НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050201 «Системна інженерія»

на тему: «Система передачі даних з використанням мережі стільникового зв'язку»

Виконав:
студент IV курсу, групи ІА-51
Міщенко Дмитро Максимович _____

Керівник:
Старший викладач кафедри АУТС
Шимкович В.М.

_____

Рецензент:
Доцент кафедри ТК, кандидат технічних наук,
доцент, Тимошин Ю.А. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 рік

Пояснювальна записка
до дипломного проекту
на тему:
«Система передачі даних з використанням мережі стільникового зв’язку»

Київ – 2019 рік

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UTP – Unshielded Twisted Pair

SFP – Small Form-factor Pluggable

МК – медіаконвертер

3G – 3rd Generation

USB – Universal Serial Bus

I2C – Inter-Integrated Circuit

SPI – Serial Peripheral Interface

UART – Universal Asynchronous Receiver/Transmitter

IDE – Integrated Development Environment

ЦАП – цифро-аналоговий перетворювач

АЦП – аналого-цифровий перетворювач

DMA – Direct Memory Access

ВСТУП

Мережеві технології розвиваються стрімкими темпами. Цей розвиток обумовлено вимогами сучасного світу, у якому кожна людина хоче мати доступ до мережі Інтернет[1] з будь-якого місця та у будь-який час. Кількість пристроїв та їх видів, що можуть бути підключені до всесвітньої мережі збільшується з кожним днем.

Великі міста обслуговують десятки-сотні Інтернет провайдерів. Це можуть бути провайдери, що надають послуги підключення до мережі Інтернет як для домашнього користування фізичними особами, так і для підключень різного роду підприємств. Також існують й такі, що не займаються вищесказаним, а тільки надають послуги іншим провайдерам. Але всі вони змушені йти у ногу з часом і розширювати власні мережі або шляхом укладення угод та використання ресурсів колег, або шляхом модернізації власного устаткування.

Який спосіб росту не вибирали б провайдери, перед кожним з них постають декілька важливих аспектів побудови власної мережі. Одними з найважливіших є надійність, відмовостійкість обладнання та можливість швидкого реагування у разі його поломки.

При виявленні несправності мережевого обладнання виникає необхідність швидкого реагування на дану проблему, адже будь-яка затримка може спричинити втрату доступу до мережі Інтернет клієнтами. У найпростіших випадках, доступ втратять рядові користувачі і це не принесе великих збитків, але може виникнути ситуація, при якій, із-за несправності, підключення втратить велике підприємство.

Одним з найважливіших етапів реагування на проблеми з обладнанням є його своєчасна і правильна первинна діагностика. Від неї залежить хід подальшого діагностування. Якщо можливості провести первинну діагностику немає або ж вона дасть хибний результат, то це може спричинити значну затримку у вирішенні проблеми.

Чим складнішу структуру має мережа, тим більше з’являється потенціальних джерел поломок.

Проведенням первинної діагностики, зазвичай, займаються інженери технічної підтримки. У них є певний набір інструментів, за допомогою яких вони проводять збір та аналіз інформації про несправність. Маючи досвід роботи на вищезгаданій посаді, я знаю, що цих інструментів часто не вистачає для правильного та своєчасно реагування на проблему.

Ціллю цієї роботи є розробка пристрою, що допоможе при первинному діагностуванні поломок мережевого обладнання. До даного пристрою висувається ряд основних вимог.

По-перше, пристрій повинен бути дешевим, адже кількість важливого обладнання, що може вийти з ладу у мережі одного провайдера, може бути величиною великою і затрати на резервування зв’язку з кожним таким елементом потрібно мінімізувати.

По-друге, пристрій повинен мати свій канал зв’язку. Цей канал має бути доступним та дешевим.

По третє, пристрій має бути простий у використанні та установці.

1 ОПИС ПРЕДМЕТРОЇ ОБЛАСТІ

1.1 Огляд та аналіз основних схем включення мережевого обладнання

Розроблюваний пристрій планується використовувати як додатковий канал зв’язку з мережевим обладнанням, зв’язок з яким було втрачено. Втрата зв’язку може статися з багатьох причин. Причини, частіше за все, залежать від самого обладнання та від пристроїв його підключення. Проте інколи від обладнання нічого не залежить, так як причиною його недоступності може слугувати банальні перебої з електроживленням.

При недоступності мережевого обладнання важливо знати, яким способом та за допомогою яких пристроїв воно підключене. Це значно спростить задачу первинної діагностики, адже діагностуючий вже може допустити де може бути джерело несправності.

Існує досить багато схем підключення мережевого обладнання. Далі ми розглянемо найбільш використовувані з них у сучасних мережах провайдерів.

1.1.1 Підключення за допомогою UTP

Підключення за допомогою UTP – найпростіший спосіб з’єднати два мережевих пристрої. Найчастіше використовується UTP п’ятої категорії. Даний метод підключення є найпоширенішим, оскільки звита пара дешева та проста у підключенні.

UTP має 8 провідників, котрі скручені в пари. В залежності від необхідної швидкості, задіяними можуть бути від чотирьох до восьми провідників. При втраті зв’язку з мережевим обладнанням, котре з’єднане шляхом UTP, перш за все варто перевірити саму звиту пару. Це можна зробити вручну. Але деяке мережеве обладнання здатне саме перевіряти стан звитої пари. Для цього необхідно виконати спеціальну команду у командному рядку мережевого пристрою. Однак для цього

потрібно спочатку отримати до нього доступ. Якщо обладнання знаходиться у важкодоступному місці, то потрібен резервний канал доступу.

Рис. 1.1 З’єднання двох комутаторів за допомогою UTP

На рис. 1.1 наглядно бачимо, як з’єднують мережеві пристрої за допомогою UTP. Також з ладу може вийти і порт включення. Це теж можна перевірити, підключившись до пристрою.

1.1.2 Підключення за допомогою SFP

Підключення мережевих пристроїв через SFP-модулі – один із найпоширеніших видів комутації з використанням оптоволоконного кабелю [2]. Оптичне волокно не має одного з основних недоліків, котрий є у звитої пари – опору. Тому дальність передачі цілком залежить від потужності випромінювачів, що розташовані на кінцях світловоду та затухання світла. Останнє має місце при будь-яких дефектах світловоду та при його згинанні.

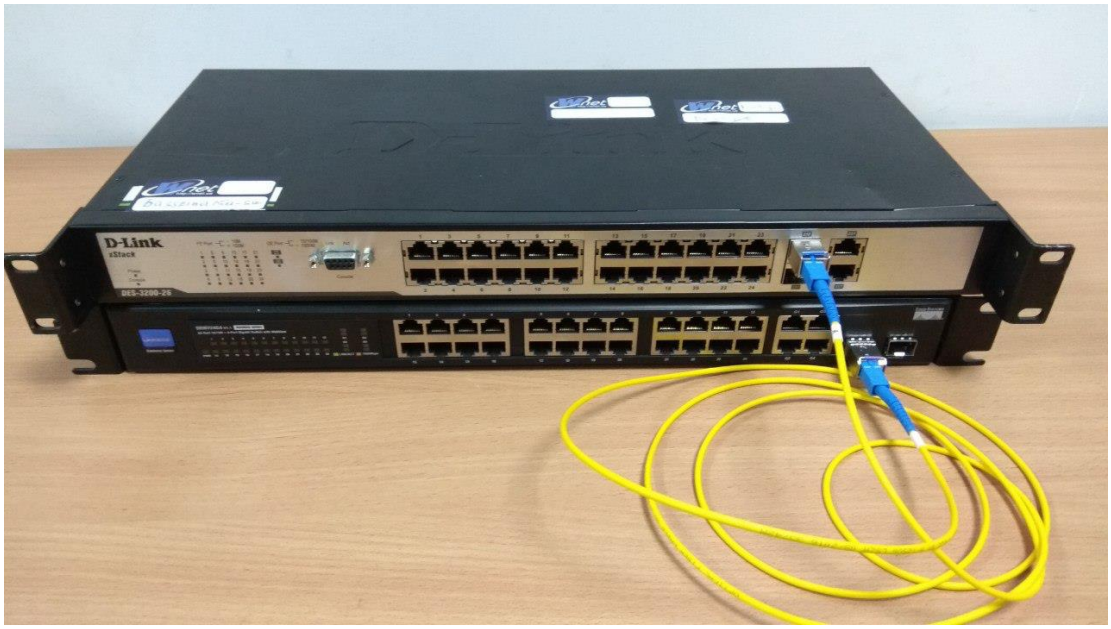

Рис. 1.2 З’єднання двох комутаторів за допомогою SFP та оптичного волокна

У такій системі вийти з ладу може не тільки оптичне волокно (часто воно страждає від вандалізму), а й самі SFP модулі та порти, до яких модулі підключені. Знаючи, що недоступне обладнання підключене саме за такою схемою, можна провести повноцінну діагностику, якщо є доступ до обох точок включення. Достатньо підключитися до них та подивитися на тип порту та його стан.

1.1.3 Підключення за допомогою медіаконвертерів

Медіаконвертер являє собою пристрій, за допомогою якого відбувається зміна середовища передачі даних. Це потрібно у випадках, коли, наприклад, відстань від з’єднуваних пристроїв більша ніж допустима дальність для UTP або немає портів під модулі SFP, що описані вище. Також це вигідно і з економічної точки зору, оскільки сумарна вартість мережевого пристрою з медіаконвертером і вартість аналогічного пристрою, але з можливістю встановлення SFP, різна. В першому випадку, зазвичай, вартість менша.

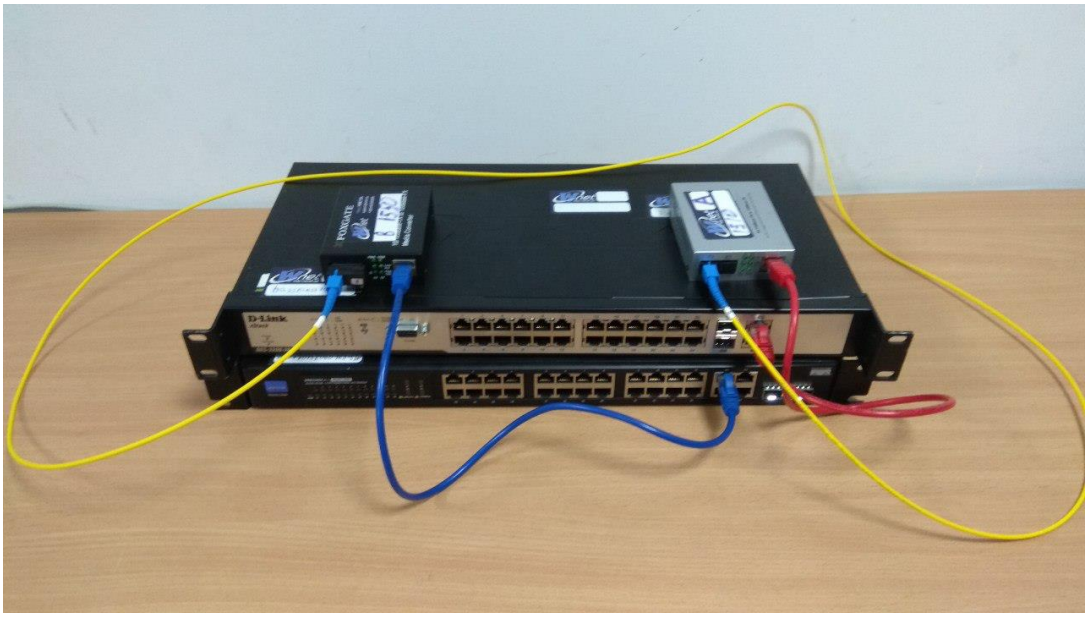

Рис. 1.3 З’єднання двох комутаторів за допомогою медіаконвертерів та оптичного волокна

Основними, вразливими до поломок, точками при даній схемі підключення є медіаконвертери. Також не варто забувати і про їх блоки живлення, які теж можуть виходити з ладу. Джерелом недоступності може також стати розрив оптичного волокна.

При розриві оптичного каналу зв’язку, порт включення медіаконвертора на мережевому обладнанні не переходить у стан «down», залишаючись у стані «up». Якщо не працює сам медіаконвертер, з будь-якої причини, то порт його включення буде у стані «down». Все це можна перевірити, якщо є доступ до обладнання.

Також не варто забувати, що є медіаконвертери, у яких замість оптичного каналу є місце для SFP модуля. Потенціальних місць для виникнення несправності в них більше, але діагностика та ж сама. Підвидом такої схеми включення є гібридна схема з використанням з одної сторони SFP модуля, увімкненого в, наприклад, комутатор, а з іншої МК з SFP.

Такі пристрої зображено на рис. 1.4.

