ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка системи передачі даних на базі мікроконтролерів сімейства STM32»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Артемій МАРЦЕНЮК
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:
здобувач вищої освіти Артемій МАРЦЕНЮК
група ІСДМ-61

Керівник: Ольга ПОЛОНЕВИЧ
науковий ступінь, вчене звання к.т.н., доцент

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС
_____ Каміла СТОРЧАК
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Марценюку Артемію Дмитровичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка системи передачі даних на базі мікроконтролерів сімейства STM32

керівник кваліфікаційної роботи Ольга ПОЛОНЕВИЧ к.т.н., доцент,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023р. №145

2. Строк подання кваліфікаційної роботи «29» грудня 2023р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, принципи розробки вбудованих систем, методи розробки програмного забезпечення вбудованих систем.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження основних аспектів та принципів розробки принципових електронних друкованих плат
Дослідження супутніх компонентів та технологій використаних для створення друкованої плати

Дослідження середовища та мови програмування мікроконтролерів сімейства STM32

Розробка програмного забезпечення системи передачі даних

5. Перелік графічного матеріалу: *презентація*
6. Дата видачі завдання«19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	20.10-03.11.23	
2	Дослідження основних аспектів та принципів розробки принципових електронних друкованих плат	03.11-15.11.23	
3	Дослідження супутніх компонентів та технологій використаних для створення друкованої плати	15.11-19.11.23	
4	Дослідження середовища та мови програмування мікроконтролерів сімейства STM32	19.11-25.11.23	
5	Розробка програмного забезпечення системи передачі даних	25.11-05.12.23	
6	Тестування системи передачі даних та виправлення програмних недоліків	05.12-10.12.23	
7	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.23	
8	Розробка демонстраційних матеріалів	21.12-29.12.23	

Здобувач вищої освіти _____ Артемій МАРЦЕНЮК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Ольга ПОЛОНЕВИЧ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 67 стор., 34 рис., 31 джерела.

Мета роботи – розробка ефективної бездротової системи передачі даних на базі мікроконтролерів STM32.

Об'єкт дослідження – проектування системи передачі даних на базі мікроконтролерів сімейства STM32.

Предмет дослідження – мікроконтролери сімейства STM32.

Короткий зміст роботи: У даній роботі досліджено методи та інструменти розробки друкованих плат, проаналізовано середовища розробки друкованих плат, та розроблено друковану плату для даної роботи на основі попереднього аналізу. Також у даній роботі детально розглянуто процес створення програмного забезпечення для вбудованої системи передачі даних на базі мікроконтролера сімейства STM32 та енергоефективних радіомодулів NRF905. Серед основних переваг розробленої системи можна виділити: швидкість, енергоефективність, бездротова складова, завадостійкість, можливість використання без завад в середовищі з великою кількістю інших бездротових систем працюючих в частотних діапазонах популярних протоколів радіопередачі даних Bluetooth і WI-FI.

КЛЮЧОВІ СЛОВА: STM32, NRF905, МІКРОКОНТРОЛЕР, РАДІОМОДУЛЬ, ВБУДОВАНА СИСТЕМА, ДРУКОВАНА ПЛАТА, СИСТЕМА ПЕРЕДАЧІ ДАНИХ, ЕНЕРГОЕФЕКТИВНІСТЬ, ЗАВАДОСТІЙКІСТЬ, ЧАСТОТА, ДІАПАЗОН ЧАСТОТ.

ABSTRACT

The text part of the qualification work for the master's degree: 67 pages, 34 pictures, 31 sources.

The purpose of the work - to develop an effective wireless data transmission system based on STM32 microcontrollers.

Object of research - designing a data transmission system based on microcontrollers of the STM32 family.

Subject of research - microcontrollers of the STM32 family.

Summary of work: This paper investigates the methods and tools of PCB development, analyzes PCB development environments, and develops a PCB for this work based on the previous analysis. Also, this paper describes in detail the process of creating software for an embedded data transmission system based on the STM32 microcontroller family and energy-efficient NRF905 radio modules. Among the main advantages of the developed system are: speed, energy efficiency, wireless component, noise immunity, the ability to use without interference in an environment with a large number of other wireless systems operating in the frequency bands of popular Bluetooth and WI-FI radio data transmission protocols.

KEYWORDS: STM32, NRF905, MICROCONTROLLER, RADIO MODULE, EMBEDDED SYSTEM, PRINTED CIRCUIT BOARD, DATA TRANSMISSION SYSTEM, ENERGY EFFICIENCY, NOISE IMMUNITY, FREQUENCY, FREQUENCY RANGE.

ЗМІСТ

ВСТУП

Актуальність теми. У світлі швидкої еволюції технологій, системи передачі даних займають центральне місце у функціонуванні різноманітних сфер нашого життя. Вони відіграють ключову роль у телекомунікаціях, промисловості, медицині, автомобільній промисловості та інших галузях, забезпечуючи ефективний обмін інформацією для оптимізації процесів та підвищення рівня автоматизації. Зростання обсягів передачі даних у реальному часі породжує вимогу до розробки більш продуктивних та надійних систем передачі інформації.

У даному контексті виникає необхідність у створенні та оптимізації бездротових систем передачі даних. Вибір мікроконтролерів сімейства STM32 для бази таких систем обумовлений їхніми унікальними технічними характеристиками, які сприяють високій продуктивності та ефективності в умовах обмежених ресурсів. Дослідження цієї теми в рамках даної наукової роботи направлене на подальший розвиток та вдосконалення бездротових технологій з використанням передових мікроконтролерів, що має потенціал змінити підхід до організації систем передачі даних у різних галузях застосування.

У сучасному світі, де технології розвиваються зі швидкістю світла, бездротові системи передачі даних стають все більш важливими. Вони забезпечують зручність і гнучкість в обміні інформацією між пристроями. Мікроконтролери сімейства STM32 відомі своєю ефективністю і надійністю, що робить їх ідеальною основою для розробки таких систем.

Об'єктом дослідження - проектування системи передачі даних на базі мікроконтролерів сімейства STM32.

Предмет дослідження – мікроконтролери сімейства STM32.

Мета і завдання дослідження - розробка ефективної бездротової системи передачі даних на базі мікроконтролерів STM32.

Завданнями дослідження є:

- опис процесу проектування системи передачи даних;

- вибір необхідного обладнання та програмування мікроконтролера;

- тестування і оптимізація системи.

Методика дослідження: для досягнення поставлених завдань використовується комплексна методика дослідження, що включає в себе аналіз літературних джерел, експериментальні дослідження та програмне моделювання системи.

Наукова новизна роботи полягає в розробці та впровадженні бездротової системи передачі даних, яка базується на використанні мікроконтролерів сімейства STM32. Результати дослідження можуть сприяти подальшому розвитку технологій бездротового зв'язку та розширенню їхнього використання в різних галузях.

Практична значущість результатів: отримані результати можуть бути використані для розробки та впровадження нових бездротових систем передачі даних у практиці. Це може знайти застосування в створенні високоефективних та надійних систем моніторингу, управління та обміну інформацією в різних галузях технологій та промисловості.

Апробація результатів магістерської роботи: Марценюк А. Д. «Розробка системи передачі даних на базі мікроконтролерів сімейства STM32» Тези доповіді на Всеукраїнській Науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу». – Київ, 28 листопада 2023 р.

1 ДОСЛІДЖЕННЯ ОСНОВНИХ АСПЕКТІВ ТА ПРИНЦИПІВ РОЗРОБКИ ПРИНЦИПОВИХ ЕЛЕКТРОННИХ ДРУКОВАНИХ ПЛАТ

1.1 Розвиток та популяризація друкованих плат

Друковані плати стали основою сучасної електроніки і відіграють важливу роль у розвитку технологій. Вони забезпечують фізичну платформу для розміщення та з’єднання різних електронних компонентів. З початку їх виникнення в середині 20-го століття, друковані плати пройшли довгий шлях розвитку.

У 1950-х роках було розроблено перші односторонні друковані плати. Вони були простими і складалися з одного шару провідного матеріалу, нанесеного на одну сторону ізоляційної плати. Цей простий дизайн був досить ефективним для ранніх електронних пристроїв.

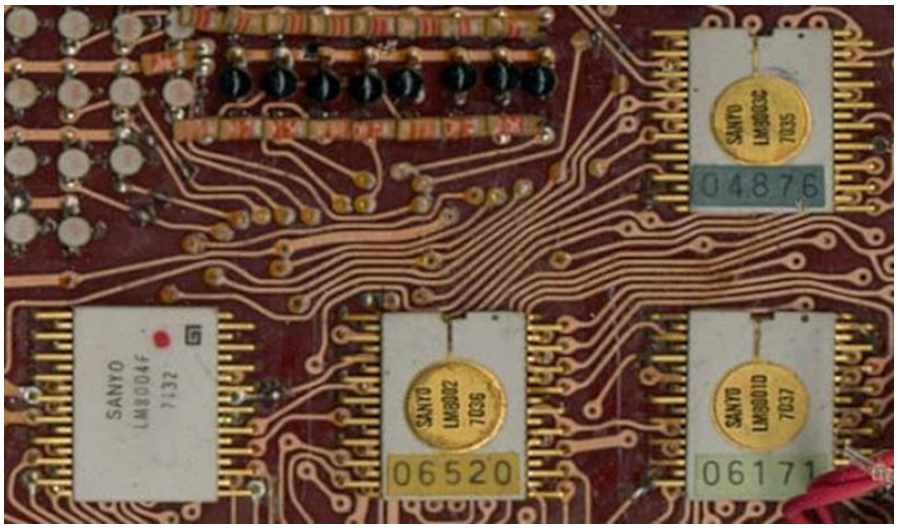


Рис. 1.1 Вигляд перших друкованих плат

З розвитком технологій та збільшенням складності електронних пристроїв з’явилися багатошарові друковані плати. Вони мають декілька шарів провідного матеріалу, розділених ізоляційними шарами. Це дозволяє розмістити більше компонентів на одній платі і забезпечити більш складні з’єднання.

Сьогодні друковані плати використовуються в майже всіх електронних пристроях, від простих годинників до складних комп’ютерів і супутникових систем. Вони є важливою частиною нашого повсякденного життя, хоча ми рідко замислюємося над тим, як вони працюють.

Популяризація друкованих плат сприяла розвитку індустрії електроніки, оскільки вони забезпечують ефективний та надійний спосіб з’єднання електронних компонентів. Вони також сприяли розвитку нових технологій, таких як інтегральні мікросхеми, які можуть бути легко розміщені на друкованих платах.

В майбутньому очікується подальший розвиток технології друкованих плат, зокрема завдяки використанню нових матеріалів і технологій, таких як 3D-друк. Це відкриває нові можливості для дизайну та виробництва електронних пристроїв, що може привести до створення ще більш потужних та компактних пристроїв у майбутньому.

Одним з ключових напрямків розвитку технології друкованих плат є гнучкі друковані плати. Вони використовують гнучкі матеріали, які дозволяють платам згинатися і тягнутися, не втрачаючи своєї функціональності. Це відкриває нові можливості для дизайну електронних пристроїв, особливо в областях, де потрібна гнучкість, таких як носимі пристрої.

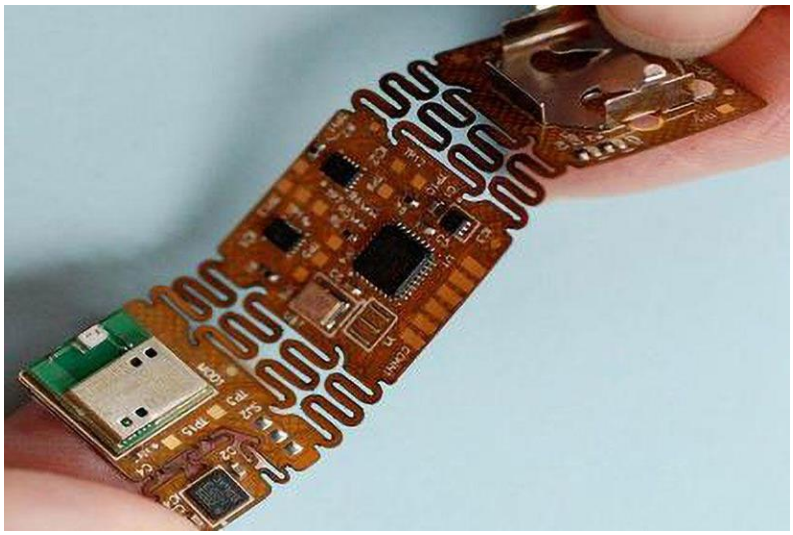


Рис. 1.2 Вигляд гнучкої друкованої плати

Іншим важливим напрямком є друкована електроніка, яка включає в себе виробництво електронних пристроїв за допомогою друкування. Це може включати в себе використання традиційних методів друку, таких як офсетний друк, для створення електронних компонентів на гнучких матеріалах. Це може привести до зниження вартості виробництва електроніки і зробити її доступнішою.

Також варто зазначити роль автоматизації в розвитку друкованих плат. Використання автоматизованих систем для виробництва друкованих плат дозволяє збільшити швидкість виробництва і забезпечити більшу точність. Це особливо важливо для виробництва високотехнологічних пристроїв, де потрібна висока точність і надійність.

Всі ці тенденції свідчать про те, що друковані плати продовжують розвиватися і адаптуватися до змінюваних потреб сучасного світу. Вони залишаються важливою частиною нашого технологічного ландшафту і, безсумнівно, продовжать відігравати ключову роль у майбутніх інноваціях.

1.2 Аналіз та вибір середовища розробки друкованих плат

У процесі розробки принципових електронних друкованих плат, вибір оптимального середовища для розробки відіграє важливу роль у забезпеченні ефективності, точності та продуктивності. Цей аналіз спрямований на розгляд існуючих середовищ розробки друкованих плат, оцінку їх можливостей та функціоналу, а також врахування аспектів відкритості та підтримки користувачів.

Визначення потреб у функціоналі є ключовим етапом при виборі середовища розробки. Аналіз можливостей, таких як автоматизація процесу маршрутизації слідів, імпорт та експорт файлів у стандартні формати, а також підтримка тривимірного моделювання, дозволить обрати інструмент, який найбільш точно відповідає вимогам дослідження.

Важливим аспектом є ступінь відкритості та підтримки спільноти користувачів обраного середовища. Активна спільнота може забезпечити необхідну

підтримку та розвиток інструменту. Розгляд відгуків користувачів та участь у форумах сприятиме уточненню переваг та недоліків кожного середовища.

Ефективна розробка друкованих плат вимагає інтеграції з іншими інструментами, такими як симулятори, віртуальні лабораторії та інші. Важливо вивчити сумісність обраного середовища з іншими засобами розробки для забезпечення безперервності та зручності процесу.

Першочерговим завданням даної роботи є аналіз існуючих середовищ розробки принципових електронних друкованих плат на ринку, таких як Altium Designer, Eagle, KiCad та інші. Кожне середовище має свої переваги та обмеження, які слід ураховувати при виборі відповідно до конкретних вимог та завдань дослідження.

Altium Designer

Переваги: Altium Designer визначається великою функціональністю та широким спектром можливостей для розробки електронних пристроїв. Він надає потужні інструменти для схемотехнічного моделювання, маршрутизації плат, а також аналізу сигналів. Інтеграція з базами даних компонентів із сервісів, таких як Octopart, спрощує вибір та замовлення компонентів.

Недоліки: Одним з недоліків є високий вартість ліцензії Altium Designer, що може бути значущим обмеженням для невеликих команд чи поодиноких розробників. Інтерфейс користувача може виявитися складним для новачків, і потребує певного часу для оволодіння всіма його можливостями.

Eagle

Переваги: Eagle вражає простотою використання та доступністю для новачків. Це безкоштовний інструмент для невеликих проектів, що робить його привабливим для студентів та ентузіастів. Інтеграція з Autodesk Fusion 360 дозволяє легко переходити від схемотехнічного моделювання до створення 3D-моделей.

Недоліки: Однак, Eagle може бути обмеженим для складних та об'ємних проектів. Можливості безкоштовної версії обмежені, і для повноцінного використання деяких функцій слід придбати платну ліцензію.

