МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
Циклова комісія *комп'ютерних систем та мереж*
(повна назва циклової комісії)

Допустити до захисту
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
(підпис) *Ірина КРАВЧУК*
(ім'я, ПРІЗВИЩЕ)
« 10 » 06 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-ПРОФЕСІЙНОГО СТУПЕНЯ
ФАХОВИЙ МОЛОДШИЙ БАКАЛАВР

Тема: *Проєктування віконного інтерфейсу керування генератора сигналів довільної форми*

Група: *3-013* Спеціальність: *123 «Комп'ютерна інженерія»*

Здобувач освіти (підпис) *Артем КАСЯНЕНКО* (ім'я, ПРІЗВИЩЕ)

Керівник роботи (підпис) *Артем КУТІН* (ім'я, ПРІЗВИЩЕ)

Консультант з оформлення пояснювальної записки (підпис) *Оксана ОСАДЧА* (ім'я, ПРІЗВИЩЕ)

Кривий Ріг 2025 р.

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Відділення *комп'ютерної та програмної інженерії*
Циклова комісія *комп'ютерних систем та мереж*
Освітньо-професійний ступінь *фаховий молодший бакалавр*
Спеціальність *123 «Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
_____ *Ірина КРАВЧУК*
(підпис) (ім'я, ПРІЗВИЩЕ)
« *01* » *03* 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ОСВІТИ

Касяненко Артем Дмитрович
(прізвище, ім'я, по батькові)

1.Тема роботи *Проєктування віконного інтерфейсу керування генератора сигналів довільної форми*

Керівник роботи *викладач, Кутін Артем Ілліч*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по коледжу від « *04* » *04* 2025 року № *50-ст*

2. Строк подання здобувачем освіти роботи з *19.05.2025* по *13.06.2025*

3. Вихідні дані до роботи *Технічні характеристики генераторів сигналів довільної форми, мова програмування Python, контролер керування цифро-аналоговим перетворювачем Arduino nano*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Призначення генераторів довільної форми сигналів
2) Огляд існуючих моделей генераторів довільних сигналів з віконним інтерфейсом та без
3) Розробка інтерфейсу програми

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

Презентація Microsoft PowerPoint

6. Консультанти розділів роботи (проекту)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Узгодження технічного завдання з керівником кваліфікаційної роботи*	*24.03.2025-28.03.2025*	*виконано*
2	*Підбір та вивчення науково-технічної літератури за темою кваліфікаційної роботи*	*31.03.2025-04.04.2025*	*виконано*
3	*Обґрунтування вибору програмних засобів*	*07.04.2025-09.04.2025*	*виконано*
4	*Опис компонентів. Обґрунтування їх вибору.*	*10.04.2025-28.04.2025*	*виконано*
5	*Розробка програмного забезпечення*	*29.04.2025-02.05.2025*	*виконано*
6	*Дослідження ефективності реалізованих методів.*	*12.05.2025-23.05.2025*	*виконано*
7	*Написання пояснювальної записки*	*26.05.2025-30.05.2025*	*виконано*
8	*Перевірка на плагіат пояснювальної записки*	*02.06.2025-06.06.2025*	*виконано*
9	*Попередній захист кваліфікаційної роботи*	*09.06.2025-13.06.2025*	*виконано*
10	*Захист кваліфікаційної роботи*		

Здобувач освіти _____ *Артем КАСЯНЕНКО*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Артем КУТІН*
(підпис) (ім'я, ПРІЗВИЩЕ)

StrikePlagiarism.com

Дата звіту 6/16/2025
Дата редагування ---

Документ прийнятий

Звіт подібності

метадані

Назва організації
Ukrainian national aviation university
Заголовок
Артем_Касяненко_Нормоконтроль_та_антиплагіат_16_06_2025
Автор Науковий керівник / Експерт
КасяненкоКутін А.
підрозділ
Криворізький Фаховий коледж

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

19.72% 19.72% КП 1

2.36% 2.36% КЦ

25	10683	79561
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	100

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://tsibrov.blogspot.com/2018/06/ad9833.html	159 1.49 %
2	Методика передачі телеметрії БПЛА з використанням аудіоканалу аналогового відеопередавача [redacted] 1/21/2025 National University "Zaporizhzhia Polytechnic" (Кафедра "Радіотехніка та телекомунікації")	144 1.35 %
3	Методика передачі телеметрії БПЛА з використанням аудіоканалу аналогового відеопередавача [redacted] 1/21/2025 National University "Zaporizhzhia Polytechnic" (Кафедра "Радіотехніка та телекомунікації")	140 1.31 %

РЕФЕРАТ

Пояснювальна записка до дипломної роботи на тему: «Проєктування віконного інтерфейсу керування генератора сигналів довільної форми». Обсяг: 55 сторінки, 47 рисунків, 17 використаних джерел, 2 додатки.

Python, Arduino IDE, AD9833, Arduino Nano, генератор сигналів, графічний інтерфейс, DDS, керування приладом, модуляція, користувацький інтерфейс.

Мета роботи – створення віконного графічного інтерфейсу для зручного керування генератором сигналів довільної форми, що дозволяє реалізувати точне формування і виведення електричних сигналів заданої конфігурації.

У роботі досліджено функціональне призначення генераторів сигналів, проведено огляд існуючих моделей *SIGLENT*, *RIGOL*, *OWON* тощо. Проаналізовано архітектуру пристроїв, алгоритми формування хвиль та їх технічні параметри.

Інтерфейс реалізовано засобами мови програмування *Python*. Графічна оболонка забезпечує взаємодію з апаратною частиною генератора на базі модуля *AD*9833, керованого через *Arduino Nano*. Програма дозволяє задавати форму, амплітуду, частоту, фазу та інші параметри сигналу.

Система підтримує збереження конфігурацій, візуалізацію сигналів, модуляцію (*AM*, *FM*, *PM*), керування декількома каналами. Забезпечено роботу з графіками та збереження логів сеансів. Передбачено інтуїтивний дизайн для мінімізації часу навчання оператора.

Розроблене ПЗ ефективне у навчальних лабораторіях, де необхідна наочна генерація сигналів, а також у виробничих середовищах для перевірки й калібрування радіоелектронного обладнання.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ГДС – Генератор довільної форми сигналів

ЦАП – Цифро-аналоговий перетворювач

GUI (*graphical user interface*) – Графічний інтерфейс кори́стувача

***CSV*-модуль** – Модуль, який надає функціональність для роботи з файлами у форматі *CSV* (*Comma Separated Values*)

ПЗ (англ. *software*) – Програмне забезпечення

МГц (мегагерц) – це одиниця вимірювання частоти, що дорівнює одному мільйону герц (106 Гц) або одному мільйону коливань за секунду

ЕКГ, або **електрокардіографія**, – це метод дослідження роботи серця, який ґрунтується на вимірюванні та запису електричної активності серцевого м'яза

***COM*-порт**, або **послідовний порт**, це апаратний інтерфейс для передачі даних по одному біту за раз (послідовно) між комп'ютером та іншими пристроями.

PNG (*Portable Network Graphics*) — растровий формат збереження графічної інформації, що використовує стиснення без втрат.

МБ – Мегабайт (*megabyte*) одиниця вимірювання обсягу даних.

кБ – Кілоба́йт (кбайт, кБ, *kByte* , *kB*) одиниця вимірювання обсягу даних, яка застосовується в обчислювальній техніці та телекомунікації, дорівнює 103 = 1000 байт.

Мвиб/с (Мега-вибірок за секунду) – це одиниця, яка використовується для вимірювання швидкості дискретизації (*sampling rate*) в цифрових пристроях, наприклад у генераторах сигналів або аналого-цифрових перетворювачах.

ВСТУП

У сучасному світі розвиток цифрових технологій та електроніки сприяє зростанню потреби у гнучких і функціональних приладах для тестування та діагностики електронних систем. Одним із таких приладів є генератор сигналів довільної форми — універсальний інструмент, що дозволяє створювати електричні сигнали різної конфігурації з високою точністю. Ефективне використання цього пристрою значною мірою залежить від зручності та інтуїтивності його програмного інтерфейсу, що забезпечує взаємодію користувача з технічними можливостями генератора.

З огляду на це, актуальність теми дипломної роботи полягає в необхідності розробки віконного інтерфейсу керування, який би поєднував ергономіку, наочність та повну функціональність приладу. Такий підхід дозволить спростити роботу користувача, підвищити ефективність керування та мінімізувати ймовірність помилок при генерації сигналів.

Метою даної роботи є проєктування та реалізація інтерфейсу керування генератором сигналів довільної форми з використанням сучасних технологій програмування. У межах роботи передбачається аналіз вимог до інтерфейсу, розробка структури вікон, вибір інструментальних засобів реалізації, а також тестування готового програмного продукту.

Об'єктом дослідження є процес взаємодії користувача з генератором сигналів, а предметом — засоби програмної реалізації віконного інтерфейсу.

Практичне значення дипломного проєкту полягає у створенні прикладного програмного забезпечення, яке може бути інтегроване у реальні вимірювальні системи, що використовуються в освітніх, наукових або виробничих цілях.

РОЗДІЛ 1
ПРИЗНАЧЕННЯ ГЕНЕРАТОРІВ ДОВІЛЬНОЇ ФОРМИ СИГНАЛІВ

1.1 Поняття та призначення генератора довільної форми сигналів

У сучасній електроніці та радіотехніці генератори сигналів відіграють ключову роль. Серед них особливе місце посідають генератори довільної форми сигналів (ГДС).

Генератор довільної форми сигналів Рисунок 1.1 — це електронний пристрій, призначений для формування напруги або струму з попередньо визначеними або довільно заданими часовими характеристиками. ГДС дозволяє моделювати широкий спектр електричних сигналів з високим ступенем точності, що є необхідним для тестування, діагностики, калібрування та дослідження радіоелектронних, телекомунікаційних і автоматизованих систем.

Використання таких генераторів дозволяє імітувати поведінку складних електросигналів, зокрема біомедичних, акустичних, імпульсних або шумових, із точним контролем параметрів (амплітуди, частоти, фази, тривалості, повторюваності).

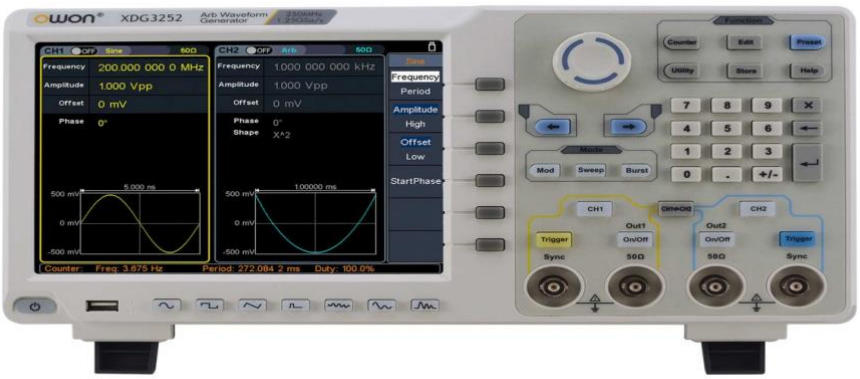


Рисунок 1.1 – Генератор довільної форми сигналу

1.2 Принцип дії ГДС

Основний принцип роботи генератора цифрових сигналів (ГДС) базується на формуванні цифрового представлення сигналу та подальшому перетворенні його в

аналогову форму за допомогою цифро-аналогового перетворювача (ЦАП). Сигнал спочатку дискретизується, тобто представляється у вигляді послідовності чисел, які зберігаються в оперативній пам’яті пристрою. Після цього, з заданою частотою дискретизації, ці дані подаються на вхід ЦАП, який формує неперервний аналоговий сигнал. Цей процес забезпечує високу точність відтворення та дає змогу змінювати форму, частоту, амплітуду й фазу сигналу без втручання в апаратну частину пристрою.

Формування сигналу у цифровому вигляді відбувається наступним чином. Користувач задає бажану форму сигналу, наприклад, через графічний інтерфейс або програму. Потім форма сигналу оцифровується: сигнал розбивається на дискретні значення амплітуди у визначені моменти часу. Ці значення зберігаються в пам’яті пристрою — зазвичай у вигляді таблиці або масиву значень.

Після збереження, дискретні значення зчитуються з пам’яті у заданому темпі, тобто з певною частотою. ЦАП перетворює кожне цифрове значення у відповідну аналогову напругу. Таким чином, на виході формується неперервний аналоговий сигнал, який відповідає заданій формі. Чим швидше відбувається зчитування сигналу, тим більш високочастотний сигнал формується на виході ЦАП.

ГДС зазвичай дозволяє налаштовувати основні параметри сигналу: частоту дискретизації, амплітуду, офсет (зсув по напрузі), фазу, тривалість або повторюваність сигналу.

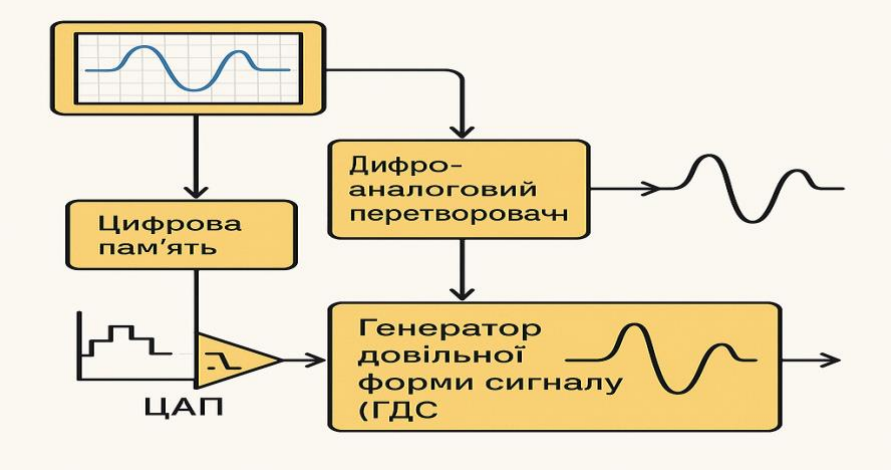


Рисунок 1.2 – Принцип дії ГДС у вигляді схеми

1.3 Основні характеристики генераторів

До ключових технічних характеристик генератора цифрових сигналів (ГДС) належать частотний діапазон, роздільна здатність ЦАП, швидкість оновлення, кількість каналів, рівень вихідного сигналу (амплітуда) та розмір пам’яті для збереження форм сигналу.

Частотний діапазон (*frequency band*) — це визначений інтервал частот, у межах якого коливання мають подібні властивості. Це може бути, наприклад, діапазон радіочастот, які приймає радіоприймач (УКХ, ДХ, КХ), діапазон звукових частот, який чує людина (від 20 Гц до 20 000 Гц), або частоти, на яких працює *Wi-Fi* (2,4 ГГц і 5 ГГц). Інші приклади — сантиметрові хвилі (*SHF*), які використовуються в радіолокації, або діапазон *ISM* для безліцензійних пристроїв. Сабвуфери працюють у діапазоні приблизно 40–200 Гц, а частота поділу кросовера визначає, які частоти направляються на низькочастотні динаміки.

