ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «МУЛЬТИМЕТР НА БАЗІ STM32 З МОБІЛЬНИМ УПРАВЛІННЯМ»

на здобуття освітнього ступеня магістра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

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

_____ *Роман АВРАМЧУК*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. КСДМ-62
Роман Аврамчук
Ім'я, ПРІЗВИЩЕ

Керівник: Сергій КОРОТКОВ
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

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

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерної інженерії

Ступень вищої освіти Магістр

Спеціальність 123 Комп'ютерна інженерія

Освітньо-професійна програма Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
Ім'я, ПРІЗВИЩЕ
«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Аврамчук Роман Сергійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Мультиметр на базі STM32 з мобільним управлінням
керівник кваліфікаційної роботи: Сергій КОРОТКОВ, phD (доктор філософії), доцент
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024р. №320
2. Строк подання кваліфікаційної роботи «29» грудня 2024р.
3. Вихідні дані до кваліфікаційної роботи:
   1. Мікроконтролери.
   2. Мультиметр.
   3. Застосунок
   4. Науково-технічна література по темі магістерської роботи.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. Аналіз актуальності теми та написання вступу до роботи
   2. Реалізація практичної частини роботи та їх взаємозв'язків.
   3. Проектування та перевірка системи мікроконтролерної частини та частини застосунка
5. Перелік ілюстративного матеріалу: *презентація*
   1. Мета, об'єкт, предмет та актуальність дослідження
   2. Принцип дії тестера
   3. Базові компоненти проєкту
   4. Компоненти проєкту
   5. Програмні компоненти
   6. Принцип роботи lc-контуру
   7. Алгоритм програми lc-контуру

8. Алгоритм програми осцилографу
9. Алгоритм тестеру транзисторів
10. Перевірка роботи тестеру
11. Порівняння з подібними мультитестерами
12. Висновки
13. Публікації та апробація роботи

6. Дата видачі завдання «01» вересня 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Підбір технічної літератури	05.09.24-10.09.24	
2.	Визначення характеристик електричних компонентів.	14.09.24-15.09.24	
3.	Вибір елементарної бази проекту.	20.09.24-25.09.24	
4.	Визначення характеристик електричних компонентів.	2.10.24-6.10.24	
5.	Реалізація практичної частини роботи та їх взаємозв'язків.	8.10.24-25.11.24	
6.	Оформлення та висновки по роботі	25.11.24-03.12.24	
7.	Оформлення роботи	12.11.24-25.12.24	
8.	Розробка демонстраційних матеріалів, доповіді.	25.11.24-5.12.24	

Здобувач вищої освіти _____ Роман Аврамчук
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Сергій КОРОТКОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття ступня магістр: 70 стор., 18 рис., 1 табл., 18 джерел.

Мета роботи - Розробка мультиметра на базі мікроконтролера STM32 з можливістю мобільного управління для вимірювання електричних параметрів, таких як напруга, струм, опір, а також індуктивність і ємність, з відображенням результатів у мобільному додатку.

Об'єкт дослідження - Процес вимірювання електричних величин за допомогою мікроконтролерів.

Предмет дослідження - Апаратно-програмна реалізація мультиметра з використанням STM32 і бездротового зв'язку для мобільного управління.

Методи дослідження – Аналіз схемотехнічних рішень для побудови мультиметрів. Розробка прошивки для STM32 з використанням HAL-бібліотеки. Реалізація Bluetooth-зв’язку між мікроконтролером і мобільним додатком. Тестування функціональних можливостей і точності вимірювань.

У даній роботі було створено макет багатофункціонального вимірювального пристрою, здатного проводити різні типи вимірювань. У процесі роботи детально досліджено функціональні можливості цього пристрою та проведено тестування його застосування в реальних умовах. Особливу увагу приділено моделюванню сценаріїв використання пристрою для проведення вимірювань різних електричних величин, що забезпечує його практичну цінність.

Результати дослідження підтвердили, що при дотриманні правил використання Android-додатка, створеного для управління пристроєм, та самого вимірювального модуля, цей інструмент може бути ефективно застосований у лабораторних умовах. Його відмінними рисами є зручність у використанні та

економія часу під час проведення вимірювань. Отримані висновки вказують на можливість використання пристрою як сучасного і доступного інструменту для навчальних цілей, технічного аналізу та експериментальної роботи.

Галузь використання – Електроніка, радіоінженерія, навчальні лабораторії, автоматизація тестування електронних компонентів, а також побутові й професійні пристрої для вимірювання електричних параметрів.

КЛЮЧОВІ СЛОВА: ВИМІРЮВАННЯ, BLUETOOTH, МІКРОКОНТРОЛЕР STM32, NUCLEO, ANDROID, ОПЕРАЦІЙНИЙ ПІДСИЛЮВАЧ.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 70 pages, 18 figures, 1 tables, 18 sources.

Purpose of the work - Development of a multimeter based on the STM32 microcontroller with the possibility of mobile control for measuring electrical parameters, such as voltage, current, resistance, as well as inductance and capacitance, with the display of results in a mobile application.

Object of research - The process of measuring an electrical quantity using microcontrollers.

Subject of research - Hardware and software implementation of a multimeter using STM32 and wireless communication for mobile control.

Research methods - Analysis of circuit solutions for building multimeters. Development of firmware for STM32 using the HAL library. Implementation of Bluetooth communication between the microcontroller and the mobile application. Testing of functionality and measurement accuracy.

In this work, a model of a multifunctional measuring device capable of performing various types of measurements was created. In the process of work, the functionality of this device was studied in detail and its application in real conditions was tested. Particular attention is paid to modeling scenarios for using the device to measure various electrical quantities, which ensures its practical value.

The results of the study confirmed that, subject to the rules for using the Android application created to control the device and the measuring module itself, this tool can be effectively used in laboratory conditions. Its distinctive features are ease of use and time saving during measurements. The conclusions obtained indicate the possibility of using the device as a modern and affordable tool for educational purposes, technical analysis and experimental work.

Field of application – Electronics, radio engineering, educational laboratories, automation of testing of electronic components, as well as household and professional devices for measuring electrical parameters.

KEYWORDS: MEASUREMENT, BLUETOOTH, STM32 MICROCONTROLLER, NUCLEO, ANDROID, OPERATIONAL AMPLIFIER.

ЗМІСТ

ВСТУП

У сучасному світі електроніка займає ключове місце в розвитку технологій, забезпечуючи функціонування складних систем, які використовуються в повсякденному житті, промисловості, науці та інших сферах. Для обслуговування, діагностики та розробки електронних пристроїв необхідне використання вимірювальних приладів, зокрема мультиметрів, які дозволяють вимірювати основні електричні параметри: напругу, струм, опір та інші характеристики.

Традиційні мультиметри зазвичай мають обмежений функціонал і вимагають фізичної присутності оператора для виконання вимірювань. З розвитком технологій бездротового зв’язку та мікроконтролерів з’являється можливість створення більш гнучких та функціональних вимірювальних приладів. Інтеграція мобільних пристроїв з мультиметрами дозволяє віддалено контролювати процес вимірювань, аналізувати результати та інтегрувати дані з іншими системами.

Історія розвитку мультиметрів почалася на початку XX століття з винаходу аналогових вимірювальних приладів, які поєднували функції вольтметра, амперметра та омметра. У 1920-х роках компанія Automatic Coil Winder and Electrical Equipment Company розробила перший у світі комбінований вимірювальний прилад під назвою "Avometer". Цей пристрій став прототипом сучасних мультиметрів. У наступні десятиліття розвиток технологій сприяв мініатюризації, підвищенню точності та функціональних можливостей цих приладів. Перехід від аналогових до цифрових мультиметрів у 1970-х роках став революційним кроком, забезпечивши точність вимірювань, автоматизацію процесів і наочність результатів.

У цій дипломній роботі розглядається розробка мультиметра на базі сучасного мікроконтролера STM32 із підтримкою мобільного управління. Використання STM32 як основи забезпечує високу продуктивність,

енергоефективність і можливість інтеграції різноманітних периферійних модулів. Впровадження мобільного управління розширює функціональні можливості пристрою, дозволяючи здійснювати дистанційне керування та передачу даних через інтерфейси Bluetooth.

Актуальність теми роботи полягає у створенні мультиметра нового покоління, який поєднує високу точність вимірювань із сучасними засобами комунікації. Такий підхід не лише забезпечує зручність використання, але й відкриває можливості для інтеграції мультиметра в системи Інтернету речей (IoT), автоматизовані виробничі лінії чи навчальні лабораторії.

Мета роботи – розробити функціональний мультиметр на базі мікроконтролера STM32 з можливістю мобільного управління, який відповідатиме сучасним вимогам до вимірювальних приладів.

Для досягнення цієї мети в роботі було проведено аналіз існуючих рішень та визначення їх переваг й недоліків; розробка архітектури мультиметра, включаючи апаратну частину та програмне забезпечення; реалізувано мобільний інтерфейс для управління пристроєм і отримання даних; протестувано створений пристрій на практичних прикладах.

Дана розробка може бути корисною для інженерів, науковців і студентів, що працюють з електронними схемами, а також для впровадження в освітніх та виробничих середовищах.

РОЗДІЛ 1 СТРУКТУРНА СХЕМА

1.1 Пошук існуючих рішень

Створення мультиметрів на базі мікроконтролерів STM32 відкриває широкі можливості для розробників завдяки гнучкості програмування та підтримці різноманітних функцій. Такі пристрої можуть виконувати багато завдань, включаючи вимірювання напруги, струму, опору, ємності, індуктивності, а також аналіз частоти сигналів.

Як видно на рис. 1.1., такі пристрої можуть бути корисними для радіоаматорів, електронних інженерів, у навчальних лабораторіях і навіть у промислових умовах. Їх можна адаптувати для вимірювання параметрів у вузькоспеціалізованих сферах, наприклад, у медичних або наукових дослідженнях.

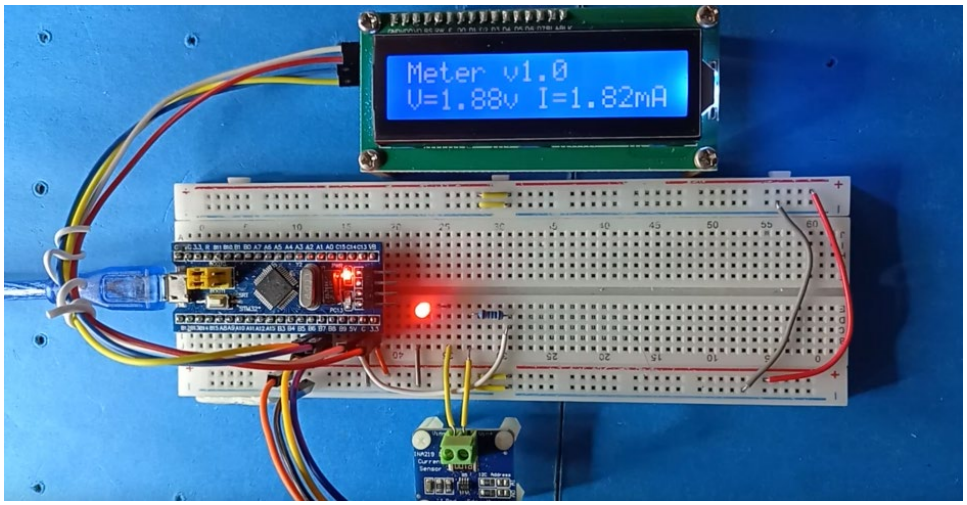

Рисунок 1.1 – Побудований радіоаматором вольт-амперметру на базі плати STM32 BluePill

Завдяки можливості гнучкої адаптації, мультиметри на базі STM32 здатні

виконувати як стандартні, так і унікальні завдання, задовольняючи потреби різних користувачів.

Основне завдання апаратної частини проєкту полягає у створенні приладу для визначення індуктивності (зокрема таких компонентів, як дроселі й трансформатори) і ємності (конденсаторів). Такий пристрій зазвичай називають LC-метром. Приклад роботи подібного тестера наведено на рис. 1.1.

Для вимірювання індуктивності існують різні підходи. Серед найпоширеніших методів можна виділити застосування мостової схеми Максвелла або використання вольтамперметра;

Ємність конденсаторів також може бути визначена використання мосту Макса Віна або методом заснований на визначенні часу заряджання конденсатора.

Проте універсальним рішенням, яке підходить для обох типів вимірювань, є резонансний метод. Його суть полягає у застосуванні LC-генератора, що працює спільно з компаратором. У цій схемі використовуються відомі параметри ємності та індуктивності, що дає змогу точно обчислити характеристики невідомих елементів.

Під час вимірювання змінюється частота коливань у резонансному контурі залежно від величин ємності чи індуктивності досліджуваного елемента. Отриманий сигнал підсилюється операційним підсилювачем, що забезпечує можливість точного визначення частоти. На основі цього обчислюються значення індуктивності або ємності з використанням відповідних формул.

Резонансний підхід є ефективним та універсальним способом аналізу характеристик електронних компонентів. Завдяки простоті реалізації та високій точності він часто використовується у сучасних LC-метрах, забезпечуючи їх функціональність і надійність.

Пристрій також можна оснастити датчиками температури, вологості або тиску, що розширює його функціональність і робить його корисним для спеціалізованих застосувань.

1.2 Загальна блок-схема тестеру та його складових

Робота тестера ґрунтується на принципі, представленому на схемі рис. 1.2. Основний процес починається з генерації мікроконтролером імпульсу, який забезпечує заряд конденсатора. Цей конденсатор разом із котушкою індуктивності формує LC-контур, у якому виникають коливання з певною резонансною частотою. Сигнал, що утворюється на виході LC-контуру, проходить через операційний підсилювач, який перетворює синусоїдальну хвилю в прямокутну форму.

Отриманий сигнал обробляється мікроконтролером, для визначення тривалості імпульсу. Виміряна тривалість дозволяє обчислити частоту коливань, що є ключовим параметром для визначення значень ємності та індуктивності компонентів.

Перемикання режимів роботи передбачена через застосунок, яка дозволяє обрати, які параметри будуть визначатися: ємність (для конденсаторів) або індуктивність (для котушок). Це забезпечує універсальність пристрою та можливість роботи з різними типами електронних компонентів.

Обчислені мікроконтролером значення передаються по UART інтерфейсу, після чого Bluetooth-модуль відправляє дані на мобільний пристрій з операційною системою Android. Передача здійснюється бездротово після попереднього сполучення смартфона з модулем, що робить роботу з тестером зручною та мобільною.

Завдяки такому підходу, тестер забезпечує високу точність вимірювань і зручність інтеграції з сучасними гаджетами, що значно розширює його функціональні можливості.

