Дніпровський національний університет імені Олеся Гончара
Міністерство освіти та науки України

Вінницький національний технічний університет
Міністерство освіти та науки України

Кваліфікаційна наукова
праця на правах рукопису

ЧЕРНЕТЧЕНКО ДМИТРО ВОЛОДИМИРОВИЧ

УДК 616-71:004.032.26

ДИСЕРТАЦІЯ

МЕТОД ТА АПАРАТНО-ПРОГРАМНИЙ ЗАСІБ ОБРОБКИ ЕЛЕКТРОКАРДІОГРАФІЧНИХ СИГНАЛІВ ЗА ДОПОМОГОЮ ШТУЧНИХ МУЛЬТИСТАБІЛЬНИХ НЕЙРОННИХ МЕРЕЖ

05.11.17 – біологічні та медичні прилади і системи
технічні науки

Подається на здобуття наукового ступеня кандидата технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
Д.В. Чернетченко

Науковий керівник Сніжко Євген Матвійович, кандидат технічних наук, доцент

Дніпро – 2019

АНОТАЦІЯ

Чернетченко Д.В. Метод та апаратно-програмний засіб обробки електрокардіографічних сигналів за допомогою штучних мультистабільних нейронних мереж. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.11.17 “Біологічні та медичні прилади і системи” – Дніпровський національний університет імені Олеся Гончара, МОН України, Дніпро. – Вінницький національний технічний університет, МОН України, Вінниця, 2019.

Серцево-судинні захворювання є глобальною проблемою або "глобальною епідемією" (дані за 2018 рік), захворюваннями, які викликають передчасну смертність майже 17 мільйонів людей щороку. Всесвітньою організацією охорони здоров'я (ВООЗ) визначено, що люди, які страждають від серцево-судинних захворювань або чутливі до ризику виникнення таких захворювань, потребують підвищеної уваги, для раннього виявлення передвісників хвороби (кардіокатастроф) та надання своєчасної кваліфікованої допомоги. Це, в свою чергу, зумовлює необхідність розроблення інструментальних засобів точної та швидкої реєстрації та автоматизованого аналізу електрокардіографічних (ЕКГ) сигналів в режимі реального часу. В той же час, все частіше для аналізу в реальному часі ЕКГ-сигналів, починають застосовувати штучні нейронні мережі, які виконують завдання класифікації і є основою для аналізу ЕКГ, для виявлення порушень серцевого ритму, ішемії міокарда, стенокардії та інших хронічних змін. Розробники медичної апаратури зацікавлені у нейронних структурах, які здатні реорганізовуватися у реальному часі без попереднього навчання. З огляду на стрімке розширення сфери застосування штучних нейронних мереж, дослідники звертають все більшу увагу на процеси у біологічних нейронах та нейронних структурах, які свого часу стали прототипами для цієї технології. Штучні моделі окремих нейронів, що здатні відтворювати спайкову (або імпульсну) поведінку біологічних аналогів, як правило, називають

спайковими нейронами, нейронні мережі – спайковими нейронними мережами. Спайкові штучні нейронні мережі здатні суттєво підвищити якість моніторингового контролю за станом пацієнта, ефективність автоматизованої діагностики, завдяки здатності виконанувати задачі класифікації та розпізнавання. Виявлення комплексу P-QRS-T в ЕКГ-сигналі забезпечує фундаментальну можливість для виявлення всіх головних компонентів сигналу і подальшого автоматизованого аналізу та діагностування захворювань. Однак певні специфічні характеристики ЕКГ ускладнюють програмне виявлення форми ЕКГ: по-перше, це особливість морфології сигналів ЕКГ, що змінюється від людини до людини; по-друге, ЕКГ-сигнал досить часто має компоненти шуму різного походження. На сьогодні створення такої обчислювальної системи із низьким рівнем енергоспоживання, високою обчислюваною потужністю та здатністю точно виконувати детектування всіх головних ознак P-QRS-T-комплексу без додаткового калібрування, або індивідуального навчання для кожного пацієнта, залишається відкритим питанням. Автор вважає, що одним з перспективних виходів з цього технологічного глухого кута є нейроморфні апаратні модулі.

Дисертаційна робота присвячена розробці оригінального програмно-апаратного комплексу для автоматизованого моніторингу ризиків виникнення серцево-судинних захворювань методом реєстрації та аналізу ЕКГ-сигналів людини. Для цього, проведено аналіз сучасних програмних та апаратних рішень автоматизованого контролю фізіологічних параметрів стану здоров’я людини за допомогою ЕКГ-сигналів.

За допомогою сучасної теоретичної бази знань із засобів машинного навчання, запропоновано нову модель штучних нейронних моделей, яка використовується для ефективного вирішення задачі розпізнавання типових ознак ЕКГ-сигналів в режимі реального часу.

В дисертаційній роботі одержані такі нові наукові результати:

1. Вперше розроблено на основі спайкового шифратора вхідного сигналу, рекурентних нейронів внутрішнього шару та вихідних нейронів імпульсну

штучну нейронну мережу, яка представляє собою систему класифікації, що сама навчається та автоматично адаптується до змін вхідного сигналу і забезпечує тим самим оброблення в режимі реального часу клінічно-значущих випадків ЕКГ-сигналу всередині мережі.

2. Вперше розроблено метод оброблення ЕКГ-сигналу в SNN-мережі, який забезпечує збільшення щільності даних за рахунок безпосереднього кодування спайків в ЕКГ-сигналі, набуття мережею відповідних стабільних станів, які відповідають піковим моментам ЕКГ і зменшення помилки розпізнавання, що досягнуто попереднім обробленням і фільтрацією даних на вході нейронної мережі.

3. Удосконалено адаптовану модель імпульсного штучного нейрона шляхом надання йому електричної мультистабільності та здатності відтворювати патерни електричної активності біологічних об'єктів з одночасним розширенням пам'яті та збільшенням обчислювальної потужності, що зумовило її використання в якості базового компонента нейроморфного модуля.

4. Вперше висунуто гіпотезу, згідно якої нейронні структури із більшою кількістю стабільних станів спроможні генерувати більш складні патерни вихідної активності, що свідчить про можливість виконання в такій нейронній мережі обчислювань підвищеної складності.

5. Удосконалено модель структури спайкового шифратора шляхом точного кодування ЕКГ-сигналів, що створило необхідні і достатні передумови для побудови імпульсної штучної мережі і на її основі апаратно-програмного засобу для оброблення ЕКГ-сигналів.

Результати теоретичних та практичних досліджень, проведених в дисертаційній роботі мають практичну спрямованість та пройшли апробацію та впровадження у виробничий процес створення пристрою медичного призначення, а також застосовуються в освітньому та навчальному процесі.

Розроблено оригінальну модель штучного нейрону із біологічним ступенем подібності процесів та властивістю електричної мультистабільності.

Розроблено програмне забезпечення для синтезу та моделювання штучних нейронів із заздалегідь заданим числом станів стабільності, в роботі розглянуто приклади успішного синтезування нейронів із двома, трьома та чотирма стаціонарними станами стабільності. Показано, що нейронні структури із більшою кількістю стабільних станів здатні генерувати більш складні паттерни вихідної активності, а отже можуть виконувати більш складні обчислювання в составі системи обчислювальних одиниць – в нейронній мережі. Встановлені головні межі та умови застосування розробленої штучної моделі.

Головними перевагами розробленої моделі штучної нейронної мережі є велика обчислювальна потужність на ряду із реалізованою функцією пам’яті, а також можливість легкої апаратної реалізації за допомогою стандартного процесору із ПЛІС-архітектурою. Оригінальну модель штучного нейрона використано в контексті обчислювальної імпульсної нейронної мережі.

Розроблено штучну імпульсну нейронну мережу, що складається з основних одиниць: спайкового шифратора вхідного сигналу, рекурентних нейронів внутрішнього шару та вихідних нейронів мережі. Представлена нейронна мережа уявляє собою систему класифікації, що працює без вчителя та здатна адаптуватися під зміни вхідного сигналу в реальному часі без додаткової зміни чи втручання в систему. Спроектовано, змодельовано та реалізовано універсальну схему спайкового шифратора для оптимального та точного кодування сигналів електрокардіографічного походження для обробки нейронною мережею.

Виконано підбір сучасної елементної бази для реалізації схемо-технічного рішення апаратної частини комплексу. Розроблено універсальну схему реєстрації ЕКГ-сигналу, або ЕКГ-фронтенд, що складається із аналогової схеми фільтрації сигналу з електродів, диференціального (інструментального) підсилювача та тракту фільтрації аналогового сигналу і адаптивної регуляції підсилення. Розроблено оригінальне програмне забезпечення для керування мікроконтролерним блоком та аналоговим ЕКГ-фронтендом. Розроблено оригінальне вбудоване програмне забезпечення процесору із ПЛІС-

архітектурою для апаратної реалізації штучної нейронної мережі, або нейроморфний модуль (нейро-модуль).

Розроблено повний пакет конструкторської документації для виготовлення опитного зразку пристрою для автоматизованої реєстрації та аналізу ЕКГ-сигналів. Спроектовано та виготовлено опитний зразок електронного приладу. Розроблено програмне забезпечення для ПК мовою програмування Python, для керування, зчитування та візуалізації даних з пристрою.

Розроблено методику і проведено валідацію показників точності розпізнавання QRS- і P-QRS-T-комплексів на реальних електрокардіограмах. Проведено валідаційні випробування та тестування отриманого приладу, з оцінки точності класифікації основних амплітудно-часових ознак ЕКГ-сигналу на вибірці записів з відкритої бази даних (MIT-BIH), а також в лабораторних умовах, на замірах з волонтерів за допомогою розробленого пристрою. В результаті яких встановлено, що середня похибка класифікації R-піків – 1.00 %; P-піків – 2.36 %; T-піків – 1.37 %, що є значно кращим показником ніж у сучасних діючих аналогів. Досліджено споживання енергії електронного пристрою. Випробування показали, що реалізоване рішення має у 1.25 менше споживання енергії, ніж у діючих сучасних рішень на базі спеціалізованих інтегральних схем (ASIC).

На базі інструментальних розробок та досліджень природи ЕКГ-сигналів було успішно впроваджено у виробництво носимий монітор варіабельності серцевого ритму (“SenceBand”), в якому здобувач виступав як виконавець. У співавторстві опубліковано два патенти в патентному відомстві США (United States Patent and Trademark Office). Результати дослідження впроваджено в науково-дослідну роботу та навчальний процес споріднених вищих навчальних закладів України.

Ключові слова: серцево-судинні захворювання, електрокардіограма, варіабельність серцевого ритму, штучні нейронні мережі, нейроморфні комп’ютери, мікроконтролери, імпульсні нейронні мережі, моделі нейронів, мультистабільні нейрони.

ANNOTATION

Chernetchenko D.V. The method and hardware-software implementation of electrocardiac signals processing using artificial multistable neural networks. – Manuscript copyright.

Dissertation for obtaining the scientific degree of technical sciences candidate (PhD) in specialty 05.11.17 – "Biology and medical equipment and system" – Oles Honchar Dnipro National University, MES of Ukraine, Dnipro. – Vinnytsia National Technical University, MES of Ukraine, Vinnytsia, 2019.

Cardiovascular diseases (CVD) are global problem ("global epidemic", 2018), that causes premature deaths of approximately over 17 million people every year. The World Health Organization (WHO) has identified that people who suffer from CVD or are sensitive to such diseases need special attention for early detection of precursors, provide early qualified assistance and mitigate cardio-vascular risks. This problem motivates to the development of tools for accurate and fast recording and automated analysis of electrocardiographic (ECG) signals in real-time. At the same time, artificial neural networks are often using for the analysis of ECG-signals in real-time, which perform the tasks of classification and are the basis for the detection of cardiac arrhythmias, myocardial ischemia, angina and other chronical changes. Medical device developers are interested in neural structures that are able to reorganize ECG in real-time without pre-training and supervising. Researchers are increasingly focusing on processes in biological neurons and neural structures that have become prototypes for this technology. Artificial models of individual neurons that can reproduce the spike (or impulse) behavior of their biological analogues are usually called spiking neurons. Spiking artificial neural networks (SNN) are able to significantly improve the quality of monitoring patient's control and the effectiveness of automated medical diagnostics, due of providing good signal's patterns classification accuracy. The detection of the P-QRS-T complex in ECG-signal provides a fundamental opportunity to detect all major signal components and further automated analysis. However, specific characteristics of the ECG strongly complicate the detection process of the common components of

the ECG-morphology. Firstly, the features of ECG-signals morphology strongly varies from person to person. Secondly, the ECG-signal often distorted because of the noises of different nature. Today the question of creation of a low-power and high perfomance computing portable system remains an open question. Author believes that the neuromorphic hardware modules are one of the most promising solution for this technological issue.

The dissertation project is related to the development of the original software and hardware complex for automated monitoring of the risks of CVD by the method of registration and analysis of ECG-signals.

The analysis of modern software and hardware solutions of automated control of physiological parameters of human's health with aim of ECG-signals were discovered. With the help of modern theoretical knowledge based on machine learning methodology, a new model of artificial neural models was proposed, which successfully used for effectively solution of the problem of recognizing of the typical ECG-signal features in real-time.

The following new research results were obtained in the dissertation:

1. The SNN was created for the first time, which is a self-learning classification system and automatically provide adaptation to the changes in the input signal and thus provides real-time processing for significant clinical cases of ECG-signal. Neuron network was developed on the basis of an input signal encoder, recurrent internal neurons layer and output readout neurons.

2. A new original method for ECG-signal processing in the SNN was developed for the first time, which provides data density increasing by directly encoding spikes within ECG-signal, and drive the network domains to the stabilization at the stable states that correspond to the peaks moments of ECG-singal. By the way, pre-processing and filtering data at the input of the neural network leads to the significant reducing error of P-QRS-T pattern recognition.

3. Adapted model of spiking neuron by providing it with electrical multistability and ability to reproduce the patterns of electrical activity of biological objects

providing internal system memory and increase of processing power and used as a basic component for the neuromorphic module.

4. A hypothesis has been suggested that neural structures with more stable states are capable of generating more complex patterns of output activity, which leads to the possibility to process more complex calculations in a neural network.

5. The model and structure of the ECG-signal encoder was improved by precise temporal coding of ECG-signals, which created the necessary and sufficient prerequisites for the construction of SNN and basis for hardware and software solution for ECG-signals processing.

The results of theoretical and practical research of the dissertation have a clear practical orientation and have been tested and implemented in the development and mass production of medical devices, as well as applied in the educational and research process.

The original model of artificial neuron with biological complexity of processes and the property of electric multistability was developed. The model described using a formal mathematical apparatus, and has analytical solutions for the case of piece-wise linearized function of neuronal activation. Software for the synthesis and modeling of artificial neurons with a predetermined number of stability states was developed. Examples of successful synthesis of neurons with two, three, and four stationary stability states were considered. It is shown that the neuronal structures with more number of stable states are capable to generate more complex patterns of output electrical activity, and therefore can perform more complex calculations within the system of computing units. The main limits and conditions of application of the developed artificial model were established.

The main advantages of the developed model of the SNN: high computing power, which is necessary for real-time signals processing, implemented memory function, as well as the possibility of easy hardware implementation with a microprocessor based on FPGA architecture.

The original model of an artificial neuron is used in the context of the computational SNN. An SNN has been developed and consisting of the main units: the

spiking coder of the input signal, the recurrent neurons of the internal layer and the output readout neurons of the network.

The universal schematic of the spiking encoder for optimal and accurate coding of ECG-signals processing by a neural network was designed, successfully simulated and implemented. The presented neural network represents a classification system that works without a teacher and is able to adapt to the changes in the input signal in real-time without any additional reorganization and supervising.

The universal scheme of registration of an ECG-signal is developed. Electrical schematic consist of an input analog circuit for the electrodes signal filtering and patient protection (ECG-frontend), a differential (instrumental) amplifier, input signal automatical gain controlling, filtering tract and main microcontroller unit. A complete package of technical documentation for the production of a device prototype of the automated ECG-signals registration and analysis was developed. A sample of an electronic device prototype was designed and manufactured. The original embedded software for controlling the microcontroller and processor with FPGA architecture for the hardware implementation of the artificial neural network (neuro-module) has been developed. Artificial neural model was successfully optimized for effective hardware platform based on FPGA architecture. Desktop software for PC platform with aim of the Python programming language was developed for control, reading and visualization raw-data from the device.

For testing the accuracy of classification of the basic amplitude-time characteristics of the ECG-signal in real-time respective validation tests and testing of the complex were carried out. Method and electronic device validated on ECG-signals samples from an open Physionet database (MIT-BIH), as well as in laboratory conditions for volunteers. The accuracy of the system is investigated, the average error of classification of R peaks is 1.0 %; P-peaks – 2.36 %; T-peaks – 1.37 %, which are a significantly better than for current electronic analogues in production. Energy consumption of the electronic device was discovered and shown that implemented solution has 1.25 less energy consumption than solutions based on application-specific integral circuits elements (ASIC).

Commercial product was successfully implemented (also known as SenceBand) is cardiac heart rate variability (HRV) monitor, which is based of instrumental development and research of the dissertation work. Also, co-authored of two patents at United States Patent and Trademark Office (USPTO). The research results are implemented in the research work of related higher educational institutions of Ukraine.

Key words*:* cardiovascular diseases, electrocardiogram, heart rate variability, artificial neural networks, neuromorphic computers, microcontrollers, spiking neural networks, neuron models, multistable neurons.

СПИСОК ПУБЛІКАЦІЇ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

[1] Є.М. Сніжко, Д. В. Чернетченко, “Динаміка електричних потенціалів моделі мережі нейронів із нелінійними функціями активації”, *Вісник Дніпропетровського університету. Фізика. Радіоелектроніка*, т. 20, № 2(19), с. 50-57, 2012.

[2] М. М. Милых, Е. М. Снежко, И. В. Тимченко, и Д. В. Чернетченко, “Реализация алгоритмов преобразования АДМ-ИКМ в системах автоматики и передачи данных”, *Гірнича електромеханіка та автоматика,* № 2 (93), с. 72-76, 2014.

[3] Є. М. Сніжко, М. М. Мілих, М. П. Моцний, та Д. В. Чернетченко, “Мобільна система для вимірювання кольорових характеристик об’єктів”, *Електромагнітна сумісність та безпека на залізничному транспорті,* № 14, с. 102-106, 2017.

Наукові праці, в яких опубліковані основні наукові результати дисертації:

[4] Y. M. Snizhko, O. O. Boiko, N. P. Botsva, D. V. Chernetchenko, and M. M. Milykh, “Methods for increasing the accuracy of recording the parameters of the cardiovascular system in double-beam photoplethysmography”, *Regulatory Mechanisms in Biosystems,* vol. 9(3), pp. 335-339, 2018.

[5] Д. В. Чернетченко, М. М. Мілих, та К. В. Луданов, “Апаратна реалізація імпульсної штучної нейронної мережі для детектування параметрів

електрокардіографічного сигналу (ЕКГ)", *Вісник Хмельницького національного університету. Технічні науки*, № 4(275), с. 126-133, 2019.

[6] D. V. Chernetchenko, "A Novel Method of Preprocessing and Spike Encoding of Electrocardiographic Signal for Multi-stable Spiking Neuronal Networks Application", *Вчені записки Таврійського національного університету ім. В.І. Вернадського, Серія технічні науки*, т. 30(69), № 3, ч. 1, с. 191-199, 2019.

[7] Д. В. Чернетченко, М. П. Моцний, Н. П. Боцьва, О. В. Єліна та М. М. Мілих, "Автоматизована система реєстрації біоелектричних потенціалів", *Вісник Дніпропетровського університету. Біологія, екологія*, т. 21(2), с. 70-75, 2013.

[8] S. M. Korogod, and D. V. Chernetchenko, "Nature of electrical tristability in a neuron model with bistable asymmetrical dendrites", *Neurophysiology*, vol. 40, no. 5/6, pp. 412-416, 2008.

[9] N. Botsva, I. Naishtetik, L. Khimion, and D. Chernetchenko, "Predictors of aging based on the analysis of heart rate variability", *Pacing Clinical Electrophysiology*, vol. 40(11), pp. 1269-1278, 2017.

[10] D. V. Chernetchenko, R. S. Romaniuk, D. Sawicki, and G. Yusupova, "Analysis of electrical patterns activity in artificial multi-stable neural networks", *Proceedings of SPIE –The International Society for Optical Engineering*, 7 p., 2019.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

[11] T. Tkachenko, N. Botsva, T. Komendar, and D. Chernetchenko, "Autonomous hardware-software complex for biosignals registration and processing", на *III Всеукр. наук.-практ. конф. Перспективні напрямки сучасної електроніки, інформаційних і комп'ютерних систем (MEICS – 2018)*, Дніпро, 2018, с. 95-96.

[12] A. Lavrenjuk, D. Chernetchenko, N. Botsva, and K. Pustova, "Blood pressure monitoring", in *Proc. XIII Międzyn. nauk.-prakt. konf. Naukai Inowacja* – 2017, Przemyśl, 2017, pp. 53-55.

[13] K. Pustova, D. Chernetchenko, N. Botsva, and A. Lavrenjuk, "Non-invasive monitoring tools for automatic stress detection", in *Proc. XIII Mezin. věd.-prakt. konf. Zprěvy VědeckéIdeje–2017*, Praha, 2017, pp. 21-23.

[14] Д. В. Чернетченко, Д. І. Золотова, Н. П. Боцьва, Е. М. Гасанов, та В. С. Олійник, “Автономний апаратно-програмний комплекс реєстрації та обробки кардіографічних сигналів”, in *Proc. XIIInt. scientific and pract. conf. Cutting-Edge Science – 2016*, Sheffield, 2016, pp.113-116.

[15] Д. В. Чернетченко, Е. М. Гасанов, Д. І. Золотова, та В. С. Олійник, “Апаратно-програмний комплекс реєстрації міографічних сигналів на базі мікроконтролера”, in *Proc. XII Mezin. věd.-prakt. konf. Efektivní nástroje moderníchvěd – 2016*, Praha, 2016, pp. 23-26.

[16] М. П. Моцний, Д. В. Чернетченко, Н. П. Боцьва, та О. В. Єліна, “Реєстрація біоелектричних потенціалів засобами комплексної автоматизованої системи”, *in Proc. X Mezin. věd.-prakt. konf. Veda a technologie: krok do budouchnosti– 2014,* Praha, 2014, pp. 102-105.

[17] С. В. Тимчик, С. М. Злепко, І. О. Криворучко, та Д. В. Чернетченко, “Професійне здоров’я людини і психічна, фізіологічна і клінічна складові”, на *Міжнар. наук.-практ. конф. Сучасні наукові дослідження у психології та педагогіці – прогресс майбутнього,* Одеса, 2019, с. 45-48.

[18] Д. В. Чернетченко, Д. І. Золотова, Е. М. Гасанов, та В. С. Олійник, “Комплекс реєстрації та обробки біопотенціалів на базі мікроконтролера”, in *XII Międzynarodowej naukowo-praktycznej konferencji Perspektywiczne Opracowania Sa Nauka i technikami – 2016*, Przemysl, 2016, pp. 77-80.

[19] D. V. Chernetchenko, T. A. Botsva, “Method of registering the intervals between adjacent R-peaks of the ECG signal with the one hand in order to diagnose and assess the state of the human body and Heart Rate Variability wearable monitoring device”, *U.S. Patent and Trademark Office, US20180242858A1*, 2018.

[20] D. V. Chernetchenko D.V., T. A. Botsva, “Method and apparatus for cuff less blood pressure monitoring based on simultaneously measured ECG and PPG signals designed in wristband form for continuous wearing”, *U.S. Patent and Trademark Office, US20190059752A1*, 2019.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ВСР – варіабельність серцевого ритму

ЕКГ – електрокардіограма

ПЛІС – програмовані логічні інтегральні схеми

ССЗ – серцево-судинні захворювання

ЧСС – частота серцевих скорочень

ADC – analog-to-digital converter (аналогово-цифровий перетворювач)

ASIC – application-specific integrated circuit (спеціалізована інтегральна схема)

ANN – artificial neural network (штучна нейронна мережа)

CMRR – common-mode rejection ratio (коефіцієнт синфазного послаблення)

CVD – cardiovascular disease (кардіо-васкулярні захворювання)

DSP – digital signal processor (цифровий сигнальний процесор)

ECG – electrocardiogram (електрокардіограма)

HR – heart-rate (частота серцевих скорочень)

HRV – heart rate variability (варіабельність серцевого ритму)

IA – instrumentation amplifier (інструментальний підсилювач)

IC – integrated circuit (інтегральна схема)

PGA – programmable gain amplifier (підсилювач із програмованим коефіцієнтом підсилення)

SNR – signal-to-noise ratio (відношення сигнал/шум)

SNN – spiking neural network (спайкова нейронна мережа)

UART – universal asynchronous receiver/transmitter (універсальний асинхронний приймач/передавач)

USB – universal serial bus (універсальна послідовна шина)

FDA – Food and Drug Administration (Управління по Контролю за Продуктами та Ліками)

FPGA – field programmable gate arrray (програмовані логічні інтегральні схеми)

ВСТУП

Обґрунтування вибору теми дослідження. Серцево-судинні захворювання (ССЗ) є глобальною проблемою або "глобальною епідемією", що за даними всесвітньої організації охорони здоров'я (ВООЗ) [1], викликають передчасну смерть 17 мільйонів людей щороку. Статистика American Heart Association свідчить, що від ССЗ щодня помирають приблизно 2300 американців і в середньому кожні 38 секунд для однієї людини фіксується одна кардіокатастрофа (інсульт, інфаркт, тощо)[2]. Серцево-судинні захворювання створюють значне навантаження на економіку країни і потребують значних коштів на їх діагностику і лікування.

Нижче наведено графік (рис. 1), який демонструє динаміку смертності на 100 тис. населення від ССЗ у чоловіків до 65 років у чотирьох колишніх радянських країнах: Росії, Казахстані, Україні та Білорусі. Графік починається з 1980 року і продовжується до 2015 року [2].

ВООЗ визначено [1], що люди, які страждають на ССЗ або чутливі до ризику виникнення таких захворювань (за наявності одного або декількох факторів ризику, таких як підвищений кров'яний тиск, діабет, гіперліпідемія тощо), потребують підвищеної уваги, раннього виявлення передвісників хвороби та надання своєчасної кваліфікованої допомоги. Це, в свою чергу, зумовлює необхідність розроблення інструментальних засобів реєстрації та автоматизованого аналізу електрокардіографічних сигналів в режимі реального часу для передбачення та завчасного попередження ризиків виникнення ССЗ та кардіокатастроф [72]. Водночас, існуючі засоби для точного інструментального аналізу і моніторингу залишаються або досить дорогими рішеннями з недостатньою зручністю їх використання у повсякденному житті, або потребують перенесення основних обчислювальних операцій на окрему серверну частину. Це призводить до залежності таких систем від наявності та якості Інтернет-зв’язку з “хмарою”, отже до їх неспроможності своєчасно забезпечити миттєву реакцію лікаря на сигнал від пацієнта про проблеми зі

здоров'ям [74], [77]. Наведена аргументація додатково підтверджує актуальність теми дисертаційного дослідження і необхідність розроблення апаратно-програмного комплексу для реєстрації та автоматизованого аналізу електрокардіографічних сигналів на базі штучних нейронних мереж [72].

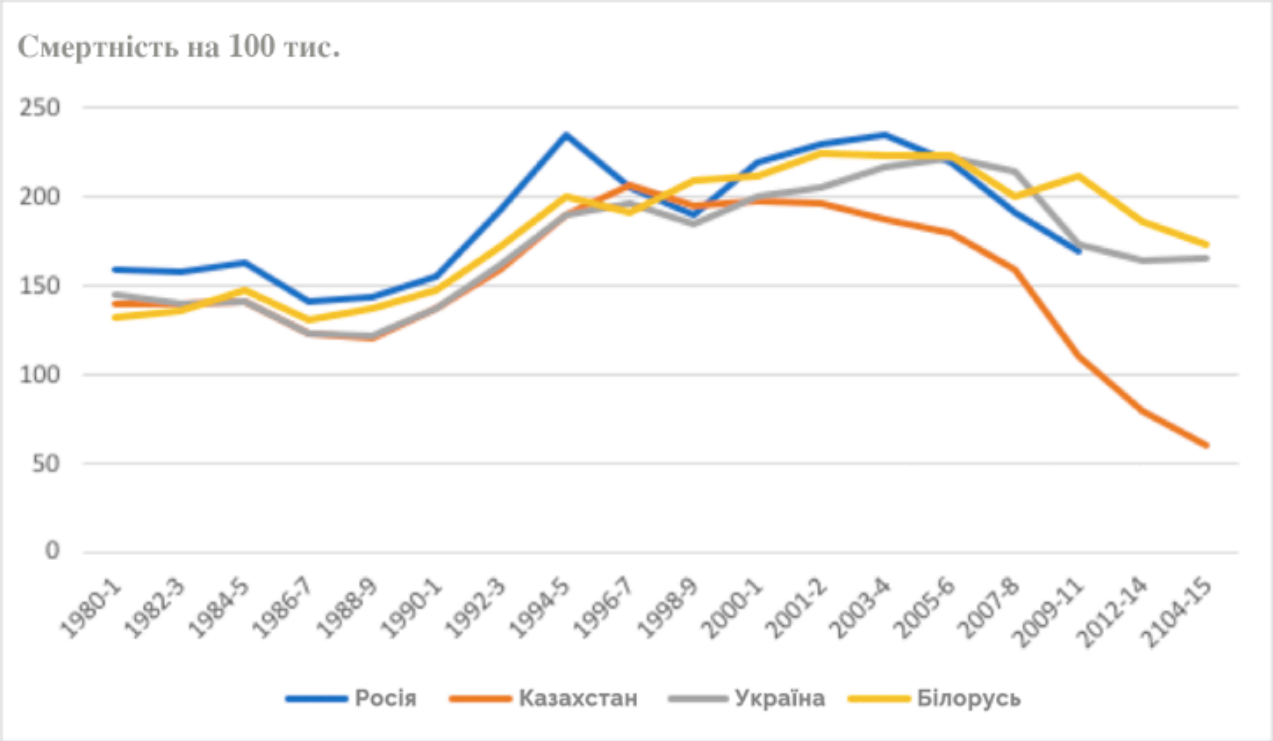


Рисунок 1 – Статистика смертності від ССЗ з 1980 по 2015 роки у країнах колишніх республік СРСР: Росії, Казахстані, Україні, Білорусі [2]

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась відповідно до договору науково-дослідних і дослідно-конструкторських робіт для ТОВ "Науково-виробниче підприємство "СМД" (акт впровадження №11/09 від 11.09.2018), а також згідно з планом науково-дослідної роботи Дніпровського національного університету ім. Олеся Гончара "Дослідження принципів обробки інформації та управління в біомедичних та технічних системах" (№ держреєстрації 0116U003588), в яких здобувач брав участь як виконавець.

Мета і завдання дослідження. Мета роботи полягає в підвищенні ефективності оброблення та точності детектування головних ознак P-QRS-T-

комплексу ЕКГ-сигналів шляхом розроблення методу та апаратно-програмного засобу на основі штучних мультистабільних нейронних мереж.

Для досягнення поставленої мети було визначено такі завдання:

1. Узагальнити та проаналізувати наявні методи та апаратно-програмні засоби для оброблення ЕКГ-сигналів.

2. Розробити адаптовану модель імпульсного штучного нейрона і визначити обмеження на її застосування у складі нейронної мережі.

3. Розробити референтний метод оброблення та детектування головних ознак ЕКГ-сигналу (P-, Q-, R-, S-, T-зубці та відповідні інтервали).

4. Розробити структурну схему і на її основі – апаратно-програмний засіб для оброблення ЕКГ-сигналів.

5. Запропонувати схемо-технічну реалізацію апаратного забезпечення засобу.

6. Обґрунтувати вибір та адаптувати методику валідації показників точності розпізнавання структурних комплексів ЕКГ-сигналу до умов дисертаційного дослідження.

7. Провести експериментальне дослідження та апробацію розробленого апаратно-програмного засобу.

Об’єкт дослідження – процес оброблення ЕКГ-сигналу із застосуванням штучних мультистабільних нейронних мереж.

Предмет дослідження – модель штучного нейрона, референтний метод та апаратно-програмний засіб для оброблення ЕКГ-сигналу.

Методи дослідження. Для оброблення даних застосовували методи дискретного оброблення інформації і математичної статистики; розроблення структурної схеми та апаратно-програмного засобу виконано на базових принципах системного підходу; для розроблення моделі нейронів та нейронних мереж використовували методи математичного моделювання, штучного інтелекту і програмні пакети NEURON та Matlab; програмне забезпечення побудовано з використанням середовища проектування IAR Embedded Workbench, мови програмування VHDL та пакету Xilinx ISE WebPack;

проектування схемо-технічного рішення та розробку друкованої плати апаратного засобу реалізовано за допомогою системи автоматизованого проектування Altium Designer; використано бібліосемантичний метод для вивчення вітчизняного та світового контенту.

Наукова новизна отриманих результатів полягає в тому що:

1. Вперше розроблено на основі спайкового шифратора вхідного сигналу, рекурентних нейронів внутрішнього шару та вихідних нейронів імпульсну штучну нейронну мережу, яка представляє собою систему класифікації, що сама навчається та автоматично адаптується до змін вхідного сигналу і забезпечує тим самим оброблення в режимі реального часу клінічно-значущих випадків ЕКГ-сигналу всередині мережі.

2. Вперше розроблено метод оброблення ЕКГ-сигналу в SNN-мережі, який забезпечує зниження щільності даних за рахунок безпосереднього кодування спайків в ЕКГ-сигналі, набуття мережею відповідних стабільних станів, які відповідають піковим моментам ЕКГ і зменшення помилки розпізнавання, що досягнуто попереднім обробленням і фільтрацією даних на вході нейронної мережі.

3. Удосконалено адаптовану модель імпульсного штучного нейрона шляхом надання йому електричної мультистабільності та здатності відтворювати патерни електричної активності біологічних об’єктів з одночасним розширенням пам’яті та збільшенням обчислювальної потужності, що зумовило її використання в якості базового компонента нейроморфного модуля.

4. Вперше висунуто гіпотезу, згідно якої нейронні структури із більшою кількістю стабільних станів спроможні генерувати більш складні патерни вихідної активності, що свідчить про можливість виконання в такій нейронній мережі обчислювань підвищеної складності.

5. Удосконалено модель структури спайкового шифратора шляхом точного кодування електрокардіографічних сигналів, що створило необхідні і достатні передумови для побудови імпульсної штучної мережі і на її основі апаратно-програмного засобу для оброблення ЕКГ-сигналів.

Практичне значення отриманих результатів.

1. Обґрунтовано вибір сучасної елементної бази для схемо-технічної реалізації апаратного забезпечення засобу, що забезпечило розроблення ЕКГ-фронтенду (універсальної схеми реєстрації ЕКГ-сигналу), схеми аналогової фільтрації та адаптивного регулювання підсилювання вхідного сигналу, що надходить з ЕКГ-електродів та інструментального підсилювача, який керується вбудованим в мікроконтролер програмним забезпеченням.

2. Запропоновано оригінальний засіб, інструментальні елементи та дослідження якого лягли в основу двох патентів США і на його основі розроблено і виготовлено дослідний зразок апаратно-програмного засобу з комплектом конструкторської документації для подальшого дрібносерійного виробництва.

3. Розроблено методику і проведено валідацію показників точності розпізнавання QRS- і P-QRS-T-комплексів на реальних електрокардіограмах. Проведено валідаційні випробування та тестування отриманого приладу, з оцінки точності класифікації основних амплітудно-часових ознак ЕКГ-сигналу на вибірці записів з відкритої бази даних (MIT-BIH), а також на волонтерах при реальному використанні апаратного засобу. Проведено оцінювання показників точності розпізнавання QRS- і P-QRS-T, в результаті було показано, що похибка детектування не перевищує 1 % для R-зубця; 2.36 % для P-зубця і 1.37 % для T-зубця, що суттєво краще ніж у діючих аналогів.

4. Результати дисертаційної роботи впроваджено у виробничий процес на підприємстві ТОВ “Науково-виробниче підприємство СМД” − акт впровадження №11/09 від 11.09.2018 р.), що дозволило оптимізувати технічні рішення в приладі для реєстрації та аналізу ЕКГ з зап’ястя людини, і навчальний процес кафедри експериментальної фізики та фізики металів Дніпровського національного університету ім. Олеся Гончара (акт впровадження від 10.04.2019 р.), що сприяло поглибленню знань студентів в напрямку створення вбудованих систем та підвищило якість викладання відповідних дисциплін.