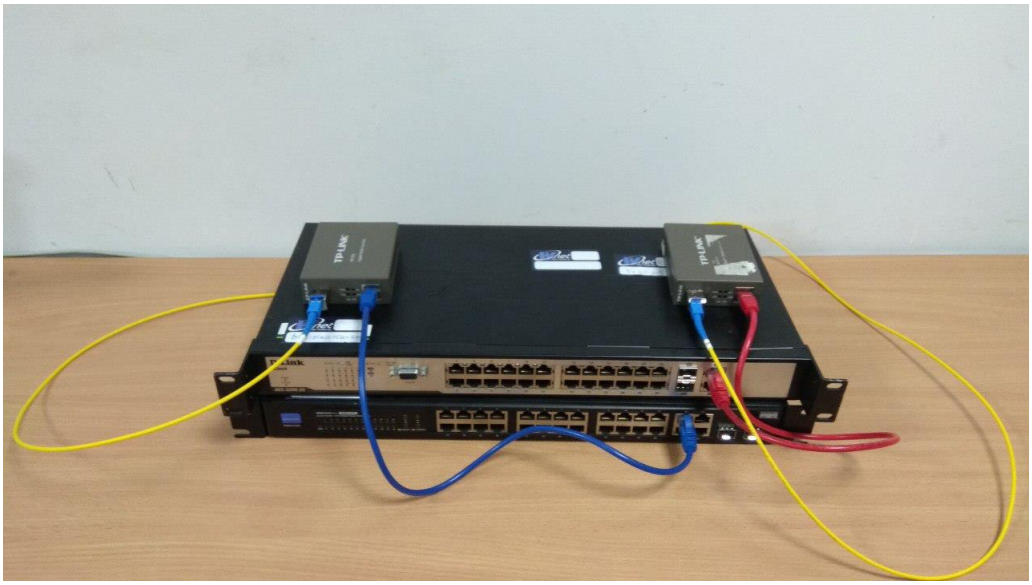

Рис. 1.4 З’єднання двох комутаторів за допомогою медіаконвертерів з SFP та оптичного волокна

У переважної більшості МК є функція (частіш за все виконана у вигляді фізичного перемикача на його корпусі), що викликає перехід порту включення МК на обладнанні у стан «down» при обриві оптичного волокна. Слід пам’ятати про це при проведенні діагностики та подальших дій.

1.2 Огляд існуючих рішень

Для доступу до мережевого обладнання, при втраті зв’язку по основному каналу, будують резервні канали зв’язку. Вони можуть слугувати для різних цілей. Одні, наприклад, можуть повноцінно використовуватись як заміна основного каналу, тобто мають такі ж або схожі з основним характеристики. Інші слугують тільки для доступу до обладнання, з яким було втрачено зв’язок для його діагностики або конфігурації. Існує досить багато способів побудувати такі канали. Вони відрізняються один від одного складністю, ціною, швидкістю встановлення та рядом інших характеристик, що можуть бути визначаючими при виборі резервного каналу. Розглянемо основні, найбільш використовувані способи. Це резервування за допомогою 3G модему, резервування за допомогою додаткового

підключення з іншого обладнання та резервування за допомогою супутникового зв’язку.

1.2.1 Резервування за допомогою 3G модему

3G модем може бути встановлено як в мережеве обладнання, так і в обладнання кінцеве. Він не потребує додаткового живлення адже, найчастіше, підключається за допомогою USB інтерфейсу.

Рис. 1.5 3G модем від «Київстар», встановлений в маршрутизатор

Даний спосіб резервування має досить високу доступність, адже кількість провайдерів, що надають подібні послуги, достатня. Наданням 3G модемів у оренду займаються всі великі оператори мобільного зв’язку України. За даними з Інтернету[3], покриття 3G охоплює близько 70% території України та тяжіє до населених пунктів.

Також, до переваг даного способу, можна віднести досить високу швидкість обміну даними. Якщо за цільовим обладнанням знаходиться користувачі, що генерують невелику кількість трафіку, то такий спосіб резервування цілком може впоратись з його передачею. На сьогоднішній день, швидкість передачі за допомогою 3G на території України досягає 28 Мбіт/с[4].

У даного методу є свої недоліки. Перш за все – ціна. Для встановлення модему потрібно його купити. Також, вартість трафіку у перерахунку на один мегабайт більша, ніж у провайдерів, що надають класичний доступ до мережі Інтернет. Абонплату потрібно виконувати кожного місяця в залежності або від тарифного плану, і/або від кількості використаного трафіку.

Також, через особливості реалізації обміну даними, сила сигналу часто виявляється недостатньою у підвальних приміщеннях або приміщеннях з товстими стінами. Для вирішення подібного роду проблем, рекомендують встановлювати антени, що підключаються до модему. Це, у свою чергу, додаткові фінансові витрати і складність при підключенні.

Варто відмітити, що хоч даний спосіб і має порівняно високу швидкість, але затримки, порівняно з «наземним» каналом значно більші, що може негативно вплинути на комфорт роботи.

До недоліків теж можна віднести і залежність модему від живлення всього обладнання, адже переважна більшість модемів за основний інтерфейс має USB.

1.2.2 Резервування за допомогою додаткового підключення з іншого обладнання

Мережеве обладнання можна підключити з одного, а з декількох різних пристроїв. Гарною ілюстрацією такого способу є топологія «кільце» [5]. При недоступності по одному каналу, обмін даними піде по іншому. Однак, це не захистить від, наприклад, ряду системних збоїв, при яких доступ до обладнання можливий тільки через COM-порт. Також, якщо обидва канали побудовані по схемі з використанням МК, то частковому при знеструмленні, обидва МК можуть перестати працювати.

Даний метод є досить дорогим, оскільки потрібно витратитись на проведення додаткової лінії зв’язку, хоча це й компенсується повноцінним резервним каналом та швидкістю обміну.

1.2.3 Резервування за допомогою супутникового зв’язку

Супутниковий зв’язок – один із найдорожчих методів отримання доступу до мережі. Для його реалізації потрібно придбати комплект специфічного обладнання. Також залежний від електроживлення основного обладнання, хоча деякі моделі мають вбудований акумулятор.

Швидкість обміну даними не велика, оскільки базові станції знаходяться на орбіті і постійно рухаються.

Даний спосіб використовують тільки для доступу до самого мережевого обладнання.

Рис. 1.6 Комплект обладнання для супутникового зв’язку [6]

Хоч швидкість обміну даними і не висока, але підключення стабільне, оскільки супутників, що використовуються у цій технології вистачає для постійного підтримання з’єднання. Варто також звернути увагу на затримки, що обов’язково будуть виникати при резервуванні каналу у такий спосіб.

1.3 Висновки до розділу 1

У результаті виконання першого розділу дипломного проекту було описано та проаналізовано широко поширені сьогодні способи підключення мережевого обладнання. Також, проаналізовано наявні методи резервування зв’язку з обладнанням.

Кожна зі схем включення має свої особливості, переваги та недоліки. Зі збільшенням кількості елементів обладнання та його складності, зростає і кількість можливих місць для поломок. Для кожної зі схем визначено потенціальні місця виникнення неполадок та поведінку мережевого обладнання при їх появі. Варто зазначити, що це лише невелика частина із можливих видів проблем з мережевим обладнанням.

Було проаналізовано одні із найпопулярніших методів резервування зв’язку. Це резервування за допомогою 3G модему, резервування за допомогою додаткового підключення з іншого обладнання та резервування за допомогою супутникового зв’язку. Всі методи потребують відчутних фінансових затрат задля забезпечення резервування та залежать від електроживлення всієї системи. Деякі мають складнощі з первинним підключенням, оскільки потребують встановлення додаткового обладнання.

Для того, щоб віддалено провести первинну діагностику при недоступності обладнання, потрібно мати до нього доступ. Для цього слугують резервні канали зв’язку.

Резервні канали зв’язку канали можуть виконувати різні функції: одні можуть повністю заміняти основний канал, інші ж лише надають доступ до самого обладнання. В будь-якому випадку, при виникненні неполадок, резервний канал зв’язку допоможе швидше знайти та прийняти міри по ліквідації несправності. При відсутності зв’язку з обладнанням, потрібно буде надсилати інженера для діагностики безпосередньо у місце, де обладнання розташоване.

Проаналізувавши переваги та недоліки найрозповсюдженіших засобів резервування зв’язку з мережевим обладнанням, були сформовані додаткові вимоги до проектованої системи:

- низька ціна підключення та обслуговування;
- швидке та зручне встановлення додаткового обладнання;
- можливість роботи незалежно від наявності електроживлення;
- стабільність каналу зв’язку.

2 ПРОЕКТУВАННЯ СИСТЕМИ ПЕРЕДАЧІ ДАНИХ З ВИКОРИСТАННЯМ МЕРЕЖ СТІЛЬНИКОВОГО ЗВ'ЯЗКУ

2.1 Вибір середовища та способу передачі даних

Перед проектуванням системи необхідно визначитись з деякими її важливими параметрами. Ці параметри є визначаючими для її подальшої побудови. Перш за все, потрібно обрати середовище передачі даних. В залежності від середовища, розробити спосіб передачі та сформувати кадр даних.

2.1.1 Вибір середовища передачі даних

До системи були поставлені деякі вимоги, які допоможуть обрати середовище передачі даних. Проектована система має бути основана на такому способі передачі, який є загальнодоступним та не потребує суттєвих затрат на реалізацію, експлуатацію та підтримку системи. Також шукане середовище має бути досить поширеним, оскільки цільове мережеве обладнання може бути встановлене будь-де. За ціль було поставлено не досягнення великих швидкостей передачі даних, а забезпечення мінімального стабільного доступу до обладнання.

Проаналізувавши дані вимоги, було вирішено використати у якості експерименту середовище передачі звуку при виконанні телефонного дзвінку. Такий спосіб обміну даними є майже безкоштовним, оскільки тарифні плани всіх великих операторів України включають в себе безкоштовні вхідні дзвінки та підтримку номера. Звісно, досягти високої швидкості не вийде, але обмін даними з лише одним мережевим пристроєм реалізувати можна. Стабільність планується досить висока, адже оператори зв'язку України покривають мережею майже всю територію країни. За прийом та передачу сигналу будуть відповідальні звичайні телефони. До них є певні вимоги. Перша з них – наявність роз'єму для підключення гарнітури. Друга – можливість підзарядки телефону за допомогою microUSB.

2.1.2 Вибір способу передачі даних

Як вже було сказано вище, у якості прийомо-передавача буде використано пару звичайних телефонів. Але, звісно, двійкову логіку по звуковому каналу без її перетворення передати не вдасться. Буде відбуватися перетворення двійкового коду в синусоїдальний сигнал. Потрібно визначитися з модуляцією, котру буде використано для перетворення двійкового коду в сигнал, придатний для передачі по звуковому каналу.

Одним із принципів, якого було вирішено притримуватись при розробці пристрою – простота. Тому було вирішено відмовитись від складних видів модуляції і зупинитись для розгляду на базових її видах. Для реалізації складних видів модуляції можуть знадобитись додаткові модулі, що збільшить вартість пристрою.

Амплітудна модуляція – модуляція, при якій зміна амплітуди несучого сигналу відбувається по закону сигналу інформаційного. Даний вид модуляції реалізувати простіше за все. Але вона не підходить для застосування у рамках проекту, оскільки має досить погану завадостійкість. В телефонному каналі присутня адитивна похибка, що сильно спотворить сигнал при застосуванні амплітудної модуляції.

Фазова модуляція – модуляція, при якій зміна фази несучого сигналу відбувається по закону сигналу інформаційного. Даний вид модуляції має більш високий рівень завадостійкості ніж амплітудна модуляція. Але вона теж нам не підходить, оскільки для телефонного каналу притаманні короткочасні пропуски. Вони будуть провокувати виникнення помилок та втрату даних. Звісно, якість передачі залежить від потужності сигналу, але сподіватись на те, що проектований пристрій завжди буде працювати в умовах гарного сигналу, не варто.

Частотна модуляція – модуляція, при якій зміна частоти несучого сигналу відбувається по закону сигналу інформаційного. Адитивна похибка впливає при даному виді модуляції на сигнал менше. Пропуски невеликих частин звукового сигналу, притаманні телефонному каналу передачі даних, не критичні, тому що

можна підібрати такий час відтворення однієї частоти, що пропуски не зіграють великої ролі. Промодулювати сигнал досить просто.

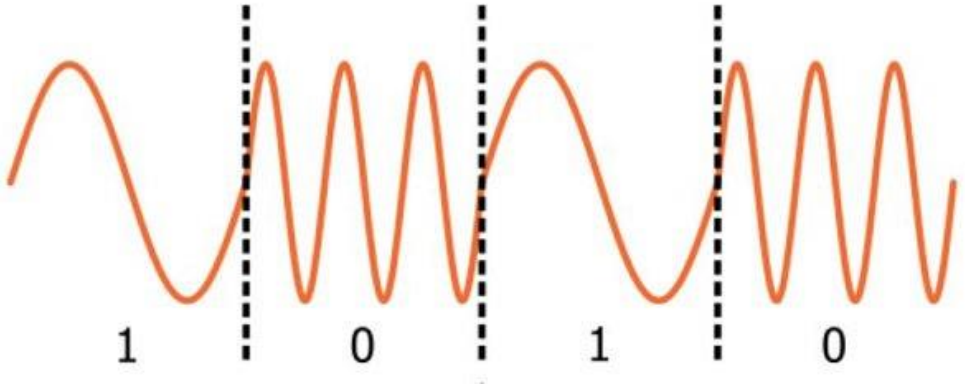

Рис. 2.1 Частотна модуляція [7]

Отже, використаємо частотну модуляцію задля формування сигналу, що буде передаватися телефонною лінією. Фазову та інші види модуляцій відкинуто через ускладнення реалізації та необхідності включити додаткові елементи до системи, що, у свою чергу, суттєво відобразиться на остаточній вартості кінцевого приладу.

2.2 Використання багаторівневого сигналу

Оскільки швидкість передачі по даному каналу не може бути високою, то застосуємо багаторівневу логіку [8]. Реальну швидкість передачі даних можна буде дізнатись на етапі тестування обладнання та відлагодження коду. Це допоможе збільшити кількість інформації, що буде передаватися за один такт.

$$R = \frac{\log_2 L}{T}, \qquad (2.1)$$

де L – число рівнів, з яких можна робити вибір в кожному такті;

T – тривалість тактового інтервалу.

З формули (2.1) бачимо, що при використанні восьмирівневої логіки, кількість символів на такт дорівнює 3. При дворівневому сигналі, кількість символів на такт буде дорівнювати одному.