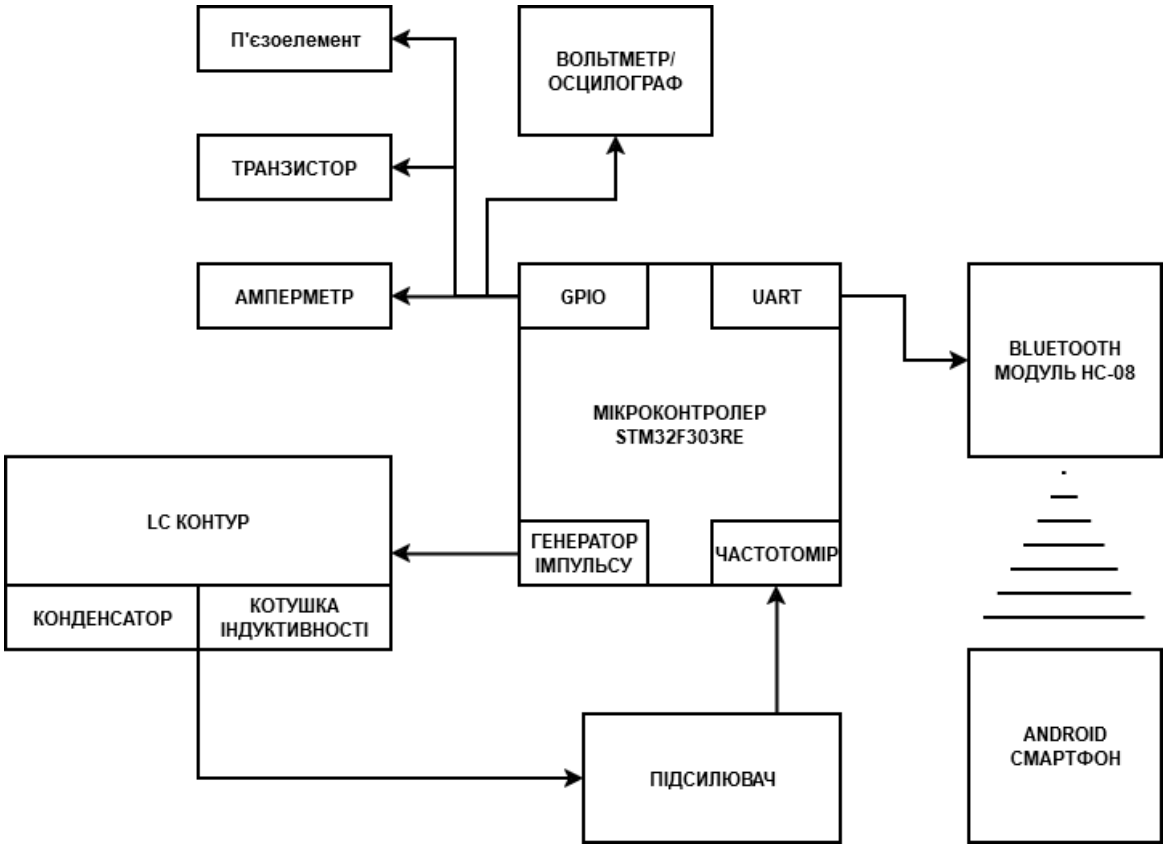

Рисунок 1.2 – Структурна схема проєкту

В інших режимах мікроконтролер працює в режимах виміру біполярних транзисторів, виміру току, вольтажу, супротиву, зчитування осцилограми або відіграванні звукового сигналу при замиканні вимірювальних щупів.

РОЗДІЛ 2 ЕЛЕМЕНТНА БАЗА СИСТЕМИ

2.1 Мікроконтролер STM32 F303RE

Сімейство STM32F303xD/E базується на високопродуктивній 32-бітній ARM® Cortex®-M4 Ядро RISC з FPU, що працює на частоті 72 МГц, і вбудованим числом з плаваючою комою блок (FPU), блок захисту пам’яті (MPU) і вбудована макрокомірка трасування (ETM).

Завдяки DSP-можливостям і швидкісним АЦП мікроконтролер використовується для обробки аудіо- та відеосигналів, у вимірювальних приладах та системах зв’язку. Точні АЦП та операційні підсилювачі дозволяють використовувати мікроконтролер у медичних моніторах, портативних діагностичних пристроях тощо.

STM32F303RE, що зображено на рис 2.1 часто застосовується у мультиметрах, осцилографах, аналізаторах сигналів, завдяки його здатності працювати з аналоговими сигналами.

Завдяки багатофункціональним таймерам і PWM-модуляції, цей мікроконтролер використовується в системах управління двигунами, промислових контролерах та робототехніці. Інтерфейси UART, SPI та I2C дозволяють інтегрувати мікроконтролер у розумні пристрої та мережі Інтернету речей (IoT).

Рисунок 2.1 – Мікроконтролер STM32F303RE в корпусі LQFP64

Комплексний набір режимів енергозбереження дозволяє проектування малопотужних програм. Сімейство STM32F303xD/E пропонує пристрої в різних компановках від 64 до 144 піна. Залежно від вибраного пристрою в комплект входять різні набори периферійних пристроїв. Про те загальні характеристики в лінійці однакові та описані в Табл. 2.1

Таблиця 2.1 – Загальні характеристики мікроконтролера сімейства STM32F303x

Ядро	Ядро: ARM ® Cortex ® -M4 32-розрядний процесор з 72 МГц FPU, однотактне множення і Апаратне відділення, 90 DMIPS (від CCM), DSP інструкція та MPU (блок захисту пам'яті).
Живлення	Діапазон напруг V DD, V DDA: від 2,0 В до 3,6 В.
Керування живленням	Скидання живлення під час увімкнення/вимкнення живлення (POR/PDR) – Програмований детектор напруги (PVD); Режими низького енергоспоживання: Сон, Стоп і Режим очікування; Напруга від батареї для RTC і резервних регістрів.

Таблиця 2.1 – Загальні характеристики мікроконтролера сімейства STM32F303x

Пам’ять	До 512 Кбайт флеш-пам’яті; 64 Кбайт SRAM, з апаратною перевіркою парності реалізовано на перших 32 Кбайтах.; Статичний прискорювач: 16 Кбайт SRAM увімкнено шина інструкцій і даних, з паритетом HW чек (CCM); Гнучкий контролер пам’яті (FSMC) для статичні пам'яті, з чотирма Chip Select.
Тактування	Кварцування генератора від 4 до 32 МГц; Генератор 32 кГц для RTC з калібруванням; Внутрішній 8 МГц RC з опцією x16 PLL. Внутрішній генератор 40 кГц.
Таймери	Підтримка до 14 таймерів; Один 32-бітний таймер і два 16-бітних таймера з до чотирьох IC/OC/PWM або лічильників імпульсів і квадратурний (інкрементний) вхід кодера; Три 16-розрядних 6-канальних розширеного керування таймери, до шести каналів ШІМ, генерація мертвого часу та аварійна зупинка; Один 16-бітний таймер з двома IC/OC, один OCN/PWM, генерація мертвого часу та аварійна зупинка; Два 16-бітних таймера з IC/OC/OCN/PWM, генерація мертвого часу та аварійна зупинка; Два сторожових таймера (незалежний, прозорий) – Один таймер SysTick: 24-розрядний лічильник; Два 16-розрядних базових таймера для управління ЦАП.

Таблиця 2.1 – Загальні характеристики мікроконтролера сімейства STM32F303x

I/O Порти	До 115 швидких операцій введення/виведення; Усі відображаються на зовнішніх векторах переривань; Кілька 5 V-толерантних.
АЦП	Чотири АЦП 0,20 мкс (до 40 каналів) з вибір роздільної здатності 12/10/8/6 біт, 0 до Діапазон перетворення 3,6 В, окремий аналог живлення від 2,0 до 3,6 В.
ЦАП	Два 12-розрядних канали ЦАП з аналоговим живленням від 2,4 до 3,6 В.
Інтерфейси зв'язку	CAN (2.0B Active); Три I2 C Швидкий режим плюс (1 Мбіт/с) зі споживчим струмом 20 мА, SMBus/PMBus, пробудження від STOP; До п’яти USART/UART (ISO 7816 інтерфейс, LIN, IrDA, керування модемом); До чотирьох SPI, від 4 до 16 програмованих бітів кадрів, два з мультиплексним напів/повним дуплексом Інтерфейс I2; Швидкісний інтерфейс USB 2.0 з LPM; Підримка інфрачервоного передавача.

Також, із особливостей, мікроконтролер має блок розрахунку CRC; матрицю внутрішніх з'єднань; 12-канальний контролер DMA (Direct Access Memory); сім надшвидких аналогових компараторів «rail to rail». з аналоговим живленням від 2,0 до 3,6 В; чотири операційних підсилювача, які можна використовувати в Режим PGA, доступ до всіх пінів аналогового живлення від 2,4 до 3,6 В; підтримка до 24 ємнісних каналів вимірювання сенсорна клавіша, лінійний і поворотний датчик дотику; календар RTC з будильником, періодичним пробудженням із режиму зупинки/очікування.

2.2 Плата STM32 Nucleo-64

Плата Nucleo-64 рис. 2.2 — це відладочна плата для розробки на основі мікроконтролерів STM32. Вона призначена для прискорення створення, тестування та налагодження застосунків для мікроконтролерів STM32. Плата Nucleo-64 підтримує мікроконтролери з 64-виводним корпусом (наприклад, STM32F303RE, STM32F401RE), а її дизайн оптимізовано для простоти використання та інтеграції з різними периферійними пристроями, у тому числі з платами Arduino.

Завдяки багатій функціональності та доступності плат розробки широко використовується для навчання мікроелектроніці.

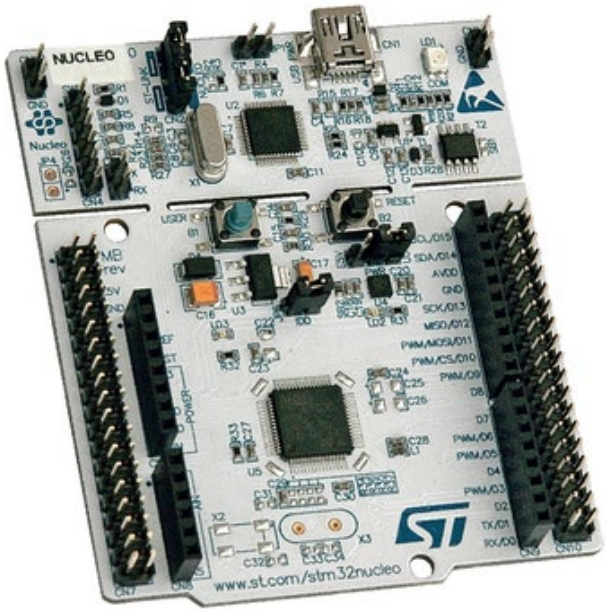

Рисунок 2.2 – Зовнішній вигляд відладочної плати Nucleo-64

Плата STM32 Nucleo має встановлений мікроконтролер STM32 в корпусі LQFP64; три світлодіоди(USB зв’язок (LD1), світлодіод користувача (LD2), світлодіод живлення (LD3)); Дві кнопки: USER і RESET; Два типи ресурсів розширення для підключення Arduino Uno V3 та розширювальні контактні роз’єми ST morpho для повного доступу до всіх входів/виходів STM32.

Живиться плата через USB VBUS або зовнішнє джерело (3,3 В, 5 В, 7-12 В).

Плата має вбудований налагоджувач і програматор ST-LINK/V2-1 з роз'ємом SWD та перемикач режимів вибору з використанням комплекту як автономний ST-LINK/V2-1.

Можливість повторного перерахування USB. На USB підтримуються три різні інтерфейси: віртуальний COM-порт, загальне сховище та порт налагодження.

2.3 Модуль HC-08

Bluetooth HC-08 рис. 2.3 - модуль Bluetooth з ультра низьким споживанням енергії. Цей пристрій використовує новітній протокол зв'язку - Bluetooth v4.0 BLE (Bluetooth Low Energy), що забезпечує включення передавача тільки на час відправки даних, що забезпечує можливість роботи від однієї батарейки типу CR2032 протягом декількох років.

Рисунок 2.3 – Bluetooth модуль HC-08

Модуль побудований на чіпі CC2540 Texas Instruments та підтримує протокол зв'язку Bluetooth v4.0 BLE. Радіус дії до 8-10 метрів. Цей модуль сумісний з усіма

Bluetooth-адаптерами, які підтримують SPP. Частота радіосигналу 2.40 - 2.48 ГГц. Інтерфейс USB 1.1 / 2.0 або UART

Енергоспоживання в залежності від режиму роботи і енергоспоживання - від 0,4 мкА до 21 мА.

Бездротова робоча частота модуля 2,4 ГГц ISM, модуляція GFSK. Максимальна потужність передавача 4 дБм, чутливість прийому -93 дБм. Виробник заявляє, що iphone4s може досягти 80 метрів наддалекого зв'язку у відкритому середовищі. Модуль використовує чіп CC2540, конфігурація простору 256K Bytespace, підтримує команду AT, користувач може відповідно за потребою змінити роль, серійний номер, швидкість передачі даних, назву модуля тощо.

2.4 Операційний підсилювач OPA2134PA

Операційний підсилювач (ОП) є одним із ключових елементів сучасної електроніки, який виконує широкий спектр функцій, пов’язаних з обробкою сигналів. Цей електронний компонент використовується для підсилення сигналів, їхньої фільтрації, зміни форми, а також для виконання математичних або логічних операцій. Завдяки своїй універсальності, операційні підсилювачі знаходять застосування у різних пристроях, таких як аудіосистеми, датчики, генератори сигналів та інші види електронного обладнання.

ОП можуть працювати з сигналами з різними характеристиками, включаючи джерела зі змінним рівнем імпедансу, наприклад, мікрофони, сенсори або інші аналого-цифрові модулі. Їхні можливості включають точне підсилення слабких сигналів, реалізацію інтеграторів, диференціаторів, активних фільтрів або компараторів, що дозволяє вирішувати широкий спектр інженерних задач.

У межах розробки цього проєкту операційний підсилювач відіграє важливу

роль, забезпечуючи коректну роботу LC тестера. Схема операційного підсилювача OPA2134PA, показана на рис. 2.4. Використання ОП, є ключовим у виконанні поставлених задач даної роботи.

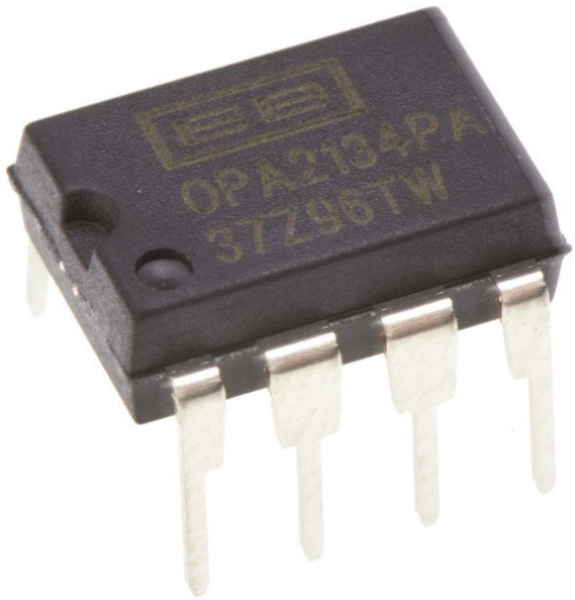

Рисунок 2.4 – Вигляд OPA2134PA з двома ОП в корпусі DIP8

Операційний підсилювач, який використовується в проєкті, має низку технічних характеристик, що забезпечують його ефективність у різних електронних схемах. Він випускається в корпусах PDIP або SOIC, що дозволяє використовувати його як у прототипах, так і в серійних виробах. Основні параметри підсилювача включають вхідний струм на рівні 5 пА, рекомендовану напругу живлення в межах 2.5–18 В, а максимальна напруга живлення може досягати 36 В. Завдяки високій швидкості приросту напруги на виході — 20 В/мкс і частоті одиничного підсилення 8 МГц, цей компонент ідеально підходить для швидких і точних обчислень.

У конструкції підсилювача передбачено два незалежні операційні блоки, що дозволяє використовувати його для виконання кількох задач одночасно. Розташування роз’ємів входів і виходів підсилювача наведено на рис. 2.4, що

полегшує його інтеграцію в електронні схеми. Ці характеристики роблять операційний підсилювач важливим елементом проєкту, забезпечуючи стабільну та надійну роботу системи.

