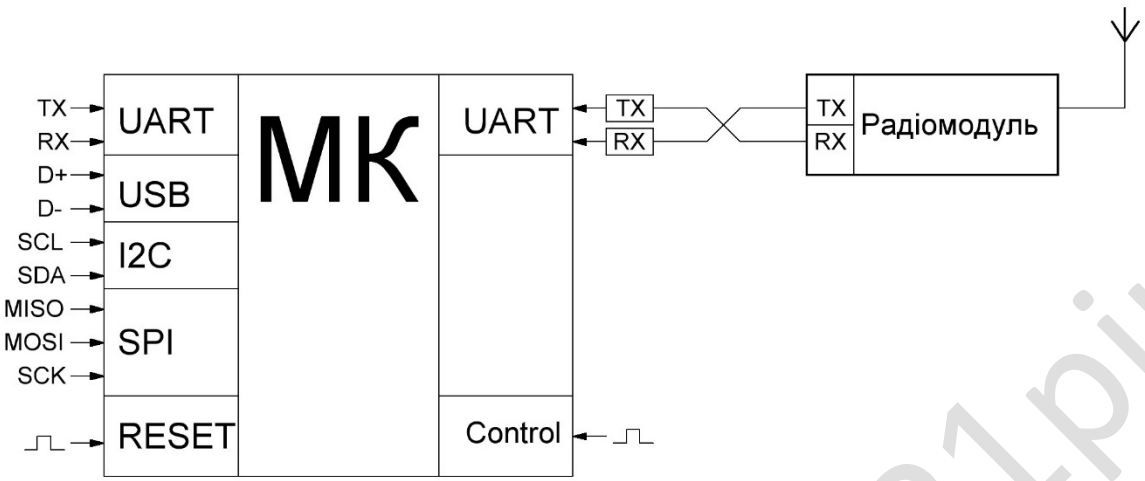


Рисунок 5.2 - Функціональна схема модуля на основі МК та радіомодуля

Але якщо потрібна шифрована передача даних на основі старих аналогових комунікацій можливий варіант передачі даних через модуль чкий сприймає аналоговий сигнал відцифровує його через аналогово-цифровий перетворювач (АЦП) шифрує та перетворює в аналовий сигнал через цифро-аналоговий перетворювач (ЦАП). Функціональна схема модуля з аналоговим вводом та виводом зображена на рисунку 5.3.

Рисунок 5.3 - Функціональна схема модуля з аналоговим вводом та виводом

Отже система на основі шифрованої передачі даних може мати будь який вигляд.

6 НАУКОВА НОВИЗНА

Перевага даного алгоритму в тому що його можна використовувати на будь-якій платформі. На сьогоднішній день вже існують мікроконтролери з вбудованими блоками криптографії. Вони коштують набагато більше відносно контролерів загального призначення, а також є вже добре вивченими зловмисниками.

Наукова новизна отриманих результатів полягає в тому, що:

- вперше застосовано для шифрування трафіку алгоритм шифрування побудований на засадах хаотичного (ентропійного) математичного більярду;

- досліджено та промодельовано швидкість шифрування та дешифрування, стійкість проти впливу зовнішніх факторів на реальних пристроях зокрема на ARM мікроконтролерах;

- алгоритм і ПЗ можуть бути використані в системах швидкої передачі повідомлень як по цифрових так і по аналогових каналах зв’язку.

Впровадження у інші системи можливе у вигляді як заздалегідь запрограмованого мікроконтролера так і у вигляді модуля. У випадку з заздалегідь запрограмованим мікроконтролером користувач просто монтує мікросхему у свою систему. Для аналогової передачі можливе використання окремого модуля з аналоговим входом та аналоговим виходом адаптованими по стандарту використовуваного обладнання. При необхідності цифрової передачі можливе використання окремого модуля з мікроконтролером та радіомодулем. На основі алгоритму можливо виготовити пристрій який буде підключатися до ПК та просто генерувати випадкову послідовність чисел передаючи в порт ПК. Головною перевагою розробленого проекту – це безпечна шифрована передача даних різними каналами зв’язку. Можлива адаптація до старих аналогових систем зв’язку. На основі даної інформації можна зробити висновок, що алгоритм не потребує значних обчислювальних ресурсів для реалізації та окремих апаратних спеціалізованих блоків.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми дипломного проекту

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація шифрованого трафіку через аналоговий тракт.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
Кількість розроблених програм період, шт.	N	1
Кількість екземплярів програм, шт.	Ne	120 (2 ост. цифри № зал*10^1)
Запланований термін розробки, днів	Fpq	60 (3 місяці)
Група задачі підсистеми управління (1-6)	–	1
Ступінь новизни задачі (А, Б, В, Г)	–	Б
Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
Кількість макетів вхідної інформації	–	3
Кількість форм вихідної інформації.	–	4
Мова програмування (1-6)	–	2
Попередній досвід (1-6)	–	3
Гнучкість проекту ПП (1-6)	–	3
Детальність проекту ПП (1-6)	–	2
Рівень спрацьованості колективу (1-6)	–	2
Ступінь вимірності процесів (1-6)	–	3
Необхідна надійність програмного забезпечення (1-6)	–	2
Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
Складність кінцевого програмного продукту (1-6)	–	2
Необхідний рівень забезпечення повторного використання (1-6)	–	2
Документованість відповідно до планованого життєвого циклу (1-6)	–	2
Вимоги до швидкодії ПП (1-6)	–	2
Обмеження на розміри основного сховища даних (1-6)	–	2
Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
Професійний рівень аналітиків (1-6)	–	2
Професійний рівень програмістів (1-6)	–	2
Постійність складу команди розробників (1-6)	–	2
Досвід розробки додатків (1-6)	–	2
Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
Досвід роботи з програмними інструментами розробки (1-6)	–	3
Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
Вартість ПЗ у розробника (НМА), грн.	–	120000 (2 ост. цифри № зал*10^4)
Норматив додаткової зарплати, % :	$Н_д$	10
Норматив відрахувань у соціальні фонди, %	Нс	22
Норматив загальногосподарських витрат, %	Нг	15
Норматив витрат на освоєння нових мов програмування, %	Нп	15
Рівень рентабельності програмної продукції, %	Ре	50
Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \quad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \quad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$
$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \quad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 100 = 168 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3{,}8 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор–маршрутизатор	155	2	310	5
Кабельні господарства ЛВС на 1 м. п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_{ч} \cdot n_{міс}}{1{,}2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{227 \cdot 3}{1{,}2} = 567{,}5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 567{,}5/(60 \cdot 8) = 1{,}2 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	12000	36000
Продакт-менеджер	0,25	8000	6000
Інженер-програміст	3,8	8000	91200
Інженер-електронщик	1,2	6000	21600
Інженер-системотехнік	0,25	6000	4500
Адміністратор мережі	0,5	6000	9000
Системний програміст	0,25	6000	4500
Дизайнер WEB	0,25	8000	6000
Інженер-верстальник	0,25	6000	4500
Бухгалтер-економіст	0,5	10000	15000
Всього за період розробки	$R_{сп}$ = 8,25	-	$Ф_{роб}$ = 198300

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{198300}{8,25 \cdot 60} = 401 \, грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет магазину Компбест за 06.11.21 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core i7 860 8*3.5GHz/8MB/2.5GTs 870 s1156, box	1750
Системна плата	MB MSI H55M-E33 s1156 mATX (H55M-E33)	1200
Відеокарта	VC VTX Radeon HD6570 1GB GDDR3, 128bit, 650 MHz/1334 MHz, PCI-E 2.1, DVI, HDMI, VGA (VX6570 1GBK3-H)	750
Жорсткий диск	HDD: 320 Gb 7200 Serial ATA WD 16MB	1200
Оперативна пам’ять	DIMM 2048Mb DDR3 PC3-12800 Patriot, 1600Mhz, CL9, (9-9-9-28), 1.5V, Retail (PSD32G16002H) 2 модуля	900
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slim Black Bulk (DVR-TD10RS)	416
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA-489, PSU 350W(FSP Brand: ATX-350PNR, 12cm), black, (front bezel – black+light silver; body material – 0.6mm), 80mm fan (rear), 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-less chassis design,Thermally Advantaged Chassis	911

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR -35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	220
Інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1, 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
Будівлі	1280000	-	-
Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Група 5, 6			
Вимірювальні пристрої	5190	25	1297,5
Транспортні засоби	0	20	0,0
Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
Нематеріальні активи	120000	10	12000
Разом	$К_р$ = 1760367		$А_р$ = 190286

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 401 \cdot 209/120 = 698 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 698 \cdot 10 \cdot 0{,}01 = 70 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 37\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(698+70) = 169 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 698 \cdot 15 \cdot 0{,}01 = 105 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.;

$З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картріджей, тонеру, грн.;

N_e – кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо одну пачку паперу на три місяці розробки (n_p=0,33). Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 105 грн., визначаємо вартість паперу за період розробки $N_м$ = 3 міс:

$$З_{М1} = Ц_п \cdot n_р \cdot N_м. \quad (7.16)$$

$$З_{М1} = 105 \cdot 0{,}33 \cdot 3 = 105 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 2 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 2 грн./шт.

$$З_{М2} = 120 \cdot 12 = 1440 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з.., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (105 + 1440 + 1702)/120 = 27 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 698 \cdot 15 \cdot 0{,}01 = 105 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 120 прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 190286 \cdot 3/(120 \cdot 12) = 396 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + C_{оц} + Г_{осп} + З_м + О_п + A_м. \quad (7.21)$$

$$C_п = 698+70+168+105+27+105+396 = 1570 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1. Основна зарплата виконавців	$З_о$	698
2. Додаткова зарплата виконавців	$З_д$	70
3. Відрахувння на соціальні потреби	$C_{оц}$	169
4. Загальногосподарські витрати	$Г_{осп}$	105
5. Витрати на матеріали	$З_М$	27
6. Освоєння нових операційних систем, мов програмування	$О_п$	105
7. Амортизація основних фондів	$A_м$	396
8. Повна собівартість програмного забезпечення	$C_п$	1570
9. Плановий прибуток	$П_р$	785
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	2355
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	471
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	2826

Визначимо плановий прибуток за рівнем рентабельності (P_n) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \quad (7.22)$$

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 1570 = 785 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.9.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	2826
Всього капітальних витрат	–	2826

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	15190	1519
2. Витрати на електроенергію	$З_{ел}$	227	157
Витрати на амортизацію	$З_{ам}$	0	707
Всього витрат за рік	$І$	15417	2383

Витрати на профілактичні роботи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення витрати на обслуговування системи з 15190 грн до 1519 грн на рік.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	2826	–	706,5
Всього відрахувань	-	–	2826	–	706,5

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($T_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot T_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 0{,}545 \cdot 130 \cdot 3{,}2 = 227 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}545 \cdot 90 \cdot 3{,}2 = 157 \text{ грн.}$$

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{р_m} \cdot K_{р_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (2355 - 1570) \cdot 120 - (0{,}05 \cdot 1408000 + 0{,}4 \cdot 199177 +$$
$$+0{,}25 \cdot 33190 + 0{,}1 \cdot 120000) \cdot 3/12 = 51608 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р$ – балансова вартість основних фондів розробника.

$$T_в = \frac{1760367}{(2355 - 1570) \cdot 120 \cdot 12 / 3} = 4{,}6 \text{ роки}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н (K_н - K_б), \quad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (15417\text{-}2383)\text{-}0{,}25 \cdot 2826 = 12327{,}5 \text{ грн.}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	120
2. Повна собівартість розробленої програми	Грн.	1570
3. Ціна розробленої програми	Грн.	2355
4. Плановий прибуток від реалізації розробленої програми	Грн.	785
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1760367
7. Загальний прибуток від реалізації програмної продукції	Грн.	94200
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	51608
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	4,6
10. Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	2826
11. Величина економічного ефекту у користувача програмної продукції	Грн.	12327,5
12. Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,2

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{2826}{15417 - 2383} = 0,2 \ року .$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

ЗАХОДИ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Вступ

«З давніх давен людство приділяє прискіпливу увагу безпеці життя і охорони праці як її складової частини. Умови праці розглядали Арістотель (384-322 до н.е.) та Гіппократ (460-377 до н.е.)» [54].