Експериментальним шляхом було виявлено, що діапазон звукових частот, що може передати канал без значних спотворювань, коливається від 50 Герц до 5000 Герц. Виберемо початкову частоту в 3000 Герц і будемо визначати кожен наступний рівень з кроком 250 Герц для мінімізації похибок. Порахуємо та занесемо отриманий результат до таблиці 2.1.

Таблиця 2.1. Відповідність частоти кодовій комбінації

Частота, Герц	Кодова комбінація
3000	000
3250	001
3500	010
3750	011
4000	100
4250	101
4500	110
4750	111

У процесі розробки програмної частини реалізації проекту, частоти можна буде змінити при необхідності.

2.3 Формат кадру для передачі даних

Необхідно розробити власний формат кадру, що буде використовуватись для обміну даними. Для цього додамо до вже закодованих частотами восьми комбінацій ще одну, котра буде виконувати роль службової і передаватися частотою 2000 Герц.

Передаватимемо дані по одному байту, оскільки у майбутньому планується використовувати COM-порт для зв’язку з мережевим обладнанням. Байт буде закодовано трьома логічними рівнями (всього 9 біт). Передача буде відбуватись від молодших бітів до старших. Перший біт в першій трійці буде встановлено в 0. Наприклад, байт 10011101 буде закодовано трьома комбінаціями: 010, 111 і 001. Комбінації будуть передаватися безперервно. Перед трьома комбінаціями, що утворюють байт, та після них йде обов’язкова службова комбінація, котра встановлюється щонайменше на один тактовий інтервал. Довжина тактового інтервалу буде визначена при проведенні тестування пристрою. При відсутності даних для передачі, безперервно передається службова комбінація.

СЛ.К.	$0X_0X_1$	$X_2X_3X_4$	$X_5X_6X_7$	СЛ.К.

Рис. 2.2. Формат кадру даних

На рисунку 2.2 наглядно зображений кадр, що буде переданий при байті $X_7X_6X_5X_4X_3X_2X_1X_0$. Всього на передачу одного кадру даних буде витрачатися 5 тактових інтервалів. Такий формат кадру даних забезпечить можливість розрізняти байти, що будуть передані, один від одного та будуть легко прийматися мікроконтролером, оскільки мають чіткі початок та кінець.

2.4 Вибір мікроконтролера

Мікроконтролер – мікросхема, яка застосовується для керування різними електронними пристроями та компонентами. Для розробки існують спеціальні плати, котрі мають набір необхідної периферії та виконані у зручному для проектування форматі. Для розробки та тестування будемо використовувати саме таку плату на основі мікроконтролеру, який оберемо далі.

Мікроконтролер – основа апаратної частини розроблюваного пристрою. Саме він відповідатиме за зв’язок пристрою з мережевим обладнанням, прийом та

передачу даних, буде здійснювати модуляцію та генерацію сигналу. Тому від його вибору цілком і повністю буде залежати робота всієї системи.

Вибирати мікроконтролер будемо з огляду на вимоги, що поставлені до проектованої системи. Необхідно, щоб він мав щонайменше один швидкий АЦП та відносно високу тактову частоту, оскільки потрібно одночасно виконувати декілька операцій протягом всього часу роботи, а саме виконувати обмін даними з мережевим пристроєм по UART, зчитувати вхідний сигнал та обробляти його, модулювати дані для відправки та генерувати сигнал, що буде надходити до телефонної лінії зв’язку.

Також, бажано, щоб мікроконтролер був досить популярний, оскільки це спростить процес написання програмного забезпечення та відлагодження. Також, не варто забувати про цінову політику – розроблюваний пристрій повинен бути дешевший за вже існуючі рішення.

З огляду на вищеописані вимоги, було сформовано список з двох мікроконтролерів, а саме ATmega328P та STM32F103C8T6. Розглянемо кожен з них. Визначимо переваги та недоліки та оберемо один як основу для системи.

Розглянемо ATmega328P. Це мікроконтролер сімейства AVR від виробника Atmel. Заснований на 8-bit архітектурі та має 16 МГц тактової частоти.

Рис. 2.3. Arduino Pro mini з ATmega328P

ATmega328P має три таймери, 6 десятибітних АЦП, достатню кількість портів вводу-виводу, а саме 23. Також наявні інтерфейси I2C, SPI, UART. Має 32 кілобайта пам’яті для програм.

Писати код програм для ATmega328P зручно в Arduino IDE оскільки даний мікроконтролер є основним для плат розробки Arduino. При цьому використовується високорівнева бібліотека, яка дозволяє абстрагуватись від низькорівневого програмування та оперувати сутностями. Але це можна віднести як до переваг, так і до недоліків, оскільки використання даної бібліотеки унеможливлює оперування, наприклад, таймерами. Також неможливо буде налаштувати АЦП – він буде завжди працювати на частоті, заданій в бібліотеці.

Розглянемо STM32F103C8T6. Побудований на 32-bit архітектурі. Працює на швидкостях до 72 МГц.

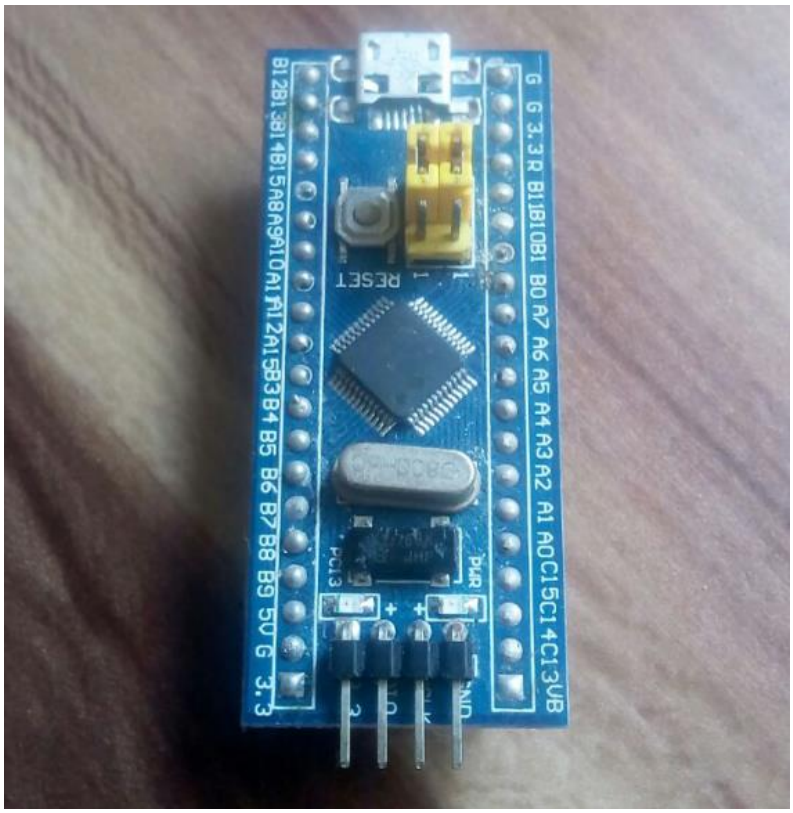

Рис. 2.5 STM32F103C8T6 на однойменній платі розробки

Має 64 кілобайти пам’яті для програм, чотири таймери 16-bit, годинник реального часу. Наявний 51 порт вводу-виводу та до 16 12-bit АЦП. Має 3 інтерфейси UART, по 2 інтерфейси SPI та I2C.

Програмувати даний мікроконтролер можна різними способами. Можливе використання аналогічної Arduino бібліотеки – STM32duino. При цьому, програмування відбувається або за допомогою USB-UART конвертора, або за допомогою вбудованого USB роз’єму. При використанні даної бібліотеки, недоліки зберігаються, переваги не з’являються. Однак, ситуація змінюється, якщо застосувати бібліотеку HAL. Вона є більш низькорівневою і вимагає конфігурування таймерів, тактування АЦП і т.д.. Це, у свою чергу, досить зручно, оскільки з’являється можливість налаштувати всю периферію так, як того потребує проект.

Проаналізувавши характеристики мікроконтролерів, було обрано STM32F103C8T6. При різниці в ціні в 10%, функціонал обраного мікроконтролера є значно ширшим, ніж у іншого кандидата. Основними вирішальними факторами стали більш точний АЦП та можливість його конфігурування, можливість використання таймерів та більший об’єм пам’яті, більша тактова частота.

2.5 Вибір ЦАП

Обраний мікроконтролер значно перевершує по характеристикам інші мікроконтролери з його цінового діапазону, однак ЦАП в нього немає. ЦАП необхідний в рамках проектованого пристрою як модуль, котрий буде генерувати сигнал для передачі по мобільному каналу зв’язку.

Одним з найдоступніших та найпоширеніших модулів ЦАП є MCP4725.

Рис. 2.6 Модуль ЦАП MCP4725

Обраний модуль ЦАП з’єднується з мікроконтролером шляхом I2C інтерфейсу. Має розрядність 12-bit. Це дає можливість досить точного регулювання вихідної напруги. MCP4725 має достатню швидкість для генерування синусоїдального сигналу частотою до 200 КГц. Цього з запасом вистачає для проектованого пристрою.

2.6 Обмін даними з мережевим обладнанням

У якості інтерфейсу, для обміну даними між мережевим пристроєм та мікроконтролером, було обрано послідовний порт передачі даних або UART.

Даний інтерфейс було обрано через те, що досить багато моделей мережевого обладнання підтримує обмін даними за допомогою послідовного порту.

Рис. 2.7 Порт RS232 на D-Link DES-3200-25

У обраного мікроконтролера відсутній найрозповсюдженіший інтерфейс RS232. Однак є декілька UART. Їх і будемо використовувати у подальшому. Конвертувати сигнал UART- RS232 будемо за допомогою спеціального модулю, у основі якого лежить мікросхема MAX3232, що виконує описану вище задачу.

Рис. 2.8 Модуль UART to RS232 на MAX3232

Даний модуль має невисоку вартість, що підходить під бюджетну політику розробки пристрою, може забезпечити швидкості обміну даними до 460800 бод/с.

2.7 Вибір блоку живлення

Вибір блоку живлення для даного проекту є важливим етапом, оскільки якість живлення напряму впливає на роботу всієї системи. Блок живлення повинен мати достатню потужність, щоб задовільнити потреби не тільки мікроконтролера і його модулів, а й телефону, оскільки останній буде заряджатися від проектованої системи. Також блок живлення повинен мати гарну стабілізацію напруги та низький рівень пульсацій. Напруга має бути 5 Вольт.

Для проектованого пристрою гарно підходить будь-який блок живлення на 5 Вольт від мережевих пристроїв, оскільки в них закладені всі характеристики, описані вище.

Рис. 2.9 Блок живлення FT-5-5PL

Для проектованого пристрою був обраний блок живлення FT-5-5PL. При стабілізованій напрузі в 5 Вольт він здатний забезпечувати струм в 1 Ампер. Має відносно невисоку ціну. Обраний блок живлення захищений від короткого замикання та має захист від перенавантажень. Корпус захищає електроніку від потрапляння пилу та бруду.

2.8 Прийом та передача сигналу

У даному розділі необхідно розробити алгоритми прийому та передачі даних у вигляді сигналу звукової частоти, на яких буде базуватись основна робота мікроконтролеру. Для досягнення максимальної швидкості взаємодії будемо використовувати неблокуючі функції, тобто такі, що основані на перериваннях.

2.8.1 Розробка алгоритму передачі даних

Для передачі даних будемо використовувати одну з основних переваг обраного мікроконтролера – один з чотирьох 16-bit таймерів. У мікроконтролерній техніці, таймери використовуються майже у будь-якому проекті, оскільки відіграють ключову роль у періодичних процесах, лічбі і т.д..

Таймери можуть застосовуватись для:

- виміру довільних проміжків часу;
- рахування заданих подій;
- генерації переривань [9];
- генерування ШИМ-сигналу
- тактування інших пристроїв;
- генерації подій при певних умовах.

Також існують сторожові таймери, які слугують для перезавантаження всього мікроконтролера у разі виникнення його зависання.

Будемо використовувати один із таймерів як генератор переривань через сталі проміжки часу. Це потрібно для встановлення на ЦАП певної напруги.

Від мережевого обладнання на вхід UART інтерфейсу будуть надходити дані. Їх потрібно зберігати до моменту передачі. Тому було прийнято рішення про створення акумулятору. Дані будуть надходити і записуватись у акумулятор починаючи з його першої комірки і, по мірі прийому даних через UART, до другої і т.д.. При переповнені акумулятора дані будуть втрачатись, тож необхідно зробити його максимально великим.

При старті процесу передачі, перший байт з акумулятору буде зчитуватись у змінну, а останні – зміщуватись на один байт вліво.

Зчитаний байт буде приведено до набору комбінацій, описаних в розділі 2.3. Буде сформовано кадр даних, готовий до передачі. Передача ж буде відбуватися поетапно. Через рівні проміжки часу таймером буде генеруватися переривання, під час якого на ЦАП буде встановлена напруга. Оскільки бажана форма сигналу на виході – синусоїда частотою до 5000 Гц (період 200 мікросекунд), то час між перериванням встановимо 5 мікросекунд задля забезпечення достатньо точного генерування синусоїди. На один період синусоїди буде приходитись щонайменше 40 переривань.

Після закінчення передачі останньої комбінації відбудеться спроба зчитування з акумулятора наступного байту. При його відсутності, буде передаватись службова комбінація.

Блок-схема алгоритму передачі зображена на ІА51.180БАК.005 Д1.

2.8.2 Розробка алгоритму прийому даних

Для прийому аналогового сигналу скористаємось тим же механізмом, що і при передачі даних – перериваннями таймеру. Для цього задіємо інший таймер. Всього у нашому пристрої чотири таймери.