Роздільна здатність ЦАП — це характеристика, яка визначає точність перетворення цифрового сигналу в аналоговий. Вона залежить від кількості бітів у цифровому коді: чим більше бітів, тим плавнішим і точнішим буде аналоговий вихід. Висока роздільна здатність покращує якість звуку (у випадку аудіо), дозволяє передавати більше нюансів, підвищує динамічний діапазон, зменшує спотворення і похибки. Роздільна здатність пов'язана з кількістю розрядів, нелінійністю перетворення та зсувом нуля на виході.

Швидкість оновлення — це кількість разів на секунду, коли оновлюються дані сигналу. Вона впливає на частоту та точність відтворення, а в графічних системах — на плавність виведення зображень. Вимірюється в герцах (Гц).

Кількість каналів — це параметр, що визначає, скільки незалежних сигналів може одночасно генерувати пристрій. Наприклад, двоканальний ГДС може одночасно створювати два різні сигнали.

Рівень вихідного сигналу або амплітуда — це максимальне відхилення сигналу від нуля. Амплітуда визначає силу або інтенсивність сигналу: у звукових системах — це гучність, в електроніці — напруга або струм. Збільшення амплітуди призводить до збільшення потужності сигналу. У підсилювачах рівень вихідного

сигналу зазвичай більший за вхідний. Амплітуду вимірюють осцилографами, мультиметрами або іншими вимірювальними приладами.

Розмір пам’яті для збереження форм сигналу залежить від тривалості сигналу, частоти дискретизації та кількості бітів на зразок. Чим довший сигнал, вища частота дискретизації або точніше представлення даних — тим більше пам’яті потрібно. Сигнали можуть зберігатися у різних форматах (наприклад, *WAV*, *MP*3, *PCM*), і вибір методу кодування впливає на розмір та якість збереженого сигналу.

Рисунок 1.3 – Основні характеристики генераторів довільної форми

1.4 Переваги використання ГДС

Докладний аналіз переваг для використання у навчальних, наукових та промислових умовах.

Головною перевагою ГДС є можливість створення абсолютно будь-якого сигналу. Це означає, що користувач не обмежується базовими формами (синус, прямокутник, пилка), а може задати хвильову форму, яка точно імітує сигнал з реального середовища, наприклад, біомедичний імпульс чи радіоперешкоду.

Цифрове зберігання та генерація сигналу дозволяє забезпечити дуже високу точність. На відміну від аналогових генераторів, де параметри можуть дрейфувати,

ГДС гарантують стабільність частоти, амплітуди, фази і форми сигналу протягом тривалого часу.

ГДС підтримують завантаження сигналів через комп'ютер або безпосередньо з інтерфейсу пристрою. Через інтерфейси *SCPI*, *USB*, *GPIB*, *Ethernet* користувач може автоматизувати генерацію, будувати складні послідовності та керувати сигналами з лабораторного програмного забезпечення (*MATLAB*, *LabVIEW*, *Python* тощо).

Багато моделей ГДС дозволяють використовувати модуляцію (*AM*, *FM*, *PM*), накладати шум, створювати послідовності імпульсів з точною затримкою та зміною параметрів. Це корисно для тестування систем зв'язку, цифрових фільтрів та ПЗС.

Завдяки використанню високоякісних цифро-аналогових перетворювачів (ЦАП), ГДС забезпечують частоти оновлення до декількох Гігагерц і роздільну здатність 14–16 біт. Це дає змогу відтворювати як повільні, так і надшвидкі сигнали без спотворень.

У деяких ГДС є декілька незалежних каналів, які можна використовувати для створення багатосмугових або синхронізованих сигналів. Це дозволяє тестувати багатоканальні системи або створювати складні сценарії сигналізації.

Сучасні генератори мають зручне програмне забезпечення, яке дозволяє візуально редагувати форму сигналу. Користувач може імпортувати графіки з *Excel* або *CSV*, малювати їх вручну, чи використовувати математичні рівняння.

Існують компактні портативні ГДС, що дозволяє проводити тестування у польових умовах, при виїзних обстеженнях або в навчальних лабораторіях, де немає змоги встановити повнорозмірне обладнання.

ГДС підтримують різні протоколи й стандарти, що дає змогу легко інтегрувати їх у тестові стенди, лабораторні комплекси та автоматизовані системи контролю.

Завдяки своїй гнучкості, ГДС активно використовуються в телекомунікаціях, радіоелектроніці, медицині, автомобілебудуванні, оборонній промисловості,

освітніх закладах та НДІ. Це універсальні пристрої, які здатні виконувати десятки типів вимірювальних завдань.

Рисунок 1.4 – Переваги використання ГДС

1.5 Недоліки та обмеження

Недоліки та обмеження генераторів цифрових сигналів (ГДС) включають кілька важливих аспектів, які слід враховувати під час вибору або використання таких пристроїв.

Перш за все, це вища вартість порівняно з простими генераторами. Наприклад, у порівнянні з генератором функцій, який здатен генерувати лише базові форми сигналів (синусоїда, прямокутник, трикутник), генератор довільної форми сигналу (*AWG*) може коштувати в десятки разів дорожче.

Ще одним обмеженням є потреба у спеціальному програмному забезпеченні або технічних знаннях. Для ефективної роботи з ГДС часто необхідно володіти навичками програмування (наприклад, використовуючи мову *SCPI*), працювати через спеціальні інтерфейси, такі як *LabVIEW* або *Matlab*, мати базове розуміння математичної обробки сигналів і вміти перетворювати сигнали у цифрову форму. Це створює бар’єр для студентів-початківців або користувачів без відповідної підготовки.

У недорогих моделях ГДС можуть бути обмеження щодо частоти оновлення (*sampling rate*) та глибини дискретизації (*bit depth*). Обмежена частота оновлення знижує здатність відтворювати високочастотні сигнали, а недостатня глибина дискретизації призводить до зниження точності, що особливо важливо при моделюванні складних або швидкоплинних сигналів. Усе це впливає на якість сигналу та сферу практичного застосування пристрою.

Рисунок 1.5 – Недоліки ГДС

1.6 Області застосування

ГДС використовуються в таких сферах як радіоелектроніка та телекомунікації для тестування радіомодулів, автоматизація для імітації вхідних сигналів, медицина для тестування ЕКГ/ЕЕГ приладів, а також наука й освіта для лабораторних досліджень.

У радіоелектроніці та телекомунікаціях ГДС допомагають із просторовим плануванням випробувань за допомогою геоінформаційних систем для визначення оптимальних точок розміщення антен, базових станцій або сенсорів. Системи також моделюють покриття території сигналом у реальному або симульованому середовищі, включаючи міське середовище, ліси та гори. Аналіз радіочастотного покриття здійснюється шляхом поєднання *RF*-моделей з просторовими даними для

візуалізації і вимірювання зон покриття та затінення сигналу, при цьому враховуючи висоту місцевості, наявність перешкод та інші геофактори. ГДС використовуються для моніторингу рухомих об'єктів під час тестування дронів або мобільних пристроїв з радіомодулями, відстежуючи маршрут та якість зв'язку в реальному часі. Тестування в реальних умовах проводиться завдяки інтеграції ГДС з телеметрією для аналізу якості зв'язку у визначених географічних зонах, виявлення мертвих зон, зон з перешкодами або затримками. Оптимізація мережі відбувається через проектування і тестування розміщення телекомунікаційної інфраструктури з урахуванням ландшафту і густоти населення.

В автоматизації ГДС застосовуються для тестування систем управління, подаючи напруги, струми або цифрові сигнали на входи програмованих логічних контролерів для перевірки правильності реакції логіки управління без запуску справжнього обладнання. Імітація датчиків здійснюється через відтворення сигналів від термопар, тензодатчиків, енкодерів та інших вимірювальних пристроїв, що дозволяє перевірити алгоритми обробки даних і поведінку системи при різних умовах. Налагодження *SCADA*-систем та автоматизованих систем управління технологічними процесами проводиться через подачу умовних або аварійних сигналів без фізичного втручання в реальні процеси.

У медицині ГДС використовуються для калібрування приладів через генерацію стандартних біоелектричних сигналів, що імітують ЕКГ або ЕЕГ, дозволяючи перевірити правильність зчитування, відображення та запису сигналів. Тестування точності проводиться створенням сигналів з відомими параметрами амплітуди, частоти та форми хвилі для визначення точності вимірювання. Симуляція патологій здійснюється через створення сигналів, що імітують аритмії, епілептичні напади, ішемію та інші стани для перевірки виявлення та реагування системи на критичні ситуації. ГДС також використовуються для тренування персоналу через симуляцію різних клінічних сценаріїв без ризику для пацієнтів.

У науці й освіті ГДС застосовуються для вивчення руху рідин, включаючи моделювання ламінарного й турбулентного потоку та дослідження властивостей в'язких рідин, поведінки струменів і вихорів. Дослідження теплообміну і

масообміну проводяться для вивчення процесів передачі тепла в рідинах, розчинення та дифузії речовин. Моделювання природних процесів включає відтворення умов річкових течій, підземних вод та океанічних струмів. Демонстраційні установки використовуються для вивчення студентами основ гідравліки й гідродинаміки в лабораторіях механіки рідин. Дослідження параметрів потоку охоплює вимірювання швидкості, тиску та турбулентності, а симуляція інженерних процесів включає створення лабораторних моделей трубопроводів, гідравлічних машин і насосів.

1.7 Структура та архітектура ГДС

Генератори довільної форми сигналів є складними електронними пристроями, архітектура яких інтегрує апаратні та програмні компоненти для формування сигналів з високою точністю та відтворюваністю. Типова структурна модель сучасного ГДС включає декілька функціональних підсистем, кожна з яких виконує специфічне завдання у процесі генерації сигналу.

Базова архітектура ГДС складається з семи основних компонентів, які забезпечують повний цикл генерації сигналу від цифрової форми до аналогового виходу.

Пам'ять хвильових форм використовується для зберігання цифрового представлення сигналів. Ці дані формуються користувачем або за допомогою програмного забезпечення і завантажуються у вигляді масиву дискретних амплітудних значень, що відповідають часовому ряду сигналу.

Цифро-аналоговий перетворювач забезпечує перетворення цифрових даних, збережених у пам'яті, у неперервний аналоговий сигнал. Важливими параметрами ЦАП є розрядність у бітах, яка визначає точність відтворення, та швидкість дискретизації, що впливає на максимальну частоту сигналу.

Генератор опорної частоти являє собою високостабільне джерело тактових імпульсів, яке визначає частоту оновлення сигналу. Найчастіше використовуються кварцові генератори з високою добротністю. У деяких реалізаціях можлива підтримка зовнішніх опорних джерел для синхронізації з іншими приладами.

Блок управління відповідає за конфігурування режимів роботи пристрою, обробку команд користувача, генерацію адрес зчитування з пам'яті, формування інтерфейсних протоколів. Найчастіше у якості контролера застосовуються мікроконтролери або програмовані логічні інтегральні схеми.

Блок фільтрації складається з аналогових фільтрів нижніх частот, які згладжують дискретні вихідні сигнали ЦАП, усуваючи високочастотні складові та артефакти квантування.

Інтерфейси зв'язку реалізують обмін даними з комп'ютером або зовнішніми керуючими пристроями. Серед поширених інтерфейсів використовуються *USB*, *RS*-232/*COM*, *Ethernet*, *SPI*, *I2C*, *GPIB*. Через інтерфейс відбувається завантаження хвильових форм, зміна параметрів, керування модуляцією тощо.

Інтерфейс користувача являє собою апаратну або програмну оболонку, що дозволяє оператору конфігурувати пристрій.

На концептуальному рівні архітектура ГДС реалізується у вигляді послідовності логічних етапів, кожен з яких виконує специфічну функцію в процесі формування сигналу.

Генерація цифрових зразків сигналу включає створення набору цифрових значень, що описують форму хвилі у вигляді функції часу. Можливе як програмне формування через графічний інтерфейс або скрипти, так і апаратне формування шаблонів з використанням вбудованих форм сигналів, таких як синус, прямокутник, пила тощо.

Цифрова обробка сигналу опційно застосовується для реалізації алгоритмів фільтрації, масштабування, нормалізації або модуляції сигналів ще до передачі їх на цифро-аналоговий перетворювач. У складних пристроях можливе використання спеціалізованих *DSP*-процесорів або програмованих логічних інтегральних схем.

Тактування та адресація забезпечує подачу керуючих імпульсів для послідовного зчитування зразків із пам'яті, синхронізованих з частотою дискретизації. Здійснюється за допомогою таймерів або *PLL*-блоків, що забезпечують точну часову базу.

Формування аналогового сигналу відбувається на етапі, коли аналоговий сигнал після ЦАП проходить через лінійний підсилювач та фільтри нижніх частот для формування стабільного сигналу, придатного для подальшого використання в електронних схемах або вимірювальних системах.

Модуляційні блоки реалізуються в деяких генераторах для забезпечення модуляції сигналу за амплітудою, частотою, фазою, або підтримки імпульсних режимів типу *Burst* і *Sweep*. Це дозволяє моделювати складні сценарії, що імітують реальні умови роботи систем зв'язку або сенсорних пристроїв.

У залежності від рівня складності пристрою розрізняють три основні архітектурні реалізації генераторів довільної форми сигналів.

Реалізація на основі мікроконтролера використовується для простих, низькочастотних ГДС. Відповідний варіант часто застосовується в навчальних або лабораторних стендах, наприклад, у поєднанні *Arduino* з мікросхемою *AD*9833.

Реалізація на базі програмованих логічних інтегральних схем призначена для високопродуктивних систем із потребою в паралельній генерації багатоканальних сигналів або реалізації спеціалізованих алгоритмів обробки.

1.8 Роль ГДС у системах тестування

Генератори довільної форми сигналу відіграють ключову роль у верифікації нових схем і приладів, забезпечуючи комплексне тестування різноманітних електронних компонентів та систем. При тестуванні аналогових фільтрів ГДС використовуються для перевірки частотної характеристики, дозволяючи точно оцінити поведінку фільтра в різних частотних діапазонах та виявити можливі відхилення від проектних параметрів.

У системах обробки сигналів генератори застосовуються для оцінки швидкодії і стабільності, подаючи контрольовані тестові сигнали з відомими параметрами, що дозволяє аналізувати якість обробки та виявляти потенційні проблеми в алгоритмах або апаратній реалізації.

При тестуванні радіоелектронних модулів ГДС виконують імітацію переданих сигналів з додаванням шумів та перешкод, що максимально наближує

умови тестування до реальних умов експлуатації. Це особливо важливо для перевірки стійкості приймальних трактів до різного роду завад та оцінки якості декодування сигналів.

ГДС дозволяють відтворити реальні умови експлуатації електронних систем у контрольованому середовищі, що значно покращує якість розробки та зменшує ризики виникнення проблем після впровадження продукту в експлуатацію. Завдяки можливості генерації складних сигналів з точно заданими параметрами, інженери можуть проводити детальну діагностику систем та виявляти потенційні недоліки на ранніх стадіях розробки.