Законом України “Про охорону праці” [53] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [55], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [52].

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [52], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста вулючають наступні фактори:

Щ

Р

Т

Найменування	З
Ширина	
Довжина	
Висота	

Т

Г	О	Н	Ф
П	м	н	
О	м	н	

У

Т

З

М

У

Т

приміщення

П	О			Фактичні		
	Темпе	Воло	Швид	Темпе	Воло	Швид
Холодна				22		
Тепла				45		

П

В

Д

У приміщенні знаходяться наступні джерела шуму: принтер Epson L3151 Wi-Fi, електродвигуни вентиляторів ЕОМ.

З 2019 року діють Державні будівельні норми україни “Природне і штучне освітлення” – ДБН В.2.5-28:2018 [51], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5-розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

Розробка заходів з умов поліпшення охорони праці

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним [57].

Р о з р…

П

Для захисного штучного заземлення застосовуються ветикальні електроди з метелевого прутка діаметром 30 мм. (*D=30 мм.=0,03 м.)* довжиною *L*=2,5 *м.* та

Г

У

Р

Н

Р

В

Т

Р

частина

ρ

д

ρ

ρ

О

$$R_o = 0,366\frac{\rho}{L}(lg\frac{2L}{D} + \frac{1}{2}lg\frac{4T+L}{4T-L}) = 0,366\frac{54,5}{2,5}(lg\frac{2\cdot 2,5}{0,03} + \frac{1}{2}lg\frac{4\cdot 2,05+2,5}{4\cdot 2,05-2,5}) = 20,1 \text{ Ом.}$$

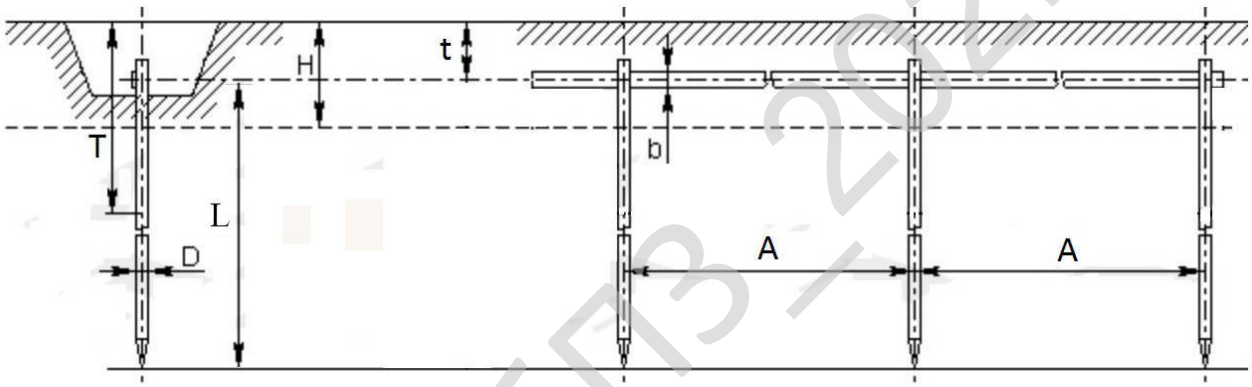
В

В

В

В

Вим	Арк.	№ докум.	Підпис	Дата	ВКРМ-123.21.0012.00.00.ПЗ	Арк. 92



9 ОСНОВНІ ВИСНОВКИ

В результаті дипломної роботи була розглянута та реалізована передача даних по радіоканалу для подальшого впровадження алгоритму шифрування.

В даній роботі було підготовлено програму передачі даних та апаратна реалізація. З програмної сторони було створено керуючу програму для мікроконтролера, яка повинна налагодити радіообмін даними через радіо модулі загального призначення.

Було створено макети, які передбачають обмін даними точка-точка з зв'язком через радіо модулі. Радіо модулі загального призначення на сьогоднішній день наявні в великому асортименті та коштують не дорого. Тому вартість побудованих пристроїв не велика. Але головна перевага – це безпечна шифрована передача даних різними каналами зв'язку. Також можлива адаптація до старих аналогових систем.

Впровадження у інші системи можливе у вигляді як заздалегідь запрограмованого мікроконтролера так і у вигляді модуля. У випадку з заздалегідь запрограмованим мікроконтролером користувач просто монтує мікросхему у свою систему. Якщо ж потрібна аналогова передача на вже існуючих системах можливо використання окремого модуля з аналоговим входом та аналоговим виходом адаптованими по стандарту використовуваного обладнання. При необхідності цифрової передачі можливе використання окремого модуля з мікроконтролером та радіомодулем.

Використання хаотичного більярда Сіная дуже перспективне в системах передачі даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Комп'ютерна безпека [Електронний ресурс]. Режим доступу: ttps://uk.wikipedia.org/wiki/Комп'ютерна_безпека.

2. Атака на ГПВЧ [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Атака_на_ГПВЧ.

3. Гальперин Г.А., Земляков А.Н. Математические бильярды. Бильярдные задачи и смежные вопросыматематики и механики – М.:Наука, 1990. – 288 с.

4. Ганапольский Е.М. О природе квантового хаоса в рассеивающей бильярдной К-системе / Е. М. Ганапольский // Доповіді Національної академії наук України. - 2012. - № 3. - С. 85-91.

5. Собінов О. Г. Простий генератор псевдовипадкової послідовності / О. Г. Собінов // Інформаційні технології та комп'ютерна інженерія : зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 груд. 2014 р. – Кіровоград: КНТУ, 2014. – С. 184.

6. Новости электроники №10 (156), 2016 г. Информационно технический журнал. Учредитель – ООО «КОМПЭЛ».

7. Новости электроники №6 (44), 2012 г. Информационно технический журнал. Учредитель – ООО «КОМПЭЛ».

8. Ганапольский Е.М. О природе квантового хаоса в рассеивающей бильярдной К-системе / Е. М. Ганапольский // Доповіді Національної академії наук України. - 2012. - № 3. - С. 85-91.

9. Шифрування [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Шифрування.

10. Кодування [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Кодування.

11. Хеш-функція [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Хеш-функція.

12. MAC-підпис [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/MAC-підпис.

13. Асиметричні алгоритми шифрування [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Асиметричні_алгоритми_шифрування.

14. Шифрування даних: криптозахист STM32 [Електронний ресурс]. Режим доступу: https://www.compel.ru/lib/78828.

15. STM32 Nucleo Boards [Електронний ресурс]. Режим доступу: https://www.st.com/en/evaluation-tools/stm32-nucleo-boards.html.

16. X-CUBE-CRYPTOLIB [Електронний ресурс]. Режим доступу: https://www.st.com/en/embedded-software/x-cube-cryptolib.html#resource.

17. STMicroelectronics [Електронний ресурс]. Режим доступу: https://www.st.com/content/st_com/en.html.

18. Канал зв'язку [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Канал_зв'язку.

19. Advanced Encryption Standard [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Advanced_Encryption_Standard.

20. Data Encryption Standard [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Data_Encryption_Standard.

21. SecureCryptographicModule [Електронний ресурс]. Режим доступу: https://www.cryptsoft.com/fips140/vendors/140sp2200.pdf.

22. KWD-AE31 AES/DES Encryption Module [Електронний ресурс]. Режим доступу: https://kenwoodcommunications.co.uk/acc/modules/KWD-AE31%20AES/DES%20Encryption%20Module/.

23. Hardware security module [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Hardware_security_module.

24. A Pseudo Random Number Generator Based on Chaotic Billiards [Електронний ресурс]. Режим доступу: http://ijns.femto.com.tw/contents/ijns-v19-n3/ijns-2017-v19-n3-p479-486.pdf.

25. STM32CubeIDE [Електронний ресурс]. Режим доступу: https://www.st.com/en/development-tools/stm32cubeide.html.

26. C та C++ [Електронний ресурс]. Режим доступу:

https://uk.wikipedia.org/wiki/C%2B%2B.

27. SWIM [Електронний ресурс]. Режим доступу: https://www.st.com/en/development-tools/st-link-v2.html.

28. STSW-LINK004 [Електронний ресурс]. Режим доступу: https://www.st.com/en/development-tools/stsw-link004.html#overview.

29. STM32 ST-LINK utility software description [Електронний ресурс]. Режим доступу: https://www.st.com/content/ccc/resource/technical/document/user_manual/e6/10/d8/80/d6/1d/4a/f2/CD00262073.pdf/files/CD00262073.pdf/jcr:content/translations/en.CD00262073.pdf.

30. Saleae Logic [Електронний ресурс]. Режим доступу: https://www.saleae.com/ru/.

31. Terminal program [Електронний ресурс]. Режим доступу: https://www.narom.no/undervisningsressurser/the-cansat-book/the-primary-mission/using-the-radio/terminal-program/.

32. Прямий доступ до пам'яті [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Прямий_доступ_до_пам'яті.

33. Системи радіозв`язку [Електронний ресурс]. Режим доступу: http://openarchive.nure.ua/bitstream/document/6788/1/SR_2.2_MR.pdf.

34. STM32F103C8 [Електронний ресурс]. Режим доступу: https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html.

35. ARM Cortex-M [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/ARM_Cortex-M.

36. JDY-40 2.4G wireless serial port transmission transceiver and remote communication module [Електронний ресурс]. Режим доступу: https://sunhokey.cn/collections/wifi-module/products/jdy-40-2-4g-wireless-serial-port-transmission-transceiver-and-remote-communication-module.

37. Генератор псевдовипадкових чисел [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Генератор_псевдовипадкових_чисел.