Особистий внесок здобувача. Всі результати наукових і практичних досліджень, що увійшли до дисертаційної роботи, отримані і розроблені автором особисто. Особистий внесок здобувача в працях, написаних у співавторстві, полягає в наступному: в [1] розглянуто розробку та дослідження електричної поведінки штучної нейронної мережі, що базується на мультистабільних нейронних моделях; в [2] запропоновано новий цифровий метод адаптивної модуляції та фільтрації сигналів в реальному часі за допомогою апаратної структури програмованих логічних інтегральних схем (ПЛІС) на прикладі Spartan 3E від Xilinx; в [3] запропоновано метод та портативну апаратну реалізацію системи розпізнавання та обробки даних від сенсорів (світлові сенсори) у реальному часі на базі методів цифрової фільтрації та машинного навчання із використанням штучних нейронних мереж; в [4] наведено методику підвищення ефективності оцінки фізіологічного стану кардіо-васкулярної системи людини, за допомогою методу двох-променевої фото-плетизмографії (ФПГ), запропоновано новий підхід до детектування максимумів (піків) ФПГ сигналу людини в реальному часі, реалізовано апаратний комплекс на базі біосенсору від Maxim Dallas та мікроконтролерної платформи STM32; в [5] розроблено та протестовано апаратну реалізацію мультистабільної нейронної мережі для вирішення задачі розпізнавання та класифікації QRS-комплексів ЕКГ-сигналу; в [7] розроблено апаратно-програмний комплекс автоматизованої системи реєстрації біоелектричних потенціалів на базі USB-пристрою, з подальшою обробкою цифрових сигналів на ЕОМ; в [8] проведено аналіз та запропоновано методику розробки математичної моделі мультистабільних нейронів як головних структурних одиниць, що мають властивості пам’яті та здатні генерувати складні імпульсні (спайкові) послідовності сигналів; в [9] запропоновано нову методику оцінки біологічного віку людини за станом кардіо-васкулярної системи із застосуванням методів аналізу варіабельності серцевого ритму (ВСР) та методів машинного навчання за допомогою штучних нейронних мереж; в [10] проведено детальний аналіз мультистабільних штучних нейронних мереж на математичній моделі, показано різні імпульсні режими

роботи та структурні передумови для виникнення складних паттернів електричної активності у мережах; в [11] приведена апаратна реалізація автономного та безпроводового комплексу для реєстрації фізіологічних параметрів людини (електрокардіографічних та електроміографічних (ЕМГ) потенціалів людини); у [12] розглянуто методику оцінки стану кардіо-васкулярної системи людини та безманжетної оцінки артеріального тиску за допомогою аналізу ЕКГ та ФПГ сигналів людини, розроблено програмно-апаратний комплекс для автоматизованої оцінки артеріального тиску; в [13] приведена реалізація методики та апаратно-програмний комплекс для оцінки функціонального стану оператора, а саме параметрів стресу та відновлення автономної нервової системи, за допомогою методів ВСР за ЕКГ-сигналом; в [14] реалізовано програмно-апаратний комплекс для віддаленої реєстрації та аналізу ЕКГ- сигналів (детектування P-QRS-T комплексів) за допомогою безпроводової технології Bluetooth 4.0 та мікроконтролерної платформи Arduino; в [15] реалізовано програмно-апаратний комплекс для віддаленої реєстрації та аналізу амплітудно-частотних параметрів ЕМГ сигналів пацієнта за допомогою безпроводової технології Bluetooth 4.0 та мікроконтролерної платформи Arduino; в [16] розроблено програмно-апаратний комплекс для дослідження впливів оточуючого середовища на біопотенціали; в [17] запропоновано підхід для оцінювання процесів взаємодії особистості з інформаційними системами і комплексами в умовах зовнішнього середовища без безпосереднього спостерігання за об’єктами управління; в [18] розроблено універсальний аналогово-цифровий портативний мікроконтролерний пристрій для неінвазійної реєстрації біопотенціалів людини; в [19] наведено опис нової методики реєстрації R-піків ЕКГ-сигналу за допомогою портативним носимим мікропроцесорним пристроєм в реальному часі, та оцінки фізіологічного стану за допомогою аналізу ВСР; в [20] наведено опис нової методики оцінки артеріального тиску пацієнта за допомогою портативного носимого мікропроцесорного пристрою в реальному часі.

Апробація матеріалів дисертації. Основні положення роботи викладено та обговорено на науково-практичних конференціях різного рівня: III Всеукраїнській НПК “Перспективні напрямки сучасної електроніки, інформаційних і комп’ютерних систем (MEICS-2018)” (м. Дніпро, 2018 р.); XIII МНПК “Nauka i Inowacija – 2017” (м. Пшемисль, Польща, 2017 р.); XIII МНПК “Zprěvy Vědecké Ideje–2017” (м. Прага, Чехія, 2017 р.); XII Міжнародній науковій і практичній конференції “Cutting-Edge Science – 2016” (м. Шеффілд, Велика Британія, 2016 р.); XII МНПК “Efektivní nástroje moderních věd – 2016” (м. Прага, Чехія, 2016 р.).

Публікації. Результати досліджень відображені в 20-и публікаціях: з них 5-ть статей у наукових фахових виданнях України з технічних наук ([1] – [3], [5], [6]), 3-и статті в закордонних виданнях ([8] – [10]), 2-і статті в інших виданнях ([4], [7]), всі 10-ь статей індексуються в міжнародних наукометричних базах даних; 8-м матеріалів і тез доповідей конференцій ; 2-а патенти ([19], [20]).

Структура та обсяг дисертації. Дисертаційна робота викладена на 230 сторінках машинописного тексту, складається зі вступу, 4 розділів, загальних висновків, списку використаних джерел та 7 додатків. Обсяг основного тексту дисертації складає 125 сторінок друкованого тексту. Робота ілюстрована 7 таблицями, 66 рисунками. Список використаних джерел містить 101 найменування.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ, МОДЕЛЕЙ ТА АПАРАТНО-ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ОБРОБЛЕННЯ ЕЛЕКТРОФІЗІОЛОГІЧНИХ СИГНАЛІВ, ЗОКРЕМА ЗА ДОПОМОГОЮ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

1.1 Особливості комп'ютерного аналізу ЕКГ-сигналу

Інформативною основою біопотенціалів, які генерує біооб'єкт, є їх форма, амплітуда, період, частота, частотний спектр та інші фізичні параметри [78], [79]. Реєстрація їх певним інструментальним способом, біофізичне тлумачення та аналіз складають основу будь-яких відомих методів електрокардіографії для діагностики роботи серця [101]. ЕКГ у діагностиці застосовують для:

– визначення частоти (пульсу) і регулярності серцевих скорочень (наприклад, екстрасистоли – позачергових скорочень, аритмії – випадання окремих скорочень);

– визначення гострого або хронічного ушкодження міокарда (інфаркту міокарда, ішемії міокарда);

– виявлення порушень обміну калію, кальцію, магнію та інших електролітів;

– виявлення порушень внутрішньосерцевої провідності (різні блокади);

– скринінгу при ішемічній хворобі серця, зокрема при пробах з навантаженням;

– виявлення фізичного стану серця (гіпертрофія лівого шлуночка);

– отримання інформації про несерцеві захворювання, такі як тромбоемболія легеневої артерії;

– віддаленої діагностики гострої серцевої патології (інфаркту міокарда, ішемії міокарда) за допомогою кардіофона;

– дослідження когнітивних процесів, самостійно або в поєднанні з іншими методами;

– під час диспансеризації.

Більш якісну розшифровку ЕКГ проводять за допомогою автоматичного аналізу і розрахунку площі зубців при використанні спеціальних відведень (векторна теорія), проте на практиці, в основному, користуються таким показником, як напрям електричної вісі серця, що є сумарним вектором QRS [56], [57], [67]. Аналіз ЕКГ здійснюють у послідовному порядку, визначаючи норму і порушення:

– оцінюють серцевий ритм і вимірюють частоту серцевих скорочень (ЧСС), в нормі – ритм синусовий, ЧСС – від 60 до 80 ударів на хвилину;

– розраховують інтервали QT, що характеризують тривалість фази скорочення (систоли) за спеціальною формулою (частіше використовують формулу Базетта), норма – 390-450 мс. Якщо цей інтервал подовжується, це може вказувати на підозру виникнення ішемічної хвороби серця, атеросклероз, міокардит, ревматизм [73], [75]. Натомість гіперкальціємія призводить до скорочення інтервалу QT. Провідність імпульсів, з якою пов’язаний даний інтервал, часто розраховують за допомогою автоматизованого комп'ютерного аналізу, що може значно підвищити достовірність результатів [62];

– положення електричної вісі серця починають розраховувати від ізолінії за висотою зубців (в нормі R завжди вище S) і, якщо S перевищує R, а вісь відхиляється вправо, то можливі порушення діяльності правого шлуночка, якщо навпаки – вліво, і при цьому висота S більше R в II і III відведенях – підозрюють гіпертрофію лівого шлуночка;

– вивчають комплекс QRS, який формується при проведенні електричних імпульсів до м'язів шлуночків і визначає діяльність останніх (норма - відсутність патологічного зубця Q, ширина комплексу не більше 120 мс). У випадку, якщо даний інтервал зміщується, говорять про блокади (повні і часткові) ніжок пучка Гіса або порушення провідності. Причому неповна блокада правої ніжки пучка Гіса є електрокардіографічним критерієм гіпертрофії правого шлуночка, а неповна блокада лівої ніжки пучка Гіса може вказувати на гіпертрофію лівого шлуночка;

– описують сегменти ST, які відображають період відновлення

початкового стану серцевого м'яза після його повної деполяризації (в нормі знаходиться на ізолінії) і зубець Т, що характеризує процес реполяризації обох шлуночків; зубець спрямовано вгору, він асиметричний, нижчий R-зубця, за тривалістю довший QRS-комплексу.

На рис. 1.1 наведено електрокардіограму здорової людини, серце якої працює ритмічно і правильно [58]. Але робити аналіз досить тривалих записів за допомогою тільки візуального способу та ручного підрахунку параметрів іноді не є можливим.

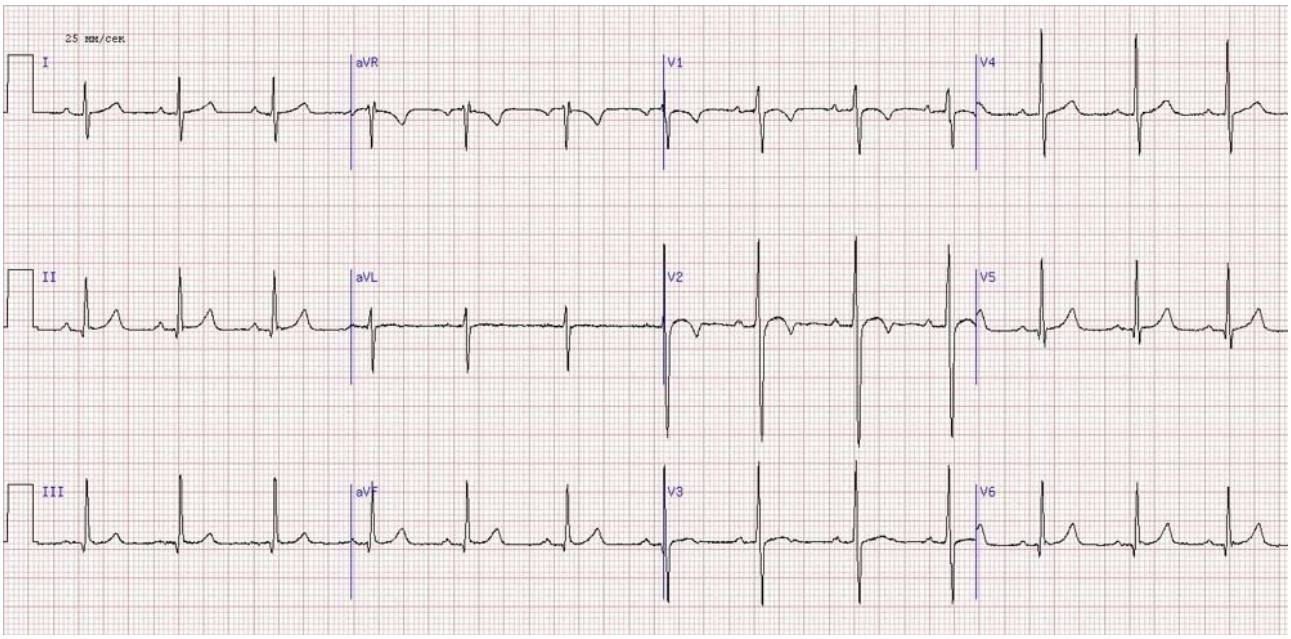


Рисунок 1.1 – ЕКГ здорової людини у нормі (відведення I, II, III, avR, avL, avF, V1-V6) [58]

За таких умов для полегшення аналізу ЕКГ, підвищення його якості, та уникнення помилок дуже актуальною стає автоматизація обробки та інтерпретації кардіосигналів за допомогою сучасних апаратних та програмних засобів [88]–[91].

1.2 Аналіз моделей штучних мультистабільних нейронів

Виявлення комплексу P-QRS-T в ЕКГ-сигналі забезпечує фундаментальну можливість для виявлення всіх головних компонентів сигналу і подальшого

автоматизованого аналізу [59]–[61]. Однак певні специфічні характеристики ЕКГ ускладнюють програмне виявлення форми ЕКГ [63]: по-перше, це особливість морфології сигналів ЕКГ, що змінюється від людини до людини; по-друге, ЕКГ- сигнал досить часто має компоненти шуму різного походження. У роботі [80] запропоновано адаптивний алгоритм фільтрації, заснований на штучній нейронній мережі (ANN) для виявлення QRS-комплексів. Але результативність роботи такої системи все ще дуже сильно залежить від якості вибірки даних для тренування системи. Ідентифікація P-QRS-T з ЕКГ-сигналу є фундаментальною необхідністю для оцінки частоти серцевих скорочень і аналізу варіабельності серцевого ритму (ВСР). Хоча методики виявлення P-QRS-T з часом були досить успішними [56]–[65], проте останні досягнення в області медичної допомоги [62], [92] мотивували дослідників повернутися до питання точності детектування QRS. Це пов'язано як з якістю необроблених ЕКГ-даних з носимих датчиків, сигнал з яких потопає у артефактах руху та дрейфах базової лінії, так і обмеженістю пристроїв з вбудованими носимими ЕКГ-датчиками, за площею, енергоспоживанням і обчислювальними можливостями. Але в той же час, виконувати обробку, анотування, а отже і класифікацію анормальних станів на стороні сенсору (пристрою) може бути дуже критичним. Оскільки будь-яка додаткова комунікація із сервером (або “хмарою”) та аналіз ЕКГ-сигналу на його стороні може призвести до втрати даних та/або дуже великих затримок при спрацьовуванні певних тригерів (наприклад, аритмія, елевація ST-сегменту, та ін.). Отже, дуже часто оброблення та розпізнавання даних на стороні пристрою є повністю оправданим. На жаль, на сьогодні створення такої обчислювальної системи із низьким рівнем енергоспоживання та високою обчислюваною потужністю залишається відкритим питанням. Автор вважає, що одним з перспективних виходів з цього технологічного глухого кута є нейроморфні апаратні модулі (далі нейро-модулі).

Більш ранні методи аналізу ЕКГ-сигналів засновані на методі аналізу часових та амплітудних характеристик сигналу за допомогою мікроконтролерів

на основі функцій цифрового процесора (так званий – DSP), але часове представлення не завжди достатнє для вивчення всіх особливостей ЕКГ-сигналів. Отже, потрібно додаткове частотне представлення сигналу в реальному часі. Для цього найчастіше використовують швидке перетворення Фур'є, або FFT (Fast Fourier Transform). Зазвичай, це неминуче обмежує аналіз, оскільки перетворення необхідно виконувати неперервно у реальному часі, досить швидко та точно, а це потребує значних ресурсів та може суттєво підвищити енергоспоживання системи [69]. Тому нові ефективні апаратні підходи стають дуже актуальними у цій галузі [93]–[97].

З огляду на стрімке розширення сфери застосування штучних нейронних мереж, дослідники звертають все більшу увагу на процеси у біологічних нейронах та нейронних структурах, які свого часу стали прототипами для цієї технології. Мозок являє собою дуже складну систему, що складається з великої кількості основних структурних одиниць – нейронів, які взаємопов'язані через синапси [50]. Особливий інтерес викликають процеси у нейронах, які приводять до електричних відповідей або імпульсів, що називають спайками. Добре відомо, що складні когнітивні процеси кори головного мозку пов'язані із різними шаблонами, або патернами, вихідної активності окремих нейронів [51], а часова синхронізація спайків є найважливішим елементом в системі обробки нейронної інформації [52], [53]. Останні дослідження математичних нейронних моделей показують, що дуже важливе значення для генерації цих унікальних патернів активності мають морфологічні властивості окремих нейронів [43]. Цей факт робить кожний окремий нейрон сам по собі досить потужною обчислювальною одиницею, яка бере участь у зберіганні інформації, виконанні розпізнавання вхідних образів та класифікації векторів [98]. Штучні моделі окремих нейронів, що здатні відтворювати спайкову поведінку біологічних аналогових, як правило, називають спайковими нейронами, нейронні мережі – спайковими нейронними мережами (Spiking Neural Networks, SNN).

Зазвичай класичні моделі клітин штучних нейронних систем не враховують просторову структуру окремих нейронів та синаптичну взаємодію

нейронів у просторі. Проте останні дослідження показують, що величезне різноманіття структур нейронів не може пов’язуватись лише зі збільшенням корисної поверхні клітини, просторова структура грає значну роль у процесах обробки сигналів нейронами [31]–[33]. Головною одиницею інформації в нейронних мережах є не одиничний спайк, а їх унікальна послідовність (паттерн), або стабільний режим генерації [36]. Також встановлено, що мультистабільні нейрони здатні генерувати стійкі патерни вихідної активності залежно від певного просторово-часового розподілення вхідних сигналів [39]. Таким чином, штучні нейрони, розроблені за аналогією з функціонуванням своїх біологічних прототипів, здатні найліпшим чином виконувати задачі класифікації або розпізнавання, в яких необхідно виділяти основні схожі елементи вхідних сигналів, незважаючи на існуючі завади, без допомоги зовнішніх вчителів.

Структура електричних відповідей біологічних нейронів головного мозку відрізняється своїм різноманіттям і унікальністю [28], [31], [38], [39]. З огляду на те, що сучасні методики застосовують мережі з сотень тисяч генеруючих нейронів, для моделі окремого нейрона виникає необхідність пошуку компромісу між двома, здавалося б, взаємовиключними вимогами: модель повинна бути обчислювально простою, але при цьому здатною до генерації такої кількості шаблонів електричної активності, яку можна порівняти із реальними біологічними нейронами [47], [48], [52]. Використання для штучних нейронних мереж біофізично точних моделей типу Ходжкіна-Хакслі є обчислювально нездійсненною задачею, оскільки в режимі реального часу можливо імітувати лише кілька нейронів одночасно. Разом з цим, елементарні моделі сумації збудження є обчислювально ефективними [3], [51], проте дуже примітивними й нездатними імітувати різноманіття динаміки, що проявляється у біологічних нейронів, наприклад, нейронів кори головного мозку (неокортексу)[37].

В останні роки для аналізу сигналів фізіологічної природи у реальному часі, наприклад, ЕКГ, з’явилося багато реалізацій з використанням штучних нейронних мереж, які здатні виконувати завдання класифікацію та розпізнавання. Такі штучні мережі є основою для аналізу за ЕКГ-сигналом

патологічних змін, включаючи аритмію, ішемію міокарда та інші хронічні випадки. Всі ці методики використовують класичні нейромережеві підходи з контрольованим навчанням із вчителем, успіх якого залежить від наявності значних обсягів вхідних даних, які вводять до мережі і які є достатніми для використання в широкому діапазоні суб'єктів із наявністю певних порушень та без них. Тому для вирішення задач класифікації дослідники зацікавлені у таких нейронних структурах, які здатні до реорганізації у реальному часі без попереднього навчання, без додаткового контролю та вчителя. Такі нейронні мережі сьогодні набувають популярності для вирішення задач комплексного розпізнавання образів [4], [7], [22], апроксимації функцій [12] та класифікації зображень [66, 22]. Ще однією причиною наявного успіху спайкових нейронних мереж є їх ефективна апаратна реалізація [44], прикладами якої є великі масштаби нейроморфних обчислень таких систем, як TrueNorth, CxQuad, NeuCube, SpiNNaker, NeuroGrid та HICANN [44], [45]. Деякі з цих систем спочатку розроблені для високопродуктивних обчислень (наприклад, TrueNorth і SpiNNaker), а інші призначені для вбудованих систем з низькою потужністю (наприклад, CxQuad і HICANN).

У дисертаційній роботі запропоновано використовувати спайкову SNN [5], [10], [11], [17] для подальшої реалізації на ПЛІС-архітектурі нейро-модуля [71] та виконання оцінки всіх головних зубців та інтервалів з ЕКГ-сигналів у реальному часі. Слід зазначити, що SNN – це дуже потужні і біологічно реалістичні моделі обчислень, що за динамікою дуже схожі на динаміку людського мозку.

При застосуванні цього підходу, аналоговий сигнал ЕКГ кодується безпосередньо в імпульси, або спайки, які використовуються для збудження резервуара рекурентно пов'язаних спайкових нейронів. Це рішення бере початок з обчислювальної моделі мультистабільного автомата [17]. Нейрони в архітектурі з'єднані між собою за допомогою синапсів з оновленням ваги з використанням спайкової час-залежної пластичності (STDP) [19], [20], – важливої варіації правила Хебба. На етапі зчитування результатів у архітектурі

використовується бі-стабільний нейрон, який інтегрує сигнали з внутрішніх спайкових нейронів і генерує вихідні розряди, коли система виявляє комплекс, як певну ознаку ЕКГ (наприклад, QRS комплекс) [76], [99], [100]. Для виділення додаткових ознак сигналу ЕКГ доцільно використовувати масиви вихідних нейронів для ініціювання відповідного виходу при наявності визначеного стану внутрішніх нейронів, що може відповідати визначеній особливості вхідного сигналу (наприклад, P, Q, S, T-зубці).

В якості структурного елементу мережі запропоновано та розроблено модель нейрона із властивістю мультистабільності. Головною перевагою запропонованої моделі є те, що її поведінка повністю детермінована, може описуватись аналітичними рівняннями та може легко адаптуватися до апаратної платформи у повній відповідності до наведених визначень.

Стійкий рівноважний стаціонарний стан нейрона – такий стаціонарний стан, що будь-яке скінченне мале відхилення потенціалу мембрани нейрона, приводить до появи струмів, що протидіють цій зміні і повертають потенціал у початковий стан, в якому нейрон може перебувати як завгодно довго.

Динамічна вольт-амперна характеристика ($ВАХ_{дин}$) – величина трансмембранного струму I, який протікає крізь мембрану при утриманні на ній потенціалу V, у різні моменти часу. При цьому, порядок часових міток визначається кінетикою іонних каналів мембрани.

Моностабільність – здатність нейрона знаходитись в одному стійкому рівноважному стаціонарному електричному стані, що визначається потенціалом спокою його пасивної мембрани.

Бі-стабільність – здатність нейрона знаходитись в одному з двох можливих стаціонарних стійких рівноважних електричних станів, низької (downstate) або високої (upstate) деполяризації [28].

Мультистабільність – здатність нейрону знаходитись в одному з трьох, та більше стійких рівноважних стаціонарних електричних станів [31]–[33].

Нестаціонарна мультистабільність – здатність нейрона бути мультистабільною структурою впродовж певних скінчених інтервалів часу [31].

Мультистабільність є однією з найголовніших електричних властивостей нейронів. Система мультистабільна, якщо за однакових параметрів системи залежно від початкових умов остаточний стаціонарний електричний стан нейрону може бути різним. Звичайні електричні поєднання нейронних частин, наприклад, соми та дендритів, із різними популяціями каналів та відповідно різними ефективними рівноважними потенціалами можуть привести до бі-стабільності клітини, коли мембрана може знаходитись в одному з двох можливих електричних станів. Водночас, максимальна кількість стабільних стаціонарних станів нейрона із моностабільною сомою та двома активними бі-стабільними дендритами зростає до трьох.

1.2.1 Активні електричні характеристики нейронів

Здатність нейрона знаходитися в одному з двох можливих електричних станів – низької (downstate) або високої (upstate) деполяризації – називають “бі-стабільністю” [29], [30], [41], [42], яку зазвичай пов’язують із *N*-подібною формою вольт-амперної характеристики (ВАХ) мембрани, що відповідає протіканню постійного іонного струму через канали вхідного струму, які неінактивуються або повільно інактивуються. Як випливає з досліджень на модельних пірамідальних нейронах [32], [43] кількість стабільних стаціонарних станів може бути більшою за два. Наявність нестаціонарних мультистабільних станів нейрона можлива завдяки швидким іонним струмам, канали яких здатні до інактивації, до яких відносять швидкі Na^+ струми (i_{Na}). Вони мають *N*-подібну характеристику лише впродовж невеликого проміжку часу [35], [41].

Спрощена модель нейрона із двома бі-стабільними дендритами із геометричною асиметрією та лінійною апроксимацією функції тонічної активації може мати три стійких електричних стани. Отримані кількісні співвідношення параметрів три-стабільної структури можуть бути використані для побудови більш складної моделі, із більшою кількістю стійких станів [33], [41]. Головні питання, які можна поставити, ураховуючи попередні

дослідження: чи приводить ускладнення структури дендритного дерева до збільшення кількості можливих стійких стаціонарних станів та як нестаціонарний характер стійких станів рівноваги впливає на процеси генерації вихідних сигналів нейрону.

Три-стабільність клітини в цілому може бути забезпечена за рахунок електричного з'єднання моностабільної соми, що змінює потенціал мембрани до низького рівня деполяризації, та двох бі-стабільних дендритів, що знаходяться при різних мембранних потенціалах. Комбінація внеску заряду від дендритів породжує додатковий стійкий стан (mid-state) [43]. Як правило, незалежно від складності структури дендритів, перший стійкий рівноважний стаціонарний стан (upstate) відповідає рівню високої стійкої деполяризації мембрани та пов'язаний із процесом лавиноподібного відкриття іонних каналів, а інший – відповідно низької деполяризації, він близький до потенціалу спокою при низькому рівні активації дендритів нейрону. Під активацією треба розуміти інтенсивність однорідної тонічної активації *NMDA*-чутливих синаптичних входів, суцільно розподілених поверхнею дендритів, тобто такі процеси активації, які дозволяють не враховувати точкові характеристики окремих синаптичних контактів [31], [35].

Існування проміжних стійких стаціонарних станів можливе завдяки різному внеску у результуючий струм від струмів з боку окремих дендритних компартментів із асиметричною геометрією. Кількість та характер стійких стаціонарних станів, за умови однорідних електричних властивостей, залежить від співвідношення геометричних параметрів сестринських дендритів.

Моделі нейронів з різними дендритними розгалуженнями мають однакові електричні характеристики мембрани та складаються з двох головних типів мембранної провідності, які є мінімально необхідними для локальної електричної бі-стабільності: перша – це пасивна провідність мембрани G_p, яка пов'язана із джерелом потенціалу спокою $E_p = -95$ мВ, та інша – активна, потенціал-залежна провідність $G_s(E)$. Дендритні компартменти моделей мають однакову активну складову, яка відповідає *NMDA*–типу провідності

глутаматергічних збуджуючих синапсів з потенціалом рівноваги E_s = 0 мВ.

Моделювання мембранних механізмів для *NMDA*-типу провідності та механізму пасивної мембранної провідності реалізується зазвичай мовою програмування Python у середовищі NEURON [34].

Морфологічна структура реального нейрона пірамідальної клітини базується на даних, отриманих у роботі [32]. Модель складається із соми, 11 первинних нейритів, 87 дендритних гілок із сумарною довжиною, що дорівнює $l_{сум}$ = 17667.6 мкм. Характеристики структурних частин представленої моделі: діаметр соми d_c = 25 мкм, довжина l_c = 35 мкм, площа поверхні S_c = 2747.9 мкм^2. Аксональна частина нейрона складається з тригерної зони, мієлінізованих сегментів із маленькою мембранною провідністю і низькою провідністю для іонів Na^+ та немієлінізованих сегментів – перехватів Ранв'є, багатонаселених Na^+-каналами.

Представлена модель характеризує електричні властивості мембрани соми, дендритів та мієлінізованого аксона пірамідального нейрону (рис. 1.2). Внутрішній опір соми R_a = 150 Ом·см, питома ємність мембрани C_m = 0.75 мкФ/см^2.

Загальна провідність соматичної мембрани складається з неспецифічної провідності g_{pas} = 1/30000 См/см^2 та провідностей коктейлю іонних каналів: натрієвих інактивуємих каналів та калієвих неінактивуємих каналів моделі Ходжкіна-Хакслі з провідностями для натрію g_{Na} = 20 пСм/мкм^2, та для калію g_k = 0.1 пСм/мкм^2 відповідно; кальцій-залежних калієвих каналів [29] g_{kca} = 3 пСм/мкм^2, кальцієвих каналів [30] із максимальною провідністю g_{ca} = 0.3 пСименс/мкм^2. Модель враховує явище кальцієвої динаміки через мембрану та відповідну зміну концентрації іонів кальцію у примембранному шарі внутрішньоклітинного простору.

Калієві мускаринові канали мають провідність для іонів калію g_{km}=0.1 пСм/мкм^2. Кальцієву динаміку та відповідну зміну концентрації іонів кальцію у примембранному шарі внутрішньоклітинного простору розраховують

за допомогою мембранного механізму *cad*, реалізованого мовою програмування Python [34].

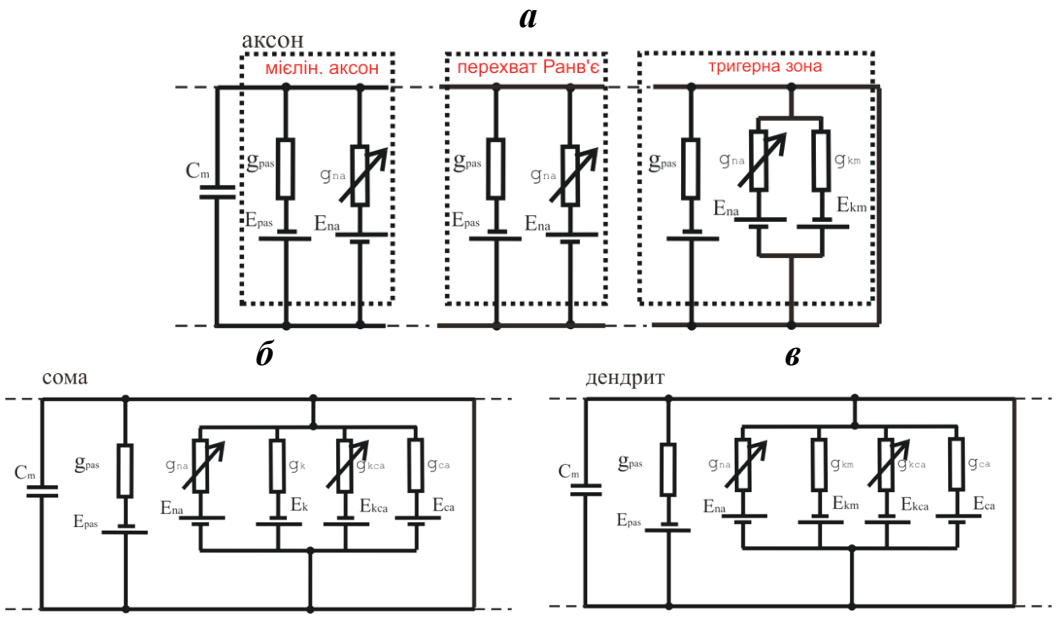


Рисунок 1.2 – Еквівалентні електричні кола та характеристики моделі пірамідального нейрона: *а* – аксон, *б* – сома, *в* – дендрит [24]

Питомі внутрішні опори всіх аксональних сегментів однакові та дорівнюють R_a = 150 Ом·см, мембранна питома ємність тригерної зони, мієлінізованих сегментів та перехватів Ранв'є C_m = 0.75 мкФ/см^2, C_m = 0.04 мкФ/см^2 та C_m = 0.75 мкФ/см^2 відповідно. Провідності мембрани аксону відрізняються для різних сегментів. Пасивні провідності перехватів Ранв'є та мієлінізованих сегментів g_{pas}=0.02 См/см^2 та g_{pas}=1/30000 См/см^2 відповідно. Специфічна провідність початкового аксонального сегменту створюється натрієвим та калієвим струмами при цьому максимальні провідності відповідних каналів g_{Na} = 3000 пСм/мкм^2 та g_k = 2000 пСм/мкм^2. Іонні струми мієлінізованих сегментів та перехватів Ранв'є мають однакову складову, що визначається струмом через натрієві канали із кінетикою моделі Ходжкіна-Хакслі, але різні значення максимальних провідностей: g_{Na} = 20 пСм/мкм^2 для мієлінізованих сегментів та g_{Na} = 30000 пСм/мкм^2, для перехватів Ранв'є. Усі мембранні механізми для

середовища NEURON реалізовані мовою програмування Python [34], та готові для подальшого використання у більш складних моделях.

Для моделювання електричних процесів та впливу геометричних параметрів у штучних нейронах розглядають еквівалентне електричне коло нейронної структури із елементарним дендритним розгалуженням (рис. 1.3).

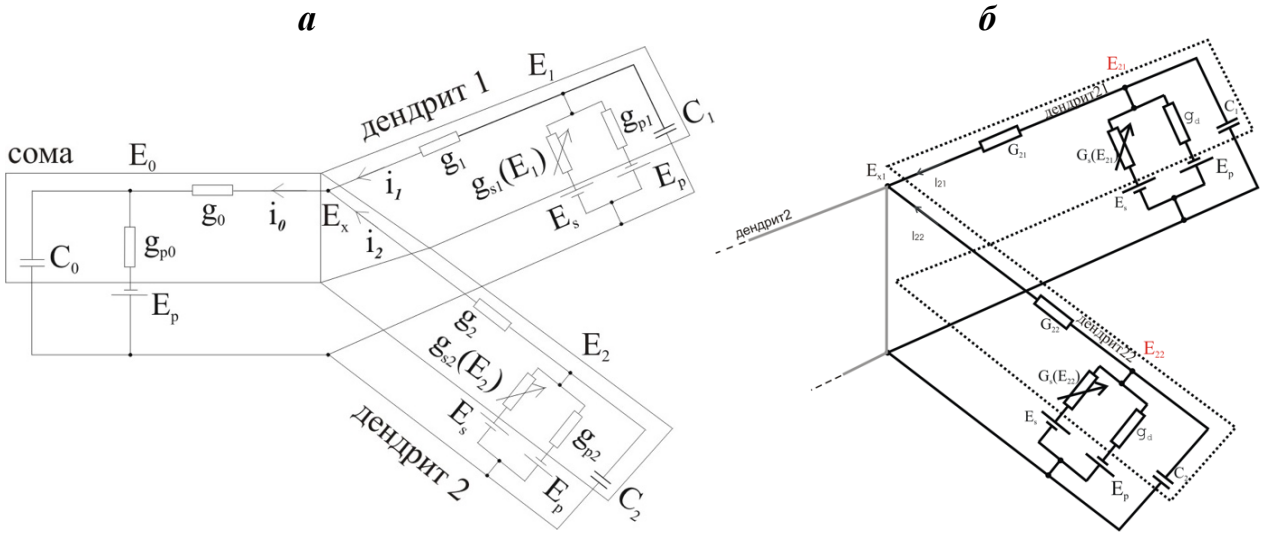


Рисунок 1.3 – Еквівалентне електричне коло та характеристики моделі нейрону із асиметричною структурою: *а* – три-компартментальна модель; б – модель із дендритами другого порядку [24]

Аксо-соматична частина моделі “сома” та два дендрити із асиметричною геометрією “дендрит 1” та “дендрит 2”, представлені з’єднаними циліндричними компартментами (рис. 1.3, *а*). Геометрична асиметрія дендритів полягала в різниці їх довжин, та за умови рівних діаметрів [24], [25]. Моделі із дендритами другого порядку утворені з три-компартментальної моделі, при біфуркації одного, або двох первинних дендритів із двома дочірніми дендритами. Аксо-соматична частина моделі “сома” та два дендрити першого порядку “дендрит 1” та “дендрит 2”. Дендритний компартмент “дендрит 2” має біфуркаційне розгалуження: два дочірніх дендрити “дендрит 21” та “дендрит 22”, аналогічно, представлені з’єднаними циліндричними компартментами (рис. 1.3, *б*).