Таким чином, генератори довільної форми сигналів являють собою універсальні інструменти, які відкривають широкі можливості для аналізу, діагностики та моделювання електронних систем. Їхнє використання є невід'ємною частиною сучасних наукових досліджень, розробок і технічного тестування, забезпечуючи високу якість та надійність створюваних електронних пристроїв.

РОЗДІЛ 2
ОГЛЯД ІСНУЮЧИХ ГЕНЕРАТОРІВ ДОВІЛЬНИХ СИГНАЛІВ

2.1 Генератори довільної форми сигнали *SIGLENT*

Генератор сигналів *SIGLENT SDG*830 (Рисунок 2.1) являє собою одноканальний генератор сигналів з частотою до 30 МГц та частотою дискретизації 125 Мвиб/с. Пристрій оснащений пам'яттю глибиною 16 кБ і підтримує широкий спектр функцій модуляції, включаючи амплітудну модуляцію, подвійну бічну смугу з пригніченою несучою, частотну та фазову модуляцію, частотну та амплітудну маніпуляцію, широтно-імпульсну модуляцію, а також режими сканування та пакетного формування сигналів. Вартість пристрою становить 17,640.00₴.

Рисунок 2.1 – *SIGLENT SDG*830

Генератор сигналів *SIGLENT SDG*1032*X* (Рисунок 2.2) представляє собою двоканальний генератор сигналів з максимальною вихідною частотою 30 МГц та частотою дискретизації 150 Мвиб/с. Пристрій забезпечує високу точність завдяки вертикальній роздільності 14 біт, що дозволяє формувати сигнали з детальним відтворенням амплітудних характеристик. Вартість становить 17,514.00₴.

Рисунок 2.2 – *SIGLENT SDG*1032*X*

Генератор сигналів *SIGLENT SDG*6022*X* (Рисунок 2.3) є високопродуктивним двоканальним генератором з розширеним діапазоном частот від 1 мкГц до 200 МГц. Пристрій оснащений потужною системою дискретизації з частотою 2,4 ГВиб/с та 16-бітною роздільністю, а також великим обсягом пам'яті 20 МБ для зберігання складних хвильових форм. Смуга пропускання становить 200 МГц, що забезпечує високоякісне відтворення широкосмугових сигналів. Вартість пристрою складає 77,441.70₴.

Рисунок 2.3 – *SIGLENT SDG*6022*X*

Генератор сигналів *SIGLENT SDG*2122*X* (Рисунок 2.4) являє собою двоканальний генератор сигналів середнього класу з вихідною частотою до 120 МГц та частотою дискретизації 1,2 Гвиб/с. Пристрій забезпечує високу точність сигналів завдяки вертикальній роздільній здатності 16 біт, що робить його

придатним для широкого спектра застосувань у сфері тестування та вимірювань. Вартість становить 43,638.00₴.

Рисунок 2.4 – *SIGLENT SDG*2122*X*

2.2 Генератори довільної форми сигнали *RIGOL*

Універсальний генератор сигналів *RIGOL DG*1062*Z* (Рисунок 2.5) являє собою двоканальний генератор сигналів з максимальною частотою 60 МГц та частотою дискретизації 200 МВиб/с. Пристрій оснащений стандартною пам'яттю обсягом 8 Мточок, що забезпечує достатній ресурс для зберігання складних хвильових форм та виконання різноманітних завдань генерації сигналів. Вартість пристрою становить ₴72,030.00.

Рисунок 2.5 – *RIGOL DG*1062*Z*

Генератор сигналів *RIGOL DG*5071 (Рисунок 2.6) представляє собою одноканальний генератор сигналів з пропускною здатністю 70 МГц та високою частотою дискретизації 1 ГВиб/с. Пристрій забезпечує точне відтворення сигналів

завдяки вертикальній роздільності 14 біт, що дозволяє формувати високоякісні сигнали з детальним контролем амплітудних характеристик. Вартість становить ₴126,044.10.

Рисунок 2.6 – *RIGOL DG*5071

Високочастотний генератор сигналів *RIGOL DSG*830 (Рисунок 2.7) є спеціалізованим високочастотним генератором, що забезпечує стандартні аналогові модуляції, включаючи амплітудну, частотну та фазову модуляцію, з максимальною робочою частотою 3 ГГц. Пристрій підтримує функції свіпування за частотою і рівнем, що робить його ідеальним інструментом для тестування високочастотних радіоелектронних систем та компонентів. Вартість пристрою складає ₴234,653.16.

Рисунок 2.7 – *RIGOL DSG*830

2.3 Генератори довільної форми сигналів

Серед сучасних генераторів довільної форми сигналів особливої уваги заслуговує двоканальний генератор сигналів довільної форми *XDG*3252 зі смугою 250 МГц виробництва компанії *OWON*, показаний на рисунку 2.8. Цей пристрій є представником нового покоління генераторів сигналів довільних форм, що забезпечує кращу тимчасову та амплітудну точність у генерації. Архітектура генератора базується на основі покращеної технології прямого цифрового синтезу (*Advanced DDS*), що значно розширює можливості створення складних форм сигналів. Пристрій оснащений великим 8-дюймовим дисплеєм з роздільною здатністю 800×600 пікселів.

Рисунок 2.8 – *XDG*3252

Генератори *OWON* серії *XDG*3000 відрізняються підвищеною точністю, стабільністю та малими спотвореннями. Дискретність установки частоти становить 1 мкГц, діапазон частот вихідного сигналу досягає 250 МГц для моделі *XDG*3252. Пристрій має вбудований 7-розрядний частотомір зі смугою до 200 МГц. Генератори серії *XDG*3000 забезпечують роботу з усіма основними видами модуляцій та їх різновидами: *AM*, *FM*, *PM*, *PWM*, *FSK*, 3*FSK*, 4*FSK*, *PSK*, *OSK*, *ASK*,

BPSK, *Sweep*, *Burst*, забезпечують лінійне та логарифмічне свипування. При необхідності генератори *OWON* мають можливість синхронізації із зовнішнім джерелом сигналу та опорної частоти. *USB* інтерфейс забезпечує підключення приладу до персонального комп'ютера для керування та обміну формами сигналу, також на передній панелі розташований вхід для флеш накопичувача. Ціна даного генератора становить 67400 грн.

Іншим представником є двоканальний генератор довільних форм сигналів *DDS UTG*2082*B* зі смугою генерації синусоїдального сигналу 80 МГц, швидкістю вибірки цифро-аналогового перетворювача 320 МВ/с та об'ємом пам'яті до 16М точок виробництва компанії *UNI-T* (див. рисунок 2.9). Основними особливостями даного генератора є висока роздільна здатність по амплітуді 16 біт (включаючи знаковий розряд), велика кількість вбудованих форм сигналів - 160, а також розширена підтримка модуляцій: *AM*, *FM*, *PM*, *ASK*, *FSK*, *PSK*, *PWM*, *QAM*, *SUM*, *BPSK*, *QPSK*, *OSK*. У генератор *UTG*-2082*B* вбудований частотомір із смугою до 200 МГц. Ціна становить 25584 грн.

Рисунок 2.9 – *UTG*2082*B*

Низькочастотний генератор довільних форм сигналів *FY*6600-60*M* зі смугою 60 МГц та 2 каналами виробництва компанії *FeelTech* показано на рисунку 2.10. Генератор виконаний за технологією прямого цифрового синтезу (*DDS*) і забезпечує високі часові та амплітудні характеристики сигналів, що генеруються. Пристрій оснащений 14-бітовим ЦАП зі швидкістю вибірки 250 МВ/с і пам'яттю 8192 пікселів на осцилограму, має вбудований частотомір до 100 МГц з функцією рахунку імпульсів. Вбудована пам'ять розрахована на 98 форм довільних форм

сигналів, є попередньо встановлені форми: синус, прямокутник, імпульс, пила, напівхвиля, повна хвиля, експонента, зворотна експонента, мультитон, ЕКГ, трапеція, білий шум (Гаусса). Генератор *FY*6600-60*M* підтримує амплітуду, усунення чи наповнення (шпаруватості) і модуляції *AM*, *FM*, *PM*, *ASK*, *FSK*, *PSK*. Серія генераторів *FeelTech FY*6600-60*M* складається з 4 однотипних моделей, що відрізняються тільки максимальною частотою генерації. Ціна становить 5300 грн.

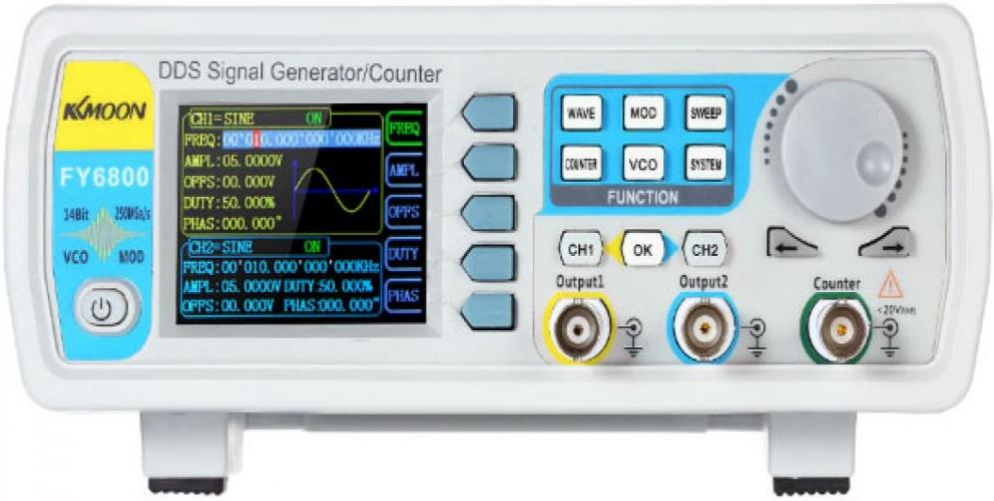


Рисунок 2.10 – *FY*6600-60*M*

Низькочастотний генератор довільних форм сигналів *JDS*6600-15*M* зі смугою 15 МГц та 2 каналами виробництва компанії *Juntek* представлений на рисунку 2.11. Генератор виконаний за технологією прямого цифрового синтезу (*DDS*) і забезпечує високі часові та амплітудні характеристики сигналів, що генеруються. Пристрій оснащений 14-бітовим ЦАП зі швидкістю вибірки 266 МВ/с і пам'яттю 2048 пікселів на осцилограму, має вбудований частотомір до 100 МГц з функцією рахунку імпульсів. Вбудована пам'ять розрахована на 98 форм довільних форм сигналів, є попередньо встановлені форми: синус, прямокутник, імпульс, пила, напівхвиля, повна хвиля, експонента, зворотна експонента, мультитон, ЕКГ, трапеція, білий шум (Гаусса). Серія *DDS*-генераторів *Juntek JDS*6600-15*M* складається з 5 однотипних моделей, що відрізняються тільки максимальною частотою генерації. Ціна становить 3973 грн.

Рисунок 2.11 – *JDS*6600-15*M*

Генератор *GW Instek AFG*-3031, показаний на рисунку 2.12, належить до серії, що включає одноканальні і двоканальні моделі 20 МГц/30 МГц, призначені для застосування в промисловості, наукових дослідженнях та освіті. У конструкції із ізольованим виходом всі вихідні канали мають заземлення, що дозволяє використовувати їх для випробувань плаваючого ланцюга. Ціна становить 81756 грн.

Рисунок 2.12 – *GW Instek AFG*-3031

Генератор сигналів довільної форми серії *AWG*70000*B* (див. рисунок 2.13) відрізняється високою частотою дискретизації, точністю відтворення сигналів та великою пам'яттю для сигналів, що робить його ідеальним приладом для експериментів, а також для розробки, випробування та перевірки функціонування складних компонентів та систем. Прилад із частотою дискретизації до 50 *GS*/*s* та роздільною здатністю по вертикалі 10 біт вважається найкращим у галузі рішенням для випробувань з швидкою та простою генерацією ідеально точних, спотворених та реальних сигналів.

Рисунок 2.13 – *AWG*70000*B*

На основі проведеного аналізу генераторів довільної форми, які доступні на ринку, можна зробити висновок, що інформація про користувацьку форму сигналу може бути завантажена у генератор наступними шляхами: за допомогою флеш пам'яті, за допомогою віконного інтерфейсу та *USB* кабелю, редагуванням доступних примітивів за допомогою органів керування самого генератора.

Порівняльний аналіз характеристик генераторів довільної форми показує широкий спектр доступних рішень. Серед моделей *SIGLENT* можна виділити *SDG*830 з одним каналом та полосою 30 МГц, частотою дискретизації 125 *MS*/*s* та 14-бітною розрядністю, який підтримує стандартні та довільні модуляції і дозволяє додавання користувацьких форм сигналів через передню панель та *USB Flash*, *PC* з *EasyWave*, через *SCPI* програмно. Двоканальні моделі серії включають *SDG*1032*X*, *SDG*1062*X* з полосами відповідно 30 та 60 МГц, частотою дискретизації 150 *MS*/*s*, а також більш потужні *SDG*2042*X*, *SDG*2082*X*, *SDG*2122*X* з полосами 40, 80 та 120 МГц відповідно та частотою дискретизації 1.2 *GS*/*s*. Найпотужнішою в лінійці є модель *SDG*6022*X* з полосою 200 МГц та частотою дискретизації 2.4 *GS*/*s*.

Генератори *RIGOL* представлені широкою лінійкою моделей, починаючи від *DG*1022*Z* з полосою 25 МГц, частотою дискретизації 200 *MS*/*s* та 14-бітною розрядністю, до більш потужних моделей серії *DG*5000 з частотою дискретизації 1 *GS*/*s*. Особливої уваги заслуговують моделі *DG*1062*Z* з полосою 60 МГц, пам'яттю 8 *Mpts* та підтримкою модуляцій *AM*, *FM*, *PM*, *FSK*, *PWM*, а також високочастотні моделі *DSG*815 та *DSG*830 з полосами 1500 та 3000 МГц відповідно. Всі генератори *RIGOL* підтримують додавання користувацьких форм через передню панель та *USB*, через *PC* з *UltraWave*, через *SCPI* або скрипти *Python*.

Серія *OWON* включає моделі *XDG*3000 з різними смугами частот від 80 до 250 МГц, включаючи *XDG*3082, *XDG*3102, *XDG*3162, *XDG*3202 та *XDG*3252. Всі моделі підтримують модуляції *AM*, *FM*, *PM*, *FSK*, *Sweep* та дозволяють завантаження користувацьких форм через *USB* у форматах *CSV*/*BIN*, фірмовий софт, через *SCPI*. Модель *DGE*2070 має полосу 70 МГц та частоту дискретизації 300 *MS*/*s*.

Бюджетні рішення представлені моделями *UNI-T UTG*2082*B* з полосою 80 МГц, *FY*2300*H*-60*M* та *FY*6600-60*M* з полосою 60 МГц, *JDS*6600-15*M* з полосою 15 МГц, частотою дискретизації 266 *MS*/*s*, 14-бітною розрядністю та пам'яттю 2048 точок. Ці моделі підтримують основні типи модуляцій та дозволяють завантаження користувацьких форм через *USB*.