38. Синай Я. Г. Теория фазовых переходов: строгие результаты — М.:

Наука, 1980.

39. Y. G. Sinai, \Dynamical systems with elastic reflections," *Russian Mathematical Surveys*, vol. 25, no. 2, pp. 137{189, 1970.

40. Visual Studio Community [Електронний ресурс]. Режим доступу: https://visualstudio.microsoft.com/ru/vs/community/.

41. Matplotlib [Електронний ресурс]. Режим доступу: https://matplotlib.org/.

42. GPIO internal peripheral [Електронний ресурс]. Режим доступу: https://wiki.st.com/stm32mpu/wiki/GPIO_internal_peripheral.

43. AN2582 Application note [Електронний ресурс]. Режим доступу: http://read.pudn.com/downloads106/sourcecode/embed/437624/stm32/STM32F Documents/Application Note/STM32F10xxx USART application examples.pdf.

44. Nested Vectored Interrupt Controller (NVIC) [Електронний ресурс]. Режим доступу: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dai0179 b/ar01s01s01.html

45. USB hardware and PCB guidelines using STM32 MCUs [Електронний ресурс]. Режим доступу: https://usermanual.wiki/Document/stm32usbguide.1433082037/html.

46. Виключна диз'юнкція [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Виключна_диз'юнкція.

47. Коди Гемінга [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Коди_Гемінга.

48. Implementation of security module to protect programme theft in microcontroller-based applications [Електронний ресурс]. Режим доступу: https://www.tandfonline.com/doi/full/10.1080/00051144.2019.1578916.

49. STLINK SWD for STM32 [Електронний ресурс]. Режим доступу: http://www.micromouseonline.com/2011/11/05/stlink-swd-for-stm32/.

50. STM32. Програмування STM32F103. Option bytes [Електронний ресурс]. Режим доступу: https://blog.avislab.com/stm32-ob/.

51. Державні будівельні норми України: ДБН В.2.5-28:2018. - Режим доступу до ресурсу: https://goo.su/9AkQ

52. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

53. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

54. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

55. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

56. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. - Кіровоград: КІСМ, 1997. - 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

57. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

58. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : "Колос", 1973. – 238 с.

59. Центр післядипломної освіти та підвищення кваліфікації. - Режим доступу до ресурсу: https://cpo.stu.cn.ua[

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.21.0012.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Майданик О.О.			Дослідж ення т а програмна реалізація шифрованого т рафіку через аналоговий т ракт	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Минайленко Р.М.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-20М	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи шифрованого трафіку через аналоговий тракт.

2 Підстава для розробки

Підставою для розробки служить завдання на магістерську роботу, видане на кафедрі програмування та захисту інформації (нак. №42-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою магістерської роботи є дослідження та програмна реалізація системи шифрованого трафіку через аналоговий тракт.

4 Джерела розробки

Джерелом цієї магістерської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи шифрованого трафіку через аналоговий тракт;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10. Мікроконтролер STM32F103C8T6.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище STM32CubeIDE 1.1.0.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 13 вересня 2021 року.

8 Вимоги щодо охорони праці

В частині охорони праці магістерської роботи повинна бути розглянута психофізіологічна сутність і структура виробничої діяльності з позиції дослідження інформаційного навантаження оператора ЕОМ.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 1 аркуш.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 102 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі магістерської роботи. Постановка задачі на виконання магістерської роботи (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень магістерської роботи.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання магістерської роботи на попередній захист 04.12.2021 р.

11.2 Подання магістерської роботи на захист 22.12.2021 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник випускної кваліфікаційної роботи
за другим (магістерським) рівнем вищої освіти
_____ Р.М. Минайленко

Дослідження та програмна реалізація шифрованого трафіку через аналоговий тракт

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 39

Літера: РП

Кропивницький – 2021 року

Основна програма

main.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f1xx_hal.h"
#include "stm32f1xx_ll_i2c.h"
#include "stm32f1xx_ll_rcc.h"
#include "stm32f1xx_ll_bus.h"
#include "stm32f1xx_ll_system.h"
#include "stm32f1xx_ll_exti.h"
#include "stm32f1xx_ll_cortex.h"
#include "stm32f1xx_ll_utils.h"
#include "stm32f1xx_ll_pwr.h"
#include "stm32f1xx_ll_dma.h"
#include "stm32f1xx_ll_usart.h"
#include "stm32f1xx.h"
#include "stm32f1xx_ll_gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

  /* USER CODE END Includes */
  /* Exported types ------------------------------------------------------------*/
  /* USER CODE BEGIN ET */

  /* USER CODE END ET */

  /* Exported constants --------------------------------------------------------*/
  /* USER CODE BEGIN EC */

  /* USER CODE END EC */

  /* Exported macro ------------------------------------------------------------*/
  /* USER CODE BEGIN EM */
```

```
/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);

/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

/* Private defines -----------------------------------------------------------*/
#define LED_Pin LL_GPIO_PIN_13
#define LED_GPIO_Port GPIOC
#define SET_Pin LL_GPIO_PIN_15
#define SET_GPIO_Port GPIOB
#define CS_Pin LL_GPIO_PIN_8
#define CS_GPIO_Port GPIOA
/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */
```

main.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "usb_device.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "usbd_cdc_if.h"

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
```

```
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
extern uint8_t UserRxBufferFS[];
extern char RX_Buff[];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_USART1_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */


  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_USART1_UART_Init();
  MX_USB_DEVICE_Init();

  /* USER CODE BEGIN 2 */
  void USART_Send_String(char *buf);
  /* USER CODE END 2 */



  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  //uint8_t MassegeLenght = 0;
```

```
  //uint8_t s = 0;
  LL_GPIO_ResetOutputPin(GPIOC, LL_GPIO_PIN_13);
 // LL_GPIO_SetOutputPin(GPIOA, CS_Pin);// Chip Select
  LL_GPIO_SetOutputPin(GPIOB, LL_GPIO_PIN_15);// Set Comand
      LL_GPIO_ResetOutputPin(GPIOA, CS_Pin);// Chip Select
      //LL_GPIO_ResetOutputPin(GPIOA, SET_Pin);// Set Comand
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

         if(UserRxBufferFS[0] != 0)
         {
               for(uint8_t i = 0; i < 30; i++)
               {
                     LL_USART_TransmitData8(USART1, UserRxBufferFS[i]);
                     if(('/' == UserRxBufferFS[i])&&('n' == UserRxBufferFS[i+1]))
                     {
                           break;
                     }
               }
               for(uint8_t i = 0; i < 30; i++)
               {
                     UserRxBufferFS[i]=0;
               }

         }
         if(RX_Buff[0] !=0)
         {
               for(uint8_t i = 0; i < 30; i++)
               {
                     CDC_Transmit_FS(RX_Buff[i], 1);
                     if(('/' == RX_Buff[i])&&('n' == RX_Buff[i+1]))
                     {
                           break;
                     }
               }
               for(uint8_t i = 0; i < 30; i++)
               {
                     RX_Buff[i]=0;
               }

         }
         if(UserRxBufferFS[0]=='0')
                     {
                     LL_GPIO_SetOutputPin(GPIOC, LL_GPIO_PIN_13);
                     USART_Send_String("Abc");

                     UserRxBufferFS[0]=0;
                     }
                else if(UserRxBufferFS[0]=='1')
                     {
                       LL_GPIO_ResetOutputPin(GPIOC, LL_GPIO_PIN_13);
                       LL_USART_TransmitData8(USART1, 'W');
                       UserRxBufferFS[0]=0;
                     }

*/
                LL_mDelay(100);
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
```

```
  */
void SystemClock_Config(void)
{
  LL_FLASH_SetLatency(LL_FLASH_LATENCY_1);

  if(LL_FLASH_GetLatency() != LL_FLASH_LATENCY_1)
  {
    Error_Handler();
  }
  LL_RCC_HSE_Enable();

   /* Wait till HSE is ready */
  while(LL_RCC_HSE_IsReady() != 1)
  {

  }
  LL_RCC_PLL_ConfigDomain_SYS(LL_RCC_PLLSOURCE_HSE_DIV_1, LL_RCC_PLL_MUL_6);
  LL_RCC_PLL_Enable();

   /* Wait till PLL is ready */
  while(LL_RCC_PLL_IsReady() != 1)
  {

  }
  LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
  LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_2);
  LL_RCC_SetAPB2Prescaler(LL_RCC_APB2_DIV_1);
  LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_PLL);

   /* Wait till System clock is ready */
  while(LL_RCC_GetSysClkSource() != LL_RCC_SYS_CLKSOURCE_STATUS_PLL)
  {

  }
  LL_SetSystemCoreClock(48000000);

   /* Update the time base */
  if (HAL_InitTick (TICK_INT_PRIORITY) != HAL_OK)
  {
    Error_Handler();
  };
  LL_RCC_SetUSBClockSource(LL_RCC_USB_CLKSOURCE_PLL);
}

/**
  * @brief I2C1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2C1_Init(void)
{

  /* USER CODE BEGIN I2C1_Init 0 */

  /* USER CODE END I2C1_Init 0 */

  LL_I2C_InitTypeDef I2C_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_GPIOB);
  /**I2C1 GPIO Configuration
  PB6   ------> I2C1_SCL
  PB7   ------> I2C1_SDA
  */
  GPIO_InitStruct.Pin = LL_GPIO_PIN_6|LL_GPIO_PIN_7;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
```

```
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_HIGH;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_OPENDRAIN;
  LL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /* Peripheral clock enable */
  LL_APB1_GRP1_EnableClock(LL_APB1_GRP1_PERIPH_I2C1);

  /* USER CODE BEGIN I2C1_Init 1 */

  /* USER CODE END I2C1_Init 1 */
  /** I2C Initialization
  */
  LL_I2C_DisableOwnAddress2(I2C1);
  LL_I2C_DisableGeneralCall(I2C1);
  LL_I2C_EnableClockStretching(I2C1);
  I2C_InitStruct.PeripheralMode = LL_I2C_MODE_I2C;
  I2C_InitStruct.ClockSpeed = 100000;
  I2C_InitStruct.DutyCycle = LL_I2C_DUTYCYCLE_2;
  I2C_InitStruct.OwnAddress1 = 0;
  I2C_InitStruct.TypeAcknowledge = LL_I2C_ACK;
  I2C_InitStruct.OwnAddrSize = LL_I2C_OWNADDRESS1_7BIT;
  LL_I2C_Init(I2C1, &I2C_InitStruct);
  LL_I2C_SetOwnAddress2(I2C1, 0);
  /* USER CODE BEGIN I2C1_Init 2 */

  /* USER CODE END I2C1_Init 2 */

}

/**
  * @brief USART1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART1_UART_Init(void)
{

  /* USER CODE BEGIN USART1_Init 0 */

  /* USER CODE END USART1_Init 0 */

  LL_USART_InitTypeDef USART_InitStruct = {0};

  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* Peripheral clock enable */
  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_USART1);

  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_GPIOA);
  /**USART1 GPIO Configuration
  PA9   ------> USART1_TX
  PA10   ------> USART1_RX
  */
  GPIO_InitStruct.Pin = LL_GPIO_PIN_9;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_HIGH;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  GPIO_InitStruct.Pin = LL_GPIO_PIN_10;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_FLOATING;
  LL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USART1 interrupt Init */
  NVIC_SetPriority(USART1_IRQn, NVIC_EncodePriority(NVIC_GetPriorityGrouping(),0, 0));
  NVIC_EnableIRQ(USART1_IRQn);
```

```
  /* USER CODE BEGIN USART1_Init 1 */

  /* USER CODE END USART1_Init 1 */
  USART_InitStruct.BaudRate = 9600;
  USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
  USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
  USART_InitStruct.Parity = LL_USART_PARITY_NONE;
  USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
  USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
  USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
  LL_USART_Init(USART1, &USART_InitStruct);
  LL_USART_ConfigAsyncMode(USART1);
  LL_USART_Enable(USART1);
  /* USER CODE BEGIN USART1_Init 2 */
  LL_USART_EnableIT_RXNE(USART1);
  /* USER CODE END USART1_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  LL_GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_GPIOC);
  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_GPIOD);
  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_GPIOB);
  LL_APB2_GRP1_EnableClock(LL_APB2_GRP1_PERIPH_GPIOA);

  /**/
  LL_GPIO_ResetOutputPin(LED_GPIO_Port, LED_Pin);

  /**/
  LL_GPIO_ResetOutputPin(SET_GPIO_Port, SET_Pin);

  /**/
  LL_GPIO_ResetOutputPin(CS_GPIO_Port, CS_Pin);

  /**/
  GPIO_InitStruct.Pin = LED_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  LL_GPIO_Init(LED_GPIO_Port, &GPIO_InitStruct);

  /**/
  GPIO_InitStruct.Pin = SET_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  LL_GPIO_Init(SET_GPIO_Port, &GPIO_InitStruct);

  /**/
  GPIO_InitStruct.Pin = CS_Pin;
  GPIO_InitStruct.Mode = LL_GPIO_MODE_OUTPUT;
  GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
  LL_GPIO_Init(CS_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */
```

```
//static void JDY_40_Init(void)
//{
void USART_Send_String(char *buf)
{
	uint8_t i = 0;
	while(buf[i] !=0)
	{
		while(!(LL_USART_IsActiveFlag_TXE(USART1)));
		LL_USART_TransmitData8(USART1, buf[i++]);
	}
}

void JDY_40_Config(void)
{
	LL_GPIO_ResetOutputPin(GPIOA, CS_Pin);// Chip Select
	LL_GPIO_ResetOutputPin(GPIOA, SET_Pin);// Set Comand
	USART_Send_String("AT");
}


/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

usb_device.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : usb_device.h
  * @version        : v2.0_Cube
  * @brief          : Header for usb_device.c file.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
```

```
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __USB_DEVICE__H__
#define __USB_DEVICE__H__

#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f1xx.h"
#include "stm32f1xx_hal.h"
#include "usbd_def.h"

/* USER CODE BEGIN INCLUDE */

/* USER CODE END INCLUDE */

/** @addtogroup USBD_OTG_DRIVER
  * @{
  */

/** @defgroup USBD_DEVICE USBD_DEVICE
  * @brief Device file for Usb otg low level driver.
  * @{
  */

/** @defgroup USBD_DEVICE_Exported_Variables USBD_DEVICE_Exported_Variables
  * @brief Public variables.
  * @{
  */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/*
 * -- Insert your variables declaration here --
 */
/* USER CODE BEGIN VARIABLES */

/* USER CODE END VARIABLES */
/**
  * @}
  */

/** @defgroup USBD_DEVICE_Exported_FunctionsPrototype
USBD_DEVICE_Exported_FunctionsPrototype
  * @brief Declaration of public functions for Usb device.
  * @{
  */

/** USB Device initialization function. */
void MX_USB_DEVICE_Init(void);
```

```
/*
 * -- Insert functions declaration here --
 */
/* USER CODE BEGIN FD */

/* USER CODE END FD */
/**
  * @}
  */

/**
  * @}
  */

/**
  * @}
  */

#ifdef __cplusplus
}
#endif

#endif /* __USB_DEVICE__H__ */
```