Встановимо час між перериваннями 5 мікросекунд. У подальшому, в ході тестування та відлагодження, цей час можна буде змінити. При даному часі між перериваннями, на один період синусоїди буде приходитись щонайменше 40 переривань. Цього достатньо, щоб сформувати точну картину сигналу, що надходить телефонним каналом зв’язку.

Під час переривань буде виконуватись перетворення АЦП.

Мікроконтролер має 12-bit АЦП, який виконує перетворення напруги в еквівалентне числове значення в межах 0 – 3.3 В. Розрядність АЦП можна змінити та встановити в 12, 10 або 8 bit. При цьому, швидкість перетворення зростає при зменшенні розрядності перетворювача, але падає точність.

Під час кожного переривання, викликаного таймером, буде відбуватися перетворення АЦП, результат буде заноситися у змінну. Буде використано модуль прямого доступу до пам’яті для пришвидшення переносу даних від АЦП до пам’яті. При цьому, сам мікроконтролер у цьому процесі буде майже незадіяний. Також буде відбуватися порівняння поточного результату з результатами перетворення на попередніх кроках. Таким чином будуть визначатися точки перетину синусоїдою відмітки в 0 Вольт. Знаючи час перетину попередньої точки та поточної, можна буде визначити період коливання синусоїди, а, отже, і частоту. Знаючи частоту, можна порівняти її з таблицею 2.1 відповідності частот та комбінацій бітів і відновити бінарну комбінацію. Прийнявши кадр даних, з’явиться можливість відновити байт. Відновлений байт буде переданий до UART, що виконає його надсилання до мережевого пристрою. Обмін даними по інтерфейсу UART буде відбуватися за допомоги модуля прямого доступу до пам’яті. Використання модулю зменшить навантаження на мікроконтролер та дасть виграш по швидкості.

Операції з прийому та передачі даних будуть виконуватись паралельно одна одній.

Блок-схема алгоритму прийому зображена на ІА51.180БАК.005 Д2.

2.9 Висновки до розділу 2

У результаті виконання другого розділу дипломного проекту було обране середовище, що буде використовуватись для обміну даними – канал, що утворюється при дзвінку з одного телефону на інший. Оскільки у даному каналі діє частота звукового діапазону, було обрано частотну модуляцію для передачі двійкових даних.

Вирішено використовувати 8 логічних рівнів для підвищення кількості інформації, яка буде передаватись за одиницю часу. Сформовано кадр даних та описано його формат. Розроблено алгоритми дій для прийому та передачі даних по

визначеному вище каналу. Обмін інформацією з мережевим пристроєм буде здійснюватися по інтерфейсу UART.

Проаналізувавши наявні пропозиції, було обрано мікроконтролер, що буде виконувати всі основні операції – STM32F103C8T6. Вій має достатньо ресурсів для успішного виконання поставленого перед ним завдання – його швидкодія дозволить виконувати одночасно передачу, прийом даних по телефонному каналу і обмін даними з мережевим пристроєм. У обраного мікроконтролеру відсутні ЦАП і інтерфейс RS232 У зв'язку з цим, було обрано відповідні модулі. Для генерації синусоїдального сигналу було обрано ЦАП MCP4725, а для переходу від UART до RS232 було обрано модуль перетворювач на мікросхемі MAX3232.

Була розроблена структурна схема системи ІА51.180БАК.005 ДЗ.

Наразі можна приступати до виконання третьої частини дипломного проекту, а саме реалізації спроектованої системи.

3 РОЗРОБКА ПРОГРАМНОЇ ТА АПАРАТНОЇ РЕАЛІЗАЦІЇ

3.1 Вибір середовища розробки та основної бібліотеки

Приступаючи до реалізації проекту варто серйозно віднестись до вибору середовища розробки, оскільки дане рішення може вплинути на швидкість реалізації та зручність роботи.

Середовище програмування має містити редактор коду, компілятор, засоби автоматизації збірки та можливість проводити відлагодження написаних програм безпосередньо на мікроконтролері.

З огляду на дані вимоги, сформуємо список IDE, котрі можуть бути використані в якості основного середовища розробки. До списку увійдуть лише найпопулярніші з них, оскільки такі середовища розробки, зазвичай, мають гарну підтримку зі сторони їх розробників, велику інформаційну базу та підтримують необхідні бібліотеки.

Для розгляду були вибрані наступні середовища розробки:

- CooCox IDE;
- IAR Embedded Workbench;
- Keil µVision IDE;
- STM32CubeMX IDE;
- Arduino IDE.

Проаналізуємо кожне з них та оберемо те, що буде використане в якості основного середовища розробки.

CooCox IDE – просте в установці та освоєнні середовище розробки, створене спеціально для програмування мікроконтролерів сімейства ARM. Розроблене на основі іншого IDE, котре має назву Eclipse, CooCox IDE є безкоштовним програмним забезпеченням із відкритим вихідним кодом. Для скачування необхідно лише зареєструватися на сайті. У даному середовищі розробки без проблем можна створити проект для конкретного мікроконтролера. При цьому IDE

саме додасть необхідні низькорівневі бібліотеки та включить їх у список автоматичної збірки проекту.

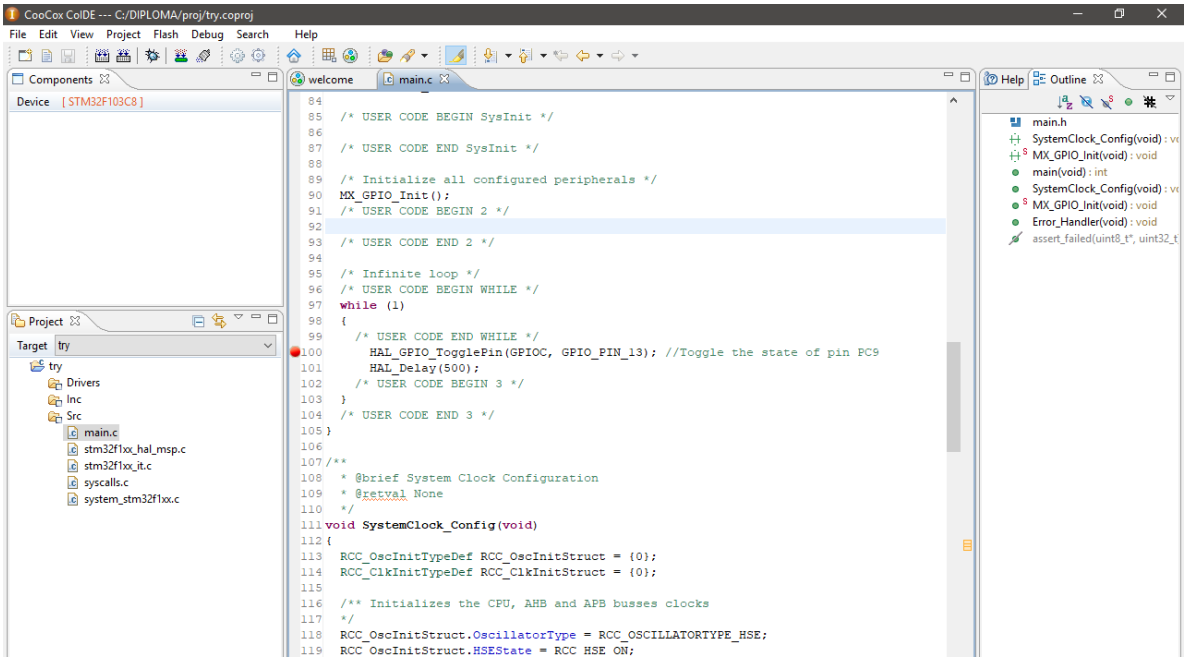

Рис. 3.1 Вигляд основного вікна CooCox IDE

IDE може працювати з програматором ST-Link та підтримує всі основні режими відлагодження програм.

Однак, у даного середовища розробки є й недоліки. По-перше, після встановлення IDE потрібно в ручному режимі встановити компілятор, оскільки у комплекті з програмою він не поставляється. По-друге, програма дуже залежна від шляху до папки встановлення. При розміщенні будь-якого компоненту програми у папці, шлях до якої містить символи кирилицею, програма починає працювати некоректно. Також, всі шляхи до файлів прописується у програмі при становленні. Тобто, якщо існує потреба перемістити той чи інший використовуваний програмою файл, то простіше перевстановити саме середовище розробки.

Розглянемо IAR Embedded Workbench. Дане середовище розробки є професійним та підтримує майже всі, доступні на сьогоднішній день мікроконтролери. Порівняно з попереднім IDE в комплект поставки одразу входить компілятор. Крім нього також поставляється весь комплект, необхідних для

розробки під конкретну платформу, утиліт та драйверів. До складу середовища розробки входять також різноманітні низько та високорівневі бібліотеки для програмування мікроконтролерів та мікропроцесорів.

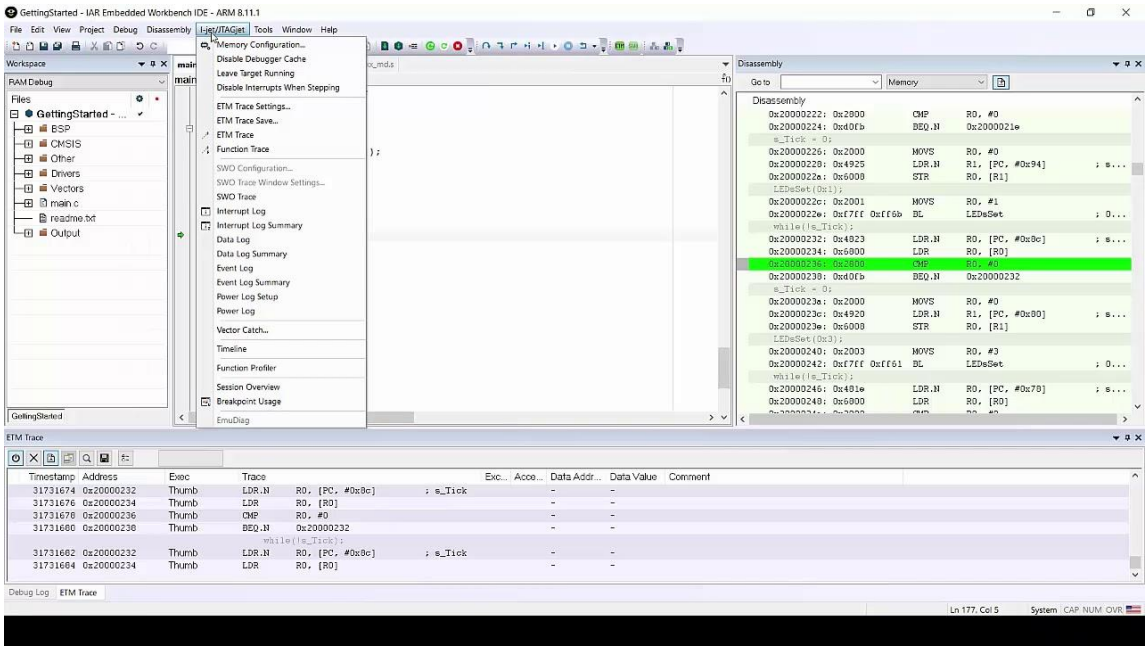

Рис. 3.2 Вигляд основного вікна IAR Embedded Workbench [10]

Хоча це середовище розробки і відповідає всім вимогам, поставленим до шуканої IDE, використовувати його для виконання практичної частини даної роботи немає можливості, оскільки IAR Embedded Workbench є платним продуктом.

Розглянемо Keil µVision IDE. Дане середовище розробки, як і попереднє, є професійним інструментом розробника. Має зручний та зрозумілий інтерфейс. Реалізовані основні, необхідні для розробки інструменти. У комплекті поставки відразу є і компілятор, і відлагоджувач.

Середовище розробки має широкий спектр різноманітних налаштувань. Конфігурувати це IDE досить просто та зручно. Присутня підтримка більшості сучасних мікроконтролерів та мікропроцесорів. Є підтримка бібліотеки HAL для програмування мікроконтролерів сімейства STM.