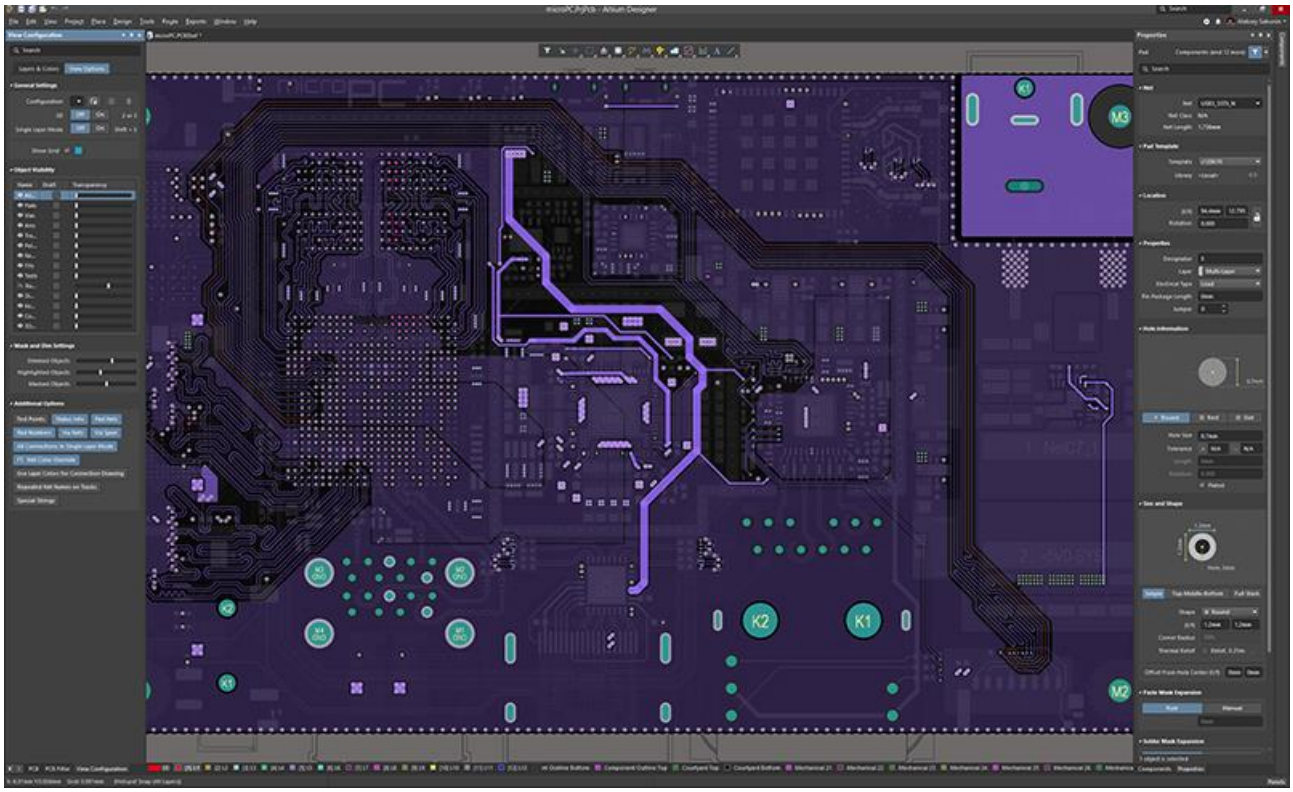


Рис 1.3 Зображення робочого вікна Altium Designer

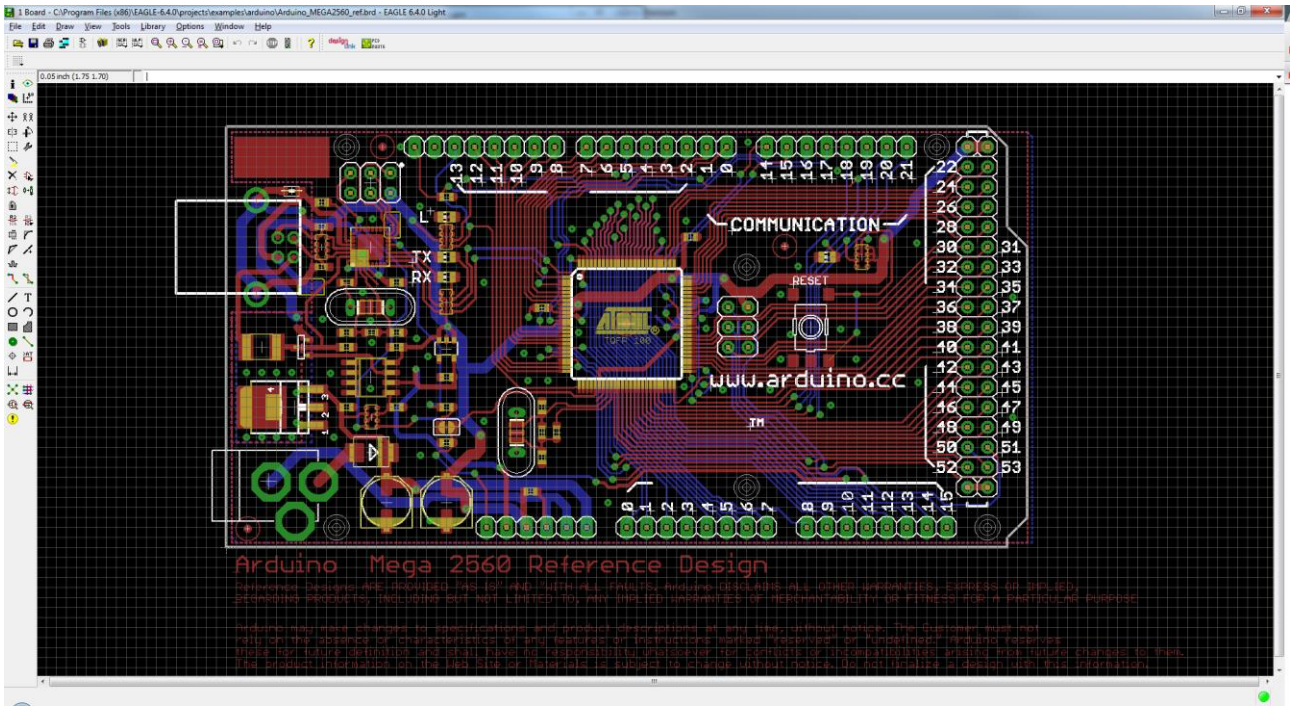


Рис 1.4 Зображення робочого вікна Eagle

KiCad

Переваги: KiCad відзначається відкритістю і безкоштовністю. Велика та активна спільнота користувачів забезпечує підтримку та розвиток програми. У

KiCad також є велика бібліотека компонентів, що дозволяє ефективно приступити до проекту.

Недоліки: Однак інтерфейс користувача може виглядати менш інтуїтивно, порівняно з іншими системами. Деякі користувачі можуть відчувати відсутність певних продвинутих функцій, які пропонують конкуренти. Великі команди можуть зіткнутися із складністю організації проектів в KiCad.

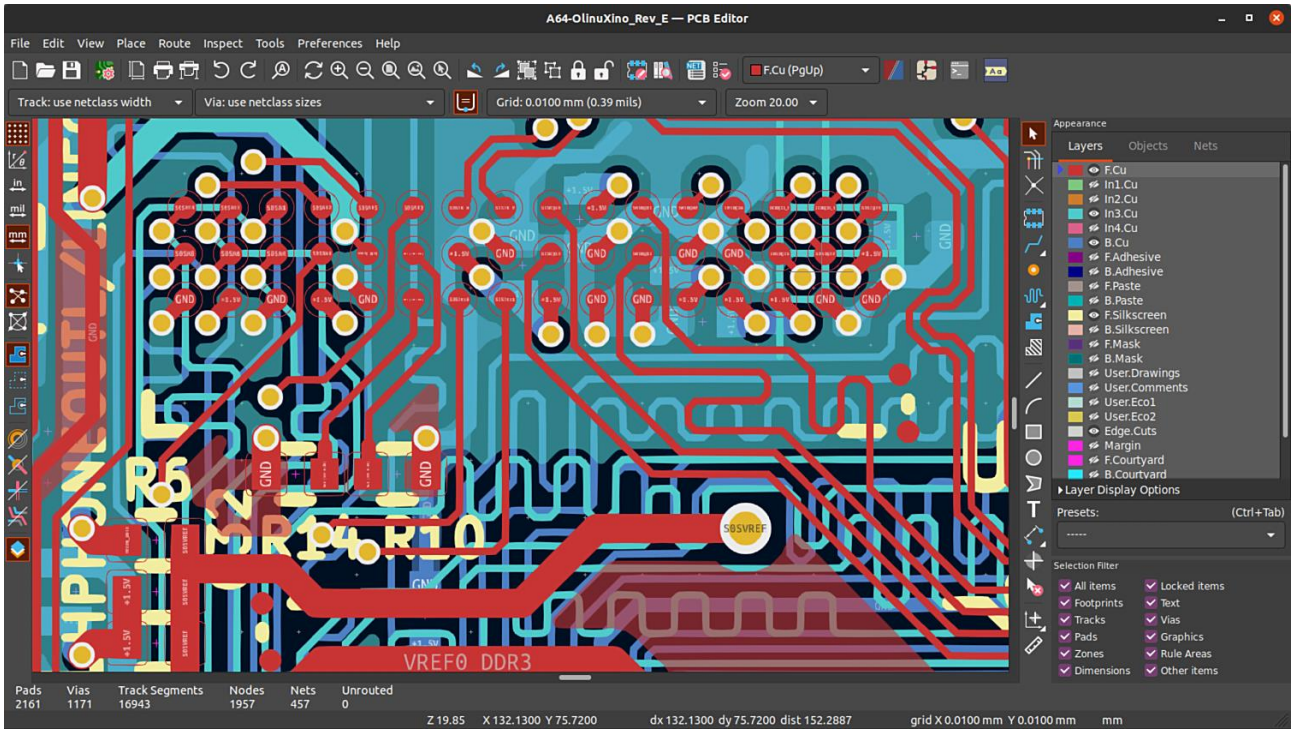


Рис 1.5 Зображення робочого вікна KiCad

На підставі проведеного аналізу та оцінки можливостей кожного середовища розробки, обрано оптимальний інструмент Altium Designer (пробна ліцензія програмного забезпечення), який відповідає визначеним вимогам та принципам дослідження. Серед основних переваг через які було обрано даний інструмент є наявність найбільшої бібліотеки електронних компонентів та функція 3D-візуалізації .

1.3 Створення схемотехніки друкованої плати

Створення схемотехніки друкованої плати – це важливий етап у розробці електронних пристроїв. Друковані плати є основною платформою для монтажу та з'єднання компонентів електронних пристроїв, сприяючи їхньому оптимальному функціонуванню. Для створення схемотехніки друкованої плати потрібно чітко визначитися з набором головних електричних елементів, ознайомитись з їхніми головними характеристиками вказаними виробником. Також потрібні базові знання з елекроніки, тому що багато складних електричних компонентів та мікросхем потребують для коректної роботи додаткового використання супутніх елементів таких як: резистори, конденсатори, діоди індуктивності та інші.

Створення схемотехніки слід розділити на декілька модулів аби в майбутньому легко виявити помилки чи замінити модуль в загальній схемі. Для схемотехніки цього проекту модна виділити такі основні модулі як, «Живлення», «Периферія мікроконтролера», «Периферія мікросхеми RTL8201CP» та «Супутня периферія».

Модуль живлення над важливий під час створення будь якої схемотехніки, адже будь яка помилка в ньому може призвести до незворотніх наслідків а саме: вивести з ладу низку інших компонентів та нанести супутніх збитків.

Для даної роботи схемотехніка живлення виглядатиме наступним чином:

1. Мікросхема LM1117MPX33 стабілізує вихідну напругу до 3.3В необхідних для живлення інших компонентів друкованої плати.

2. Високоємнісні конденсатори C1 та C4 використовуються для вирівнювання напруги а низькоємнісні конденсатори C2 та C8 використовуються для приглушення небажаних імпульсів.

3. Світлодіод VD3 сигналізує про наявність живлення на друкованій платі.

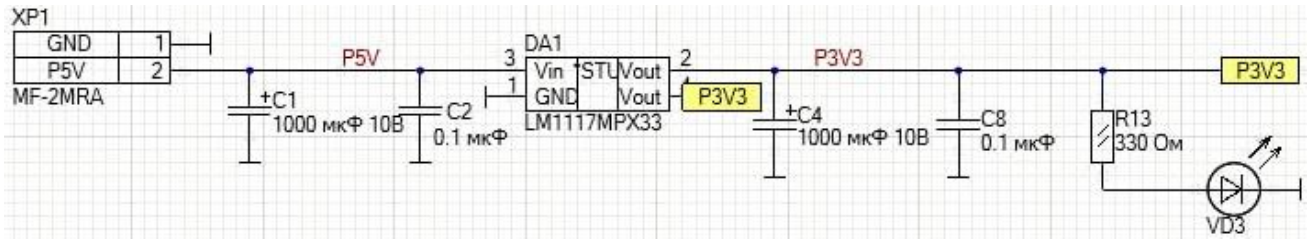


Рис 1.6 Схемотехніка живлення друкованої плати

До периферії мікроконтролера в даному випадку відноситься кварцевий генератор, роз'єм програмування за технологією JTAG, обв'язка з конденсаторів для стабільного живлення без стрибків напруги та небажаних імпульсів.

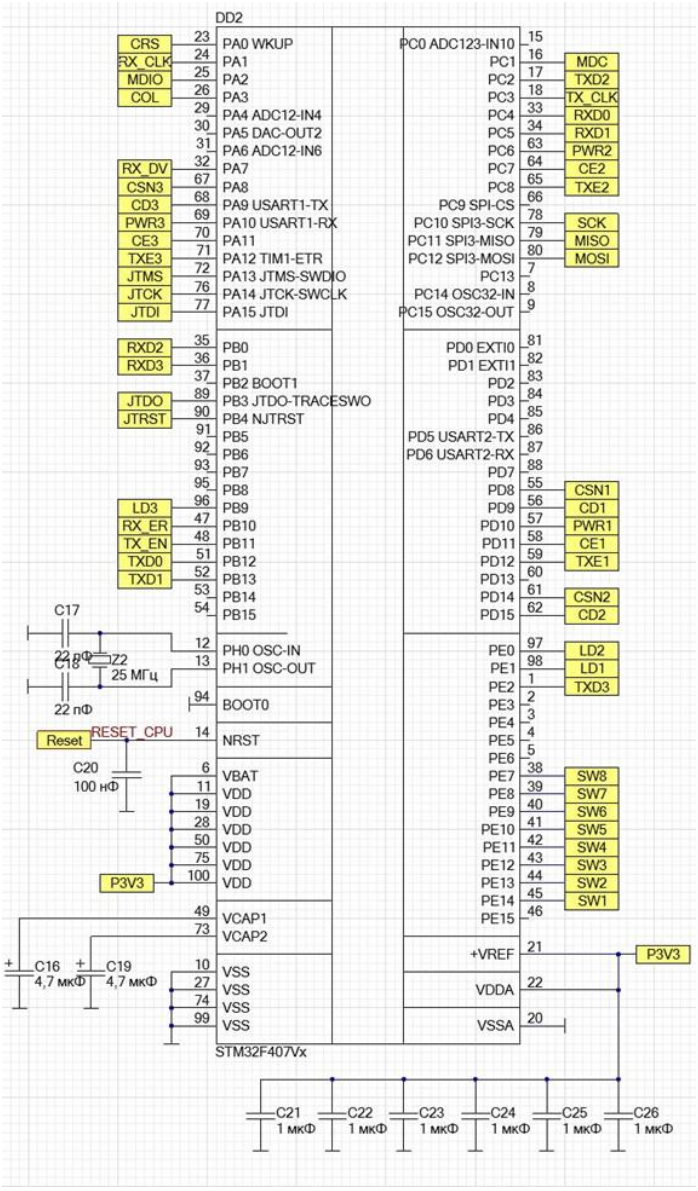


Рис 1.7 Схемотехніка мікроконтролера STM32F407VET6

Розв’язка мікросхеми RTL8201CP створена згідно з документацією до даної мікросхеми і має наступний вигляд:

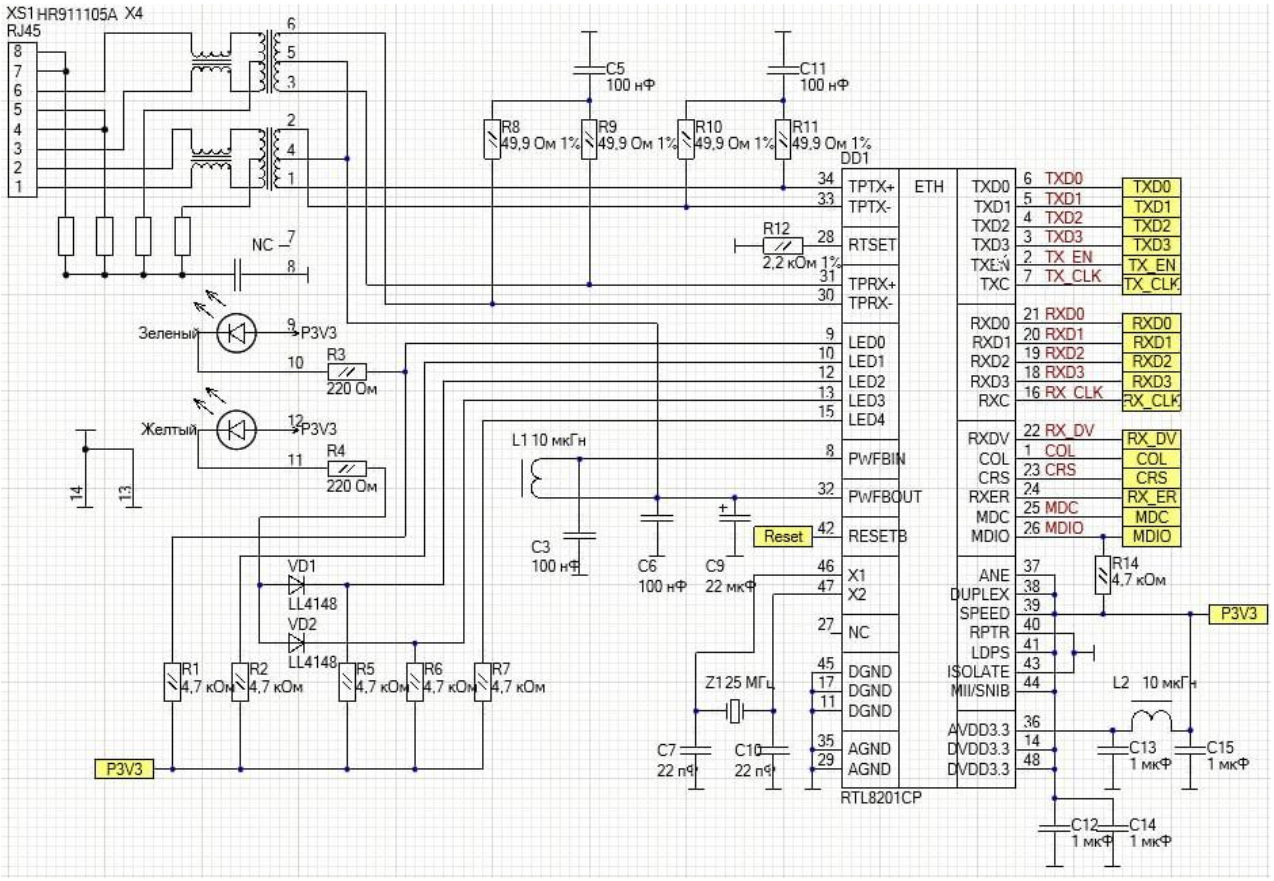


Рис 1.8 Схемотехніка мікросхеми RTL8201CP

Супутня периферія включає в себе:

1. Перемикач SW1 слугує для налаштування мережевої адреси та робочих частот радіомодулів.

2. Світлодіоди VD4, VD5, VD6 використовуються для індикації передачі та прийому даних радіомодулів.