usb_device.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : usb_device.c
  * @version        : v2.0_Cube
  * @brief          : This file implements the USB Device
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/

#include "usb_device.h"
#include "usbd_core.h"
#include "usbd_desc.h"
#include "usbd_cdc.h"
#include "usbd_cdc_if.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/

/* USER CODE END PV */

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/
```

```
/* USER CODE END PFP */

/* USB Device Core handle declaration. */
USBD_HandleTypeDef hUsbDeviceFS;

/*
 * -- Insert your variables declaration here --
 */
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/*
 * -- Insert your external function declaration here --
 */
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/**
  * Init USB device Library, add supported class and start the library
  * @retval None
  */
void MX_USB_DEVICE_Init(void)
{
  /* USER CODE BEGIN USB_DEVICE_Init_PreTreatment */

  /* USER CODE END USB_DEVICE_Init_PreTreatment */

  /* Init Device Library, add supported class and start the library. */
  if (USBD_Init(&hUsbDeviceFS, &FS_Desc, DEVICE_FS) != USBD_OK)
  {
    Error_Handler();
  }
  if (USBD_RegisterClass(&hUsbDeviceFS, &USBD_CDC) != USBD_OK)
  {
    Error_Handler();
  }
  if (USBD_CDC_RegisterInterface(&hUsbDeviceFS, &USBD_Interface_fops_FS) != USBD_OK)
  {
    Error_Handler();
  }
  if (USBD_Start(&hUsbDeviceFS) != USBD_OK)
  {
    Error_Handler();
  }

  /* USER CODE BEGIN USB_DEVICE_Init_PostTreatment */

  /* USER CODE END USB_DEVICE_Init_PostTreatment */
}
```

usbd_cdc_if.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : usbd_cdc_if.h
  * @version        : v2.0_Cube
  * @brief          : Header for usbd_cdc_if.c file.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
```

```
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                                www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __USBD_CDC_IF_H__
#define __USBD_CDC_IF_H__

#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "usbd_cdc.h"

/* USER CODE BEGIN INCLUDE */

/* USER CODE END INCLUDE */

/** @addtogroup STM32_USB_OTG_DEVICE_LIBRARY
  * @brief For Usb device.
  * @{
  */

/** @defgroup USBD_CDC_IF USBD_CDC_IF
  * @brief Usb VCP device module
  * @{
  */

/** @defgroup USBD_CDC_IF_Exported_Defines USBD_CDC_IF_Exported_Defines
  * @brief Defines.
  * @{
  */
/* USER CODE BEGIN EXPORTED_DEFINES */

/* USER CODE END EXPORTED_DEFINES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_Types USBD_CDC_IF_Exported_Types
  * @brief Types.
  * @{
  */

/* USER CODE BEGIN EXPORTED_TYPES */

  /* USER CODE END EXPORTED_TYPES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_Macros USBD_CDC_IF_Exported_Macros
  * @brief Aliases.
  * @{
  */

/* USER CODE BEGIN EXPORTED_MACRO */

/* USER CODE END EXPORTED_MACRO */
```

```
/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_Variables USBD_CDC_IF_Exported_Variables
  * @brief Public variables.
  * @{
  */

/** CDC Interface callback. */
extern USBD_CDC_ItfTypeDef USBD_Interface_fops_FS;

/* USER CODE BEGIN EXPORTED_VARIABLES */

/* USER CODE END EXPORTED_VARIABLES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_FunctionsPrototype
USBD_CDC_IF_Exported_FunctionsPrototype
  * @brief Public functions declaration.
  * @{
  */

uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len);

/* USER CODE BEGIN EXPORTED_FUNCTIONS */

/* USER CODE END EXPORTED_FUNCTIONS */


#ifdef __cplusplus
}
#endif

#endif /* __USBD_CDC_IF_H__ */
```

usbd_cdc_if.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : usbd_cdc_if.c
  * @version        : v2.0_Cube
  * @brief          : Usb device for Virtual Com Port.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "usbd_cdc_if.h"

/* USER CODE BEGIN INCLUDE */
```

```
/* USER CODE END INCLUDE */

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
//extern RX_Buff[];
/* USER CODE END PV */

/** @addtogroup STM32_USB_OTG_DEVICE_LIBRARY
  * @brief Usb device library.
  * @{
  */

/** @addtogroup USBD_CDC_IF
  * @{
  */

/** @defgroup USBD_CDC_IF_Private_TypesDefinitions USBD_CDC_IF_Private_TypesDefinitions
  * @brief Private types.
  * @{
  */

/* USER CODE BEGIN PRIVATE_TYPES */

/* USER CODE END PRIVATE_TYPES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Private_Defines USBD_CDC_IF_Private_Defines
  * @brief Private defines.
  * @{
  */

/* USER CODE BEGIN PRIVATE_DEFINES */
/* Define size for the receive and transmit buffer over CDC */
/* It's up to user to redefine and/or remove those define */
#define APP_RX_DATA_SIZE  1000
#define APP_TX_DATA_SIZE  1000
/* USER CODE END PRIVATE_DEFINES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Private_Macros USBD_CDC_IF_Private_Macros
  * @brief Private macros.
  * @{
  */

/* USER CODE BEGIN PRIVATE_MACRO */

/* USER CODE END PRIVATE_MACRO */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Private_Variables USBD_CDC_IF_Private_Variables
  * @brief Private variables.
  * @{
  */
/* Create buffer for reception and transmission           */
```

```
/* It's up to user to redefine and/or remove those define */
/** Received data over USB are stored in this buffer      */
uint8_t UserRxBufferFS[APP_RX_DATA_SIZE];

/** Data to send over USB CDC are stored in this buffer   */
uint8_t UserTxBufferFS[APP_TX_DATA_SIZE];

/* USER CODE BEGIN PRIVATE_VARIABLES */

/* USER CODE END PRIVATE_VARIABLES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_Variables USBD_CDC_IF_Exported_Variables
  * @brief Public variables.
  * @{
  */

extern USBD_HandleTypeDef hUsbDeviceFS;

/* USER CODE BEGIN EXPORTED_VARIABLES */

/* USER CODE END EXPORTED_VARIABLES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Private_FunctionPrototypes
USBD_CDC_IF_Private_FunctionPrototypes
  * @brief Private functions declaration.
  * @{
  */

static int8_t CDC_Init_FS(void);
static int8_t CDC_DeInit_FS(void);
static int8_t CDC_Control_FS(uint8_t cmd, uint8_t* pbuf, uint16_t length);
static int8_t CDC_Receive_FS(uint8_t* pbuf, uint32_t *Len);

/* USER CODE BEGIN PRIVATE_FUNCTIONS_DECLARATION */
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len);
/* USER CODE END PRIVATE_FUNCTIONS_DECLARATION */