2.5 Датчик струму ACS724

Модуль ACS724, що зображено на рис. 2.5, є простим носієм двонаправленого датчика струму Broadcom ±40A ACHS-7124 на основі ефекту Холла, який містить низькоопоровий шунт з опором (~ 0.7 мОм) та електричну ізоляцію до 3 кВ RMS. Ця версія датчика вимірює двонаправлений струм з величиною до 40 А і виводить пропорційну аналогову напругу (50 мВ/А) з центром 2.5 з типовою похибкою ±1.5%. Датчик живиться напругою від 4.5 до 5.5 В і призначений для використання в системах з напругою живлення 5 В.

Рисунок 2.5 - Датчик струму ACS724

Використання датчика Холла відкриває можливості для електричної ізоляції між струмовим шляхом і датчиком, забезпечуючи безпечну експлуатацію навіть за високих напруг. Завдяки цьому інтегральна схема (IC) може бути встановлена у будь-якому місці струмового шляху, зберігаючи ефективність і функціональність. Така ізоляція витримує напругу до 3 кВ RMS, що особливо важливо для застосувань, де потрібно уникати прямого контакту між електронікою і силовими компонентами. Крім того, смуга пропускання датчика становить 80 кГц, і її можна налаштувати, додавши конденсатор до фільтра. Заводське калібрування забезпечує високу точність – загальна похибка становить ±1,5% при кімнатній температурі, а стабільність роботи підтримується навіть за складних умов експлуатації. Мінімальний магнітний гістерезис дозволяє отримувати максимально точні результати.

Додаткові зручності включають маркування контактів на платі, яке спрощує підключення. Також стрілка на шовкографії вказує напрямок позитивного потоку струму через "+i". Для роботи з кількома версіями (±10A, ±20A, ±30A, ±40A, ±50A) передбачено білий квадрат для маркування та текст на корпусі датчика для ідентифікації. Схема датчику ACHS-7124 показана на рис. 2.6. Він розрахований на напругу живлення 5 В і вимірює двонаправлений струм у діапазоні від -40 А до 40 А. Його вихідний сигнал центрується на 2.5 В і змінюється на 50 мВ на ампер, де позитивний струм підвищує напругу, а негативний знижує.

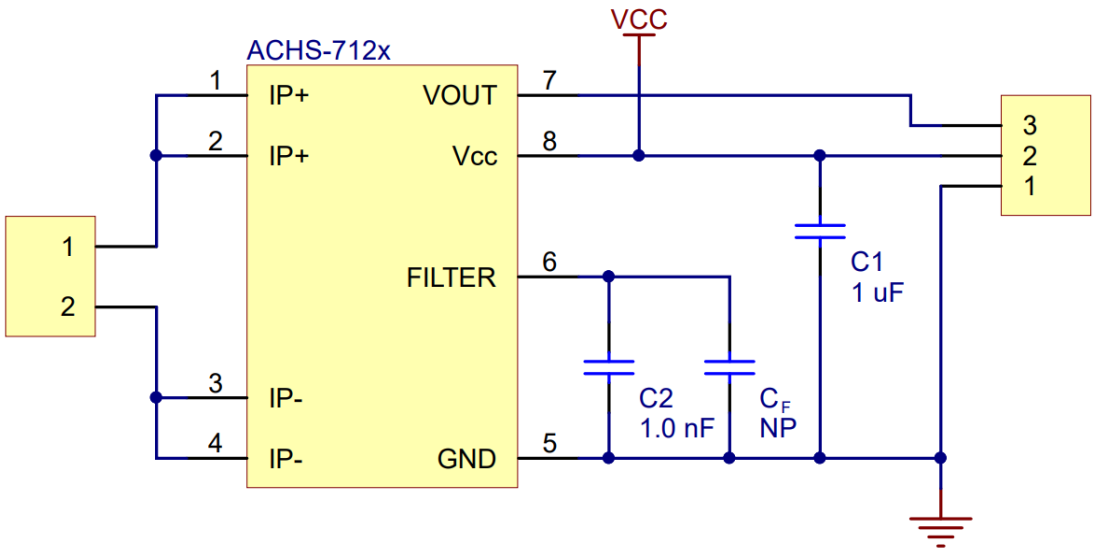

Рисунок 2.6 – Схема Датчику струму ACS724

Контакт FILTER дозволяє налаштовувати пропускну здатність плати, додаючи зовнішній конденсатор CF, який підключається до заземлення. Для зручності поруч із цим контактом передбачена заземлююча площадка, що полегшує монтаж. На платі вже встановлено конденсатор ємністю 1 нФ, але за потреби можна підключити додатковий конденсатор, щоб змінити характеристики фільтрації. У стандартній конфігурації, без зовнішнього конденсатора, смуга пропускання становить 80 кГц. Якщо потрібно зменшити цю смугу, наприклад, для усунення високочастотних перешкод, варто додати конденсатор з потрібною ємністю.

2.6 Кварцовий резонатор

Кварцовий резонатор на частоту 32,768 кГц, також відомий як "часовий кварц" типу HF-26, широко застосовується для організації годинників реального часу (RTC) та інших пристроїв, які потребують точного відліку часу. Ці

компоненти забезпечують високу стабільність частоти та надійність в роботі, що робить їх незамінними в аналогово-цифрових ланцюгах для стабілізації та генерації коливань певної частоти. У порівнянні з іншими пристроями, що використовуються для стабілізації частоти, кварцові резонатори демонструють кращу стабільність характеристик, навіть за умов зміни зовнішніх факторів, таких як температура чи вологість. Зовнішній вигляд зображено на рис 2.7.

Рисунок 2.7 – Кварцовий резонатор на 32 МГц

Основні характеристики цього резонатора роблять його особливо привабливим для точних застосувань. Частота становить 32,768 кГц, а корпус виконаний у вигляді компактного циліндра розміром 2×6 мм (HF-26). Максимальне річне відхилення частоти становить лише ±3 ppm, а температурна стабільність — ±20 ppm у робочому діапазоні температур від -10 до +70 °C. Крім того, резонатор має ємність навантаження 12,5 пФ, що дозволяє ефективно інтегрувати його у сучасні електронні системи.

2.7 Мобільний застосунок

Android — це одна з найпоширеніших операційних систем для мобільних пристроїв, яка базується на ядрі Linux. Вона підтримує широкий спектр пристроїв, включаючи смартфони, планшети, розумні годинники, телевізори та автомобільні мультимедійні системи. Android надає розробникам гнучкі можливості для створення програм через використання мови програмування Java або Kotlin через IDE Android Studio, логотип зображено на рис 2.8, а також багатий набір бібліотек, інструментів та API.

Для створення мультиметра з керуванням через Android необхідно забезпечити двосторонній обмін даними між мікроконтролером (STM32) та мобільним пристроєм

Android-додаток надсилає команду на STM32 для вибору режиму роботи (наприклад, вимірювання напруги). STM32 виконує вимірювання, обробляє дані та передає їх через Bluetooth. Додаток отримує дані, відображає їх у реальному часі та зберігає, якщо це необхідно.

Рисунок 2.8 – Логотип Android Studio

Користувач підключає вимірювальні зонди мультиметра до ланцюга. За допомогою Android-додатка він обирає режим "Вимірювання напруги". Додаток підключається до STM32 через Bluetooth, і після цього дані про напругу з’являються на екрані мобільного пристрою.

2.8 Інші компоненти

Для створення прототипу тестера використовувались різноманітні електронні компоненти, необхідні для забезпечення його коректної роботи. Зокрема, були залучені конденсатори, які виступали основними елементами для тестування ємності в схемах. Їх застосування дозволило вивчити поведінку LC-контуру, визначаючи його резонансну частоту. Також до складу прототипу включені котушки індуктивності, які забезпечили можливість вимірювання

індуктивності, що є ключовим параметром у роботі індуктивних компонентів.

Окрім цього, для повноцінної роботи тестера знадобилися транзистори, які використовувались для оцінки їхніх параметрів, таких як тип, робочий стан і здатність до перемикання. Резистори номіналом 100 Ом і 10 кОм виконували роль обмежувачів струму та забезпечували стабільну роботу схем. П’єзоелемент був використаний для реалізації функції звукового сигналу, що дозволило додати інтуїтивний зворотний зв'язок під час роботи з тестером, наприклад, у режимі "прозвонки". Такий набір компонентів дозволив побудувати універсальний пристрій для вимірювання параметрів різноманітних електронних елементів.

РОЗДІЛ 3 АЛГОРИТМ РОБОТИ МІКРОКОНТРОЛЕРНОЇ СИСТЕМИ

3.1 LC тестер

LC-метр — це електронний прилад, основним завданням якого є вимірювання індуктивності (L) і ємності (C) електричних компонентів. Його робота базується на принципах аналізу параметрів електричних ланцюгів, що дозволяє точно оцінювати характеристики різних елементів, таких як котушки індуктивності та конденсатори. Конструкція LC-метра зазвичай включає дві основні частини: генератор змінного струму та вимірювальний блок. Генератор виробляє сигнал певної частоти, який надходить на вимірювальний блок, де здійснюється аналіз і вимірювання параметрів підключеного електричного компонента.

Широка область застосування LC-метрів охоплює як повсякденні завдання, так і спеціалізовані технічні дослідження. В електроніці такі прилади незамінні для діагностики індуктивних і ємнісних компонентів, що особливо актуально для інженерів та техніків, які займаються обслуговуванням та ремонтом електронних пристроїв. Крім того, LC-метри знаходять своє місце у науковій сфері, де вони використовуються для визначення параметрів матеріалів і конструкцій, що потребують детального аналізу індуктивності та ємності.

Ключовим елементом, на якому базується робота LC-метра, є LC-контур. Ця схема володіє унікальною властивістю — резонансом, тобто здатністю досягати специфічної частоти, за якої індуктивність і ємність взаємодіють у гармонії. Резонансна частота, яка позначається як f, є основою для визначення характеристик компонентів. Цікаво, що резонансну частоту можна регулювати, змінюючи значення індуктивності або ємності контуру. Це відкриває можливості для широкого спектра застосувань, таких як фільтрація сигналів, налаштування частоти

в коливальних схемах, стабілізація напруги у джерелах живлення та інших електричних системах.

LC-контури також знаходять застосування як чутливі елементи у датчиках, які реагують на зміну зовнішніх параметрів, таких як температура, вологість або тиск. У таких пристроях зміна фізичного параметра впливає на індуктивність або ємність контуру, що призводить до зсуву резонансної частоти. Аналіз цих змін дозволяє точно визначати значення фізичних величин.

Для обчислення значень індуктивності або ємності використовуються математичні методи, що базуються на залежності частоти та параметрів сигналу. Зокрема, це може бути реалізовано через формулу (1.1), яка зв'язує частоту сигналу з характеристиками прямокутної хвилі. Цей підхід забезпечує високу точність вимірювань і дозволяє LC-метру стати універсальним інструментом для вирішення завдань у галузі електроніки та суміжних дисциплін.

$$f = \frac{1}{2t} \tag{1.1}$$

Тут t — час, який визначається вона дозволяє вимірювати тривалість одного імпульсу прямокутної хвилі, що є основою для подальших обчислень. LC-контур має резонансну частоту, яка розраховується за відомою формулою (1.2):

$$f = \frac{1}{2\pi\sqrt{(LC)}} \tag{1.2}$$

Цей вираз дозволяє знайти як індуктивність (L)(1.3), так і ємність (C)(1.4) при відомих інших параметрах:

$$L = \frac{1}{4\pi^2 f^2 C} \tag{1.3}$$

$$C = \frac{1}{4\pi^2 f^2 L} \tag{1.4}$$

Особливістю роботи LC-контуру є те, що синусоїдальний сигнал, який генерується під час резонансу, має однакову тривалість позитивної та негативної напівхвиль. Для спрощення вимірювань цей сигнал перетворюється на прямокутну хвилю за допомогою компаратора, збудованого на основі операційного підсилювача. В результаті отримуємо сигнал із коефіцієнтом заповнення 50% (тривалість високого та низького рівнів однакова).

Функція *HAL_TIM_OnePulse_Start* в STM32 дозволяє визначити тривалість одного імпульсу прямокутної хвилі. Оскільки частота визначається як зворотна величина періоду, а період повної хвилі складається з двох напівперіодів, отримане, значення необхідно помножити на 2, щоб коректно розрахувати повний період хвилі.

На рис. 3.1 наведено повну схему тестерної частини розробленого проєкту. При створенні макета тестера було використано низку компонентів, що забезпечують стабільну роботу пристрою. Одним із основних елементів є резистори, серед яких R2 з опором 100 Ом, а також R1 з номіналом 10 кОм. Для захисту схеми були додані діоди D1 та D2 типу 1N4148, розраховані на струм до 0,15 А. Вони відіграють важливу роль у запобіганні пошкодженням електронних компонентів у разі перенапруги.

Застосовувалися конденсатори та котушки індуктивності різних номіналів для формування резонансного контуру. Для живлення операційного підсилювача використовувалося джерело напруги 3 В.

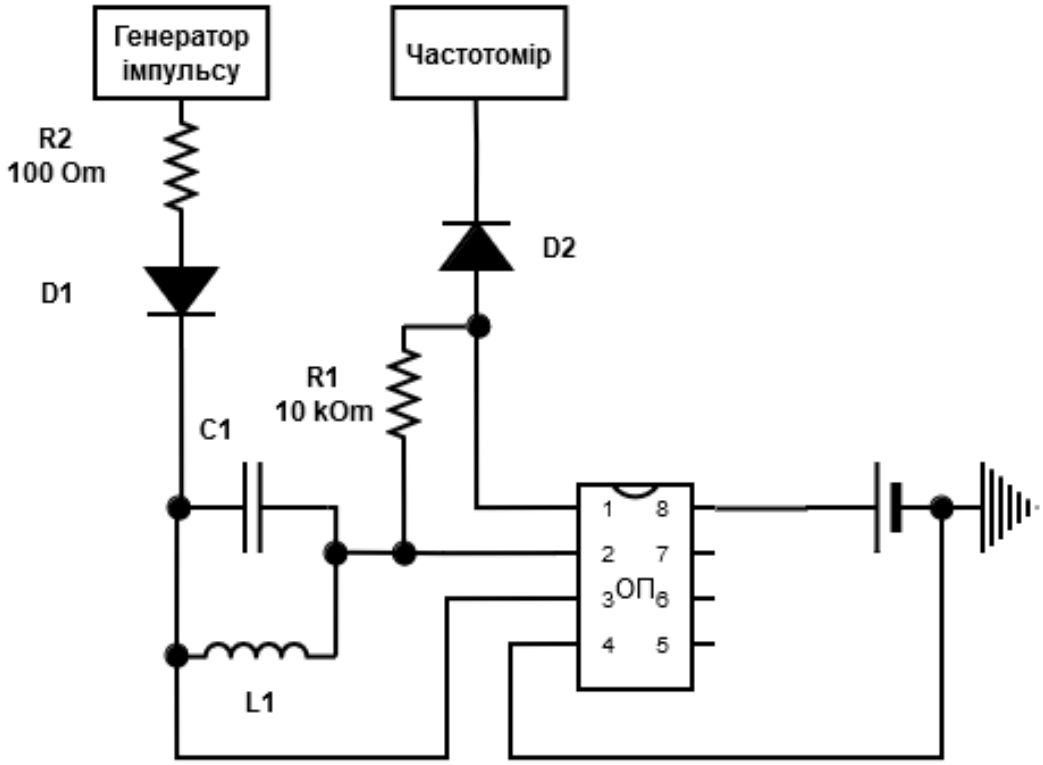

Рисунок 3.1 – Принципова схема блоку LC тестера

На схемі контакт операційного підсилювача: 8 – зовнішнє живлення 3В; 4 – земля; 2 – негативний зворотній зв’язок; 3 – позитивний.