Аксо-соматичний компартмент (сома) складається тільки з пасивної мембранної провідності $g_{p0} = G_p \cdot S_0$, а два дендритних компартмента (“дендрит 1”

та “дендрит 2”), має пасивні $g_{p1} = G_p \cdot S_1$, $g_{p2} = G_p \cdot S_2$ і потенціал-залежні провідності *NMDA*-типу, які є провідностями каналів глутаматергічних синапсів $g_{s1}(E) = G_s(E) \cdot S_1$ та $g_{s2}(E) = G_s(E) \cdot S_2$. Питома ємність C = 1 мкФ/см^2, цитоплазматичний опір R_i = 100 Ом·см та пасивна мембрана провідність G_p = 0.677 мкСм/см^2 обрані однорідні для всієї клітини. Аналогічно, дочірні дендрити мають пасивні $g_{pi1} = G_p \cdot S_{i1}$, $g_{pi2} = G_p \cdot S_{i2}$, та потенціал-залежні провідності *NMDA*-типу $g_{si1}(E) = G_s(E) \cdot S_{i1}$ та $g_{si2}(E) = G_s(E) \cdot S_{i2}$, де i – це номер первинного дендритного компартменту. При суцільно розподіленому цитоплазматичному опорі R_i об’єднуючі провідності будуть обчислюватися у наступний спосіб:

$$g_0 = \frac{\pi D_0^2}{4 R_i l_0}, \quad g_1 = \frac{\pi D_1^2}{4 R_i l_1}, \quad g_2 = \frac{\pi D_2^2}{4 R_i l_2}, \quad g_{ij} = \frac{\pi D_{ij}^2}{4 R_i l_{ij}}, \tag{1.1}$$

де i = 1, 2 та j = 1, 2 – номер дендриту другого порядку.

Стани компартментів характеризувалися їх трансмембранними потенціалами E_0, E_1 E_2, E_{i1} і E_{i2} та струмами I_{m0}, I_{m1}, I_{m2}, I_{mi1} та I_{mi2}, якщо вони були ізольовані. I_0, I_1, I_2, I_{i1} та I_{i2} у випадку, коли вони об’єднані [24], [25].

1.2.2 Локальні вольт-амперні характеристики компартментів моделей нейронів

Кількість стабільних станів та їх стійкість для окремих ізольованих та об’єднаних компартментів визначається кількістю точок нульового струму на відповідно локальних ВАХ та глобальній вхідній ВАХ$_{вх}$, якщо спостерігати з боку соматичної частини нейрону ($E_{in} = E_0$).

Умови для різної кількості стабільних станів клітини в цілому були визначені на основі обчислення міжкомпартментальних струмів обміну. Кусково-лінійна апроксимація кінетичної функції активації дозволяє отримати розв’язки системи рівнянь моделі та знайти нульові точки на локальних ВАХ та вхідній ВАХ$_{вх}$ [24], [25], [43].

Стаціонарні локальні ВАХ для одиниці площі аксо-соматичного та дендритної мембрани (рис. 1.4, *а, б*) були обчислені у такий спосіб :

$$J_{m0} = G_p\big(E - E_p\big), \tag{1.2}$$

$$J_{m1} = J_{m2} = J_{m21} = J_{m22} = G_p\big(E - E_p\big) + G_s(E)(E - E_s),$$

де J_m – густина струму для кожного компартменту штучного нейрона;

E – мембранний потенціал;

G_p – пасивна мембранна провідність;

G_s – синаптична мембранна провідність.

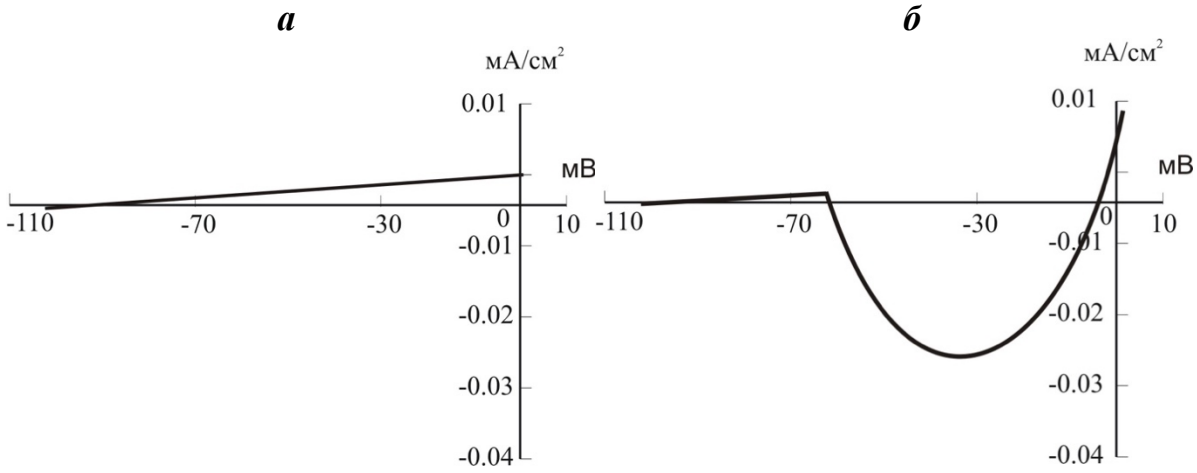


Рисунок 1.4 – Локальні ВАХ для мембрани одиничної площини: *а* – аксо-соматичної, *б* – дендритної. Вісь ординат – густина струму, мА/см^2. Вісь абсцис – мембранний потенціал, мВ [24]

Локальні ВАХ для ізольованих аксо-соматичної та всіх дендритних компартментів в цілому визначаються відповідно:

$$\begin{aligned} I_{m0}(E) &= G_p(E - E_p)\pi D_0 l_0, \\ I_{m1}(E) &= \big[G_p\big(E - E_p\big) + G_s(E)(E - E_s)\big])\pi D_1 l_1, \\ I_{m2}(E) &= \big[G_p\big(E - E_p\big) + G_s(E)(E - E_s)\big])\pi D_2 l_2\ , \\ I_{m21}(E) &= \big[G_p\big(E - E_p\big) + G_s(E)(E - E_s)\big])\pi D_{21} l_{21}, \\ I_{m22}(E) &= \big[G_p\big(E - E_p\big) + G_s(E)(E - E_s)\big])\pi D_{22} l_{22}, \end{aligned} \tag{1.3}$$

де I_m – струм відповідних компартментів нейрона;

E – мембранний потенціал;

G_p – пасивна мембранна провідність;

G_s – синаптична мембранна провідність;

D – діаметр компартменту;

l – довжина компартменту.

Графіки залежності $I_{m0}(E)$, $I_{m1}(E)$ та $I_{m2}(E)$ показані на рис. 1.5 (суцільні лінії). ВАХ для поєднаних компартментів з урахуванням струмів обміну були відповідно

$$I_0 = I_{m0} + i_0 = I_{m0} + (i_1 + i_2),\ \ I_1 = I_{m1} - i_1\,,$$
$$I_2 = I_{m2} - i_{20} + (i_{21} + i_{22}), \qquad (1.4)$$
$$I_{21} = I_{m21} - i_{21},\ I_{22} = I_{m22} - i_{22}$$

де i_0, i_1, i_2 – аксіальні струми обміну між відповідними компартментами (з мембранними потенціалами E_0, E_1 та E_2, відповідно) і точкою розгалуження (з мембранним потенціалом E_x);

i_{21}, i_{22} – струми сполучення дендритів.

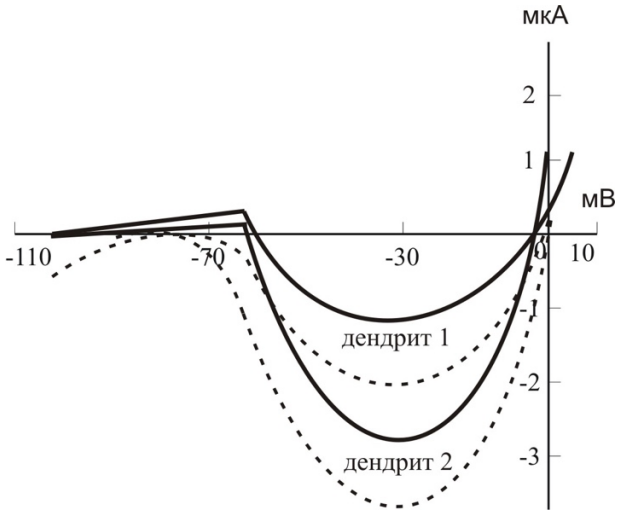


Рисунок 1.5 – ВАХ ізольованих дендритних компартментів (суцільні лінії) та соматопетальні струми обміну (штрихові лінії) для дендритів 1 та 2 (вища та нижча криві, відповідно). Вісь ординат: струм компартменту, мкА. Вісь абсцис: мембранний потенціал, мВ [24]

Ці значення були обчисленні з наступних виразів:

$$i_0 = g_0(E_0 - E_x),\ i_1 = g_1(E_1 - E_x),$$
$$i_2 = g_2(E_2 - E_x), i_{21} = g_{21}(E_{21} - E_{x2}), \qquad (1.5)$$
$$i_{22} = g_{22}(E_{22} - E_{x2}),$$

Відповідно до правила Кірхгофа: $i_0 + i_1 + i_2 = 0$. За законом Ома: $E_x = (E_0g_0 + E_1g_1 + E_2g_2)/(g_0 + g_1 + g_2)$, де E_{x2} – потенціал точки розгалуження дендрита (“дендрит 2”).

1.2.3 Електричні профілі трансмембранного потенціалу дендритів мультистабільних нейронів

Як зазначено раніше різну кількість стабільних станів клітини визначають струми обміну між окремими компартментами. Якщо сусідні компартменти, заряд з яких стікається в одну точку, мають однакові геометричні параметри, то їх внесок заряду буде однаковий. У випадку асиметрії параметрів аксіальні провідності компартментів (1.1) розрізняються, це визначає різні латеральні струми обміну згідно з (1.5). Більш того, асиметричні компартменти знаходяться при різних значеннях трансмембранних потенціалів (E_1, E_2, E_{21} і E_{22} для нейрона з чотирма дендритами). Стабільні рівноважні стани для моделей із різною дендритною структурою визначалися за допомогою розробленого метода. Раніше було показано [24], [25], [31], [33], що нейрон із двома первинними дендритами за умови їх геометричної асиметрії може мати три стабільні стаціонарні стани. Ускладнення структури додаванням двох дочірніх дендритів із взаємною асиметрію за певних співвідношень їх довжин, приводить до появи другого додаткового стабільного (mid-state) стану (рис. 1.5, *б*).

Дендритні компартменти знаходяться під дією тонічної синаптичної активації, тому їх мембрана більш деполяризована ніж мембрана аксо-соматичної частини. Сумарна провідність ізольованої дендритної мембрани, та ефективний рівноважний мембранний потенціал дендритів визначаються, відповідно:

$$
\begin{aligned}
&G_m(E) = G_p + G_s(E),\\
&E_{q1} = \left[\frac{G_s(E)}{G_m+G_1}E_s + \frac{G_p}{G_m+G_1}E_p + \frac{g_1}{G_m+g_1}E_x\right],\\
&E_{q3} = \left[\frac{G_s(E)}{G_m+G_2}E_s + \frac{G_p}{G_m+G_2}E_p + \frac{g_2}{G_m+g_2}E_x\right].
\end{aligned}
\tag{1.6}
$$

За умов гомогенності електричних властивостей та симетрії геометричної структури дендритів, їх потенціали будуть рівні $E_{q1} = E_{q2}$. Геометрична асиметрія приводить до протікання різних латеральних струмів сполучення. У моделі три-стабільного нейрона спостерігався розподіл мембранного потенціалу вздовж всієї структури для трьох стійких стаціонарних станів, показаний на рис. 1.6. Рис. 1.6, *а* ілюструє електричний профіль для стійкого стану високої деполяризації upstate, обидва дендрити мають сильно деполяризовану мембрану, потенціали обох дендритів перевищують потенціал активації *NMDA*-каналів $E_{a1} < E_{1,2} < E_{a2}$ [11], де E_{a1} – потенціал порогу для відкритого стану вхідних синаптичних каналів; E_{a2} – потенціал, при якому всі *NMDA*-канали звільнюються від Mg^{2+} блоку. Значення $E_{a1} = –56$ та $E_{a2} = 10$ мВ відповідають NMDA-чутливим каналам [31], [35], [40].

Якщо струм i_1 не дорівнює струму i_2, то згідно з виразом (1.4) трансмембранні струми I_1 та I_2 також відрізняються, при цьому електричні профілі для дендритів розходяться відповідно до рівняння (1.6).

У середньому стабільному стані mid-state (рис. 1.6, *б*) спостерігається максимальна різниця потенціалів дендритів. У даному випадку $E_1 = –51$ мВ, $E_1 < E_{a1}$, $E_2 = –9$ мВ, з $E_2 > E_{a1}$ та потенціал “дендрита 1” прямує до стабільного стану downstate. Рис. 1.6, *в* показує електричні профілі для стабільного стаціонарного стану нейрону downstate [24], [25], [31].

Таким чином, структура дендритів визначає їх гетерогенність у електричній поведінці: чим більша геометрична асиметрія дендритів, тим більша різниця між їх електричними станами. Згідно з виразами (1.4), визначені провідності сполучення дендритів: $g_0 = 34$ мкСм/см^2, $g_1 = 0.013$ мкСм/см^2, $g_2 = 0.031$ мкСм/см^2; а згідно з (1.5) – соматопетальні струми сполучення для кожного стабільного стану upstate, mid-state та downstate відповідно: $i_1 = 0.31$, $i_2 = 0.4$, $i_1 = 0.61$ $i_2 = 0.23$, $i_1 = 0.23$, $i_2 = 0.3$ мкА.

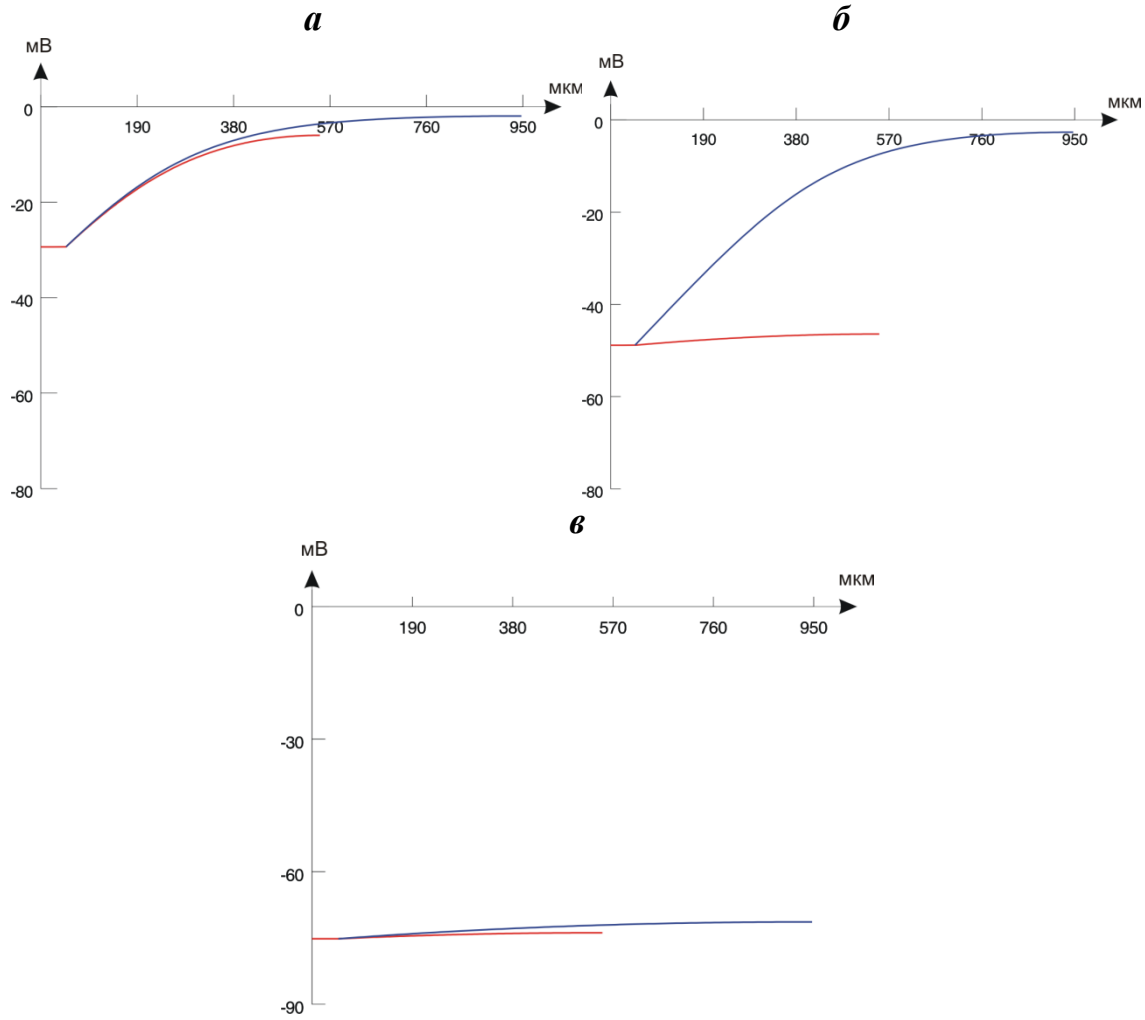


Рисунок 1.6 – Профілі мембранного потенціалу компартментів три-стабільного нейрона для стабільних стаціонарних станів: *а* – upstate, *б* – mid-state, *в* – downstate. Синя крива - розподіл потенціалу “дендрит 2”, червона – “дендрит 1”. Вісь ординат: мембранний потенціал, мВ. Вісь абсцис: довжина, мкм [24]

Розподіли мембранного потенціалу розраховані для моделі із чотирма дендритами. Модель із квадри-стабільністю використана для розрахунку величин струмів обміну. Електричні профілі для нейрона із чотирма дендритами показані на рис. 1.7.

Стабільні стани upstate та downstate (рис. 1.7, *а*, *г*) встановлювалися на рівнях E_{in} = –27 мВ та E_{in} = –80 мВ. Проміжні стани стабільності супроводжуються значною різницею потенціалів окремих дендритів. Слід зазначити, що струми сполучення між компартментами дендритів із

максимальною електричною асиметрією мають найбільший вплив на виникнення стабільного стану.

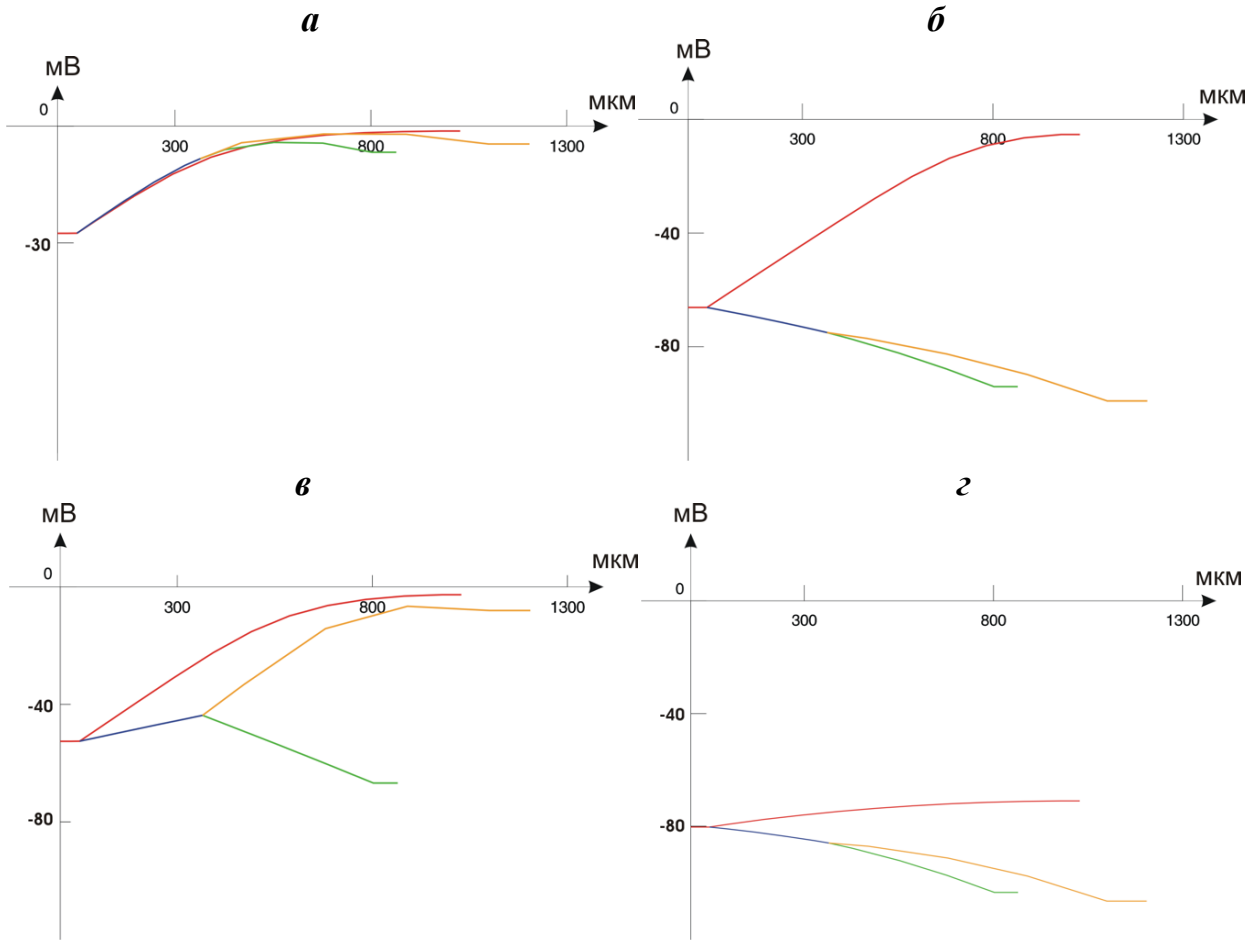


Рисунок 1.7 – Профілі мембранного потенціалу вздовж нейрона із чотирма дендритами для різних стабільних станів: *а* – upstate, *б* – mid-state, *в* – mid-state, *г* – downstate. Синя крива – розподіл потенціалу “дендрит 2”, червона – “дендрит 1”, зелена та жовта – сестринські дендрити “дендрит 21” та “дендрит 22”. Вісь ординат: мембранний потенціал, мВ. Вісь абсцис: довжина, мкм [24]

Для стаціонарних станів рівноваги розраховувано величини латеральних струмів обміну, мінімально необхідні для їх стабільності. У випадку наведеному на рис. 1.6, латеральні електричні провідності становлять: g_0 = 34 мкСм/см^2, g_1 = 0.013 мкСм/см^2, g_2 = 0.031 мкСм/см^2, g_{21} = 0.033 мкСм/см^2 та g_{22} = 0.009 мкСм/см^2. Відповідно, для кожного вузла моделі струми обміну у даному випадку дорівнюють: i_1 = 0.6 мкА, i_2 = 0.21 мкА, i_{21} = 0.1 мкА та i_{22} = 0.26

мкА. Для стаціонарного стабільного стану mid-state аксіальні струми дендритів: $i_1 = 0.84$ мкА, $i_2 = 0.35$ мкА, $i_{21} = 0.13$ мкА та $i_{22} = 0.05$ мкА.

Для визначення ролі струмів сполучення між окремими дендритами у формуванні мультистабільних станів модель нейрону модифіковано, додаванням двох сестринських дендрити до другого первинного дендриту. Довжина “дендрита 1” l_1 змінювалася та набувала трьох фіксованих значень: 920, 830 та 395 мкм (рис. 1.8).

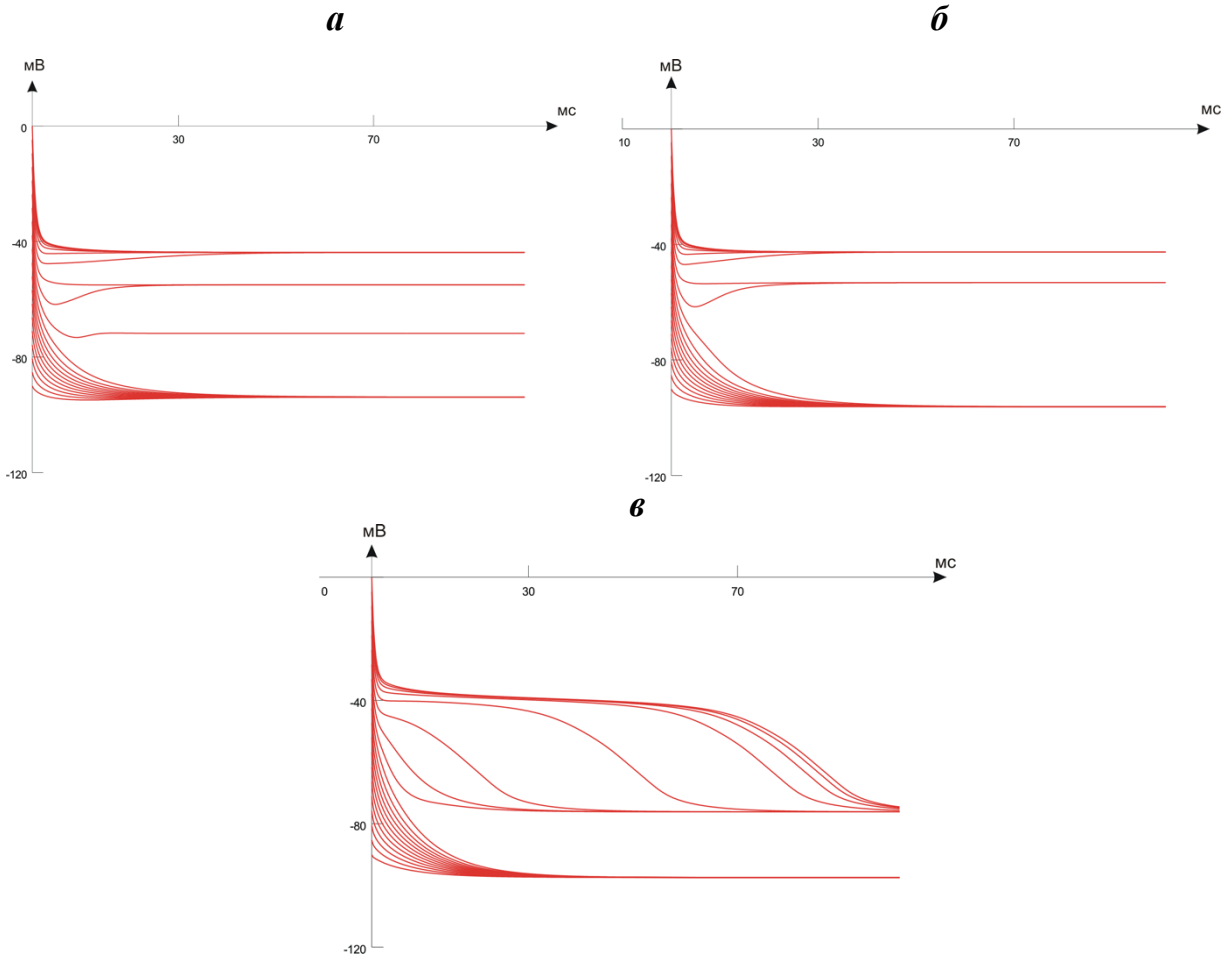


Рисунок 1.8 – Стабілізація трансмембранного потенціалу соми на різних рівнях залежно від однорідних початкових умов при фіксованому значенні $l_2 = 395$ мкм для трьох значень довжини l_1 дендрита (в мкм): *а* – 920, *б* – 830, *в* – 395. Вісь ординат: мембранний потенціал соми, мВ. Вісь абсцис: час, мс [25]

Можна запропонувати наступну гіпотезу: якщо знов зробити структуру дендритного дерева нейрона симетричною додаванням ще одного дендритного розгалуження (у розглянутому прикладі $l_2 = 395$ мкм, $l_{21} = l_{11} = 210$ мкм, $l_{22} = l_{12} =$

770 мкм), то латеральні струми від дендритів (“дендрит 1” та “дендрит 2”) будуть врівноважені $i_1 = i_2$ (рис. 1.7, *в*). Таким чином, нейрон із симетричних дендритним деревом буде завжди лише бі-стабільним у стаціонарному випадку. При збільшенні довжини “дендрит 1” до $l_1 = 830$ мкм нейрон має три стабільних стани (рис. 1.8, *б*), подальше збільшення асиметрії при $l_1 = 920$ мкм приводить до квадри-стабільності (рис. 1.8, *а*).

Результуючі латеральні струми для представлених модифікацій структури (рис. 1.7, *а*, *б*, *в*) дорівнюють відповідно: $i_1 = 0.26$ мкА, $i_2 = 0.67$ мкА, $i_1 = 0.4$ мкА та $i_2 = 0.52$ мкА, $i_1 = 0.408$ та $i_2 = 0.41$ мкА.

Залежності між мембранним потенціалом та густиною струму, при соматичних та дендритних компартментах ізольованих один від іншого, можуть бути визначені як локальні мембранні властивості індивідуальних компартментів, які ідентичні локальним ВАХ для одиниці площі поверхні. Підставимо значення функції $F(E)$ у рівняння (1.2) та розв’яжемо цю систему відносно струму для визначення нульових точок струму локальної ВАХ для дендритів:

$$G_p\left(E - E_p\right) = 0;\ E_p < E \le E_{a1}, \tag{1.7}$$

$$G_p\left(E - E_p\right) + G_s \frac{(E_{a1} - E)}{(E_{a1} - E_{a2})}(E - E_s) = 0;\ E_{a1} < E < E_s$$

Розв’язками цих рівнянь є значення (в мВ): $E_1 = -95$, $E_2 = -54.9$ та $E_3 = -2.5$. Вони показані на ВАХ мембрани дендрита (рис. 1.9) , яка має три точки нульового струму, дві з яких знаходяться на позитивному нахилі кривої, та є стійкими, у той час як інша точка – на участку від’ємного нахилу кривої, тому вона нестійка (рис. 1.8).

Поєднання аксо-соматичного та дендритних компартментів призводить до появи латеральних струмів i_0, i_1, i_2, i_{21}, i_{22} які дають додатній або від’ємний внесок у трансмембранні струми відповідних компартментів.

Синаптичні трансмембранні струми деполяризують мембрану дендритів, що приводить до генерації соматопетальних аксіальних струмів i_1 та i_2, (рис. 1.8, штрихові лінії) які протікають з ділянок більшої деполяризації у дендритах до

ділянок із меншою деполяризацією (E_0). Метрична асиметрія дендритів у відповідності тільки до різниці їх довжин приводить до різних провідностей дендритних компартментів g_1 та g_2, котрі, як і провідності соматичного компартменту, залежать від геометрії. Відповідно, для різних заданих трансмембранних потенціалів кожен компартмент генерує різні струми, залежно від того, чи були вони з'єднані, чи ні з іншими компартментами. Це можна розглядати як зсув локальної ВАХ вгору, якщо струм обміну додатній, або донизу, якщо від'ємний.

Електричне об'єднання соми та дендритної структури змінює стабільні стаціонарні потенціали кожного компартмента, (E_0, E_1, E_2, E_{21}, E_{22} відповідно) у порівнянні з потенціалами, які спостерігаються на локальних стабільних ВАХ цих компартментів, якщо вони ізольовані один від інших. Ці нові значення можуть бути обчислені виходячи із рівнянь обміну:

$$\begin{cases} I_0(E_0) = G_p \pi D_0 l_0 (E_0 - E_p) + g_0 (E_0 - E_x), \\ I_1(E_1) = \left[G_p (E_1 - E_p) + G_s(E_1)(E_1 - E_s) \right] \pi D_1 l_1 - g_1 (E_1 - E_x), \\ I_2(E_2) = \left[G_p (E_2 - E_p) + G_s(E_2)(E_2 - E_s) \right] \pi D_2 l_2 - \frac{g_2}{2}(E_2 - E_x) + \frac{g_2}{2}(E_2 - E_{x2}), \\ I_{21}(E_{21}) = \left[G_p (E_{21} - E_p) + G_s(E_{21})(E_{21} - E_s) \right] \pi D_{21} l_{21} - g_{21} (E_{21} - E_{x2}), \\ I_{22}(E_{22}) = \left[G_p (E_{22} - E_p) + G_s(E_{22})(E_{22} - E_s) \right] \pi D_{22} l_{22} - g_{22} (E_{22} - E_{x2}). \end{cases} \quad (1.8)$$

Як можна бачити з електричних профілів дендритів (рис. 1.7), можливі такі стани:

– якщо мембрані потенціали в дендритах були меншими за потенціал активації каналів типу *NMDA* E_{a1}, тобто $E_p < E_{1,2,21,22} < E_{a1}$, тоді в дендритах протікають лише пасивні струми, які мають ВАХ з позитивним постійним нахилом та стійким електричним станом в області гіперполяризації (downstate);

– при активації лише "дендрит 1", із найбільшим опором ($E_{a2} > E_1 > E_{a1}$) у системі встановлюється новий стійкий розподіл струмів. На вхідній $ВАХ_{вх}$, ця область відповідає значному позитивному зсуву кривої;

– подальше збільшення мембранного потенціалу приводить до активації лише “дендрита 22” за рахунок шунтування “дендрита 2” коротшим “дендритом 21” ($E_{a2} > E_{22} > E_{a1}$) та ($E_p < E_{21} < E_{a1}$). Більший внесок заряду з дендритів при цьому зумовлює більший зсув $ВАХ_{вх}$, ніж у попередньому випадку;

– за високої деполяризації мембрани потенціали всіх дендритів перевищують рівень NMDA-активації та ($E_{a1} < E_{1,2,21,22} < E_{a2}$), це відповідає ділянці позитивного нахилу $ВАХ_{вх}$.

Треба зазначити, що додатковий від’ємний нахил $ВАХ_{вх}$ можливий, якщо електричні стани “дендрита 21” та “дендрита 22” відрізняються достатньо для виникнення сумарного позитивного струму в сому. Другий та третій з описаних стабільних станів визначаються рівняннями для струмів:

$$\begin{cases} I_0(E_0) = G_p \pi D_0 l_0 (E_0 - E_p) + g_0(E_0 - E_x) = \\ = G_p \pi D_0 l_0 (E_0 - E_p) + g_1(E_1 - E_x) + g_2(E_2 - E_x), \\ I_{21}(E_{21}) = G_p (E_{21} - E_p) \pi D_{21} l_{21} - g_{21}(E_{21} - E_{x2}), \\ I_{22}(E_{22}) = \left[G_p (E_{22} - E_p) + G_s \dfrac{(E_{a1} - E)}{(E_{a1} - E_{a2})} (E_{22} - E_s) \right] \pi D_{22} l_{22} - \\ - g_{22}(E_{22} - E_{x2}). \end{cases}$$

(1.9)

$$\begin{cases} I_0(E_0) = G_p \pi D_0 l_0 (E_0 - E_p) + g_0(E_0 - E_x) = \\ = G_p \pi D_0 l_0 (E_0 - E_p) + g_1(E_1 - E_x) + g_2(E_2 - E_x), \\ I_{21}(E_{21}) = \left[G_p (E_{21} - E_p) + G_s \dfrac{(E_{a1} - E)}{(E_{a1} - E_{a2})} (E_{21} - E_s) \right] \pi D_{21} l_{21} - g_{21}(E_{21} - E_{x2}), \\ I_{22}(E_{22}) = \left[G_p (E_{22} - E_p) + G_s \dfrac{(E_{a1} - E)}{(E_{a1} - E_{a2})} (E_{22} - E_s) \right] \pi D_{22} l_{22} - g_{22}(E_{22} - E_{x2}). \end{cases}$$

(1.10)

Розв’язуючи рівняння (1.9) та (1.10) для визначеної моделі, отримаємо струм $I_0 < 0$, графічно показаний на інтервалі потенціалів (–60; –41) мВ на рис. 1.9.

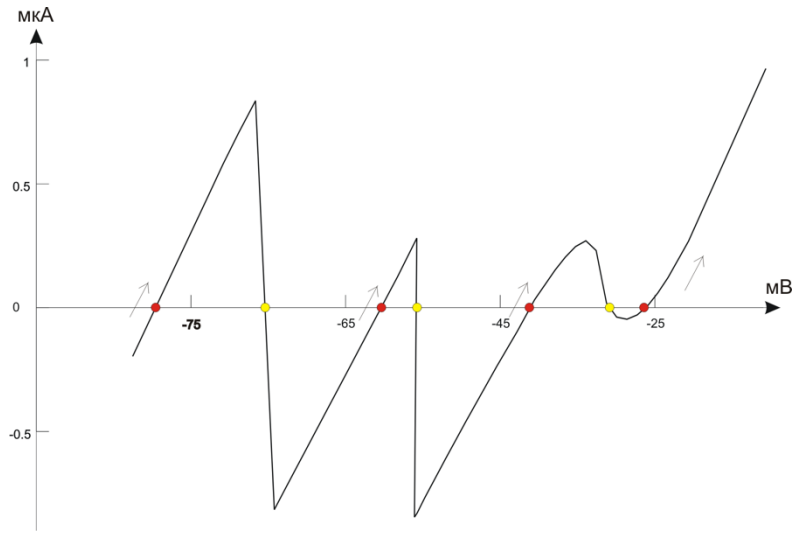


Рисунок 1.9 – Результуюча вхідна $ВАХ_{вх}$ об’єднаних аксо-соматичного компартменту та дендритів, розрахована для окремого випадку довжин (в мкм) l_1=920, l_2=395, l_{21}=210 та l_{22}=770. [24]

Слід зазначити, що три області від’ємного нахилу на стаціонарній ВАХвх та квадри-стабільність клітини можливі тільки за умови взаємної геометричної асиметрії дистальних розгалужень дендритів (“дендрит 21” та “дендрит 22”).