Серія *GW Instek* включає моделі *AFG*-3000 з полосами 20-30 МГц, частотою дискретизації 250 *MS*/*s*, 16-бітною розрядністю та пам'яттю 8 *MB*, що підтримують модуляції *AM*, *FM*, *PM*, *Sweep*, *Burst*. Також представлені спеціалізовані моделі *SFG*-1013 з полосою 3 МГц та *AFG*-125*P* з полосою 25 МГц, частотою дискретизації 120 *MS*/*s*, 10-бітною розрядністю та пам'яттю 4*k* точок.

Професійні рішення включають серію *Tektronix AWG*7000*B* з частотою дискретизації 6-24 *GS*/*s*, 10-бітною розрядністю та широкою підтримкою категорій сигналів, а також модель *RIGOL DG*2021*A* з полосою 20 МГц, частотою дискретизації 100 *MS*/*s*, 14-бітною розрядністю та пам'яттю 4 *KB*.

РОЗДІЛ 3
РОЗРОБКА ІНТЕРФЕЙСУ ПРОГРАМИ

3.1 Програмне забезпечення генератора довільної форми сигналів

Програмне забезпечення є ключовим компонентом у функціонуванні генератора довільної форми сигналів, оскільки саме воно забезпечує інтерпретацію, управління, обчислення та взаємодію з апаратною частиною пристрою. У сучасних реалізаціях програмне забезпечення не лише виконує базову функцію створення та передачі хвильових форм, але й забезпечує розширену функціональність: обробку сигналів, автоматизацію вимірювань, модуляцію, інтеграцію з іншими програмними платформами.

До основних завдань програмного забезпечення генератора довільної форми сигналів належить генерація форми сигналу та керування параметрами генерації. На основі вибраного типу сигналу та введених параметрів, таких як амплітуда, частота, фаза та тривалість, програмне забезпечення формує відповідний масив числових значень, що описують часову залежність амплітуди сигналу. Важливою функцією є візуалізація сигналу, яка полягає у побудові графіка сигналу на основі сформованого масиву значень з використанням спеціалізованих бібліотек візуалізації, таких як *matplotlib* або *pyqtgraph*. Це дозволяє користувачеві візуально контролювати відповідність сигналу технічним вимогам.

Передача даних на апаратну частину є критично важливим завданням, при якому програмне забезпечення формує пакет даних згідно з протоколом *SPI*, *UART* або *USB* і надсилає його на пристрій генератора довільної форми сигналів для подальшого відтворення сигналу. У випадку мікросхеми *AD*9833 це являє собою послідовність бітів конфігурації регістрів генератора. Додатково програмне забезпечення забезпечує інтеграцію з зовнішніми інструментами, підтримуючи експорт у формати, сумісні з *MATLAB*, *Excel*, *LabVIEW*, що дозволяє виконувати розширений аналіз сформованого сигналу.

Архітектура програмного забезпечення побудована на модульному принципі та включає декілька основних компонентів. Графічний інтерфейс користувача

реалізує засоби взаємодії користувача із системою та створений на базі бібліотеки *Tkinter*. Він забезпечує введення параметрів сигналу, вибір типу сигналу, запуск процесів генерації, експорт та передавання даних.

Модуль генерації сигналів відповідає за створення масиву значень сигналу відповідно до заданих параметрів. Він реалізований за допомогою бібліотеки *NumPy*, яка забезпечує швидке та ефективне обчислення математичних функцій, включаючи синусоїдальні, прямокутні, трикутні та інші форми сигналів. Модуль візуалізації використовує *matplotlib* для графічного відображення сигналу та включає інструменти масштабування, оновлення та збереження графіка.

Модуль зв'язку з пристроєм реалізований через *pyserial* для забезпечення передачі конфігураційних даних у форматі послідовного інтерфейсу через *COM*-порт. Він підтримує вибір доступного порту, перевірку з'єднання та обробку винятків. Мовний модуль відповідає за адаптацію інтерфейсу до вибраної мови користувача, використовуючи словникову структуру для підстановки відповідних текстових шаблонів у графічний інтерфейс користувача.

Модулі *CSV* та *PNG* відіграють ключову роль у процесах експорту даних та графічної інформації. *CSV*-модуль відповідає за збереження цифрових даних сигналу у текстовому табличному форматі, сумісному з популярними програмами для аналізу даних, такими як *Microsoft Excel*, *LibreOffice Calc* або *MATLAB*. Це дозволяє користувачеві зручно зберігати та передавати результати генерації сигналів, а також здійснювати подальший аналіз поза межами програмного середовища.

При проєктуванні програмного забезпечення генератора довільної форми сигналів враховуються як функціональні, так і нефункціональні вимоги. Функціональні вимоги включають підтримку базових типів сигналів, таких як синусоїдальний, прямокутний, трикутний та довільний, гнучке налаштування параметрів сигналу, відображення сигналу у графічному вигляді, збереження сигналу у форматах *CSV* та *PNG*, а також передачу параметрів у зовнішній пристрій.

Нефункціональні вимоги передбачають мінімальне використання системних ресурсів, забезпечення інтуїтивного інтерфейсу користувача, стійкість до помилок введення та можливість масштабування для подальшого розширення функціональності, включаючи додавання нових типів сигналів та підтримку інших пристроїв.

3.2 Інтерфейс користувача

Інтерфейс програми умовно складається з двох основних панелей: панелі керування параметрами сигналу, що розташована у лівій та верхній частині вікна, та графічної області для побудови сигналу, розміщеної у правій нижній частині. Інтерфейс розділено на логічні функціональні зони, що забезпечують зручність роботи користувача.

Перша функціональна зона передбачає вибір типу сигналу, включаючи синусоїдальний, прямокутний, трикутний та довільний типи. Інтерфейс програмного забезпечення передбачає можливість вибору типу генерованого сигналу зі списку стандартних форм: синусоїдальний, прямокутний, трикутний, а також довільний. Тип сигналу задається за допомогою елемента управління *Combobox*, як показано на рисунку 3.1 та рисунку 3.2, що забезпечує користувача швидким вибором із заздалегідь визначеного переліку. Вибір відповідного типу сигналу безпосередньо впливає на алгоритм розрахунку значень амплітуди на кожному часовому кроці при генерації сигналу.

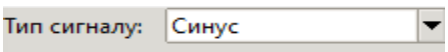


Рисунок 3.1 – Вибір типу генерованого сигналу

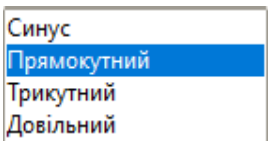


Рисунок 3.2 – Список стандартних форм сигналу

Для налаштування основних параметрів сигналу, таких як частота у герцах, амплітуда в умовних одиницях напруги та фаза у радіанах, передбачено відповідні поля введення типу *Entry*, як показано на рисунку 3.3. Користувач має можливість вручну задати числові значення, які впливають на форму та динаміку синтезованого сигналу. Введені значення валідуються в межах допустимих діапазонів, що забезпечує коректну генерацію сигналів у реальному часі.

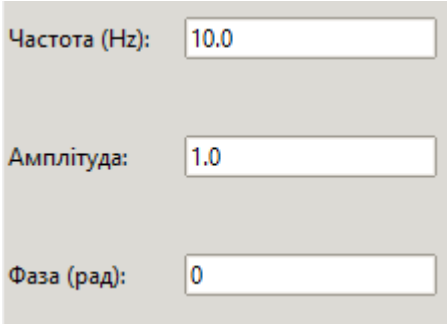


Рисунок 3.3 – Поля для введення параметрів

Для підвищення зручності у взаємодії з параметрами сигналу реалізовано паралельну можливість їх зміни за допомогою повзунків *Scale*, як продемонстровано на рисунку 3.4. Ці елементи дозволяють динамічно змінювати значення без необхідності введення чисел вручну. Слайдери інтерактивно пов'язані з відповідними текстовими полями, що забезпечує двосторонню синхронізацію введених даних. Таким чином, користувач має змогу швидко проводити експериментальні налаштування параметрів сигналу.

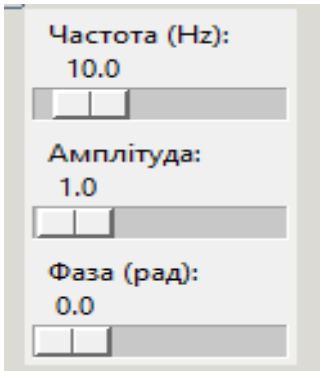


Рисунок 3.4 – Слайдери для динамічного налаштування параметрів

Для графічної візуалізації сформованого сигналу використовується вбудована область побудови графіка на основі бібліотеки *matplotlib*, як показано на рисунку 3.5. Побудова здійснюється у вигляді двовимірного графіка залежності амплітуди від часу. Компонент *FigureCanvasTkAgg* забезпечує інтеграцію графіка у вікно інтерфейсу, дозволяючи користувачу переглядати результат генерації в режимі реального часу. Також підтримується масштабування, збереження та навігація по графіку за допомогою *NavigationToolbar2Tk*.

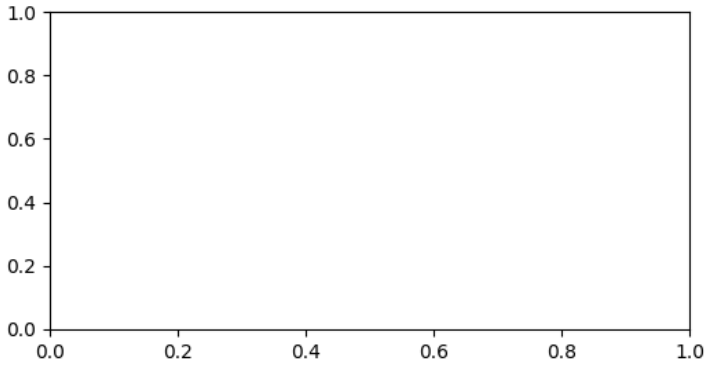


Рисунок 3.5 – Область побудови сигналу

Функціональні кнопки інтерфейсу, представлені на рисунку 3.6, дозволяють виконувати основні дії із сигналом. Кнопка «Згенерувати сигнал» ініціює побудову сигналу на основі заданих параметрів, «Зберегти сигнал» зберігає дані сигналу у форматі *CSV*, що є зручним для подальшого аналізу, «Завантажити сигнал» дозволяє імпортувати сигнал із *CSV*-файлу, «Експорт у *PNG*» зберігає побудований графік у вигляді зображення, а «Показати дані» виводить таблицю значень часу та амплітуди у форматі *Treeview*. Ці елементи забезпечують повний цикл роботи із сигналами: від налаштування до збереження та виводу результатів.

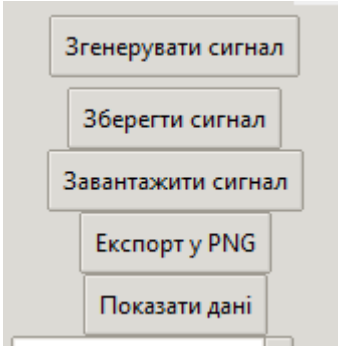


Рисунок 3.6 – Кнопки керування

Для забезпечення обміну даними з зовнішнім мікроконтролерним пристроєм, наприклад на основі *AD*9833, у програмному забезпеченні реалізовано модуль серійного зв'язку. Елемент *Combobox* дозволяє вибрати один із наявних *COM*-портів, як показано на рисунку 3.7, після чого здійснюється ініціалізація з'єднання. Надсилання параметрів сигналу відбувається у форматі рядка, закодованого відповідно до протоколу обміну. Всі операції супроводжуються повідомленнями про успішність або помилки підключення.

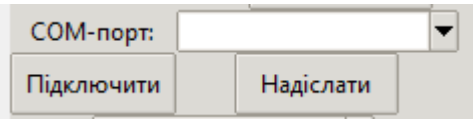


Рисунок 3.7 – Вибір *COM*-порту, підключення та передача даних

З метою підвищення універсальності застосунку реалізовано багатомовний інтерфейс. Перемикач мов *Combobox*, зображений на рисунку 3.8, дозволяє обрати між українською та англійською мовами. Після вибору, усі текстові елементи інтерфейсу автоматично оновлюються згідно з вибраною локалізацією. Механізм реалізовано через словник мовних шаблонів *LANGS*, що забезпечує масштабованість для додавання інших мов у майбутньому.

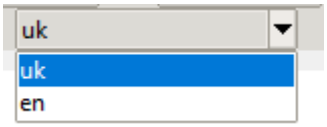


Рисунок 3.8 – Перемикач мови інтерфейсу

Інтерфейс, загальний вигляд якого представлено на рисунку 3.9, був розроблений для легкого, інтуїтивного та ефективного використання генератора довільних сигналів. Такий підхід до проектування інтерфейсу забезпечує зручність роботи як для досвідчених користувачів, так і для початківців у сфері генерації та обробки сигналів.

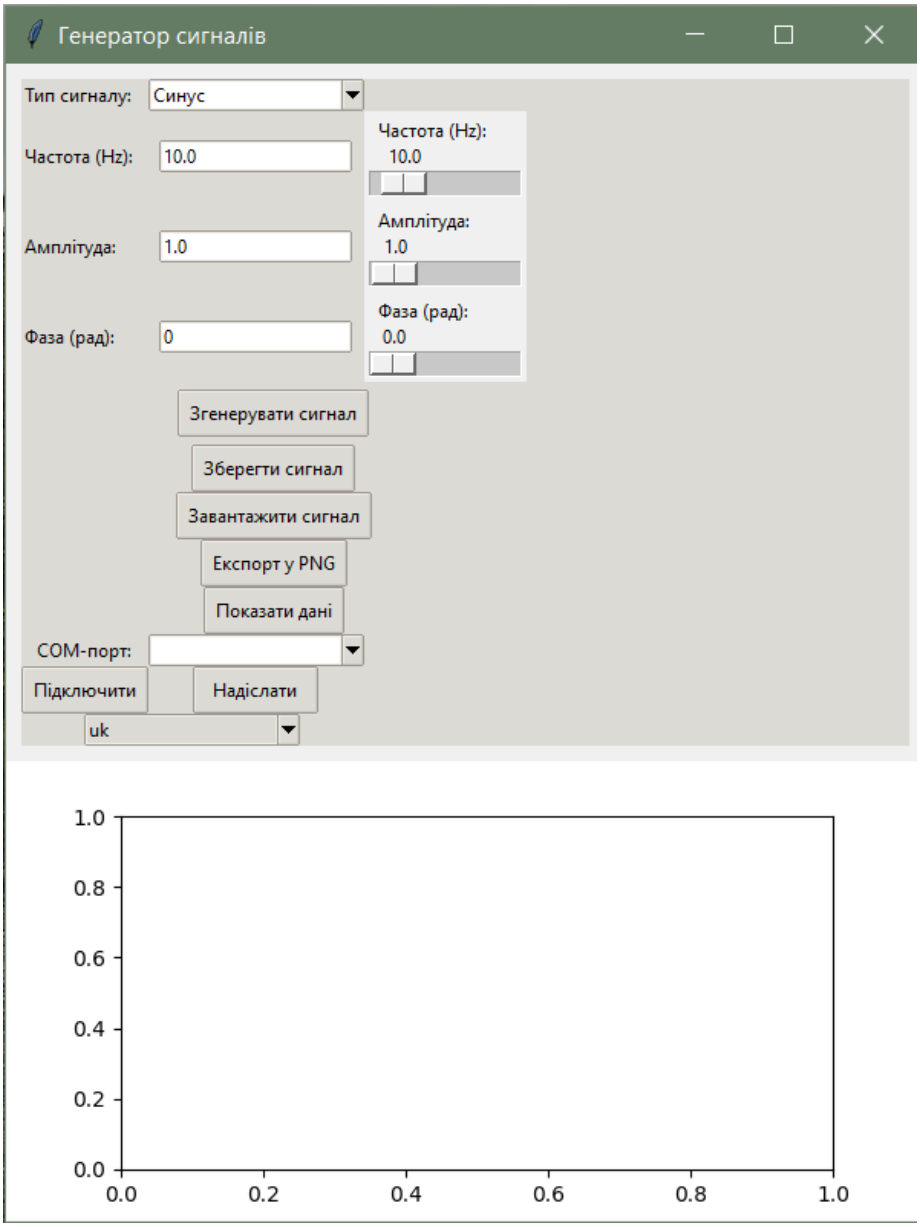


Рисунок 3.9 – Інтерфейс генератора сигналів

3.3 Алгоритм взаємодії користувача з програмним забезпеченням

Процес взаємодії користувача з програмним забезпеченням для керування генератором сигналів довільної форми реалізований за принципами інтуїтивного управління, що передбачає покрокову активацію функціональних модулів інтерфейсу відповідно до логіки роботи генератора.