На рис. 3.2 наведено загальну блок-схему роботи тестера, яка відображає його функціональну структуру. Вся система умовно поділяється на три основні частини: визначення частоти, обчислення індуктивності або ємності, а також перевірку загального стану тестера. Така логічна сегментація забезпечує чітку послідовність виконання операцій, що необхідні для точного вимірювання параметрів електронних компонентів.

Перший етап роботи алгоритму розпочинається з початкових налаштувань. Зокрема, мікроконтролер конфігурує режими роботи контактів, оголошує змінні, необхідні для обчислень, і встановлює частоту роботи інтерфейсу UART, який забезпечує зв’язок із зовнішніми пристроями. Крім цього, створюється спеціальна

структура, яка відповідає за вибір і контроль поточного режиму роботи тестера. Така підготовча частина алгоритму є основою для стабільного й ефективного функціонування системи.

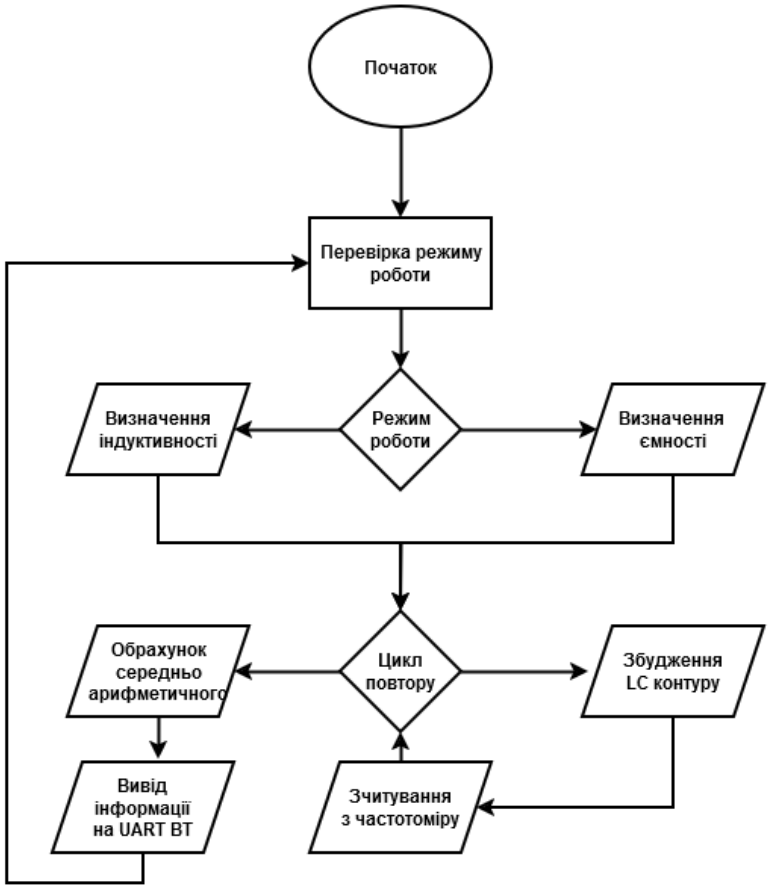

Рисунок 3.2 – Алгоритм LC тестеру

Подальший процес роботи тестера включає послідовне виконання вимірювань залежно від вибраного режиму. На основі вхідних сигналів визначається частота коливань, яка використовується для розрахунку індуктивності чи ємності вимірюваного компонента. Завдяки багаторівневій перевірці стану системи тестер здатний коректно реагувати на зміни умов роботи та своєчасно повідомляти користувача про можливі помилки або несправності,

забезпечуючи зручність та точність у повсякденному використанні.

```
#include "stm32f3xx_hal.h"
#include "math.h" // Для математичних операцій

// Глобальні змінні
volatile uint8_t mode = 0; // 0 - вимірювання індуктивності, 1 - вимірювання ємності
volatile uint8_t flag_ready = 0;
float frequency = 0.0;
float inductance = 0.0;
float capacitance = 0.0;

// Прототипи функцій
void SystemClock_Config(void);
void GPIO_Init(void);
void UART_Init(void);
void TIM_Init(void);
float MeasureFrequency(void);
void CalculateInductance(void);
void CalculateCapacitance(void);

int main(void) {
    HAL_Init(); // Ініціалізація HAL
    SystemClock_Config(); // Налаштування системного такту
    GPIO_Init(); // Ініціалізація GPIO
    UART_Init(); // Ініціалізація UART
    TIM_Init(); // Ініціалізація таймера
```

```
    while (1) {
      if (flag_ready) {
        flag_ready = 0; // Скидання прапорця

        // Збудження LC контуру та вимірювання частоти
        frequency = MeasureFrequency();

        if (mode == 0) {
          CalculateInductance(); // Обчислення індуктивності
          printf("Inductance: %.2f uH\r\n", inductance * 1e6); // Вивід через
UART
        } else {
          CalculateCapacitance(); // Обчислення ємності
          printf("Capacitance: %.2f nF\r\n", capacitance * 1e9); // Вивід через
UART
        }
      }
    }
  }

  // Ініціалізація GPIO
  void GPIO_Init(void) {
    __HAL_RCC_GPIOA_CLK_ENABLE();
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    // імітування кнопки зміни режиму
    GPIO_InitStruct.Pin = GPIO_PIN_0;
```

```
    GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    // Переривання від кнопки
    HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(EXTI0_IRQn);
}

// Ініціалізація UART
void UART_Init(void) {
    __HAL_RCC_USART2_CLK_ENABLE();
    UART_HandleTypeDef huart2;
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 9600;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
    HAL_UART_Init(&huart2);
}

// Ініціалізація таймера
void TIM_Init(void) {
    __HAL_RCC_TIM2_CLK_ENABLE();
    TIM_HandleTypeDef htim2;
```

```
    htim2.Instance = TIM2;
    htim2.Init.Prescaler = 7999; // Таймер працює на частоті 10 кГц
    htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim2.Init.Period = 0xFFFFFFFF;
    htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    HAL_TIM_Base_Init(&htim2);
    HAL_TIM_Base_Start(&htim2);
}
```

Для збудження LC-контуру використовується GPIO, який генерує початковий імпульс, що запускає коливання в контурі. Після цього таймер здійснює підрахунок тривалості імпульсів, що дозволяє визначити частоту коливань LC-контуру з високою точністю.

```
// Вимірювання частоти
float MeasureFrequency(void) {
    uint32_t start, end;
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET); // Збудження LC контуру
    HAL_Delay(1);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);

    start = __HAL_TIM_GET_COUNTER(&htim2);
    while (!HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_2)); // Чекаємо на імпульс
    end = __HAL_TIM_GET_COUNTER(&htim2);

    return (1.0 / ((end - start) * 1e-4)); // Частота в Гц
}
```

```
// Розрахунок індуктивності
void CalculateInductance(void) {
    float C = 1e-9; // Відома ємність (1 нФ)
    inductance = 1.0 / (4.0 * M_PI * M_PI * C * frequency * frequency);
}

// Розрахунок ємності
void CalculateCapacitance(void) {
    float L = 1e-3; // Відома індуктивність (1 мГн)
    capacitance = 1.0 / (4.0 * M_PI * M_PI * L * frequency * frequency);
}

// Обробник переривання для кнопки
void EXTI0_IRQHandler(void) {
    if (__HAL_GPIO_EXTI_GET_IT(GPIO_PIN_0)) {
        __HAL_GPIO_EXTI_CLEAR_IT(GPIO_PIN_0);
        mode = !mode; // Зміна режиму
        flag_ready = 1; // Активуємо вимірювання
    }
}
```

Далі, на основі отриманих даних частоти, спеціальні формули обчислюють значення індуктивності або ємності, залежно від обраного режиму. Результати обчислень передаються через UART-інтерфейс, що дає можливість легко передати їх на зовнішні пристрої, наприклад, Bluetooth-модуль для подальшої інтеграції з мобільними додатками на Android.

3.2 Осцилограф

Принцип роботи програми осцилографа ґрунтується на обробці даних, отриманих з аналогового входу. Як показано на блок-схемі рис. 3.3, значення сигналу з певним інтервалом часу записуються в масив, після чого аналізуються. На основі цих даних визначаються максимальна та мінімальна напруга сигналу, що дозволяє обчислити середнє значення та амплітуду. Для визначення частоти сигналу аналізується момент, коли він двічі перетинає своє середнє значення за період, що дає змогу знайти тривалість одного циклу і розрахувати частоту. Для забезпечення чіткості візуалізації, побудова графіка починається з точки перетину сигналу з центральною віссю дисплея, що створює ефект синхронізації зображення.

На екрані користувач бачить робоче поле, яке поділено на дві частини: графічне представлення сигналу та текстове відображення його характеристик, таких як амплітуда, середнє значення, частота тощо. Якщо користувач хоче затримати певний кадр осцилографа, наприклад, для детального аналізу, передбачено спеціальну функцію. Програма в кожному циклі перевіряє стан кнопки, доступної через Android-додаток, і у випадку активації затримує оновлення зображення, дозволяючи зафіксувати потрібний момент для детального розгляду параметрів сигналу.

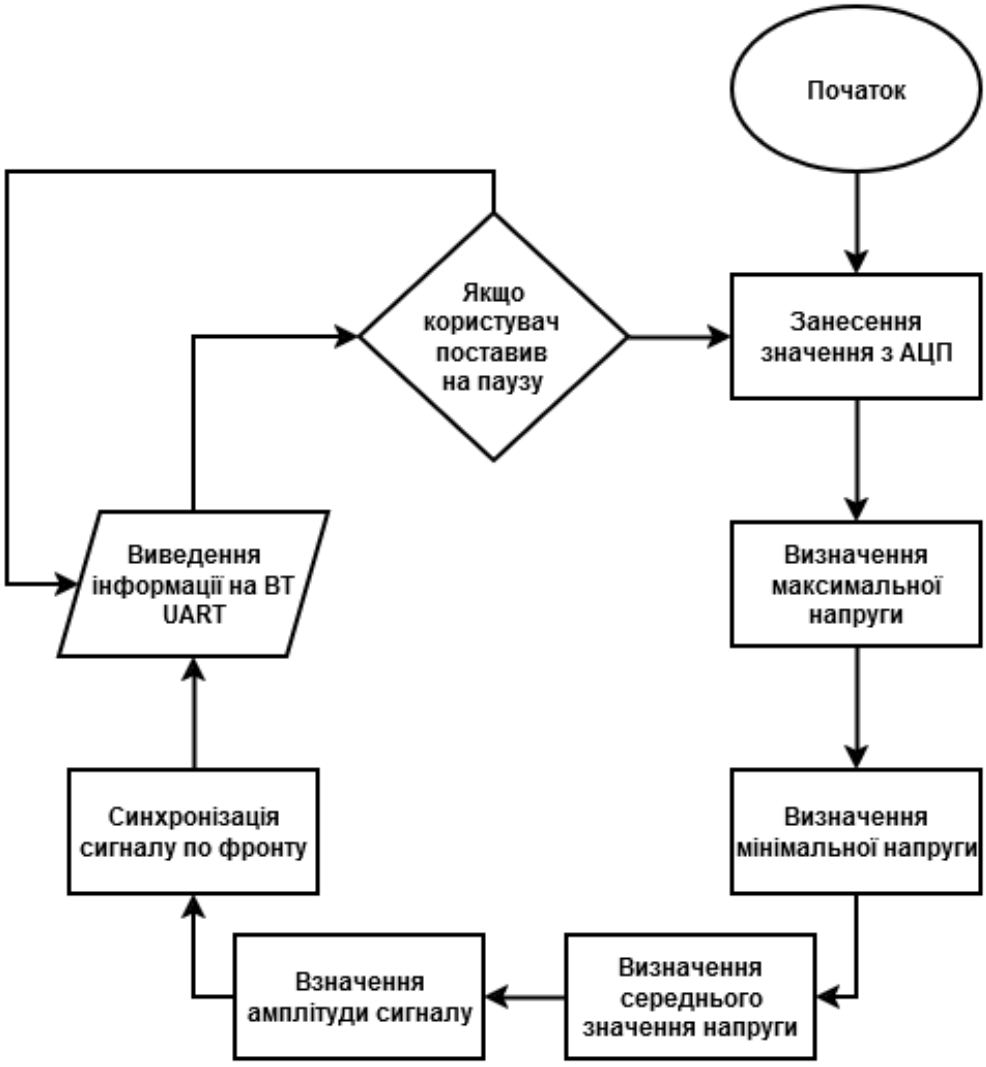

Рисунок 3.3 – Алгоритм осцилографу

На початковому етапі роботи програми виконується налаштування периферійних модулів мікроконтролера. Аналогово-цифровий перетворювач (АЦП) конфігурується для безперервного зчитування сигналів і передачі даних у спеціальний буфер за допомогою DMA, що забезпечує ефективність та зручність обробки великих обсягів даних. У той же час UART використовується для передачі результатів обчислень, відкриваючи можливість виведення даних на зовнішній пристрій або передачі на Android-додаток. Такий підхід дозволяє організувати потік інформації з мінімальними затримками.

```
#include "stm32f3xx_hal.h"

#define ADC_BUFFER_SIZE 256  // Розмір буфера для зчитування з АЦП
#define UART_BUFFER_SIZE 128 // Розмір буфера для передачі через UART

ADC_HandleTypeDef hadc1;       // Об'єкт АЦП
UART_HandleTypeDef huart2;     // Об'єкт UART
uint16_t adcBuffer[ADC_BUFFER_SIZE]; // Буфер для значень АЦП
float maxVoltage = 0.0f, minVoltage = 0.0f, amplitude = 0.0f, avgVoltage = 0.0f;
uint8_t uartBuffer[UART_BUFFER_SIZE]; // Буфер для передачі через UART
volatile uint8_t pauseState = 0;      // Статус кнопки паузи

// Ініціалізація АЦП
void ADC_Init(void) {
    // Налаштування АЦП
    hadc1.Instance = ADC1;
    hadc1.Init.Resolution = ADC_RESOLUTION_12B;
    hadc1.Init.ContinuousConvMode = ENABLE;
    HAL_ADC_Init(&hadc1);
}

// Ініціалізація UART
void UART_Init(void) {
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 9600;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
```

```
    huart2.Init.Mode = UART_MODE_TX_RX;
    HAL_UART_Init(&huart2);
}
```