3. Роз’єми J1, J2, J3 мають пінування для підключення радіомодулів NRF905

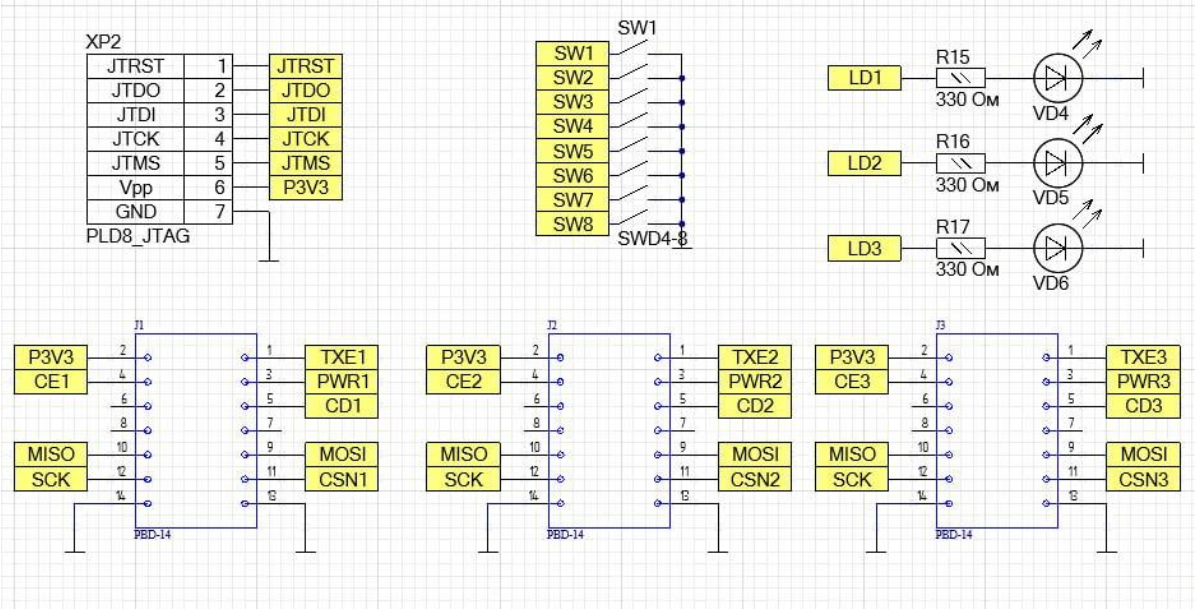


Рис 1.9 Схемотехніка супутньої периферії

1.4 Розробка друкованої плати

Розробка друкованої плати полягає в створенні шарів плати, розміщення на них електричних компонентів, трасуванню провідників та перехідних отворів між компонентами згідно з розробленою раніше схемотехнікою. Для початку розробки друкованої плати слід визначитися з розмірами та бажаним виглядом майбутнього виробу, за потреби розмістити отвори для кріплення, якщо потрібно максимально зменшити розміри плати слід збільшити кількість сигнальних шарів майбутньої плати. Друкована плата складається з кількох шарів, кожен з яких виконує певну функцію і має свої особливості. Основні шари, що зустрічаються у багатьох проектах, включають наступні:

Signal Layer (Шар сигналів): Це основний шар, на якому розташовуються провідники, що передають електричні сигнали між компонентами плати. Кількість шарів сигналів може варіюватися в залежності від складності проекту.

Power and Ground Plane Layers (Шари живлення і заземлення): Ці шари використовуються для живлення компонентів та забезпечення стабільного заземлення. Вони зазвичай знаходяться у найнижчому та найвищому шарах плати і використовуються для створення площадок живлення та заземлення.

Internal Signal Layers (Внутрішні шари сигналів): У складних платах можуть бути використані додаткові внутрішні шари сигналів для розміщення провідників. Це дозволяє зменшити перетинання та електромагнітну замішаність між сигналами, а також поліпшити шляхи трасування.

Solder Mask Layer (Шар маски паяльної пасти): Цей шар використовується для захисту провідників від ненавмисного контакту з паяльною пастою під час процесу монтажу компонентів.

Silkscreen Layer (Шар шовкодруку): Шар шовкодруку використовується для нанесення текстових міток, символів та інших інформаційних елементів на поверхню плати. Це може включати назви компонентів, значки положення, індикатори полюсності тощо.

Крім основних шарів, також можуть використовуватися додаткові шари, такі як шари для трасування диференційних пар, шари для розміщення вбудованих компонентів (наприклад, вбудовані конденсатори), шари для захисту від ЕМП-випромінювання, шари для захисту від зовнішніх впливів тощо.

Для трасування друкованої плати в програмі Altium Designer є безліч інструментів. Ось декілька з них:

Interactive Routing (Інтерактивне трасування): Це основний інструмент для трасування плат в Altium Designer. Для початку потрібно обрати шар та початкову точку трасування, а потім вести трасування між відповідними пінами або компонентами на платі. Інтерактивне трасування дає повний контроль над лініями трас та кутами, використовуючи різні режими трасування (90 градусів, 45 градусів, вільний кут тощо).

Auto-Interactive Router (Автоматичне інтерактивне трасування): Цей інструмент використовує автоматизований підхід до трасування плат. Для початку потрібно обрати початкову точку та напрям трасування, і Altium Designer автоматично трасує лінії трас, враховуючи обмеження, такі як правила мінімальної ширини траси, кліренсу, шарів тощо.

Glossing (Глосування): Цей інструмент дозволяє оптимізувати трасування плат, згладжуючи та оптимізуючи їх форму та кути. Він автоматично аналізує вже

трасовані лінії трас і змінює їх форму для поліпшення прохідності сигналу та відповідності правилам трасування.

Differential Pair Routing (Трасування диференційних пар): Altium Designer також має спеціальні інструменти для трасування диференційних пар, які вимагають точного збереження парного розташування та паритету сигналів.

За замовчуванням слід користуватись інтерактивним трасуванням тому що даний інструмент найбільш універсальний та дозволяє користувачеві самостійно прокладати оптимальні маршрути доріжок з урахуванням встановлених правил трасування друкованих плат.

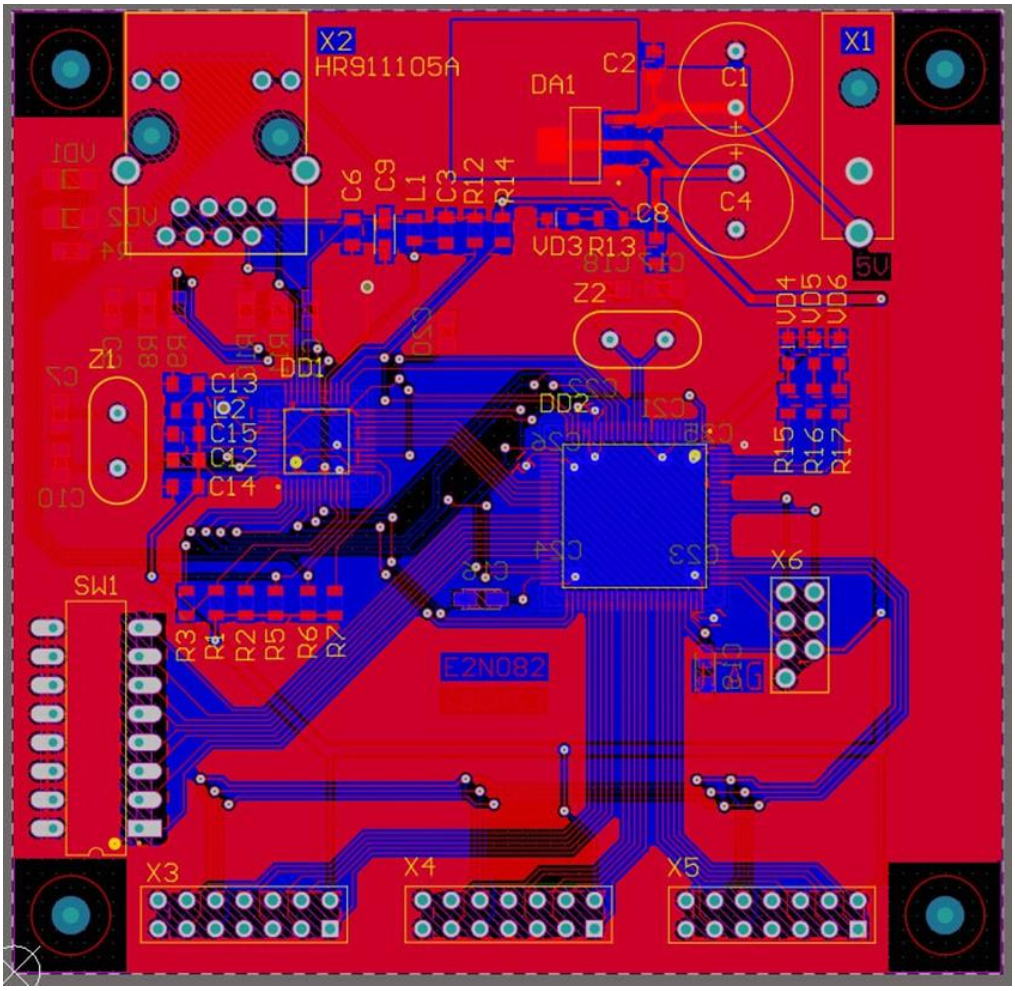


Рис 1.10 Створення друкованої плати

На Рис 1.10 зображено вигляд плати в середовищі розробки Altium Designer. Червоним кольором зображується верхній сигнальний шар, а синім – нижній. Голубий колір позначає отвори для кріплення та отвори для монтажу елементів.

Сірим кольором зображено між шарові поля для пайки та між шарові переходи, як правило між шарові переходи супроводжуються отворами. Жовтим кольором позначено шар шовкографії, або шовкодруку, даний шар дозволяє маркувати компоненти на платі та вносити інформативні позначення. Фіолетовим кольором зображено межі друкованої плати. Дані кольори слугують маркерами за замовчуванням для програми Altium Designer, але за потреби дані кольори можливо змінити на будь які обрані користувачем за бажанням.

В даній друкованій платі верхній та нижній сигнальні шари залиті земляними(GND) полігонами. Земляні полігони, також відомі як заземлювальні області або області заземлення, є важливою частиною друкованих плат. Головна їхня функція полягає в забезпеченні електричного з'єднання землі між різними елементами друкованої плати і компонентами, які вони обслуговують.

Основні завдання земляних полігонів у друкованих платах включають:

- Заземлення: Забезпечення низького опору для відведення струму на землю. Це дозволяє зменшити електричні шуми, електромагнітні перешкоди та інші проблеми, пов'язані з електричною безпекою та електромагнітною сумісністю.
- Розсіювання тепла: Великі земляні полігони можуть використовуватися для поліпшення розсіювання тепла від деяких компонентів, таких як потужні транзистори чи інші елементи, які виробляють тепло.
- З'єднання між шарами: Земляні полігони можуть бути використані для з'єднання заземлення між різними шарами друкованої плати.
- Зменшення електромагнітних перешкод: Земляні полігони можуть виграти роль у зменшенні електромагнітних перешкод внаслідок внутрішньої збудженості.
- Підтримка схеми: Земляні полігони можуть використовуватися для підтримки маршрутизації слідів і підвищення механічної міцності друкованої плати.

- Загалом, правильне проектування земляних полігонів є важливою аспектом розробки друкованих плат, що дозволяє забезпечити стабільну роботу електронних пристроїв і забезпечити електричну надійність системи.

Також важливим елементом при трасуванні друкованих плат є дотримання мінімальних відстаней між компонентами, елементами, доріжками полігонами масками та ін. Для цього існує інструмент Design Rule Check(DRC або перевірка правил проектування). Даний інструмент аналізує розробку відносно раніше встановлених правил проектування та вказує користувачеві на помилки допущені при проектуванні друкованої плати.

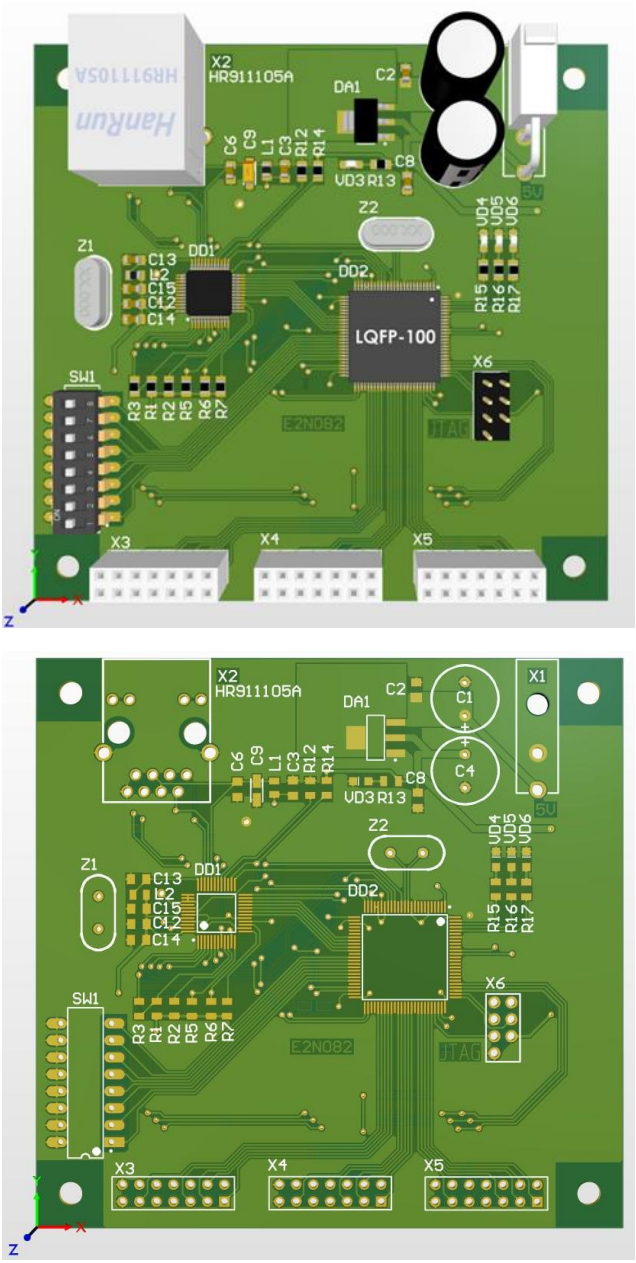


Рис.1.11. 3D вигляд друкованої плати: а) без елементів б) з елементами

2 ДОСЛІДЖЕННЯ СУПУТНІХ КОМПОНЕНТІВ ТА ТЕХНОЛОГІЙ ВИКОРИСТАНИХ ДЛЯ СТВОРЕННЯ ДРУКОВАНОЇ ПЛАТИ

2.1 Аналіз та вибір мікроконтролера

Мікроконтролери є ключовим елементом у вбудованих системах та електроніці загалом. Вони використовуються для керування та моніторингу різних пристроїв, від простих домашніх пристроїв до складних промислових систем. Правильний вибір мікроконтролера визначає успіх розробки проекту, ефективність та надійність системи. У даній доповіді розглянемо процес аналізу та вибору мікроконтролера.

Першим кроком у виборі мікроконтролера є визначення вимог до системи. Це включає в себе функціональність, швидкість обробки даних, вимоги до введення-виведення, енергоефективність та інші технічні параметри. Чітке розуміння цих вимог допомагає обмежити кількість потенційних варіантів мікроконтролерів.

Після визначення вимог проводиться аналіз технічних характеристик доступних мікроконтролерів. Серед ключових параметрів слід враховувати такі:

- Архітектура: RISC або CISC, відомі виробники (наприклад, ARM, STM32, AVR, PIC).
- Частота та продуктивність: Швидкість обробки даних, кількість ядер, можливості апаратного прискорення.
- Введення-виведення: Кількість та типи портів, можливості роботи з різними пристроями.
- Енергоефективність: Важлива для мобільних пристроїв та пристроїв з обмеженим живленням.
- Розмір пам'яті: Обсяг оперативної та постійної пам'яті для програм та даних.

Для даного проекту оптимальним рішенням було обрано мікроконтролер STM32F407, його потужності та програмних функцій вистачатиме для одночасної

роботи швидкісного інтерфейсу передачі даних SPI та одночасної роботи зі стеком технологій TCP/IP.

Рис 2.1 Вигляд мікроконтролера STM32F407IGT6

Мікроконтролер STM32F407 є потужним електронним компонентом з ядром ARM Cortex-M4 та численними перевагами для вбудованих систем. Здатен працювати на високій частоті до 100 МГц, що робить його високопродуктивним для різноманітних завдань, також варто заздалегіть подбати про запас продуктивності мікроконтролера для подальшого удосконалення та масштабування проекту.

Його пам'ять включає 512 КБ флеш-пам'яті для зберігання програм та 128 КБ оперативної пам'яті для ефективного виконання завдань. Має різноманітні порти введення-виведення, включаючи GPIO, USART, SPI, I2C, ADC та інші, що надає гнучкість для взаємодії з різними пристроями.

Однією з ключових переваг є його вбудована підтримка DSP і FPU, що дозволяє використовувати апаратне прискорення для операцій з плаваючою комою та обробки цифрового сигналу. Важливо відзначити його енергоефективність та розумне керування живленням, що робить STM32F407 ідеальним для пристроїв з обмеженим живленням.

Загальною перевагою є його готовність до розширення та широкий набір інструментів розробки, які сприяють ефективній та швидкій розробці пристроїв.

STM32F407 виступає як надійна та потужна основа для реалізації різноманітних проектів в сфері вбудованих систем та електроніки.

2.2 Використання зв’язку TCP/IP

У світі вбудованих систем і мікроконтролерів STM32 використання зв’язку TCP/IP стало невід’ємною частиною для забезпечення ефективного обміну даними між пристроями. Використання протоколу TCP/IP дозволяє мікроконтролерам взаємодіяти в мережі, що відкриває безліч можливостей для забезпечення зручності та ефективності систем.

Один з основних принципів використання зв’язку TCP/IP на мікроконтролерах STM32 - це можливість побудови клієнт-серверних застосунків. Наприклад, STM32 може діяти як сервер і надавати послуги іншим пристроям у мережі. Це може бути зчитування даних з датчиків, управління пристроями або навіть дистанційне програмування.

Однією з ключових переваг використання TCP/IP є надійність передачі даних. Протокол TCP забезпечує гарантовану доставку пакетів, а також управління потоком і перевірку цілісності даних. Це особливо важливо в задачах, де точність і надійність даних мають високий пріоритет, таких як системи моніторингу або керування.

Ще однією вагомою перевагою є можливість використання інтернет-протоколів для взаємодії зі світом. Наприклад, вбудовані пристрої STM32 можуть взаємодіяти з хмарними сервісами, обмінюватися даними з віддаленими серверами або використовувати принципи Інтернету речей (IoT).

Використання зв’язку TCP/IP та мікросхеми RTL8201CP для мікроконтролерів STM32 дозволяє створювати ефективні та потужні мережеві рішення вбудованими системами. Забезпечуючи здатність передачі даних через мережу з використанням протоколу TCP/IP, це поєднання дозволяє здійснювати комунікацію між STM32 та іншими пристроями у мережі, що є критичним для розвитку Інтернету речей (IoT) та інших застосувань.

Мікросхема RTL8201CP виконує функцію фізичного рівня (PHY) для забезпечення передачі даних по мережі. Вона підтримує стандарти Ethernet і Fast Ethernet, що робить її ідеальною для використання в мікроконтролерах STM32, які мають обмежений обсяг ресурсів. Завдяки своїй ефективності та низькому енергоспоживанню, RTL8201CP стає ключовою складовою для створення енергоефективних мережевих рішень.