На першому етапі користувач здійснює вибір форми сигналу із запропонованого переліку, що включає синусоїдальний, прямокутний, трикутний та довільний типи сигналів. Для цього використовується комбінований список *Combobox*, що забезпечує єдиний формат введення та унеможливлює помилкові

або непідтримувані значення. Обраний тип сигналу визначає математичну модель функції, яка буде використана на наступному етапі для генерації масиву значень амплітуди.

Після вибору типу сигналу користувач встановлює числові параметри: частоту у герцах, амплітуду в умовних одиницях напруги та фазовий зсув у радіанах. Введення параметрів можливе як через текстові поля, так і за допомогою слайдерів, що дозволяє оперативно адаптувати характеристики сигналу до потреб дослідження. Значення перевіряються на відповідність допустимим діапазонам у межах програмної валідації, що знижує ймовірність генерації некоректних сигналів.

Після встановлення всіх параметрів користувач ініціює процес генерації натисканням кнопки «Згенерувати сигнал». Відповідний обробник подій виконує обчислення за допомогою математичних функцій, наприклад *numpy.sin*() для синусоїдального сигналу, формуючи масив дискретних значень амплітуди в часовому домені. Формула враховує всі введені параметри та генерує масив із фіксованою кількістю точок, що забезпечує плавність графіка.

Після генерації сигналу програма автоматично ініціює побудову графіка у відповідному віджеті *matplotlib*, інтегрованому в графічне середовище інтерфейсу. Графік відображає залежність амплітуди сигналу від часу, має сітку координат, підписи осей та панель навігації. Це дозволяє користувачеві візуально оцінити правильність сформованого сигналу, зокрема його частотну характеристику, періодичність та симетрію.

У разі необхідності користувач має змогу виконати декілька додаткових дій. Експорт графіка у формат *PNG* дозволяє зберегти графік як растрове зображення для подальшого включення в технічну документацію або презентаційні матеріали. Збереження сигналу у *CSV* означає експорт масиву значень часу та амплітуди у вигляді таблиці, що дозволяє обробляти його в сторонніх програмних середовищах, таких як *Excel* або *MATLAB*.

Передача параметрів через *COM*-порт здійснюється за допомогою попередньо налаштованого серійного порту, через який параметри сигналу у

форматі структурованого рядка надсилаються на зовнішній пристрій, наприклад мікроконтролер з мікросхемою *AD*9833, що дозволяє здійснити фізичну генерацію сигналу апаратним засобом.

Усі дії супроводжуються відповідними повідомленнями про успішність операції або наявність помилок, наприклад відсутність підключеного пристрою або некоректне значення параметра, що забезпечує контроль на кожному етапі виконання. Такий підхід до організації взаємодії гарантує зручність використання програмного забезпечення як для досвідчених користувачів, так і для початківців у сфері генерації та обробки сигналів.

3.4 Роз’яснення коду програми

Роз'яснення коду програми *Python* включає детальний аналіз усіх компонентів програмного забезпечення генератора сигналів довільної форми. Структура коду побудована таким чином, щоб забезпечити максимальну функціональність при збереженні читабельності та модульності.

Перший блок коду містить імпорти необхідних бібліотек, як показано на рисунку 3.10. Бібліотеки *tkinter* та *ttk* використовуються для створення графічного інтерфейсу користувача. Модулі *messagebox*, *filedialog* та *Toplevel* забезпечують функціональність для показу повідомлень, збереження та завантаження файлів, а також відкриття нових вікон. Бібліотека *numpy* застосовується для математичних обчислень і формування сигналів, *matplotlib* для побудови графіків усередині інтерфейсу, а *serial* для обміну даними через *COM*-порт.

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog, Toplevel
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
import serial
import serial.tools.list_ports
```

Рисунок 3.10 – Імпорти бібліотек

Словник локалізацій, представлений на рисунку 3.11, визначає тексти інтерфейсу українською та англійською мовами. Вибір мови реалізовано через

змінну *language_var* з елементом *Combobox*, що дозволяє користувачеві динамічно перемикатися між підтримуваними мовами інтерфейсу.

```
LANGS = {
    "uk": {
        "title": "Генератор сигналів",
        "signal_type": "Тип сигналу:",
        "freq": "Частота (Hz):",
        "amp": "Амплітуда:",
        "phase": "Фаза (рад):",
        "generate": "Згенерувати сигнал",
        "com_port": "COM-порт:",
        "connect": "Підключити",
        "send": "Надіслати",
        "save": "Зберегти сигнал",
        "load": "Завантажити сигнал",
        "export": "Експорт у PNG",
        "table": "Показати дані",
        "language": "Мова"
    },
    "en": {
        "title": "Signal Generator",
        "signal_type": "Signal Type:",
        "freq": "Frequency (Hz):",
        "amp": "Amplitude:",
        "phase": "Phase (rad):",
        "generate": "Generate Signal",
        "com_port": "COM Port:",
        "connect": "Connect",
        "send": "Send",
        "save": "Save Signal",
        "load": "Load Signal",
        "export": "Export to PNG",
        "table": "Show Data",
        "language": "Language"
    }
}
```

Рисунок 3.11 – Мови інтерфейсу

Клас *SignalGeneratorApp*, зображений на рисунку 3.12, інкапсулює всю логіку програми. Основні компоненти класу включають метод ініціалізації *init*, який встановлює параметри вікна, тему оформлення, завантажує локалізацію, ініціалізує *COM*-порт та будує інтерфейс і графік. Методи *translate*, *switch_language* та *refresh_labels* забезпечують багатомовну підтримку: *translate* повертає переклад ключа згідно з поточною мовою, *switch_language* обробляє подію зміни мови, а *refresh_labels* перебудовує графічний інтерфейс після зміни мови.

```
class SignalGeneratorApp:
    def __init__(self, root):
        self.root = root
        self.root.geometry("800x600")
        self.root.resizable(True, True)
        self.style = ttk.Style()
        self.style.theme_use("clam")
        self.lang = "uk"
        self.translations = LANGS[self.lang]

        self.serial_port = None
        self.last_t = None
        self.last_y = None

        self.build_interface()
        self.build_plot()

    def translate(self, key):
        return self.translations.get(key, key)

    def switch_language(self, *args):
        self.lang = self.language_var.get()
        self.translations = LANGS[self.lang]
        self.refresh_labels()

    def refresh_labels(self):
        self.root.title(self.translate("title"))
        for widget in self.root.winfo_children():
            widget.destroy()
        self.build_interface()
        self.build_plot()
        if self.last_t is not None and self.last_y is not None:
            self.ax.plot(self.last_t, self.last_y)
            self.canvas.draw()
```

Рисунок 3.12 – *SignalGeneratorApp*

Метод *build_interface*, продемонстрований на рисунку 3.13, створює основний графічний інтерфейс, який включає вибір типу сигналу, поля для введення параметрів частоти, амплітуди та фази через текстові поля та слайдери, кнопки керування для генерації, збереження, завантаження, експорту та показу таблиці даних, налаштування *COM*-порту для вибору, підключення та надсилання даних, а також вибір мови інтерфейсу. Особливістю реалізації є синхронізація параметрів сигналу між елементами *Entry* та *Scale* через спеціальну функцію *update_entry*.

```
def build_interface(self):
    self.root.title(self.translate("title"))
    frm = ttk.Frame(self.root)
    frm.grid(row=0, column=0, sticky="nsew", padx=10, pady=10)

    ttk.Label(frm, text=self.translate("signal_type")).grid(row=0, column=0, sticky="w")
    self.signal_type = tk.StringVar(value="Синус")
    signal_options = ["Синус", "Прямокутний", "Трикутний", "Довільний"]
    ttk.Combobox(frm, textvariable=self.signal_type, values=signal_options).grid(row=0, column=1)

    entries = [("freq", "10"), ("amp", "1"), ("phase", "0")]
    self.entries = []
    for i, (key, default) in enumerate(entries):
        ttk.Label(frm, text=self.translate(key)).grid(row=i+1, column=0, sticky="w")
        entry = ttk.Entry(frm)
        entry.insert(0, default)
        entry.grid(row=i+1, column=1)
        self.entries.append(entry)

    self.sliders = []
    for i, (key, default) in enumerate([("freq", 10), ("amp", 1), ("phase", 0)]):
        def make_slider_command(index):
            return lambda val: self.update_entry(index, val)
        slider = tk.Scale(frm, from_=0, to=100, orient="horizontal",
                          label=self.translate(key), resolution=0.1,
                          command=make_slider_command(i))
        slider.set(default)
        slider.grid(row=i+1, column=2)
        self.sliders.append(slider)

    ttk.Button(frm, text=self.translate("generate"), command=self.generate_signal).grid(row=4, column=0, columnspan=3, pa
    ttk.Button(frm, text=self.translate("save"), command=self.save_signal).grid(row=5, column=0, columnspan=3)
    ttk.Button(frm, text=self.translate("load"), command=self.load_signal).grid(row=6, column=0, columnspan=3)
    ttk.Button(frm, text=self.translate("export"), command=self.export_plot).grid(row=7, column=0, columnspan=3)
    ttk.Button(frm, text=self.translate("table"), command=self.show_data_table).grid(row=8, column=0, columnspan=3)

    ttk.Label(frm, text=self.translate("com_port")).grid(row=9, column=0)
    self.port_var = tk.StringVar()
    self.port_box = ttk.Combobox(frm, textvariable=self.port_var, values=self.get_serial_ports())
    self.port_box.grid(row=9, column=1)
    ttk.Button(frm, text=self.translate("connect"), command=self.connect_serial).grid(row=10, column=0)
    ttk.Button(frm, text=self.translate("send"), command=self.send_to_device).grid(row=10, column=1)

    self.language_var = tk.StringVar(value=self.lang)
    lang_box = ttk.Combobox(frm, textvariable=self.language_var, values=list(LANGS.keys()), state="readonly")
    lang_box.grid(row=11, column=0, columnspan=2)
    lang_box.bind("<<ComboboxSelected>>", self.switch_language)
```

Рисунок 3.13 – Основний графічний інтерфейс

Метод *build_plot*, показаний на рисунку 3.14, відповідає за відображення графіка сигналу. Він створює *matplotlib*-фігуру, вставляє її у *Tkinter*-інтерфейс та додає *NavigationToolbar2Tk* для забезпечення функціональності масштабування та збереження графіка.

```
def build_plot(self):
    self.plot_frame = ttk.Frame(self.root)
    self.plot_frame.grid(row=1, column=0, sticky="nsew")

    self.fig, self.ax = plt.subplots(figsize=(6, 3))
    self.canvas = FigureCanvasTkAgg(self.fig, master=self.plot_frame)
    self.canvas.draw()
    self.canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1

    toolbar = NavigationToolbar2Tk(self.canvas, self.plot_frame)
    toolbar.update()
    toolbar.pack(side=tk.BOTTOM, fill=tk.X)
```

Рисунок 3.14 – Відображення графіку сигналу

Функція *generate_signal*, представлена на рисунку 3.15, генерує сигнал згідно з вибраним типом, включаючи синусоїдальний, прямокутний, трикутний або довільний шум. Вона зберігає згенерований сигнал у змінних *self.last_t* та *self.last_y* та оновлює графік на полотні.

```
def generate_signal(self):
    try:
        t = np.linspace(0, 1, 1000)
        f = float(self.entries[0].get())
        a = float(self.entries[1].get())
        p = float(self.entries[2].get())
        typ = self.signal_type.get()
        if typ == "Синус":
            y = a * np.sin(2 * np.pi * f * t + p)
        elif typ == "Прямокутний":
            y = a * np.sign(np.sin(2 * np.pi * f * t))
        elif typ == "Трикутний":
            y = a * 2 * np.abs(2 * (t * f - np.floor(t * f + 0.5))) - a
        else:
            y = a * (2 * np.random.rand(len(t)) - 1)
        self.last_t = t
        self.last_y = y
        self.ax.clear()
        self.ax.plot(t, y)
        self.ax.grid(True)
        self.canvas.draw()
    except Exception as e:
        messagebox.showerror("Error", str(e))
```

Рисунок 3.15 – Генерація сигналу, збереження сигналу та оновлення

Робота з *COM*-портом реалізована через декілька методів. Функція *get_serial_ports* повертає список доступних *COM*-портів, *connect_serial* відкриває з'єднання з вибраним портом з параметрами 9600 бод та таймаутом 1 секунда, а *send_to_device* формує рядок із параметрами сигналу та надсилає його через *COM*-порт, як показано на рисунку 3.16.

```
def get_serial_ports(self):
    return [port.device for port in serial.tools.list_ports.comports()]

def connect_serial(self):
    try:
        self.serial_port = serial.Serial(self.port_var.get(), 9600, timeout=1)
        messagebox.showinfo("Success", f"Connected to {self.port_var.get()}")
    except Exception as e:
        messagebox.showerror("Error", str(e))

def send_to_device(self):
    if self.serial_port and self.serial_port.is_open:
        data = f"{self.signal_type.get()},{self.entries[0].get()},{self.entries[1].get()},{self.entries[2].get()}\n"
        try:
            self.serial_port.write(data.encode('utf-8'))
            messagebox.showinfo("Sent", "Data sent to device.")
        except Exception as e:
            messagebox.showerror("Error", str(e))
    else:
        messagebox.showwarning("Warning", "COM port not connected.")
```

Рисунок 3.16 – COM-порт

Функціональність збереження та завантаження, зображена на рисунку 3.17, включає метод *save_signal*, який зберігає сигнал у *CSV*-файл з двома колонками для часу та амплітуди, та *load_signal*, який завантажує *CSV*-файл та відображає сигнал на графіку.