Обробка отриманих даних починається з їх конвертації в значення напруги, використовуючи формули для 12-бітного АЦП. Потім здійснюється аналіз сигналу: визначаються його максимальні й мінімальні значення, середня напруга та амплітуда. Для коректного відображення графіка сигналу на осцилографі використовується синхронізація — знаходиться точка, в якій сигнал перетинає середнє значення. Також передбачено режим паузи, який реалізується через переривання від кнопки, що дозволяє зупинити оновлення даних для більш детального аналізу сигналу. У підсумку програма надає користувачеві зручний інструмент для аналізу характеристик сигналів із можливістю їх передачі на інші пристрої.

```
// Читання даних з АЦП
void Read_ADC_Data(void) {
    HAL_ADC_Start_DMA(&hadc1, (uint32_t *)adcBuffer,
ADC_BUFFER_SIZE);
}

// Обчислення характеристик сигналу
void Calculate_Signal_Params(void) {
    maxVoltage = 0.0f;
    minVoltage = 3.3f; // Для 12-бітного АЦП з напругою 3.3В
    uint32_t sumVoltage = 0;

    for (int i = 0; i < ADC_BUFFER_SIZE; i++) {
```

```
        float voltage = (adcBuffer[i] / 4095.0f) * 3.3f; // Перетворення вольтажу
        if (voltage > maxVoltage) maxVoltage = voltage;
        if (voltage < minVoltage) minVoltage = voltage;
        sumVoltage += voltage;
    }

    avgVoltage = sumVoltage / ADC_BUFFER_SIZE; // Середнє значення
    amplitude = (maxVoltage - minVoltage) / 2; // Амплітуда
}

// Синхронізація сигналу
void Synchronize_Signal(void) {
    for (int i = 1; i < ADC_BUFFER_SIZE; i++) {
        float prevVoltage = (adcBuffer[i - 1] / 4095.0f) * 3.3f;
        float currVoltage = (adcBuffer[i] / 4095.0f) * 3.3f;

        if ((prevVoltage < avgVoltage && currVoltage >= avgVoltage)) {
            // Початок синхронізації
            break;
        }
    }
}

// Виведення даних через UART
void Transmit_Data(void) {
    snprintf((char *)uartBuffer, UART_BUFFER_SIZE,
             "Max: %.2fV, Min: %.2fV, Avg: %.2fV, Amplitude: %.2fV\r\n",
             maxVoltage, minVoltage, avgVoltage, amplitude);
```

```
    HAL_UART_Transmit(&huart2, uartBuffer, strlen((char *)uartBuffer), HAL_MAX_DELAY);
}

// Головна функція
int main(void) {
    HAL_Init();
    SystemClock_Config();
    ADC_Init();
    UART_Init();

    while (1) {
        if (pauseState == 0) { // Якщо пауза не активна
            Read_ADC_Data();
            Calculate_Signal_Params();
            Synchronize_Signal();
            Transmit_Data();
        }
    }
}
```

3.3 Тестер транзисторів

Схема роботи тестера, представлена на рисунку 3.4, відображає основний алгоритм визначення параметрів транзистора. На початковому етапі алгоритм зчитує значення з усіх підключених контактів, які зберігаються у змінних. Після

цього виконується перевірка реакції транзистора на позитивний струм. Якщо перехід колектор-емітер відкривається, транзистор класифікується як NPN. У протилежному випадку, коли перехід залишається закритим, визначається, що це PNP транзистор. Після виведення типу транзистора на екран виконується ідентифікація базового контакту, і отримані дані також виводяться на дисплей. Завершивши всі ці кроки, алгоритм повторюється для наступного циклу зчитування.

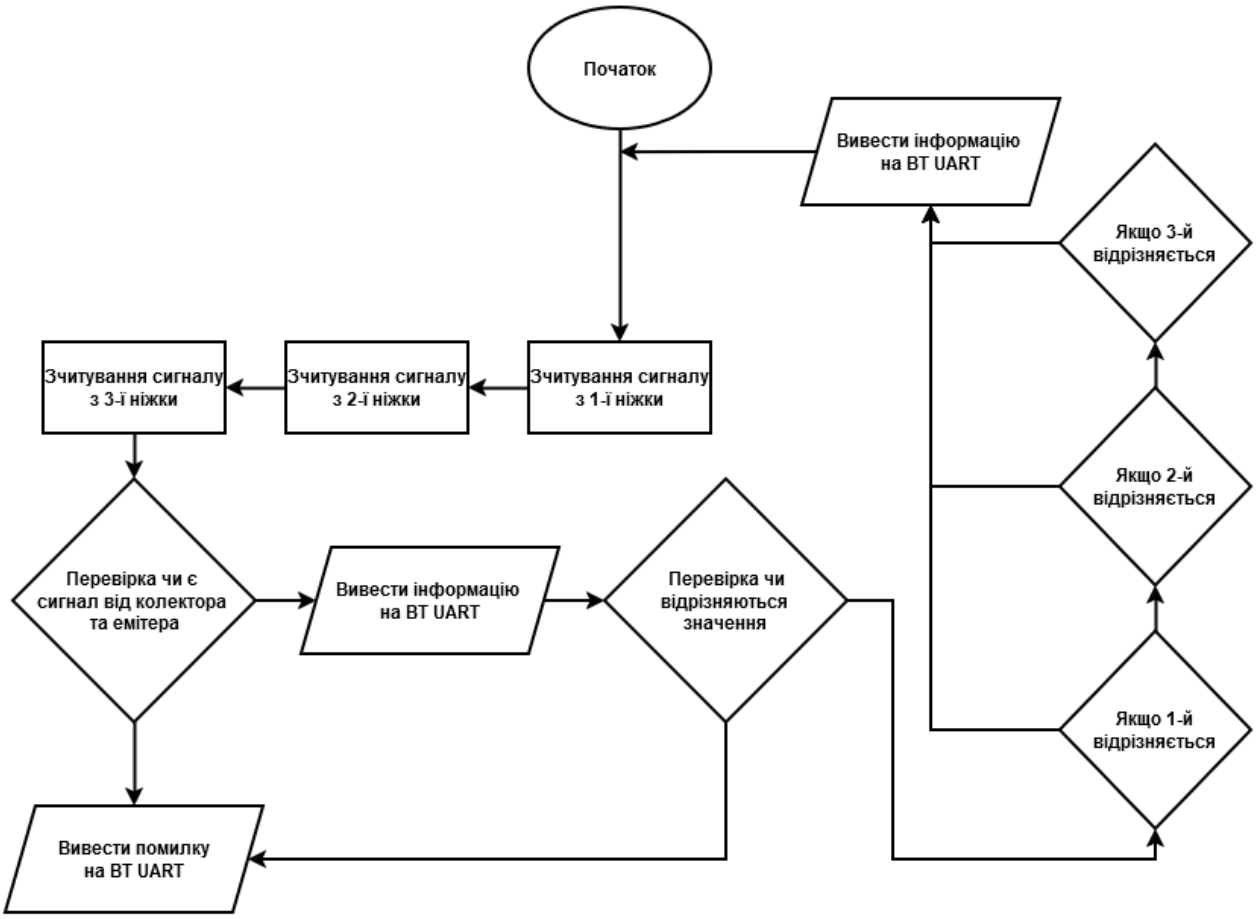

Рисунок 3.4 - Алгоритм транзисторного тестера

Після зчитування значень проводиться визначення типу біполярного транзистора. Принцип роботи транзистора полягає в тому, що він вважається відкритим, якщо на базу подається напруга, і між колектором та емітером проходить струм. Для NPN-транзистора під час тестування сума отриманих

значень змінних має дорівнювати одиниці. Однак у реальних умовах ця сума часто становить два. Це пояснюється тим, що перехід база-колектор має опір близько 500 Ом, чого достатньо для фіксації проходження струму через цей перехід. Для PNP-транзистора, який відкривається при подачі негативної напруги на базу та колектор, отримані значення будуть протилежними, і сума змінних дорівнюватиме одиниці.

У разі, якщо жодна з умов не виконується, програма генерує повідомлення про помилку, вказуючи на те, що транзистор несправний. Це дозволяє користувачеві оперативно виявляти пошкоджені компоненти і виключати їх із роботи. Такий підхід забезпечує точність визначення типу транзистора та надійність перевірки його стану, що є важливим для діагностики електронних схем.

Незалежно від типу транзистора — NPN чи PNP, значення, отримані від бази, завжди будуть протилежними значенням колектора та емітера. Це ключова особливість, яка використовується для визначення базового контакту. Алгоритм аналізує кожну пару змінних, шукаючи відмінності між ними, щоб точно встановити, який із контактів є базою.

Якщо жодна з пар змінних не відповідає очікуваним критеріям, програма генерує повідомлення про помилку. Це вказує на те, що перевірка не вдалася, і, можливо, транзистор несправний або підключений неправильно. Такий підхід дозволяє не тільки ідентифікувати базу, а й виявити потенційні проблеми з компонентом.

```
#include "main.h"
#include "string.h"

UART_HandleTypeDef huart2;

// Функція для ініціалізації UART
void UART_Init(void) {
```

```
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 9600;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    HAL_UART_Init(&huart2);
}

// Функція для зчитування значення з GPIO (імітація АЦП)
int Read_Pin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin) {
    return HAL_GPIO_ReadPin(GPIOx, GPIO_Pin);
}

// Основна програма
int main(void) {
    HAL_Init();
    SystemClock_Config();
    UART_Init();

    // Ініціалізація GPIO для пінів транзистора
    // Налаштовуємо GPIO для емітера, бази і колектора
    MX_GPIO_Init();

    char message[64];
    int base_signal, collector_signal, emitter_signal;
```

```
        while (1) {
            // Зчитуємо сигнал із бази, колектора та емітера
            base_signal = Read_Pin(GPIOB, GPIO_PIN_0);      // Приклад бази
            collector_signal = Read_Pin(GPIOB, GPIO_PIN_1); // Приклад колектора
            emitter_signal = Read_Pin(GPIOB, GPIO_PIN_2);   // Приклад емітера

            // Перевірка чи є сигнал на базі
            if (base_signal > 0) {
                // Перевірка колектора та емітера
                if (collector_signal > 0 && emitter_signal > 0) {
                    snprintf(message, sizeof(message), "Transistor is working.\n");
                } else {
                    snprintf(message, sizeof(message), "Error: No signal on emitter or collector.\n");
                }
            } else {
                snprintf(message, sizeof(message), "Error: No signal on base.\n");
            }

            // Виведення результату через UART
            HAL_UART_Transmit(&huart2, (uint8_t *)message, strlen(message), HAL_MAX_DELAY);

            // Затримка для уникнення зайвих перевірок
            HAL_Delay(500);
        }
    }
```

Розглянутий код виконує завдання тестування транзисторів, забезпечуючи ефективну взаємодію між мікроконтролером та зовнішніми пристроями. Спочатку програма ініціалізує HAL-бібліотеку через функцію HAL_Init(), що дозволяє працювати з периферійними пристроями STM32F303. Одночасно налаштовується системний годинник для забезпечення коректної роботи мікроконтролера. Для передачі результатів тестування використовується інтерфейс UART, який дозволяє виводити повідомлення на зовнішній пристрій, наприклад, комп'ютер. Це забезпечує зручний контроль та моніторинг стану транзистора.

Основна логіка програми включає зчитування сигналів із трьох пінів транзистора (база, колектор, емітер) за допомогою функції Read_Pin(). Алгоритм перевіряє, чи присутній сигнал на базі транзистора. У разі позитивного результату виконується додаткова перевірка колектора та емітера. Якщо сигнали на цих пінів також присутні, програма повідомляє про справність транзистора. У разі відсутності сигналу на будь-якому з пінів виводиться повідомлення про помилку. Щоб уникнути зайвих перевірок і забезпечити стабільну роботу, між циклами опитування транзистора додається затримка за допомогою функції HAL_Delay(500). Такий підхід дозволяє в реальному часі зчитувати та відображати результати тестування, роблячи процес зручним і простим для користувача.

3.4 Вольтметр

Мікроконтролер використовує вбудований аналогово-цифровий перетворювач (АЦП) для вимірювання напруги. За допомогою цього перетворювача відбувається зчитування аналогового сигналу, який потім конвертується в цифрове значення. Результат цього вимірювання передається через UART на зовнішній пристрій, що дозволяє вивести отриману інформацію на

дисплей або передати її на комп'ютер чи інший інтерфейс для подальшого аналізу.

Програма для вимірювання напруги складається з кількох етапів. Спочатку налаштовується АЦП для зчитування значень з аналогового сигналу, а UART використовується для передачі отриманих даних на зовнішній пристрій. Після цього, отримане значення з АЦП конвертується в напругу за допомогою 12-бітного перетворення, а результат передається через UART для виведення або подальшої обробки.

```
#include "stm32f3xx_hal.h"
#include "stdio.h"

// Ініціалізація для UART і АЦП
ADC_HandleTypeDef hadc1;
UART_HandleTypeDef huart2;

// Функція для ініціалізації периферії
void SystemClock_Config(void);
void GPIO_Init(void);
void ADC_Init(void);
void UART_Init(void);

int main(void) {
    // Ініціалізація HAL
    HAL_Init();

    // Ініціалізація системного такту
    SystemClock_Config();
```

```
// Ініціалізація GPIO, АЦП і UART
GPIO_Init();
ADC_Init();
UART_Init();

// Змінна для збереження результату АЦП
uint32_t adcValue = 0;
float voltage = 0;

// Головний цикл
while (1) {
    // Запуск АЦП
    HAL_ADC_Start(&hadc1);

    // Очікування завершення перетворення
    if (HAL_ADC_PollForConversion(&hadc1, 1000) == HAL_OK) {
        // Читання значення з АЦП
        adcValue = HAL_ADC_GetValue(&hadc1);

        // Перетворення в напругу (12-бітне АЦП, Vref = 3.3V)
        voltage = (adcValue / 4095.0) * 3.3;

        // Форматування результату для відправки через UART
        char msg[50];
        snprintf(msg, sizeof(msg), "Voltage: %.2f V\r\n", voltage);

        // Відправка результату через UART
        HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), 1000);
```

```
        }

        // Затримка перед наступним вимірюванням
        HAL_Delay(1000);
    }
}

// Функція ініціалізації UART
void UART_Init(void) {
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 115200;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart2) != HAL_OK) {
        Error_Handler();
    }
}

// Функція ініціалізації GPIO для АЦП
void GPIO_Init(void) {
    __HAL_RCC_GPIOA_CLK_ENABLE();    // Увімкнення тактування для порту A
    GPIO_InitTypeDef GPIO_InitStruct = {0};
```

```
    // Ініціалізація піну PA0 для АЦП (ADC_CHANNEL_0)
    GPIO_InitStruct.Pin = GPIO_PIN_0;
    GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
}

// Функція ініціалізації АЦП
void ADC_Init(void) {
    __HAL_RCC_ADC34_CLK_ENABLE();  // Увімкнення тактування для АЦП1
    hadc1.Instance = ADC1;
    hadc1.Init.Resolution = ADC_RESOLUTION_12B;
    hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
    hadc1.Init.ContinuousConvMode = DISABLE;
    hadc1.Init.DiscontinuousConvMode = DISABLE;
    hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    hadc1.Init.NbrOfConversion = 1;
    hadc1.Init.DMAContinuousRequests = DISABLE;
    hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;

    if (HAL_ADC_Init(&hadc1) != HAL_OK) {
        Error_Handler();
    }

    // Налаштування каналу для вимірювання на PA0
```

```
ADC_ChannelConfTypeDef sConfig = {0};
sConfig.Channel = ADC_CHANNEL_0;
sConfig.Rank = ADC_REGULAR_RANK_1;
sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
sConfig.Offset = 0;
if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK) {
   Error_Handler();
}
```

Початок коду починається з ініціалізації бібліотеки HAL, що здійснюється через виклик функції HAL_Init(). Ця функція підготовлює всю необхідну інфраструктуру для подальшої роботи з периферійними модулями мікроконтролера, такими як GPIO та АЦП. Завдяки HAL бібліотеці, весь код налаштування периферії стає значно спрощеним і уніфікованим, що дозволяє зосередитися на основній логіці програми.