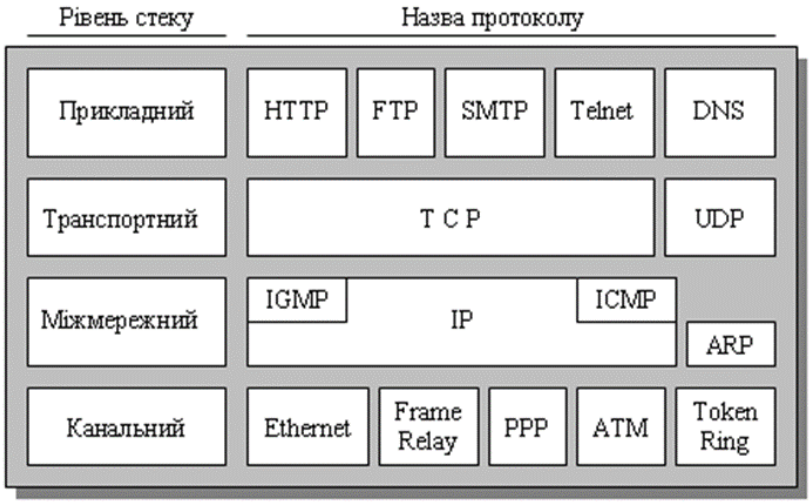


Рис 2.2 Стек протоколів TCP/IP

Використання протоколу TCP/IP (Transmission Control Protocol/Internet Protocol) дозволяє забезпечити надійний та послідовний обмін даними між мікроконтролерами та іншими пристроями в мережі. TCP гарантує доставку даних без втрат та у правильному порядку, забезпечуючи стабільну комунікацію. IP визначає адресу та маршрутизацію даних в мережі, що робить можливим взаємодію між різними пристроями на великій території.

Інтеграція цих технологій в мікроконтролери STM32 відкриває широкі можливості для розробників у сферах IoT, автоматизації та вбудованих систем. Вони можуть створювати продукти, які можуть здійснювати збір та обробку даних, взаємодіяти з іншими пристроями в мережі та надавати користувачам ефективний інтерфейс взаємодії через мережу.

Загалом, поєднання зв’язку TCP/IP та мікросхеми RTL8201CP для мікроконтролерів STM32 є важливим етапом у розвитку вбудованих систем, що

відкриває нові можливості для реалізації різноманітних мережевих застосувань з використанням потужностей мікроконтролерів.

2.3 Основні принципи роботи з радіопередавачем NRF905

NRF905 - це модуль-радіопередавач, що працює в трьох міжнародних ISM діапазонах: 433, 868 та 915 МГц. Ці діапазони не вимагають ліцензії для використання, тому модуль NRF905 підходить для створення бездротових пристроїв, що обмінюються даними на відстані до 1000 метрів на відкритій місцевості(в ідеальних умовах без радіозавад та антеною з коефіцієнтом підсилення 5дБ та більше). В умовах міста та з використанням комплектної антени під час тестування вдалося отримати результат біля 30 метрів в приміщенні та 50 метрів на вулиці. Можливо дані результати не вражаючі але для даного проекту вони цілком прийнятні.

Рисунок 2.3 Радіомодуль NRF905

Модуль NRF905 має інтерфейс SPI, який дозволяє підключати його до мікроконтролерів, таких як Arduino, AVR, PIC та інших. Модуль NRF905 використовує власний протокол передачі даних фірми Nordic, який забезпечує високу швидкість (до 76,8 кбіт/с), низьке споживання енергії (1 мкА в режимі

очікування) та високу стабільність частоти (PLL). Також завдяки власному протоколу передачі даних даний модуль захищає дані користувача від несанкціонованого прослуховування в ефірі. Модуль NRF905 може працювати як передавач або приймач, залежно від налаштувань регістрів.

Модуль NRF905 має два режими роботи: режим конфігурації та режим передачі/прийому. В режимі конфігурації модуль NRF905 приймає команди від мікроконтролера по інтерфейсу SPI, які визначають параметри роботи модуля, такі як частота, потужність, довжина пакета, адреса тощо. В режимі передачі/прийому модуль NRF905 виконує функції радіопередавача або радіоприймача, залежно від стану вхідного піна TXEN. Якщо TXEN = 1, модуль NRF905 передає дані, які надійшли по інтерфейсу SPI, по радіо. Якщо TXEN = 0, модуль NRF905 приймає дані по радіо та виводить їх по інтерфейсу SPI.

Для підключення модуля NRF905 до мікроконтролера потрібно використовувати п'ять провідників: VCC, GND, SCK, MOSI, MISO, а також два піни управління: CE та TXEN. Для роботи модуля NRF905 потрібно живлення від 1,9 до 3,6 В, але оптимальною є напруга 3,3 В. Модуль NRF905 має розміри 32 х 18 х 63 мм та вагу 9,2 грама.

Під час розробки системи передачі даних постав вибір між радіомодулем і мікросхемою NRF905, в результаті було обрано саме радіомодуль через його нижчу вартість і доступність. Також перевагою радіомодулів є наявність інших електронних компонентів для функціонування радіомодуля.

3 ДОСЛІДЖЕННЯ СЕРЕДОВИЩА ТА МОВИ ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ СІМЕЙСТВА STM32

3.1 Мови програмування мікроконтролерів

Мови програмування для мікроконтролерів відіграють важливу роль у розробці вбудованих систем та електроніки. Оскільки мікроконтролери зазвичай використовуються для виконання конкретних завдань в обмеженому середовищі, вибір правильної мови може значно вплинути на ефективність, продуктивність та зручність розробки.

Мова асемблера є однією з найбільш близьких до апаратного рівня і часто використовується для програмування мікроконтролерів. Вона дозволяє точно контролювати кожен байт пам'яті та кожну команду процесора, що особливо важливо при оптимізації роботи систем з обмеженими ресурсами.

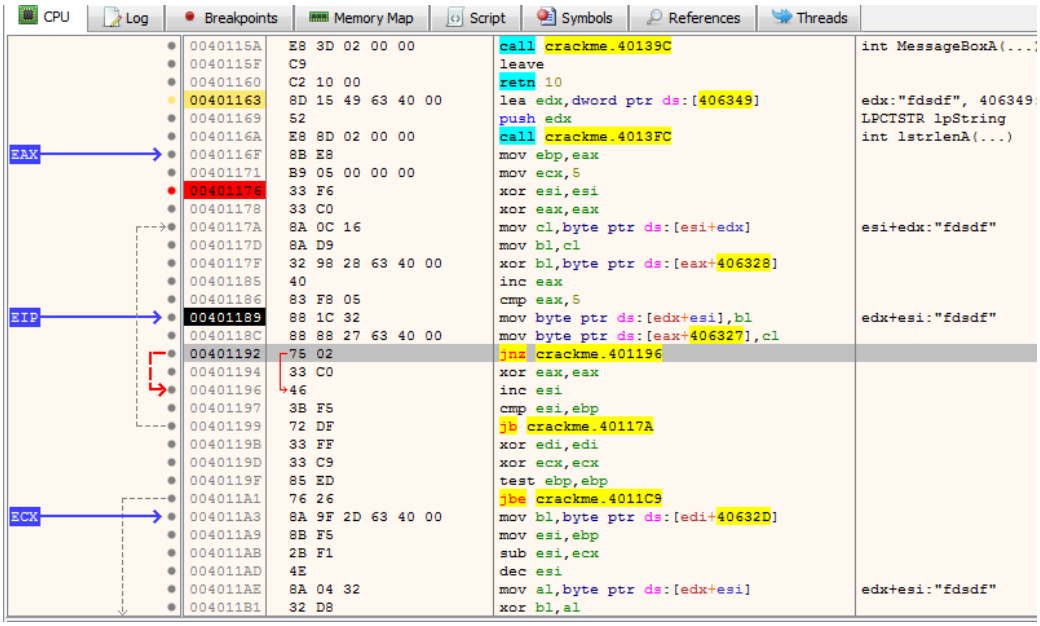


Рис 3.1 Приклад програмування мовою асемблера

З метою полегшення розробки та забезпечення більш високого рівня абстракції, часто використовуються мови вищого рівня, такі як C або C++. Вони дозволяють розробникам працювати на більш високому рівні абстракції, що

полегшує розробку складних програм та сприяє переносу коду між різними апаратними платформами.

Python, зазвичай використовуваний для веб-розробки та аналізу даних, також отримав популярність для програмування мікроконтролерів завдяки своїй простоті та зручності використання. Він дозволяє швидко прототипувати та використовувати високорівневі функції, але може потребувати додаткових ресурсів.

Програмування мікроконтролерів STM32 вимагає глибокого розуміння характеристик цієї сімейства мікроконтролерів та вибір оптимальної мови програмування для досягнення ефективності та продуктивності. Однією з основних мов для роботи з STM32 є мова програмування C.

Мова C відома своєю ефективністю та можливістю працювати безпосередньо з апаратним рівнем. Вона надає розробникам широкі можливості для оптимізації коду та прямого доступу до функцій мікроконтролера. Багато ресурсів та бібліотек для STM32 написані саме на мові C, сприяючи швидкому розвитку вбудованих систем.

Однак для розширення можливостей та полегшення розробки, розробники також використовують мову C++. C++ дозволяє використовувати об'єктно-орієнтований підхід та забезпечує більшу абстракцію для розробників. Використання C++ може бути особливо корисним для складних проектів та командної роботи, де важлива читабельність та підтримка великих кодових баз.

Окрім того, для розвитку високопродуктивних та надійних систем на мікроконтролерах STM32 використовують мови Ada та Rust. Ada славиться своєю високою рівнем безпеки та стійкістю до помилок, що може бути важливим для критичних застосувань. Rust також визначається безпечністю та захистом від гонок, що може забезпечити високий рівень стабільності у вбудованих системах.

Загалом, вибір мови програмування для мікроконтролерів STM32 залежить від конкретних вимог проекту, а також від зручності та досвіду розробників. Кожна мова має свої переваги та особливості, і правильний вибір може значно полегшити розробку та підтримку вбудованих систем на основі STM32.

3.2 Вибір середовища програмування мікроконтролерів

Вибір середовища програмування для мікроконтролерів STM32 може залежати від різних факторів, включаючи вимоги до проекту, досвід розробника та доступність ресурсів.

Одним з найпопулярніших середовищ програмування для STM32 є STM32CubeIDE, яке базується на Eclipse та включає в себе компілятор GCC. STM32CubeIDE підтримує всі сімейства STM32 і надає графічний інтерфейс для налаштування периферії мікроконтролера.

Іншим популярним вибором є Keil uVision, який є потужним середовищем програмування з вбудованим компілятором ARMCC. Keil uVision відрізняється високою продуктивністю та широким набором функцій, але він є платним для деяких варіантів STM32.

IAR Embedded Workbench - це ще одне потужне середовище програмування, яке підтримує STM32. Воно включає в себе компілятор IAR C/C++ та набір інструментів для налагодження. IAR Embedded Workbench відомий своєю ефективністю та швидкістю оптимізації коду, але, як і Keil uVision, він є платним.

Вибір середовища програмування для STM32 в кінцевому підсумку залежить від конкретних потреб та обставин. Розробники повинні враховувати такі фактори, як вартість, вимоги до проекту, досвід розробника та доступність підтримки та ресурсів при виборі найкращого середовища програмування для своїх проектів на STM32.

Давайте розглянемо переваги та недоліки деяких з найпопулярніших середовищ програмування для мікроконтролерів STM32.

1. Keil uVision

- Переваги: Keil uVision має потужні інструменти для налагодження, велику кількість вбудованих бібліотек та функцій, що можуть бути корисними для великих проектів.

- Недоліки: Keil uVision може бути складним для новачків через велику кількість функцій та налаштувань. Також це платне середовище, хоча існує безкоштовна версія з обмеженим обсягом пам'яті.

2. IAR Embedded Workbench

- Переваги: IAR Embedded Workbench також має потужні інструменти для налагодження та багато вбудованих бібліотек. Він також відомий своїм ефективним компілятором, який може генерувати дуже компактний та ефективний код.

- Недоліки: IAR Embedded Workbench - це платне середовище, і його повна версія може бути дорогою. Безкоштовна версія має обмеження на розмір коду.

3. STM32CubeIDE

- Переваги: STM32CubeIDE - це безкоштовне середовище, яке має простий та інтуїтивно зрозумілий інтерфейс. Воно ідеально підходить для новачків або для менших проектів.

- Недоліки: STM32CubeIDE може не мати всіх функцій та бібліотек, які можуть бути доступні в більш розширених середовищах, таких як Keil або IAR.

4. System Workbench for STM32

- Переваги: System Workbench for STM32 - це ще одне безкоштовне середовище, яке має простий інтерфейс та легко встановлюється. Воно також має добре документовану підтримку та активну спільноту.

- Недоліки: Як і STM32CubeIDE, System Workbench може не мати всіх функцій, які доступні в більш розширених середовищах.

Незалежно від вибору, важливо пам'ятати, що найкраще середовище програмування - це те, яке найкраще підходить для ваших потреб та вподобань. Вибір правильного середовища може значно спростити процес розробки та зробити його більш продуктивним.

3.3 Вибір бібліотек для роботи з мікроконтролерами STM32

Мікроконтролери STM32 компанії STMicroelectronics, побудовані на ядрі ARM Cortex, є одними з найпопулярніших у світі. Їх використовують у системах

контролю та управління, системах обробки та аналізу даних. Основна перевага цих мікроконтролерів полягає в їх потужності та універсальності. Правильний вибір бібліотек для роботи з мікроконтролером може значно спростити розробку рішень будь якої складності а також запобігти виникненню критичних помилок в роботі мікроконтролера та систем якими він керує.

Існує велика кількість бібліотек для роботи з мікроконтролерами STM32, з них варто виділити бібліотеку STM32Cube HAL (Hardware Abstraction Layer) від розробника даних мікроконтролерів . Ця бібліотека, яка надає абстракцію від апаратного рівня, що робить розробку більш простою та переносною між різними пристроями STM32. Основні характеристики та функції STM32Cube HAL включають:

Апаратна абстракція

HAL надає абстракцію від конкретних деталей апаратного забезпечення, таких як реєстри регулювання, реєстри статусу, тощо. Це робить розробку більш зручною та зменшує залежність від конкретного апаратного виконання.

Конфігурація за допомогою STM32CubeMX

STM32Cube HAL інтегрується з STM32CubeMX, графічним інструментом конфігурації, що дозволяє швидко налаштовувати параметри мікроконтролера, включаючи підключені периферійні пристрої та генерацію основного коду HAL.

Підтримка різних моделей STM32

STM32Cube HAL підтримує різні моделі мікроконтролерів STM32, що дозволяє розробникам легко переносити свій код між різними пристроями. Це особливо корисно при масштабуванні проектів або при зміні моделі мікроконтролера.

Готові функції та драйвери

HAL містить готові функції для роботи з основними периферійними пристроями, такими як GPIO, UART, SPI, I2C, таймери, ADC і багато інших. Це дозволяє розробникам швидко використовувати ці функції без глибокого розуміння конкретних реєстрів та налаштувань.

Легко розширюваний код

Розробники можуть легко доповнювати генерований STM32Cube HAL код своїми функціями, щоб задовольнити конкретні потреби проекту.

Щоб використовувати STM32Cube HAL, розробники можуть скористатися STM32CubeMX для налаштування проекту та генерації початкового коду, а потім використовувати функції HAL для роботи з апаратним забезпеченням. Це полегшує розробку вбудованого програмного забезпечення та зменшує час, витрачений на низькорівневу настройку мікроконтролера.

CMSIS

CMSIS, або Cortex Microcontroller Software Interface Standard, є стандартом, який розроблений компанією ARM для створення єдиної інтерфейсної оболонки для вбудованих систем на базі їхніх мікропроцесорів Cortex-M. CMSIS надає стандартні інтерфейси та функції для роботи з різними мікроконтролерами, які використовують ядра Cortex-M.

Основні компоненти CMSIS включають:

CMSIS-CORE

Цей компонент містить стандартний інтерфейс до основних функцій ядра Cortex-M, таких як системні та службові реєстри, інструкції взаємодії з перериваннями, вектори переривань, інструкції забезпечення критичних секцій і інші.

CMSIS-DSP (Digital Signal Processing)

Цей компонент містить набір функцій для обробки цифрових сигналів. Він допомагає розробникам легко використовувати апаратні можливості обробки сигналів, що доступні на пристроях з ядрами Cortex-M.

CMSIS-RTOS API

Цей інтерфейс визначає стандарт для роботи з операційними системами реального часу (RTOS) на пристроях, які використовують ядра Cortex-M. Це спрощує портативність програмного забезпечення між різними RTOS або між пристроями, які використовують різні RTOS.

CMSIS-SVD (System View Description)

Цей компонент дозволяє описати структуру реєстрів та периферійних пристроїв на мікроконтролері. Це допомагає забезпечити легший доступ до апаратних ресурсів та полегшує роботу інструментів розробки.

CMSIS-Pack

Це стандарт для упаковки (packaging) периферійних пристроїв та ресурсів мікроконтролерів, які використовуються для розробки вбудованого програмного забезпечення. Це полегшує розповсюдження та використання сторонніх пакетів, таких як бібліотеки чи набори конфігурацій.

CMSIS сприяє стандартизації та спільності у розробці програмного забезпечення для мікроконтролерів на базі ядер Cortex-M, що полегшує переносимість коду між різними пристроями. Розробники можуть використовувати CMSIS для більш ефективного використання апаратного забезпечення та розширення функціональності своїх проектів.

RTOS

RTOS (Real-Time Operating System) - це операційна система реального часу, яка використовується для розробки вбудованих систем, де обробка подій відбувається з точністю до часу. RTOS надає інтерфейс для роботи з тасками, перериваннями, семафорами, чергами та іншими елементами для забезпечення контролю над виконанням програми в реальному часі.

STM32 RTOS (RTOS2) - це операційна система реального часу, яка була розроблена для використання на мікроконтролерах STM32 компанії STMicroelectronics. Операційна система входить до складу STM32Cube HAL та STM32CubeMX і може використовуватися для розробки вбудованих програм, де важлива точність та пріоритет виконання завдань.

Основні характеристики RTOS2 для STM32 включають:

Основні складові

RTOS2 включає ряд основних складових, таких як ядро операційної системи, менеджер завдань, планувальник, менеджер переривань, таймери та інші компоненти, які забезпечують взаємодію та синхронізацію задач в системі.