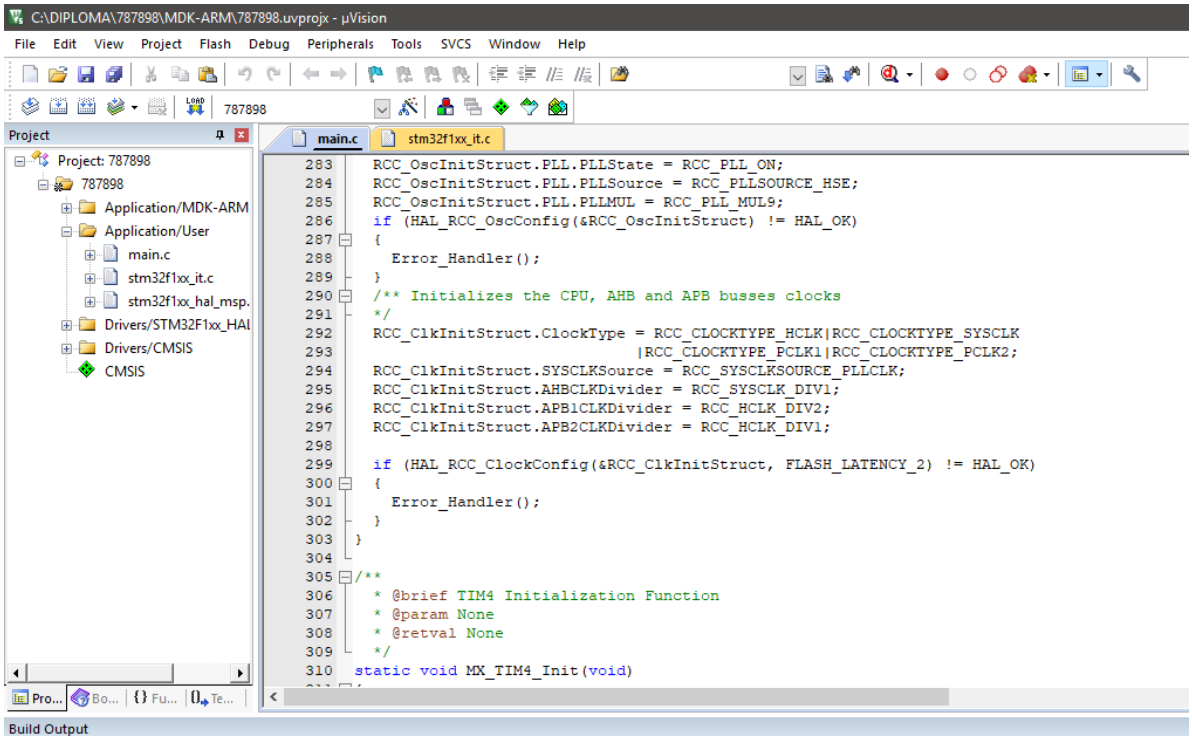

Рис. 3.3 Вигляд основного вікна Keil µVision IDE

Дане середовище розробки, як і попереднє, розповсюджується розробником та коштує грошей. Однак, пройшовши реєстрацію на сайті, можна завантажити демо-версію. Вона майже не відрізняється від повноцінної, але має один недолік – розмір згенерованої програми не може бути більше 32 кілобайт. Цей факт накладає досить вагомі обмеження на програмне забезпечення для мікроконтролера і може стати причиною неможливості написання коду наприкінці етапу реалізації.

Розглянемо Arduino IDE. Це середовище розробки задумувалось як основне для ряду плат розробки Arduino. Але, оскільки для програмування була створена досить високорівнева бібліотека, яка значно спрощувала процес написання коду і знижувала поріг входження у програмування мікроконтролерів, в Arduino IDE почали програмувати й інші мікроконтролери. Бібліотеку успішно змінювали під потрібний мікроконтролер. Так і з'явилась бібліотека STM32duino.

Бібліотека STM32duino є досить зручною, оскільки в ній немає необхідності оперувати низькорівневою периферією.

Рис. 3.4 Вигляд основного вікна Arduino IDE

Цей же фактор і змушує відмовитись від використання даного середовища розробки, оскільки потрібно буде використовувати більше можливостей мікроконтролера, ніж це може дозволити бібліотека STM32duino в парі з Arduino IDE.

Розглянемо STM32CubeMX IDE. Це середовище розробки по інтерфейсу і функціоналу дуже схоже на CooCox IDE, тому що створене на його основі.

У STM32CubeMX IDE виправлені всі ключові недоліки CooCox IDE, тому немає необхідності встановлювати компілятор – він поставляється разом з IDE. Також, файли можна переносити без потенціальної можливості зруйнувати проект.

Найбільшим плюсом даного середовища розробки є наявність генератора коду з використанням бібліотеки HAL. Це значно спростить процес використання периферії мікроконтролеру. Розробнику потрібно вибрати налаштування у графічному інтерфейсі, а генератор коду створить та підключить необхідні файли та запише в них код ініціалізації таймеру, АЦП, порту вводу-виводу, тощо.

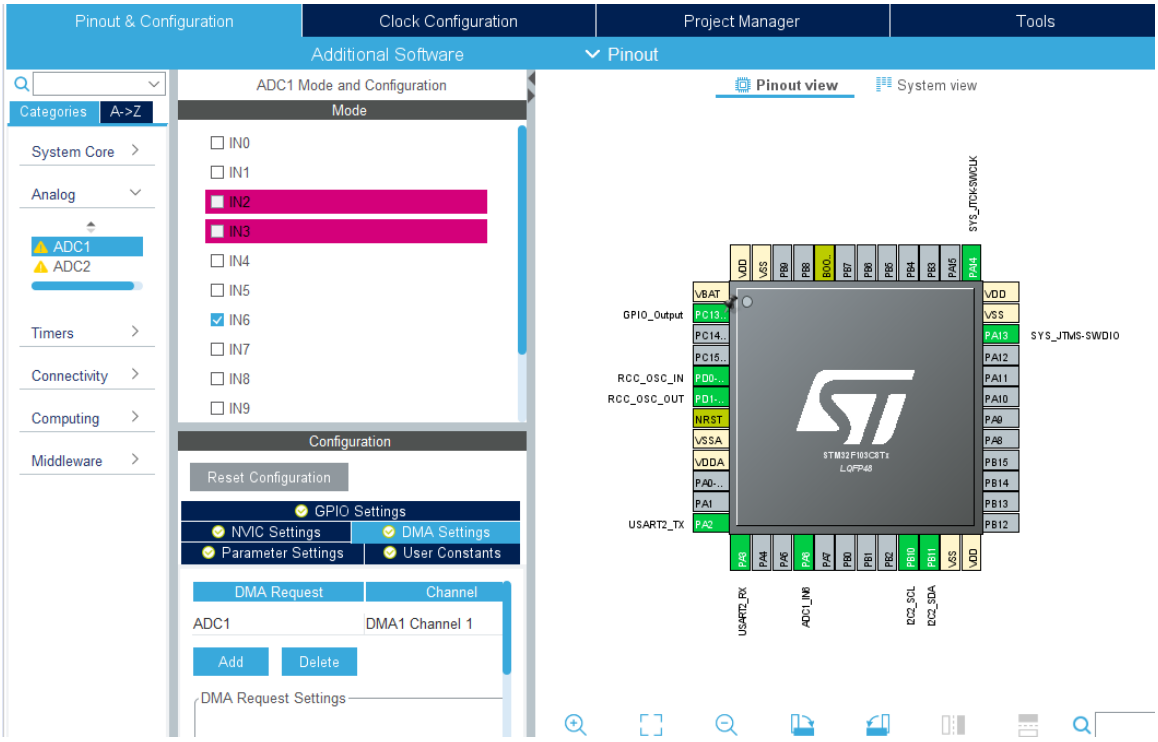

Рис. 3.5 Вигляд вікна генератора коду ініціалізації в STM32CubeMX

Отже, для розробки програмного забезпечення буде використовуватись середовище STM32CubeMX. З його допомогою можна буде здійснити повний цикл розробки програми для мікроконтролера та її тестування, використовуючи реальні модулі.

Відлагодження керуючої програми буде відбуватися безпосередньо на самому мікроконтролері, що дасть можливість більш об’єктивно оцінювати його можливості.

Для початкової ініціалізації буде використовуватись згенерований середовищем розробки код на основі бібліотеки HAL. Згенерований код можна буде змінювати в ручному режимі після його створення, але якщо потрібно буде знову скористатися генератором коду, то користувацький код, написаний не у відведених для цього місцях, буде безповоротно видалено. Це варто враховувати і. при необхідності, переписувати код ініціалізації лише у тих місцях, що виділені спеціальним коментарем. В ньому обов’язково фігурують кодові слова «user code»

Використання даного середовища програмування з генератором коду ініціалізації значно скоротить затрати часу, необхідні для написання коду.

3.2 Створення проекту та ініціалізація периферійного обладнання

Після встановлення середовища розробки STM32CubeMX на робочий комп’ютер переходимо до створення проекту, в якому будемо реалізовувати необхідний функціонал.

На початку створення, буде запропоновано вибрати цільову плату розробки або мікроконтролер. Так як необхідної плати в STM32CubeMX немає, то необхідно вибрати мікроконтролер STM32F103C8T6. Після цього стає доступним вікно ініціалізації та початкового налаштування периферії мікроконтролера.

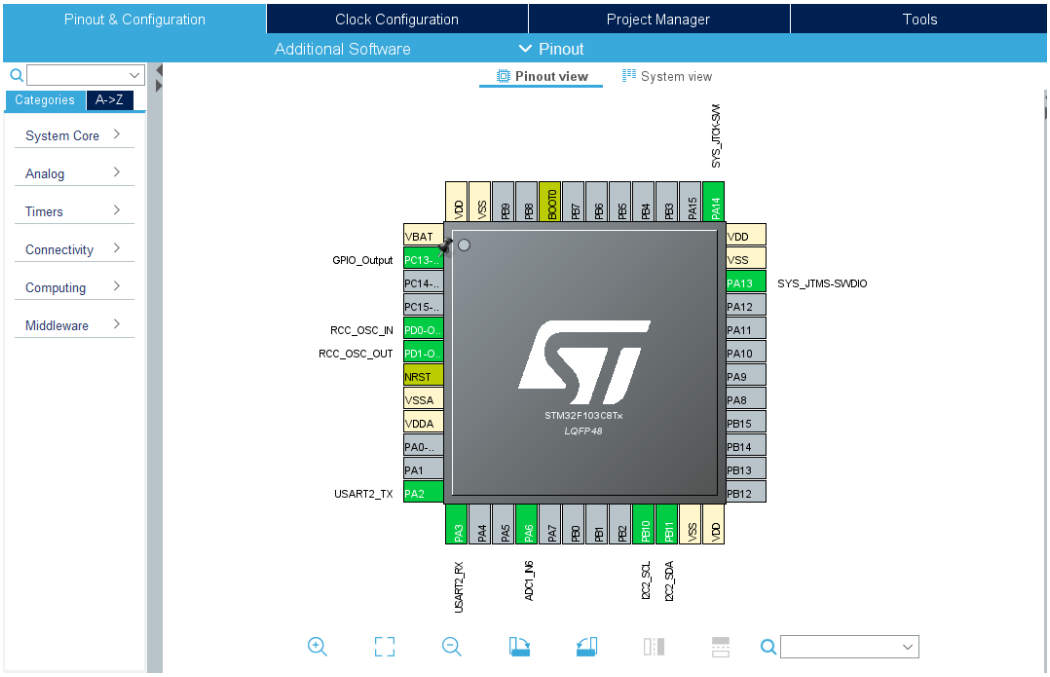

Рис. 3.6 Вікно ініціалізації периферії мікроконтролера

У вкладці System Core в RCC вибираємо Crystal/Ceramic Resonator як джерело тактування для мікроконтролера. Це необхідно зробити задля досягнення максимальних, для цього мікроконтролера, частот роботи усіх, пов’язаних з системним таймером, вузлів. Ними є АЦП та таймери. Максимальна частота роботи даного мікроконтролера складає 72 МГц. Також STM32F103C8T6 має можливість збільшити частоту до 128 МГц, але це, по-перше, може негативно сказатись на ресурсі роботи мікроконтролера, по-друге, не всі шини мають можливість

працювати на такій частоті. Потрібно буде постійно зменшувати її до нормальної робочої, а, при необхідності, знову збільшувати.

Переходимо до вкладки Clock Configuration. В ній маємо можливість зконфігурувати переддільники та множники частот для різних елементів системи.

Переддільник частоти – це лічильник, що використовується для зменшення частоти сигналу. Необхідний для подальшого більш зручного використання електричного сигналу. Найчастіше, значення переддільника обмежені там мають значення степенів двійки.

Середовище розробки саме повідомить нам про максимальні допустимі значення частот для тих чи інших елементів. При перевищенні допустимих частот, комірка з числовим значенням стане червоною, а середовище розробки запропонує автоматично вирішити конфлікт.

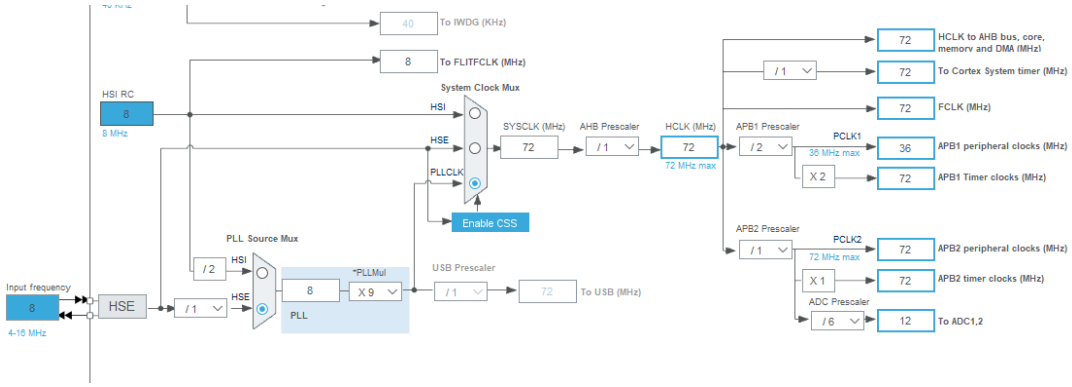

Рис. 3.7 Налаштування у вкладці Clock Configuration

На рисунку 3.7 показані необхідні налаштування переддільників для розроблюваної системи. Особливу увагу потрібно звернути на шину APB1 та ADC. Від них будуть тактуватися таймери, що будуть використані для подальшої розробки та аналогово-цифровий перетворювач.

Далі у вкладці SYS потрібно встановити значення Serial Wire у полі Debug. Ця дія надасть можливість проводити відлагодження написаної програми безпосередньо на мікроконтролері. Програмувати та відлагоджувати програмне забезпечення потрібно через програматор. Буде використаний ST-Link V2.

3.2.1 Ініціалізація таймерів

При натисканні на вкладку Timers можна побачити доступні розробнику таймери. Їх у обраному мікроконтролері чотири.