На наступному етапі здійснюється ініціалізація GPIO та АЦП. За допомогою функції GPIO_Init() налаштовується пін PA0 як аналоговий вхід, через який буде зчитуватись сигнал для подальшого перетворення. Потім функція ADC_Init() конфігурує параметри роботи АЦП, зокрема визначається роздільна здатність 12 біт та вибір одного каналу для зчитування значення. Після налаштування цих параметрів, АЦП готовий до роботи і може здійснювати вимірювання напруги на вході.

Завершальним етапом є налаштування UART через функцію UART_Init(), яка конфігурує USART2 для передачі отриманих даних на зовнішні пристрої, такі як комп'ютер або мобільний телефон. В основному циклі програма періодично зчитує значення з АЦП, перетворює його у відповідну напругу за формулою Voltage = (ADC value / 4095) * 3.3, і передає результат через UART кожну секунду. Цей код дозволяє реалізувати основну функцію вольтметра, яка може бути адаптована для більш складних задач, таких як калібрування або вимірювання з використанням

зовнішніх джерел напруги для підвищення точності.

3.5 Амперметр

Для створення амперметра на мікроконтролері STM32F303 із використанням датчика струму ACS724, ми використовуємо аналоговий вихід цього датчика для вимірювання струму. Датчик ACS724 генерує напругу, яка пропорційна вимірюваному струму. При цьому вихідна напруга при нульовому струмі становить 2.5V, а зміна напруги на виході прямо пропорційна струму, що протікає через датчик. Це дозволяє точно вимірювати струм у широкому діапазоні, використовуючи просту формулу для перерахунку напруги в струм.

Програма для цього амперметра складається з кількох основних етапів. Спочатку налаштовується периферія, де АЦП конфігуровано для зчитування аналогового сигналу з датчика ACS724, а UART — для передачі результатів на зовнішні пристрої, такі як комп'ютери чи дисплеї. Далі зчитується значення з АЦП, яке перетворюється в напругу (від 0 до 3.3V), і на основі цієї напруги обчислюється значення струму, використовуючи характеристики датчика ACS724. Отримане значення струму передається через UART, що дозволяє відобразити його на підключених пристроях.

```
#include "stm32f3xx_hal.h"
#include "stdio.h"

// Налаштування UART
UART_HandleTypeDef huart2;
```

```
// Налаштування АЦП
ADC_HandleTypeDef hadc1;

// Буфер для зчитування значення з АЦП
uint32_t adcValue = 0;

// Напруга з АЦП та константи для перерахунку в струм
float voltage = 0;
float current = 0;
const float V_REF = 3.3f;          // Опорна напруга (3.3V)
const float ADC_RESOLUTION = 4095.0f; // Роздільна здатність АЦП (12 біт)
const float OFFSET_VOLTAGE = 2.5f;    // Напруга на виході при нульовому струмі (2.5V)
const float SENSITIVITY = 0.185f;    // Чутливість ACS724 (мВ на ампер)

// Ініціалізація АЦП
void ADC_Init(void)
{
   __HAL_RCC_ADC34_CLK_ENABLE();  // Включення тактування АЦП
   hadc1.Instance = ADC1;
   hadc1.Init.Resolution = ADC_RESOLUTION_12B;
   hadc1.Init.ScanConvMode = DISABLE;
   hadc1.Init.ContinuousConvMode = ENABLE;
   hadc1.Init.DiscontinuousConvMode = DISABLE;
   hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
   hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
   HAL_ADC_Init(&hadc1);
```

```
}

// Ініціалізація UART
void UART_Init(void)
{
  __HAL_RCC_USART2_CLK_ENABLE();      // Включення тактування USART2
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  HAL_UART_Init(&huart2);
}

// Зчитування значення з АЦП
float Read_ADC(void)
{
  // Запускаємо перетворення АЦП
  HAL_ADC_Start(&hadc1);
  if (HAL_ADC_PollForConversion(&hadc1, 1000) == HAL_OK) {
    adcValue = HAL_ADC_GetValue(&hadc1);
  }
  HAL_ADC_Stop(&hadc1);

  // Перетворюємо значення в напругу
  voltage = (adcValue / ADC_RESOLUTION) * V_REF;
```

```
    return voltage;
}

// Перерахунок напруги в струм
float Calculate_Current(float voltage)
{
    // Вираховуємо струм за допомогою характеристик датчика ACS724
    float voltage_difference = voltage - OFFSET_VOLTAGE;
    current = (voltage_difference / SENSITIVITY);
    return current;
}

// Головна програма
int main(void)
{
    HAL_Init();
    ADC_Init();
    UART_Init();

    char uart_buffer[50];
    float voltage, current;

    while (1)
    {
        // Зчитуємо значення з АЦП
        voltage = Read_ADC();
```

```
            // Обчислюємо струм
            current = Calculate_Current(voltage);

            // Формуємо рядок для відправки через UART
            sprintf(uart_buffer, "Current: %.3f A\n", current);

            // Виводимо результат через UART
            HAL_UART_Transmit(&huart2, (uint8_t*)uart_buffer, strlen(uart_buffer),
1000);

            HAL_Delay(1000); // Затримка 1 секунда
        }
    }
```

Програма для вимірювання струму з використанням датчика ACS724 на мікроконтролері STM32F303 починається з ініціалізації периферії. Для цього використовуються функції ініціалізації ADC_Init() та UART_Init(). Перша з них налаштовує АЦП для зчитування аналогових сигналів, а друга налаштовує UART для передачі результатів на зовнішні пристрої, наприклад, комп'ютер чи мобільний телефон. Це забезпечує необхідну інфраструктуру для вимірювання та передачі результатів в реальному часі.

Подальша обробка даних включає зчитування значень з АЦП за допомогою функції Read_ADC(). Після цього отримане значення перетворюється у відповідну напругу, враховуючи параметри АЦП, такі як роздільна здатність і опорна напруга. Для обчислення значення струму з цієї напруги використовується функція Calculate_Current(), яка враховує чутливість датчика ACS724 (0.185 В/А) і зсув напруги (2.5 В при нульовому струмі). Це дозволяє точно визначити струм, що проходить через датчик.

Останнім кроком є передача результату через UART на зовнішній пристрій.

Функція HAL_UART_Transmit() формує отримане значення у вигляді рядка та передає його для подальшого відображення на підключеному пристрої. Основний цикл програми безперервно повторює процес: зчитування значень з АЦП, обчислення струму та передача результатів через UART. Така організація забезпечує реальний моніторинг струму в режимі реального часу за допомогою простого амперметра.

3.6 Омметр

Для реалізації омметра, використовуємо принцип вимірювання опору через створення схемного підключення. Один зі способів — це вимірювання напруги на резисторі за допомогою АЦП. Знаючи опір, ми можемо обчислити величину іншого резистора за допомогою закону Ома.

Одну з точок можна підключити до джерела напруги (наприклад, 3.3V). Іншу точку підключаємо до резистора (який хочемо виміряти) та через виведений пін АЦП на землю. Для вимірювання потрібно знати опір резистора, через який проходить струм, та напругу, що вимірюється на АЦП.

```
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_USART2_UART_Init();
```

```
  while (1)
  {
    uint32_t adc_value = Read_ADC();
    float resistance = Calculate_Resistance(adc_value);

    // Виведення результату через UART
    char buffer[50];
    sprintf(buffer, "Resistance: %.2f Ohm\r\n", resistance);
    HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer),
HAL_MAX_DELAY);
    HAL_Delay(1000); // Затримка для оновлення
  }
}

uint32_t Read_ADC(void)
{
  HAL_ADC_Start(&hadc1);
  HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
  return HAL_ADC_GetValue(&hadc1);
}

float Calculate_Resistance(uint32_t adc_value)
{
  float voltage = (float)adc_value * 3.3f / 4095.0f; // вираховуєм напругу АЦП
  float resistance = (3.3f - voltage) / voltage * 1000.0f; // Резистор 1 кОм
  return resistance;
}
```

```
void SystemClock_Config(void)
{
  // Налаштування системного такту
}

static void MX_GPIO_Init(void)
{
  // Ініціалізація GPIO для UART
}

static void MX_ADC1_Init(void)
{
  // Ініціалізація АЦП для зчитування значення
  __HAL_RCC_ADC12_CLK_ENABLE();
  hadc1.Instance = ADC1;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  HAL_ADC_Init(&hadc1);
}
```

У програмі для вимірювання опору використовуються різні компоненти мікроконтролера STM32F303. Для вимірювання напруги на резисторі застосовується АЦП1 (ADC1). Зчитування даних з цього АЦП виконується через функцію Read_ADC, яка забезпечує отримання значень аналогового сигналу. Після

отримання значення напруги на резисторі, програма обчислює опір, використовуючи формулу закону Ома, в межах функції Calculate_Resistance.

Отримане значення опору виводиться на зовнішній пристрій через UART. Для цього застосовується функція HAL_UART_Transmit, яка передає результат у вигляді рядка. Це дає змогу переглядати результати вимірювань на дисплеї або на іншому підключеному пристрої, забезпечуючи зручний спосіб моніторингу параметрів вимірюваного елементу.

3.7 Тестер на коротке замикання «Прозвонка»

Для реалізації прозвонки використовується метод, який дозволяє перевірити наявність з'єднання між двома точками за допомогою мультиметра або мікроконтролера. Якщо між цими точками є з'єднання, це свідчить про те, що сигнал проходить, що дає можливість отримати відповідний результат, наприклад, включити світлодіод або вивести повідомлення через UART. Такий підхід дає змогу ефективно перевіряти наявність електричного з'єднання в схемах.

У схемі прозвонки підключають два піні GPIO: один використовується для виведення сигналу, а інший — для перевірки наявності з'єднання. Якщо між цими піном є з'єднання, наприклад, через провід або інші компоненти, мікроконтролер зчитує сигнал і на основі цього здійснює відповідний вивід – запускає п’єзоелемент

```
#include "stm32f3xx_hal.h"

#define PIEZO_PIN GPIO_PIN_5 // Пін для підключення п'єзоелемента
#define PIEZO_GPIO_PORT GPIOA // Порт для п'єзоелемента
```

```
#define TEST_PIN GPIO_PIN_0 // Пін для виведення сигналу для перевірки
#define TEST_GPIO_PORT GPIOB // Порт для піну перевірки

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
void buzzer_on(void);
void buzzer_off(void);

int main(void)
{
  // Ініціалізація HAL
  HAL_Init();

  // Налаштування системного годинника
  SystemClock_Config();

  // Ініціалізація GPIO
  MX_GPIO_Init();

  while (1)
  {
    // Перевірка з'єднання між двома точками
    if (HAL_GPIO_ReadPin(TEST_GPIO_PORT, TEST_PIN) ==
GPIO_PIN_SET)
    {
      // Якщо з'єднання є, виводимо звуковий сигнал через п'єзоелемент
      buzzer_on();
    }
```

```
    else
    {
      // Якщо з'єднання відсутнє, вимикаємо п'єзоелемент
      buzzer_off();
    }

    // Затримка для уникнення постійного сигналу
    HAL_Delay(200);
  }
}

// Функція для увімкнення п'єзоелемента (виведення звукового сигналу)
void buzzer_on(void)
{
  HAL_GPIO_WritePin(PIEZO_GPIO_PORT, PIEZO_PIN, GPIO_PIN_SET);
}

// Функція для вимкнення п'єзоелемента
void buzzer_off(void)
{
  HAL_GPIO_WritePin(PIEZO_GPIO_PORT,                    PIEZO_PIN,
GPIO_PIN_RESET);
}

// Функція ініціалізації GPIO
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
```

```
  // Включення тактування для GPIOA та GPIOB
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  // Налаштування п'єзоелемента (як вихід)
  GPIO_InitStruct.Pin = PIEZO_PIN;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(PIEZO_GPIO_PORT, &GPIO_InitStruct);

  // Налаштування піну для перевірки з'єднання (як вхід)
  GPIO_InitStruct.Pin = TEST_PIN;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(TEST_GPIO_PORT, &GPIO_InitStruct);
}
```

Програма починається з ініціалізації бібліотеки HAL за допомогою функції HAL_Init(), що дозволяє працювати з периферією мікроконтролера. Окрім того, для налаштування системного годинника викликається функція SystemClock_Config(), яка може бути змінена або пропущена в залежності від конкретних вимог системи. Після цього програма налаштовує GPIO: пін для п'єзоелемента (PIEZO_PIN) конфігурується як вихід, щоб можна було подавати сигнал для генерування звукового сигналу, а пін для перевірки з'єднання (TEST_PIN) — як вхід, через який буде перевірятися наявність сигналу для визначення стану з'єднання.

У головному циклі програма постійно перевіряє з'єднання між двома точками, зчитуючи значення з піну перевірки. Якщо з'єднання є (пін знаходиться в логічній одиниці), функція buzzer_on() активує п'єзоелемент для відтворення звукового сигналу. Якщо з'єднання відсутнє (пін знаходиться в логічному нулі), функція buzzer_off() вимикає п'єзоелемент. Для уникнення постійного шуму і забезпечення стабільної роботи програми застосовується затримка HAL_Delay(200), що дає час між перевірками.

РОЗДІЛ 4 АЛГОРИТМ РОБОТИ ПРОГРАМИ ANDROID ЗАСТОСУНКУ

Для створення Android програми, яка буде здійснювати зв'язок з мікроконтролером STM32F303 через Bluetooth модуль HC-08, необхідно розробити додаток, що дозволить підключатися до Bluetooth, отримувати дані від мікроконтролера та відображати ці дані на екрані пристрою. Основною метою такого додатка є забезпечення зручного інтерфейсу для взаємодії з мікроконтролером, наприклад, для перегляду виміряних значень напруги чи струму в реальному часі.