Таким чином, початкова гіпотеза підтверджена – стаціонарна мультистабільна структура нейрона отримана шляхом введення геометричної асиметрії її окремих активних компартментів. Межі існування і кількість станів такої нейронної структури задаються аналітичною системою рівнянь. А отже, це дає змогу синтезувати математичну модель нейрона із активними характеристиками із заздалегідь заданою кількістю стабільних стаціонарних станів.

1.3 Оцінювання наявних апаратно-програмних засобів для аналізу ЕКГ-сигналу

Сучасні пристрої, що дозволяють реєструвати ЕКГ вже не таке великі, громіздкі, як це було раніше. Навіть деякі з повнофункціональних 12-канальних кардіографів сьогодні маленькі, портативні та можуть працювати із настільними і портативними комп'ютерами або мобільними пристроями.

На сьогодні існують ще менші та самодостатні одноканальні пристрої для реєстрації ЕКГ, які є взагалі портативними, кишеньковими, працюють від звичайних акумуляторів і мають свої власні вбудовані дисплеї. Як правило, при роботі із сучасними пристроями потрібно лише правильно розташувати чутливі датчики на грудній клітині та/або зап'ястях і натиснути потрібну кнопку на інтерфейсі приладу. Результати реєстрації ЕКГ зберігаються в пам'яті поряд з попередніми дослідженнями, можуть виводяться на монітор приладу, передаватися на комп'ютер і дистанційно відправлятися в діагностичний центр (на телефон лікаря кардіологічного диспансеру тощо). Габарити і вага (близько 100 г) деяких сучасних моделей портативних кардіографів дійсно, на законних підставах дозволяють називати їх “кишеньковими” (рис. 1.9).

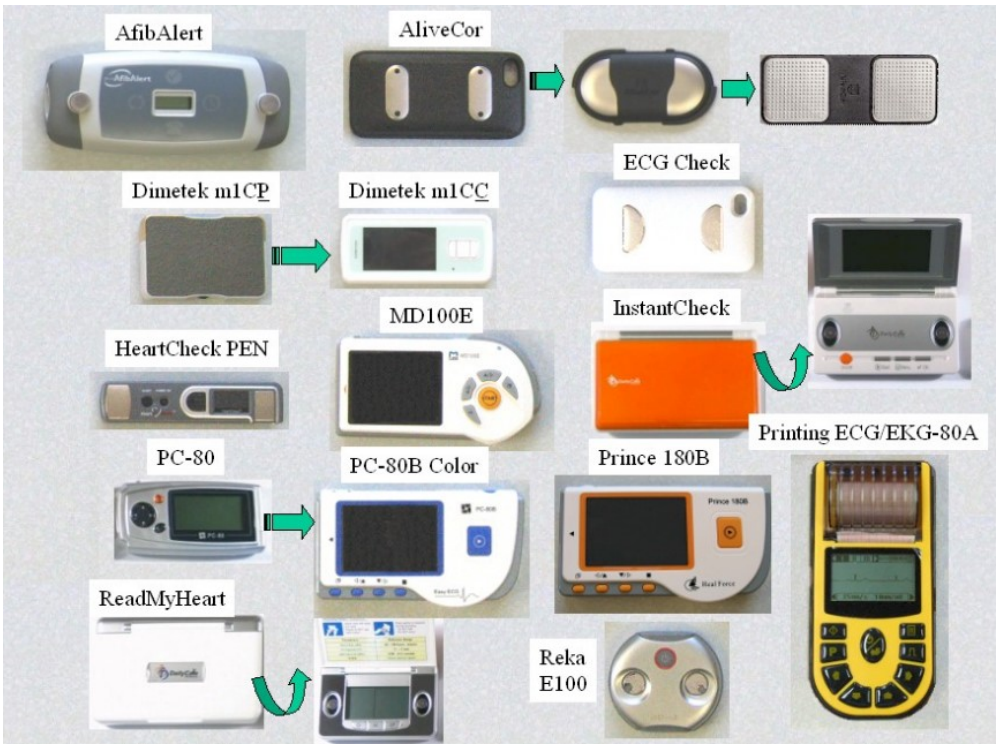


Рисунок 1.10 – Сучасні портативні пристрої для реєстрації ЕКГ-сигналів. [62]

Ціни на такі кардіографи цілком можна порівняти з сучасними цінами на мобільні телефони. Деякі з таких пристроїв: AfibAlert (спеціально призначені тільки для Afib), AliveCor, EKG CheckIn і HeartCheck Pen, EKG REKA, та багато інших. Саме ці пристрої є орієнтованими на використання “хмарних” технологій

з додатками, призначеними для Apple IPhone/IPad і Android смартфонів, та із програмним забезпеченням для ПК, нетбуків і т. ін.

Всі ці пристрої, використовують металеві контакти для шкіри і роблять короткі записи менш ніж за хвилину. Наприклад, пристрій REKA придатний для моніторингу в очікуванні патологічних подій [62], [65], а також має варіанти використання кабелю з клейкими електродами на шкірі для запису сигналу більше однієї хвилини. На ринку починають з’являтися пристрої, які для реєстрації ЕКГ використовують безконтактні електроди [46], [73], що дозволяють значно підвищити можливості застосування кардіо-моніторів, оскільки іноді можна реєструвати потенціали, крізь одежу, або навіть реалізовувати електроди в вигляді елементів одягу (“Smart Clothes”) [73].

а ***б***

Рисунок 1.10 – Приклади носимих серцевих моніторів: *а* – серцевий монітор GARMIN у вигляді нагрудного ременю та годинник, для синхронізації та візуалізації даних; *б* – носимий монітор варіабельності ритму серця SenceBand, що здатен реєструвати електричну активність серця з зап’ястя однієї руки [68]

Також поширеним типом приладів є серцеві монітори, які користувачі носять на зап'ясті у вигляді годинника, або нагрудного реміня (GARMIN) (рис. 1.10, *а)*. Такий монітор невеликий за розміром і не викликає незручностей у користувача. Його монітор в основному використовують у спорті для вимірювання серцевого ритму під час тренування. Недоліком пристрою є те, що

він має дуже високий рівень чутливості, та водночас високий рівень шумів внаслідок артефактів рухів, що призводить до значних перешкод при детектуванні корисного сигналу [87]. Тому він все ще не підходить для використання в якості засобу діагностики захворювань, оскільки його точність в реальних умовах використання викликає сумніви.

Браслет SenceBand (рис. 1.10, *б*) призначений для реєстрації ЕКГ з обмеженої ділянки шкіри руки за допомогою сухих електродів і подальшого розрахунку варіабельності серцевого ритму за допомогою RR-інтервалів [5], [6]. Дані, отримані в браслеті, передаються на Android або iOS-пристрій за технологією Bluetooth 4.2 BLE. Програмне забезпечення дозволяє спостерігати за станом здоров'я людини в безперервному режимі, заміри ЕКГ відбуваються протягом 130 або 260 секунд (опціонально) кожні 30 хвилин, або за запитом користувача (On-demand), але не частіше ніж 1 раз в 5 хвилин. Всі результати зберігаються на віддаленому сервері та автоматично синхронізуються при наявності активного Інтернет-з’єднання. SenceBand фіксує найбільш високоамплітудні коливання ЕКГ – R-піки та обчислює послідовність RR-інтервалів, яку використовують для подальшого аналізу ВСР. На основі якого оцінюють фізіологічний та психо-емоційний стан людини. При роботі алгоритму оцінюється більше ніж 35 математичних параметрів для часового та частотних доменів сигналу варіабельності. Але незважаючи на ергономічний форм-фактор та здатність до безперервного моніторингу кардіоінтервалографії людини у фоновому режимі, в поточній реалізації пристрій не може відтворювати повну структуру морфології ЕКГ-сигналу.

Аналіз поточного стану проблеми дозволяє сформулювати головну мету дисертаційної роботи:

– створити стабільну модель штучної спайкової нейронної мережі (SNN);

– створити нейроморфний модуль на базі ПЛІС-архітектури;

– застосувати програмно-апаратне рішення для вирішення задачі класифікації ЕКГ-сигналів;

– провести тестування та валідацію запропонованої системи реєстрації та

аналізу ЕКГ-параметрів у реальному часі.

Висновки до 1-го розділу

За результатами аналізу літературного контенту виявлено об’єктивні і суб’єктивні фактори, що впливають на якість оброблення ЕКГ-сигналів із застосуванням нейронних мереж, та проведено оцінювання існуючих апаратно-програмних засобів для аналізу ЕКГ-сигналів, результати якого стали основою системної і схемотехнічної реалізації апаратного забезпечення розробленого засобу. Запропоновано модельну реалізацію мультистабільного штучного нейрона із біологічною подібністю (спайковий нейрон), показано можливість реалізації нейрона із 2-а, 3-а та 4-а станами стаціонарної стабільності введенням поняття про геометричні параметри моделі, розглянуті умови існування станів стабільності. Модель нейрона може бути використана для подальшої апаратної реалізації нейроморфного модуля, як базової обчислювальної одиниці нейронної мережі для задачі оброблення та детектування клінічно-значущих характеристик ЕКГ-сигналів.

Результати досліджень цього розділу наведено в таких публікаціях:

[1] S. M. Korogod, and D. V. Chernetchenko, “Nature of electrical tristability in a neuron model with bistable asymmetrical dendrites”, *Neurophysiology*, vol. 40, no. 5/6, pp. 412-416, 2008.

[2] Є.М. Сніжко, Д. В. Чернетченко, “Динаміка електричних потенціалів моделі мережі нейронів із нелінійними функціями активації”, *Вісник Дніпропетровського університету. Фізика. Радіоелектроніка*, т. 20, № 2(19), с. 50-57, 2012.

[3] N. Botsva, I. Naishtetik, L. Khimion, and D. Chernetchenko, “Predictors of aging based on the analysis of heart rate variability”, *Pacing Clinical Electrophysiology*, vol. 40(11), pp. 1269-1278, 2017.

РОЗДІЛ 2
МОДЕЛЮВАННЯ ПРОЦЕСУ СТВОРЕННЯ НЕЙРОМОРФНОГО МОДУЛЯ ДЛЯ ОБРОБЛЕННЯ ЕЛЕКТРОКАРДІОГРАФІЧНИХ СИГНАЛІВ

2.1 Передумови застосування штучних мультистабільних нейронних мереж для розроблення нейроморфного модуля

Метод, який розроблено в дисертаційній роботі, ґрунтується на обчислювальній моделі мультистабільної нейронної мережі (MNN) [24], [41], [53], [92] що складається з блоку попереднього оброблення з рекурентно з'єднаними нейронами і блоку зчитування для декодування внутрішнього стану системи та може здійснювати будь-яке нелінійне перетворення вхідного сигналу. Моделі нейронів внутрішнього шару впроваджено на основі модельних досліджень даного розділу. Внутрішньо система інтегрує часову інформацію в лінійно незалежні стани системи, не обчислюючи точно будь-яке нелінійне перетворення; вихідний шар нейронів не має уявлення про часові аспекти вхідного сигналу і може навчитись відображати у своїх внутрішніх станах відповідь на апроксимацію/класифікацію функцій.

Нижче наведено певні характеристики MNN, які пов’язані з проблемою оцінювання параметрів ЕКГ, яку необхідно вирішити в дисертаційній роботі.

– Вхідні сигнали нейронів в системі реалізовано у вигляді вхідних послідовностей спайків із певною періодичністю. Це потрібно, оскільки морфологічні та деякі інші динамічні особливості ЕКГ є наслідком серцевих змін протягом певного періоду часу.

– Розділення обчислень на внутрішній та вихідний шари дає змогу створювати декілька різних структур (реалізуючи широкий діапазон функціональних можливостей), використовуючи один і той самий підхід. В роботі показано таку конкретну реалізацію, яка використовує алгоритм навчання

“без вчителя”, керований просторово-часовою конфігурацією спайків на вході нейронів.

– MNN дозволяє реалізовувати в реальному часі швидке та надійне навчання з потокових даних, що робить обчислювальну модель придатною спеціально для оцінювання фізіологічних параметрів ЕКГ: ЧСС, окремі зубці, інтервали, тощо (рис 2.1).

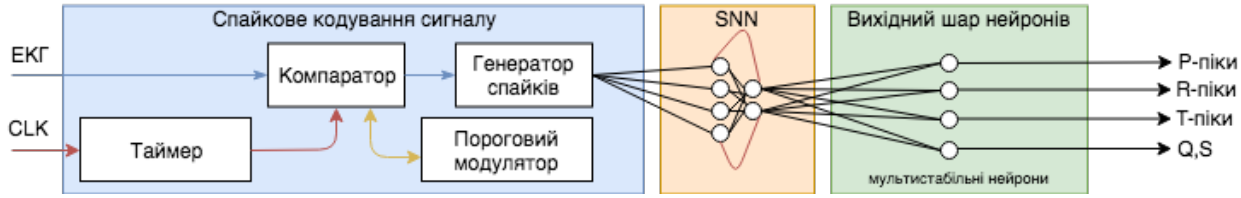


Рисунок 2.1 – Загальна структура запропонованої системи визначення параметрів ЕКГ-сигналу

Спайки, що генеруються шифратором (Спайкове кодування сигналу), використовуються для збудження штучних рекурентно пов’язаних нейронів (Spiking Neuronal Network, або SNN). Рівні станів системи використовують для визначення точних часових міток певних характеристик сигналу (стандартні зубці P-QRS-T комплексу) у блоці зчитування (Вихідний шар нейронів). Головною метою розробки даного методу є можливість реалізації моделі на апаратній базі, що забезпечує створення спеціалізованого нейро-модуля для оброблення ЕКГ-сигналу та виділення його основних просторово-часових характеристик в реальному часі з мінімальними енергозатратами.

Інформація в нейронній мережі може кодуватися з використанням двох методів – кодування на основі частоти спайків та темпорального кодування [12], [18], [21]. Кодування на основі частоти спайків подає закодовану інформацію у вигляді кількості спайків у вікні кодування без урахування часових характеристик сигналу. Кодування на основі частоти спайків було успішно застосовано до завдань просторової класифікації, таких як розпізнавання рукописних цифр [23], [26]. Темпоральне кодування кодує інформацію у інтервали між спайками [19], [20], [53] враховуючи просторово-темпоральну

структуру вхідного сигналу, що зумовило його успішне застосування для оброблення складних сигналів, таких як обробка мови [14], [53] та інтерфейси мозку з машиною на основі електроенцефалографії [14]. Для задач оброблення ЕКГ необхідно брати до уваги часові характеристики комплексів P-QRS-T, що саме і обґрунтовує застосування в цій роботі темпорального, або часового кодування.

Спайковий шифратор кодує вхідний сигнал ЕКГ в інтервал між спайками, використовуючи комбінацію порогових модуляторів, компаратора напруги, генератора спайків і таймера (рис. 2.2).

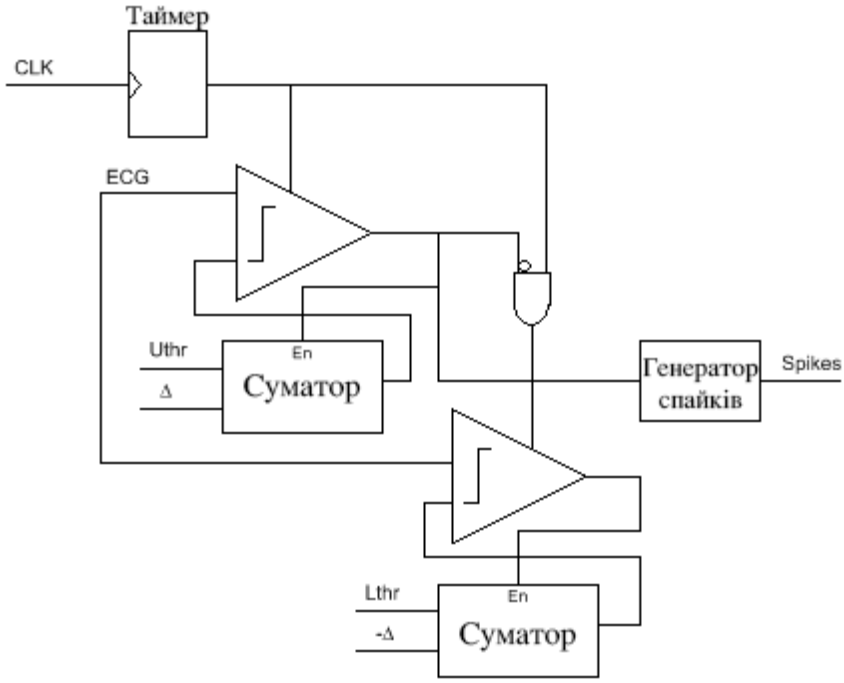


Рисунок 2.2 – Запропонована схема для генерації спайк-послідовності з ЕКГ-сигналу із застосуванням темпорального кодування.

Примітки: CLK – тактуючий сигнал, ECG – вхідний ЕКГ-сигнал, Uthr – верхній поріг, Lthr – нижній поріг, Δ – приріст значення порогу, Spikes – вихідний сигнал.

Порогові модулятори реалізовано з використанням повних суматорів, які, в свою чергу, ефективно можуть бути реалізовані, наприклад, за допомогою тонкоплівкових транзисторів [18] або на базі ПЛІС-архітектури. Таймер може бути реалізований за допомогою D-тригеру типу flip-flops (DFF) та тактової

частоти зовнішнього кварцового генератора для створення інтервалів перемикання в компараторах. Використано інтервали тригерних сигналів 2 мс, що забезпечує роботу спайкового шифратора на частоті 500 Гц. Точність/споживання електроенергії – компроміс, який досягається і регулюється шляхом зміни даного тригерного інтервалу (розділ 4).

На рисунку 2.3 представлена структурна схема такого шифратора з максимальною затримкою поширення 5 мкс. Два аналогових перемикача мультиплексують сигнал, що поступає на компаратор.

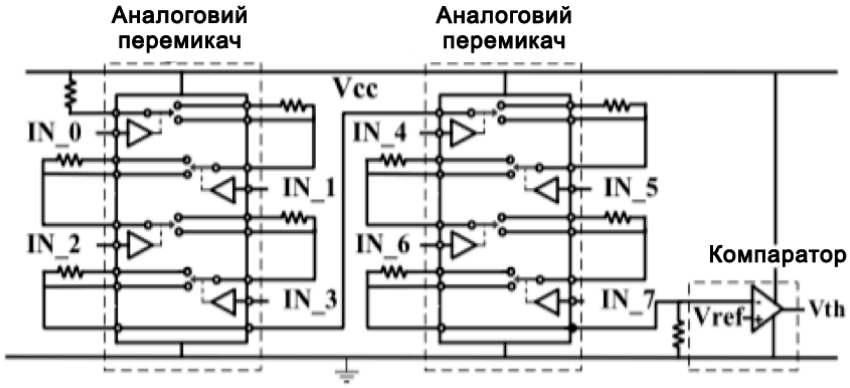


Рисунок 2.3 – Структурна схема шифратора вхідної напруги [18]

Для запобігання ефекту “накладання сигналу” реалізовано метод адаптації інтервалу таймера у відповідь на швидкість зміни вхідного сигналу. Це дозволяє SNN краще розрізнювати дві спайкові пачки і підвищує точність. Один з підходів до адаптації інтервалів таймера полягає в тому, щоб встановити його тактову частоту пропорційною амплітуді найбільшого частотного компоненту Фур'є-перетворення, який отримано з вхідного сигналу. Тактова частота F_{CLK} залежить від швидкості зростання сигналу і може бути представлена як функція з двома рівнями насиченості

$$F_{CLK}(A_{f0}) = \begin{cases} F_{CLK\_min}\ ,\ F_{CLK} \le F_{CLKmin} \\ \frac{(F_{CLK\_max} - F_{CLK\_min})}{k} \cdot (A_{f0} - a) + F_{CLK\_min},\ F_{CLK\_min} < F_{CLK} < F_{CLK\_max} \\ F_{CLK\_max},\ F_{CLK} \ge F_{CLK\_max} \end{cases} \quad (2.1)$$

де $F_{\text{CLK\_max}}$ – це максимум тактової частоти таймера;

$F_{\text{CLK\_min}}$ – мінімум тактової частоти таймера;

A_{f0} – частота вхідного сигналу найбільш значущої гармоніки;

параметри k і a – константи, які належать до певних частотних параметрів вхідного ЕКГ-сигналу.

Для ЕКГ-сигналу параметри дорівнюють k = 14.0 Гц і a = 1.0 Гц відповідно. Мінімальна тактова частота також постійна і дорівнює F_{CLK_min} = 5 Гц, максимальна частота F_{CLK_max} варіюється від 100 до 1,000 Гц.

Нижче наведено блок-схему описаного алгоритму (рис. 2.4) та основні правила роботи.

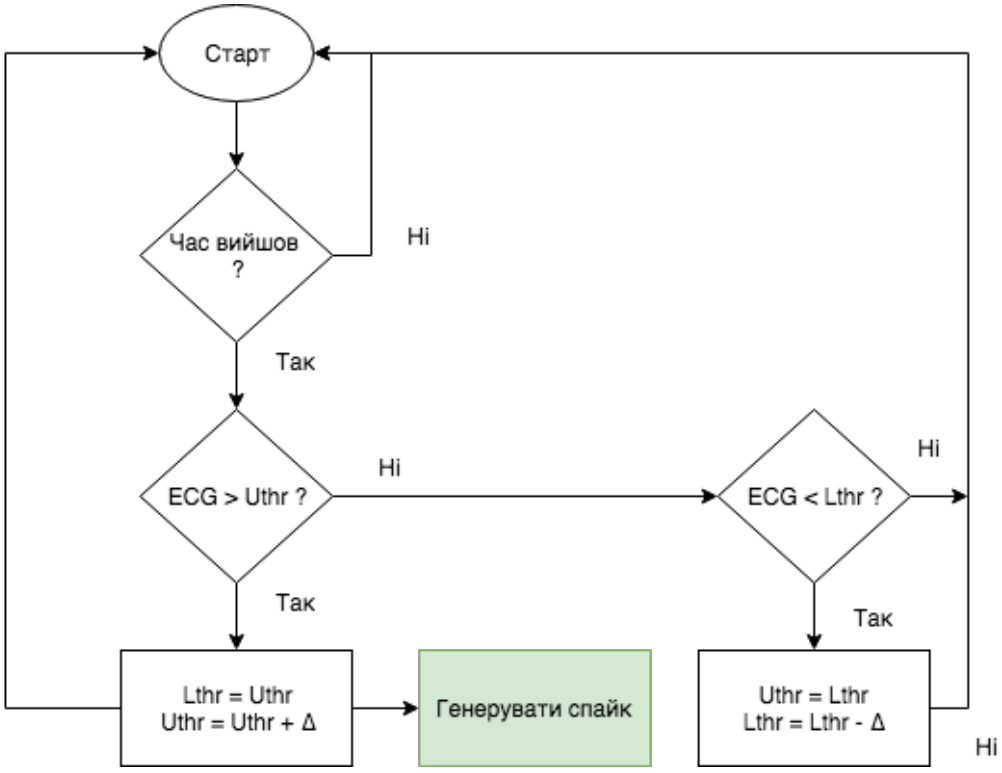


Рисунок 2.4 – Блок-схема для кодування та генерації спайкової послідовності сигналу

Принцип роботи спайкового шифратора може бути представлений такими сценаріями: а) амплітуда сигнала ЕКГ збільшується; б) амплітуда сигнала ЕКГ зменшується; в) амплітуда сигнала ЕКГ стабільна протягом часового вікна Δ. Кодування базується на двох порогах: *Lthr* і *Uthr* (*Uthr* > *Lthr*). Порівняння напруги виконують в межах фіксованого вікна часу, яке керується таймером. На кожному інтервалі компаратор увімкнено для порівняння миттєвої напруги ЕКГ і порогу *Uthr*.

Тестування запропонованого алгоритму проведено в два етапи. На першому в якості вхідного сигналу використано відфільтрований та піднесений до квадрату сигнал ЕКГ, який здатен передати інформацію тільки про QRS-комплекс ЕКГ (рис. 2.5).

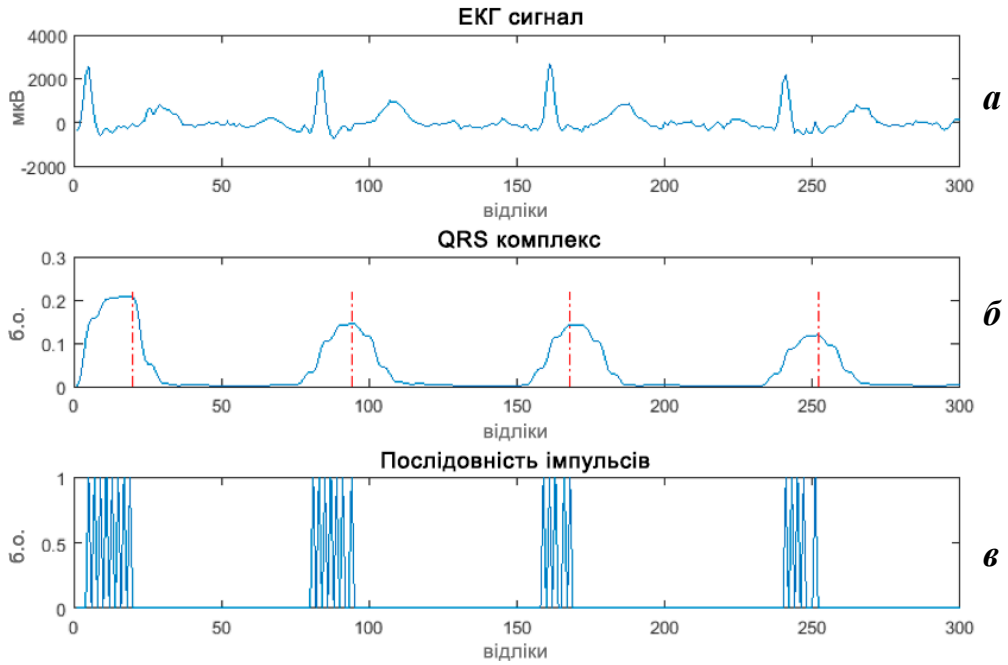


Рисунок 2.5 – Збільшений фрагмент сигналу ЕКГ і відповідні генеровані спайкові послідовності: *а* – вхідний необроблений ЕКГ-сигнал показано на першому графіку; *б* – відфільтрований, квадратичний сигнал після стадії попереднього оброблення з R-піковими моментами; *в* – відповідна послідовність закодованих спайків. Вісь ординат: *а* – амплітуда вхідного сигналу в мікровольтах, *б, в* - амплітуда у безрозмірних одиницях (б. о.). Вісь абсцис: номер відліку сигналу

У другій серії експериментів в якості вхідного сигналу для спайкового кодування всіх фрагментів та хвиль сигналу та їх подальшого аналізу мережею використано відфільтрований ЕКГ-сигнал із повною просторово-часовою морфологією P-QRS-T комплексу (рис. 2.6).

Припустимо, що амплітуда сигналу ЕКГ зростає. Порівняння напруги ініціює таку послідовність кроків:

– оновлення порогів: *Lthr* = *Uthr* і *Uthr* = *Uthr* + Δ;

– генерація одного спайку на виході;

– перезапуск таймера і повернення до режиму очікування.

Якщо результат порівняння є від'ємними, то виконується друге порівняння, при якому напруга ЕКГ порівнюється з порогом *Lthr*. Якщо це порівняння є додатнім і амплітуда сигнала ЕКГ зменшується, то ініціюється така послідовність кроків:

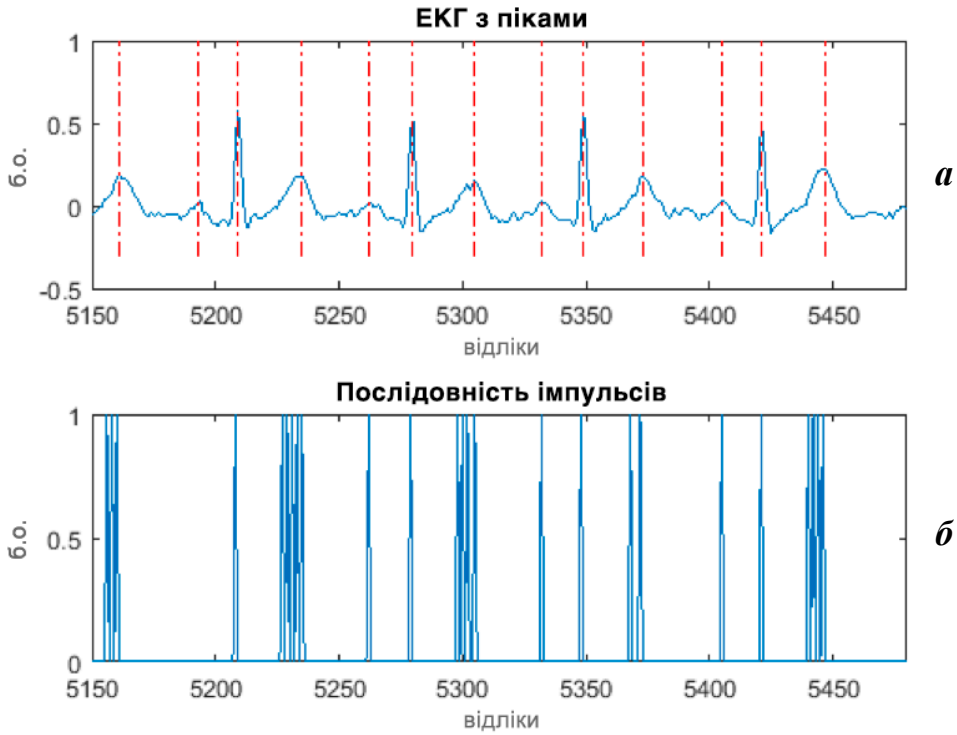


Рисунок 2.6 – Збільшена частина ЕКГ сигналу та відповідна згенерована спайкова послідовність: *а* – необроблений ЕКГ-сигнал, *б* – відповідна закодовану послідовність спайків. Вісь абсцис: номер відліку сигналу. Вісь ординат: *а* – амплітуда вхідного сигналу в мкВ; *б* – амплітуда у безрозмірних одиницях (б. о.).

– оновлення порогів: *Uthr* = *Lthr* і *Lthr* = *Lthr* – Δ;

– перезапуск таймеру і повернення до режиму очікування.

– порогові оновлення виконуються для відстеження зміни амплітуди сигналу ЕКГ у напрямку зростання або спадання;

– при зменшенні амплітуди сигналу ЕКГ не утворюється ніяких спайків, оскільки просторово-часові характеристики P-QRS-T можуть бути захоплені позитивною частиною форми хвилі напруги;

– спайки відсутні за умови стабільності сигналу ЕКГ.

2.1.1 Модель імпульсної штучної нейронної мережі (SNN)

Нейронна структура складається з трьох шарів: вхідний, рекурентний і вихідний (рис. 2.7) . Перший шар є вхідним шаром, який генерує послідовність спайків.

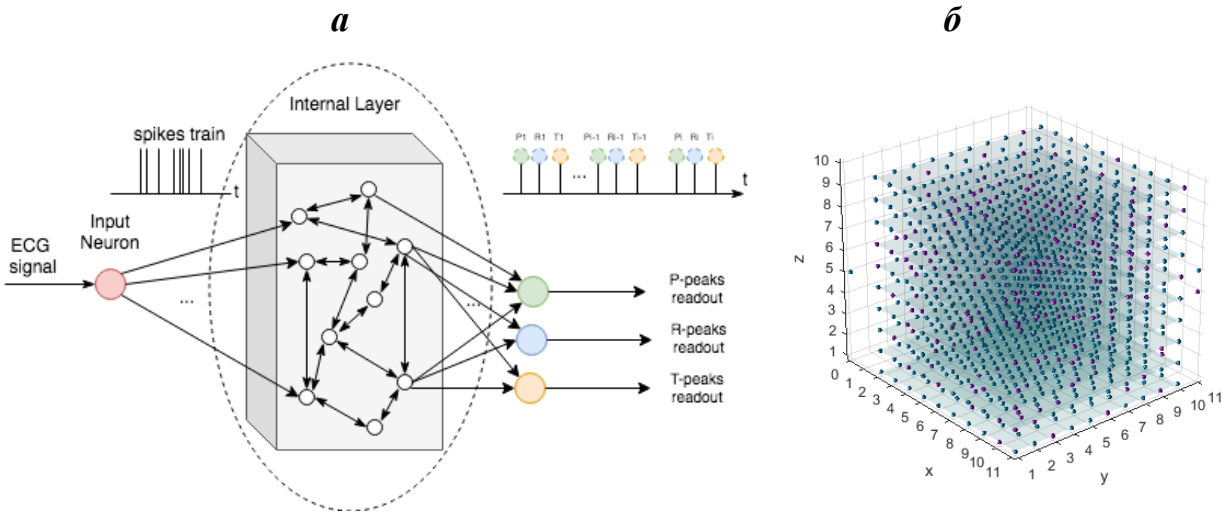


Рисунок 2.7 – Структура спайкової мережі нейронів: *а* – загальний вигляд: вхідний шар, внутрішній шар з рекурентно пов'язаними нейронами та вихідний шар вихідними нейронами; *б* – вигляд у 3D-перспективі

Примітки: ECG signal – вхідний ЕКГ-сигнал, Input Neuron – вхідний спайковий шифратор, spikes train – послідовність спайків, Internal Layer – внутрішній шар нейронів, P-peaks readout – вихід визначення P-піків, R-peaks readout – вихід визначення R-піків, T-peaks readout – вихід визначення T-піків.

Другий шар мережі є рекурентним шаром [16], [27] і складається з $N = N_E + N_I$ рекурентно пов'язаних нейронів, де N_E – число збуджуючих нейронів, а N_I – число гальмівних нейронів. Внутрішній мережевий шар складається з мульти-стабільних нейронів, при цьому мульти-стабільність нейронів зумовлена

метричною асиметрією їх активної дендритної структури, як показано у [24], [25]. Вихідна активна функціональність кожного нейрона описується диференціальними рівняннями:

$$\begin{cases} v' = 0.04v^2 + 5v + 140 - u + I_0, \\ u' = a(bv - u). \end{cases} \quad (2.2)$$

Скидання системи до початкового стану після генерації спайку:

$$\begin{cases} v \leftarrow c, \\ \text{якщо } v \geq 30 \text{ мВ, тоді } u \leftarrow u + d \end{cases} \quad (2.3)$$

де v – мембранний потенціал (в мВ);

u – змінна відновлення мембранного потенціалу;

I_0 – сумарний струм сполучення нейронної структури;

a, b, c, d – безрозмірні параметри.