/**
  * @}
  */

USBD_CDC_ItfTypeDef USBD_Interface_fops_FS =
{
  CDC_Init_FS,
  CDC_DeInit_FS,
  CDC_Control_FS,
  CDC_Receive_FS
};

/* Private functions ---------------------------------------------------------*/
/**
  * @brief  Initializes the CDC media low layer over the FS USB IP
  * @retval USBD_OK if all operations are OK else USBD_FAIL
  */
static int8_t CDC_Init_FS(void)
{
  /* USER CODE BEGIN 3 */
  /* Set Application Buffers */
  USBD_CDC_SetTxBuffer(&hUsbDeviceFS, UserTxBufferFS, 0);
```

```
  USBD_CDC_SetRxBuffer(&hUsbDeviceFS, UserRxBufferFS);
  return (USBD_OK);
  /* USER CODE END 3 */
}

/**
  * @brief  DeInitializes the CDC media low layer
  * @retval USBD_OK if all operations are OK else USBD_FAIL
  */
static int8_t CDC_DeInit_FS(void)
{
  /* USER CODE BEGIN 4 */
  return (USBD_OK);
  /* USER CODE END 4 */
}

/**
  * @brief  Manage the CDC class requests
  * @param  cmd: Command code
  * @param  pbuf: Buffer containing command data (request parameters)
  * @param  length: Number of data to be sent (in bytes)
  * @retval Result of the operation: USBD_OK if all operations are OK else USBD_FAIL
  */
static int8_t CDC_Control_FS(uint8_t cmd, uint8_t* pbuf, uint16_t length)
{
  /* USER CODE BEGIN 5 */
  switch(cmd)
  {
    case CDC_SEND_ENCAPSULATED_COMMAND:

    break;

    case CDC_GET_ENCAPSULATED_RESPONSE:

    break;

    case CDC_SET_COMM_FEATURE:

    break;

    case CDC_GET_COMM_FEATURE:

    break;

    case CDC_CLEAR_COMM_FEATURE:

    break;

  /*******************************************************************************/
  /* Line Coding Structure                                                       */
  /*-----------------------------------------------------------------------------*/
  /* Offset | Field       | Size | Value  | Description                         */
  /* 0      | dwDTERate   |   4  | Number |Data terminal rate, in bits per second*/
  /* 4      | bCharFormat |   1  | Number | Stop bits                           */
  /*                                        0 - 1 Stop bit                       */
  /*                                        1 - 1.5 Stop bits                    */
  /*                                        2 - 2 Stop bits                      */
  /* 5      | bParityType |  1   | Number | Parity                              */
  /*                                        0 - None                             */
  /*                                        1 - Odd                              */
  /*                                        2 - Even                             */
  /*                                        3 - Mark                             */
  /*                                        4 - Space                            */
  /* 6      | bDataBits  |   1   | Number Data bits (5, 6, 7, 8 or 16).          */
  /*******************************************************************************/
    case CDC_SET_LINE_CODING:
```

```
    break;

    case CDC_GET_LINE_CODING:

    break;

    case CDC_SET_CONTROL_LINE_STATE:

    break;

    case CDC_SEND_BREAK:

    break;

  default:
    break;
  }

  return (USBD_OK);
  /* USER CODE END 5 */
}

/**
  * @brief  Data received over USB OUT endpoint are sent over CDC interface
  *         through this function.
  *
  *         @note
  *         This function will block any OUT packet reception on USB endpoint
  *         untill exiting this function. If you exit this function before transfer
  *         is complete on CDC interface (ie. using DMA controller) it will result
  *         in receiving more data while previous ones are still not sent.
  *
  * @param  Buf: Buffer of data to be received
  * @param  Len: Number of data received (in bytes)
  * @retval Result of the operation: USBD_OK if all operations are OK else USBD_FAIL
  */
static int8_t CDC_Receive_FS(uint8_t* Buf, uint32_t *Len)
{
  /* USER CODE BEGIN 6 */
  USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
  USBD_CDC_ReceivePacket(&hUsbDeviceFS);
  return (USBD_OK);
  /* USER CODE END 6 */
}

/**
  * @brief  CDC_Transmit_FS
  *         Data to send over USB IN endpoint are sent over CDC interface
  *         through this function.
  *         @note
  *
  *
  * @param  Buf: Buffer of data to be sent
  * @param  Len: Number of data to be sent (in bytes)
  * @retval USBD_OK if all operations are OK else USBD_FAIL or USBD_BUSY
  */
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
{
  uint8_t result = USBD_OK;
  /* USER CODE BEGIN 7 */
  USBD_CDC_HandleTypeDef *hcdc = (USBD_CDC_HandleTypeDef*)hUsbDeviceFS.pClassData;
  if (hcdc->TxState != 0){
    return USBD_BUSY;
  }
  USBD_CDC_SetTxBuffer(&hUsbDeviceFS, Buf, Len);
  result = USBD_CDC_TransmitPacket(&hUsbDeviceFS);
  /* USER CODE END 7 */
```

```
  return result;
}

/* USER CODE BEGIN PRIVATE_FUNCTIONS_IMPLEMENTATION */

/* USER CODE END PRIVATE_FUNCTIONS_IMPLEMENTATION */

                                        usbd_desc.c
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : usbd_desc.c
  * @version        : v2.0_Cube
  * @brief          : Header for usbd_conf.c file.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                              www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __USBD_DESC__C__
#define __USBD_DESC__C__

#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "usbd_def.h"

/* USER CODE BEGIN INCLUDE */

/* USER CODE END INCLUDE */

/** @addtogroup STM32_USB_OTG_DEVICE_LIBRARY
  * @{
  */

/** @defgroup USBD_DESC USBD_DESC
  * @brief Usb device descriptors module.
  * @{
  */

/** @defgroup USBD_DESC_Exported_Constants USBD_DESC_Exported_Constants
  * @brief Constants.
  * @{
  */
#define         DEVICE_ID1          (UID_BASE)
#define         DEVICE_ID2          (UID_BASE + 0x4)
#define         DEVICE_ID3          (UID_BASE + 0x8)

#define  USB_SIZ_STRING_SERIAL       0x1A

/* USER CODE BEGIN EXPORTED_CONSTANTS */

/* USER CODE END EXPORTED_CONSTANTS */

/**
```

```
  * @}
  */

/** @defgroup USBD_DESC_Exported_Defines USBD_DESC_Exported_Defines
  * @brief Defines.
  * @{
  */

/* USER CODE BEGIN EXPORTED_DEFINES */

/* USER CODE END EXPORTED_DEFINES */

/**
  * @}
  */

/** @defgroup USBD_DESC_Exported_TypesDefinitions USBD_DESC_Exported_TypesDefinitions
  * @brief Types.
  * @{
  */

/* USER CODE BEGIN EXPORTED_TYPES */

/* USER CODE END EXPORTED_TYPES */

/**
  * @}
  */

/** @defgroup USBD_DESC_Exported_Macros USBD_DESC_Exported_Macros
  * @brief Aliases.
  * @{
  */

/* USER CODE BEGIN EXPORTED_MACRO */

/* USER CODE END EXPORTED_MACRO */

/**
  * @}
  */

/** @defgroup USBD_DESC_Exported_Variables USBD_DESC_Exported_Variables
  * @brief Public variables.
  * @{
  */

/** Descriptor for the Usb device. */
extern USBD_DescriptorsTypeDef FS_Desc;

/* USER CODE BEGIN EXPORTED_VARIABLES */

  /* USER CODE END EXPORTED_VARIABLES */

/**
  * @}
  */

/** @defgroup USBD_DESC_Exported_FunctionsPrototype
USBD_DESC_Exported_FunctionsPrototype
  * @brief Public functions declaration.
  * @{
  */