```
def save_signal(self):
    if self.last_t is not None and self.last_y is not None:
        filename = filedialog.asksaveasfilename(defaultextension=".csv", filetypes=[("CSV files", "*.csv")])
        if filename:
            np.savetxt(filename, np.column_stack((self.last_t, self.last_y)), delimiter=",", header="Time,Amplitude", comments='')
            messagebox.showinfo("Saved", f"Signal saved to {filename}")

def load_signal(self):
    filename = filedialog.askopenfilename(filetypes=[("CSV files", "*.csv")])
    if filename:
        try:
            data = np.loadtxt(filename, delimiter=",", skiprows=1)
            self.last_t, self.last_y = data[:, 0], data[:, 1]
            self.ax.clear()
            self.ax.plot(self.last_t, self.last_y)
            self.ax.grid(True)
            self.canvas.draw()
        except Exception as e:
            messagebox.showerror("Error", str(e))
```

Рисунок 3.17 – Збереження/Завантаження

Метод *export_plot*, показаний на рисунку 3.18, забезпечує збереження графіку сигналу у *PNG*-файл для подальшого використання в документації або презентаціях.

```
def export_plot(self):
    filename = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG Image", "*.png")])
    if filename:
        self.fig.savefig(filename)
        messagebox.showinfo("Exported", f"Plot saved as PNG to {filename}")
```

Рисунок 3.18 – Збереження графіку сигналу у *PNG*-файл.

Функція *show_data_table*, представлена на рисунку 3.19, відкриває нове вікно *Toplevel* з компонентом *Treeview*, що відображає табличні дані сигналу. Значення форматуються до чотирьох знаків після коми для зручності читання.

```
def show_data_table(self):
    if self.last_t is not None and self.last_y is not None:
        top = Toplevel(self.root)
        top.title("Signal Data")
        tree = ttk.Treeview(top, columns=("Time", "Amplitude"), show='headings')
        tree.heading("Time", text="Time")
        tree.heading("Amplitude", text="Amplitude")
        for t, y in zip(self.last_t, self.last_y):
            tree.insert("", "end", values=(f"{t:.4f}", f"{y:.4f}"))
        tree.pack(fill="both", expand=True)
```

Рисунок 3.19 – Відкриття табличних даних

Основний цикл програми, показаний на рисунку 3.20, створює екземпляр головного вікна та запускає головний цикл подій для обробки взаємодії користувача з інтерфейсом.