Основні етапи реалізації програми включають налаштування з'єднання з Bluetooth модулем HC-08, отримання даних через Bluetooth від мікроконтролера STM32F303 та відображення результатів на екрані Android пристрою. Це дозволяє користувачеві отримувати інформацію про параметри електричних характеристик, таких як напруга або струм, що вимірюються мікроконтролером, у зручному для аналізу вигляді.

```
package com.example.bluetoothstm32;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.widget.TextView;
import android.widget.Toast;
```

```
import androidx.appcompat.app.AppCompatActivity;

import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;
import java.util.UUID;

public class MainActivity extends AppCompatActivity {

    private BluetoothAdapter mBluetoothAdapter;
    private BluetoothSocket mSocket;
    private BluetoothDevice mDevice;
    private InputStream mInputStream;
    private OutputStream mOutputStream;

    private TextView dataTextView;
    private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    private static final String DEVICE_ADDRESS = "00:14:03:07:1A:B1"; // MAC адрес HC-08

    private Handler mHandler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        dataTextView = findViewById(R.id.dataTextView);

        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if (mBluetoothAdapter == null) {
            Toast.makeText(this, "Bluetooth не підтримується",
Toast.LENGTH_SHORT).show();
            finish();
        }

        if (!mBluetoothAdapter.isEnabled()) {
            mBluetoothAdapter.enable();
        }

        mHandler = new Handler(new Handler.Callback() {
            @Override
            public boolean handleMessage(Message msg) {
                String data = (String) msg.obj;
                dataTextView.setText(data);  // Вивести отримані дані на екран
                return true;
            }
        });

        connectToBluetooth();
    }

    private void connectToBluetooth() {
        mDevice = mBluetoothAdapter.getRemoteDevice(DEVICE_ADDRESS);
        try {
```

```
            mSocket = mDevice.createRfcommSocketToServiceRecord(MY_UUID);
            mSocket.connect();
            mInputStream = mSocket.getInputStream();
            mOutputStream = mSocket.getOutputStream();

            // Почати читання даних
            beginListenForData();

        } catch (IOException e) {
            e.printStackTrace();
            Toast.makeText(this, "Не вдалося підключитися до пристрою",
Toast.LENGTH_SHORT).show();
        }
    }

    private void beginListenForData() {
        Thread workerThread = new Thread(new Runnable() {
            @Override
            public void run() {
                byte[] buffer = new byte[1024];
                int bytes;

                while (true) {
                    try {
                        bytes = mInputStream.read(buffer);
                        String readMessage = new String(buffer, 0, bytes);
                        mHandler.obtainMessage(0, readMessage).sendToTarget();
                    } catch (IOException e) {
```

```
                    e.printStackTrace();
                    break;
                }
            }
        }
    });
    workerThread.start();
}

@Override
protected void onDestroy() {
    super.onDestroy();
    try {
        mSocket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Та перед цим у AndroidManifest.xml треба додати дозвіл на використання Bluetooth в смартфоні

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission    android:name="android.permission.BLUETOOTH_ADMIN"
/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Програма для Android, що взаємодіє з Bluetooth модулем HC-08, починається з ініціалізації Bluetooth адаптера. Для цього використовується клас BluetoothAdapter, який перевіряє, чи підтримує пристрій Bluetooth. Якщо Bluetooth увімкнений, програма намагається підключитися до модуля HC-08. Якщо він вимкнений, програма спробує його увімкнути для подальшого з'єднання.

Після того, як Bluetooth активовано, програма встановлює з'єднання з модулем HC-08 за допомогою його MAC-адреси. Для цього використовується стандартний UUID для RFCOMM з'єднання. Це дозволяє забезпечити стабільне і безпечне з'єднання між Android пристроєм і мікроконтролером STM32F303 через Bluetooth.

Після успішного з'єднання програма починає отримувати дані від мікроконтролера через Bluetooth. Для цього використовується InputStream, через який програма зчитує дані з Bluetooth модуля. Кожного разу, коли нові дані надходять, вони передаються на головний потік через Handler, що дозволяє оновлювати інформацію в реальному часі на екрані пристрою.

Отримані дані, такі як значення напруги або струму, відображаються в елементі TextView на екрані Android пристрою. Це дозволяє користувачеві візуально відстежувати значення, що вимірюються мікроконтролером, зручним і інтуїтивно зрозумілим способом. Важливою частиною цього процесу є налаштування правильного UUID для HC-08 та вказівка правильної MAC-адреси Bluetooth модуля, а також наявність необхідних дозволів на використання Bluetooth в Android.

В результаті, дана програма дозволяє реалізувати систему для отримання і відображення даних від мікроконтролера STM32F303 через Bluetooth, що може бути використано для моніторингу різних параметрів, таких як напруга, струм, чи перевірка з'єднань в реальному часі.

РОЗДІЛ 5 ПЕРЕВІРКА ТА ДІАГНОСТИКА

5.1 Тестування роботи LC тестеру

Зовнішній вигляд користувацького інтерфейсу застосунку, в режимі виміру індуктивності, зображено на рис. 5.1 та рис. 5.2, в режимах тестування індуктивності та ємності відповідно.

Рисунок 5.1 – Зовнішній вигляд застосунку при тестуванні котушки

Рисунок 5.2 – Зовнішній вигляд застосунку при тестуванні ємності

Результати проведеного тестування показали, що робота тестера, побудованого на основі методу вимірювання частоти резонансу LC-контуру, суттєво залежить від впливу зовнішніх факторів. Зокрема, однією з головних причин відхилень у вимірюваннях є вплив навколишнього середовища, який змінює параметри електричних компонентів, таких як індуктивність котушки. Наприклад, наявність у безпосередній близькості дротів або інших провідників може змінювати магнітне поле, створюване котушкою, що призводить до зміни її індуктивності та, відповідно, до похибок у визначенні частоти резонансу.

Таким чином, точність роботи тестера залежить не лише від його внутрішніх компонентів, але й від правильності організації вимірювального процесу. Для отримання більш точних результатів важливо звести до мінімуму вплив зовнішніх факторів. Це може включати ізоляцію тестера від зайвих провідників, правильну прокладку дротів і дотримання рекомендацій щодо розміщення елементів LC-

контуру. Такі заходи дозволять значно підвищити надійність і точність отриманих вимірювань.

4.2 Тестування осцилографу

Зовнішній вигляд користувацького інтерфейсу застосунку, в режимі виміру індуктивності Зображено на рис. 5.3.

Рисунок 5.3 - Результат роботи програми осцилографу

Результати тестування програми підтвердили її функціональність у взаємодії з Bluetooth-модулями та можливість коректного зчитування даних із мікроконтролера. На зображенні показано графік осцилограми, який відображає отримані дані. Частота сигналу становить 10 Гц, амплітуда — 2 В, а розподіл

сигналу представлено відповідно до налаштувань часової та амплітудної шкал. Це свідчить про те, що передача даних через UART та їх відображення у вигляді графіка працює коректно.

4.3 Тестування тестеру транзисторів

Якщо підключити транзистор, на екрані автоматично з’явиться інформація. Як приклад взято N-P-N біполярний транзистор C945 в корпусі TO-92. У більшості схожих транзисторів, ніжка бази знаходиться посередині, але у цього транзистора вона знаходиться на 3-му контакті, що можна побачити на на рис. 5.4.

Рисунок 5.4 - Результат тестування транзистору

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено функціональний мультиметр на базі мікроконтролера STM32 із мобільним управлінням, що відповідає сучасним вимогам до вимірювальних приладів. У процесі роботи були виконані наступні завдання:

Проведено аналіз існуючих мультиметрів, визначено їхні переваги та недоліки, що дозволило обґрунтувати вибір мікроконтролера STM32 як основи для розробки пристрою.

Розроблено апаратну частину мультиметра, яка забезпечує можливість вимірювання напруги, струму, опору, ємності конденсаторів, а також підтримує підключення через бездротовий інтерфейс.

Створено програмне забезпечення з використанням бібліотеки HAL для керування вимірювальним процесом, обробки даних та передачі результатів на мобільний пристрій.

Реалізовано мобільний додаток, який забезпечує зручний інтерфейс для керування мультиметром, моніторингу показників у реальному часі та збереження даних для подальшого аналізу.

Проведено тестування пристрою, яке підтвердило його точність, стабільність роботи та відповідність заданим технічним вимогам.