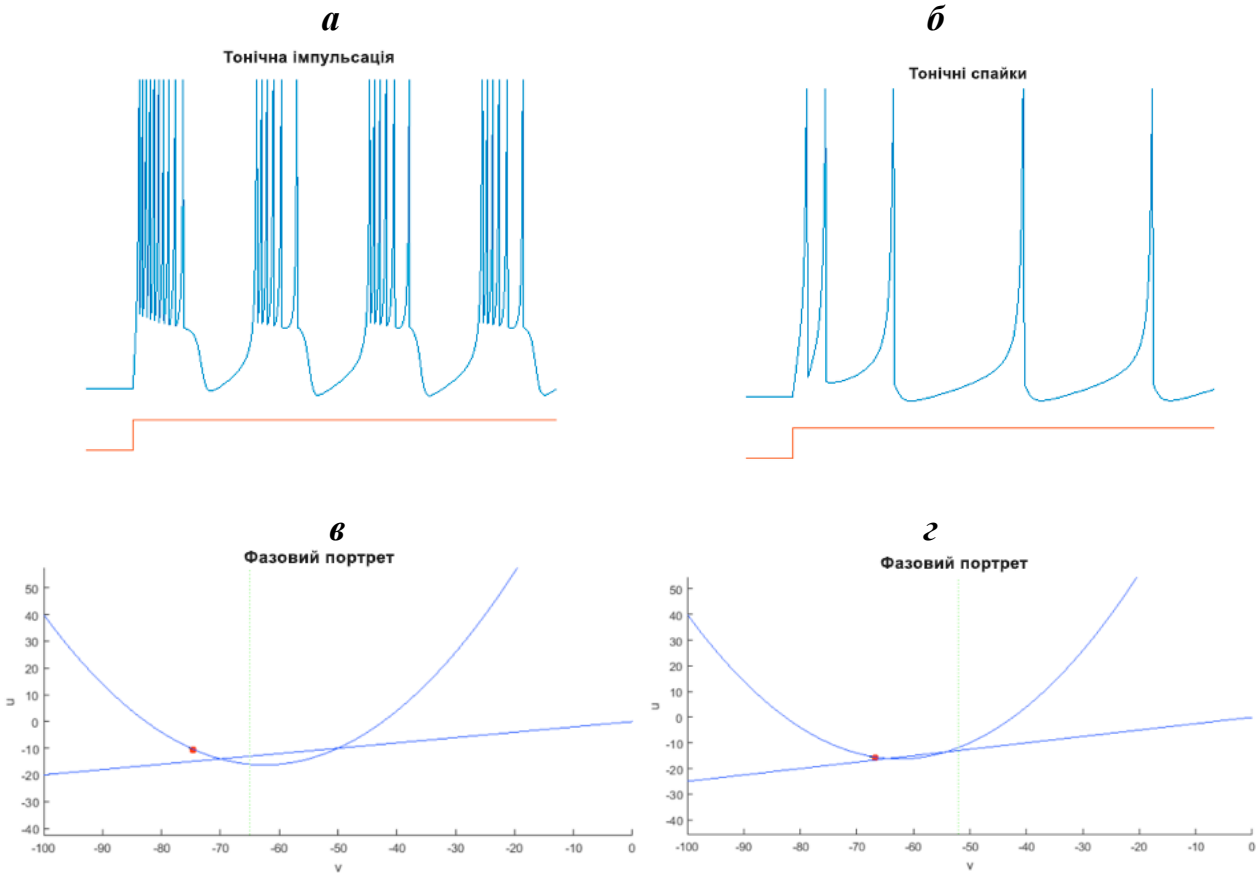


Рисунок 2.8 – Процес тонічної генерації та відповідні фазові портрети мульти-стабільного нейрона: *а* – тонічні одиночні спайки; *б* – тонічний процес генерації пачок спайків; Фазові портрети для мембранного потенціалу v та змінної відновлення u для режимів генерації спайків: *в* – тонічні одиночні спайки; *г* – пачок тонічних розрядів

Кожен нейрон має три стабільні стани соматично-дендритної структури, зумовлені їх дендритною метричною асиметрією. Стабільні стани внутрішніх нейронів визначаються внутрішнім розподілом струму I_0 і структурою нейрона відповідно до розв'язку рівняння (1.9). Типовий процес тонічної генерації спайків і пачок розрядів, що генеруються мережевими нейронами описано системами рівнянь (2.2) і (2.3) та показано на рис. 2.8, *а* і *б*.

Фазові портрети показані для змінних *v* і *u* на рис. 2.8, *в* та *г*. Рис. 2.8, *а* відповідає набору параметрів: $a = 0.02$, $b = 0.2$, $c = -65$, $d = 6$ та двом стаціонарним стабільним станам нейрону. Модель з набором параметрів: $a = 0.03$, $b = 0.25$, $c = -52$, $d = 0$ відповідає випадку три-стабільної нейронної структури, вихідні коливання якої показані на рис. 2.8, *б*.

У табл. 2.1 наведено параметри, що використовуються для моделювання збуджуючих, як регулярних пікових, а також інгібіторних нейронів [27], [28], [49]. Індекс *E* застосовується для позначення збуджуючих нейронів, тобто нейронів, які є глутаматергічними, тоді як індекс *I* – для нейронів інгібіторного типу з ГАМКергічними синапсами. Струм в постсинаптичному нейроні *j* обчислюється за формулою:

$$I_j = \sum_{i=0}^{n_j} i_{iJ} \cdot w_{ij} \qquad (2.4)$$

де n_j – загальна кількість пресинаптичних нейронів;

i_{iJ} – струм через синаптичний зв'язок між i_{iJ} попереднім синаптичним нейроном і j^{th} постсинаптичним нейроном;

w_{ij} – відповідна вага, яка моделюється шляхом зміни провідності.

Нейрональний каркас складається з $N = 1{,}000$ внутрішніх нейронів з часткою $N_I = 0.25 \cdot N_E$. З'єднання між двома окремими нейрональними вузлами встановлюються з імовірністю, розрахованою за правилом:

$$P(D) = C \cdot \exp\left(\frac{-D(A,B)}{\lambda}\right), \qquad (2.5)$$

де $D(A, B)$ – евклідова відстань між двома нейронними вузлами *A* і B;

λ – щільність з'єднань.

Параметр C залежить від типу пресинаптичних і постсинаптичних нейронів, тобто від того, чи є вони збуджуючими (*Ex*) чи гальмівними (*Inh*) клітинами. При моделюванні параметр C встановлюється як: C_{Ex-Ex}=0.3 (для зв'язків між двома збуджуючими нейронами), C_{Ex-Inh}=0.2 (для зв'язків між збуджуючими та гальмівними нейронами), $C_{Inh-Inh}$=0.1 (для сполуки між двома гальмівними нейронами), C_{Inh-Ex}=0.4 (для зв'язків між гальмівними і збуджуючими нейронами. Початкова синаптична вага визначається W_0 = 0.0045 мкСм/см^2, а її зміна обмежена діапазоном від 0 до 10·W_0. Синаптичні затримки між двома нейронами встановлюються випадково у інтервалі між 1 мс і 2 мс.

Таблиця 2.1

Параметри нейромережі для моделювання нейронів

Клас параметрів	Параметр	Значення
Нейрон	a_E, a_I b_E, b_I c_E, c_I d_E, d_I	0.02,0.1 0.2, 0.2 –65, –65 8, 2
Початкові синаптичні сили	W_0	0.1 мкСм/см^2
Збуджуюча / гальмівна спайкова час-залежна синаптична пластичність (E-STDP / I-STDP)	A_{+E}, A_{+I} τ_{+E}, τ_{+I} A_{-E}, A_{-I} τ_{-E}, τ_{-I}	0.1, –0.1 20, 20 мс –0.1, 0.1 20, 20 мс
Гомеостаз (збуджуючий)	α, T, R_{target}	0.1, 10, 35
Гомеостаз (гальмівний)	α, T, R_{target}	0.1, 2, 3.5

Вихідний шар мережі представлено мульти-стабільними нейронами з активними дендритними структурами, які збирають всі синаптичні входи від

нейронів другого внутрішнього шару. Вони здатні генерувати одиночні спайки чи знаходитись в стабільно низькому або перманентному стані насичення у відповідь на велику кількість одночасних синаптичних входів. Вихідний шар представлений три-стабільними нейронами (Nout_1, Nout_2, Nout_3, Nout_4) з активними дендритними структурами, які акумулюють синаптичні входи від нейронів другого внутрішнього шару і здатні генерувати одиничні спайки у відповідь на велику кількість одночасних синаптичних входів. Кожен з відповідних вихідних нейронів тренується для роботи наступним чином: нейрон Nout_1 працює як тригер на детектування комплексу P-QRS-T і генерує високорівневий стабільний стан (upstate) при надходженні P-QRS-T комплексу. Після інтервалу в 320 мс, який відповідає нормальній максимальній довжині послідовності P-QRS-T, Nout_1 повертається до стабільного стану низького рівня (downstate). Nout_2 генерує спайк тільки тоді, коли Nout_1 на високому рівні та виявлено P-пік комплексу ЕКГ. Якщо P-QRS-T комплексу не виявлено Nout_2 не генерує спайк. Нейрон Nout_3 генерує спайк тільки тоді, коли Nout_1 на високому рівні і виявлено R-пік комплексу ЕКГ. Нарешті, нейрон Nout_4 генерує спайк тільки тоді, коли Nout_1 на високому рівні і виявлено T-зубець комплексу ЕКГ.

Оновлення синаптичних ваг вихідних нейронів відключається після інтервалу часу T_i. Часовий інтервал від 0 (початок запису ЕКГ) до T_i є фазою тренування спайкової нейронної мережі. Тривалість тренувальної фази T_i фіксована і має два значення: 10 та 30 с. Вплив на результат збільшення фази тренування буде показано у наступному розділі 4. Під час цієї фази вагові коефіцієнти оновлюються за допомогою спайк-часової пластичності (STDP) [17], [19], [20]. STDP використовує кореляцію між спайками (на основі моменту часу кожного спайку) для виведення потенційного причинного зв'язку між пре- і постсинаптичними нейронами. Який потім використовується для впливу на зміни ваги [65]. Синаптичні ваги, що з'єднують збуджуючий нейрон з будь-якими іншими нейронами, оновлюються за допомогою STDP (E-STDP) збуджуючого типу, тоді як синаптичні ваги, що з'єднують гальмівний нейрон із

збуджуючими нейронами, оновлюються за допомогою STDP інгібіторного типу (I-STDP). Водночас відбувається синхронізація спайк-залежної пластичності STDP (E-STDP / I-STDP) з використанням експоненційної функції:

$$W = \begin{cases} A_+ \exp\left(\dfrac{-t}{\tau_+}\right) & t > 0 \\ A_- \exp\left(\dfrac{-t}{\tau_-}\right) & \textit{інакше} \end{cases} \tag{2.6}$$

Для запобігання насичення всіх нейронів використовують гомеостатичне синаптичне масштабування, за якого рівняння комбінованої зміни ваги задається як

$$\frac{dw_{ij}}{dt} = \left[\alpha w_{ij} \left(1 - \frac{R_{avg}}{R_{target}} \right) + \beta(w_{after_{ij}}) \right] K , \tag{2.7}$$

де α – гомеостатичний коефіцієнт масштабування;

β – коефіцієнт масштабування STDP;

R_{avg} – середня швидкість стрільби нейрона спайками протягом великого періоду часу;

R_{target} – попередня визначена швидкість “стрільб” нейрона (вибір дизайну);

K – коефіцієнт масштабування, який визначається рівнянням

$$K = \frac{R_{avg}}{T(1 + \left| 1 - \dfrac{R_{avg}}{R_{target}} \right| \gamma)}, \tag{2.8}$$

де T – інтервал часу, протягом якого усереднюється швидкість генерації спайків.

2.1.2 Моделювання імпульсних режимів генерації мультистабільних нейронів

Для випадку три-стабільного рішення отримані шаблони вихідних спайків активності нейронів, що описані рівняннями (2.2) та (2.3) (рис. 2.9, *а, б, в, г, д*).

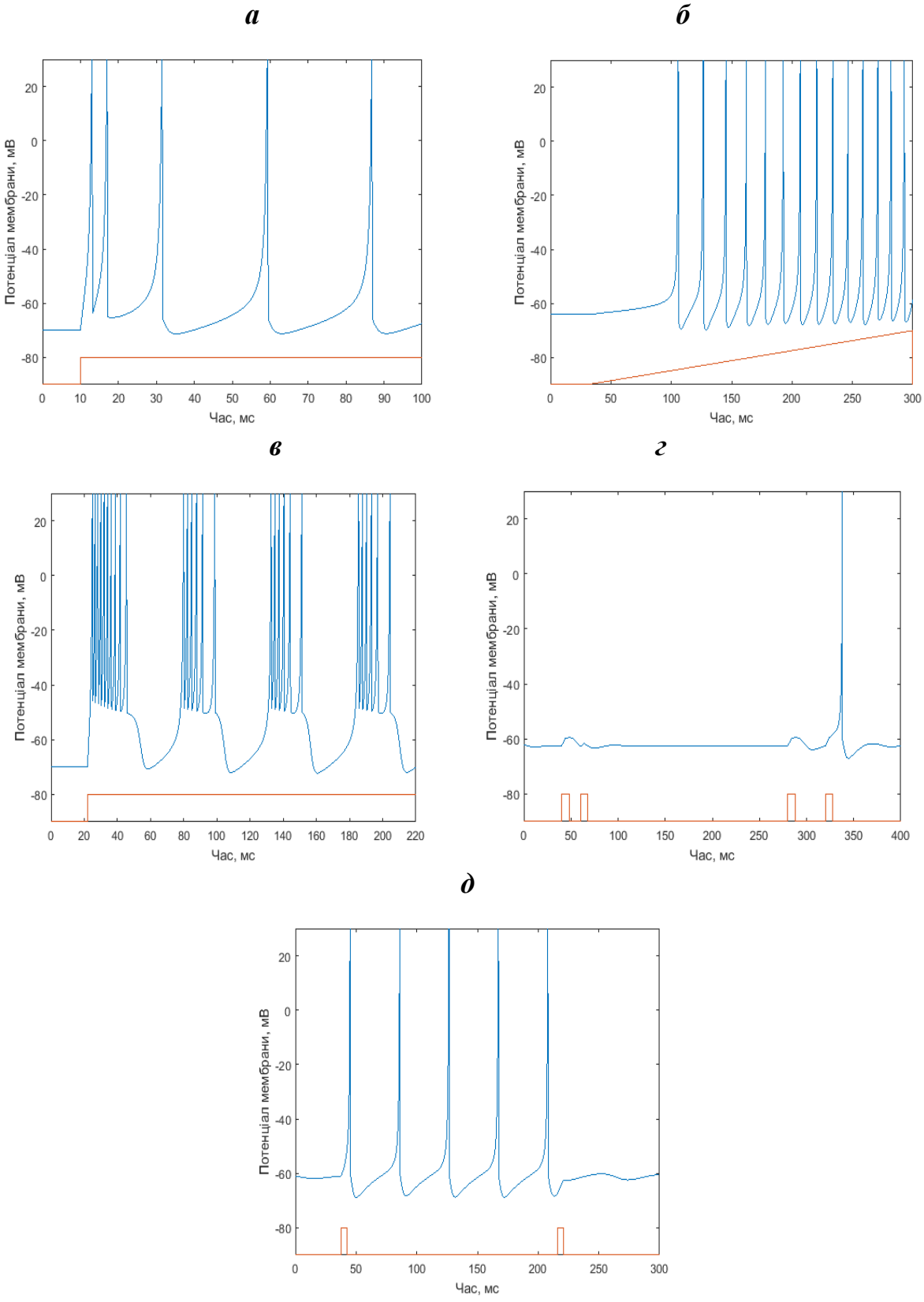


Рисунок 2.9 – Морфологія електричних відповідей модельних мультистабільних нейронів: *а* – регулярні спайки; *б* – адаптація частоти спайків; *в* – розряди; *г* – осциляції; *д* – бі-стабільні перемикання

Регулярні спайки. При активації вхідних синапсів нейронів генерується кілька швидких спайків з коротким періодом, після чого процес стабілізується, а період збільшується (рис. 2.9, *а*). Цей процес називається адаптацією частоти спайків. Збільшення сили постійного струму інжекції збільшує частоту міжсмугового перемикання. У моделі це відповідає параметрам $c = –65$ мВ і $d = 8$ (великий стрибок *и* після спайку) та стаціонарному стану слабкого насичення соматичної частини downstate або upstate при інтенсивностях вхідної синаптичної активації від $G_{smax} = 0.003$ до 0.0045 мкСм/см^2 та $G_{smax} = 0.0059$ до 0.009 мкСм/см^2 відповідно. Це типові регулярні піки біологічних нейронів кори із постійною частотою спайків.

Нейрони в цьому режимі можуть генерувати періодичні спайки з надзвичайно високою частотою практично без будь-якої адаптації, як це можна бачити на рис. 2.9, *б*. У моделі це відповідає значенню $a = 0.1$ (швидке відновлення).

Пачки розрядів. Під час процесу активації нейрони генерують стереотипні сплески (рис. 2.9, *в*), тобто пачки імпульсів. Частота пачки може досягати 40 Гц. У моделі це відповідає $c = 50$ мВ (дуже висока напруга перезаряду), $d = 2$ (помірний стрибок *и* після спайку) та середньому стабільному стану соми mid-state при інтенсивностях вхідної синаптичної активації від $G_{smax} = 0.0045$ до 0.0055 мкСм/см^2.

Осциляції. Під час осциляцій нейрони мають затухаючий або стійкий підпороговий рівень коливань мембранного потенціалу (рис. 2.9, *г*). Вони резонують із постійною власною частотою. Така поведінка відповідає $a = 0.1$ і $b = 0.26$ та середньому стабільному стану mid-state. В цьому режимі існує бі-стабільність станів спокою і повторювання спайків.

Нейрон може перемикатися між станами на відповідний час після стимуляції. Також варіант бі-стабільного режиму можливий із генерацією звичайних регулярних спайків (рис. 2.9, *д*), коли інтенсивність вхідної синаптичної активації переходить граничне значення 0.0059 мкСм/см^2, що відповідає переходу нейронів у стан над-активації upstate.

Проведено дослідження з різним числом збуджуючих і гальмівних нейронів, мережа побудована з 800 збуджуючих і 200 гальмівних нейронів.

Поведінку нейронної мережі досліджено у просторі та часі (рис. 2.10) У серії експериментів зображено часові діаграми спайків нейронів мережі впродовж часу моделювання 13 с. Можна бачити, що більшість нейронів мережі генерують регулярні розряди спайків практично у синхронному режимі після збудження мережі лише первинним стимулом (рис. 2.10, *а*). У другому досліді частина нейронів $n = 180$, що була обрана випадково, переведена у середній стан стабільності за допомогою імпульсу інжектованого зовнішнього струму. Можна бачити, що в цьому випадку (рис. 2.10, *б*) мережа мала певну асинхронну динаміку, при цьому середня швидкість генерації спайків була близькою до 8 Гц. Темні вертикальні лінії вказують на випадкові епізоди синхронізованих генерацій спайків у діапазоні частот від 10 до 40 Гц.

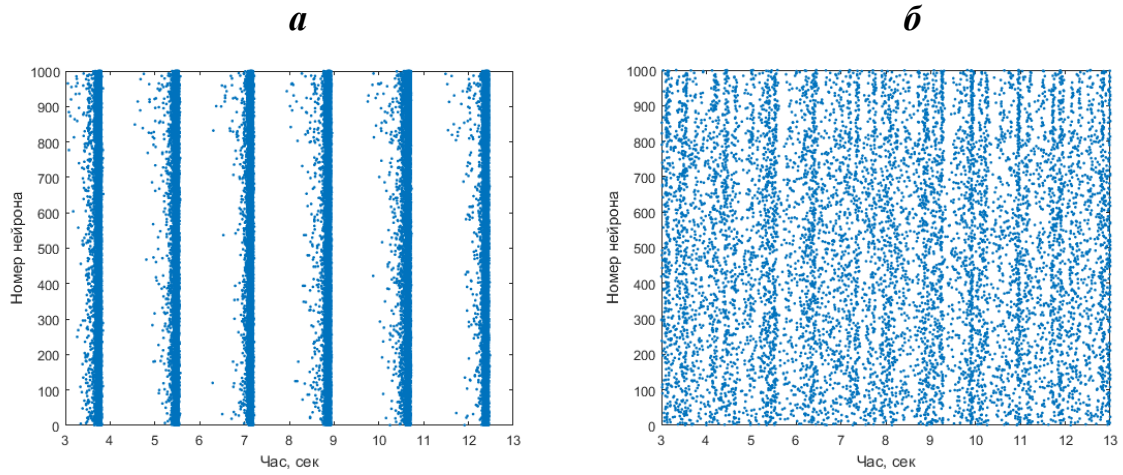


Рисунок 2.10 – Моделювання мережі з N=1,000 поєднаних нейронів: *а* – без зовнішнього стимулу, *б* – з додатковою активацією 180 нейронів. Вісь ординат: номер нейрона. Вісь абсцис: час, с

Важливо, що хоча нейрони у мережі пов’язані між собою випадково та відсутня синаптична пластичність, нейрони самоорганізуються в збірки і проявляють колективну ритмічну поведінку в певному частотному діапазоні, що дуже схоже на нейрони неокортексу головного мозку в активному стані. Зміна відносної сили синаптичних з’єднань може привести до інших типів колективної поведінки, включаючи хвилі збудження нейронів під час сну. При цьому

симуляція такої системи відбувається в реальному часі та без застосування будь яких чисельних методів, що дозволить у майбутньому розраховувати подібні нейронні мережі набагато більшого порядку без втрат ефективності алгоритму.

2.2 Розроблення структури і змісту моделі нейрону та нейронної мережі на VHDL

Для розроблення моделі нейрона використано мульти-стабільну модель спайкового нейрона, яка описана вище за допомогою рівнянь (2.2) та (2.3). Цифрова електроніка пропонує багато інструментів та платформ, на яких можна виконувати моделювання та перевірку апаратних засобів: Hardware Design Languages (HDL), VHDL, Verilog, а також багато інших. По-перше, проекти в цьому класі мов можуть бути синтезовані на апаратному забезпеченні і працювати в режимі реального часу. По-друге, існують змішані розширення цих мов (наприклад, VHDL-AMS, Verilog-A), які дозволять додати у модель аналогові компоненти. Це може дозволити вбудувати реальні біо-фізіологічні взаємодії в існуючу модель.

На першому етапі здобувачем було створено власну бібліотеку “Neuron” мовою VHDL, що реалізує модель імпульсного штучного нейрона із властивістю мультистабільності для подальшого моделювання мереж будь-якої складності.

2.2.1 Розроблення структури імпульсних нейронів та мережі SNN для FPGA-архітектури

Цифрові системи, що впроваджені для емуляції нейронних мереж, все ще далекі від рівня ефективності нейронних обчислень або нейронного кодування у порівнянні з реальними біологічними нейронними мережами [8], [9], [13]. Тому існує необхідність побудови систем, які за адекватною продуктивністю та обчислювальною потужністю відповідають ефективності їх біологічних аналогів [44], [45], [54]. Але виникає певна проблема при спробі встановити зв'язок між

нейронами емульованої спайкової нейронної мережі [17]. Насправді, нейронні системи повинні з'єднувати мільйони нейронів і таким чином реалізовувати ефективність інтегральних електронних реалізації. Тому дуже важливим питанням є застосування системи AER (Address-Event Representation), яка частково має вирішити цю проблему і буде реалізована в нейронній мережі даного дисертаційного дослідження. Одними з перших, хто реалізував цю систему були Mahowald і Sivilotti [64], які запропонували систему розподілу подій у системі, для того щоб мати можливість передавати імпульси шару нейронів на процесорі до відповідного місця в масиві нейронів другого процесора.

Система AER, яка реалізована в нейронній мережі, має у своєму складі шифратор, або енкодер, який зчитує спайки всіх нейронів мережі SNN у кожний момент часу і здатний перевести їх у відповідні нейронні адреси, щоб у результаті визначити адресу нейрона, що згенерував спайк (пресинаптичний нейрон), та передати цей спайк потрібному адресату (постсинаптичному нейрону). Водночас, ця структура вирішує проблему оброблення двох або більше подій одночасно, яка існує через неможливість передавати більше однієї адреси через шину зв'язку AER, додаючи при цьому до енкодера умову пріоритету доступу.

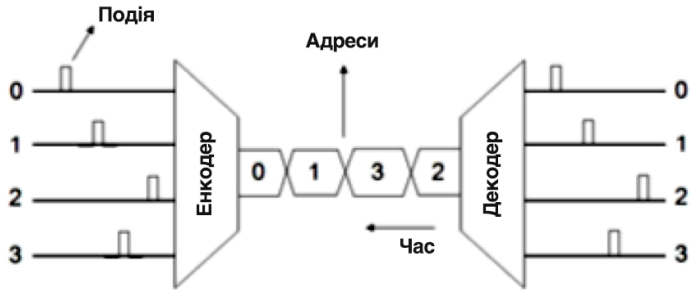


Рисунок 2.11 – Схематичне зображення системи AER

У схемі, що наведено на рис. 2.11, енкодер призначає унікальну адресу для кожного нейрона, який генерує спайк. Потім ці адреси по шині надходять у декодер, який вибирає відповідне розташування спайка. Така система зв'язку є досить ефективною, для того щоб уникнути вузьких місць, які виникають при

обміні інформацією у взаємопов’язаній системі, в нашому випадку – це передача спайків в нейронній мережі. Досягнення ефективної реалізації системи AER зумовило необхідність розглянути випадок, коли дві або більше події відбуваються одночасно, і система повинна вирішити, в якому порядку вони повинні передаватися через шину, оскільки можлива передача тільки однієї адреси за одиницю часу. Цю проблему було вирішено додаванням до модуля AER буферу FIFO (First In First Out), регістру пріоритету енкодера (Priority Encoder). Було реалізовано наступну логіку роботи: всі синаптичні події, що надійшли до нейрона спочатку заповнюють FIFO буфер, який відіграє роль елемента пам’яті, на наступних кроках тактування кожна з подій буфера обробляється окремо. Якщо таких подій було де-кілька, то вони обробляються пост-синаптичним нейроном у тому порядку, якому вони надійшли до входу. Загальний вигляд архітектури модуля AER, розробленого здобувачем наведено на рис. 2.12.

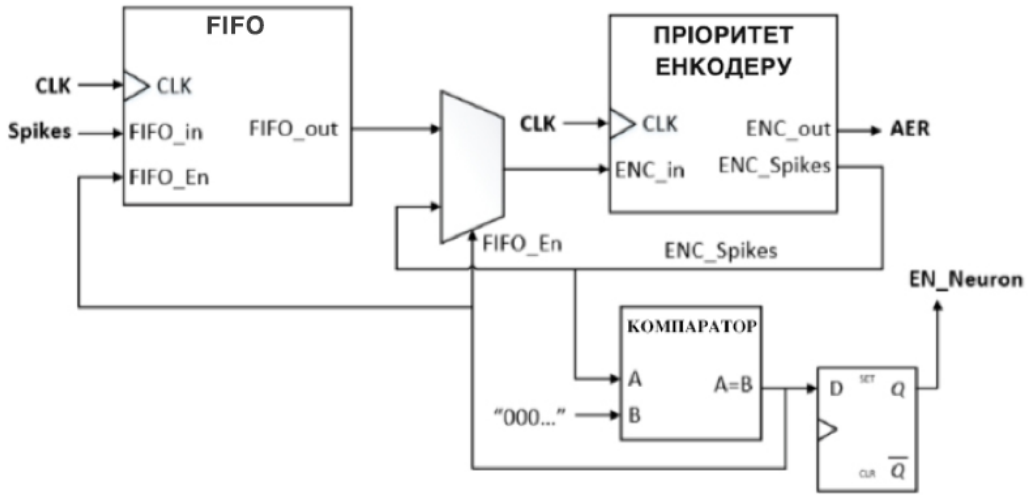


Рисунок 2.12 – Реалізація модуля AER із буферною структурою та регістром пріоритету енкодера для усунення конфліктів передачі спайків у мережі

2.2.2 Побудова оптимізованої моделі імпульсного штучного нейрона (ІШН)

Одним з найскладніших завдань при реалізації апаратної моделі нейрона із біологічною подібністю є відтворення процесу генерації спайкового механізму,

оскільки це потрібно робити із досить великою швидкістю (до 500 Гц) для великої кількості обчислювальних одиниць (в поточній реалізації це 1,000 нейронів). Генеративний процес утворення спайків імпульсного нейрона показано за допомогою рівнянь (2.2) та (2.3). Реалізація цієї задачі можлива при достатньо фундаментальній математичній оптимізації цих рівнянь для конкретної платформи.

Спочатку рівняння для цифрової системи були представлені знаковим вектором довжиною 13 біт, отже, з одним бітом для знаку числа, а параметри, що відповідають змінним a, b, c і d, встановлені в їх рекомендовані значення, щоб отримати звичайну регулярну модель спайкового нейрона. Потім було вирішено працювати з двійковим поданням цілих чисел, для чого у двійковому векторному представлені множенням на десять. Таким чином, для мембранного потенціалу v і напруги відновлення u мембранного потенціалу отримано таку модель, яка записується формулами:

$$v' = \frac{1}{250}v^2 + 5v + 1400 - u + I \tag{2.9}$$

$$u' = \frac{1}{50}(\frac{1}{5}v - u) \tag{2.10}$$

$$\text{Якщо } v \geq 300, mo \begin{cases} v \leftarrow -650 \\ u \leftarrow u + 80 \end{cases} \tag{2.11}$$

З іншого боку, існує спосіб збереження ресурсів у реалізації на FPGA – це операції множення або ділення за допомогою бітової операції зсуву. Тому, якщо необхідно обчислити результат множення на число степені двійки у двійковому числі, потрібно лише зсунути вліво бінарний вектор на кількість позицій, відповідну до степені двійки. Навпаки, якщо зсунути бінарний вектор вправо на певну кількість позиції, отримаємо результат ділення вектора на число, яке є ступенем двійки.

Отже, коефіцієнти рівнянь були скориговані для множення та ділення за степені двійки для оптимізації рівнянь. По-перше, для рівняння (2.9) число 250 замінено на 256, оскільки $256 = 2^8$. По-друге, для рівняння (2.10) дільники 50 і 5

були замінені на 64 і 4 відповідно, оскільки $64 = 2^6$ і $4 = 2^2$. $5v$ і v в рівнянні (2.9) замінені на реалізацію суми шести членів мембранного потенціалу, яка може бути виражена як сума $2v$ і $4v$. Ці два числа подаються як степінь двійки, множення реалізовані у вигляді операцій зсуву вліво. Це дозволило отримати такі рівняння для адаптованої моделі:

$$v[n+1] = 2v[n] + \frac{2}{2^8}[n] + 2^2 v[n] + 1400 - u[n] + I[n], \tag{2.12}$$

$$u[n+1] = u[n] + \frac{1}{2^6}(\frac{1}{2^2}v[n] - u[n]),$$

$$\text{Якщо } v[n+1] \geq 300, mo \begin{cases} v[n+1] \leftarrow -650 \\ u[n+1] \leftarrow u[n+1] + 80 \end{cases} \tag{2.13}$$

2.2.3 Особливості архітектури та дизайну ІШН

Як вже відзначалося, в дисертаційній роботі запропонована модульна, гнучка і масштабована цифрова система для аналізу ситуацій, які можуть виникнути при симуляції спайкової нейронної мережі, а саме цифровий блок нейрона, який має сім входів, один вихід та оперативну пам'ять (RAM), в якій зберігаються ваги синаптичних зв'язків з іншими нейронами (рис. 2.13).

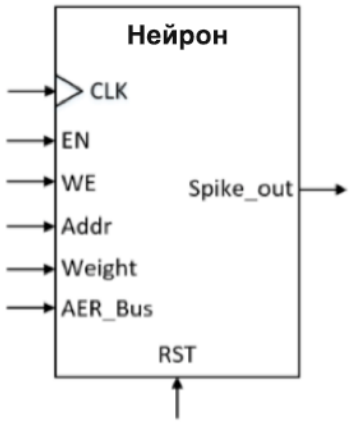


Рисунок 2.13 – Цифровий блок синтезованого нейрона із зазначеними сигналами введення/виведення

Водночас, тактовий сигнал (CLK) відповідає за координацію різних дій нейрона; активаційний сигнал (EN) служить для активації його функціонування; сигнали увімкнення запису (WE), адреси (Addr) і синаптичної ваги (Weight) використовують для запису у внутрішню пам’ять RAM нейрона; вхід шини AER (AER_Bus) використовують для зчитування адреси нейрона, що генерує спайк; двійковий вихідний сигнал (Spike_out), показує, чи згенерував нейрон спайк чи ні (рис. 2.14).

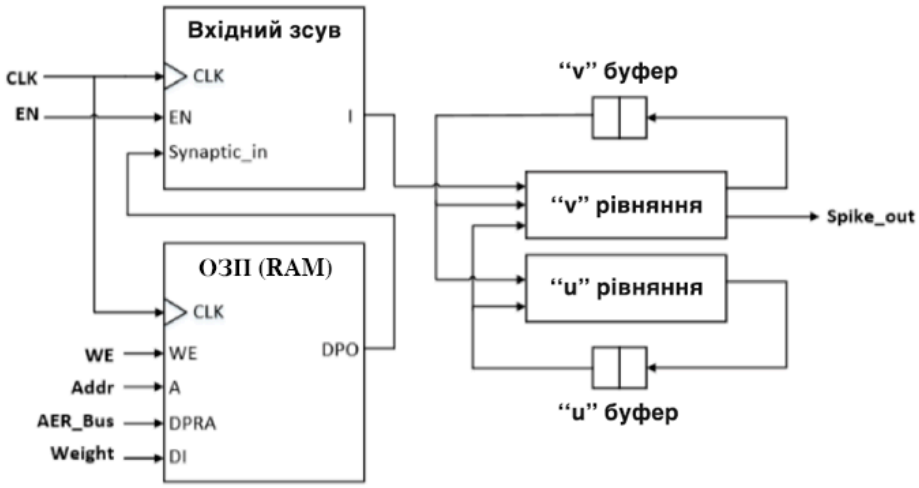


Рисунок 2.14 – Загальна структурна схема синтезованого нейрона

Також існують різні послідовні і комбінаційні цифрові блоки, які оперують діями нейрона: внутрішня RAM пам’ять; послідовний блок для значень вхідних синаптичних ваг; два регістри для мембранного потенціалу v і змінної відновлення напруги u; два комбінаційних блока для реалізації диференціальних рівнянь (2.12).

Внутрішня оперативна пам'ять нейрона (RAM) представлена матрицею, яка складається з номерів нейронів і колонки відповідних синаптичних ваг із цими нейронами. Як показано в табл. 2.2, кожна позиція в стовпці відповідає вазі синаптичного зв'язку з нейроном, тому перший стовпець відповідає синаптичний вазі зв'язку з нульовим нейроном нейронної мережі, другий – другому і т. д. Таким чином, отримаємо матрицю зв’язків для кожного нейрона мережі. В табл. 2.2 показано приклад формування таблиці синаптичних взаємозв’язків для п’яти нейронів. Якщо зв’язок між нейрона відсутній, то у відповідному стовбці

записується нульове значення синаптичної ваги. Додатнє значення синаптичної ваги вказує на збуджуючий синаптичний зв’язок, від’ємне – гальмівний.

Таблиця 2.2

Таблиця значень блока синаптичних ваг нейрона (RAM)

Номер нейрона	Синаптична вага				
	Номер нейрона				
	0	1	2	3	4
0	0	70	0	0	0
1	0	0	– 40	0	0
2	0	0	0	120	0
3	0	0	0	0	– 15
4	0	0	0	0	0

Процес функціонування нейрона виглядає таким чином: спочатку він зчитує шину зв'язку AER для нейрона, який “вистрілив” спайком; потім оперативна пам'ять, що відповідає за зчитування синаптичної ваги, пов'язаної із записаним на шині AER номером, передає її послідовному блоку (“Вхідний зсув”). Послідовний блок відправляє синаптичну вагу до комбінаційного блоку диференціального рівняння мембрани, водночас, відповідаючи за додавання синаптичних ваг у випадку, коли кілька нейронів “обмінюються” спайками одночасно. Для цього шина AER зупиняє активність нейрона, дезактивуючи сигнал EN, що приводить до неможливості передачі більше одного спайка одночасно і запису однієї з адрес нейронів, які передаються в кожному тактовому циклі. В той самий час блок “Вхідний зсув” обмежує від'ємне значення синаптичної ваги до значення, що відповідає мембранному потенціалу v = – 140 мВ, оскільки при такому значенні мембранного потенціалу нейрон, з

біологічної точки зору, не зможе генерувати будь-які спайки через явище глибокої гіперполяризації.

Схему цифрової реалізації оптимізованих рівнянь можна побачити на рис. 2.15. Вершина діаграми відповідає рівнянню (2.2), де сигнал v_1 генерується квадратом сигналу *v_n* і бінарним зсувом вправо на вісім позицій. Сигнал v_2 генерується сумою двох значень: бітового зміщення *v_n* вліво на одну та дві позиції відповідно. Відповідно до рівняння (2.2) виконується сума інших факторів, в т. ч. і синаптичної ваги. Нарешті, використовуючи компаратор, блок цифрових рівнянь визначає, чи перевищує значення напруги мембрани порогове значення, вказане в рівнянні (2.3). Якщо це так, сигнал спайку активується, вказуючи на "стрільбу" нейрона. Крім того, якщо потенціал на виході нейрона перевищує порогове значення, мембранний потенціал скидається до значення *c*, інакше він продовжує працювати з сигналом *v3* до наступного тактового циклу, в якому регістр *v* замінить сигнал *v_n* на виході *v_n1*.