```
if __name__ == "__main__":
    root = tk.Tk()
    app = SignalGeneratorApp(root)
    root.mainloop()
```

Рисунок 3.20 – Запуск основного циклу програми

Програма реалізує повний функціонал генератора сигналів з графічним інтерфейсом, включаючи побудову синусоїдальних, трикутних, прямокутних та випадкових сигналів, візуалізацію та експорт результатів, обмін даними через *COM*-порт, підтримку багатомовного інтерфейсу з українською та англійською

мовами, збереження та завантаження сигналів у *CSV*-форматі, а також відображення сигналу у вигляді таблиці. Така архітектура забезпечує гнучкість використання програмного забезпечення для різноманітних завдань генерації та аналізу сигналів.

3.5 Розробка генератора сигналів на *Arduino* з використанням *AD*9833

*AD*9833, показаний на рисунку 3.21, є програмованим генератором сигналів із низьким енергоспоживанням. Дозволяє генерувати сигнали з частотою до 12.5МГц синусоїдальної, трикутної та прямокутної форми.

Програмування здійснюється з використанням трипровідного інтерфейсу *SPI* і не складає труднощів. Основними характеристиками мікросхеми є цифрове програмування частоти та фази, потужність 12.65 мВт при напрузі 3 В, діапазон вихідних частот від 0 до 12.5 МГц, роздільна здатність 28 біт що забезпечує точність 0.1 Гц при частоті опорного сигналу 25 МГц. Мікросхема може генерувати синусоїдальні, трикутні та прямокутні вихідні коливання, працює від напруги живлення від 2.3 до 5.5 В, використовує трипровідний інтерфейс *SPI*, має розширений температурний діапазон від -40 до +105°*C* та опцію зниженого споживання енергії.

Рисунок 3.21 – *AD*9833

Більш детальну інформацію можна знайти в датасіті. У характеристиках також заявлено, що мікросхема не вимагає зовнішніх компонентів, але тут виробник лукавить: обв'язка та джерело опорної частоти все ж таки потрібні. На Алі продаються модулі *AD*9833 з необхідною обв'язкою та кварцовим генератором на 25 МГц, саме з таким модулем проводяться експерименти. Цей модуль має наступні висновки: *VCC* є плюсом живлення для цифрових та аналогових ланцюгів генератора, *DGND* служить цифровою землею, *SDATA* є входом даних інтерфейсу *SPI* де передача здійснюється 16-бітовими словами, *SCLK* є входом тактового сигналу *SPI* з використанням другого режиму роботи коли *CPOL* дорівнює 1 та *CPHA* дорівнює 0, *FSYNC* відповідає за вибір мікросхеми і перед початком передачі даних повинен бути встановлений 0 а по завершенні 1, *AGND* є аналоговою землею, *OUT* служить виходом генератора.

Для підключення цього модуля до Ардуїно необхідно ознайомитися з його функціональною схемою, показаною на рисунку 3.22. *AD*9833 складається з наступних основних частин: два регістри вибору частоти, акумулятор фази, два регістри вибору фази і суматор зміщення фази, які разом складають генератор з цифровим управлінням *NCO*, *SIN ROM* для перетворення інформації про фазу в амплітуду та 10-розрядний цифро-аналоговий перетворювач. Зі схеми видно, що дані з інтерфейсу *SPI* передаються в регістр, що управляє, регістри вибору фази і частоти. Саме вони визначають сигнал на виході генератора. І програмування генератора зводиться до зміни вмісту зазначених регістрів.

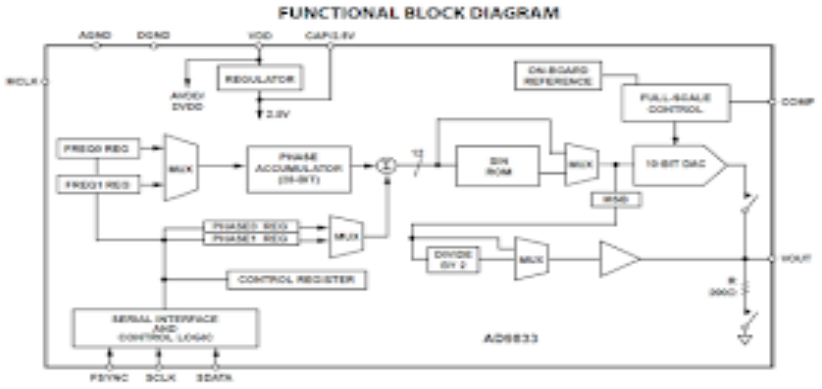
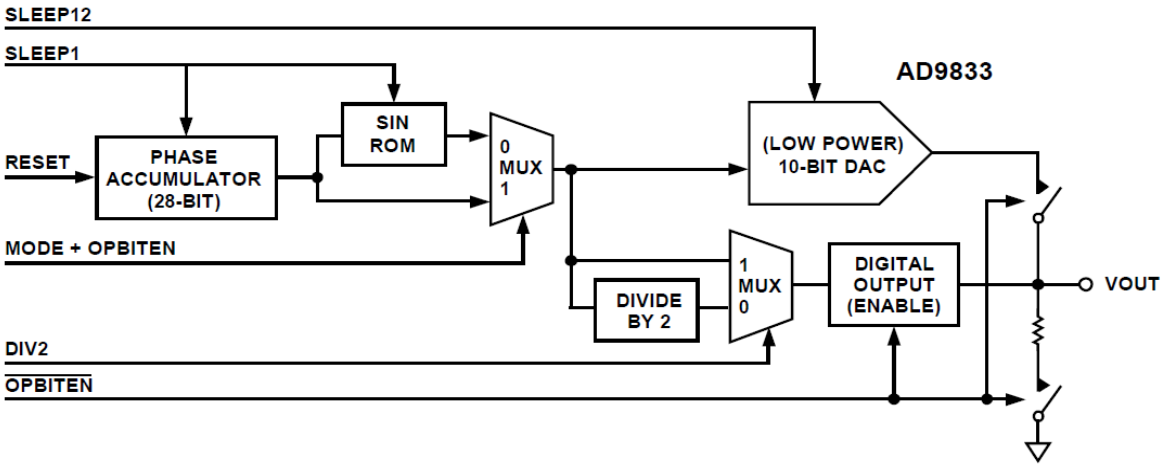


Рисунок 3.22 – Функціональна схема модуля

Керівний регістр є 16-розрядним регістром, керуючим роботою генератора. Детальний опис його бітів представлений нижче. Рисунок 3.23 з даташиту також наочно демонструє їхнє призначення. Біти 15 та 14 *DB*15 і *DB*14 повинні бути встановлені в 0, щоб *AD*9833 зрозумів, що прийняте *SPI* 16-бітне слово містить нове значення для керуючого регістру. Біт 13 *B*28 використовується для управління оновленням регістрів частоти. Оскільки регістри частоти *AD*9833 мають розрядність 28 біт, для зміни їх вмісту потрібна передача двох 16-бітних слів. *B*28 дорівнює 1 говорить про те, що необхідно оновити регістр частоти і його нове значення буде передано двома послідовними записами. Перший запис містить 14 молодших біт, другий 14 старших біт. *B*28 дорівнює 0 дозволяє оновити окремо старшу або молодшу частину регістру.



DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	B28	HLB	FSELECT	PSELECT	0	RESET	SLEEP1	SLEEP12	OPBITEN	0	DIV2	0	MODE	0

Рисунок 3.23 – 16-розрядний регістр, керуючий роботою генератора.

Біт 12 *HLB* визначає, яка частина регістру частоти буде перезаписана, використовується за *B*28 дорівнює 0. *HLB* дорівнює 1 дозволяє оновити старші 14 біт регістра частоти, *HLB* дорівнює 0 дозволяє оновити молодші 14 біт регістру частоти. Біт 11 *FSELECT* визначає, який із регістрів використовується в акумуляторі фази *FREQ*0 або *FREQ*1. Біт 10 *PSELECT* визначає, дані якого з регістрів *PHASE*0 або *PHASE*1 додаються до виходу фази акумулятора. Біт 9 зарезервований і має бути встановлений в 0. Біт 8 *RESET* при встановленні в 1

скидає внутрішні регістри генератора в 0, але скидання не зачіпає регістри управління, частоти та фази. Біт 7 *SLEEP*1 при встановленні в 1 забороняє внутрішнє тактування, припиняє роботу *NCO* та залишає вихід генератора у своєму поточному стані. Біт 6 *SLEEP*12 при встановленні в 1 відключає внутрішній ЦАП.

Біт 5 *OPBITEN* разом з бітом *MODE* керують виходом генератора. При *OPBITEN* дорівнює 1 внутрішній ЦАП відключається від виходу *VOUT* і для генерації вихідного сигналу використовується значення старшого біта з входу ЦАП, що дозволяє отримати на виході генератора прямокутні імпульси. Біт 4 зарезервований і має бути встановлений в 0. Біт 3 *DIV*2 використовується в парі зі значенням *OPBITEN* дорівнює 1. При *DIV*2 дорівнює 1 значення старшого біта даних з входу ЦАП подається безпосередньо на вихід *VOUT*. *DIV*2 дорівнює 0 дозволяє задіяти дільник частоти та зменшити частоту вихідного сигналу вдвічі. Біт 2 зарезервований і має бути встановлений в 0. Біт 1 *MODE* разом із бітом *OPBITEN* управляють виходом генератора. При *OPBITEN* дорівнює 1 біт *MODE* має бути встановлений 0. Значення *MODE* дорівнює 0 дозволяє отримати на виході генератора синусоїдальний сигнал, у разі *MODE* дорівнює 1 на виході буде трикутний сигнал. Біт 0 зарезервований і має бути встановлений в 0.

Для розуміння призначення бітів *OPBITEN*, *MODE* та *DIV*2 наведено їх допустимі комбінації та форму результуючих сигналів на виході. При *OPBITEN* дорівнює 0 та *MODE* дорівнює 0 незалежно від значення *DIV*2 на виході буде синусоїдальний сигнал. При *OPBITEN* дорівнює 0 та *MODE* дорівнює 1 незалежно від значення *DIV*2 на виході буде трикутний сигнал. При *OPBITEN* дорівнює 1, *MODE* дорівнює 0 та *DIV*2 дорівнює 0 на виході буде прямокутний сигнал із частотою *F*/2. При *OPBITEN* дорівнює 1, *MODE* дорівнює 0 та *DIV*2 дорівнює 1 на виході буде прямокутний сигнал із частотою *F*. Комбінація *OPBITEN* дорівнює 1 та *MODE* дорівнює 1 незалежно від значення *DIV*2 зарезервована.

Генератор *AD*9833 має 2 регістри частоти та 2 регістри фази розрядністю 28 біт і 12 біт відповідно. Вибір активного регістру частоти здійснюється установкою біта *FSELECT*: при *FSELECT* дорівнює 0 активним є *FREQ*0, при *FSELECT* дорівнює 1 активний регістр *FREQ*1. Результуюча частота на виході генератора

визначається за формулою: частота дорівнює добутку опорної частоти *MCLK* поділеної на 2 в степені 28 та значення *FREQREG* завантаженого в активний регістр частоти. Таким чином, якщо необхідно отримати на виході генератора сигнал із частотою 400Гц при опорній частоті 25МГц, активний регістр має бути завантажене значення: *FREQREG* дорівнює добутку вихідної частоти та 2 в степені 28 поділеному на опорну частоту, що становить приблизно 4295 для частоти 400Гц.

Для завантаження значення *FREQREG* в регістр частоти необхідно старші біти значення, що передається по *SPI*, встановити в 01 для завантаження в *FREQ*0 або 10 для завантаження в *FREQ*1. Спілкування з *AD*9833 здійснюється за *SPI* 16-бітовими словами. Фаза вихідного сигналу визначається за формулою: фаза дорівнює добутку 2π поділеному на 2 в степені 12 та значення *PHASEREG*. Відповідно, значення для регістра фази обчислюється за формулою: *PHASEREG* дорівнює добутку фази та 2 в степені 12 поділеному на 2π. У наведених формулах *PHASEREG* є значенням активного регістру фази. Вибір активного регістру здійснюється установкою біта *PSELECT*: при *PSELECT* дорівнює 0 активним є *PHASE*0, при *PSELECT* дорівнює 1 активний регістр *PHASE*1. При записі нового значення в регістр фази старші біти повинні бути встановлені в 11, а вибір регістра здійснюється установкою біта 13: при нульовому його значенні буде оновлений регістр *PHASE*0, при встановленні зазначеного біта в 1 буде оновлено регістр *PHASE*1. 12й біт не використовується, а біти з 0 по 11 містять значення для регістра фази. Розрядність регістра частоти 28 біт при опорній частоті 25МГц забезпечує крок 0.1Гц для встановлення частоти сигналу на виході. А 12-бітовий регістр фази забезпечує роздільну здатність 2π/4096.

Тепер можна написати першу програму для *AD*9833. Схема підключення модуля *AD*9833 до Ардуїно наведена на рисунку 3.24. З підключенням все просто: спілкування з модулем відбувається за інтерфейсом *SPI*, для якого на Ардуїно відведені наступні піни. *D*10 є *SS* (*Slave Select* – вибір веденого), до нього підключається висновок *FSYNC* модуля. *D*11 є *MOSI* (*Master Out Slave In* – вихід ведучого, вхід веденого), до нього підключається висновок *SDATA*. *D*13 є *SCK* (*Serial Clock* – Тактовий сигнал), до нього підключається висновок *SCLK*.

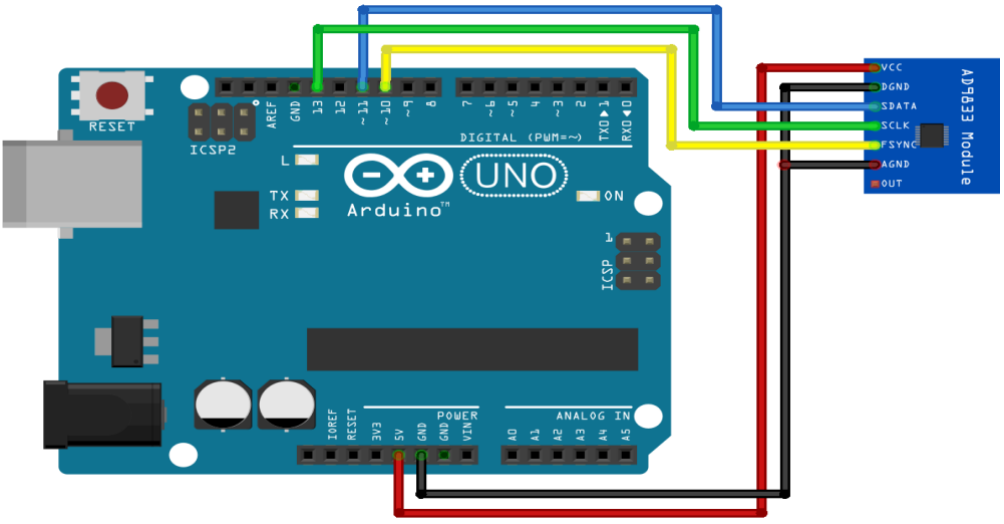


Рисунок 3.24 – Тестова програма для *AD*9833 на Ардуїно

У скетчі виконуються наступні дії. При першому виклику функції *WriteAD*9833 проводиться установка регістра, що управляє: біт *RESET* встановлюється в 1 для виконання скидання, біт *DB*28 встановлюється 1 для перезапису всього вмісту регістра частоти, біти *FSELECT* та *PSELECT* містять 0, тому для генерації вихідного сигналу будуть використовуватися регістри *FREQ*0 та *PHASE*0. Наступні два виклики передають значення 4295 в регістр частоти *FREQ*0. Це значення міститься в 14 молодших розрядах, тому в старші розряди регістру записуються нулі. Зсув по фазі не потрібен, тому в регістр *PHASE*0 записується значення 0. Останнім викликом *WriteAD*9833 у процедурі *setup* знімається біт *RESET*, дозволяючи цим роботу генератора. Результуючий сигнал надходить на висновок *VOUT*. Наступні виклики *WriteAD*9833 в функції *loop* оновлюють вміст керуючого регістру, перебираючи комбінації бітів *MODE*, *OPBITEN* і *DIV*2 для генерації сигналу синусоїдальної, трикутної і прямокутної форм.

Вихідний сигнал генератора у віртуальному осцилографі показано на рисунках 3.25-3.28. Синусоїдальний сигнал показано на рисунку 3.25, трикутний сигнал показано на рисунку 3.26, прямокутний сигнал *MSB*/2 показано на рисунку 3.27, прямокутний сигнал *MSB* показано на рисунку 3.28. При генерації синусоїдальних та трикутних імпульсів, коли сигнал знімається з виходу ЦАП, його амплітуда змінюється в діапазоні 38мВ...0,65В. При генерації імпульсів

прямокутної форми маємо справу зі звичайним цифровим сигналом із відповідними рівнями напруги. Так, у останніх двох осцилограмах логічної одиниці відповідає напруга приблизно 4,5В.

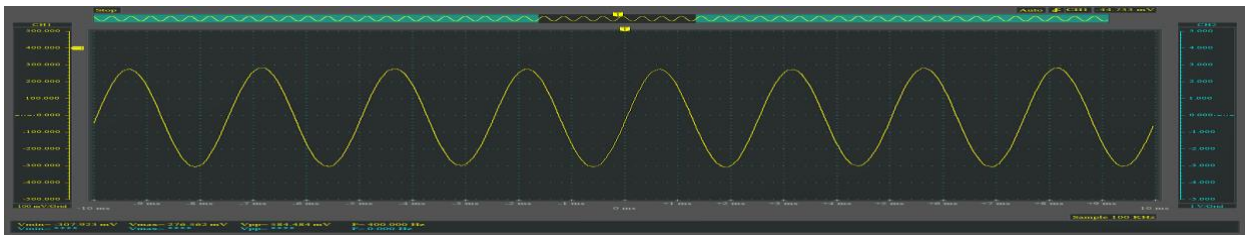


Рисунок 3.25 – Синусоїдальний сигнал

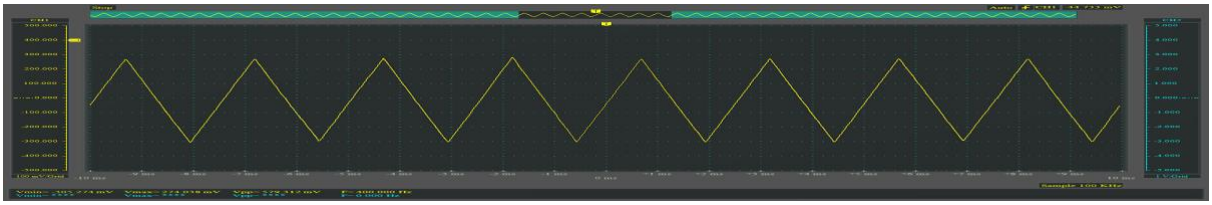


Рисунок 3.26 – Трикутний сигнал

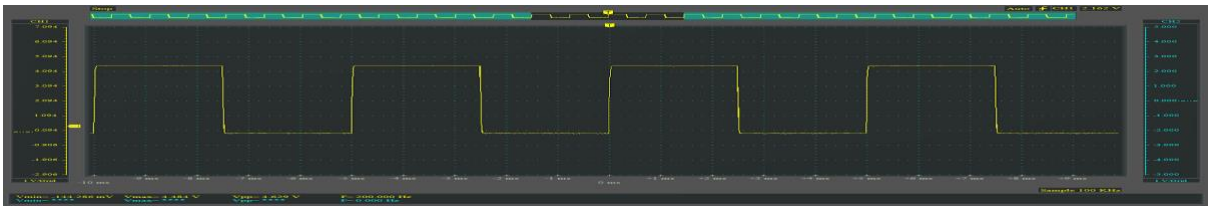


Рисунок 3.27 – Прямокутний (*MSB*/2)

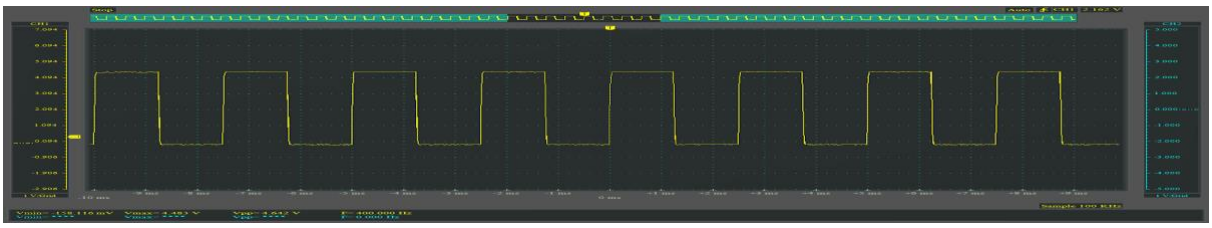


Рисунок 3.28 – Прямокутний (*MSB*)

Розібравшись з керуванням *AD*9833 можна розпочинати створення генератора з інтерфейсом управління та індикацією. Для цього в схему додається енкодер обертання та рідкокристалічний дисплей, як показано на рисунку 3.29. При включенні живлення *AD*9833 налаштовується на генерацію синусоїдального

сигналу частотою 100Гц, відповідна інформація відображається на дисплеї. Повертаючи ручку енкодера можна змінювати його частоту, а при натисканні викликається меню. У меню доступні наступні опції: встановлення частоти з можливістю встановити довільне значення від 1 до 12,5МГц, встановлення фази в діапазоні 0-360 градусів, вибір форми сигналу, вибір значення на яке змінюється частота при обертанні ручки енкодера. Залишається тільки помістити всі компоненти у відповідний корпус та вийде закінчений пристрій.

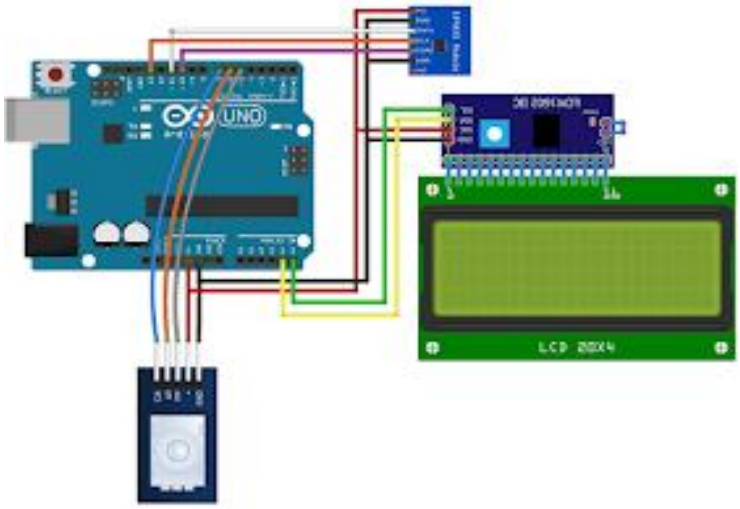


Рисунок 3.29 – Генератор на *AD*9833 з дисплеєм та енкодером

ВИСНОВКИ

Кваліфікаційна робота присвячена розробці програмного забезпечення – віконного інтерфейсу керування генератором сигналів довільної форми (ГДС). Метою було створення інтуїтивно зрозумілого та функціонального інструменту для взаємодії користувача з апаратною частиною генератора, зокрема на базі мікросхеми *AD*9833, та забезпечення можливостей генерації, візуалізації, збереження та передачі сигналів.

У ході виконання роботи було проаналізовано актуальність створення зручних програмних засобів для керування ГДС, оскільки ефективність використання таких приладів значною мірою залежить від якості інтерфейсу.

У даній роботі було спроєктовано та реалізовано віконний застосунок з модульною архітектурою на мові програмування *Python*.

Розроблений інтерфейс дозволяє користувачеві обирати тип сигналу (синусоїдальний, прямокутний, трикутний, довільний шум), налаштовувати його параметри (частоту, амплітуду, фазу) як через поля вводу, так і за допомогою інтерактивних повзунків. Реалізовано функціонал візуалізації сигналу в реальному часі, збереження даних сигналу у форматі *CSV*, експорту графічного представлення у формат *PNG*, а також відображення даних у табличному вигляді. Забезпечено можливість передачі параметрів згенерованого сигналу на зовнішній пристрій (на прикладі модуля *AD*9833 на базі *Arduino*) через *COM*-порт.

Таким чином, створене програмне забезпечення є комплексним рішенням, що спрощує процес роботи з генераторами довільної форми сигналів, підвищує ефективність керування та мінімізує ймовірність помилок при генерації. Практичне значення роботи полягає у створенні прикладного програмного продукту, який може бути інтегрований у реальні вимірювальні системи, що використовуються в освітніх, наукових або лабораторних цілях для тестування та налагодження електронних систем, а також для демонстрації принципів роботи ГДС.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *AD*9833 *Low Power*, 10 *MSPS*, *Programmable Waveform Generator – Analog Devices*. [Електронний ресурс] – Режим доступу: *https*://*www.analog.com*/*media*/*en*/*technical-documentation*/*data-sheets*/*AD*9833.*pdf* (дата звернення: 12.05.2025)

2. *Python Tkinter* – офіційна документація інтерфейсу *GUI*. [Електронний ресурс] – Режим доступу: *https*://*docs.python.org*/3/*library*/*tkinter.html* (дата звернення: 08.05.2025)

3. *NumPy* – офіційна документація бібліотеки наукових обчислень. [Електронний ресурс] – Режим доступу: *https*://*numpy.org*/*doc*/*stable*/ (дата звернення: 23.05.2025)

4. *Matplotlib* – керівництво з візуалізації даних *Python*. [Електронний ресурс] – Режим доступу: *https*://*matplotlib.org*/*stable*/*tutorials*/*index.html* (дата звернення: 17.05.2025)

5. *PySerial* – документація бібліотеки серійного зв'язку *Python*. [Електронний ресурс] – Режим доступу: *https*://*pyserial.readthedocs.io*/*en*/*latest* (дата звернення: 04.05.2025)

6. *Arduino SPI Reference* – офіційна документація протоколу *SPI*. [Електронний ресурс] – Режим доступу: *https*://*www.arduino.cc*/*reference*/*en*/*language*/*communication*/*spi* (дата звернення: 29.05.2025)

7. *Direct Digital Synthesizer – Wikipedia*. [Електронний ресурс] – Режим доступу: *https*://*en.wikipedia.org*/*wiki*/*Direct_digital_synthesizer* (дата звернення: 15.05.2025)

8. *AD*9833 *Waveform Generator Arduino Tutorial – Last Minute Engineers*. [Електронний ресурс] – Режим доступу: *https*://*lastminuteengineers.com*/*ad*9833-*waveform-generator-arduino-tutorial* (дата звернення: 21.05.2025)

9. *Interfacing AD*9833 *Function Generator with Arduino – Electronics Hub*. [Електронний ресурс] – Режим доступу: *https*://*www.electronicshub.org*/*interfacing-ad*9833-*function-generator-with-arduino* (дата звернення: 06.05.2025)

10. *Tektronix Arbitrary Waveform Generators* – офіційний каталог продукції. [Електронний ресурс] – Режим доступу: *https*://*www.tek.com*/*en*/*products*/*arbitrary-waveform-generators* (дата звернення: 28.05.2025)

11. *Keysight Waveform Generators* – офіційна документація приладів. [Електронний ресурс] – Режим доступу: *https*://*www.keysight.com*/*us*/*en*/*products*/*source-measure-units*/*waveform-generators.html* (дата звернення: 11.05.2025)

12. *Siglent Waveform Generators* – каталог генераторів сигналів. [Електронний ресурс] – Режим доступу: *https*://*siglent.eu*/*products-category*/*waveform-generators* (дата звернення: 26.05.2025)

13. *Introduction to Direct Digital Synthesis – All About Circuits*. [Електронний ресурс] – Режим доступу: *https*://*www.allaboutcircuits.com*/*technical-articles*/*introduction-to-direct-digital-synthesis* (дата звернення: 19.05.2025)

14. *Python CSV* – офіційна документація роботи з *CSV* файлами. [Електронний ресурс] – Режим доступу: *https*://*docs.python.org*/3/*library*/*csv.html* (дата звернення: 03.05.2025)

15. *PNG Specification – World Wide Web Consortium*. [Електронний ресурс] – Режим доступу: *https*://*www.w*3.*org*/*Graphics*/*PNG* (дата звернення: 14.05.2025)

16. *RIGOL Waveform Generators* – офіційний каталог приладів. [Електронний ресурс] – Режим доступу: *https*://*www.rigolna.com*/*products*/*waveform-generators* (дата звернення: 07.05.2025)

17. *MATLAB Arbitrary Waveform Generation* – офіційне керівництво. [Електронний ресурс] – Режим доступу: *https*://*www.mathworks.com*/*help*/*signal*/*ug*/*arbitrary-waveform-generation.html* (дата звернення: 30.05.2025)

ДОДАТОК А

Графічний генератор сигналів із використанням *Tkinter*

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog, Toplevel
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,
NavigationToolbar2Tk
import serial
import serial.tools.list_ports
import csv

LANGS = {
    "uk": {
        "title": "Генератор сигналів",
        "signal_type": "Тип сигналу:",
        "freq": "Частота (Hz):",
        "amp": "Амплітуда:",
        "phase": "Фаза (рад):",
        "generate": "Згенерувати сигнал",
        "com_port": "COM-порт:",
        "connect": "Підключити",
        "send": "Надіслати",
        "save": "Зберегти сигнал",
        "load": "Завантажити сигнал",
        "export": "Експорт у PNG",
        "table": "Показати дані",
        "language": "Мова"
    },
    "en": {
        "title": "Signal Generator",
        "signal_type": "Signal Type:",
        "freq": "Frequency (Hz):",
        "amp": "Amplitude:",
        "phase": "Phase (rad):",
        "generate": "Generate Signal",
        "com_port": "COM Port:",
        "connect": "Connect",
        "send": "Send",
        "save": "Save Signal",
        "load": "Load Signal",
        "export": "Export to PNG",
        "table": "Show Data",
        "language": "Language"
    }
}

class SignalGeneratorApp:
    def __init__(self, root):
        self.root = root
        self.root.geometry("800x600")
```

```
        self.root.resizable(True, True)
        self.style = ttk.Style()
        self.style.theme_use("clam")
        self.lang = "uk"
        self.translations = LANGS[self.lang]

        self.serial_port = None
        self.last_t = None
        self.last_y = None

        self.build_interface()
        self.build_plot()

    def translate(self, key):
        return self.translations.get(key, key)

    def switch_language(self, *args):
        self.lang = self.language_var.get()
        self.translations = LANGS[self.lang]
        self.refresh_labels()

    def refresh_labels(self):
        self.root.title(self.translate("title"))
        for widget in self.root.winfo_children():
            widget.destroy()
        self.build_interface()
        self.build_plot()
        if self.last_t is not None and self.last_y is not None:
            self.ax.plot(self.last_t, self.last_y)
            self.canvas.draw()

    def build_interface(self):
        self.root.title(self.translate("title"))
        frm = ttk.Frame(self.root)
        frm.grid(row=0, column=0, sticky="nsew", padx=10, pady=10)

        ttk.Label(frm,
text=self.translate("signal_type")).grid(row=0, column=0, sticky="w")
        self.signal_type = tk.StringVar(value="Синус")
        signal_options = ["Синус", "Прямокутний", "Трикутний",
"Довільний"]
        ttk.Combobox(frm, textvariable=self.signal_type,
values=signal_options).grid(row=0, column=1)

        entries = [("freq", "10"), ("amp", "1"), ("phase", "0")]
        self.entries = []
        for i, (key, default) in enumerate(entries):
            ttk.Label(frm, text=self.translate(key)).grid(row=i+1,
column=0, sticky="w")
            entry = ttk.Entry(frm)
            entry.insert(0, default)
            entry.grid(row=i+1, column=1)
            self.entries.append(entry)
```

```
        self.sliders = []
        for i, (key, default) in enumerate([("freq", 10), ("amp", 1),
("phase", 0)]):
            def make_slider_command(index):
                return lambda val: self.update_entry(index, val)
            slider     =     tk.Scale(frm,     from_=0,     to=100,
orient="horizontal",
                                    label=self.translate(key),
resolution=0.1,
                                    command=make_slider_command(i))
            slider.set(default)
            slider.grid(row=i+1, column=2)
            self.sliders.append(slider)

        ttk.Button(frm,              text=self.translate("generate"),
command=self.generate_signal).grid(row=4,   column=0,   columnspan=3,
pady=5)
        ttk.Button(frm,                  text=self.translate("save"),
command=self.save_signal).grid(row=5, column=0, columnspan=3)
        ttk.Button(frm,                  text=self.translate("load"),
command=self.load_signal).grid(row=6, column=0, columnspan=3)
        ttk.Button(frm,                text=self.translate("export"),
command=self.export_plot).grid(row=7, column=0, columnspan=3)
        ttk.Button(frm,                 text=self.translate("table"),
command=self.show_data_table).grid(row=8, column=0, columnspan=3)

        ttk.Label(frm,   text=self.translate("com_port")).grid(row=9,
column=0)
        self.port_var = tk.StringVar()
        self.port_box = ttk.Combobox(frm, textvariable=self.port_var,
values=self.get_serial_ports())
        self.port_box.grid(row=9, column=1)
        ttk.Button(frm,               text=self.translate("connect"),
command=self.connect_serial).grid(row=10, column=0)
        ttk.Button(frm,                  text=self.translate("send"),
command=self.send_to_device).grid(row=10, column=1)

        self.language_var = tk.StringVar(value=self.lang)
        lang_box = ttk.Combobox(frm, textvariable=self.language_var,
values=list(LANGS.keys()), state="readonly")
        lang_box.grid(row=11, column=0, columnspan=2)
        lang_box.bind("<<ComboboxSelected>>", self.switch_language)

    def build_plot(self):
        self.plot_frame = ttk.Frame(self.root)
        self.plot_frame.grid(row=1, column=0, sticky="nsew")

        self.fig, self.ax = plt.subplots(figsize=(6, 3))
        self.canvas          =          FigureCanvasTkAgg(self.fig,
master=self.plot_frame)
        self.canvas.draw()
```

```
        self.canvas.get_tk_widget().pack(side=tk.TOP,   fill=tk.BOTH,
expand=1)

        toolbar = NavigationToolbar2Tk(self.canvas, self.plot_frame)
        toolbar.update()
        toolbar.pack(side=tk.BOTTOM, fill=tk.X)

    def update_entry(self, index, val):
        self.entries[index].delete(0, tk.END)
        self.entries[index].insert(0, val)

    def generate_signal(self):
        try:
            t = np.linspace(0, 1, 1000)
            f = float(self.entries[0].get())
            a = float(self.entries[1].get())
            p = float(self.entries[2].get())
            typ = self.signal_type.get()
            if typ == "Синус":
                y = a * np.sin(2 * np.pi * f * t + p)
            elif typ == "Прямокутний":
                y = a * np.sign(np.sin(2 * np.pi * f * t))
            elif typ == "Трикутний":
                y = a * 2 * np.abs(2 * (t * f - np.floor(t * f + 0.5)))
- a
            else:
                y = a * (2 * np.random.rand(len(t)) - 1)
            self.last_t = t
            self.last_y = y
            self.ax.clear()
            self.ax.plot(t, y)
            self.ax.grid(True)
            self.canvas.draw()
        except Exception as e:
            messagebox.showerror("Error", str(e))

    def get_serial_ports(self):
        return        [port.device        for        port        in
serial.tools.list_ports.comports()]

    def connect_serial(self):
        try:
            self.serial_port   =   serial.Serial(self.port_var.get(),
9600, timeout=1)
            messagebox.showinfo("Success",       f"Connected       to
{self.port_var.get()}")
        except Exception as e:
            messagebox.showerror("Error", str(e))

    def send_to_device(self):
        if self.serial_port and self.serial_port.is_open:
```

```
            data = f"{self.signal_type.get()},{self.entries[0].get()},{self.entries[1].get()},{self.entries[2].get()}\n"
            try:
                self.serial_port.write(data.encode('utf-8'))
                messagebox.showinfo("Sent", "Data sent to device.")
            except Exception as e:
                messagebox.showerror("Error", str(e))
        else:
            messagebox.showwarning("Warning", "COM port not connected.")

    def save_signal(self):
        if self.last_t is not None and self.last_y is not None:
            filename = filedialog.asksaveasfilename(defaultextension=".csv", filetypes=[("CSV files", "*.csv")])
            if filename:
                np.savetxt(filename, np.column_stack((self.last_t, self.last_y)), delimiter=",", header="Time,Amplitude", comments='')
                messagebox.showinfo("Saved", f"Signal saved to {filename}")

    def load_signal(self):
        filename = filedialog.askopenfilename(filetypes=[("CSV files", "*.csv")])
        if filename:
            try:
                data = np.loadtxt(filename, delimiter=",", skiprows=1)
                self.last_t, self.last_y = data[:, 0], data[:, 1]
                self.ax.clear()
                self.ax.plot(self.last_t, self.last_y)
                self.ax.grid(True)
                self.canvas.draw()
            except Exception as e:
                messagebox.showerror("Error", str(e))

    def export_plot(self):
        filename = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG Image", "*.png")])
        if filename:
            self.fig.savefig(filename)
            messagebox.showinfo("Exported", f"Plot saved as PNG to {filename}")

    def show_data_table(self):
        if self.last_t is not None and self.last_y is not None:
            top = Toplevel(self.root)
            top.title("Signal Data")
            tree = ttk.Treeview(top, columns=("Time", "Amplitude"), show='headings')
```

```
            tree.heading("Time", text="Time")
            tree.heading("Amplitude", text="Amplitude")
            for t, y in zip(self.last_t, self.last_y):
                tree.insert("",       "end",       values=(f"{t:.4f}",
f"{y:.4f}"))
            tree.pack(fill="both", expand=True)

if __name__ == "__main__":
    root = tk.Tk()
    app = SignalGeneratorApp(root)
    root.mainloop()
```