Результати роботи демонструють, що запропонований підхід до інтеграції мобільного управління та використання сучасних мікроконтролерів значно розширює функціональність і зручність використання мультиметра. Створений пристрій має потенціал для подальшого вдосконалення, зокрема інтеграції з хмарними сервісами, підвищення точності вимірювань і розширення спектра вимірюваних параметрів.

Розроблений мультиметр може бути застосований у різних сферах, включаючи побутове використання, освітні лабораторії, наукові дослідження та промислову діагностику. Робота демонструє ефективність використання STM32 для розробки складних електронних пристроїв та підкреслює перспективність мобільного управління у сучасних вимірювальних приладах.

Таким чином, поставлені завдання були успішно виконані, а розроблений мультиметр є внеском у розвиток сучасних вимірювальних технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. EEVblog #900 – STM32 ARM Development Boar. URL: https://www.eevblog.com/ (дата звернення 20.11.2024).
2. Learn Electronics, LearnElectronicsOnline.com, Easily Master The Basics Of Electronics Theory And Practice. URL: https://www.learnelectronicsonline.com/ (дата звернення 20.11.2024).
3. Tindie, STM32 Blue Pill LoRaWAN node, LoRaWAN node with STM32 Blue Pill, LoRa. URL: https://www.tindie.com/ (дата звернення 20.11.2024).
4. CircuitDigest, Getting Started with STM32 Nucleo64 using STM32CubeMX and TrueSTUDIO - Simple LED Control. URL: https://circuitdigest.com/ (дата звернення 20.11.2024).
5. CONTROLLERSTECH, How to calculate ADC Conversion Time. URL: https://controllerstech.com/ (дата звернення 20.11.2024).
6. PlatformIO for STM, STM32Cube HAL and Nucleo-F401RE: debugging and unit testing. URL: https://docs.platformio.org/en/latest/platforms/ststm32.html (дата звернення 20.11.2024).
7. Maker Pro, STMicroelectronics' ST730MR LDO Voltage Regulator for Medium Voltage Applications. URL: https://maker.pro/ (дата звернення 20.11.2024).
8. Udemy, ARM Assembly Language From Ground Up™ 1. URL: https://www.udemy.com/ (дата звернення 20.11.2024).
9. HACKADAY, STM32 Offers Performance Gains For DIY Oscilloscope. URL: https://hackaday.com/ (дата звернення 20.11.2024).
10. Microcontrollers Lab, HC-05 Bluetooth Module with STM32 Nucleo using STM32CubeIDE. URL: https://microcontrollerslab.com/ (дата звернення 20.11.2024).
11. Elecrtonic-lab.com, IC Mcu STMicro STM32C071: A Low-Power Microcontroller with Arm Cortex-M0+ MCU, 2x USART, 9 Timers, and 12-bit ADC for

Consumer and Industrial Applications. URL: https://www.electronics-lab.com/ (дата звернення 20.11.2024).

12. Embedded Wizard, Embedded Wizard Training. URL: https://www.embedded-wizard.de/services/gui-development (дата звернення 20.11.2024).

13. GitHub, MartinD-CZ, STM32F1-open-source-multimeter URL: https://github.com/MartinD-CZ/STM32F1-open-source-multimeter (дата звернення 20.11.2024).

14. DeepBlueMbedded, STM32 Buzzer | Piezo Buzzer Example + Tone [Active & Passive]. URL: https://deepbluembedded.com/ (дата звернення 20.11.2024).

15. GitHub, MaJerle, stm32-usart-uart-dma-rx-tx. URL: https://github.com/topics/stm32 (дата звернення 20.11.2024).

16. Vorobiov L.Y., Shevchenko O.Y., Kuzmin O.V., Romanyuk O.N., Antonenko A.V., Avramchuk R.S. et al. “DESIGN OF PROTECTED SERVER ROOMS BASED ON CISCO EQUIPMENT IN CLASS A OFFICE BUILDINGS”. THE LEVEL OF DEVELOPMENT OF SCIENCE AND TECHNOLOGY IN THE XXI CENTURY. Monographic series «European Science»Book 32. Part 2. Karlsruhe 2024. P. 106-132. URL: https://desymp.promonograph.org/index.php/sge/issue/view/sge32-02/sge32-02

17. Аврамчук Р.С., Застосування роботів у виготовленні мікроелектронних компонентів «Проблеми комп’ютерної інженерії» Всеукраїнська V науково-практична конференція. м.Київ, 03 грудня 2024 р. Подано до друку.

18. Аврамчук Р.С., Огляд рішення застосування потужних процесорів на бпла «Проблеми комп’ютерної інженерії» Всеукраїнська V науково-практична конференція. м.Київ, 03 грудня 2024 р. Подано до друку.

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістра

НА ТЕМУ: «МУЛЬТИМЕТР НА БАЗІ STM32 З МОБІЛЬНИМ УПРАВЛІННЯМ»

Виконав студент групи КСДМ-62
Аврамчук Роман Сергійович
Керівник дипломної роботи:
Коротков Сергій Станіславович,
phD (доктор філософії), доцент

Київ – 2025

- **Актуальність:** У зв'язку з розвитком сучасних технологій відбувається збільшенням кількості інженерів та спеціалістів в області комп’ютерної, електротехнічних та радіотехнічних наук. Безпосередньо виникають потреби в домашніх вимірюваннях характеристик радіодеталей та можливостях віддаленого контролю з подальшими можливостями обробки отриманих значень. Тому існує потреба у розробці багатофункціонального мультиметра, який може бути синхронізований із зовнішніми пристроями, такими як смартфон.
- **Предмет:** програмне та апаратне забезпечення мультиметра та Android пристрою, зокрема схеми підключення, макет та інтерфейсу.
- **Мета:** Розробка багатофункціонального мультиметра на базі мікроконтролера “STM32”

Об'єкт: Розробка багатофункціонального мультиметра на базі мікроконтролера «**STM32**» з можливістю синхронізації з пристроями на операційній системі Android.

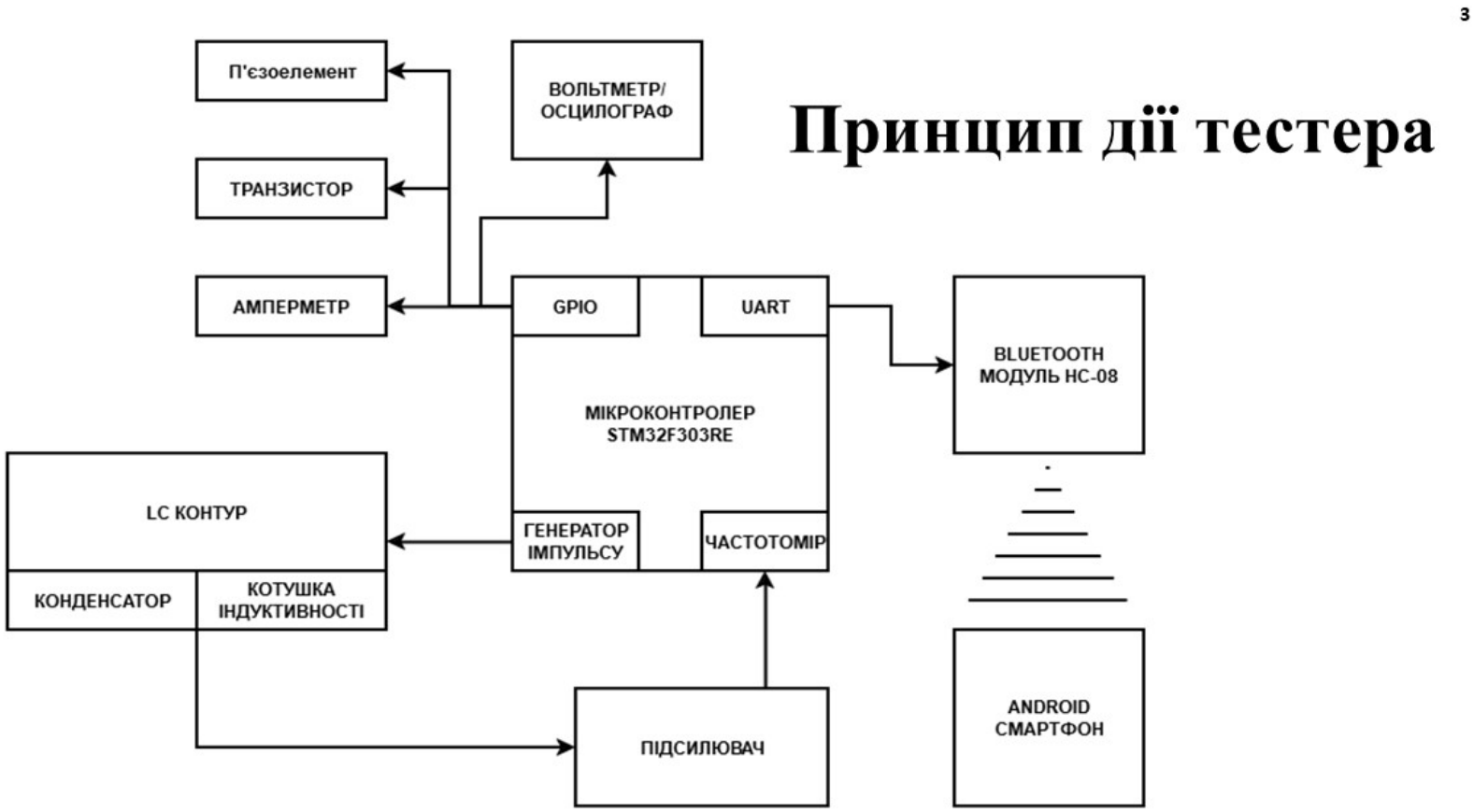

Рис. 1.2 – Структурна схема проєкту

4

БАЗОВІ КОМПОНЕНТИ ПРОЄКТУ

STM32f303RE

Рис. 2.1 – Мікроконтролер STM32F303RE в корпусі LQFP64

Платформа ST Nucleo

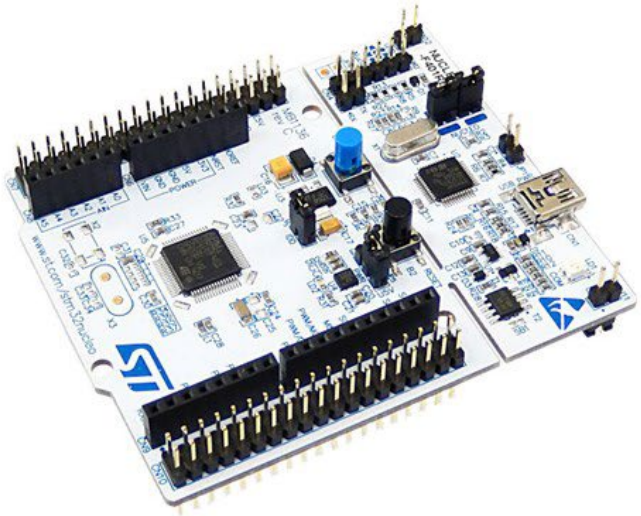

Рис. 2.2 – Зовнішній вигляд відладочної плати Nucleo-64

КОМПОНЕНТИ ПРОЄКТУ

ОП OPA2134PA

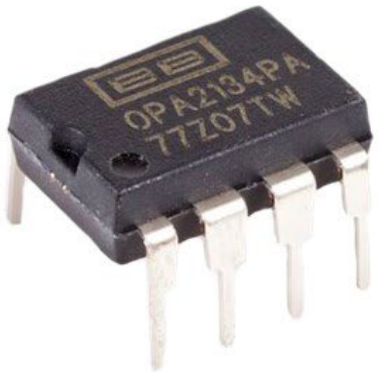

Рис. 2.4 – Вигляд OPA2134PA з двома ОП в корпусі DIP8

Bluetooth модуль HC-08

Рис. 2.3 – Bluetooth модуль HC-08

КОМПОНЕНТИ ПРОЄКТУ

Датчик ACS724

Рис. 2.5 - Датчик струму ACS724

Кварцовий резонатор

Рис. 2.7 – Кварцовий резонатор на 32 МГц

Програмні компоненти

Програмне середовище

- Розробка програмного коду відбувались в програмних середовищах STM32CubeIDE на мові програмування подібній мові C, та Android Studio на мові програмування Java.

Бібліотеки

- При написанні програмного забезпечення, використовувались бібліотеки що допомагає створювати програми шляхом реактивного програмування та спеціальні бібліотеки Android та Java. Ці бібліотеки забезпечують коректну роботу програмного забезпечення.

Принцип роботи LC-контуру

$$f = \frac{1}{2t} \quad (1.1)$$

$$f = \frac{1}{2\pi\sqrt{(LC)}} \quad (1.2)$$

$$L = \frac{1}{4\pi^2 f^2 C} \quad (1.3)$$

$$C = \frac{1}{4\pi^2 f^2 L} \quad (1.4)$$

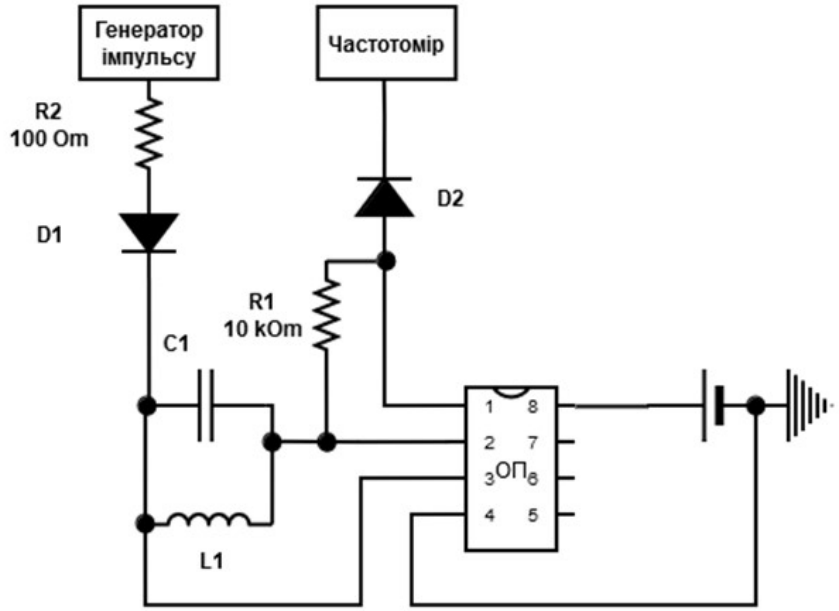

Рис. 3.1 – Принципова схема блоку LC тестера

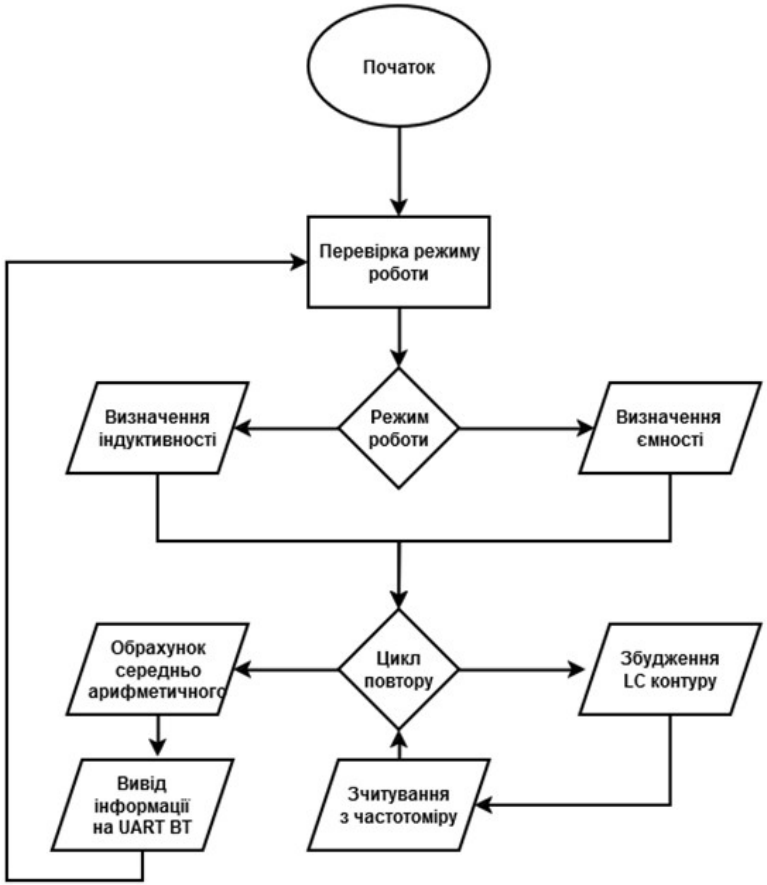

Рис. 3.2 – Алгоритм LC тестеру

9

Алгоритм програми LC-контуру

Алгоритм програми осцилографу

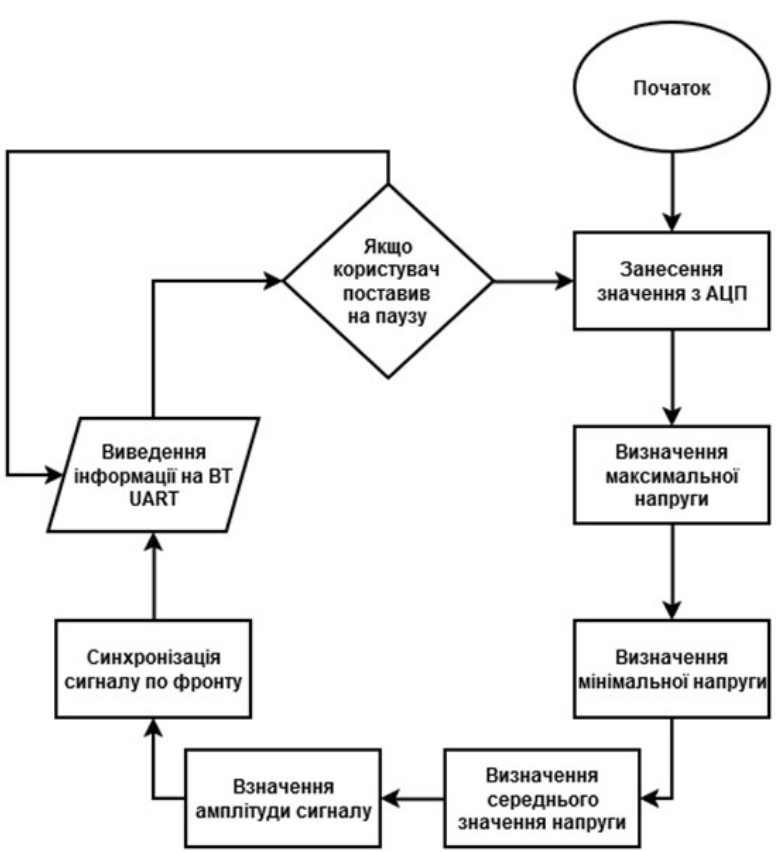

Рис. 3.3 – Алгоритм осцилографу

11

Алгоритм тестеру транзисторів

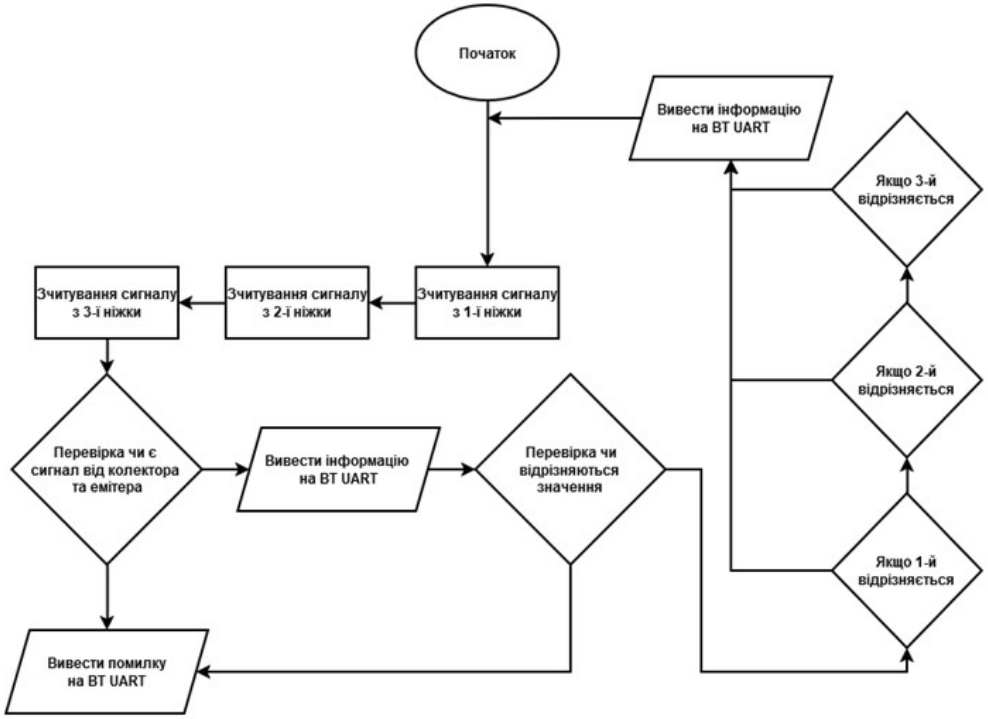

Рис. 3.4 - Алгоритм транзисторного тестера

Перевірка роботи тестеру

Рис. 5.1 – Зовнішній вигляд застосунку при тестуванні котушки

Рис. 5.2 – Зовнішній вигляд застосунку при тестуванні ємності

13

Перевірка роботи тестеру

Рис. 5.3 - Результат роботи програми осцилографу

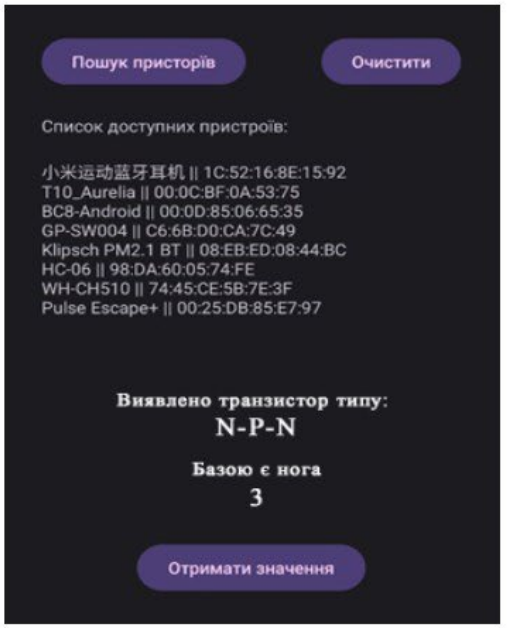

Рис. 5.4 - Результат тестування транзистору

Порівняння з подібними мультитестерами

UNI-T UT890D

QUARK

Висновки

- Було побудовано робочий макет індуктивно-ємнісного тестера, що працює в діапазонах зі 100 мкГн до 15 мГн для котушок індуктивності та з 0.1uF до 10uF для конденсаторів. Також була реалізована функція передачі обробленої інформації через Bluetooth на Android пристрій.
- Функціонал тестеру підтверджує, що мультиметр може успішно використовуватися в різних сферах, включаючи електроніку та електротехніку, для швидкої перевірки працездатності та значень радіодеталей.
- Цей проєкт має перспективу розвитку доповнивши реалізацією функції заміру потужності, тиску, освітленості. Також прилад можна вдосконалити, покращенням функціоналу Android програми, реалізувавши подальшу обробку отриманої інформації та покращення його точності та швидкості реакції.

Публікації та апробація роботи

- 1. Vorobiov L.Y., Shevchenko O.Y., Kuzmin O.V., Romanyuk O.N., Antonenko A.V., Avramchuk R.S. et al. “DESIGN OF PROTECTED SERVER ROOMS BASED ON CISCO EQUIPMENT IN CLASS A OFFICE BUILDINGS”. THE LEVEL OF DEVELOPMENT OF SCIENCE AND TECHNOLOGY IN THE XXI CENTURY. Monographic series «European Science»Book 32. Part 2. Karlsruhe 2024. P. 106-132. URL: https://desymp.promonograph.org/index.php/sge/issue/view/sge32-02/sge32-02
- 2. Аврамчук Р.С., Застосування роботів у виготовленні мікроелектронних компонентів «Проблеми комп’ютерної інженерії» Всеукраїнська V науково-практична конференція. м.Київ, 03 грудня 2024 р. Подано до друку.
- 3. Аврамчук Р.С., Огляд рішення застосування потужних процесорів на бпла «Проблеми комп’ютерної інженерії» Всеукраїнська V науково-практична конференція. м.Київ, 03 грудня 2024 р. Подано до друку.