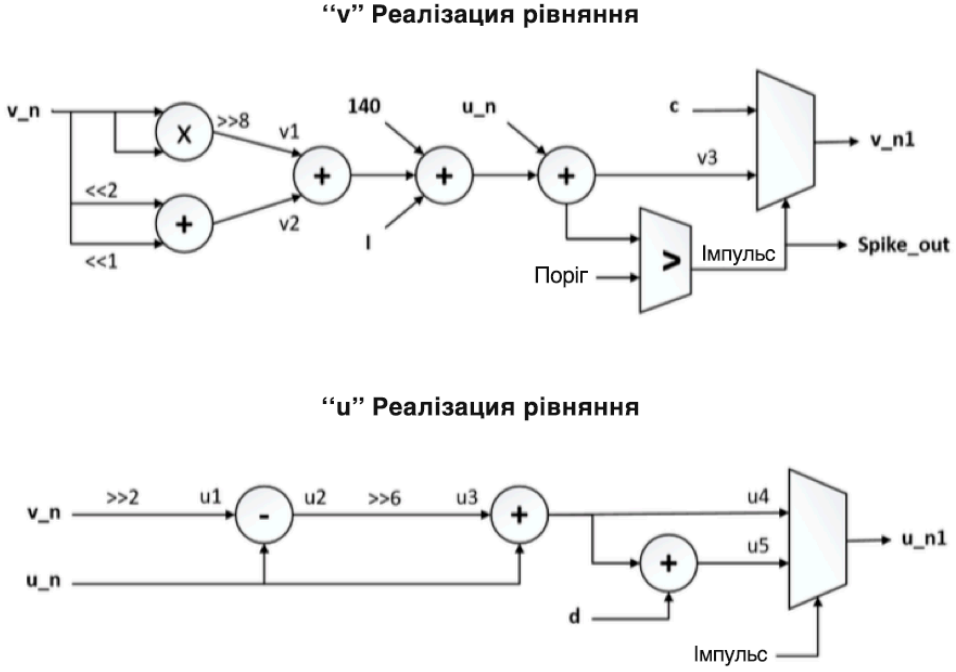


Рисунок 2.15 – Схема цифрової реалізації оптимізованих нейронних рівнянь мультистабільного нейрона

Нижня частина діаграми на рис. 2.15 відповідає рівнянню (2.13), де сигнал *u1* генерується шляхом виконання двох бінарних зсувів правої мембранної

напруги *v_n*; виходячи з цього, сигнал *u3* отримують за допомогою віднімання від *u1* сигналу *u_n* і двійкового зсуву вправо на шість розрядів. Сигнали *u4* і *u5* подаються на мультиплексор з вибором каналу за умови, що нейрон перебуває в неврівноваженому стані (зовнішній сигнал Spike) та встановлює значення відновлення потенціалу мембрани *u_n1*, за рівнянням (2.13).

Проведено 50 симуляції кожної моделі: із оптимізацією та без неї. Порівняння використання ресурсів неоптимізованої і оптимізованої моделі нейронів, що відповідає рівнянням (2.2–2.3) і (2.12–2.13) відповідно, наведено у табл. 2.3. Видно, що для реалізації одного нейрона в FPGA Xilinx Spartan 3E неадаптована модель використовує приблизно в 6 разів більше структурних вентилів FPGA, ніж оптимізована модель. Ця різниця значно збільшиться при побудові нейронної мережі з десяти, ста або більшої кількості нейронів. Таким чином, у першому наближенні можна сказати, що для реалізації нейронної мережі з 1,000 нейронів із оптимізованою моделлю необхідна FPGA-структура із кількістю логічних вентилів не менше ніж 100,000 одиниць. Тоді, як на оптимізованій моделі, можливо було б реалізувати тільки 80 нейронів за допомогою FPGA-структуру того ж розміру.

Таблиця 2.3

Використання ресурсів одного модельованого нейрону для оптимізованої та неоптимізованої цифрової моделі нейрона

Ресурс	Використання		Доступно	Використання, %	
	Оптим.	Неоптим.		Оптим.	Неоптим.
LUT	134	786	63,400	0.21	1.24
LUTRAM	16	16	19,000	0.08	0.08
FF	130	169	126,800	0.10	0.13
DSP	1	1	240	0.42	0.42
IO	28	28	210	13.33	13.33
BUFG	1	1	32	3.13	3.13

2.2.4 Симуляція поведінки ІШН

Результати різних імітацій реалізованої моделі для різних типів поведінок, які демонструє нейрон за умови різних зовнішніх умов для оптимізованої моделі представлено на рис. 2.16–2.18.

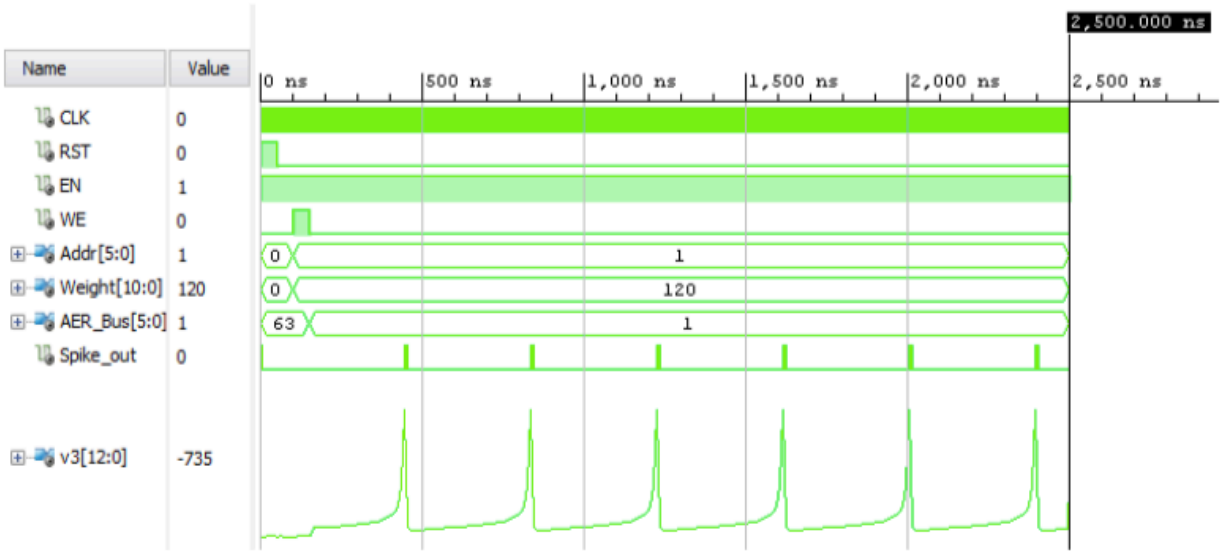


Рисунок 2.16 – Часові діаграми входів, виходів і мембранного потенціалу нейрона на вхідний прямокутний імпульс амплітудою 12 мВ

На рис. 2.16 показана відповідь нейрона на вхідний імпульс амплітудою 12 мВ, тривалістю 50 нс. Синаптична вага, що відповідає потенціалу 12 мВ, записується у внутрішню пам'ять нейрону “1”, активуючи сигнал WE. На кілька тактових циклів пізніше адреса нейрона “1” записується в шину AER, імітуючи постійну “стрільбу” цього нейрона, отже, генеруючи тим самим вхідний імпульс 12 мВ для імітованого нейрона. Це дає змогу побачити, як він починає генерувати нейрональні імпульси на вихідному сигналі Spike_out.

Нейрон демонструє поведінку, подібну, відповіді на імпульс 12 мВ у відповідь на вхідний імпульс із більшою амплітудою 30 мВ і генерує при цьому більш високу частоту вихідних спайків (рис. 2.17).

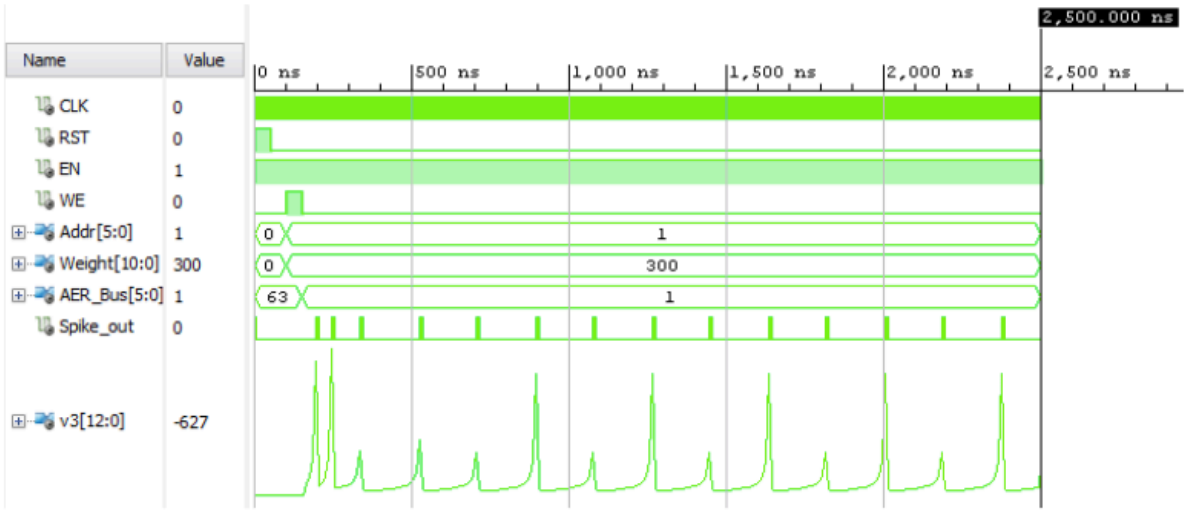


Рисунок 2.17 – Часові діаграми входів, виходів і мембранного потенціалу нейрона на вхідний імпульс амплітудою 30 мВ

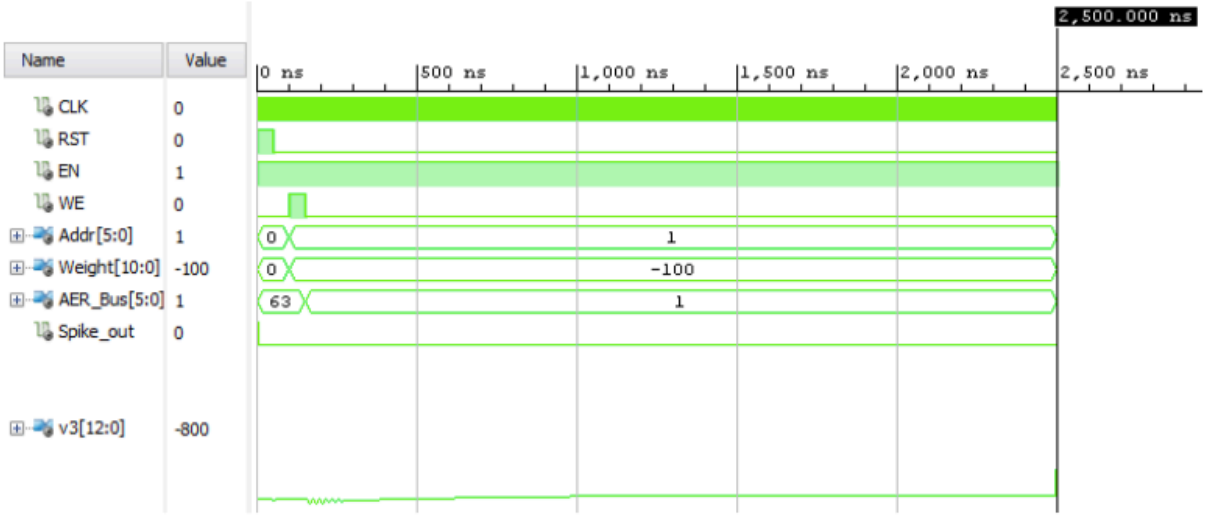


Рисунок 2.18 – Часові діаграми входів, виходів і мембранного потенціалу нейрона на вхідний імпульс амплітудою –10 мВ

Відповідь нейрона на вхідний імпульс з амплітудою –10 мВ (рис. 2.18) представляє собою відсутність будь-якої імпульсної генерації та низький рівень стабілізації мембранного потенціалу нейрона downstate, що відповідає поведінці оригінальної математичної моделі. Це свідчить про те, що за різних умов була доведена адекватність поведінки оптимізованої апаратної реалізації модельного нейрона.

2.2.5 Емуляція роботи нейронної мережі нейромодуля

В нейронній мережі, що складається з декількох шарів (рис. 2.8), вхідний шар приймає зовнішні стимули; далі йде змінна кількість прихованих шарів, які відповідають за виконання операцій нейронної мережі, а потім вихідний шар, в якому мережа транслює свої стимули або відповіді назовні. Наведена вище архітектура дозволяє реалізувати стільки нейронів, скільки потрібно, що можна продемонструвати на процесі підключення двох нейронів.

Можна побачити, що система AER відповідає за встановлення зв'язку між усіма нейронами, зчитуючи спайки, для переведення їх у відповідні адреси і передачі їх на шину AER. Всі нейрони підключені до сигналу EN_Neuron, що дозволяє за необхідності зупинити активність нейронів, оскільки, шина AER може передавати тільки одну адресу за такт. Більш того, нейрони вхідного шару є фактично зовнішніми стимулами, тому введення інформації в нейронну мережу здійснюється шляхом введення в систему вектора спайків, які певним чином кодують вхідну інформацію.

Як відзначалося раніше, кожен нейрон формується за допомогою цифрового модуля та навчального модуля STDP, взаємозв'язок між якими встановлюється за допомогою сигналів вмикання запису (WE), адреси (Addr) і сигналів синаптичної ваги, які дозволяють записувати дані в оперативну пам'ять нейрона. Загальна структурна схема нейро-модуля із властивістю синаптичної пластичності STDP показана на рис. 2.19.

З рис. 2.19 видно, що існує кілька взаємопов'язаних комбінаційних і послідовних блоків для функціонування модуля STDP: адреса лічильника (“Ліч. адр.”) для вибору синаптичного з'єднання, для якого застосовано правило STDP; селектор посилання на приріст або декремент (“I/D Виб.”), який активує відповідний сигнал; синаптичний лічильник ваги (ліч. ваги), який відповідає за збереження і модифікацію синаптичної ваги всіх з'єднань нейрона; набір комбінаційних блоків, що дозволяють реалізувати цифрову логіку рівнянь (2.6) і (2.7).

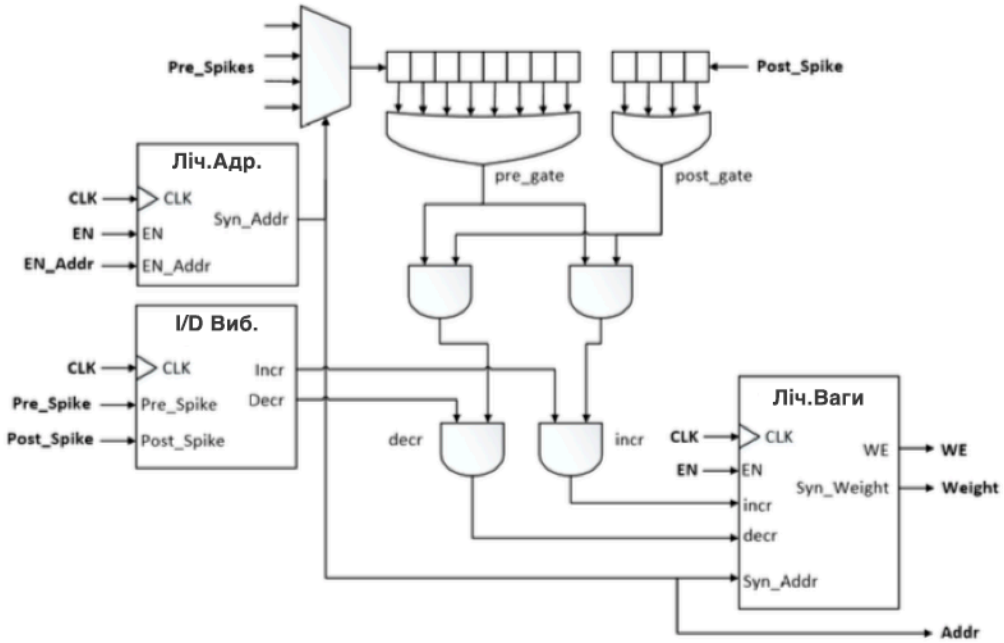


Рисунок 2.19 – Структурна схема нейро-модуля із властивістю синаптичної пластичності STDP

Операційний процес навчального модуля здійснюється у певній послідовності. Спочатку зчитуються сигнали попередніх спайків і пост-спайків, лічильник адреси відповідає за вибір з'єднання для застосування правила STDP з сигналом Syn_Addr, який виконує функції селектору каналу для мультиплексора і вказує на відповідну адресу оперативної пам'яті нейрона.

Після того як, до нейрона з підключеним модулем STDP надходить наступний спайк, блок “I/D Виб.” активує відповідний вихідний сигнал, щоб вказати, чи потрібно лічильнику ваги збільшити або зменшити значення ваги з'єднання для цього синапсу. Крім того, спайк поширюється через регістр зсуву і активує сигнал pre_gate. Потім, якщо нейрон навчального модуля спрацьовує, його спайк поширюється відповідним регістром зсуву, що активує сигнал post_gate. Активація цих двох логічних затворів разом генерує імпульс збільшення (інкремент) для лічильника синаптичної ваги. Тому значення ваги синаптичного з'єднання збільшиться відповідно до тривалості цього імпульсного лічильника (фактично відстань між двома сусідніми спайками). Крім того, він

активує сигнал увімкнення запису (WE), щоб дозволити оновлення оперативної пам'яті нейрона.

В цілому, в модулі STDP сигнали EN і EN_Addr разом з сигналом Pre_Spikes повинні розглядатися за допомогою конкретних цифрових блоків. Ці блоки не включені в діаграму на рис. 2.19, оскільки вони залежать або від архітектури шарів, або від функцій мережі для симуляції. Таким чином, неможливо забезпечити повністю універсальне рішення для будь-якої нейронної мережі, таке рішення буде залежати від архітектури.

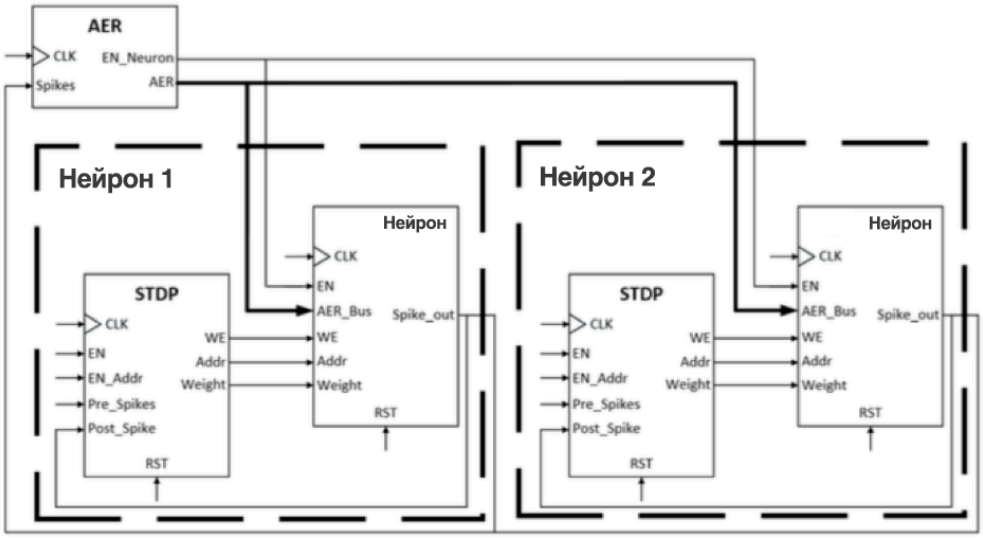


Рисунок 2.20 – Схема взаємозв'язку різних блоків для емуляції SNN двох або більше нейронів

2.3 Впровадження властивості мультистабільності імпульсних штучних нейронів для апаратної реалізації

Для задачі синтезу вихідних мульти-стабільних нейронів, моделі яких описані у розділі 1, необхідно спочатку вирішити декілька фундаментальних питань та оптимізації які можна буде покласти в основу VHDL-нейрона, реалізація якого описана у розділі 2.2 та наступною апаратною реалізацією.

Отже, класичні вагові коефіцієнти для вхідних синапсів було замінено на структурно-залежну вагову матрицю, що визначається геометричними

параметрами математичної моделі – довжиною l_i та діаметром d_i, де $i = 1..n$ при загальній кількості n дендритних розгалужень:

$$(G_{smax}, l_i, d_i), \;\; i = 1..n \tag{2.14}$$

Розроблено простий нейрон з трьома дендритами ($n = 3$) – входами, кожен з яких характеризується двома параметрами: довжиною l_i та діаметром d_i. Інтенсивність кожного синаптичного входу характеризується величиною максимальної провідності синаптичних каналів дендрита G_{smaxi}. Сома виконує роль інтеграції вхідних струмів і визначає поточний інтегральний потенціал соми E_x та відповідний стабільний стаціонарний стан мультистабільного нейрона в цілому. Для визначення такого стану необхідні значення порогових потенціалів, які є розв'язками рівняння (1.10) $E_0(1)$, $E_0(2)$, $E_0(3)$, $E_0(4)$ та $E_0(5)$ для випадку три-стабільності. Рівням стабільних станів серед них відповідають розв'язки $E_0(1)$, $E_0(3)$ та $E_0(5)$, діапазон атракції до кожного стану визначається локальними максимумами функції $I_0(E_0)$. Для визначення цих рівнів необхідно розв'язати наступні рівняння:

$$\begin{aligned} &\frac{dI(\mathrm{E_0})}{d(\mathrm{E_0})} = 0\,, \qquad \text{якщо } \mathrm{E_0}(3) > \mathrm{E_0} > \mathrm{E_0}(1) \\ &\frac{dI(\mathrm{E_0})}{d(\mathrm{E_0})} = 0\,, \qquad \text{якщо } \mathrm{E_0}(5) > \mathrm{E_0} > \mathrm{E_0}(3) \end{aligned} \tag{2.15}$$

Нехай розв'язками рівнянь (2.15) будуть значення потенціалу $E_0(13)$, $E_0(35)$, тоді вихідну інтегральну функцію порогового елементу $E_{out}(E_0)$ можна записати у вигляді:

$$E_{out}(E_x) = \begin{cases} 0 & \text{якщо } E_0(1) \le E_0 \le E_0(13), \\ MID & \text{якщо } E_0(13) < E_0 < E_0(35), \\ 1 & \text{якщо } E_0 \le E_0(35), \end{cases} \tag{2.16}$$

де стани *"0"* та *"1"* відповідають стійкому низькому та високому рівню активації на виході, при цьому генерації вихідних спайків не відбувається, осцилятор вимкнено. Ця ситуація моделює вхідну активацію нейрона або слабку, що не приводить до генерації, або настільки потужну, що нейрон повністю переходить у збуджений стан. Проміжній стан *"MID"* характеризується

наявністю вихідної генерації, причому частота прямо пропорційна сумарній інтенсивності вхідних синапсів нейрона. Осцилятор із заданою тактовою частотою при цьому підключено до генератора спайків.

Синаптичну активацію описують функцією $G_s(E) = G_{smax} \cdot F(E)$, де E – це мембранний потенціал, G_{smax} максимум провідності, яка визначається інтенсивністю активації та $F(E)$ – активаційна змінна, що моделюється кінетикою синаптичних каналів [15]. Апроксимаційні рівняння для кусково-лінійного випадку мають такий вигляд:

$$F(E) = \begin{cases} 0 & \text{якщо } E \le E_{a1}, \\ \frac{E_{a1} - E}{E_{a1} - E_{a2}} & \text{якщо } E_{a1} < E \le E_{a1}, \\ 1 & \text{якщо } E \le E_{a1}. \end{cases} \qquad (2.17)$$

де E_{a1} – потенціал порогу для відкритого стану вхідних синаптичних каналів;

E_{a2} – потенціал, при якому всі *NMDA*-канали звільнюються від Mg^{2+} блоку. Значення $E_{a1} = -56$ та $E_{a2} = 10$ мВ обрані саме такими, якими вони є для біологічних NMDA-чутливих каналів [35]. Функціональна залежність (2.17) показана на рис. 2.21.

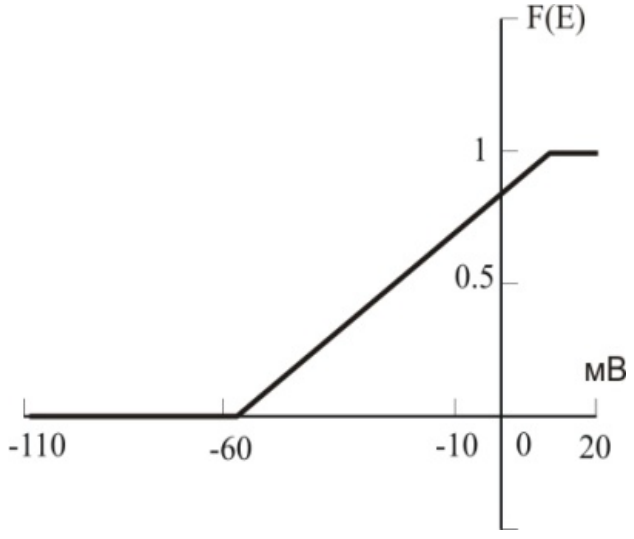


Рисунок 2.21 – Кусково-лінійна апроксимація кінетичної функції каналів, що описує потенціал-залежну характеристику синаптичної активації дендритних входів штучного нейрона

Схематично вигляд дискретного синтезованого нейрона представлений еквівалентною структурною схемою (рис. 2.22). Нейрон в реалізованій моделі розділено на чотири окремі компоненти: вхідні блоки (дендрити), пороговий блок (сома), генератор вихідних сигналів - спайків і осцилятор.

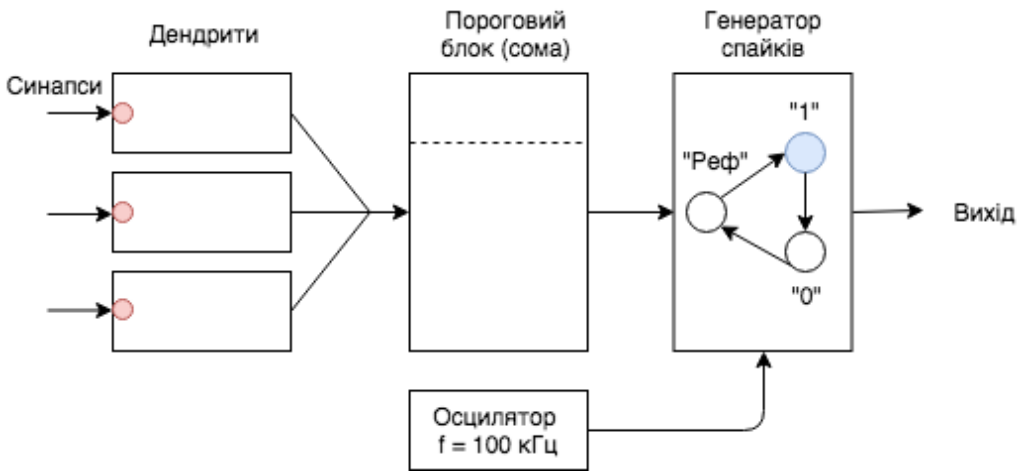


Рисунок 2.22 – Еквівалентна структурна схема моделі дискретного нейрона вихідного шару мережі

Зв'язок між блоками забезпечують односпрямовані канали передачі повідомлень (на рис. 2.22 позначені суцільними лініями зі стрілками, які вказують напрям передачі інформації в системі). Деякі канали повідомлень починаються і закінчуються в одному блоці, деякі інші – в різних.

Вхідні елементи, або дендрити, мають кусково-лінійну активацію, відповідно до рівняння (2.17). Пороговий елемент, або сома, реалізує сумацію всіх вхідних сигналів, при цьому сумарний рівень струму, який обчислюється за рівнянням (2.16), встановлює поточний стан нейрона відповідно до рівняння (2.15). Генератор спайків реалізує кінцеву машину станів, яка послідовно перемикається між активним станом *“1”*, неактивним станом *“0”* та станом рефрактерності *“Реф”*. Перемикання між станами виконуються автоматично у режимі генерації із затримкою, що становить не менше 0.5 мс, та формується за допомогою таймеру. Для тактування блоку генерації спайків підключено блок осцилятора із фіксованою частотою генерації тактового сигналу, що задається як

параметр. Для кожного зв'язку між блоками встановлено затримку передавання сигналу 10 нс.

2.4 Моделювання роботи нейроморфного модуля на базі FPGA

Для перевірки роботи імпульсного штучного мультистабільного нейрона на вхідні порти програмного модуля подавалися різні послідовності векторів IN_i синаптичних активацій: із однакової для всіх входів та постійною у часі синаптичною інтенсивністю $G_{smax} = 0.005$ мкСм/см^2 у першій серії досліджень та із варіацією рівнів інтенсивності на входах у другій.

Активація із постійною інтенсивністю. При однаковій інтенсивності синаптичного збудження на різних входах асиметричних дендритів внесок результуючого струму буде різним, тому результуючий потенціал соми матиме різний зсув. Ефективність синаптичного входу досить добре характеризується структурною матрицею (2.14), результати моделювання якої наведено на рис. 2.23. Тут три нейронні входи надійшли до нейрона з трьох дендритних гілок. Якщо входи IN=000, то вихід завжди буде на низькому логічному рівні, й активація лише одного дендрита IN=100 або IN=001, згідно з рівнянням стану системи, не змінить стабільного стану нейрона, отже він буде перебувати у стані низької активації, або downstate. Якщо активувати всі входи IN=111, то нейрон перейде у стан над-активації, або upstate (рис. 2.23, *в*). Активація тільки двох зважених асиметричних дендритних входів IN=110 переведе нейрон у середній стабільний стан mid-state (рис. 2.25, *г*), при цьому на виході спостерігаються ритмічні спайки з постійною частотою.

Активація зі змінною інтенсивністю. У другій серії експериментів проводилася активація всіх трьох входів, тобто вхідний вектор був постійний у часі IN=111, але інтенсивність активації змінювалась для всіх входів. Встановлено, що при однакових значеннях інтенсивності активації для всіх входів $G_{smax1} = G_{smax2} = G_{smax3} \leq 0.0043$ мкСм/см^2 нейрон перебував у стані downstate; при збільшенні інтенсивності від 0.0043 до 0.0065 мкСм/см^2 нейрон

знаходився в середньому стабільному mid-state стані; при подальшому збільшенні інтенсивності активації за значення 0.0065 мкСм/см^2 нейрон перебуває тільки у стані над-активації, або up-state.

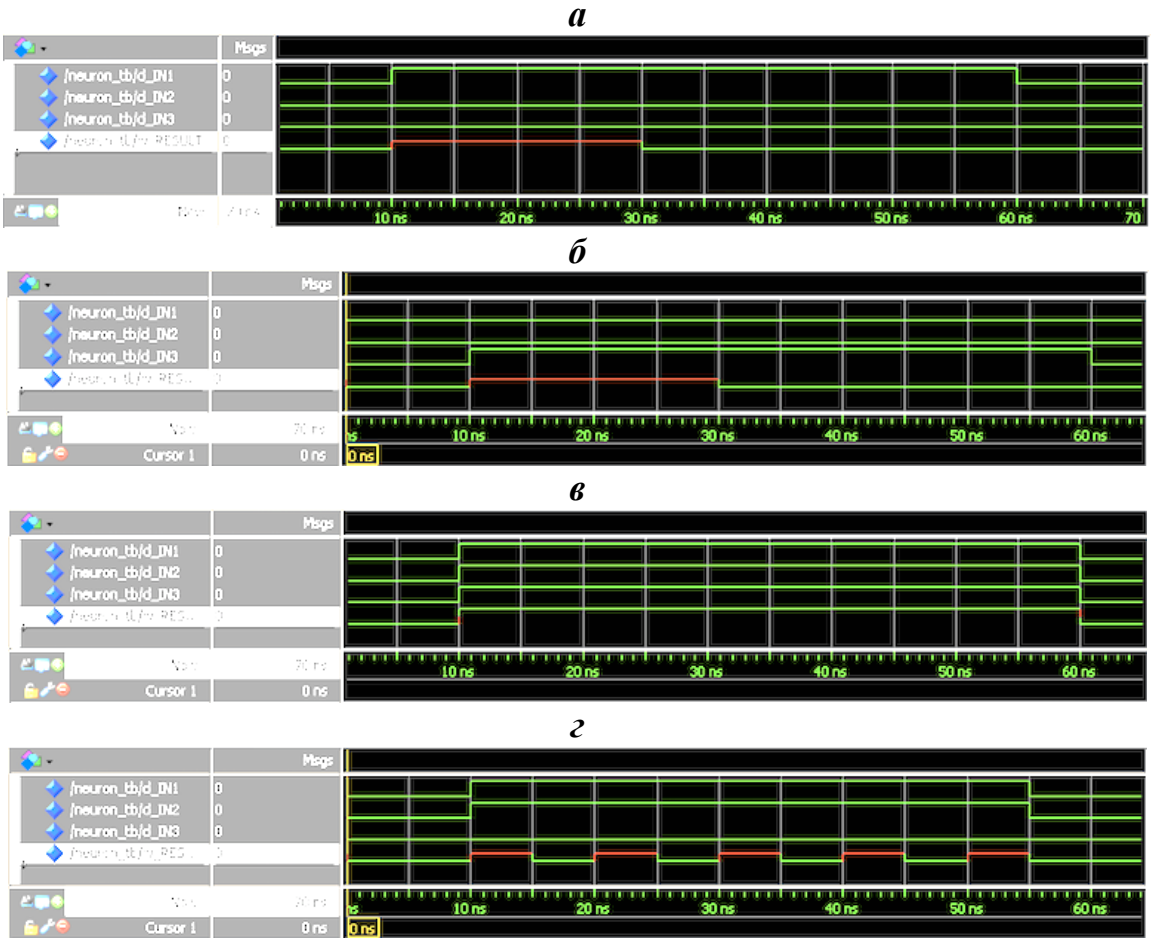
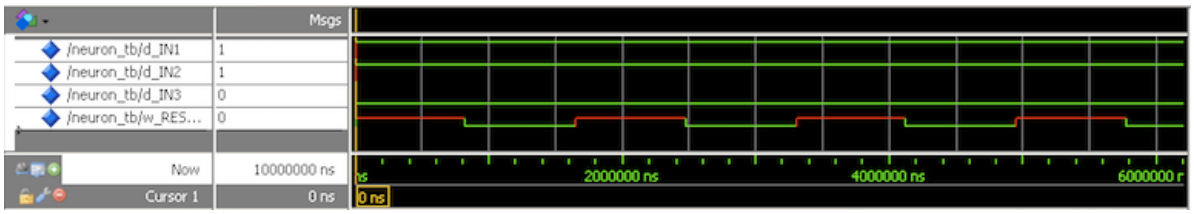


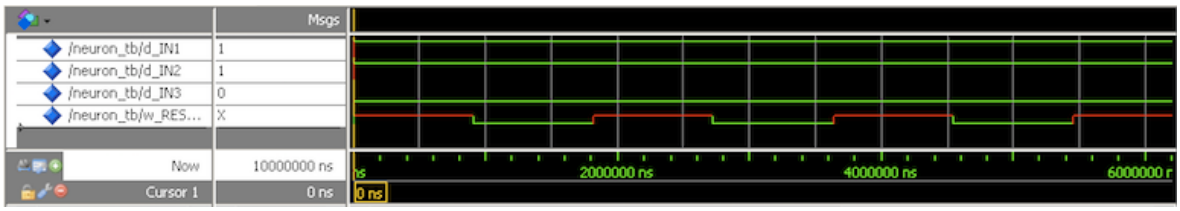
Рисунок 2.23 – Часові діаграми генерації вхідних та вихідного сигналів синтезованої моделі: *а, б* – downstate, *в* – upstate, *г* – mid-state

На рис. 2.24 показана генерація вихідних імпульсів для трьох значень інтенсивності активації G_{smax} (в мкСм/см^2): 0.0045 (рис. 2.24, *а*), 0.0055 (рис. 2.24, *б*) та 0.0061 (рис. 2.24, *в*). При підвищенні інтенсивності вхідної активації частота генерації вихідних імпульсів зростає прямопропорційно. Фактична вимірювана частота вихідних спайків для випадків (рис. 2.24, *а, б, в*) становила відповідно 900, 1100 та 1220 Гц.