Підтримка багатьох завдань

RTOS2 дозволяє вам визначати та запускати багато задач, які працюють паралельно. Кожна задача має свій власний стек та пріоритет, що дозволяє керувати порядком виконання.

Система пріоритетів

Операційна система використовує систему пріоритетів для визначення порядку виконання завдань. Це дозволяє визначити, яка задача важливіша та отримає можливість виконання в першу чергу.

Механізми синхронізації та взаємодії

RTOS2 надає механізми для синхронізації та взаємодії між задачами, такі як семафори, черги, м'ютекси та інші. Це дозволяє задачам спілкуватися та координувати свою діяльність.

Реальний час та таймери

RTOS2 підтримує роботу в реальному часі, надаючи точні таймери та можливість визначати терміни виконання для задач.

Підтримка ARM Cortex-M

Операційна система розроблена з урахуванням архітектури ядер Cortex-M, що дозволяє ефективно використовувати можливості цих процесорів.

RTOS2 входить до складу набору STM32Cube та є важливою складовою для розробки складних та високопродуктивних вбудованих систем на мікроконтролерах STM32.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ПЕРЕДАЧІ ДАНИХ

4.1 Налаштування основних периферій мікроконтролера

Налаштування системного годинника є чи не найважливішим етапом в створенні програмного забезпечення для мікроконтролерів. Системний годинник впливає на безліч аспектів роботи мікроконтролера, ось декілька з них:

- Відстеження часу:

У деяких випадках мікроконтролери використовуються для вимірювання реального часу, особливо у вбудованих системах, де потрібно точно відстежувати час. Це може бути важливим для логування подій, планування завдань, реалізації таймерів і т. д.

- Системні таймери:

Системний годинник часто використовується як основа для таймерів та інших системних лічильників. Це може бути корисним для обчислення часових інтервалів, планування подій та виконання різних завдань у визначені моменти часу.

- Енергозбереження:

В низькопотужних додатках мікроконтролер може переходити в сплячий режим для економії енергії. В цьому режимі годинник може служити для визначення періодів активності та сну.

- Синхронізація:

У системах з кількома мікроконтролерами або пристроями може бути важливою синхронізація подій. Системний годинник дозволяє забезпечити узгодженість в часі між різними пристроями.

Ведення журналу дій та метадані:

Додавання дати та часу до логів та записів дозволяє стежити за хронологією подій і полегшує відладку та аналіз системи.

В даному проекті системний годинник налаштовано на частоту 168МГц відносно зовнішнього кварцевого генератора на 25 МГц. Налаштування частоти системного годинника для мікроконтролера STM32 являє собою налаштування коефіцієнтів перетворення від джерела частоти: внутрішнього або зовнішнього кварцевого генератора. В більшості мікроконтролерів STM32 кварцевий генератор є вбудованим в мікросхему(внутрішній), але для підвищення точності вимірювання в даній роботі використовується зовнішній кварцевий генератор.

```
 99  static void SystemClock_Config(void) // 168MHz
100  {
101    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
102    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
103    __HAL_RCC_PWR_CLK_ENABLE();
104    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
105    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
106    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
107    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
108    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
109    RCC_OscInitStruct.PLL.PLLM = 25;
110    RCC_OscInitStruct.PLL.PLLN = 336;
111    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
112    RCC_OscInitStruct.PLL.PLLQ = 4;
113    HAL_RCC_OscConfig(&RCC_OscInitStruct);
114    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
115                                |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
116    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
117    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
118    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
119    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
120    HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5);
121
122  }
```

Рис 4.1. Функція налаштування системного годинника

Також для швидкого налаштування внутрішнього годинника в додатку STM32CubeMX є динамічна діаграма візуалізації налаштувань годинників. За допомогою даної діаграми можна не лише підібрати коефіцієнти налаштування частоти а й згенерувати код за попередньо підібраними налаштуваннями. Для використання цієї функції потрібно лише вибрати вхідну частоту, та бажану частоту на виході у вікні SYSCLK, якщо введена частота можлива до застосування після процесу перерахунку дане вікно залишиться синім, якщо ж була введена неможлива частота то вікно стане червоним. Завершивши налаштування інших периферій і

згенерувавши код для вибраної IDE, в файлі main.h згенерується функція SystemClock_Config() з обраними налаштуваннями системного годинника.

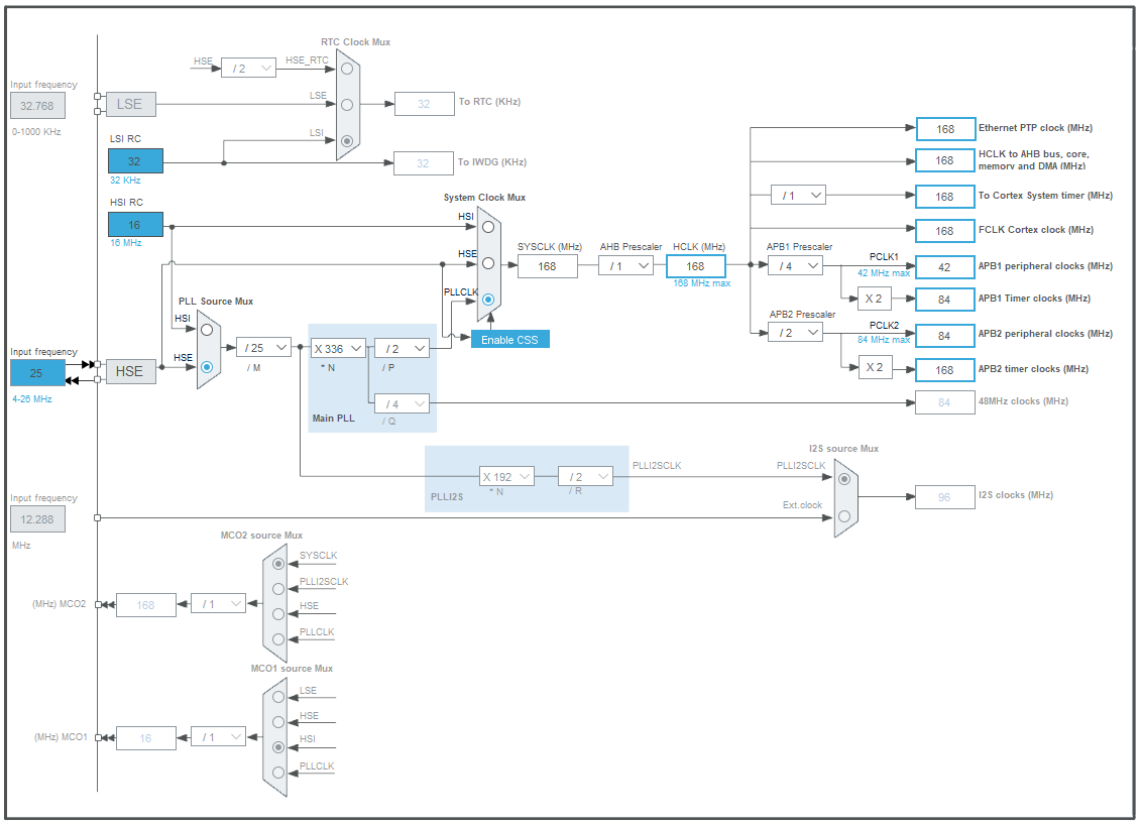


Рис 4.2. Налаштування системного годинника за допомогою програми STM32CubeMX

Реалізація операційної системи реального часу RTOS

Операційна система реального часу має велику кількість функцій та застосувань для полегшення програмування рішень які потребують точного вимірювання або роботи з часом. Для цієї роботи за допомогою RTOS створено два незалежні потоки app_main та device_main. Ініціалізація операційної системи зображена на рисунку 4. та виглядає наступним чином:

- Функція osKernelInitialize() в RTOS (Real-Time Operating System) використовується для ініціалізації ядра операційної системи реального часу. Основні завдання цієї функції включають: ініціалізація ядра, конфігурація системних ресурсів, резервування ресурсів, підготовка до запуску RTOS.

- Функція osThreadNew() в RTOS (Real-Time Operating System) використовується для створення нової задачі (потоку) в операційній системі реального часу. Вона має три аргументи, перший параметр вказує на функцію яка буде виконуватися в цьому потоці, другий параметр вказує на передачу аргументів в функцію яка буде виконуватись(NULL- відсутність даних аргументів), третій параметр вказує на налаштування параметрів потоку(NULL – параметри за замовчуванням).
- Функція osKernelStart() в RTOS (Real-Time Operating System) використовується для запуску планувальника (ядра) операційної системи реального часу. Після виклику цієї функції RTOS розпочинає свою роботу, і задачі (потоки) починають виконуватися відповідно до їх пріоритетів та планувальнику. Основні дії, які виконує функція osKernelStart(), включають: запуск планувальника, виконання основного циклу RTOS, виклик задач, обробка переривань, управління ресурсами. Функція osKernelStart() визначає початок виконання RTOS та забезпечує початок роботи всієї системи реального часу. Обов'язковим є виклик цієї функції після ініціалізації системи та створення задач.

Також варто зазначити, що при використанні RTOS не варто зловживати перериваннями та паузами в коді, а якщо це необхідно слід використовувати спеціальні функції для цього саме з бібліотеки RTOS.

```
47  osKernelInitialize();                 // Initialize CMSIS-RTOS
48  osThreadNew(app_main, NULL, NULL);    // Create application main thread
49  osThreadNew(device_main, NULL, NULL); // Create application main thread
50  osKernelStart();                      // Start thread execution
```

Рис 4.3. Ініціалізація операційної системи реального часу RTOS2

Потоки app_main та device_main є головними функціями програми які паралельно виконують два найважливіші завдання проекту, а саме app_main відповідає за обмін даними з комп’ютером по протоколу UDP, в той час як device_main відповідає за обмін одразу з трьома передавачами NRF905.

Функція потоку app_main зображена на рисунку 4. та виконує наступні функції:

- Функція netInitialize () ініціалізує використання обміну даних за протоколами TCP та UDP.
- Функція netDHCP_Disable (0) припиняє використання DHCP (Dynamic Host Configuration Protocol) на мережевому інтерфейсі з індексом 0. DHCP є протоколом, який дозволяє автоматично настроювати IP-адреси, мережеві параметри та інші налаштування для пристроїв в мережі. Зазвичай, коли вмикається DHCP на мережевому інтерфейсі, пристрій автоматично отримує IP-адресу, шлюз, DNS-сервери та інші необхідні налаштування від DHCP-сервера в мережі.
- Функція SetNetParam (netParam) встановлює значення IP-адресу та MAC-адресу на основі змінної netParam, значення якої надається функцією Read_SW_Net () яка зчитує перемикачі на платі та відповідно від встановленої конфігурації надає значення цієї конфігурації.
- tcp_sock = netTCP_GetSocket (tcp_cb_server); даний рядок коду створює TCP сокет.
- if (tcp_sock > 0) {netTCP_Listen (tcp_sock, PortTCP);} даний рядок коду перевіряє чи дійсно створений TCP сокет (tcp_sock більше 0 якщо сокет успішно створений), якщо сокет успішно створений то викликається функція netTCP_Listen(), яка розпочинає прослуховування (listening) TCP-сокета на певному порті PortTCP. Це означає, що сокет готовий приймати нові вхідні підключення.
- udp_sock = netUDP_GetSocket (udp_cb_func); даний рядок коду створює UDP сокет.
- if (udp_sock > 0) { netUDP_Open (udp_sock, PortUDP);} у цьому рядку коду перевіряється, чи значення udp_sock більше 0 (якщо сокет успішно створений). Якщо це умова виконується, то викликається функція netUDP_Open(), яка відкриває

UDP-сокет і асоціює його з певним портом PortUDP. Після виклику цієї функції, сокет готовий для прийому та відправлення даних за допомогою UDP-протоколу.

- while(1) { osThreadFlagsWait (0, osFlagsWaitAny, osWaitForever); } Основна структура циклу while(1) означає, що код усередині циклу буде виконуватись постійно, бо умова 1 є завжди істинною. У циклі викликається функція osThreadFlagsWait(0, osFlagsWaitAny, osWaitForever). Ця функція очікує на виникнення прапорців (flags) і блокує виконання поточного потоку, доки не виникне хоча б один прапорець.

Аргументи, передані у функцію osThreadFlagsWait(), вказують наступне:

0 - це бітова маска прапорців, за якими слід очікувати. У даному випадку, 0 означає, що будь-який прапорець може виникнути. osFlagsWaitAny - це режим очікування, де потік буде розблоковано при виникненні будь-якого прапорця. osWaitForever - це параметр, що вказує, що потік буде блокований до виникнення прапорця без обмежень у часі.

```
57 /*----------------------------------------------------------------------------
58   Main Thread 'main': Run Network
59  *----------------------------------------------------------------------------*/
60 __NO_RETURN void app_main (void *arg) {
61   (void)arg;
62
63   netInitialize ();
64   netDHCP_Disable (0);
65   netParam = Read_SW_Net ();
66   SetNetParam (netParam);
67
68   tcp_sock = netTCP_GetSocket (tcp_cb_server);
69   if (tcp_sock > 0) {netTCP_Listen (tcp_sock, PortTCP);}
70   udp_sock = netUDP_GetSocket (udp_cb_func);
71   if (udp_sock > 0) { netUDP_Open (udp_sock, PortUDP);}
72
73
74   while(1) { osThreadFlagsWait (0, osFlagsWaitAny, osWaitForever); }
75 }
```

Рис 4.4. Основна частина потоку app_main

Функція потоку device_main зображена на рисунку 4. та виконує наступні функції:

- Функція HAL_TIM_Base_Start(&htim4) запускає таймер TIM4 який забезпечує відлік часу між роботою трьох каналів радіоперадачі.

- if(netParam != Read_SW_Net()) NVIC_SystemReset() даний рядок коду забезпечує перезапуск мікроконтролера якщо було змінено конфігурацію параметрів мережі на перемикачі плати.
- if(ChannelParam != Read_SW_Channel()) NVIC_SystemReset() даний рядок коду забезпечує перезапуск мікроконтролера якщо було змінено конфігурацію каналів радіопередавачів на перемикачі плати.
- Функція Exchange() здійснює обмін даними радіопередавачів NRF905

```
78 /*-----------------------------------------------------------------------------
79   Main Thread 'main': Run Device
80  *----------------------------------------------------------------------------*/
81 __NO_RETURN void device_main (void *arg) {
82   (void)arg;
83   HAL_TIM_Base_Start(&htim4);
84 
85   while(1)
86   {
87     if(netParam != Read_SW_Net()) NVIC_SystemReset();
88     if(ChannelParam != Read_SW_Channel())  NVIC_SystemReset();
89 
90     Exchange();
91   }
92 }
```

Рис 4.5. Основна частина потоку device_main

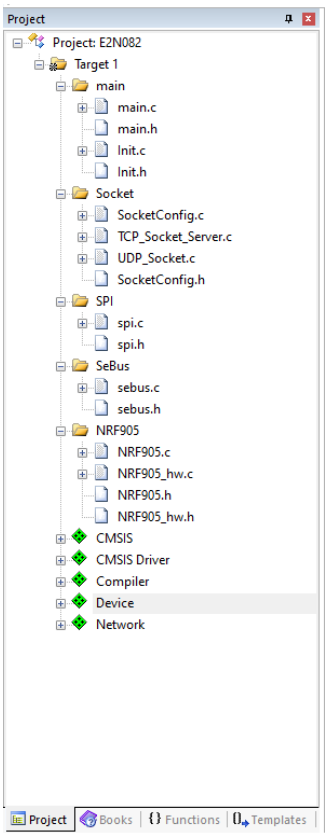


Рис 4.6. Загальна структура проекту

4.2 Налаштування обміну за допомогою протоколу UDP

Налаштування мережевого обміну за допомогою протоколів TCP/UDP для мікроконтролера вимагає комплексного підходу до вибору та налаштування бібліотек та функцій передачі даних. Перш за все потрібно налаштувати низку параметрів для правильного функціонування мережевої бібліотеки. Для мережевої бібліотеки Keil uVision це наступні файли заголовків(Рис 4.): Net_Config_ETH_0.h, Net_Config_TCP.h, Net_Config_UDP.h. Також на рисунку 4. зображено приклад налаштування файлу заголовків Net_Config_ETH_0.h, серед основних налаштувань слід виділити: IP Address, Subnet mask, Primary Gateway, Primary DNS Server, Secondary DNS Server, MAC Address та ін.

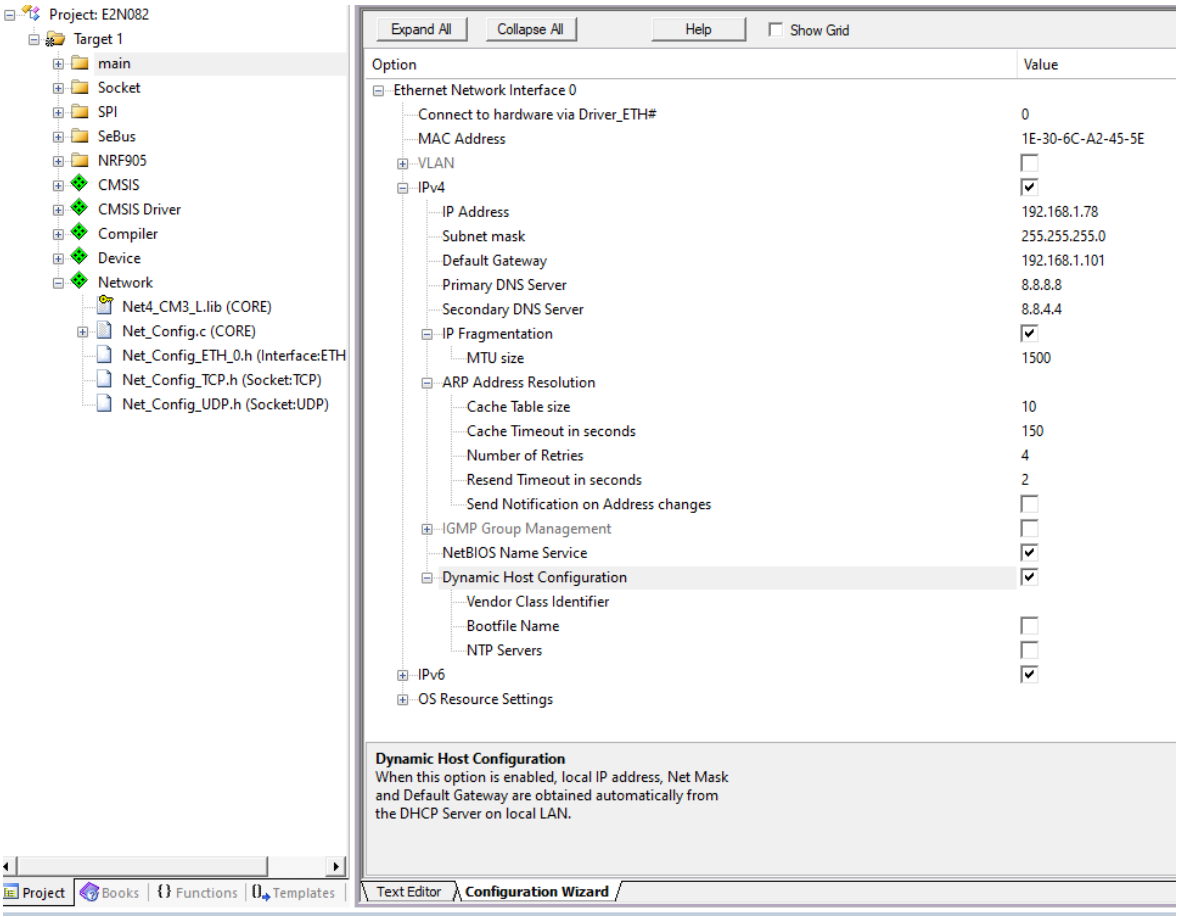


Рис 4.7. Приклад налаштування мережі, файл Net_Config_ETH_0.h

Для обміну за протоколом UDP в даному проекті використовуються дві функції send_udp_data() та udp_cb_func(). Ці функції забезпечують двосторонній

обмін даними між комп’ютером та мікроконтролером, в даному випадку мікроконтролер виступає в ролі сервера а комп’ютер в ролі клієнта. Функція send_udp_data() відповідає за надсилання даних клієнту, вона виконується після отримання даних в udp_cb_func(). Функція udp_cb_func() виконується коли до мікроконтролера надходять дані на визначений UDP порт. Дана функція обробляє вхідні дані та надсилає дані у відповідь.