Таймер – один з найважливіших пристроїв у мікроконтролері. Він може виконувати багато видів задач, серед яких:

- вимір проміжків часу;
- рахування заданих подій;
- генерація переривань, подій при певних умовах;
- генерування сигналів, тощо.

Розробники мікроконтролерів намагаються додати до своїх плат максимально багато різноманітних таймерів. З одного боку, це дуже спрощує процес написання коду для досвідченого розробника, оскільки йому набагато простіше використати вже підготовлений, спеціалізований таймер. Але, у той же час, початківцю буде важко розібратися, який таймер і для чого використовувати.

У обраній для проекту платі досягнуто компромісу у кількості таймерів та їх функціоналі. Чотири таймери – достатня кількість для реалізації майже будь-якого проекту, з огляду на відносно невисоку кількість пам’яті (64 КБ). Усі таймери можуть бути як головними(master) так і веденими (slave). Побудовані на 16-bit архітектурі. Джерелом тактових імпульсів може виступати як системна шина, до якої під’єднаний таймер, так і, наприклад, інший таймер. Деякі мають фізичні входи для прийняття тригерного сигналу.

В проекті будуть використані TIM4 та TIM3. Перший таймер буде використовуватись для генерування переривань, в яких буде відбуватись зміна напруги на цифро-аналоговому перетворювачі. Працюватиме таймер у режимі Master і буде використовувати для тактування системну шину.

Переривання вирішено генерувати кожні 5 мкс. Для встановлення часу між перериваннями будуть використані два параметри таймеру – переддільник та лічильник. Частота системної шини таймерів дорівнює 36 МГц. Переривання викликається при переповненні лічильника, отже потрібно встановити його

значення в 359, оскільки нумерація починається з нуля. При цьому, переддільник, нумерація якого теж починається з нуля, буде встановлений в нуль. Це дасть змогу генерувати переривання з частотою в 100 КГц.

Другий таймер, TIM3, буде використовуватись як генератор подій для аналогово-цифрового перетворювача. Частоту вимірів АЦП теж було вирішено встановити в 100 КГц. Для налаштування обраного таймера використаємо ті ж значення відповідних параметрів таймеру TIM4.

Інші таймери використовуватись не будуть.

3.2.2 Використання режиму прямого доступу до пам'яті

У подальшій розробці будо прийнято рішення використовувати режим прямого доступу до пам'яті – DMA. DMA – це модуль мікроконтролера, який може виконувати обмін даними між його частинами з мінімальним застосуванням ресурсів самого мікроконтролера. Режим прямого доступу до пам'яті значно скорочує кількість мікроконтролерного часу, необхідного на роботу з периферійними пристроями.

В STM32F103C8T6 наявний один 7-канальний DMA контролер. За кожним каналом закріплене певне периферійне обладнання. Це означає, що при використанні одним обладнанням певного каналу, інше обладнання, що підключене до цього ж каналу, використовувати його не зможе. Також, канали мають свій пріоритет. Система пріоритетів застосовується у тому випадку, якщо існує вірогідність одночасного використання декількох каналів. Більш пріоритетним визнається той канал, у якого відповідне числове значення пріоритету менше.

Table 78. Summary of DMA1 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1	-	-	-	-	-	-
SPI/I^2S	-	SPI1_RX	SPI1_TX	SPI2/I2S2_RX	SPI2/I2S2_TX	-	-
USART	-	USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I^2C	-	-	-	I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1	-	TIM1_CH1	-	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
TIM2	TIM2_CH3	TIM2_UP	-	-	TIM2_CH1	-	TIM2_CH2 TIM2_CH4
TIM3	-	TIM3_CH3	TIM3_CH4 TIM3_UP	-	-	TIM3_CH1 TIM3_TRIG	-
TIM4	TIM4_CH1	-	-	TIM4_CH2	TIM4_CH3	-	TIM4_UP

Рис 3.8 Канали DMA та обладнання, що до них підключене [11]

DMA будемо використовувати для швидкого обміну даними між UART та мікроконтролером. Також для швидкого запису даних після аналогово-цифрового перетворення.

З рисунку 3.8 бачимо, що для АЦП варто використати перший канал, оскільки ADC1 розташований тільки там. Для UART використаємо шостий та сьомий канали, тобто використовуватиметься UART3. Також, знадобиться I2C для підключення модуля ЦАП. Використаємо для цього четвертий та п’ятий канали. Другий та третій канали залишимо незадіяними для подальшої можливої модернізації пристрою. На них може бути розташований пристрій, що працює з використанням протоколу обміну даними SPI.

3.2.3 Ініціалізація АЦП

АЦП [12] це пристрій, що приймає на вхід аналоговий сигнал довільної величини (в певних рамках), а на виході видає еквівалентне числове значення. STM32F103C8T6 має два АЦП, кожен з яких має розрядність 12 bit. З цього слідує, що результатом перетворення буде число від 0 до 4095.

Всього мікроконтролер має 18 каналів і один додатковий у вигляді температурного сенсору. Вимірювати можна напругу від 0 до 3.3 Вольт.

Для кожного з АЦП та їх каналів доступні декілька режимів перетворення:

- одноразове;
- безперервне;
- по тригеру;
- за таймером.

У режимі одноразового перетворення, АЦП, відповідно, виконує одне перетворення, після чого зупиняється та чекає наступного виклику. Управління здійснюється за допомогою керуючого біта ADON у регістрі ADC_CR1. При встановленні біта в 1, починається перетворення. Після його завершення, біт встановлюється в 0.

У режимі безперервного перетворення, АЦП виконує ті ж самі одноразові перетворення, але послідовно. При цьому біт ADON керує початком та закінченням серії перетворень. Даний режим можна увімкнути встановленням біта CONT у регістрі ADC_CR2 в 1 та вимкнути встановленням того ж біта в 0. Після завершення кожного перетворення, генерується переривання, якщо встановлений відповідний біт, для того, щоб користувач міг зберегти результат. Завдяки режиму безперервного перетворення, можна досягнути скорочення часу серії з декількох перетворень у порівнянні з виконанням такої ж по кількості серії при використанні режиму одноразового перетворення.

У режимі перетворення по тригеру, АЦП починає перетворення після певної події. Такою подію може генерувати таймер. Також є спеціальні вводи на мікроконтролері (для STM32F103C8T6 це A6 та D0), встановлюючи на яких напругу, що відповідає високому логічному рівню, можна згенерувати подію, що призведе до виконання перетворення аналогово-цифровим перетворювачем.

У режимі перетворення за таймером, виконання перетворення буде викликатись перериванням таймеру. При цьому, буде виконане просте одноразове перетворення.

Для проекту було обране переривання по таймеру, оскільки перетворення має виконуватись через певні проміжки часу, які важко точно зконфігурувати, використовуючи лише налаштування АЦП.

У вкладці Analog вибираємо ADC1 та канал IN0. У вікні конфігурації потрібно виставити значення Timer 3 Trigger Out event для комірки External Trigger Conversion Source. Таким чином, переривання по переповненню лічильника таймеру TIM3 буде викликати початок перетворення АЦП. Після перетворення буде викликатись callback-функція, у якій потрібно буде зберегти результат перетворення та обробити його. Значення комірки Data Alignment було виставлено як Right Alignment. Дані з АЦП будуть заноситись до змінної, починаючи з молодших бітів. При Left Alignment – зі старших. У вкладці DMA того ж вікна, додаємо необхідний канал. Середовище розробки саме запропонує доступні канали. У випадку АЦП, доступним буде тільки перший канал.

Всі інші налаштування були залишені зі значеннями за замовчуванням. У майбутньому, при модернізації системи, їх можна буде змінювати.

3.2.4 Ініціалізація УАПП

STM32F103C8T6 має три інтерфейса УАПП. Між собою вони не відрізняються.

УАПП – послідовний інтерфейс для обміну даними. Використовуватися активно почав ще у 1973 році [13], але залишається актуальним і до сьогодення, оскільки простий та дешевий у реалізації. Він реалізований на більшості пристроїв, що мають мікроконтролери або мікропроцесори.

UART буде використано для обміну даними з цільовим мережевим пристроєм. Найчастіше, у мережевих пристроях використовується інтерфейс RS232, тому будемо використовувати модуль-перехідник на мікросхемі MAX3232.

У вкладці Connectivity було активовано UART2 шляхом вибору у його налаштуваннях режиму «асинхронний». Був обраний саме UART2, оскільки ми визначили шостий та сьомий канали DMA як канали, що будуть використані для прийому та передачі даних між мережевим пристроєм та мікроконтролером. Їх потрібно виставити у вкладці DMA Settings. Середовище розробки саме запропонує необхідні канали, тому помилитися буде неможливо.

У вікні Configuration виставляємо необхідні нам параметри:

- Baud Rate – 115200 Bits/s;
- Word Length – 8 Bits;
- Parity – None;
- Stop Bits – 1.

Baud Rate відповідає за швидкість роботи інтерфейсу. Вимірюється у кількості бод, переданих за одну секунду. 115200 бод/с – найрозповсюдженіше значення швидкості Серед послідовних інтерфейсів мережевих пристроїв. Word Length – довжина слова даних з бітом парності, а, оскільки, Parity було виставлено в None, то довжина слова складе 8 біт. Кількість стоп-бітів (Stop Bits) виставлена в 1.

Такий набір параметрів є найрозповсюдженішим, однак зконфігурований таким чином інтерфейс буде працювати не з усіма мережевими пристроями. Дану проблему можна вирішити шляхом програмної зміни параметрів інтерфейсу.

3.2.5 Ініціалізація I2C інтерфейсу

Для зв’язку з ЦАП буде використано інтерфейс I2C. Це обумовлено використанням стороннього цифро-аналогового перетворювача, у якого наявний лише такий спосіб обміну даними.

I2C – інтерфейс, що дозволяє підключити до однієї шини більш ніж 100 пристроїв (до 127 при використанні 7-bit адреси [14]). Використовуються 2 провідники (не враховуючи «землі») – SDA та SCL. Перший відповідає за безпосередню передачу даних, другий за передачу синхросигналу. Інтерфейс працює на швидкостях 100 КБіт/с та 400 КБіт/с. Один з пристроїв на шині має виконувати роль керуючого. Керуючий пристрій виконує тактування Недоліком такого способу передачі даних є дальність. Довжина провідників є обмеженою, але в рамках даного проекту цей недолік не відіграє значної ролі.

STM32F103C8T6 має два інтерфейси I2C. Використовуватись буде другий – I2C2, оскільки було обрано п’ятий та шостий канали DMA.

У вкладці Connectivity потрібно обрати I2C2 та увімкнути його. У вікні Configuration була задана швидкість 100 КБіт/с, тому що модуль цифро-аналогового перетворювача працює саме на такій швидкості. Також був заданий розмір адреси в 7 біт.

3.3 Результат роботи генератору коду

Після підключення та конфігурування всіх пристроїв, для них було задіяно десять контактів мікроконтролеру.

Результатом роботи генератору коду є проект, який включає в себе всі необхідні бібліотеки, драйвери, хедери та файли проекту.

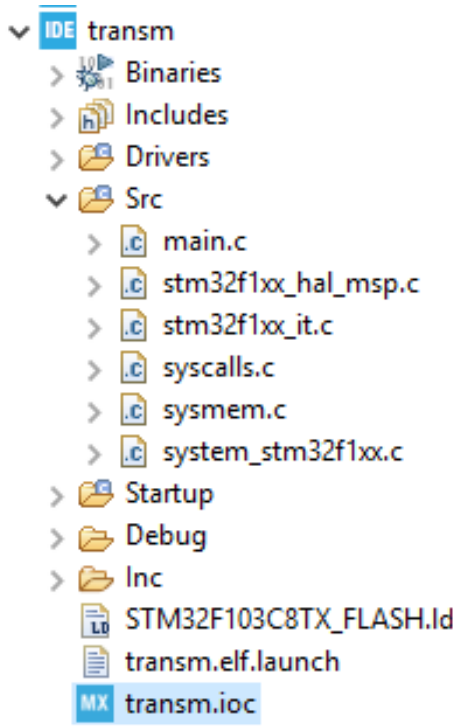

Рис. 3.9 Структура створеного проекту

Основні файли для написання прошивки для мікроконтролера зберігаються у папці Src. Перш за все, це файли main та stm32f1xx_it.c. У першому файлі відбувається ініціалізація всіх периферійних пристроїв та їх конфігурування. Також там знаходиться безкінечний цикл, у якому команди виконуються від першої

до останньої по черзі. Після виконання останньої команди, виконання відбувається спочатку.

Другий файл – файл переривань. У ньому знаходяться, необхідні для подальшої розробки, переривання таймерів, DMA, АЦП.

Також у файлі stm32f1xx_it.c знаходяться різні переривання, що викликаються системою при виникненні різного роду помилок у роботі мікроконтролеру.

3.4 Реалізація алгоритму передачі даних

Реалізовувати механізм передачі було вирішено у файлі stm32f1xx_it.c, оскільки у ньому знаходиться ключова функція переривання, котре генерує таймер.

Задача механізму передачі полягає у генеруванні синусоїдального сигналу, відповідно до даних, що потрібно передати.

Щоб задати необхідне значення напруги, через інтерфейс I2C у цифро-аналоговий перетворювач необхідно передати масив, що складається з трьох елементів. Перший – байт, що повідомляє ЦАП інформацію про необхідність запису даних про напругу у постійну пам’ять модулю. Дана функція забезпечує відновлення певного рівня напруги при включенні модулю. У розроблюваному пристрої використовувати цей механізм немає потреби. Другий і третій байти – це старші та молодші біти значення напруги, що необхідно встановити.

```
void mcp4725SetVoltage(int output)
{
  buffer[0] = mcp4725_dac;
  buffer[1] = (output / 16);       // Upper data bits     (D11.D10.D9.D8.D7.D6.D5.D4)
  buffer[2] = (output % 16) << 4;  // Lower data bits     (D3.D2.D1.D0.x.x.x.x)
  HAL_I2C_Master_Transmit_DMA(&hi2c2, mcp4725_address, buffer, 3);
```

Рис. 3.10 Функція передачі даних до ЦАП

Безпосередня передача відбувається викликом функції HAL_I2C_Master_Transmit_DMA, до якої необхідно передати адресу структури

даних, що описує I2C інтерфейс, адресу пристрою, масив з трьох байтів даних та кількість елементів у ньому.