/* USER CODE BEGIN EXPORTED_FUNCTIONS */

/* USER CODE END EXPORTED_FUNCTIONS */
```

```
#ifdef __cplusplus
}
#endif

#endif /* __USBD_DESC__C__ */
```

usbd_desc.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : App/usbd_desc.c
  * @version        : v2.0_Cube
  * @brief          : This file implements the USB device descriptors.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "usbd_core.h"
#include "usbd_desc.h"
#include "usbd_conf.h"

/* USER CODE BEGIN INCLUDE */

/* USER CODE END INCLUDE */

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/

/* USER CODE END PV */

/** @addtogroup STM32_USB_OTG_DEVICE_LIBRARY
  * @{
  */

/** @addtogroup USBD_DESC
  * @{
  */

/** @defgroup USBD_DESC_Private_TypesDefinitions USBD_DESC_Private_TypesDefinitions
  * @brief Private types.
  * @{
  */

/* USER CODE BEGIN PRIVATE_TYPES */

/* USER CODE END PRIVATE_TYPES */

/**
  * @}
  */
```

```
/** @defgroup USBD_DESC_Private_Defines USBD_DESC_Private_Defines
  * @brief Private defines.
  * @{
  */

#define USBD_VID     1155
#define USBD_LANGID_STRING     1033
#define USBD_MANUFACTURER_STRING     "STMicroelectronics"
#define USBD_PID_FS     22336
#define USBD_PRODUCT_STRING_FS     "STM32 Virtual ComPort"
#define USBD_CONFIGURATION_STRING_FS     "CDC Config"
#define USBD_INTERFACE_STRING_FS     "CDC Interface"

/* USER CODE BEGIN PRIVATE_DEFINES */

/* USER CODE END PRIVATE_DEFINES */

/**
  * @}
  */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/** @defgroup USBD_DESC_Private_Macros USBD_DESC_Private_Macros
  * @brief Private macros.
  * @{
  */

/* USER CODE BEGIN PRIVATE_MACRO */

/* USER CODE END PRIVATE_MACRO */

/**
  * @}
  */

/** @defgroup USBD_DESC_Private_FunctionPrototypes USBD_DESC_Private_FunctionPrototypes
  * @brief Private functions declaration.
  * @{
  */

static void Get_SerialNum(void);
static void IntToUnicode(uint32_t value, uint8_t * pbuf, uint8_t len);

/**
  * @}
  */


/** @defgroup USBD_DESC_Private_FunctionPrototypes USBD_DESC_Private_FunctionPrototypes
  * @brief Private functions declaration for FS.
  * @{
  */

uint8_t * USBD_FS_DeviceDescriptor(USBD_SpeedTypeDef speed, uint16_t *length);
uint8_t * USBD_FS_LangIDStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length);
uint8_t * USBD_FS_ManufacturerStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length);
uint8_t * USBD_FS_ProductStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length);
uint8_t * USBD_FS_SerialStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length);
uint8_t * USBD_FS_ConfigStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length);
uint8_t * USBD_FS_InterfaceStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length);

/**
  * @}
```

```
  */

/** @defgroup USBD_DESC_Private_Variables USBD_DESC_Private_Variables
  * @brief Private variables.
  * @{
  */

USBD_DescriptorsTypeDef FS_Desc =
{
  USBD_FS_DeviceDescriptor
, USBD_FS_LangIDStrDescriptor
, USBD_FS_ManufacturerStrDescriptor
, USBD_FS_ProductStrDescriptor
, USBD_FS_SerialStrDescriptor
, USBD_FS_ConfigStrDescriptor
, USBD_FS_InterfaceStrDescriptor
};

#if defined ( __ICCARM__ ) /* IAR Compiler */
  #pragma data_alignment=4
#endif /* defined ( __ICCARM__ ) */
/** USB standard device descriptor. */
__ALIGN_BEGIN uint8_t USBD_FS_DeviceDesc[USB_LEN_DEV_DESC] __ALIGN_END =
{
  0x12,                       /*bLength */
  USB_DESC_TYPE_DEVICE,       /*bDescriptorType*/
  0x00,                       /*bcdUSB */
  0x02,
  0x02,                       /*bDeviceClass*/
  0x02,                       /*bDeviceSubClass*/
  0x00,                       /*bDeviceProtocol*/
  USB_MAX_EP0_SIZE,           /*bMaxPacketSize*/
  LOBYTE(USBD_VID),           /*idVendor*/
  HIBYTE(USBD_VID),           /*idVendor*/
  LOBYTE(USBD_PID_FS),        /*idProduct*/
  HIBYTE(USBD_PID_FS),        /*idProduct*/
  0x00,                       /*bcdDevice rel. 2.00*/
  0x02,
  USBD_IDX_MFC_STR,           /*Index of manufacturer  string*/
  USBD_IDX_PRODUCT_STR,       /*Index of product string*/
  USBD_IDX_SERIAL_STR,        /*Index of serial number string*/
  USBD_MAX_NUM_CONFIGURATION  /*bNumConfigurations*/
};

/* USB_DeviceDescriptor */

/**
  * @}
  */

/** @defgroup USBD_DESC_Private_Variables USBD_DESC_Private_Variables
  * @brief Private variables.
  * @{
  */

#if defined ( __ICCARM__ ) /* IAR Compiler */
  #pragma data_alignment=4
#endif /* defined ( __ICCARM__ ) */

/** USB lang indentifier descriptor. */
__ALIGN_BEGIN uint8_t USBD_LangIDDesc[USB_LEN_LANGID_STR_DESC] __ALIGN_END =
{
     USB_LEN_LANGID_STR_DESC,
     USB_DESC_TYPE_STRING,
     LOBYTE(USBD_LANGID_STRING),
     HIBYTE(USBD_LANGID_STRING)
};
```

```
#if defined ( __ICCARM__ ) /* IAR Compiler */
  #pragma data_alignment=4
#endif /* defined ( __ICCARM__ ) */
/* Internal string descriptor. */
__ALIGN_BEGIN uint8_t USBD_StrDesc[USBD_MAX_STR_DESC_SIZ] __ALIGN_END;

#if defined ( __ICCARM__ ) /*!< IAR Compiler */
  #pragma data_alignment=4
#endif
__ALIGN_BEGIN uint8_t USBD_StringSerial[USB_SIZ_STRING_SERIAL] __ALIGN_END = {
  USB_SIZ_STRING_SERIAL,
  USB_DESC_TYPE_STRING,
};

/**
  * @}
  */

/** @defgroup USBD_DESC_Private_Functions USBD_DESC_Private_Functions
  * @brief Private functions.
  * @{
  */

/**
  * @brief  Return the device descriptor
  * @param  speed : Current device speed
  * @param  length : Pointer to data length variable
  * @retval Pointer to descriptor buffer
  */
uint8_t * USBD_FS_DeviceDescriptor(USBD_SpeedTypeDef speed, uint16_t *length)
{
  UNUSED(speed);
  *length = sizeof(USBD_FS_DeviceDesc);
  return USBD_FS_DeviceDesc;
}

/**
  * @brief  Return the LangID string descriptor
  * @param  speed : Current device speed
  * @param  length : Pointer to data length variable
  * @retval Pointer to descriptor buffer
  */
uint8_t * USBD_FS_LangIDStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length)
{
  UNUSED(speed);
  *length = sizeof(USBD_LangIDDesc);
  return USBD_LangIDDesc;
}

/**
  * @brief  Return the product string descriptor
  * @param  speed : Current device speed
  * @param  length : Pointer to data length variable
  * @retval Pointer to descriptor buffer
  */
uint8_t * USBD_FS_ProductStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length)
{
  if(speed == 0)
  {
    USBD_GetString((uint8_t *)USBD_PRODUCT_STRING_FS, USBD_StrDesc, length);
  }
  else
  {
    USBD_GetString((uint8_t *)USBD_PRODUCT_STRING_FS, USBD_StrDesc, length);
  }
  return USBD_StrDesc;
```

```
}

/**
  * @brief  Return the manufacturer string descriptor
  * @param  speed : Current device speed
  * @param  length : Pointer to data length variable
  * @retval Pointer to descriptor buffer
  */
uint8_t * USBD_FS_ManufacturerStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length)
{
  UNUSED(speed);
  USBD_GetString((uint8_t *)USBD_MANUFACTURER_STRING, USBD_StrDesc, length);
  return USBD_StrDesc;
}

/**
  * @brief  Return the serial number string descriptor
  * @param  speed : Current device speed
  * @param  length : Pointer to data length variable
  * @retval Pointer to descriptor buffer
  */
uint8_t * USBD_FS_SerialStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length)
{
  UNUSED(speed);
  *length = USB_SIZ_STRING_SERIAL;

  /* Update the serial number string descriptor with the data from the unique
   * ID */
  Get_SerialNum();
  /* USER CODE BEGIN USBD_FS_SerialStrDescriptor */

  /* USER CODE END USBD_FS_SerialStrDescriptor */
  return (uint8_t *) USBD_StringSerial;
}

/**
  * @brief  Return the configuration string descriptor
  * @param  speed : Current device speed
  * @param  length : Pointer to data length variable
  * @retval Pointer to descriptor buffer
  */
uint8_t * USBD_FS_ConfigStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length)
{
  if(speed == USBD_SPEED_HIGH)
  {
    USBD_GetString((uint8_t *)USBD_CONFIGURATION_STRING_FS, USBD_StrDesc, length);
  }
  else
  {
    USBD_GetString((uint8_t *)USBD_CONFIGURATION_STRING_FS, USBD_StrDesc, length);
  }
  return USBD_StrDesc;
}

/**
  * @brief  Return the interface string descriptor
  * @param  speed : Current device speed
  * @param  length : Pointer to data length variable
  * @retval Pointer to descriptor buffer
  */
uint8_t * USBD_FS_InterfaceStrDescriptor(USBD_SpeedTypeDef speed, uint16_t *length)
{
  if(speed == 0)
  {
    USBD_GetString((uint8_t *)USBD_INTERFACE_STRING_FS, USBD_StrDesc, length);
  }
  else
```

```
  {
    USBD_GetString((uint8_t *)USBD_INTERFACE_STRING_FS, USBD_StrDesc, length);
  }
  return USBD_StrDesc;
}

/**
  * @brief  Create the serial number string descriptor
  * @param  None
  * @retval None
  */
static void Get_SerialNum(void)
{
  uint32_t deviceserial0, deviceserial1, deviceserial2;

  deviceserial0 = *(uint32_t *) DEVICE_ID1;
  deviceserial1 = *(uint32_t *) DEVICE_ID2;
  deviceserial2 = *(uint32_t *) DEVICE_ID3;

  deviceserial0 += deviceserial2;

  if (deviceserial0 != 0)
  {
    IntToUnicode(deviceserial0, &USBD_StringSerial[2], 8);
    IntToUnicode(deviceserial1, &USBD_StringSerial[18], 4);
  }
}

/**
  * @brief  Convert Hex 32Bits value into char
  * @param  value: value to convert
  * @param  pbuf: pointer to the buffer
  * @param  len: buffer length
  * @retval None
  */
static void IntToUnicode(uint32_t value, uint8_t * pbuf, uint8_t len)
{
  uint8_t idx = 0;

  for (idx = 0; idx < len; idx++)
  {
    if (((value >> 28)) < 0xA)
    {
      pbuf[2 * idx] = (value >> 28) + '0';
    }
    else
    {
      pbuf[2 * idx] = (value >> 28) + 'A' - 10;
    }

    value = value << 4;

    pbuf[2 * idx + 1] = 0;
  }
}
/**
```

usbd_conf.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : usbd_conf.h
  * @version        : v2.0_Cube
  * @brief          : Header for usbd_conf.c file.
  ******************************************************************************
  * @attention
```

```
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                              www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __USBD_CONF__H__
#define __USBD_CONF__H__

#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "main.h"
#include "stm32f1xx.h"
#include "stm32f1xx_hal.h"

/* USER CODE BEGIN INCLUDE */

/* USER CODE END INCLUDE */

/** @addtogroup USBD_OTG_DRIVER
  * @{
  */

/** @defgroup USBD_CONF USBD_CONF
  * @brief Configuration file for Usb otg low level driver.
  * @{
  */

/** @defgroup USBD_CONF_Exported_Variables USBD_CONF_Exported_Variables
  * @brief Public variables.
  * @{
  */

/**
  * @}
  */

/** @defgroup USBD_CONF_Exported_Defines USBD_CONF_Exported_Defines
  * @brief Defines for configuration of the Usb device.
  * @{
  */

/*---------- -----------*/
#define USBD_MAX_NUM_INTERFACES     1
/*---------- -----------*/
#define USBD_MAX_NUM_CONFIGURATION     1
/*---------- -----------*/
#define USBD_MAX_STR_DESC_SIZ     512
/*---------- -----------*/
#define USBD_DEBUG_LEVEL     0
/*---------- -----------*/
#define USBD_SELF_POWERED     1
/*---------- -----------*/
```

```
#define MAX_STATIC_ALLOC_SIZE     512

/***************************************/
/* #define for FS and HS identification */
#define DEVICE_FS         0

/**
  * @}
  */

/** @defgroup USBD_CONF_Exported_Macros USBD_CONF_Exported_Macros
  * @brief Aliases.
  * @{
  */

/* Memory management macros */

/** Alias for memory allocation. */
#define USBD_malloc         (uint32_t *)USBD_static_malloc

/** Alias for memory release. */
#define USBD_free           USBD_static_free

/** Alias for memory set. */
#define USBD_memset         /* Not used */

/** Alias for memory copy. */
#define USBD_memcpy         /* Not used */

/** Alias for delay. */
#define USBD_Delay          HAL_Delay

/* For footprint reasons and since only one allocation is handled in the HID class
   driver, the malloc/free is changed into a static allocation method */
void *USBD_static_malloc(uint32_t size);
void USBD_static_free(void *p);

/* DEBUG macros */

#if (USBD_DEBUG_LEVEL > 0)
#define USBD_UsrLog(...)    printf(__VA_ARGS__);\
                            printf("\n");