ДОДАТОК Б

Програма для керування генератором сигналів *AD*9833 на *Arduino*

```
#include <SPI.h>
void setup() {
  SPI.begin();
  WriteAD9833(0x2100); //0010 0001 0000 0000 - Reset + DB28
  WriteAD9833(0x50C7); //0101 0000 1100 0111 - Freq0 LSB (4295)
  WriteAD9833(0x4000); //0100 0000 0000 0000 - Freq0 MSB (0)
  WriteAD9833(0xC000); //1100 0000 0000 0000 - Phase0 (0)
  WriteAD9833(0x2000); //0010 0000 0000 0000 - Exit Reset
}

void WriteAD9833(uint16_t Data){
  SPI.beginTransaction(SPISettings(SPI_CLOCK_DIV2, MSBFIRST,
SPI_MODE2));
  digitalWrite(SS, LOW);
  delayMicroseconds(1);
  SPI.transfer16(Data);
  digitalWrite(SS, HIGH);
  SPI.endTransaction();
}

void loop() {
  WriteAD9833(0x2000); //0010 0000 0000 0000 - Синусоїдальний сигнал
  delay(5000);
  WriteAD9833(0x2002); //0010 0000 0000 0010 - MODE=1 - Трикутний
  delay(5000);
  WriteAD9833(0x2020); //0010 0000 0010 0000 - OPBITEN=1 -
Прямокутний (MSB/2)
  delay(5000);
  WriteAD9833(0x2028); //0010 0000 0010 1000 - OPBITEN=1, DIV2=1 -
Прямокутний (MSB)
  delay(5000);
 }
```

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

ВІДГУК
керівника кваліфікаційної роботи

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Артем КАСЯНЕНКО
(ім'я, прізвище)

1. *Кваліфікаційна робота на тему «Проєктування віконного інтерфейсу керування генератора сигналів довільної форми» виконана в повному обсязі.*
2. *Метою кваліфікаційної роботи є розробка програми керування та формування сигналів довільної форми для універсального генератора.*
3. *Кваліфікаційна робота відповідає темі, затвердженій наказом начальника коледжу.*
4. *Кваліфікаційна робота виконана здобувачем освіти самостійно.*
5. *Здобувач освіти виконав аналіз літературних джерел та наявної в доступі інформації про генератори довільних сигналів та сформував технічне завдання на проектування, створив відповідне програмне забезпечення та виконав тестування його роботи.*
6. *Артем Касяненко показав достатній рівень дотримання вимог державних стандартів при виконанні кваліфікаційної роботи в цілому та оформленні пояснювальної записки .*
7. *Рівень виконаної кваліфікаційної роботи заслуговує оцінку «відмінно», відповідає набутих випускником знань, умінь та навичок, вимогам освітньої характеристики фахівця і можливість присвоєння йому кваліфікації фахівця освітнього ступеня «фаховий молодший бакалавр» спеціальності 123 «Комп'ютерна інженерія».*

Керівник кваліфікаційної роботи

«____»_____2025 р. _____ Артем КУТІН
(підпис) (ім'я, прізвище)

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

РЕЦЕНЗІЯ
на кваліфікаційну роботу

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Артем КАСЯНЕНКО
(ім'я, прізвище)

1. Актуальність теми: Обрана тема кваліфікаційної роботи «Проєктування віконного інтерфейсу керування генератора сигналів довільної форми» є актуальною.

2. Кваліфікаційна робота відповідає темі, затвердженій наказом.

3. Завдання на виконання кваліфікаційної роботи виконано у повному обсязі.

4. В результаті виконання кваліфікаційної роботи було реалізовано програмне забезпечення що синтезує відліки сигналу вибраної оператором форми та передає їх на плату керування цифро-аналогового перетворювача для формування сигналу. Розроблені інструменти програми дозволяють вибрати як стандартну (із заздалегідь визначеними параметрами) так і завантажити довільну власну форму сигналу.

5. Якість виконання пояснювальної записки та ілюстративного (графічного) матеріалу відповідає вимогам Державних стандартів.

6. Здобувачем освіти було створено програму як персонального комп'ютера, так і для контролера керування цифро-аналоговим перетворювачем.

7. Кваліфікаційна робота заслуговує оцінку «добре».

Рецензент _____
(науковий ступінь, посада)

«____»_____ 2025 р. _____
Артем КУТІН
(підпис) (ім'я, прізвище)

З рецензією ознайомлений _____ Артем КАСЯНЕНКО
(підпис) (ім'я, прізвище)