Оскільки точно відомі проміжки часу, а саме 5 мікросекунд, через які буде викликатись переривання, то було вирішено обрахувати та зберігати значення синусу у певні моменти часу для кожного з рівнів сигналу.

```
float s000[60] = {0.105, 0.208, 0.309, 0.407, 0.5, 0.588, 0.669, 0.743, 0.809, 0.866, 0.914, 0.951, 0.978, 0.995, 1, 0.99
float s001[62] = {0.101, 0.201, 0.299, 0.394, 0.485, 0.571, 0.651, 0.725, 0.791, 0.849, 0.898, 0.938, 0.968, 0.988, 0.999
float s010[64] = {0.098, 0.195,0.29,0.383,0.471,0.556,0.634,0.707,0.773,0.831,0.882,0.924, 0.957,0.981,0.995,1,0.995,0.98
float s011[66] = {0.095, 0.189,0.282,0.372,0.458,0.541,0.618,0.69,0.756,0.815,0.866,0.91,0.945,0.972,0.99,0.999,0.999,0.9
float s100[68] = {0.092, 0.184,0.274,0.361,0.446,0.526,0.603,0.674,0.739,0.798,0.85,0.895,0.932,0.962,0.983,0.996,1,0.996
float s101[70] = {0.09, 0.179,0.266,0.351,0.434,0.513,0.588,0.658,0.723,0.782,0.835,0.881,0.92,0.951,0.975,0.991,0.999,0.
float s110[72] = {0.087, 0.174,0.259,0.342,0.423,0.5,0.574,0.643,0.707,0.766,0.819,0.866,0.906,0.94,0.966,0.985,0.996,1,0
float s111[74] = {0.085,0.169,0.252,0.333,0.412,0.488,0.56,0.628,0.692,0.751,0.804,0.852,0.893,0.928,0.956,0.978,0.992,0.
float s1[50] = {0.125,0.249,0.368,0.482,0.588,0.685,0.771,0.844,0.905,0.951,0.982,0.998,0.998,0.982,0.951,0.905,0.844,0.7
```

Рис 3.11 Масиви зі значеннями синусів

Такий підхід було вирішено використовувати для зменшення навантаження на мікроконтролер. При застосуванні даного підходу, йому не доведеться виконувати підрахунок значення синусу. Операція читання елементу з масиву займає менше часу ніж обрахунок синусу числа.

В функції переривання по таймеру викликається функція start_transmit.

```
void start_transmit()
{
    if(current_transmission_stage == (uint8_t)-1)
      {
          parse_next_byte();
          transmit();
      }
      else
      {
          transmit();
      }

}
```

Рис 3.12 Тіло функції start_transmit

Ціллю цієї функції є перевірка значення змінної current_transmission_stage. Ця змінна приймає значення номеру частини слова даних, що передається у даний момент. Якщо значення даної змінної дорівнює -1, то це означає, що передача

закінчилась і необхідно зчитати та розкласти на частини слова наступний байт для передачі. Якщо значення не -1, то необхідно продовжувати передачу.

За формування частин слова для передачі відповідає функція parse_next_byte. У ній відбувається виклик функції зчитування байту з акумулятора прийому по UART та його розбиття на три частини по три біти (у початок зчитаного байту дописується 0). Далі, відповідно до числового значення утворених змінних, вибирається три масиви з значеннями синусів.

```
void parse_next_byte()
{
    uint8_t temp;
    next_byte_to_transmit = get_next_byte_to_transmit();
    if(next_byte_to_transmit == (uint8_t)-1)
    {
        current_transmission_stage = 0;
        current_transmission_step = 49;
    }
    else
    {
        for(int i = 1; i < 4; i++)
        {
            temp = ((next_byte_to_transmit & masks[i-1]) >> (i-1)*3);
            stages[i] =  freq[temp];
            transmission_steps_quantity[i] = sizes[temp];
        }
        current_transmission_stage = 4;
    }
}
```

Рис. 3.13 Тіло функції parse_next_byte

Оскільки формат слова даних передбачає передачу службової послідовності на початку та в кінці слова, то перша та остання частини слова можна задати завчасно. Для зберігання масивів значень синусів використовується масив вказівників. Перший та останній його елементи завжди вказують на масив, котрий використовується для передачі службового символу. Інші три елементи задаються у відповідності до, утворених шляхом розбиття байту для передачі на 3 частини, чисел.

Також, на основі отриманих чисел, визначається і кількість елементів масивів з синусами та заноситься до відповідних змінних у інший масив. Перший та

останній його елементи дорівнюють числу елементів у масиві синусів для службової послідовності.

Функція зчитування байту з акумулятора може повернути значення, що дорівнює -1. Це означає, що на даний момент нові дані по UART від мережевого пристрою ще не надійшли. У такому разі змінна current_transmission_stage буде встановлена в 0, для одноразової передачі службової послідовності, а змінна, що вказує на наступний для передачі елемент масиву синусів, стає рівною кількості елементів.

Функція get_next_byte_to_transmit відповідає за зчитування байту з акумулятора, в котрий записуються дані, прийняті по UART від мережевого пристрою.

```
uint8_t get_next_byte_to_transmit()
{
    if(last_received_byte == (uint8_t)-1)
    {
        return -1;
    }
    else
    {
        result = acc[0];
        for(int i=0; i < sizeof(acc)/sizeof(*acc); i++)
        {
            temp=acc[i];
            acc[i]=acc[i+1];
            acc[i+1]=temp;
        }
        last_received_byte--;
        return result;
    }
}
```

Рис. 3.14 Тіло функції get_next_byte_to_transmit

Змінна last_received_byte буде використовуватись як вказівник на останній прийнятий байт. Вона інкрементується кожний раз, коли до акумулятору записується новий байт та декрементується при виклику функції, що розглядається. Початкове значення -1 дає зрозуміти, що у акумуляторі даних немає.

При виклику функції перевіряється значення змінної last_received_byte. Якщо значення не -1, то це означає, що у акумуляторі є дані. Відбувається зчитування даних з першої комірки акумулятора. Після цього відбувається циклічний зсув усіх елементів масиву акумулятора на один ліворуч та декремент змінної, що вказує на останній прийнятий елемент. Це потрібно для того, аби при наступному виклику цієї функції знову зчитати перший елемент при його наявності.

Після розбиття байту даних на частини, викликається функція transmit.

```
void transmit()
{
    if(current_transmission_step == (uint8_t)-1)
    {
        current_transmission_step = transmission_steps_quantity[current_transmission_stage];
        dac_setup();
    }
    else
    {
        dac_setup();
    }
}
```

Рис. 3.15 Тіло функції transmit

В ній перевіряється стан змінної current_transmission_step, що приймає значення номеру наступного переданого значення синусу. Якщо значення змінної дорівнює -1, то це означає, що всі елементи масиву було передано і далі буде передаватись інша частина слова. У такому випадку необхідно до цієї змінної занести значення кількості елементів наступного масиву.

Якщо значення змінної не дорівнює -1, то передача продовжується. Викликається функція dac_setup.

У функції dac_setup відбувається зчитування елементу масиву з синусами відповідно до значення частини слова, що передається, да номеру елементу відповідного масиву.

```
void dac_setup()
{
    mcp4725SetVoltage((int)dac_voltage_koef*stages[current_transmission_stage][current_transmission_step]);
    current_transmission_step--;
    if(current_transmission_step == (uint8_t)-1)
    {
        if(transm_length == 0)
        {
            current_transmission_stage--;
            transm_length = 100;
        }
        else
        {
            transm_length--;
            current_transmission_step = transmission_steps_quantity[current_transmission_stage];
        }
    }
}
```

Рис. 3.16 Тіло функції dac_setup

Після зчитування, елемент помножується на сталий коефіцієнт напруги для ЦАП та передається до вже реалізованої та описаної функції mcp4725SetVoltage. Змінна current_transmission_step декрементується, тому що елемент масиву був прочитаний та використаний для передачі. Далі у тілі функції dac_setup був реалізований механізм подовження часу передачі одного логічного рівня.

3.5 Реалізація алгоритму прийому даних

Реалізовувати механізм прийому було вирішено у файлі main.c, оскільки у ньому знаходиться ключова функція переривання, котре генерується після завершення аналогово-цифрового перетворення.

Задача механізму прийому полягає у вимірі частоти синусоїдального сигналу, що надходить з телефонного каналу та формуванні байтів для передачі по UART до мережевого пристрою.

Функція HAL_ADC_ConvCpltCallback викликається кожні п’ять мікросекунд, тому за допомогою неї теж можна виміряти час. Для цього введемо змінну counter. Ця змінна буде інкрементуватись при кожному виклику функції HAL_ADC_ConvCpltCallback. При множенні цієї змінної на 5x2 мікросекунд, зможемо отримати період синусоїди.

Вимірювати частоту сигналу будемо шляхом відслідковування переходів синусоїди через нуль.