а



б



в

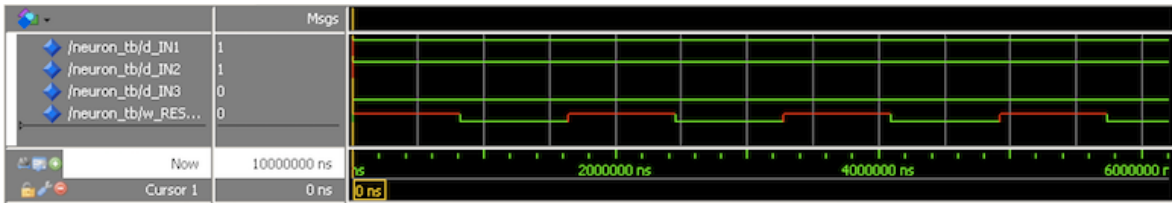


Рисунок 2.24 – Часові діаграми генерації вихідних спайків нейрона для випадку стану mid-state при змінній інтенсивності активації входів G_{smax} (в мкСим/см^2): *а* – 0.0045, *б* – 0.0055, *в* – 0.0061

Залежність частоти генерації від рівня синаптичної активації показано на рис 2.25.

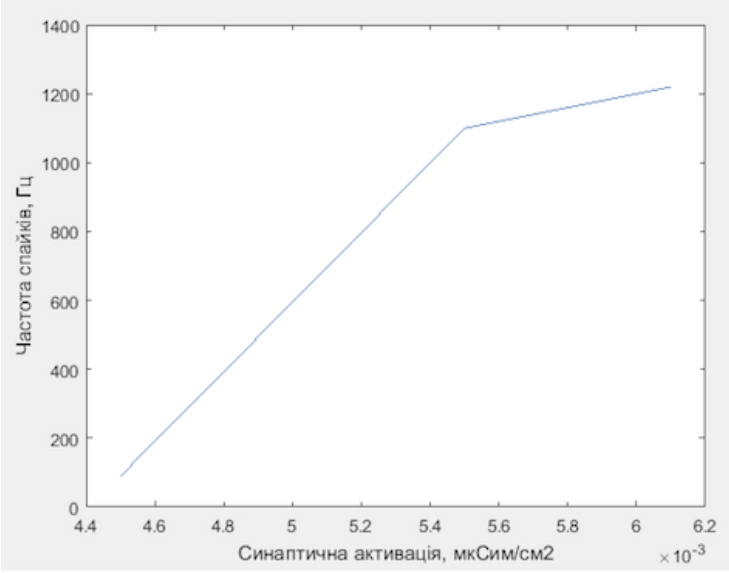


Рисунок 2.25 – Графік залежності вихідної частоти генерації спайків від вхідної інтенсивності синаптичної активації нейрона. Вісь ординат – частота спайків, Гц. Вісь абсцис – синаптична активація, мкСм/см^2

Набір вихідних паттернів спайків нейрона можна ускладнювати за допомогою синтезу мультистабільної моделі із блоком вихідної генерації із більшою кількістю режимів та отримувати, наприклад, пачки вихідних спайків або бі-стабільні режими перемикання, які показано на рис. 2.26, *а, б*.

а ***б***

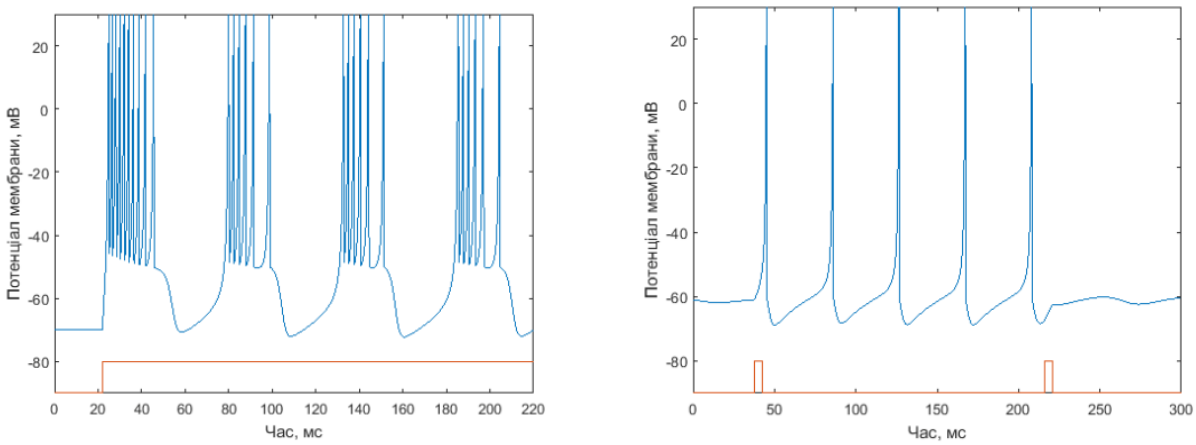
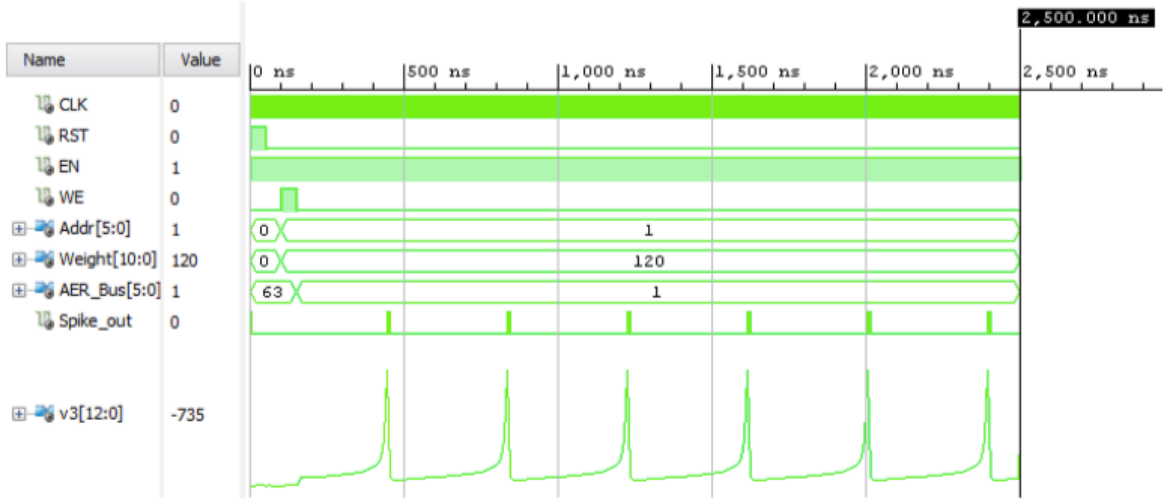


Рисунок 2.26 – Електричні вихідні режими генерації математичного мультистабільного нейрона: *а* – розряди спайків, *б* – бі-стабільний. Вісь ординат – потенціал мембрани, мВ. Вісь абсцис – час, мс

Також досліджена імпульсна поведінка нейронів мережі нейро-модуля. Загальний вигляд імпульсації для відповідних налаштувань показано на рис. 2.27, *а*, *б*. Слід відзначити високий рівень співпадіння процесів, що було описано в модельному дослідженні за допомогою програмного забезпечення, та отриманих реальних даних з нейро-модуля для адаптованої синтезованої моделі нейронної мережі. Це свідчить про можливість застосування такого підходу для вирішення певних прикладних задач (розпізнавання паттернів сигналів у конкретному випадку) та подальшої валідації результатів обчислень.

а



б

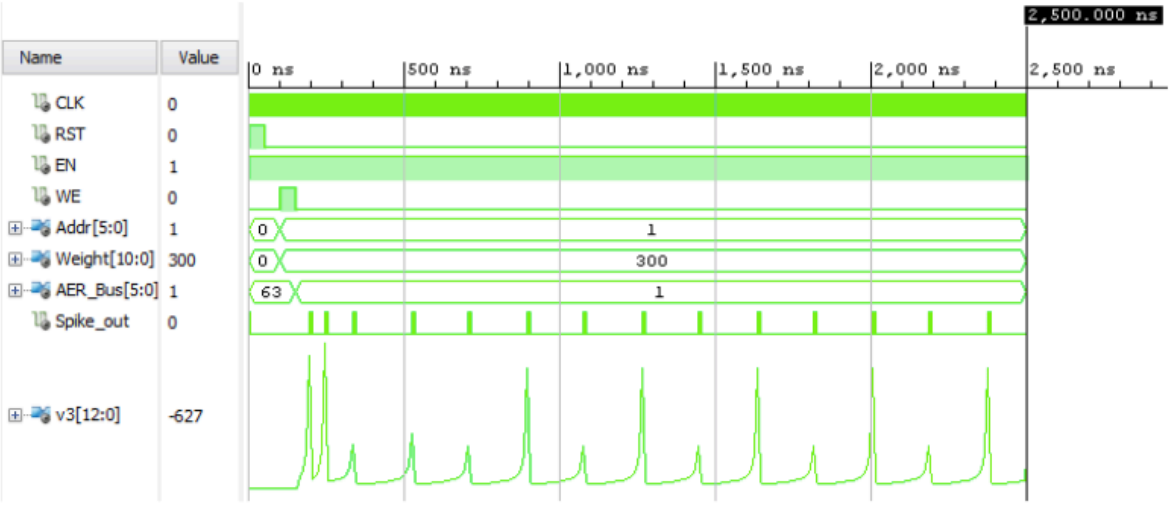


Рисунок 2.27 – Електричні вихідні режими генерації апаратної реалізації мультистабільного нейрона на нейро-модулі: *а* – розряди, *б* – бі-стабільні перемикання

Висновки до 2-го розділу

1. Комплексне застосування спайкового шифратора вхідного сигналу, рекурентних нейронів внутрішнього шару та вихідних нейронів мережі зумовило побудову імпульсної штучної нейронної мережі у вигляді системи класифікації, яка самонавчається і здатна автоматично адаптуватися до змін вхідного сигналу в режимі реального часу, що може бути застосовано для оброблення складних клінічно-значущих випадків ЕКГ-сигналу.

2. Надання властивостей адаптивності оптимізованій моделі імпульсного штучного нейрона досягнуто забезпеченням електричної мультистабільності та здатності відтворювати патерни електричної активності біологічних об’єктів з одночасним збільшенням обчислювальної потужності завдяки використанню моделі в якості базового компонента нейроморфного модуля.

3. Реалізовано апаратний модуль штучного нейрона та мережі мовою програмування VHDL для подальшої розробки нейромодуля на базі інтегральної компонентної бази засобу із FGPA архітектурою.

Результати експериментальних досліджень даного розділу наведено в таких публікаціях:

[1] Д. В. Чернетченко, М. М. Мілих, та К. В. Луданов, “Апаратна реалізація імпульсної штучної нейронної мережі для детектування параметрів електрокардіографічного сигналу (ЕКГ)”, *Вісник Хмельницького національного університету. Технічні науки*, № 4(275), с. 126-133, 2019.

[2] D. V. Chernetchenko, “A Novel Method of Preprocessing and Spike Encoding of Electrocardiographic Signal for Multi-stable Spiking Neuronal Networks Application”, *Вчені записки Таврійського національного університету ім. В.І. Вернадського, Серія технічні науки*, т. 30(69), № 3, ч. 1, с. 191-199, 2019.

[3] D. V. Chernetchenko, R. S. Romaniuk, D. Sawicki, and G. Yusupova, “Analysis of electrical patterns activity in artificial multi-stable neural networks”, *Proceedings of SPIE –The International Society for Optical Engineering*, 7 p., 2019.

РОЗДІЛ 3
РОЗРОБЛЕННЯ АПАРАТНО-ПРОГРАМНОГО ЗАСОБУ ДЛЯ ОБРОБЛЕННЯ ЕЛЕКТРОКАРДІОГРАФІЧНИХ СИГНАЛІВ

3.1 Вибір платформи для розроблення нейроморфного модуля

Оптимальному рішенню за співвідношенням функціонал/ціна найбільш повно відповідає апаратно-програмне рішення від компанії Digilent, яка поставляє повний комплект розробника FPGA серії Basys на основі Xilinx® Spartan®-3E.

Модуль розроблення Digilent Basys II FPGA є платформою для професіонального розроблення різноманітних схемотехнічних рішень. Побудований на базі Xilinx® Spartan®-3E FPGA і контролера USB Atmel® AT90USB2 модуль Basys II надає розробнику повне, готове до використання обладнання, яке придатне для створення програмно-апаратних рішень від базових логічних пристроїв до складних контролерів.

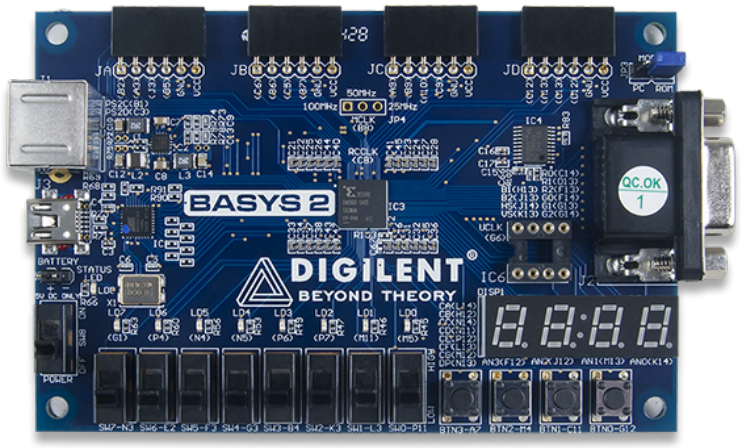


Рисунок 3.1 – Загальний вигляд модуля розроблення Digilent Basys II [20]

Чотири стандартні роз'єми розширення дозволяють збільшити функціонал Basys II, використовуючи плати, розроблені користувачем або Pmods – аналогові та цифрові модулі введення/виведення, які пропонують аналогово-цифрові перетворення A/D & D/A, драйвери двигунів і багато інших функцій. Сигнали на 6-контактних роз'ємах захищені від пошкодження і короткого замикання, що

забезпечує тривалий термін служби в будь-якому середовищі. Плата Basys II працює з усіма версіями інструментів Xilinx ISE®, включаючи безкоштовний WebPACK™. Він поставляється з кабелем USB, для забезпечення живлення та інтерфейсу програмування.

Вхідні і вихідні відповіді моделюються у середовищі ModelSim за допомогою стандартного тестового інструменту Wave. Програмування FPGA-плати Basys II відбувається за допомогою утиліти Digilent Adept (рис. 3.2, *а*). Еквівалентна електрична схема із вводами-виводами, а також цифровою панеллю індикації, яку використовують на етапі розроблення для відлагодження роботи пристрою, показана на рис. 3.2, *б*. При розробленні вбудованого програмного забезпечення для ПЛІС використано мову програмування VHDL, а моделювання та синтезу здійснені з використанням програмних засобів ModelSim і WebPACKTM ISE 13.3.

а ***б***

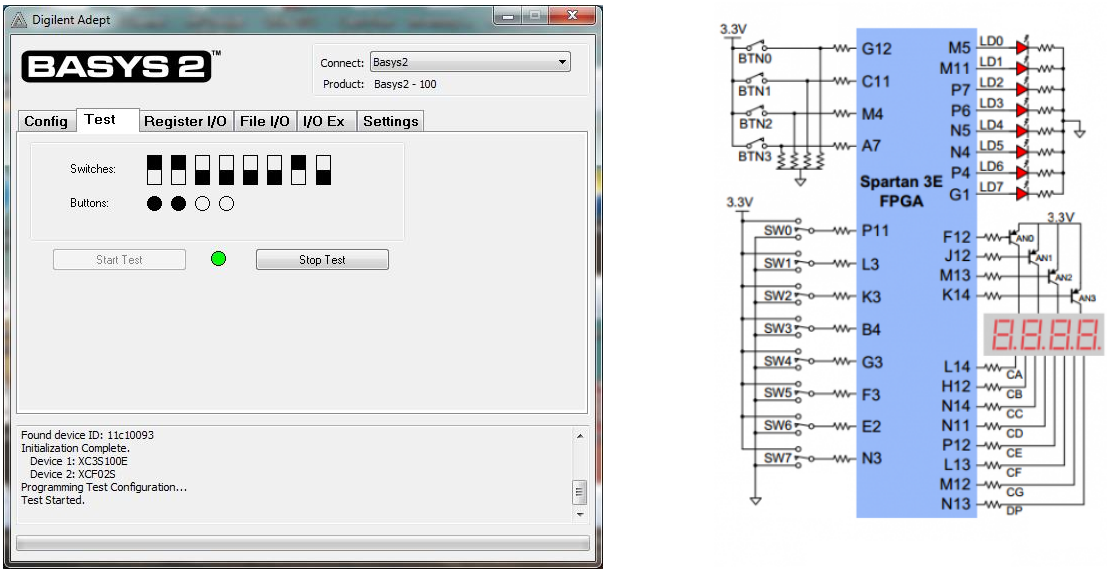


Рисунок 3.2 – Програмно-апаратна платформа розроблення FPGA: *а* – утиліта для програмування плати Basys II, *б* – еквівалентна електрична схема стендового пристрою на базі платформи розробника Digilent Basys II

Модель розроблена з використанням підходу “зверху вниз” шляхом декомпозиції складної конструкції на прості або модулі, кожний з яких визначається як автономна підсистема з більшою деталізацією з розділенням на

більшу кількість підсистем для зручності їх реалізації апаратними засобами ПЛІС. Синтезований код перевірено на інтегральній мікросхемі SPARTAN-3E XC3S100E за допомогою комплекту розробника BASYS II від DIGILENT, який сумісний з усіма інструментами Xilinx CAD, включаючи ChipScope, EDK та ISE WebPackTM [55].

Розроблення прототипу нейро-модуля. Spartan-3E є представником сьомого покоління в серії Spartan Series і третього покоління сімейства Xilinx, що випускаються за передовою технологією 90 нм. Повний дуплексний зв'язок ПЛІС плати із ПК здійснюється за допомогою інтерфейсу універсальної асинхронної передачі (UART) з параметрами: швидкість обміну даними – 115200 біт/с, довжина слова – 8 біт; парність - відсутня, стоп-біти – 1 біт.

Дані ЕКГ в цифровому форматі поступають на ПЛІС через інтерфейс UART безпосередньо до вхідного нейронного шару. Водночас, вхідні сигнали вводу/виводу нейрона виведені на ПЛІС платі на відповідні цифрові порти G12 (порт InNN_in), M5 (порт InNN_out). Вихід нейрону вихідного шару контролюється за допомогою осцилографа на виході M11 (порт OutNN_out) плати в реальному часі. Крім того, LED1 служить для візуальної індикації процесу генерації спайків на виході мережі. Тактуючий сигнал таймеру синхронізації відображається на виході C11 (порт CLK).

На рис. 3.3 показана еквівалентна електрична схема прототипу нейромодуля. UART TX-контакт з’єднано з P6 (TX), а RX-контакт – з M4 (RX). Для управління процесом додано кнопку користувача, яка підключена до вхідного порту A7. Процес навчання відображається з використанням світлодіодного індикатору LED2, а поточний стан системи – чотирьох панелей 7-сегментних індикаторів (DS1 – DS4).

Перетворення сигналу UART-TTL для подачі на стандартний USB-інтерфейс здійснюється типовим перетворювачем USB-UART на основі інтегрального рішення CH341A. Вихідні спайки поступають від ПЛІС-плати до ПК, де вони зберігаються в стандартному файлі в текстовому форматі з мітками часу для подальшої візуалізації та аналізу.

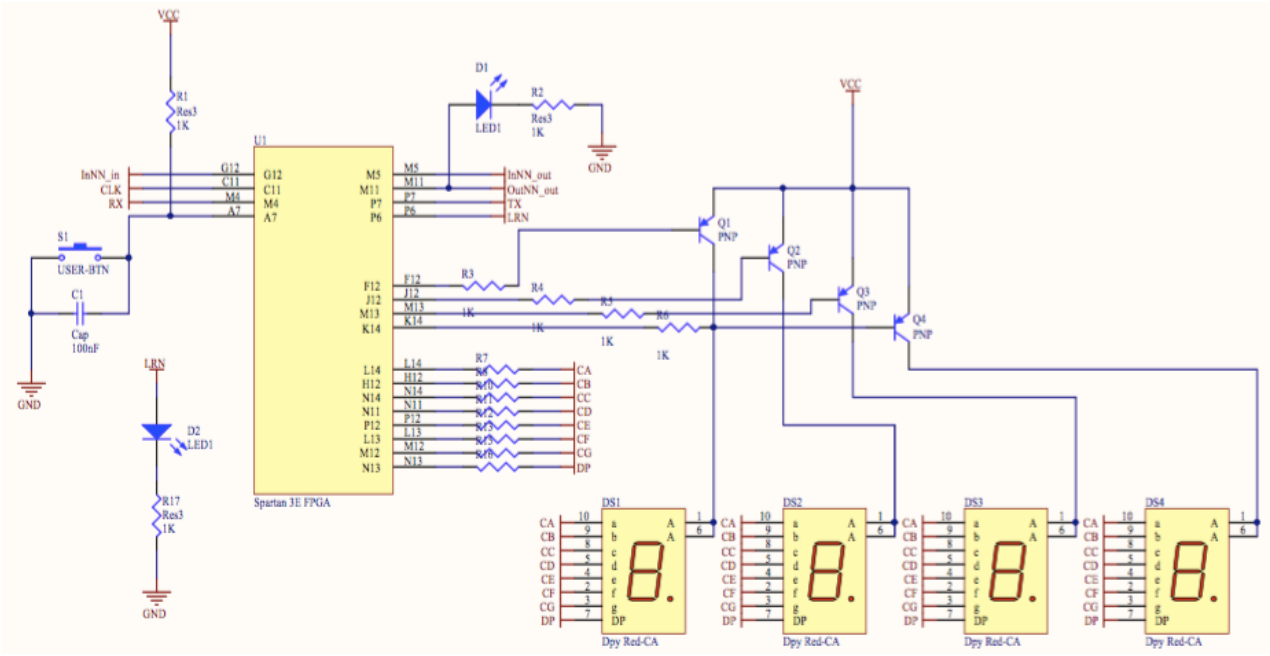


Рисунок 3.3 – Еквівалентна електрична схема нейро-модуля на базі платформи Digilent Basys II

Для керування аналоговою схемою, нейро-модулем та передачі даних до ПК використано мікроконтролер (МК) STM32L151C8T6, який є вбудованим рішенням на базі ядра ARM Cortex-M3, оскільки однією з головних тенденцій сучасного розвитку мікроконтролерних систем управління є перехід на 32-розрядні архітектури та програмування мовами високого рівня. Значний сегмент ринку 32-розрядних вбудованих систем займають вироби на основі мікропроцесорів і мікроконтролерів (МК) з архітектурою ARM (Advanced RISC Machine).

Сучасна структура і технологія виробництва дозволили досягти показника енергоспоживання ядра Cortex-M3 мікроконтролерів STM32 в 0.19 мВт/МГц. При цьому продуктивність за тестом Dhrystone становить 1.25 DMIPS/МГц проти 0.95 DMIPS/МГц для ARM7TDMI з набором команд Thumb. Максимальна тактова частота МК STM32 становить 72 МГц. Компанія STMicroelectronics заявляє про здатність своїх МК з ядром Cortex-M3 перевершувати DSP інших компаній при проведенні цілочисельних обчислень.

Розроблення схеми електричної принципової та друкованої плати пристрою було виконано за допомогою Altium Designer – системи

автоматизованого проектування (САПР) радіоелектронних засобів яка розроблена австралійською компанією Altium. На відміну від більш ранніх варіантів системи, є комплексною і забезпечує розроблення електронного засобу без використання інших програм і систем. Весь цикл проектування є наскрізним і складається з етапів розроблення електричної схеми, комп'ютерного моделювання її роботи, розроблення друкованої плати, розроблення комплекту конструкторської документації.

Особливістю програмного забезпечення системи є проектна структура і наскрізна цілісність ведення розроблення на різних рівнях проектування, коли зміни в розробленні на рівні плати можуть бути миттєво передані на рівень ПЛІС – схеми і навпаки. Водночас, перевагою даної програми є інтеграція ECAD і MCAD систем. Це зумовлює можливість розроблення друкованої плати в тривимірному вигляді з двосторонньою передачею інформації в механічні САПР (Solid Works, Pro/ENGINEER, NX та ін.).

В процесі розроблення апаратного засобу було використано два головних пакети Altium Designer, які базуються на єдиній інтегрованій платформі DXP:

– Altium Designer Custom Board Front-End Design – проектування ПЛІС, схемотехнічне проектування і моделювання [20];

– Altium Designer Custom Board Implementation – проектування друкованих плат і ПЛІС [21].

До складу програмного забезпечення Altium Designer входить необхідний інструментарій для розроблення, редагування та налагодження проектів на базі електричних схем і ПЛІС. Редактор схем (рис. 3.4) дозволяє вводити багато ієрархічні і багатоканальні схеми будь-якої складності, а також проводити змішане цифро-аналогове моделювання. Бібліотеки програми містять понад 90 тисяч готових компонентів, у багатьох з яких є моделі посадочних місць, SPICE та IBIS-моделі, а також тривимірні моделі, будь-яку з яких можна створити внутрішніми засобами програми.

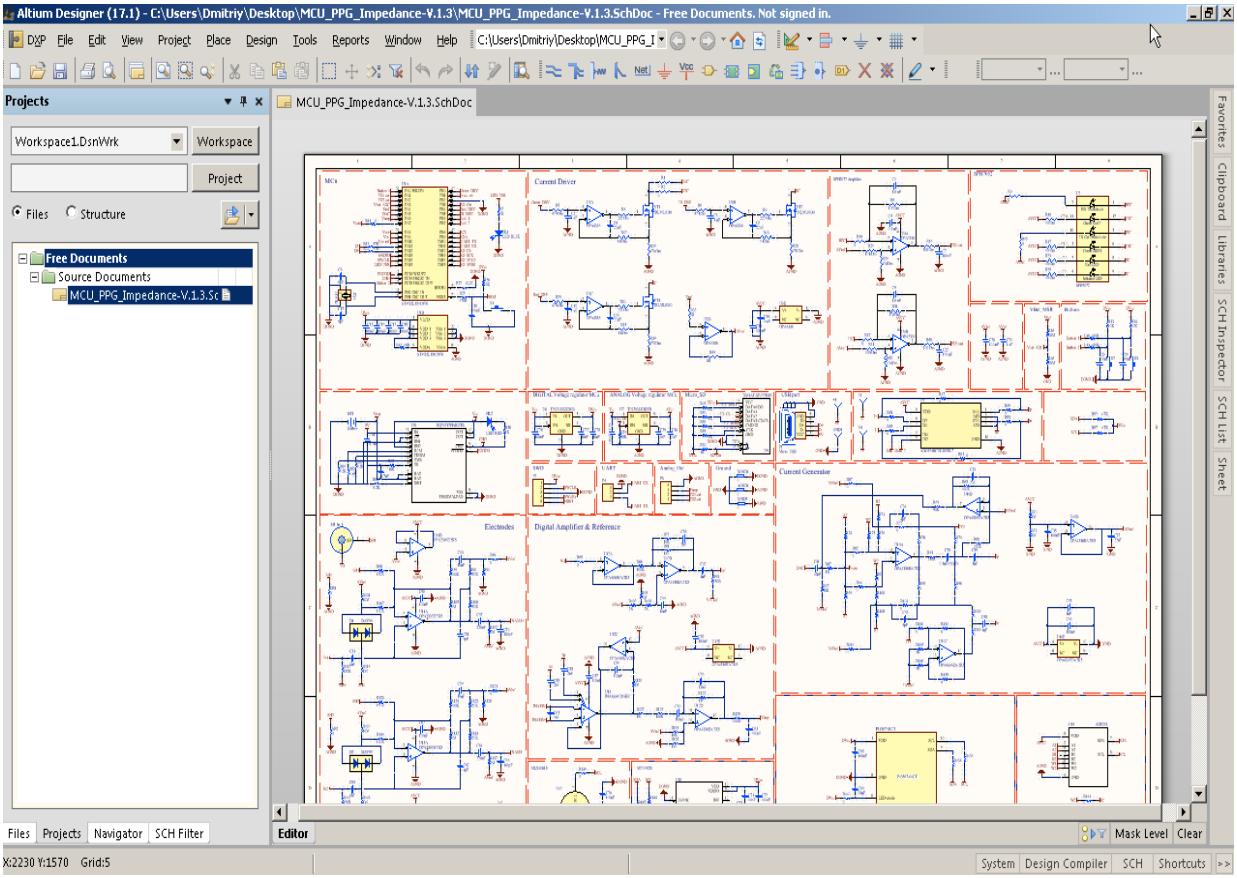


Рисунок 3.4 – Програмний проект розробленої схеми електричної принципової в редакторі електричних схем пакету Altium Designer

Редактор друкованих плат Altium Designer має потужні засоби інтерактивного розміщення компонентів і трасування провідників, які враховують всі вимоги, що висувають сучасні технології розробок, наприклад, при трасуванні диференціальних пар або високочастотних ділянок плат.

3.2 Розроблення архітектури та структурної схеми засобу

Структурна схема апаратно-програмного засобу для оброблення ЕКГ-сигналу включає такі головні компоненти (рис. 3.5): стандартні ЕКГ-електроди; блок фільтрації та формування сигналу (аналоговий ЕКГ-фронтенд); нейро-модуль на базі FPGA Xilinx Spartan 3E для реалізації SNN; мікроконтролерний пристрій для формування та передачі пакетів даних і керування системою; блок керування зарядом акумуляторної батареї (АКБ); ПК із спеціальним програмним

забезпеченням для реєстрації та аналізу отриманих даних на базі пакету Matlab і мови програмування Python.

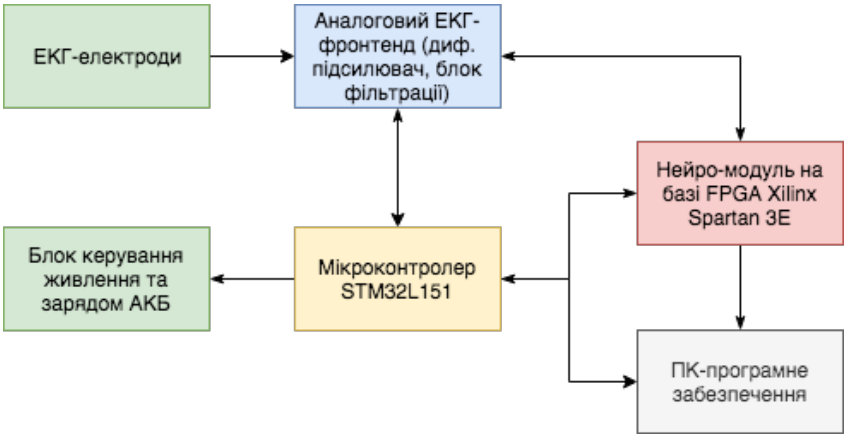


Рисунок 3.5 – Узагальнена структурна схема засобу для оброблення ЕКГ-сигналів

При роботі апаратно-програмного засобу в процесі оброблення та аналізу даних виникають шуми різного походження, які практично завжди змішуються з ЕКГ-сигналом і зумовлюють дрейф ізолінії, артефакти і дрібні коливання [66], [67], [70]. Експерименти показують, що стадія попереднього оброблення необхідна для нормального і стабільного вивчення SNN для реальних даних. Водночас, при оброблені вихідного ЕКГ-сигналу виникає необхідність введення етапу скидання фільтраційного попереднього оброблення, який складається з: фільтрації низьких частот з частотою зрізу 15 Гц; фільтрації високих частот з частотою зрізу 5 Гц; диференціальної фільтрації; фільтрації ковзного середнього (з вікном усереднення 0.125 с). Для забезпечення ефективної роботі нейронної мережі архітектура засобу має передбачати та перевірку та підтримку високого ступеню точності розпізнавання ЕКГ. Загальна архітектура аппаратно-програмного засобу із зазначеними потоками даних в системі та функціональними блоками показана на рис. 3.6.

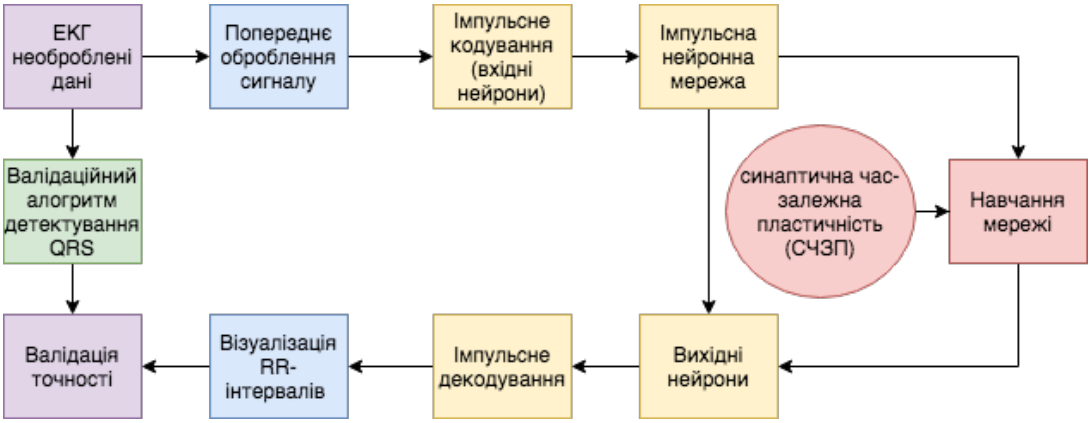


Рисунок 3.6 – Загальна архітектура засобу

3.3 Схемотехнічне рішення АПЗ на рівні схеми електричної принципової

ЕКГ-сигнал зазвичай характеризується амплітудою від 6 до 200 мкВ та частотним діапазоном від 10 до 500 Гц. Для того, щоб мати можливість працювати з таким сигналом, необхідно попереднє його підсилення, що необхідно для побудови відображення ЕКГ-сигналу та його подальшого детектування та розпізнавання.

Найбільш розповсюджений класичний за своєю суттю підхід для підсилення ЕКГ-сигналу – це використання диференційного підсилювача. Завдяки закладеній у більшості диференційних підсилювачів властивості придушення синфазної завади, завади, що наводяться від зовнішніх полів (які є синфазними для всього тіла людини), досить добре придушуються ще на первинній стадії диференційного підсилення сигналу, який надходить з електродів.

В дисертаційній роботі в якості базового обрано диференційний підсилювач фірми Texas Instruments INA333 [73]. Серед головних його переваг – коефіцієнт підсилення, що можна програмувати; нульовий дрейф ізолінії; низький струм витоку по входу підсилювача – порядку 100 пА; низька напруга живлення (від 1.8 В); досить мале споживання струму під час активного режиму роботи – не більше 40 мА; досить хороший коефіцієнт придушення синфазної

завади (до 100 дБ у діапазоні до 3 кГц), що робить можливим використання даного підсилювача у портативних пристроях для контролю стану серця.

В роботі розроблено аналоговий ЕКГ-фронтенд пристрою (рис. 3.7, *а*, *б*). Розглянемо окремі функціональні блоки схеми пристрою.