#else
#define USBD_UsrLog(...)
#endif

#if (USBD_DEBUG_LEVEL > 1)

#define USBD_ErrLog(...)    printf("ERROR: ") ;\
                            printf(__VA_ARGS__);\
                            printf("\n");
#else
#define USBD_ErrLog(...)
#endif

#if (USBD_DEBUG_LEVEL > 2)
#define USBD_DbgLog(...)    printf("DEBUG : ") ;\
                            printf(__VA_ARGS__);\
                            printf("\n");
#else
#define USBD_DbgLog(...)
#endif

/**
  * @}
  */
```

```
/** @defgroup USBD_CONF_Exported_Types USBD_CONF_Exported_Types
  * @brief Types.
  * @{
  */

/**
  * @}
  */

/** @defgroup USBD_CONF_Exported_FunctionsPrototype
USBD_CONF_Exported_FunctionsPrototype
  * @brief Declaration of public functions for Usb device.
  * @{
  */

/* Exported functions ------------------------------------------------------*/


#ifdef __cplusplus
}
#endif

#endif /* __USBD_CONF__H__ */
```

usbd_conf.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : Target/usbd_conf.c
  * @version        : v2.0_Cube
  * @brief          : This file implements the board support package for the USB device
library
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "stm32f1xx.h"
#include "stm32f1xx_hal.h"
#include "usbd_def.h"
#include "usbd_core.h"
#include "usbd_cdc.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
```

```
/* USER CODE END PV */

PCD_HandleTypeDef hpcd_USB_FS;
void Error_Handler(void);

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */

/* Private functions ---------------------------------------------------------*/
static USBD_StatusTypeDef USBD_Get_USB_Status(HAL_StatusTypeDef hal_status);
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCDEx_SetConnectionState(PCD_HandleTypeDef *hpcd, uint8_t state);
else
void HAL_PCDEx_SetConnectionState(PCD_HandleTypeDef *hpcd, uint8_t state);
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */

/*******************************************************************************
                       LL Driver Callbacks (PCD -> USB Device Library)
*******************************************************************************/
/* MSP Init */

void HAL_PCD_MspInit(PCD_HandleTypeDef* pcdHandle)
{
  if(pcdHandle->Instance==USB)
  {
  /* USER CODE BEGIN USB_MspInit 0 */

  /* USER CODE END USB_MspInit 0 */
    /* Peripheral clock enable */
    __HAL_RCC_USB_CLK_ENABLE();

    /* Peripheral interrupt init */
    HAL_NVIC_SetPriority(USB_LP_CAN1_RX0_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(USB_LP_CAN1_RX0_IRQn);
  /* USER CODE BEGIN USB_MspInit 1 */

  /* USER CODE END USB_MspInit 1 */
  }
}

void HAL_PCD_MspDeInit(PCD_HandleTypeDef* pcdHandle)
{
  if(pcdHandle->Instance==USB)
  {
  /* USER CODE BEGIN USB_MspDeInit 0 */

  /* USER CODE END USB_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_USB_CLK_DISABLE();

    /* Peripheral interrupt Deinit*/
    HAL_NVIC_DisableIRQ(USB_LP_CAN1_RX0_IRQn);

  /* USER CODE BEGIN USB_MspDeInit 1 */

  /* USER CODE END USB_MspDeInit 1 */
  }
}
```

```
/**
  * @brief  Setup stage callback
  * @param  hpcd: PCD handle
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCD_SetupStageCallback(PCD_HandleTypeDef *hpcd)
#else
void HAL_PCD_SetupStageCallback(PCD_HandleTypeDef *hpcd)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  USBD_LL_SetupStage((USBD_HandleTypeDef*)hpcd->pData, (uint8_t *)hpcd->Setup);
}

/**
  * @brief  Data Out stage callback.
  * @param  hpcd: PCD handle
  * @param  epnum: Endpoint number
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCD_DataOutStageCallback(PCD_HandleTypeDef *hpcd, uint8_t epnum)
#else
void HAL_PCD_DataOutStageCallback(PCD_HandleTypeDef *hpcd, uint8_t epnum)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  USBD_LL_DataOutStage((USBD_HandleTypeDef*)hpcd->pData, epnum, hpcd-
>OUT_ep[epnum].xfer_buff);
}

/**
  * @brief  Data In stage callback.
  * @param  hpcd: PCD handle
  * @param  epnum: Endpoint number
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCD_DataInStageCallback(PCD_HandleTypeDef *hpcd, uint8_t epnum)
#else
void HAL_PCD_DataInStageCallback(PCD_HandleTypeDef *hpcd, uint8_t epnum)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  USBD_LL_DataInStage((USBD_HandleTypeDef*)hpcd->pData, epnum, hpcd-
>IN_ep[epnum].xfer_buff);
}

/**
  * @brief  SOF callback.
  * @param  hpcd: PCD handle
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCD_SOFCallback(PCD_HandleTypeDef *hpcd)
#else
void HAL_PCD_SOFCallback(PCD_HandleTypeDef *hpcd)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  USBD_LL_SOF((USBD_HandleTypeDef*)hpcd->pData);
}

/**
  * @brief  Reset callback.
  * @param  hpcd: PCD handle
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
```

```
static void PCD_ResetCallback(PCD_HandleTypeDef *hpcd)
#else
void HAL_PCD_ResetCallback(PCD_HandleTypeDef *hpcd)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  USBD_SpeedTypeDef speed = USBD_SPEED_FULL;

  if ( hpcd->Init.speed != PCD_SPEED_FULL)
  {
    Error_Handler();
  }
    /* Set Speed. */
  USBD_LL_SetSpeed((USBD_HandleTypeDef*)hpcd->pData, speed);

  /* Reset Device. */
  USBD_LL_Reset((USBD_HandleTypeDef*)hpcd->pData);
}

/**
  * @brief  Suspend callback.
  * When Low power mode is enabled the debug cannot be used (IAR, Keil doesn't support
it)
  * @param  hpcd: PCD handle
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCD_SuspendCallback(PCD_HandleTypeDef *hpcd)
#else
void HAL_PCD_SuspendCallback(PCD_HandleTypeDef *hpcd)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  /* Inform USB library that core enters in suspend Mode. */
  USBD_LL_Suspend((USBD_HandleTypeDef*)hpcd->pData);
  /* Enter in STOP mode. */
  /* USER CODE BEGIN 2 */
  if (hpcd->Init.low_power_enable)
  {
    /* Set SLEEPDEEP bit and SleepOnExit of Cortex System Control Register. */
    SCB->SCR |= (uint32_t)((uint32_t)(SCB_SCR_SLEEPDEEP_Msk |
SCB_SCR_SLEEPONEXIT_Msk));
  }
  /* USER CODE END 2 */
}

/**
  * @brief  Resume callback.
  * When Low power mode is enabled the debug cannot be used (IAR, Keil doesn't support
it)
  * @param  hpcd: PCD handle
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCD_ResumeCallback(PCD_HandleTypeDef *hpcd)
#else
void HAL_PCD_ResumeCallback(PCD_HandleTypeDef *hpcd)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  /* USER CODE BEGIN 3 */

  /* USER CODE END 3 */
  USBD_LL_Resume((USBD_HandleTypeDef*)hpcd->pData);
}

/**
  * @brief  ISOOUTIncomplete callback.
  * @param  hpcd: PCD handle
  * @param  epnum: Endpoint number
```

```
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCD_ISOOUTIncompleteCallback(PCD_HandleTypeDef *hpcd, uint8_t epnum)
#else
void HAL_PCD_ISOOUTIncompleteCallback(PCD_HandleTypeDef *hpcd, uint8_t epnum)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  USBD_LL_IsoOUTIncomplete((USBD_HandleTypeDef*)hpcd->pData, epnum);
}

/**
  * @brief  ISOINIncomplete callback.
  * @param  hpcd: PCD handle
  * @param  epnum: Endpoint number
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCD_ISOINIncompleteCallback(PCD_HandleTypeDef *hpcd, uint8_t epnum)
#else
void HAL_PCD_ISOINIncompleteCallback(PCD_HandleTypeDef *hpcd, uint8_t epnum)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  USBD_LL_IsoINIncomplete((USBD_HandleTypeDef*)hpcd->pData, epnum);
}

/**
  * @brief  Connect callback.
  * @param  hpcd: PCD handle
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCD_ConnectCallback(PCD_HandleTypeDef *hpcd)
#else
void HAL_PCD_ConnectCallback(PCD_HandleTypeDef *hpcd)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  USBD_LL_DevConnected((USBD_HandleTypeDef*)hpcd->pData);
}

/**
  * @brief  Disconnect callback.
  * @param  hpcd: PCD handle
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCD_DisconnectCallback(PCD_HandleTypeDef *hpcd)
#else
void HAL_PCD_DisconnectCallback(PCD_HandleTypeDef *hpcd)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  USBD_LL_DevDisconnected((USBD_HandleTypeDef*)hpcd->pData);
}

/*******************************************************************************
                       LL Driver Interface (USB Device Library --> PCD)
*******************************************************************************/

/**
  * @brief  Initializes the low level portion of the device driver.
  * @param  pdev: Device handle
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_Init(USBD_HandleTypeDef *pdev)
{
  /* Init USB Ip. */
  /* Link the driver to the stack. */
```

```
  hpcd_USB_FS.pData = pdev;
  pdev->pData = &hpcd_USB_FS;

  hpcd_USB_FS.Instance = USB;
  hpcd_USB_FS.Init.dev_endpoints = 8;
  hpcd_USB_FS.Init.speed = PCD_SPEED_FULL;
  hpcd_USB_FS.Init.low_power_enable = DISABLE;
  hpcd_USB_FS.Init.lpm_enable = DISABLE;
  hpcd_USB_FS.Init.battery_charging_enable = DISABLE;
  if (HAL_PCD_Init(&hpcd_USB_FS) != HAL_OK)
  {
    Error_Handler( );
  }

#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
  /* Register USB PCD CallBacks */
  HAL_PCD_RegisterCallback(&hpcd_USB_FS, HAL_PCD_SOF_CB_ID, PCD_SOFCallback);
  HAL_PCD_RegisterCallback(&hpcd_USB_FS, HAL_PCD_SETUPSTAGE_CB_ID,
PCD_SetupStageCallback);
  HAL_PCD_RegisterCallback(&hpcd_USB_FS, HAL_PCD_RESET_CB_ID, PCD_ResetCallback);