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    if(hadc->Instance == ADC1) //check if the interrupt comes from ACD1
    {
        is_low_state = is_low;
          adc = HAL_ADC_GetValue(&hadc1);
          if (adc < 5 && is_low == 0)
          {
              is_low_checks++;
          }
          if (is_low_checks > checks_number)
          {
              is_low = 1;
            is_low_checks=0;
          }
          if (adc > 5 && is_low == 1)
          {
              is_up_checks++;
          }
          if (is_up_checks > checks_number)
          {
              is_low = 0;
            is_up_checks=0;
          }
          if (is_low_state != is_low)
          {
              is_low_state = is_low;
              if(prev_counter + 1 > counter && prev_counter - 1 < counter){c++;}
              if(c>2){save_counter();}
              counter = 0;
          }
          counter++;
        adc = 0;
    }
}
```

Рис. 3.17 Тіло функції HAL_ADC_ConvCpltCallback

Після виклику функції відбувається перевірка, чи є дані, які можна зчитати з АЦП. Якщо є, то їх буде зчитано у змінну adc. Далі відбувається ряд перевірок для того, щоб виявити перехід через нуль. Щоб збільшити точність, виміри АЦП мають двічі показати перехід через нуль. Такий крок також зменшить кількість помилок при обробці сигналу. Якщо сигнал таки перейшов через нуль, то буде змінено стан змінної is_low. Ця змінна показуватиме теперішній стан сигналу. Попередній стан зберігатиметься у змінній is_low_state.

Після цього відбудеться перевірка, чи змінився теперішній стан сигналу по відношенню до попереднього. Якщо так, то змінній попереднього стану буде надано значення змінної теперішнього стану. Далі, для зменшення кількості помилок, буде використана перевірка, яка спрацює тільки за умови однакових часових інтервалів між переходами. Якщо зафіксовано три однакових по

тривалості переходи, то перед нами певний логічний рівень. Збережемо його, використовуючи функцію save_counter.

У функції save_counter відбувається відновлення прийнятого байту до вигляду, у якому його можна буде передати використовуючи UART інтерфейс до мережевого пристрою.

3.6 Тестування системи

Для тестування системи було виготовлено два штекера. Кожен з них має чотири контакта.

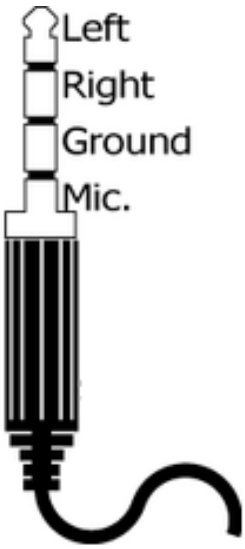

Рис. 3.18 Схема підключення штекеру типу Jack [15]

Загальний провід підлючимо до будь-якого піну, що має позначення GND. Лівий або правий канал з’єднаємо з PA0. Канал мікрофону буде з’єднано з каналом OUT на модулі ЦАП.

Модуль ЦАП з’єднаємо з мікроконтролером наступним чином:

- SCL з PB10;
- SDA з PB11;
- VCC з 3.3V;
- GND з GND.

Для тестування будемо використовувати два USB-to-Serial модулі. Їх контакти потрібно під’єднати так:

– GND з GND;
– TX з PA3;
– RX з PA2.

Більш детально схема електричних з’єднань зображена у додатку 4 ІА51.180БАК.005 Д4.

При підключенні до штекерів типу Jack телефонів, що підтримують підключення гарнітур, на екранах телефонів з’являється спеціальний знак. Зателефонуємо з одного телефону на інший, утворивши, тим самим, два звукових однонаправлених канали передачі даних. Після цього, потрібно подати живлення на мікроконтролери.

Необхідно відкрити обидва COM порти у будь-якій програмі, що надає змогу надсилати та читати дані з UART.

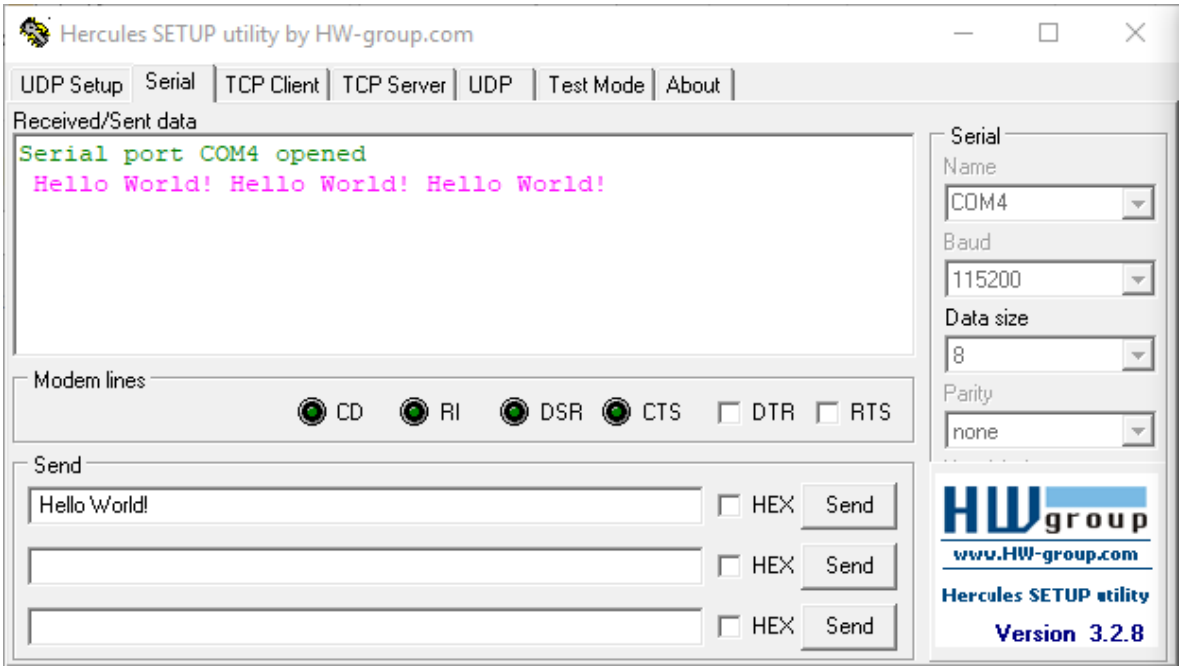

Рис. 3.19 Надсилання «Hello World!» по UART до першого пристрою

Було надіслано 39 байт даних комбінаціями по 13 байт. На приймаючому пристрої спостерігаємо за прийомом.

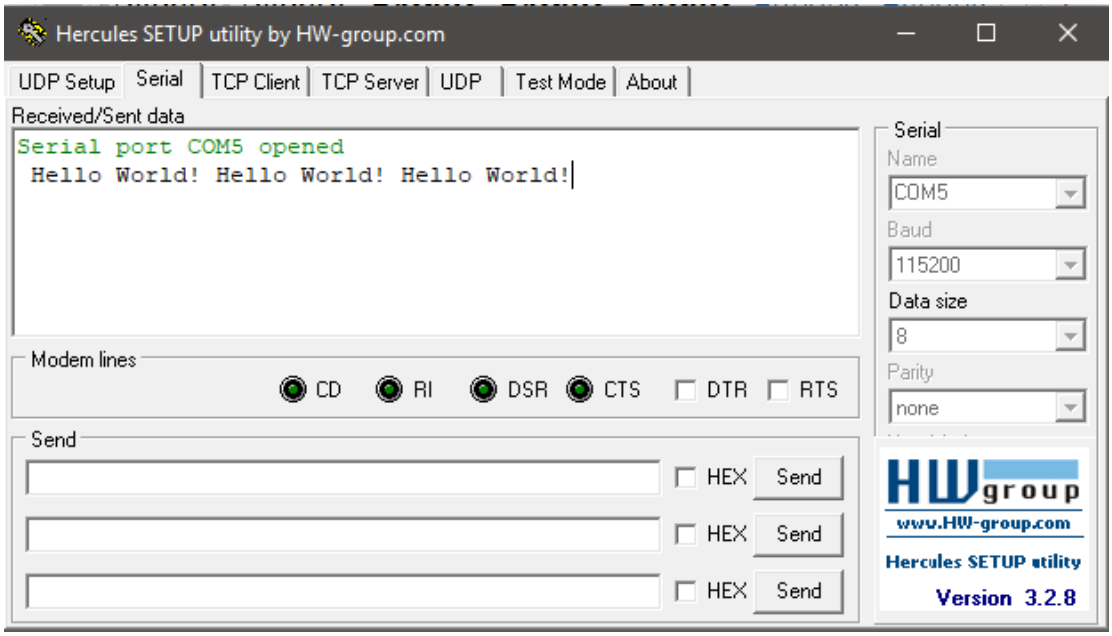

Рис. 3.20 Прийом «Hello World!» по UART до другого пристрою

Проаналізувавши отримані первинні результати, можна сказати, що пристрої працюють правильно. Передача даних відбулась без помилок та у повному обсязі. Було передано 39 символів і стільки ж прийнято.

При передачі можна було візуально помітити затримку передачі даних близько 500 мілісекунд. Це обумовлено особливостями каналу зв’язку. Також, дані, після затримки, з’являлися не миттєво, а по черзі.

Після проведення більш детальних досліджень, було встановлено, що швидкість передачі складає близько трьох символів за секунду. Досить непоганий результат, з огляду на ціну, що за нього треба буде заплатити.

Цієї швидкості вистачить для доступу та обміну даними з консоллю мережевого пристрою Варто лише пам’ятати, що при виклику команд, що мають великий об’єм даних на виході, буде затрачений певний час на їх повну передачу.

Було встановлено, що більш старі телефони використовувати в якості каналоутворюючого обладнання не варто, оскільки вони сильно спотворюють сигнал і зростає кількість помилок. При передачі 1000 символів з використанням старих телефонів правильно отримано було 732.

Ситуація змінюється при використанні сучасних телефонів. При передачі 1000 символів було правильно отримано 971.

Також на кількість помилок при обміні даними впливає сила та якість сигналу. Якщо будь-який з телефонів буде знаходитись у, наприклад, приміщенні з товстими стінами, то похибка зростатиме.

3.7 Інструкція користувачеві

Для резервування зв’язку з мережевим обладнанням потрібна пара розроблених пристроїв. Перший пристрій необхідно встановити поруч з цільовим та з’єднати їх консольним кабелем. На телефоні необхідно активувати функцію автоматичного прийняття дзвінку. Телефон необхідно під’єднати до штекеру пристрою. Блок живлення телефону та пристрою потрібно увімкнути у мережу.

Пристрій, який буде ініціювати зв’язок, необхідно з’єднати з комп’ютером за допомогою USB to TTL конвертера. Його вигляд можна побачити на рисунку 3.21. Телефон під’єднується за допомогою штекеру до пристрою.

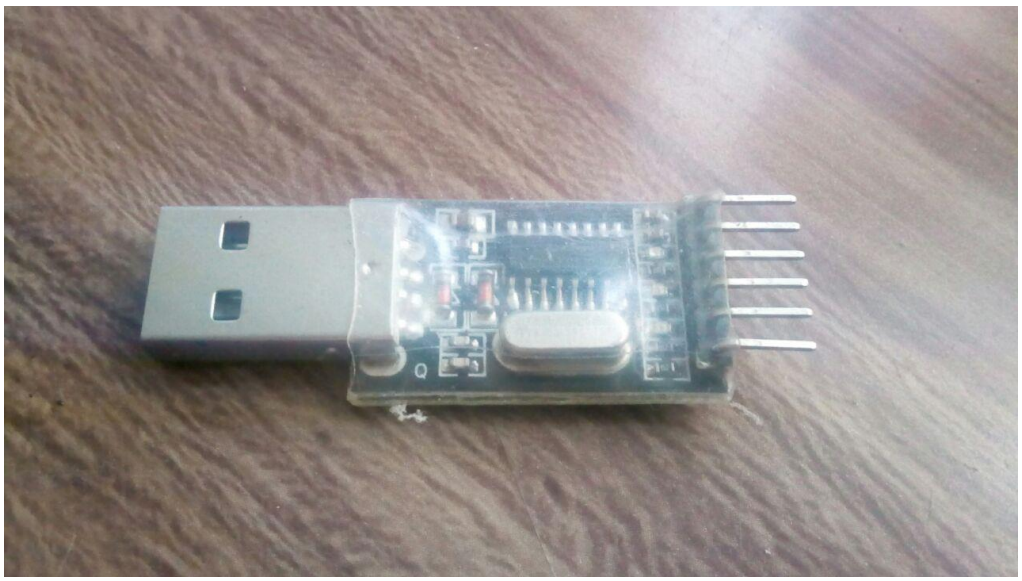

Рис. 3.21 Конвертер

При необхідності здійснити сеанс зв’язку, потрібно зателефонувати на телефон пристрою, що знаходиться поруч з мережевим обладнанням. На комп’ютері необхідно мати встановлену програму, що дозволить надсилати та зчитувати дані з COM порту.

Телефони бажано встановлювати найбільш сучасні (не означає «дорогі»), оскільки при їх використанні зменшується кількість помилок при обміні даними. Варто пам’ятати, що передача великої кількості даних займе певний час.

3.8 Висновки до розділу 3

У результаті виконання третього розділу дипломного проекту було виконано програмну апаратну реалізацію пристрою передачі даних з використанням стільникового зв’язку.

Було обране середовище розробки – STM32CubeMX IDE. Вибір був зроблений з огляду на простоту та необхідний функціонал, що надає дане середовище.

За допомогою генератору коду ініціалізації з використанням бібліотеки HAL був згенерований проект. При початковому налаштуванні було обрано та зконфігуровано периферійні пристрої, що знадобляться для реалізації проекту, а саме аналогово-цифровий та цифро-аналоговий перетворювачі, УАПП, інтерфейс I2C, таймери.

При написанні коду програми для мікроконтролера були враховані результати роботи у попередньому розділі, а саме алгоритми прийому та передачі даних. Завдяки ним, були написані функції, що реалізували механізми прийому та передачі даних.

Було проведене тестування системи, завдяки якому стало зрозуміло, що система працює саме так, як і планувалось. Також тестування виявило деякі особливості роботи стільникової мережі, що були враховані при наданні інструкцій користувачеві.

За допомогою тестування, були визначена точність передачі даних. Відносна похибка коливалась від 2,9% до 26.8% в залежності від умов використання каналоутворюючого обладнання.

ВИСНОВКИ

Під час виконання бакалаврської роботи було виконано розробку апаратних та програмних засобів системи передачі даних, яка дозволяє підтримувати обмін даними з цільовим мережевим пристоєм при втраті з ним зв’язку по основному каналу.

За ціль було поставлено розробити дешевий, простий у користуванні та установці пристрій, що буде допомагати при діагностуванні поломок мережевого обладнання та його віддаленого конфігурування.

Запропоновано використовувати стільниковий зв’язок як дешевий та розповсюджений спосіб обміну даними. У зв’язку з особливостями вибраного середовища передачі, була вирішена задача по перетворенню двійкового представлення даних у звукові коливання.

У рамках дослідження, було розроблене програмне забезпечення для мікроконтролера, який буде виконувати основні операції по передачі та прийому даних. При написанні керуючої програми було максимально задіяно закладені розробниками можливості мікроконтролера та його периферійного обладнання.

При розробці у пристрій було закладено потенціал до модернізації та покращення роботи системи.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Internet World Stats. Usage and Population statistics [Електронний ресурс] // - Режим доступу: https://www.internetworldstats.com/stats.htm

2. «Optical Fiber». The Fiber Optic Association. [Електронний ресурс] // - Режим доступу: www.thefoa.org

3. MobUA.net - зведена карта покриття всіх 4G/3G операторів України, мобільний 4G та 3G інтернет [Електронний ресурс] // - Режим доступу: https://www.mobua.net

4. Реальные скорости 3G Интернета всех операторов мобильной связи – заявленные показатели скорости Интернета 3G. Тесты скорости 3G Интернета, Интернет-трафик [Електронний ресурс] // - Режим доступу: https://mir3g.com.ua/skorost-3g-interneta

5. Bradley Mitchell. «Introduction to Computer Network Topology» [Електронний ресурс] // - Режим доступу: https://www.lifewire.com/computer-network-topology-817884

6. Iridium OpenPort Pilot [Електронний ресурс] // - Режим доступу: https://globsat.ru/iridium/mobilnyy-sputnikovyy-terminal-iridium-openport/mobilnyy-morskoy-sputnikovyy-terminal-iridium-openport-pilot/

7. Частотная модуляция: теория, временная и частотная области [Електронний ресурс] // - Режим доступу: https://radioprog.ru/post/399

8. Multilevel Signaling // August 1995 Hewlett Packard Journal, 21p

9. Пігорілий С. Д., Слободанюк Т. Ф. Программное обеспечение микропроцессорных систем // Техніка, 1985.— 240 с, ил.—Библиогр.: с. 235— 236

10. IAR Embedded Workbench for ARM [Електронний ресурс] // - Режим доступу: https://i .ytimg.com/vi/b_Igq8N0-jQ/maxresdefault.jpg

11. RM0008. Reference manual [Електронний ресурс] // - Режим доступу: https://www.st.com/resource/en/reference_manual/cd00171190.pdf

12. «Principles of Data Acquisition and Conversion» // Texas Instruments. April 2015.

13. C. Gordon Bell, J. Craig Mudge, John E. McNamara, Computer Engineering: A DEC View of Hardware Systems Design, Digital Press, 12 May 2014, 50 стр.

14. «7-bit and 10-bit I2C Slave Addressing» [Електронний ресурс] // - Режим доступу: https://www.totalphase.com/support/articles/200349176

15. «3.5 Jack pinout» [Електронний ресурс] // - Режим доступу: https://cs8.pikabu.ru/post_img/2016/03/24/0/1458766984197547620.png