Функція send_udp_data() зображена на рисунку 4. та виконує наступні завдання:

- if (udp_sock > 0) {...}: Перевіряється, чи змінна udp_sock більше 0, що, означає, що сокет для UDP створено і готовий до використання.
- NET_ADDR addr = { NET_ADDR_IP4, RemotePortUDP, RemoteIpUDP[0], RemoteIpUDP[1], RemoteIpUDP[2], RemoteIpUDP[3] };: Створюється структура NET_ADDR для представлення IP-адреси та порту призначення. Значення IP-адреси та порту визначаються як RemoteIpUDP і RemotePortUDP відповідно.
- uint8_t *sendbuf; sendbuf = netUDP_GetBuffer (len);: Визначається вказівник sendbuf типу uint8_t, і йому призначається буфер, отриманий в результаті виклику функції netUDP_GetBuffer(len). Ця функція відповідає для перевизначення пам’яті буфера для відправки UDP-даних.
- memcpy (sendbuf, tx_udp, len);: Копіює дані з буфера tx_udp розміром len в буфер sendbuf. Це копіювання даних, які потрібно відправити.
- netUDP_Send (udp_sock, &addr, sendbuf, len);: Викликає функцію netUDP_Send для відправлення UDP-даних. Параметри цієї функції включають сокет udp_sock, структуру addr для визначення призначення, буфер sendbuf з даними та розмір даних len. Дана функція надсилає дані за протоколом UDP.

```
16  // Send UDP data to destination client.
17  void send_udp_data (void) {
18
19    unsigned short len = tmpCnt;
20    if (udp_sock > 0)
21    {
22      NET_ADDR addr = { NET_ADDR_IP4, RemotePortUDP, RemoteIpUDP[0], RemoteIpUDP[1], RemoteIpUDP[2], RemoteIpUDP[3] };
23      uint8_t *sendbuf;
24      sendbuf = netUDP_GetBuffer (len);
25      memcpy (sendbuf, tx_udp, len);
26      netUDP_Send (udp_sock, &addr, sendbuf, len);
27    }
28  }
```

Рис 4.8. Функція send_udp_data()

Функція udp_cb_func() зображена на рисунку 4. та виконує наступні завдання:

- У рядках 35-39 відбувається отримання IP-адреси та порту призначення зі структури NET_ADDR, значення IP-адреси та порту призначення зберігаються у змінних RemoteIpUDP і RemotePortUDP.
- if(len > SizeBuff) len = SizeBuff;: Умова обмежує розмір отриманих даних аби уникнути критичних помилок під час перезапису даних.
- for(i = 0; i < len; i++) { rx_udp[i] = *buf; buf++;}: Копіювання отриманих даних до буфера rx_udp за допомогою циклу, кожен байт отриманих даних копіюється в буфер rx_udp.
- tmpCnt = SEbusCRC_BufProc(rx_udp, len, CommTab_SRV, 461);: Обчислення контрольної суми за допомогою функції SEbusCRC_BufProc, значення контрольної суми зберігається у змінній tmpCnt.
- send_udp_data();: Виклик функції send_udp_data для відправки даних через UDP.

```
31  // Notify the user application about UDP socket events.
32  uint32_t udp_cb_func (int socket, const  NET_ADDR *addr, const unsigned char *buf, unsigned int len) {
33
34    unsigned char i;
35    RemoteIpUDP[0] = (unsigned char)addr->addr[0];
36    RemoteIpUDP[1] = (unsigned char)addr->addr[1];
37    RemoteIpUDP[2] = (unsigned char)addr->addr[2];
38    RemoteIpUDP[3] = (unsigned char)addr->addr[3];
39    RemotePortUDP = addr->port ;
40    if(len > SizeBuff) len = SizeBuff;
41    for(i = 0; i < len; i++) {rx_udp[i] = *buf; buf++;}
42    tmpCnt = SEbusCRC_BufProc(rx_udp, len, CommTab_SRV, 461);
43    send_udp_data ();
44    return (0);
45  }
```

Рис 4.9. Функція udp_cb_func()

4.3 Інтегрування бібліотеки для радіопередавачів NRF905

За основу для роботи з радіопередавачем використовується бібліотека з ресурсу Github від автора Wojciech Domski. В даній бібліотеці реалізовано всі функції які підтримує радіопередавач NRF905, також великою перевагою даної бібліотеки є її реалізація на рівні апаратної абстракції. Завдяки цьому дану бібліотеку легко використовувати на мікроконтролерах від різних виробників.

В даній бібліотеці реалізовано низку функцій для роботи з радіопередавачем NRF905, ці функції зображено на рис 4. Ось декілька основних функцій:

- Функція NRF905_init() ініціалізує радіомодуль та записує в нього обрані налаштування.
- Функція NRF905_read_config_register() зчитує регістри конфігурації радіопередавача NRF905.
- Функція NRF905_write_config_register() записує регістри конфігурації радіопередавача NRF905.
- Функція NRF905_tx() відповідає за передачу даних.
- Функція NRF905_rx() переводить радіопередавач в режим прийому даних.
- Функція NRF905_read() зчитує дані які отримав радіопередавач.
- Функція NRF905_data_ready() перевіряє чи надійшли дані радіопередавачеві.
- Функція NRF905_address_matched() перевіряє адрес вхідного пакету.

Функції NRF905_data_ready() та NRF905_address_matched() програмно виконують ті ж функції що і фізичні піни радіомодуля AM та DR. Завдяки цим функціям у користувача є змога перевірити чи надійшли коректні дані на радіомодуль, і чи правильний адрес у відправника цих даних. Також в даній бібліотеці є низка другорядних функцій які допомогають налаштувати даний радіомодуль, серед них:

- Функція NRF905_setAddress() встановлює адрес радіомодуля.
- Функція NRF905_set_channel() встановлює канал радіомодуля.
- Функція NRF905_power_up() вмикає радіопередавач.
- Функція NRF905_power_down() вимикає передавач.
- Функція NRF905_standby() переводить радіомодуль в режим очікування.

```
232 uint8_t NRF905_read_config_register(NRF905_t *dev, uint8_t reg);
233 int NRF905_write_config_register(NRF905_t *dev, uint8_t reg, uint8_t val);
234 int NRF905_set_config_reg1(NRF905_t *dev, uint8_t val, uint8_t mask,
235     uint8_t reg);
236 int NRF905_set_config_reg2(NRF905_t *dev, uint8_t val, uint8_t mask,
237     uint8_t reg);
238 int NRF905_setAddress(NRF905_t *dev, uint32_t address, uint8_t cmd);
239 uint8_t NRF905_read_status(NRF905_t *dev);
240 int NRF905_data_ready(NRF905_t *dev);
241 int NRF905_address_matched(NRF905_t *dev);
242 int NRF905_set_channel(NRF905_t *dev, uint16_t channel);
243 int NRF905_set_band(NRF905_t *dev, NRF905_band_t band);
244 int NRF905_set_auto_retransmit(NRF905_t *dev, NRF905_auto_retran_t val);
245 int NRF905_set_low_rx_power(NRF905_t *dev, NRF905_low_rx_t val);
246 int NRF905_set_tx_power(NRF905_t *dev, NRF905_pwr_t val);
247 int NRF905_set_CRC(NRF905_t *dev, NRF905_crc_t val);
248 int NRF905_set_clk_out(NRF905_t *dev, NRF905_outclk_t val);
249 int NRF905_set_payload_size(NRF905_t *dev, uint8_t size);
250 int NRF905_set_address_size(NRF905_t *dev, NRF905_addr_size_t size);
251 uint8_t NRF905_receive_busy(NRF905_t *dev);
252 uint8_t NRF905_airway_busy(NRF905_t *dev);
253 int NRF905_set_listen_address(NRF905_t *dev, uint32_t address);
254 uint8_t NRF905_tx(NRF905_t *dev, uint32_t sendTo, void *data, uint8_t len,
255     NRF905_nextmode_t nextMode);
256 int NRF905_rx(NRF905_t *dev);
257 int NRF905_read(NRF905_t *dev, void *data, uint8_t len);
258 int NRF905_power_down(NRF905_t *dev);
259 int NRF905_power_up(NRF905_t *dev);
260 int NRF905_standby(NRF905_t *dev);
261 int NRF905_get_config(NRF905_t *dev, void *regs);
262 int NRF905_init(NRF905_t *dev, NRF905_hw_t *hw);
263 int NRF905_set_config(NRF905_t *dev);
```

Рис 4.10. Функції для роботи з радіопередавачем NRF905

Також в даній бібліотеці в файлі заголовку NRF905.h реалізовано гнучкі налаштування радіомодуля за допомогою визначень define зображених на рисунках 4. і 4. :

- Визначення NRF905_CHANNEL встановлює радіоканал для NRF905. Радіомодуль NRF905 підтримує налаштування 512 каналів, мінімальна частота для

смуги 433 МГц складає 422.4 МГц(0 канал), а максимальна 473.5 МГц(511 канал), різниця в частоті між сусідніми каналами складає 100 КГц. Для смуги 868/915 МГц мінімальна частота складає 844.8 МГц(0 канал), максимальна частота 947.0 МГц(511 канал), різниця в частоті між сусідніми каналами складає 200 КГц.

- Визначення NRF905_BAND визначає смугу частоти на якій працюватиме радіопередавач NRF905. Даний радіопередавач підтримує три частотні смуги 433 МГц, 868 МГц та 915 МГц, але смуги 868 МГц та 915 МГц об'єднані в одну.
- Визначення NRF905_PWR встановлює потужність радіопередавача. Даний радіопередавач є досить енергоефективний, але для збільшення енергоефективності існує налаштування потужності. Наприклад при налаштуванні підсилення -10 дБм споживання енергії складає всього 100 мікроват що неймовірно мало та підійде для рішень де потрібна надзвичайна енергоефективність, та невеликий радіус роботи. При максимальному підсиленні сигналу в 10 дБм споживання енергії складатиме близько 10 міліват, що дещо більше ніж при підсиленні в -10 дБм, але даний показник також дуже енергоефективний.
- Визначення NRF905_LOW_RX надає можливість для встановлення зменшеної потужності радіопередавача в режимі прийому, дана функція може зменшити споживання електроенергії на декілька мА та є дуже корисною при використанні рішень де потрібен радіоприймач який має постійно прослуховувати радіо ефір.
- Визначення NRF905_AUTO_RETRAN вмикає функцію повторної передачі пакету в ефір для радіопередавача. Дана функція є корисною якщо потрібно передати пакет в ефірі де велика кількість радіоперешкод.
- Визначення NRF905_OUTCLK визначає частоту яку радіомодуль може передати мікроконтролеру. Дана функція може бути корисною для в енергоефективних рішеннях та для економії місця на друкованій платі. Мікросхема модуля NRF905, може передати частоту для головного мікроконтролера з кварцевого генератора розташованого на радіомодулі.

- Визначення NRF905_CRC встановлює алгоритм обчислення контрольної суми для перевірки цілісності даних. Для даного радіомодуля є можливість встановити алгоритм обчислення контрольної суми CRC-8 та CRC-16, або вимкнути обчислення контрольної суми.

```
90  // Frequency
91  // Channel 0 is 422.4MHz for the 433MHz band, each channel
92  // Channel 0 is 844.8MHz for the 868/915MHz band, each chan
93  // Max channel is 511 (473.5MHz / 947.0MHz)
94  #define NRF905_CHANNEL      100
95
96  // Frequency band
97  // 868 and 915 are actually the same thing
98  // NRF905_BAND_433
99  // NRF905_BAND_868
100 // NRF905_BAND_915
101 #define NRF905_BAND     NRF905_BAND_433
102
103 // Output power
104 // n means negative, n10 = -10
105 // NRF905_PWR_n10 (-10dBm = 100uW)
106 // NRF905_PWR_n2 (-2dBm = 631uW)
107 // NRF905_PWR_6 (6dBm = 4mW)
108 // NRF905_PWR_10 (10dBm = 10mW)
109 #define NRF905_PWR      NRF905_PWR_10
110
111 // Save a few mA by reducing receive sensitivity
112 // NRF905_LOW_RX_DISABLE (Normal sensitivity)
113 // NRF905_LOW_RX_ENABLE (Lower sensitivity)
114 #define NRF905_LOW_RX   NRF905_LOW_RX_DISABLE
115
116 // Constantly retransmit payload while in transmit mode
117 // Can be useful in areas with lots of interference, but yo
118 // It will also block other transmissions if collision avoi
119 // NRF905_AUTO_RETRAN_DISABLE
120 // NRF905_AUTO_RETRAN_ENABLE
121 #define NRF905_AUTO_RETRAN  NRF905_AUTO_RETRAN_DISABLE
122
123 // Output a clock signal on pin 3 of IC
124 // NRF905_OUTCLK_DISABLE
125 // NRF905_OUTCLK_500KHZ
126 // NRF905_OUTCLK_1MHZ
127 // NRF905_OUTCLK_2MHZ
128 // NRF905_OUTCLK_4MHZ
129 #define NRF905_OUTCLK   NRF905_OUTCLK_DISABLE
130
131 // CRC checksum
132 // NRF905_CRC_DISABLE
133 // NRF905_CRC_8
134 // NRF905_CRC_16
135 #define NRF905_CRC      NRF905_CRC_16
```

Рис 4.11. Визначення радіомодуля NRF905

- Визначення NRF905_CLK_FREQ встановлює частоту кварцевого генератора встановленого на радіомодулі.
- Визначення NRF905_ADDR_SIZE встановлює розмір адресу радіомодуля. На вибір є однобайтна адреса та чотирьохбайтна адреса. Для більшої завадозахищеності слід використовувати чотирьохбайтну адресу.

- Визначення NRF905_PAYLOAD_SIZE визначає розмір корисного навантаження від одного до тридцяти двох байт даних.

Для максимальної ефективності радіомодуля слід якомого точніше налаштувати дані визначення. В залежності від потреб слід обирати ті чи інші параметри даного радіомодуля та тестувати якість передачі даних аби досягти якомога кращі технічні характеристики.

Також дана бібліотека має наступні файли NRF905_hw.c та NRF905_hw.h, дані файли відповідають за апаратну абстракцію. Завдяки функціям реалізованим в цих файлах дана бібліотека дає змогу поєднати її з будь-якими іншими бібліотеками обслуговуючими периферію мікроконтролера. На рисунку 4. зображено визначення та функції файлу NRF905_hw.h, дані функції мають виконувати наступні завдання:

- Функція NRF905_hw_gpio_get() отримує значення введення/виведення (GPIO) для конкретного піна модуля NRF905.
- Функція NRF905_hw_gpio_set() встановлює значення виведення (GPIO) для конкретного піна модуля NRF905.
- Функція NRF905_hw_delay_ms() встановлює затримку (в мілісекундах) за допомогою апаратних засобів мікроконтролера, пов'язану із модулем NRF905.
- Функція NRF905_hw_delay_us() встановлює затримку (в мікросекундах) за допомогою апаратних засобів мікроконтролера, пов'язану із модулем NRF905.
- Функція NRF905_hw_enable_timer() вмикає апаратний таймер, пов'язаний із модулем NRF905. Даний таймер потрібен для синхронізації в роботі з модулем.
- Функція NRF905_hw_spi_transfer() здійснює передачу одного байта через SPI інтерфейс в модуль NRF905 і отримання відповіді.

Визначення файлу NRF905_hw.h виконують наступні завдання :

- Визначення NRF905_HW_POWERED_UP() перевіряє, чи у модулі NRF905 ввімкнено живлення (піднято на лінії живлення).