  HAL_PCD_RegisterCallback(&hpcd_USB_FS, HAL_PCD_SUSPEND_CB_ID, PCD_SuspendCallback);
  HAL_PCD_RegisterCallback(&hpcd_USB_FS, HAL_PCD_RESUME_CB_ID, PCD_ResumeCallback);
  HAL_PCD_RegisterCallback(&hpcd_USB_FS, HAL_PCD_CONNECT_CB_ID, PCD_ConnectCallback);
  HAL_PCD_RegisterCallback(&hpcd_USB_FS, HAL_PCD_DISCONNECT_CB_ID,
PCD_DisconnectCallback);

  HAL_PCD_RegisterDataOutStageCallback(&hpcd_USB_FS, PCD_DataOutStageCallback);
  HAL_PCD_RegisterDataInStageCallback(&hpcd_USB_FS, PCD_DataInStageCallback);
  HAL_PCD_RegisterIsoOutIncpltCallback(&hpcd_USB_FS, PCD_ISOOUTIncompleteCallback);
  HAL_PCD_RegisterIsoInIncpltCallback(&hpcd_USB_FS, PCD_ISOINIncompleteCallback);
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
  /* USER CODE BEGIN EndPoint_Configuration */
  HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , 0x00 , PCD_SNG_BUF, 0x18);
  HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , 0x80 , PCD_SNG_BUF, 0x58);
  /* USER CODE END EndPoint_Configuration */
  /* USER CODE BEGIN EndPoint_Configuration_CDC */
  HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , 0x81 , PCD_SNG_BUF, 0xC0);
  HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , 0x01 , PCD_SNG_BUF, 0x110);
  HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , 0x82 , PCD_SNG_BUF, 0x100);
  /* USER CODE END EndPoint_Configuration_CDC */
  return USBD_OK;
}

/**
  * @brief  De-Initializes the low level portion of the device driver.
  * @param  pdev: Device handle
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_DeInit(USBD_HandleTypeDef *pdev)
{
  HAL_StatusTypeDef hal_status = HAL_OK;
  USBD_StatusTypeDef usb_status = USBD_OK;

  hal_status = HAL_PCD_DeInit(pdev->pData);

  usb_status =  USBD_Get_USB_Status(hal_status);

  return usb_status;
}

/**
  * @brief  Starts the low level portion of the device driver.
  * @param  pdev: Device handle
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_Start(USBD_HandleTypeDef *pdev)
{
```

```
  HAL_StatusTypeDef hal_status = HAL_OK;
  USBD_StatusTypeDef usb_status = USBD_OK;

  hal_status = HAL_PCD_Start(pdev->pData);

  usb_status =  USBD_Get_USB_Status(hal_status);

  return usb_status;
}

/**
  * @brief  Stops the low level portion of the device driver.
  * @param  pdev: Device handle
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_Stop(USBD_HandleTypeDef *pdev)
{
  HAL_StatusTypeDef hal_status = HAL_OK;
  USBD_StatusTypeDef usb_status = USBD_OK;

  hal_status = HAL_PCD_Stop(pdev->pData);

  usb_status =  USBD_Get_USB_Status(hal_status);

  return usb_status;
}

/**
  * @brief  Opens an endpoint of the low level driver.
  * @param  pdev: Device handle
  * @param  ep_addr: Endpoint number
  * @param  ep_type: Endpoint type
  * @param  ep_mps: Endpoint max packet size
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_OpenEP(USBD_HandleTypeDef *pdev, uint8_t ep_addr, uint8_t
ep_type, uint16_t ep_mps)
{
  HAL_StatusTypeDef hal_status = HAL_OK;
  USBD_StatusTypeDef usb_status = USBD_OK;

  hal_status = HAL_PCD_EP_Open(pdev->pData, ep_addr, ep_mps, ep_type);

  usb_status =  USBD_Get_USB_Status(hal_status);

  return usb_status;
}

/**
  * @brief  Closes an endpoint of the low level driver.
  * @param  pdev: Device handle
  * @param  ep_addr: Endpoint number
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_CloseEP(USBD_HandleTypeDef *pdev, uint8_t ep_addr)
{
  HAL_StatusTypeDef hal_status = HAL_OK;
  USBD_StatusTypeDef usb_status = USBD_OK;

  hal_status = HAL_PCD_EP_Close(pdev->pData, ep_addr);

  usb_status =  USBD_Get_USB_Status(hal_status);

  return usb_status;
}

/**
```

```
  * @brief  Flushes an endpoint of the Low Level Driver.
  * @param  pdev: Device handle
  * @param  ep_addr: Endpoint number
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_FlushEP(USBD_HandleTypeDef *pdev, uint8_t ep_addr)
{
  HAL_StatusTypeDef hal_status = HAL_OK;
  USBD_StatusTypeDef usb_status = USBD_OK;

  hal_status = HAL_PCD_EP_Flush(pdev->pData, ep_addr);

  usb_status =  USBD_Get_USB_Status(hal_status);

  return usb_status;
}

/**
  * @brief  Sets a Stall condition on an endpoint of the Low Level Driver.
  * @param  pdev: Device handle
  * @param  ep_addr: Endpoint number
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_StallEP(USBD_HandleTypeDef *pdev, uint8_t ep_addr)
{
  HAL_StatusTypeDef hal_status = HAL_OK;
  USBD_StatusTypeDef usb_status = USBD_OK;

  hal_status = HAL_PCD_EP_SetStall(pdev->pData, ep_addr);

  usb_status =  USBD_Get_USB_Status(hal_status);

  return usb_status;
}

/**
  * @brief  Clears a Stall condition on an endpoint of the Low Level Driver.
  * @param  pdev: Device handle
  * @param  ep_addr: Endpoint number
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_ClearStallEP(USBD_HandleTypeDef *pdev, uint8_t ep_addr)
{
  HAL_StatusTypeDef hal_status = HAL_OK;
  USBD_StatusTypeDef usb_status = USBD_OK;

  hal_status = HAL_PCD_EP_ClrStall(pdev->pData, ep_addr);

  usb_status =  USBD_Get_USB_Status(hal_status);

  return usb_status;
}

/**
  * @brief  Returns Stall condition.
  * @param  pdev: Device handle
  * @param  ep_addr: Endpoint number
  * @retval Stall (1: Yes, 0: No)
  */
uint8_t USBD_LL_IsStallEP(USBD_HandleTypeDef *pdev, uint8_t ep_addr)
{
  PCD_HandleTypeDef *hpcd = (PCD_HandleTypeDef*) pdev->pData;

  if((ep_addr & 0x80) == 0x80)
  {
    return hpcd->IN_ep[ep_addr & 0x7F].is_stall;
  }
```

```
  else
  {
    return hpcd->OUT_ep[ep_addr & 0x7F].is_stall;
  }
}

/**
  * @brief  Assigns a USB address to the device.
  * @param  pdev: Device handle
  * @param  dev_addr: Device address
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_SetUSBAddress(USBD_HandleTypeDef *pdev, uint8_t dev_addr)
{
  HAL_StatusTypeDef hal_status = HAL_OK;
  USBD_StatusTypeDef usb_status = USBD_OK;

  hal_status = HAL_PCD_SetAddress(pdev->pData, dev_addr);

  usb_status =  USBD_Get_USB_Status(hal_status);

  return usb_status;
}

/**
  * @brief  Transmits data over an endpoint.
  * @param  pdev: Device handle
  * @param  ep_addr: Endpoint number
  * @param  pbuf: Pointer to data to be sent
  * @param  size: Data size
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_Transmit(USBD_HandleTypeDef *pdev, uint8_t ep_addr, uint8_t
*pbuf, uint16_t size)
{
  HAL_StatusTypeDef hal_status = HAL_OK;
  USBD_StatusTypeDef usb_status = USBD_OK;

  hal_status = HAL_PCD_EP_Transmit(pdev->pData, ep_addr, pbuf, size);

  usb_status =  USBD_Get_USB_Status(hal_status);

  return usb_status;
}

/**
  * @brief  Prepares an endpoint for reception.
  * @param  pdev: Device handle
  * @param  ep_addr: Endpoint number
  * @param  pbuf: Pointer to data to be received
  * @param  size: Data size
  * @retval USBD status
  */
USBD_StatusTypeDef USBD_LL_PrepareReceive(USBD_HandleTypeDef *pdev, uint8_t ep_addr,
uint8_t *pbuf, uint16_t size)
{
  HAL_StatusTypeDef hal_status = HAL_OK;
  USBD_StatusTypeDef usb_status = USBD_OK;

  hal_status = HAL_PCD_EP_Receive(pdev->pData, ep_addr, pbuf, size);

  usb_status =  USBD_Get_USB_Status(hal_status);

  return usb_status;
}

/**
```

```
  * @brief  Returns the last transfered packet size.
  * @param  pdev: Device handle
  * @param  ep_addr: Endpoint number
  * @retval Recived Data Size
  */
uint32_t USBD_LL_GetRxDataSize(USBD_HandleTypeDef *pdev, uint8_t ep_addr)
{
  return HAL_PCD_EP_GetRxCount((PCD_HandleTypeDef*) pdev->pData, ep_addr);
}

/**
  * @brief  Delays routine for the USB device library.
  * @param  Delay: Delay in ms
  * @retval None
  */
void USBD_LL_Delay(uint32_t Delay)
{
  HAL_Delay(Delay);
}

/**
  * @brief  Static single allocation.
  * @param  size: Size of allocated memory
  * @retval None
  */
void *USBD_static_malloc(uint32_t size)
{
  static uint32_t mem[(sizeof(USBD_CDC_HandleTypeDef)/4)+1];/* On 32-bit boundary */
  return mem;
}

/**
  * @brief  Dummy memory free
  * @param  p: Pointer to allocated  memory address
  * @retval None
  */
void USBD_static_free(void *p)
{

}

/**
  * @brief Software Device Connection
  * @param hpcd: PCD handle
  * @param state: Connection state (0: disconnected / 1: connected)
  * @retval None
  */
#if (USE_HAL_PCD_REGISTER_CALLBACKS == 1U)
static void PCDEx_SetConnectionState(PCD_HandleTypeDef *hpcd, uint8_t state)
#else
void HAL_PCDEx_SetConnectionState(PCD_HandleTypeDef *hpcd, uint8_t state)
#endif /* USE_HAL_PCD_REGISTER_CALLBACKS */
{
  /* USER CODE BEGIN 6 */
  if (state == 1)
  {
    /* Configure Low connection state. */

  }
  else
  {
    /* Configure High connection state. */

  }
  /* USER CODE END 6 */
}
```

```
/**
  * @brief  Retuns the USB status depending on the HAL status:
  * @param  hal_status: HAL status
  * @retval USB status
  */
USBD_StatusTypeDef USBD_Get_USB_Status(HAL_StatusTypeDef hal_status)
{
  USBD_StatusTypeDef usb_status = USBD_OK;

  switch (hal_status)
  {
    case HAL_OK :
      usb_status = USBD_OK;
    break;
    case HAL_ERROR :
      usb_status = USBD_FAIL;
    break;
    case HAL_BUSY :
      usb_status = USBD_BUSY;
    break;
    case HAL_TIMEOUT :
      usb_status = USBD_FAIL;
    break;
    default :
      usb_status = USBD_FAIL;
    break;
  }
  return usb_status;
}
```