- Визначення NRF905_HW_POWER_DOWN() вимикає живлення для модуля NRF905.
- Визначення NRF905_HW_POWER_UP() вмикає живлення для модуля NRF905.
- Визначення NRF905_HW_STANDBY_ENTER() переводить модуль NRF905 в режим очікування (standby).
- Визначення NRF905_HW_STANDBY_LEAVE() виводить модуль NRF905 із режиму очікування (standby).
- Визначення NRF905_HW_MODE_RX() встановлює модуль NRF905 у режим прийому (RX).
- Визначення NRF905_HW_MODE_TX() встановлює модуль NRF905 у режим передачі (TX).
- Визначення NRF905_HW_SPI_SELECT() встановлює модуль NRF905 для обміну даними за допомогою SPI-інтерфейсу.
- Визначення NRF905_HW_SPI_DESELECT() () знімає встановлення модуля NRF905 для обміну даними за допомогою SPI-інтерфейсу.

```
34  int NRF905_hw_gpio_get(NRF905_hw_t *hw, uint8_t gpio);
35
36  int NRF905_hw_gpio_set(NRF905_hw_t *hw, uint8_t gpio, uint8_t value);
37
38  void NRF905_hw_delay_ms(NRF905_hw_t *hw, uint32_t ms);
39
40  void NRF905_hw_delay_us(NRF905_hw_t *hw, uint16_t delay);
41
42  void NRF905_hw_enable_timer(NRF905_hw_t *hw);
43
44  int NRF905_hw_spi_transfer(NRF905_hw_t *hw, uint8_t data_tx, uint8_t *data_rx);
45
46  #define NRF905_HW_POWERED_UP(hw)    NRF905_hw_gpio_get(hw, NRF905_HW_GPIO_PWR)
47  #define NRF905_HW_POWER_DOWN(hw)    NRF905_hw_gpio_set(hw, NRF905_HW_GPIO_PWR, 0)
48  #define NRF905_HW_POWER_UP(hw)      NRF905_hw_gpio_set(hw, NRF905_HW_GPIO_PWR, 1)
49
50  #define NRF905_HW_STANDBY_ENTER(hw)   NRF905_hw_gpio_set(hw, NRF905_HW_GPIO_TRX_EN, 0)
51  #define NRF905_HW_STANDBY_LEAVE(hw)   NRF905_hw_gpio_set(hw, NRF905_HW_GPIO_TRX_EN, 1)
52
53  #define NRF905_HW_MODE_RX(hw)     NRF905_hw_gpio_set(hw, NRF905_HW_GPIO_TXEN, 0)
54  #define NRF905_HW_MODE_TX(hw)     NRF905_hw_gpio_set(hw, NRF905_HW_GPIO_TXEN, 1)
55
56  #define NRF905_HW_SPI_SELECT(hw)    NRF905_hw_gpio_set(hw, NRF905_HW_GPIO_CS, 0)
57  #define NRF905_HW_SPI_DESELECT(hw)    NRF905_hw_gpio_set(hw, NRF905_HW_GPIO_CS, 1)
```

Рис 4.12. Функції та визначення файлу NRF905_hw.h

Для використання даної бібліотеки слід адаптувати вищезазначені функції та визначення для роботи з обраним середовищем розробки та бібліотеками що використовуються в проекті. Дані функції є основою для реалізації всіх функцій бібліотеки NRF905 і адаптують використання даної бібліотеки під різні умови роботи та для різних мікроконтролерів. Завдяки правильній обробці цих базових функцій можна уникнути майже всіх помилок в роботі мікроконтролера.

4.4 Розробка та тестування швидкої та безперебійної радіопередачі даних

Розробка швидкої та безперебійної радіопередачі даних базується на можливості радіопередавача працювати одночасно в режимі прийому та передачі радіоданих. Завдяки наявності в передавача апаратної функції Chip Select є можливість задіяти декілька радіомодулів на одному інтерфейсі SPI для цього потрібно лише передавати сигнал на радіомодуль про початок чи закінчення обміну даними через SPI. Дана функція значно спрощує роботу з декількома радіопередавачами та заміняє собою використання декількох інтерфейсів SPI чи мікросхем мультиплексорів та демультиплексорів. Перед ініціалізацією радіопередавача, слід ініціалізувати інтерфейс обміну SPI. Для даного проекту буде використано інтерфейс SPI3 мікроконтролера. Основні налаштування даного інтерфейсу зображено на рисунку 4. :

- Рядок hspi3.Init.Mode = SPI_MODE_MASTER встановлює SPI3 в режим майстра.
- Рядок hspi3.Init.Direction = SPI_DIRECTION_2LINES встановлює двосторонній режим передачі.
- Рядок hspi3.Init.DataSize = SPI_DATASIZE_8BIT встановлює розмір передаваних даних 8 біт.
- Рядок hspi3.Init.NSS = SPI_NSS_SOFT вмикає функцію програмного Chip Select. NSS (Slave Select) - це сигнал у протоколі SPI, який використовується для вибору конкретного пристрою-споживача (slave device) на шині SPI.

- Рядок hspi3.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32 встановлює частоту передачі (Baud Rate). Зменшення значення Baud Rate Prescaler зазвичай призводить до збільшення швидкості обміну даними, оскільки частота передачі зменшується.
- Рядок hspi3.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE вимикає обчислення циклічного розрахунку контрольної суми (CRC) в рамках налаштувань SPI.

```
 99  void SPI_Init (void)
100  {
101    __HAL_RCC_SPI3_CLK_ENABLE();
102
103   hspi3.Instance = SPI3;
104    hspi3.Init.Mode = SPI_MODE_MASTER;
105    hspi3.Init.Direction = SPI_DIRECTION_2LINES;
106    hspi3.Init.DataSize = SPI_DATASIZE_8BIT;
107    hspi3.Init.CLKPolarity = SPI_POLARITY_LOW;
108    hspi3.Init.CLKPhase = SPI_PHASE_1EDGE;
109    hspi3.Init.NSS = SPI_NSS_SOFT;
110    hspi3.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32;
111    hspi3.Init.FirstBit = SPI_FIRSTBIT_MSB;
112    hspi3.Init.TIMode = SPI_TIMODE_DISABLE;
113    hspi3.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
114    hspi3.Init.CRCPolynomial = 7;
115    HAL_SPI_Init(&hspi3);
116
117  ////////////////////////////////////////////////
118    __HAL_RCC_GPIOC_CLK_ENABLE();
119    GPIO_InitStruct.Pin = GPIO_PIN_10|GPIO_PIN_11|GPIO_PIN_12;//SPI3
120    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
121    GPIO_InitStruct.Pull = GPIO_NOPULL;
122    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
123    GPIO_InitStruct.Alternate = GPIO_AF6_SPI3;
124    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
125  }
126
```

Рис.4.13. Ініціалізація інтерфейсу SPI

Також для роботи інтерфейсу SPI потрібно налаштувати піни GPIO. На рисунку 4. Рядки коду 118-124 виконують налаштування пінів MISO, MOSI, SCK:

- SCK (Serial Clock): Це тактовий сигнал, який синхронізує передачу даних між майстером (master) і слейвом (slave).
- MOSI (Master Out Slave In): Це вихідний сигнал від майстра до слейва, через який передаються дані від майстра до пристрою.

- MISO (Master In Slave Out): Це вихідний сигнал від слейва до майстра, через який передаються дані від пристрою до майстра.
- SS/CS (Slave Select/Chip Select): Це сигнал вибору слейва, який майстер активує для вибору конкретного пристрою на лінії SPI.

Зазвичай, майстер має додатковий пін для кожного підключеного слейва для вибору конкретного пристрою (SS/CS). Налаштування даного піна відбувається під час ініціалізації радіомодуля NRF905. Крім того, деякі пристрої можуть використовувати додаткові піни для функцій, таких як ініціалізація, переривання та інші.

Ініціалізація радіомодулів NRF905 мікроконтролером STM32F407 для подальшої передачі даних реалізовується за допомогою функцій бібліотеки NRF905 та складається з трьох функцій, кожна функція ініціює власний радіопередавач. Початок ініціалізації радіопередавача розпочинається з передачі в структуру бібліотеки NRF905 попередньо визначених та ініціалізованих пінів управління радіомодулем. Завдяки повній реалізації бібліотеки для даного модуля потрібно лише визначити дані піни, далі бібліотека буде оперувати ними самостійно в залежності від потреб користувача. В даного радіомодуля присутні одинадцять функціональних пінів, які виконують наступні функції:

- CE - ввімкнення/вимкнення радіомодуля.
- TXE - вибір режиму передачі або прийому.
- PWR - ввімкнення/вимкнення живлення передавача.
- CD - виявлення несучої частоти.
- AM - виявлення адреси.
- DR - готовність до передачі даних.
- CLK - тактовий сигнал кварцевого генератора.
- MISO - вихідний сигнал SPI.
- MOSI - вхідний сигнал SPI.
- SCK - сигнал синхронізації SPI.
- CSN - вибір пристрою SPI.

Для даного проекту не використовуються наступні піни: CLK, AM та DR. Пін CLK не потрібен так як тактовий сигнал для мікроконтролера надходить від іншого кварцевого генератора. Піни AM та DR досить важливі при налаштуванні швидкісного звязку, вони сигналізують про виявлення адреси та готовність до передачі даних відповідно, але бібліотека NRF905 дає змогу замінити ці піни програмно шляхом зчитування регістрів в яких міститься інформація аналогічна до роботи даних пінів. Це дає змогу зменшити кількість провідників та не втратити повного функціоналу даного радіопередавача.

Також для ініціалізації даного радіомодуля слід вказати структурі бібліотеки інтерфейс SPI та таймер TIM які будуть використовуватись для передачі даних.

Для ініціалізації радіопередавача слід застосувати декілька функцій з бібліотеки:

- Функція NRF905_init() вмикає та ініціалізує радіомодуль NRF905 відносно структури NRF905_hw1 з даними пінів та інтерфейсів визначених вище.
- Функція NRF905_set_config() встановлює налаштування радіомодуля на основі визначень з налаштуваннями файлу бібліотеки NRF905.h
- Функція NRF905_set_listen_address() встановлює адрес з якого потрібно приймати повідомлення.
- Функція NRF905_set_channel() встановлює радіоканал на якому працюватиме радіопередавач.

```
127  void Init_NRF1 (void) //E2N048
128  {
129
130      ////////////NRF 1/////////////
131    NRF905_hw1.gpio[NRF905_HW_GPIO_TXEN].pin = TXEN_Pin1;
132    NRF905_hw1.gpio[NRF905_HW_GPIO_TXEN].port = TXEN_GPIO_Port1;
133    NRF905_hw1.gpio[NRF905_HW_GPIO_TRX_EN].pin = TRX_CE_Pin1;
134    NRF905_hw1.gpio[NRF905_HW_GPIO_TRX_EN].port = TRX_CE_GPIO_Port1;
135    NRF905_hw1.gpio[NRF905_HW_GPIO_PWR].pin = PWR_Pin1;
136    NRF905_hw1.gpio[NRF905_HW_GPIO_PWR].port = PWR_GPIO_Port1;
137
138    NRF905_hw1.gpio[NRF905_HW_GPIO_CD].pin = CD_Pin1;
139    NRF905_hw1.gpio[NRF905_HW_GPIO_CD].port = CD_GPIO_Port1;
140    NRF905_hw1.gpio[NRF905_HW_GPIO_AM].pin = 0;
141    NRF905_hw1.gpio[NRF905_HW_GPIO_AM].port = NULL;
142    NRF905_hw1.gpio[NRF905_HW_GPIO_DR].pin = 0;
143    NRF905_hw1.gpio[NRF905_HW_GPIO_DR].port = NULL;
144
145    NRF905_hw1.gpio[NRF905_HW_GPIO_CS].pin = SPI_CS_Pin1;
146    NRF905_hw1.gpio[NRF905_HW_GPIO_CS].port = SPI_CS_GPIO_Port1;
147
148    NRF905_hw1.tim = &htim1;
149    NRF905_hw1.spi = &hspi3;
150
151
152    NRF905_init(&NRF905_1, &NRF905_hw1);
153    NRF905_set_config(&NRF905_1);
154    NRF905_set_listen_address(&NRF905_1, ADDRESS_MASTER1);
155    NRF905_set_channel(&NRF905_1, NRF905_channel_1);
156
157  }
```

Рис 4.14. Функція ініціалізації радіопередавача NRF905

Завершальним кроком реалізації радіообміну є власне функція обміну Exchange(). Дана функція за допомогою інтерфейсу SPI почергово записує та зчитує дані з трьох радіопередавачів NRF905. Дана функція є результатом низки експериментів а її робота оптимізована для передачі даних на значних швидкостях з мінімальними затримками.

```
void Exchange (void)
{
  uint8_t state_DR = 0;
  uint8_t state_AM = 0;
  switch (timer)
      {
        case 1:
          NRF905_tx(&NRF905_1, ADDRESS_SLAVE1, &DatNRF1_TX, BufSizeTX, NRF905_NEXTMODE_RX);
          NRF905_rx(&NRF905_1);
          //osDelay(2);
          state_DR = NRF905_data_ready(&NRF905_1);
          state_AM = NRF905_address_matched(&NRF905_1);
          if (state_DR && state_AM) {
            NRF905_read(&NRF905_1, &DatNRF1_RX, BufSizeRX);
            HL1_tg;
          }
          break;
        case 2:
          NRF905_tx(&NRF905_2, ADDRESS_SLAVE2, &DatNRF2_TX, BufSizeTX, NRF905_NEXTMODE_RX);
          NRF905_rx(&NRF905_2);
          //osDelay(2);
          state_DR = NRF905_data_ready(&NRF905_2);
          state_AM = NRF905_address_matched(&NRF905_2);
          if (state_DR && state_AM) {
            NRF905_read(&NRF905_2, &DatNRF2_RX, BufSizeRX);
            HL2_tg;
          }
          break;
        case 3:
          NRF905_tx(&NRF905_3, ADDRESS_SLAVE3, &DatNRF3_TX, BufSizeTX, NRF905_NEXTMODE_RX);
          NRF905_rx(&NRF905_3);
          //osDelay(2);
          state_DR = NRF905_data_ready(&NRF905_3);
          state_AM = NRF905_address_matched(&NRF905_3);
          if (state_DR && state_AM) {
            NRF905_read(&NRF905_3, &DatNRF3_RX, BufSizeRX);
            HL3_tg;
          }
          break;
       default: break;
    }

  if(__HAL_TIM_GET_FLAG(&htim4, TIM_FLAG_UPDATE) != RESET)
  {
    __HAL_TIM_CLEAR_FLAG(&htim4, TIM_FLAG_UPDATE);
    timer++;
  }
  if(timer >= 4) timer = 1;
  UpdateStruct();
}
```

Рис 4.15. Функція обміну даними Exchange()

Функція Exchange() працює за допомогою конструкції swich() case та таймера TIM4. Таймер налаштовується при загальній ініціалізації на довільний відрізок часу, але за допомогою тестів було обрано період часу в 5 мілісекунд, саме за такий період радіопередавач встигає передати та отримати дані. Радіообмін працює наступним чином: 1) дана система надсилає дані і отримує дані у відповідь; 2) інша система перебуває в режимі прослуховування очікує на дані, якщо дані отримано система перемикається в режим передавача і передає дані.

Алгоритм передачі даних зображено на рисунку 4. виглядає він наступним чином :

- Змінні state_DR та state_AM слугують маркерами які вказують що дані надійшли для радіомодуля і адрес відправника підтверджено.
- Змінна timer слугує лічильником і збільшується на одиницю кожні 5 мілісекунд завдяки таймеру, якщо лічильник стає більшим за 3 він обнуляється.
- Завдяки конструкції swich() case та лічильнику timer запускаються по черзі три канали передачі та прийому даних для трьох радіомодулів.
- Функція NRF905_tx(&NRF905_1, ADDRESS_SLAVE1, &DatNRF1_TX, BufSizeTX, NRF905_NEXTMODE_RX) приймає декілька аргуметів, серед них : NRF905_1 – структура з даними радіомодуля вказує з яким саме радіомодулем слід працювати, ADDRESS_SLAVE1 – адрес, який надається передавачеві для передачі даних, DatNRF1_TX – корисне навантаження, яке потрібно передати, BufSizeTX – розмір буферу даних який потрібно створити для передачі, NRF905_NEXTMODE_RX – вказівник що вказує радіопередавачеві що наступним буде режим прийому даних. Ця функція повністю обробляє передачу даних для радіомодуля і є достатньо функціональною та гнучкою.
- Функція NRF905_rx((&NRF905_1) приймає аргументом структуру даних радіомодуля та переводить його в режим прослуховування ефіру та прийому радіоданих.
- Функція NRF905_data_ready() заміняє собою фізичний пін DR та зчитує відповідний регістр, якщо ж дані надійшли на радіомодуль дана функція записує одиницю в змінну state_DR.
- Функція NRF905_address_matched() заміняє собою фізичний пін AM та зчитує відповідний регістр, якщо адреса відправника даних збігається з записаним адресом прийому радіомодуля дана функція записує одиницю в змінну state_AM.
- Умова if (state_DR && state_AM) перевіряє чи прийшли дані для радіомодуля, та чи збігається адрес, якщо дана умова виконується то радіомодуль зчитує дані за допомогою функції.

- Функція NRF905_read(&NRF905_1, &DatNRF1_RX, BufSizeRX) має наступні аргументи: NRF905_1 – структура з даними радіомодуля, яким потрібно оперувати, DatNRF1_RX – буфер для запису отриманих даних, BufSizeRX – розмір буферу для отриманих даних. Дана функція переписує отримані дані з буферу радіопередавача в буфер мікроконтролера.
- Функція HL1_tg змінює статус піна GPIO на протилежний, при обміні даними ця функція дає ефект швидкого миготіння світлодіоду.
- Функція UpdateStruct() оновлює дані буферу DatNRF1_TX з структури обміну даними інтерфейсу Ethernet та записує дані з буферу DatNRF1_RX в дану структуру.

На рисунку 4.16 зображено проектну розробку в робочому режимі всі три радіомодулі передають та отримують дані про що свідчать три блимаючи світлодіоди синього кольору, якщо ж обмін пропаде чи обірветься в одного з модулів про це сигналізуватимуть світлодіоди.

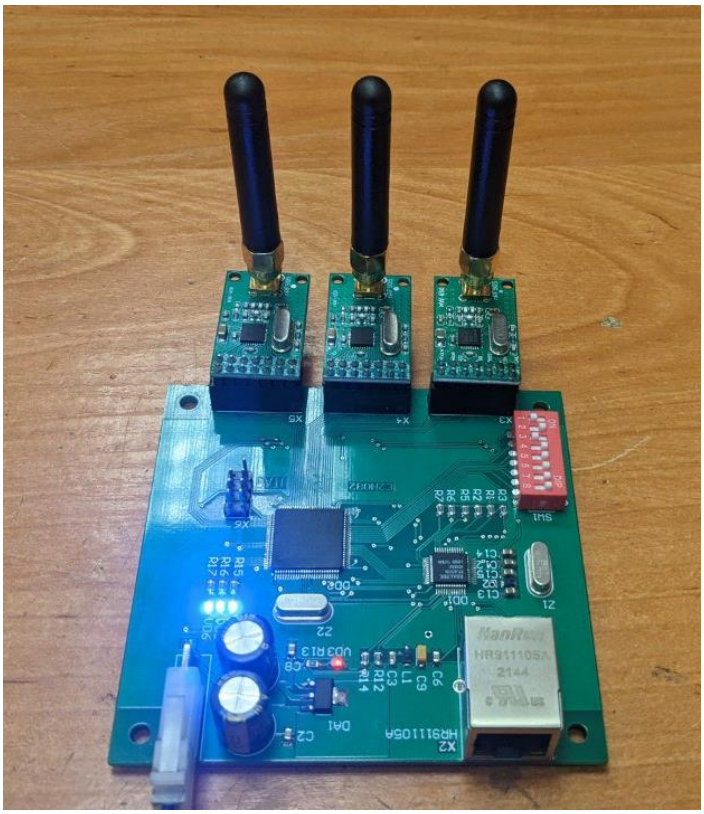


Рис 4.16. Готовий прототип девайсу в роботі

На рисунку 4.17 Зображено програму для ПК CS Igla яка обмінюється даними з розробленою платою обміну. В даному випадку ця плата застосовується для бездротового обміну з автономним тренажером ПЗРК Ігла, та передає дані гіроскопу, акселерометра та інших датчиків для програми візуалізації CS Igla та отримує дані управління. Рядки TXcnt та RXcnt відображають кількість прийнятих та відправлених пакетів що свідчить про обмін між програмою та платою обміну.

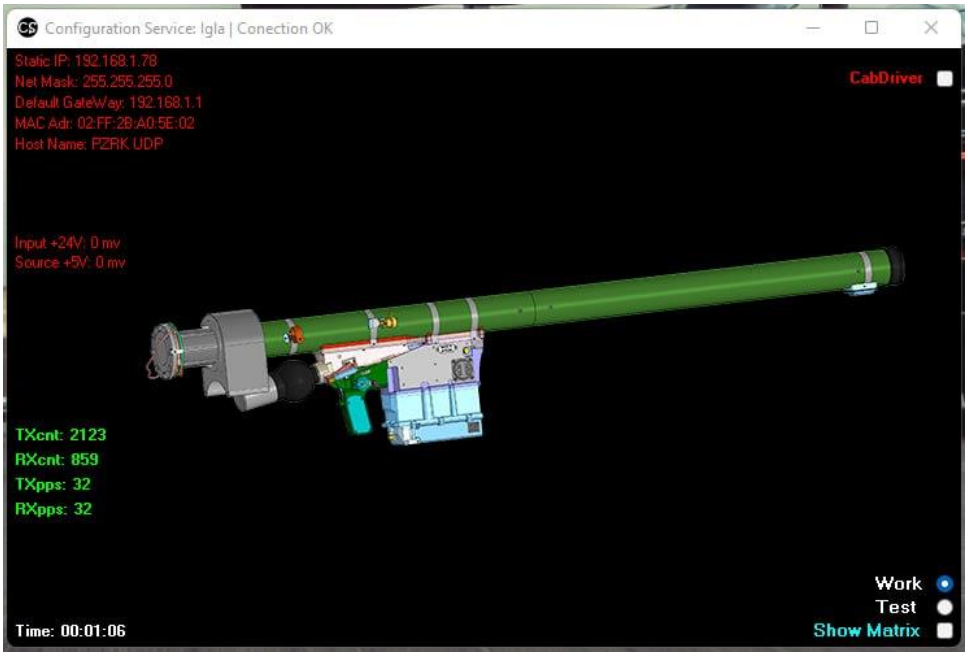


Рис. 4.17. Програма CS Igla, головне меню

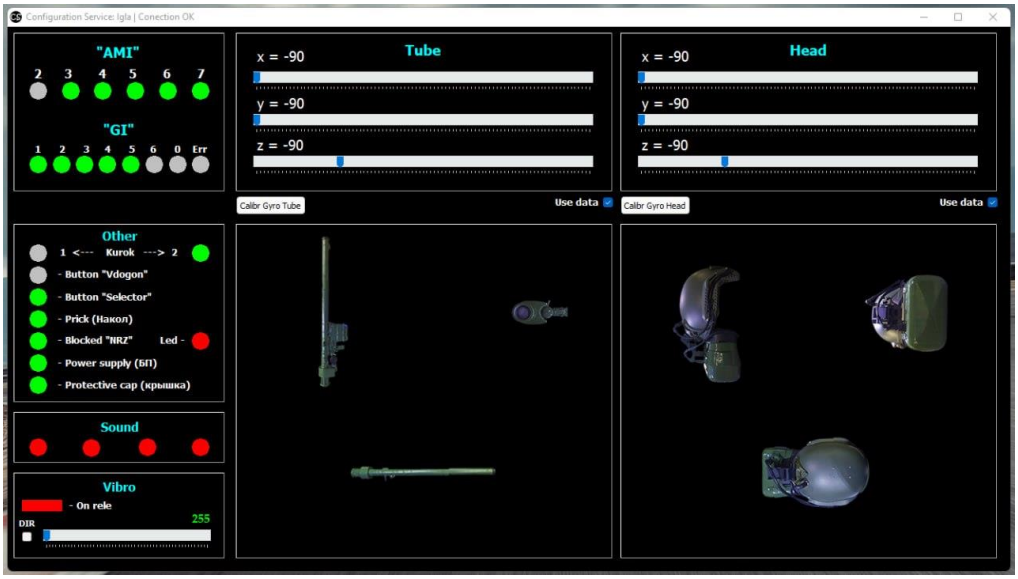


Рис 4.18. Програма CS Igla, зображення роботи датчиків

ВИСНОВОКИ

У даній науковій роботі було розроблено ефективну бездротову систему передачі даних на базі мікроконтролера сімейства STM32 та радіопередавачів NRF905. Використання цих радіомодулів дозволило досягти високої продуктивності та ефективності в умовах обмежених ресурсів. Результати дослідження показали, що така система може сприяти подальшому розвитку технологій бездротового зв’язку та розширенню їхнього використання в різних галузях.

Отримані результати можуть бути використані для розробки та впровадження нових бездротових систем передачі даних у практиці, що може знайти застосування в створенні високоефективних та надійних систем моніторингу, управління та обміну інформацією в різних галузях технологій та промисловості.

Таким чином, дана робота внесла значний вклад у розвиток бездротових систем передачі даних, відкриваючи нові можливості для подальшого вдосконалення та оптимізації цих систем.

Структура роботи: Робота складається з кількох розділів, включаючи вступ, теоретичну частину, практичну частину, висновки та список використаної літератури. В теоретичній частині розглядається теорія розробки друкованих плат, основи роботи мікроконтролерів STM32 та порівняння середовищ розробки. Практична частина включає в себе розробку друкованої плати, програмного забезпечення та тестування системи.

Теоретичні аспекти: Бездротові системи передачі даних використовують радіочастотні сигнали для передачі інформації між пристроями. Мікроконтролери STM32 відомі своєю високою продуктивністю, низьким споживанням енергії та гнучкістю, що робить їх ідеальним вибором для розробки таких систем.

Практичні аспекти: Розробка системи включає в себе проектування схеми, вибір компонентів, програмування мікроконтролера та тестування системи. В результаті було створено прототип системи, який було протестовано для перевірки його продуктивності та надійності.

Вплив на суспільство: Розробка таких систем може мати значний вплив на суспільство, оскільки вони можуть забезпечити більш ефективний обмін даними в різних галузях, таких як медицина, промисловість, телекомунікації тощо. Це може сприяти підвищенню продуктивності та ефективності цих галузей.

Майбутнє дослідження: Хоча дана робота вже внесла значний вклад у розвиток бездротових систем передачі даних, є ще багато можливостей для подальшого дослідження. Можливі напрямки майбутнього дослідження включають оптимізацію системи для підвищення її продуктивності, розробку нових алгоритмів для покращення ефективності передачі даних та дослідження можливостей використання нових технологій для покращення системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Климаш М. М., Колодій Р. С. Телекомунікаційні системи передавання інформації. Львів : Львів. політехніка, 2018. 632 с.
2. Ke-Lin Du, M. N. S. Swamy. Wireless communication systems from RF subsystems to 4G enabling technologies. Cambridge : Cambridge University Press, 2012.
3. Pakdel M. Advanced Programming with STM32 Microcontrollers. Elektor, 2020. 215 p.
4. STM32 32-bit Arm Cortex MCUs. [Електронний ресурс] – Режим доступу до
5. ресурсу:https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html.
6. Молчанов О. А., Сергієнко А. М., Романкевич В. О. Мікроконтролери зі стековою архітектурою. Problems of Informatization and Management. 2023. Т. 2, № 74. С. 74–80.
7. Aye K. S. Color Digital Sign Board using Altium Designer. International Journal of Trend in Scientific Research and Development. 2019. Volume-3, Issue-3. P. 916–918.
8. CMOS microcontrollers. Microprocessors and Microsystems. 1987. Vol. 11, no. 7. P. 395.
9. Gay W. FreeRTOS. Beginning STM32. Berkeley, CA, 2018. P. 59–72. [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.1007/978-1-4842-3624-6_5
10. Gay W. Introduction. Beginning STM32. Berkeley, CA, 2018. P. 1–16. [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.1007/978-1-4842-3624-6_1
11. Gay W. Real-Time Clock (RTC). Beginning STM32. Berkeley, CA, 2018. P. 175–193. [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.1007/978-1-4842-3624-6_10

12. Gazi O., Arlı A. Ç. Serial Peripheral Interface. State Machines using VHDL. Cham, 2021. P. 143–192. [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.1007/978-3-030-61698-4_4
13. Goldberg B. HF Radio Data Transmission. IEEE Transactions on Communications. 1961. Vol. 9, no. 1. P. 21–28.
14. Gu C. Building Embedded Systems. Berkeley, CA : Apress, 2016.
15. Kaler R. S. Microprocessors and Microcontrollers. I.K. International Publishing House Pvt. Ltd, 2019.
16. Liu G., Mao L. The Wireless Communication System Based on NRF905. Advances in Intelligent and Soft Computing. Berlin, Heidelberg, 2012. P. 87–91. [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.1007/978-3-642-27951-5_13
17. Mazidi M. A., Chen S., Ghaemi E. STM32 Arm Programming for Embedded Systems. MicroDigitalEd, 2018. 378 p.
18. Microcontrollers. Microprocessors and Microsystems. 1979. Vol. 3, no. 6. P. 294–295. [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.1016/0141-9331(79)90206-0
19. Nan J.-r., Wang D.-f., Sun F.-c. Application of wireless communication module based on nRF905 in battery Quick Change of electric vehicle. 2010 8th World Congress on Intelligent Control and Automation (WCICA 2010), Jinan, China, 7–9 July 2010. 2010[Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.1109/wcica.2010.5554295
20. Pakdel M. Fast PCB Design with Altium Designer. Central West Publishing, 2021.
21. Takenaka N. TB3157 - Serial Peripheral Interface(SPI) Communications on 8-Bit PIC MCU. Microchip Technology Incorporated, 2017.
22. The Wireless Image Monitoring System based on the Technology of nRF905 and WiFi / Z. Fu et al. The Open Cybernetics & Systemics Journal. 2015. Vol. 9, no. 1. P. 1117–1124.

23. Tyler N. Altium Designer 20 Launched. New Electronics. 2019. Vol. 51, no. 19. P. 9.
24. ZHANG Bao-jian., LIU Yan-chang., ZHAO Ming-fu. Design of Wireless Voting Control System Based on NRF905. Chinese Journal of Liquid Crystals and Displays. 2013. Vol. 28, no. 5. P. 752–758.
25. Zurawski R. Embedded Systems Handbook: Networked Embedded Systems. Taylor & Francis Group, 2017. 837 p.
26. В. Шолудько, М. Єсаулов, О. Вакуленко, Т. Гурський, М. Фомін Організація військового зв'язку. Центр навчальної літератури, 2023.
27. ARM Cortex-m3 STM32 [Електронний ресурс] – Режим доступу до ресурсу: https://bit.ly/2yMKUp4
28. Вибір дискретних компонентів [Електронний ресурс] – Режим доступу до ресурсу: https://bit.ly/2KwR1VP
29. Радіочастота [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Радіочастота
30. Поради Щодо Оптимізації Компонування Друкованої Плати [Електронний ресурс] – Режим доступу до ресурсу: https://www.jarltech.com.tw/uk/product/PCB+Layout+Tips.html
31. nRF905 [Електронний ресурс] – Режим доступу до ресурсу: https://infocenter.nordicsemi.com/pdf/nRF905_PS_v1.5.pdf

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

Дипломна робота

«Розробка системи передачі даних на базі мікроконтролерів сімейства STM32»

Київ 2024

Виконав студент:
Марценюк Артемій

Керівник:
Полоневич Ольга
Володимирівна,
к.т.н., доц.

ОСНОВНІ ХАРАКТЕРИСТИКИ РОБОТИ

Об'єкт дослідження – є бездротова система передачі даних на базі мікроконтролерів сімейства STM32.

Мета і завдання дослідження – розробка ефективної бездротової системи передачі даних на базі мікроконтролерів STM32. Завданнями дослідження є проектування системи, вибір відповідного обладнання, програмування мікроконтролера, тестування і оптимізація системи.

Методика дослідження – для досягнення поставлених завдань використовується комплексна методика дослідження, що включає в себе аналіз літературних джерел, експериментальні дослідження та програмне моделювання системи.

АКТУАЛЬНІСТЬ

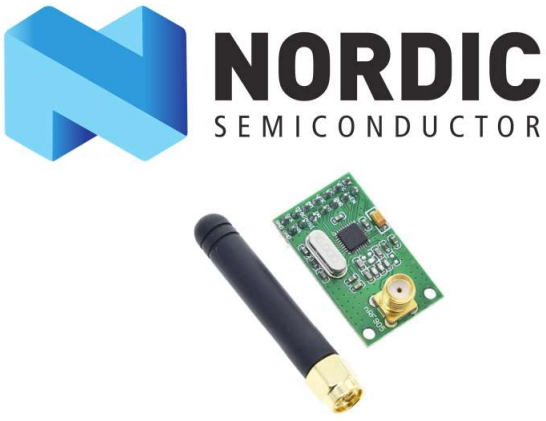
У сучасному світі, де технології розвиваються зі швидкістю світла, бездротові системи передачі даних стають все більш важливими. Вони забезпечують зручність і гнучкість в обміні інформацією між пристроями. Мікроконтролери сімейства STM32 відомі своєю ефективністю і надійністю, що робить їх ідеальною основою для розробки таких систем.

3

РАДІОПЕРЕДАВАЧ NRF905

Основні характеристики NRF905:

- Підтримка частотних діапазонів 433/868/915MHz
- Вихідна потужність +10dBm
- GFSK модуляція
- Швидкість передачі 50kbps
- Може бути як приймачем так і передавачем.

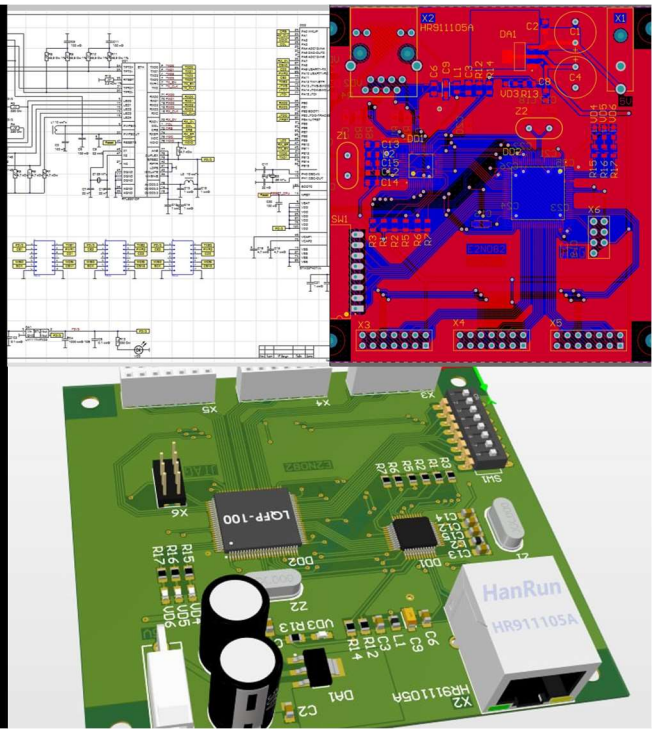


4

ДРУКОВАНА ПЛАТА

Розділ «розробка друкованої плати» складається з наступних етапів:

- Дослідження друкованих плат
- Аналіз та вибір середовища розробки друкованих плат
- Створення схемотехніки
- Трасування плати



ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Проаналізувавши переваги та недоліки було обрано наступне програмне забезпечення:

- Для розробки друкованої плати – Altium Designer
- Для написання програми – Keil uVision

Також не варто забувати про STM32 CubeMX, програму яка містить всю інформацію про мікроконтролери STM32 та безліч інструментів для написання коду та перевірки характеристик мікроконтролера.

KEIL™ Tools by ARM

Altium Designer®

STM32 CubeMX

СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕСТИ

Для даного проекту використовуються наступні бібліотеки: HAL (Hardware Abstraction Layer) STM32, RTOS2, CMSIS, PHY_DP83848C, EMAC, NRF905(GitHub) та ін.

Для збільшення швидкості та завадостійкості передачі даних було реалізовано 2 незалежні потоки за допомогою RTOS2.

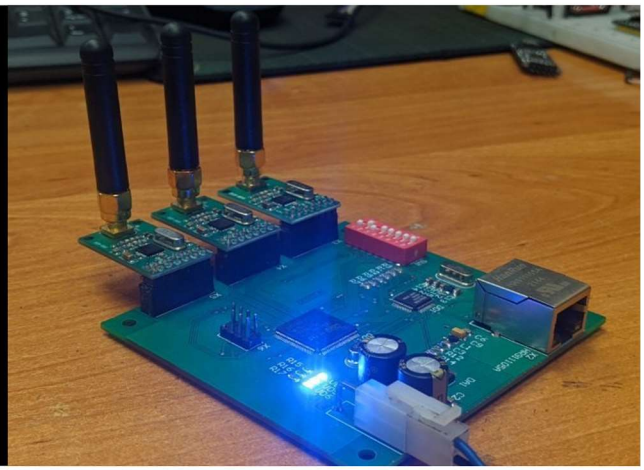
Реалізація обміну з ПК за допомогою протоколу UDP та налаштування IP-адресу задопомогою перемикачів на платі.

Реалізація безперебійної роботи трьох передавачів NRF905 та швидкісної передачі інформації в трьохканальному режимі.

ТАЙМЛАЙН ПРОЕКТУ

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Основним результатом дослідження розроблене рішення для одночасного швидкого радіообміну даними між 3 абонентами та ПК. Дане рішення може знайти багато застосувань в сфері IoT.



9

ВИСНОВКИ

Не зважаючи на стрімкий розвиток інформаційних технологій не завжди знаходиться рішення яке може повною мірою задовольнити потреби користувача. В такому випадку необхідна розробка нових рішень здатних покрити ці потреби.

Розробка нових рішень це ресурсоємкий проект який вимагає багато часу та ресурсів.

Бездротові технології передачі даних дають змогу безлічі мобільних девайсів працювати в різноманітних умовах та сценаріях.

10


END
NEAR
THANK YOU