а ***б***

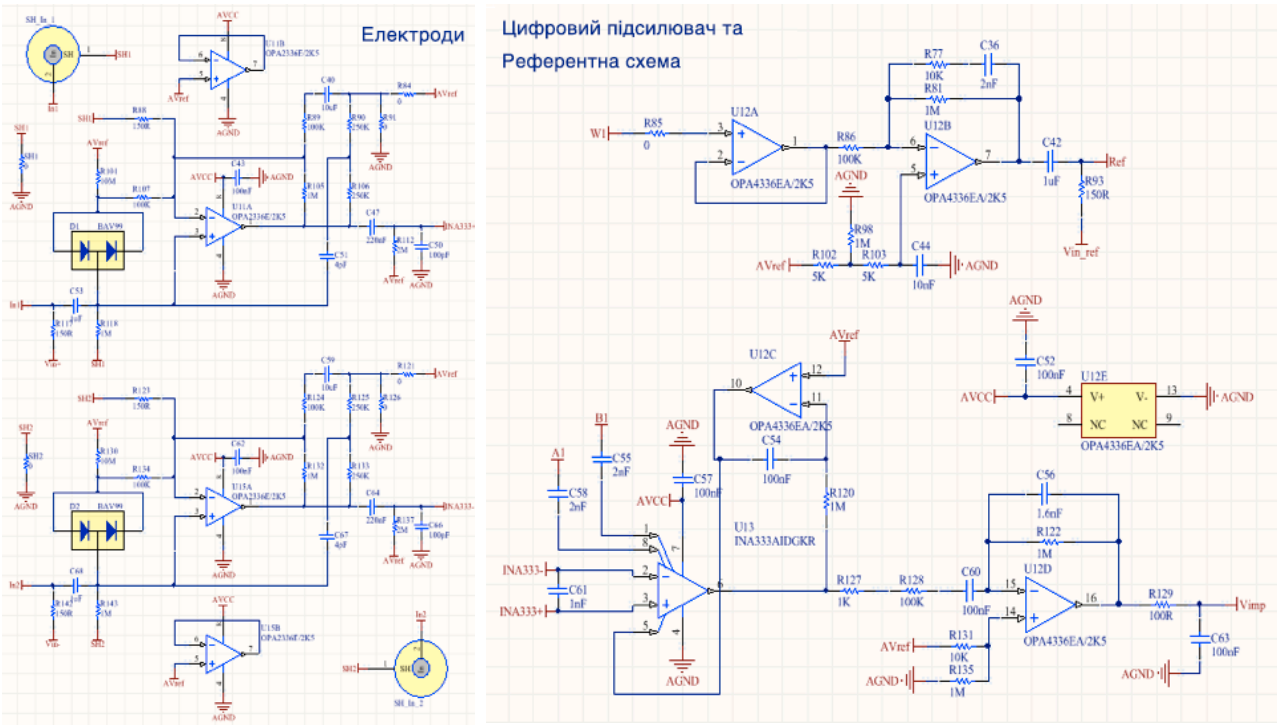


Рисунок 3.7 – Схема електрична принципова аналогового ЕКГ-фронтенду: *а* – схема каскадів підсилення та фільтрації вхідних сигналів з електродів; *б* – каскад диференційного підсилювача сигналу

Аналоговий ЕКГ-фронтенд модуль. Аналоговий сигнал з активних електродів In_1і In_2 надходить на буферні каскади до операційних підсилювачів (ОП) *U15A* OPA2336E/2K5 і *U15B* OPA2336E/2K5, призначених для узгодження імпедансу вхідних кіл: вони мають дуже значний вхідний опір ~ 10 МОм, та маленький вихідний опір. Для цього каскади ОП вмикають за схемою з негативним зворотним зв'язком і коефіцієнтом підсилення $K = 1$. Конденсатори *C3* і *C10* (*C3* = *C10* = 1 мкФ) грають роль захисних ємностей по входах ОП. Виходи ОП (*U15A* і *U15B*) з'єднуються через погоджуючи резистори *R2* і *R21* (*R2* = *R21* = 10 кОм) із захисними екранами кабелів електродів. Захисні екрани захищають систему від зовнішніх електромагнітних завад.

Вихідний сигнал з буферних ОП подається на RC-фільтри високих частот, які зібрані з конденсаторів *C6* і *C11* (*C6* = *C11* = 220 нФ) і резисторів *R10* і *R27* (*R10* = *R27* = 1 МОм). Частота зрізу f_c = 0.72 Гц. Також резистори *R10* і *R27* формують опорну напругу (V_{ref}) на входах аналогового інструментального підсилювача *U13* (INA333), що і забезпечує умовну точність номіналів *R10* і *R27* для врівноваження плечей підсилювача *U13* не повинна перевищувати 0.5%. Фільтр радіочастотних ВЧ-завад реалізовано на конденсаторах *C5*, *C9* і *C12* (*C5* = *C9* = *C12* = 100 пФ), які служать для додаткового захисту входів підсилювача від ВЧ-завад.

Інструментальний підсилювач *U13* являє собою малопотужний, прецизійний диференційний підсилювач (ДП). Основна його роль в схемі – підсилення різниці потенціалів *ΔU* = (*U1* – *U2*) на електродах *In1* і *In2*, для виділення корисного сигналу P-QRS-T комплексу ЕКГ. Коефіцієнт підсилення підсилювача *U13* задається за допомогою резисторів *R4* і *R5* (*R4* = *R5* = 5.6 кОм) і розраховується за формулою

$$G = 1 + \frac{100\ \text{кОм}}{R_g}, \tag{3.1}$$

де *Rg* = 2·*R4* = 2·*R5*.

Для резисторів *R4* і *R5* коефіцієнт підсилення дорівнює:

$$G = 1 + \frac{100\ \text{кОм}}{2{\cdot}5.6\ \text{кОм}} = 9.92{\sim}10. \tag{3.2}$$

Опорна напруга підсилювача *U13* INA333 формується за допомогою ОП *U12A* OPA4336EA/2K5 і фільтрів високих частот, включених у коло негативного зворотного зв'язку на ємності *C8* = 100 нФ і резисторі *R12* = 1 МОм. Частота зрізу фільтра

$$f_c = \frac{1}{2\pi{\cdot}C8{\cdot}R12} = 1.6\ \text{Гц}. \tag{3.3}$$

Фільтри високих частот призначені для усунення дрейфу ізолінії. З виходу підсилювача *U13* аналоговий сигнал надходить на каскад подальшого підсилення і фільтрації на ОП *U12B*. Фільтр у колі негативного зворотного зв'язку на резисторі *R7* = 1 МОм та ємності *C4* = 1.6 нФ, є RC-фільтром низьких частот з частотою зрізу

$$f_c = \frac{1}{2\pi \cdot C4 \cdot R7} = 99.5 \text{ Гц.} \quad (3.4)$$

Коефіцієнт підсилення каскаду G_2 обчислюється за формулою:

$$G_2 = \frac{R7}{R15} = 10. \quad (3.5)$$

Резистор *R24* = 1 МОм призначено для зменшення вхідного струму витоку (*Ib*), а *R22* = 10 кОм - для врівноваження плечей ОП *U12B*. Первинно підсилений і відфільтрований сигнал з виходу ОП подається на вхід програмованого аналогового підсилювача *U4* MCP6S21-I/MS, який реалізує схему автоматичного регулювання підсилення (АРП).

Програмований підсилювач *U4* має цифровий інтерфейс (типу SPI) для управління з мікроконтролерного блоку; аналоговий вхід; сигнальний вихід і вхід для подачі опорної напруги схеми (V_{ref}). Коефіцієнт підсилення АРП задається і може набувати значень *G3* = {1, 2, 4, 5, 8, 10, 16, 32}. Таким чином, загальний коефіцієнт підсилення схеми $G = G_1 \cdot G_2 \cdot G_3$ і приймає мінімальне значення: $G_{\text{min}} = 10 \cdot 10 \cdot 1 = 100$, та максимальне значення $G_{\text{max}} = 10 \cdot 10 \cdot 32 = 3{,}200$. Отже, коефіцієнт підсилення схеми варіюється в діапазоні від 100 до 3200 в залежності від керуючого сигналу мікроконтролера. Резистор *R49* = 1 МОм служить для зменшення вхідного струму витоку ОП, резистор *R16* = 10 кОм встановлює відповідний рівень вхідної опорної напруги ОП V_{ref}.

Мікроконтролерний блок керування. Блок реалізовано на базі мікроконтролера із ядром Cortex-M3 STM32L151C8T6 блок керування (рис. 3.8). Електрод *In1* з'єднаний з виходом цифро-аналоговим перетворювачем (ЦАП) МК (*DAC1*) за допомогою резистивної перемички *R30* = 0 Ом, служить для подачі на електрод опорної напруги і забезпечує контроль якості контакту електроду з шкірою. Електрод *In2* з'єднаний із входом аналогово-цифрового перетворювача (АЦП) МК (*PA3*) за допомогою резистивної перемички *R33* = 0 Ом і призначений для реєстрації напруги на електроді та контролю якості контакту зі шкірою.

При увімкненні режиму вимірювання міжелектродного імпедансу генерується імпульсна напруга протягом 0.5 с, при цьому вхід PA3 під’єднується

до землі (*GND*) через внутрішній опорний резистор МК, а через коло між електродами *In1* і *In2* починає протікати струм i_1, що призводить до падіння напруги між електродами *In1* і *In2*, величина якого залежить від імпедансу. Величина імпедансу водночас залежить від якості контакту кожного з електродів зі шкірою: чим краще контакт, тим менше величина імпедансу. Величина опору при хорошому контакті з електродом складає близько 100 – 300 кОм.

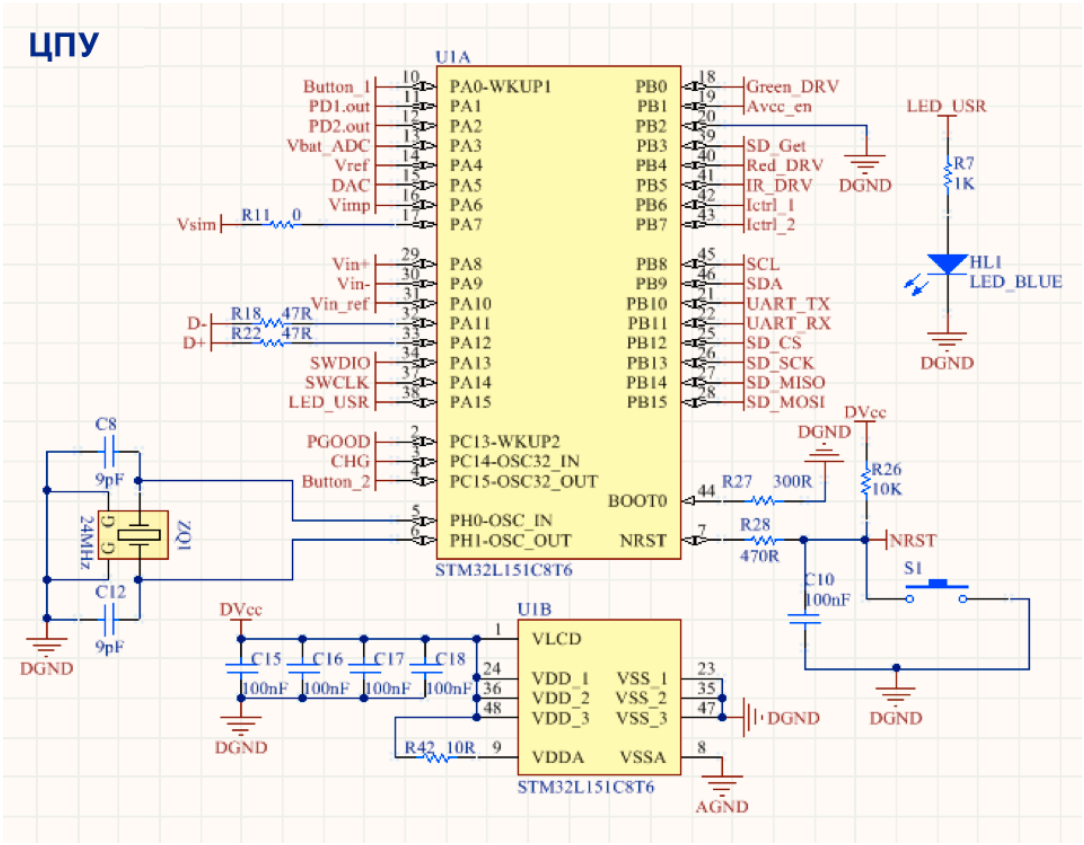
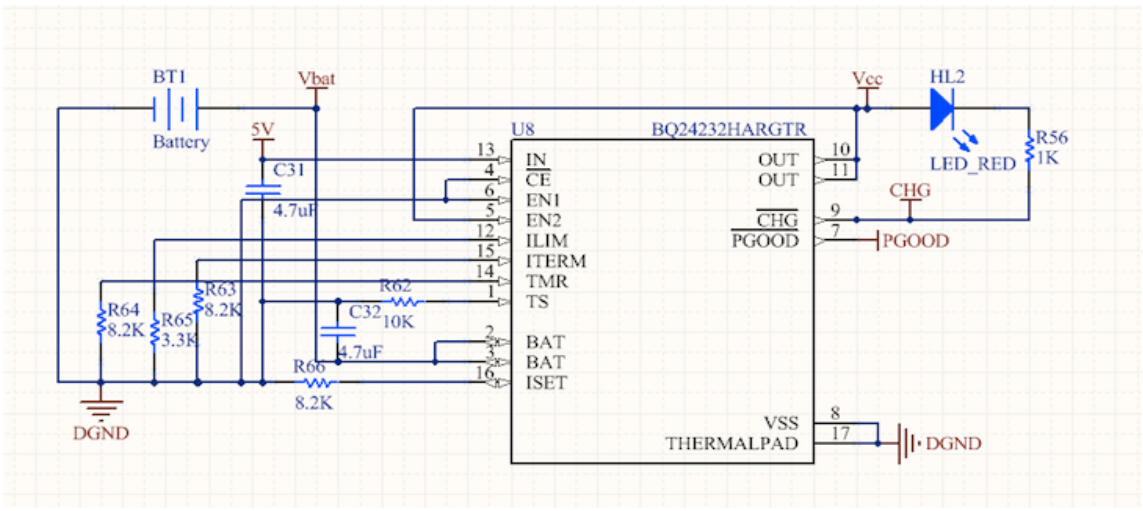


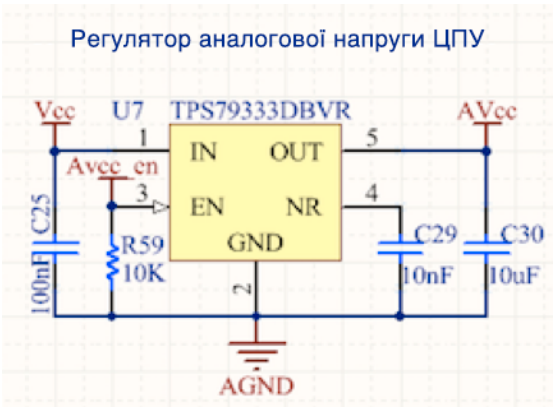
Рисунок 3.8 – Мікроконтролерний блок керування на базі мікроконтролеру STM32L151C8T6

Модуль зарядного пристрою акумуляторної батареї. Схема електрична принципова модуля містить блок зарядки і керування BQ24232RGTRG4, який працює в діапазоні напруги 3.4 В (пристрій розряджається) – 4.2 В (пристрій заряджено) (рис. 3.9).

а



б



в

Рисунок 3.9 – Схеми електричні принципові: *а* – модуля зарядного пристрою; *б* – стабілізатора для формування аналогової напруги; *в* – стабілізатора для формування цифрової напруги

Робоча напруга використаної акумуляторної батареї типу Li-Po складає 3.7 В. Процеси її зарядки та розрядки нелінійні (рис. 3.10, *а, б*). На графіках синім кольором показана зміна напруги акумулятора з часом, а червоним – зміна струму з часом.

Резистор *R43* (R_{ILIM}) має значення від 3.1 кОм до 7.8 кОм і обмежує струм в режимі програмування. Вхідний струм включає в себе струм системи навантаження і струм заряду батареї:

$$R43\ (R_{ILIM}) = K_{ILIM}/I_{I\_MAX}$$

$$I_{I\_MAX} = 0.4\ \text{A}$$

$$K_{ILIM} = 1{,}470\ \text{А}\cdot\text{Ом}$$

а ***б***

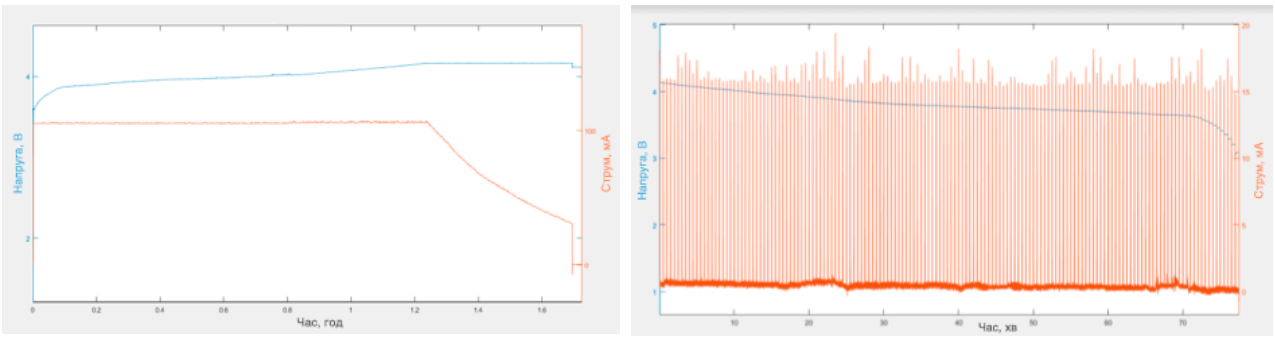


Рисунок 3.10 – Процес для акумулятора типу Li-Po (зміна струму та напруги із часом): *а* – зарядка; *б* – розрядка акумулятора в процесі роботи пристрою. Вісь ординат – напруга, В; струм, мА. Вісь абсцис – час, хв

Резистор *R45* (R_{ISET}) має значення від 3 кОм до 36 кОм і встановлює максимальний струм під час швидкої зарядки акумулятора. Зарядка вимикається, якщо I_{SET} залишається непідключеним. Під час зарядки, напруга на I_{SET} відображає фактичний заряд струму і може бути використана для контролю струму заряду:

$$R45\ (R_{ISET}) = K_{ISET}/I_{CHG}$$

$$I_{CHG} = 0.1\ \text{A}$$

$$K_{ISET} = 870\ \text{A}\cdot\text{Ом}$$

Вихід *T*s необхідно з'єднати з термістором *R39* = 10 кОм. Водночас, коли функція *T*s не використовується, фіксований резистор 10 кОм від *T*s відмикається до V_{SS} для підтримки допустимого рівня напруги на *T*s.

На вхід *IN* подається напруга від зовнішнього джерела живлення постійного струму (адаптер змінного струму або USB-порт). Вхідний робочий діапазон становить від 4.35 до 6.6 В.

Вихід *OUT* забезпечує регульований вихід, коли вхід знаходиться нижче порогового значення *OVP* і вище напруги регулювання. Коли значення входу не з робочого діапазону, *OUT* підключений до V_{BAT}. Резистор *R38* обмежує струм, що дає можливість уникнути пошкодження світлодіоду. До V_{BAT} необхідно підключити керамічний конденсатор *C23* в діапазоні від 4.7 до 47 мкФ, а до *CHG* – індикацію стану зарядки.

Світлодіод *HL2* в розрядженому стані горить червоним світлом, а в зарядженому не світиться. Час зарядки залежить від ємності акумулятора і для *C* = 200 мА/год час зарядки становить 1 годину 45 хвилин.

Схема формування напруги. ANALOG Voltage regulator (AVR, регулятор аналогової напруги) і DIGITAL Voltage regulator (DVR, регулятор цифрової напруги) стабілізують напругу 3.3 В для живлення аналогової та цифрової частини схеми.

Вхідний конденсатор *C16* у схемі AVR (або *C17* у схемі DVR) дорівнює 0.1 мкФ та використовують для фільтрації вхідної напруги, а вихідний – *C14* = 10 мкФ (або *C15* у схемі DVR) – для забезпечення оптимального часу відгуку при динамічному навантаженні. Діапазон вхідної напруги V_{cc} становить від 3 В до 4.2 В, вихідна напруга DV_{cc} – 1.8 В ± 1 %.

Схема референсного електроду. У схемі з референсним електродом резистори R17 і R18 усувають завади від мережі і радіохвиль і підключаються до референсного електроду. Принцип роботи референсного електрода оснований на тому, що на тіло людини синфазно подається підсилений вхідний сигнал. Він створює з тілом людини загальний віртуальний потенціал, який вираховується з потенціалу кожного вхідного активного електрода, що забезпечує додаткове придушення синфазної завади і завади 50/60 Гц.

Резистор *R8* = 100 Ом обмежує вихідний струм, який подається на шкіру. *R6* = 10 кОм задає коефіцієнт підсилення вихідного каскаду (для даного випадку *G* = 1 МОм / 10 кОм = 100). На операційному підсилювачі *U1C* OPA4336EA/2K5 реалізовано буферний каскад, який має великий вхідний опір і малий вихідний імпеданс. Конденсатор фільтру *C2* = 100 нФ служить для попередження різких перепадів напруги.

Плату друковану розроблено за допомогою САПР Altium Designer, верхній та нижній шар зображено на рис. 3.11, *а* та *б* відповідно.

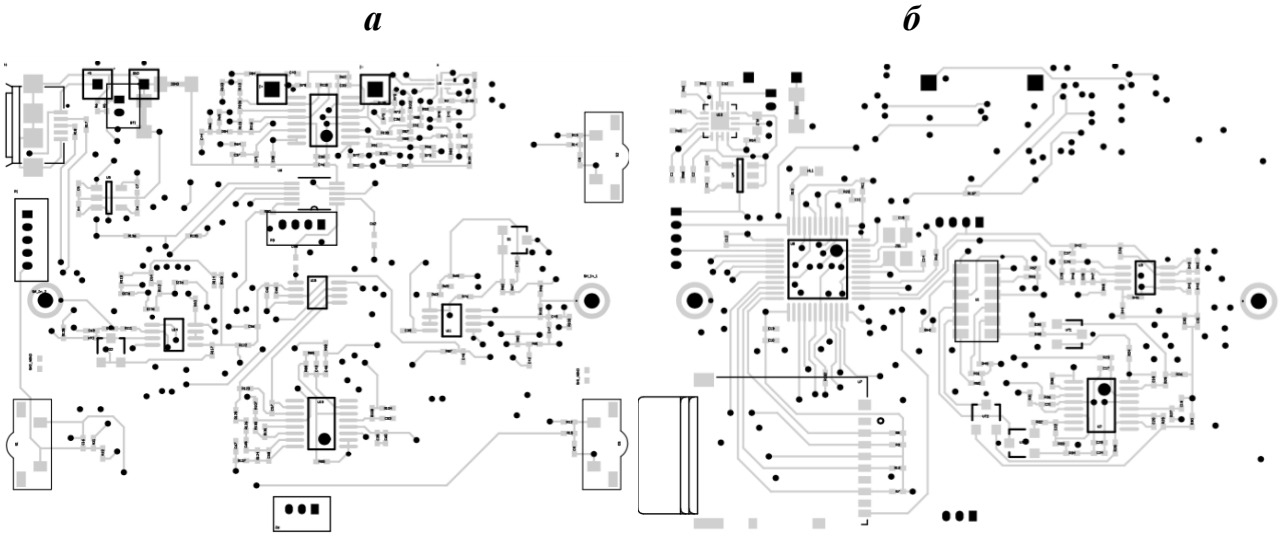


Рисунок 3.11 – Схема друкованої плати (PCB) мікроконтролерного модуля із аналоговим ЕКГ-фронтендом: *а* – верхній шар; *б* – нижній шар.

Друкована плата пристрою має 4 шари: 2 зовнішніх, та 2 внутрішніх. Загальні геометричні розміри друкованої плати – 40 х 40 мм. Використані інструменти автоматичного трасування із встановленими правилами для з’єднання всіх компонентів відповідно схемі електричній принциповій. Для збільшення інтеграції електронних компонентів та підвищення ергономічності пристрою використані інтегральні схеми в корпусі із поверхневим монтажем (SO, SOIC, SOT). Посадкові місця на платі під дискретні компоненти (резистори та конденсатори) обрано типорозмірів 0402 та 0805. Друкована плата пристрою виготовлена за допомогою склотекстоліту FR4, із наступними параметрами: мінімальна ширина провідної траси – 0.1 мм; мінімальний зазор – 0.1; діаметр металізованих перехідних отворів – 0.2 мм; товщина друкованої плати – 1.16 мм; плата виготовлена із застосуванням маскування. Зовнішній вигляд друкованої плати пристрою зображено на рис. 3.12.

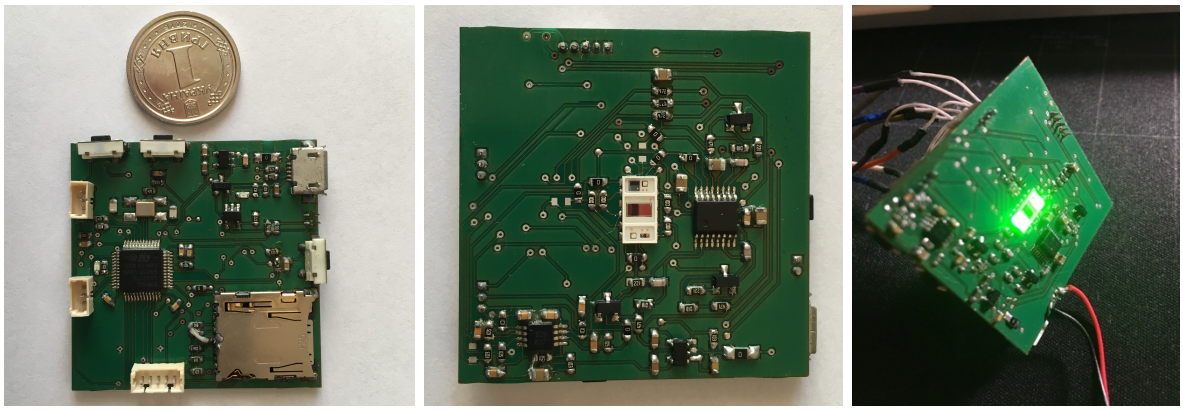


Рисунок 3.12 – Зовнішній вигляд друкованої плати прототипу пристрою.

Для реєстрації сигналу необхідно підключити електроди: з одного боку до пацієнта, з іншого – до інтерфейсу електродів *In1*, *In2*, In_{ref}. Живлення схеми забезпечується або за допомогою зовнішнього USB блоку живлення 5 В, або від акумуляторної батареї Li-Po 3.7 В.

3.4 Розроблення алгоритмічно-програмного забезпечення засобу

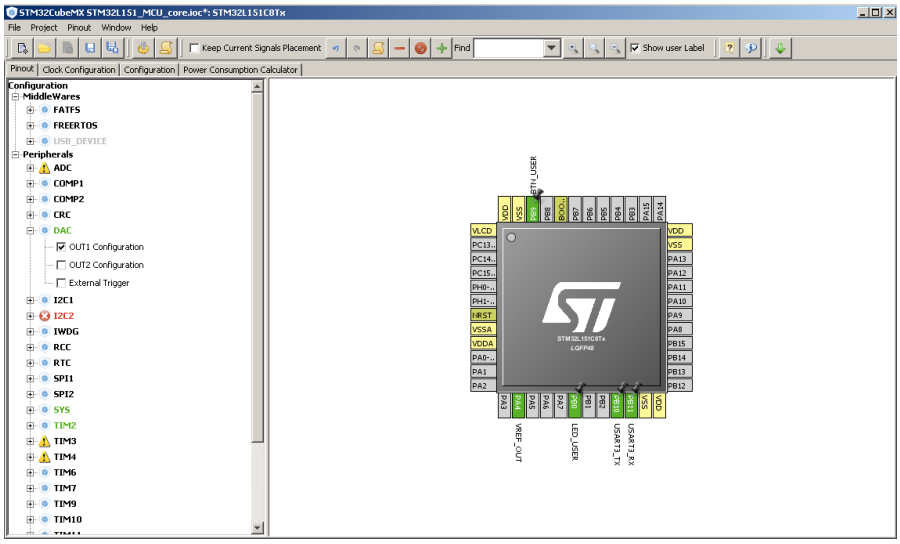
3.4.1 Вбудоване програмне забезпечення мікроконтролера

Програмне середовище розробленого засобу побудовано на основі програмного пакету STM32CubeMX, який фактично представляє собою автоматизоване робоче місце для розроблення систем на базі 32-бітових мікроконтролерів STM32, виконаних на основі ядер ARM Cortex. Пакет зарекомендував себе як зручне середовище для повноцінного налагодження конфігурації МК з видачою пакета файлів ініціалізації, придатних для подальшого використання в ряді систем розроблення і налагодження керуючого коду МК. STM32CubeMX значно спрощує створення вбудованого ПЗ, та прискорює цей процес.

STM32CubeMX є розширенням іншого інструменту – MicroXplorer, і складається з набору модулів багаторівневого вбудованого ПЗ. Він забезпечує зручне і наочне відображення та конфігурацію зовнішніх виходів і внутрішніх

ресурсів мікроконтролера зі створенням на виході відповідної програми ініціалізації мовою C/C++, яка придатна для подальшого доповнення керуючим кодом. Вікно програмного проекту для МК STM32L151C8T6 наведено на рис. 3.13, *а*.

а



б

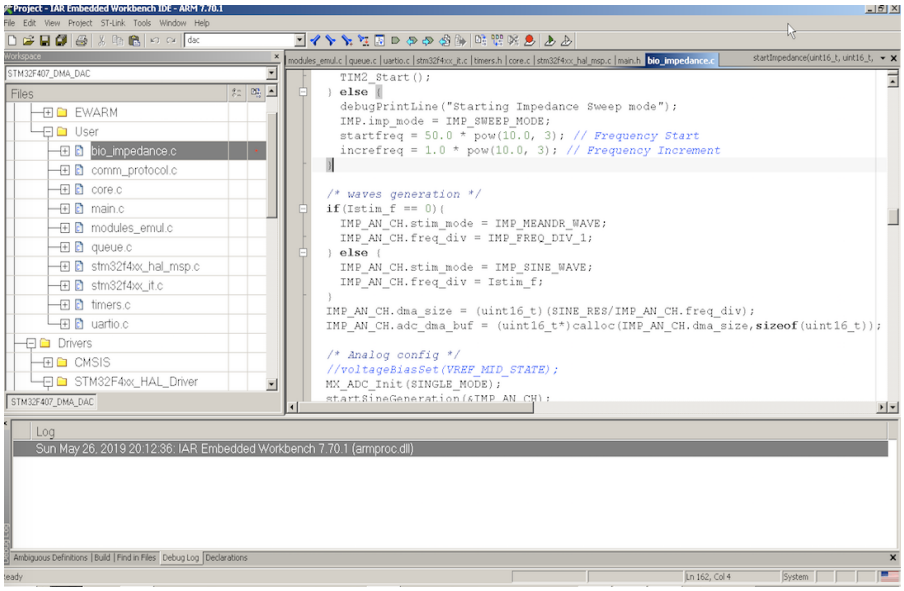


Рисунок 3.13 – Програмні засоби розробки вбудованого програмного забезпечення мікроконтролера STM32L151C8T6: *а* – STM32CubeMx; *б* – IAR Embedded Worknench

Після вибору периферії SPI2 і UART, налаштування головних вузлів МК (DAC, DMA), а також GPIO загального призначення, виконується функція Generate Source Code, за допомогою якої Cube виконує генерацію вихідного

програмного проекту. Після закінчення процесу генерації отримуємо на виході готовий програмний проект в середовищі IAR Embedded Worknench, який далі можна відкрити, редагувати, компілювати і програмувати МК. В роботі було використано IAR Embedded Worknench версії 7.50. Відкритий програмний проект пристрою в середовищі розроблення IAR Embedded Worknench виглядає таким чином (рис. 3.13, *б*).

Запропоновано та розроблено наступну структуру програмного проекту:

– ecg_module.c: головний блок проведення ЕКГ випробувань, математика оброблення та цифрової фільтрації сигналу.

– comm_protocol.c: модуль комунікаційного протоколу для зв'язку МК із ПК.

– core.c: налаштування ядра МК.

– main.c: головний цикл програмного забезпечення.

– queue.c: програмна черга для відправлення даних до ПК за допомогою інтерфейсу UART.

– timers.c: програмна реалізація процесів, керованих апаратними таймерами МК.

– uartio.c: модуль відправлення та прийняття даних за допомогою інтерфейсу UART.

Повний код розробленого проекту вбудованого програмного забезпечення наведено у Додатку Г.

Для комунікації та керування апаратним засобом за допомогою ПК розроблено програмне забезпечення мовою програмування Python. У середовищі розробки PyCharm було розроблено проект за допомогою інструментів мови програмування Python для взаємодії із мікроконтролером. Головна мета програмного забезпечення полягала у відправленні керуючих команд до пристрою та отримання зареєстрованих даних від пристрою. Програмне забезпечення має змогу за допомогою елементів графічного інтерфейсу керувати процесом запуску вимірювань, контролювати стан контакту ЕКГ-електродів із пацієнтом, виконує візуалізацію даних, отриманих з нейро-модуля у режимі

реального часу. Головним модулем програмного забезпечення є модуль взаємодії із пристроєм за допомогою інтерфейсу UART, для вирішення цієї задачі було використано бібліотека для роботи із серійним портом у Python – PySerial. Повний код розробленого проекту програмного забезпечення для ПК наведено у Додатку Д.

3.4.2 Попереднє оброблення і фільтрація сигналу

Перший крок в обробленні вхідних ЕКГ-сигналів полягає в попередньому обробленні та фільтрації сигналу від зовнішніх завад та шумів, різних за походженням: дрейф ізолінії, артефакти руху, низькоамплітудні осциляції, інтерференційні завади від зовнішніх ліній електромереж або електроміографічні артефакти [81]–[86]. Загальна схема тракту фільтрації ЕКГ складається з таких блоків: фільтр нижніх частот, фільтр верхніх частот, диференційний фільтр, квадратичний фільтр, згладжуючий фільтр (рис. 3.14).

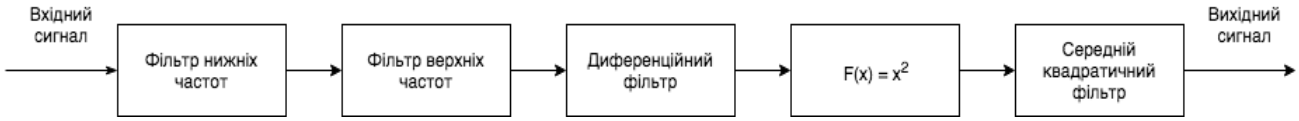


Рис. 3.14 Загальна блок-схема тракту фільтрації ЕКГ-сигналу

Етап фільтрації додатково промодельовано у середовищі MATLAB, що дозволило отримати цифровий фільтр нижніх частот із передаточною функцією *H(z)*:

$$H(z) = \frac{(1 - z^{-6})^2}{(1 - z^{-1})^2}. \tag{3.6}$$

Частота зрізу фільтра становить близько 15 Гц, затримка сигналу – п’ять семплів, а коефіцієнт підсилення – 36. Такий фільтр здатен ефективно придушувати завади від зовнішніх ліній електропередачі при оброблені ЕКГ-сигналів з частотою дискретизації 250 Гц. Цифровий фільтр було перевірено на стійкість, при $H(z) = 0$ виконується умова $z < 1$. Сигнал з виходу згладжуючого

фільтра наведено на рис. 3.15.

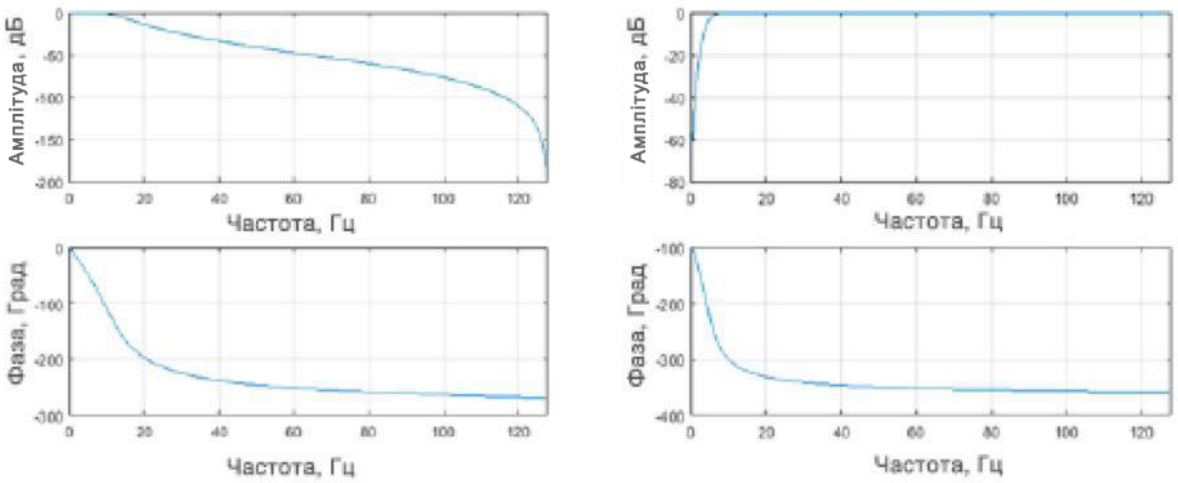


Рисунок 3.15 – Амплітудно-частотна та фазово-частотна характеристики розроблених цифрових фільтрів вхідного ЕКГ-сигналу: *а* – фільтру нижніх частот; *б* – фільтру верхніх. Вісь ординат – вісь амплітуда сигналу в дБ для амплітудної характеристики та фаза в градусах для фазово-частотної характеристики. Вісь абсцис – частота сигналу, Гц

Також було використано фільтр високих частот з частотою зрізу 5 Гц і передаточною функцією *H(z)* для боротьби з низькочастотним шумом та дрейфом ізолінії:

$$H(z) = \frac{(-1 + 32z^{-16} + z^{-32})}{(1 + z^{-1})}. \quad (3.7)$$

Цифровий фільтр високих частот було перевірено на стійкість, при $H(z) = 0$ виконується умова $z < 1$. Після фільтра високих частот необроблені дані надходять на диференційний фільтр з передаточною функцією *H(z)*:

$$H(z) = \frac{1}{8T}(-z^{-2} - 2z^{-1} + 2z + z^{2}). \quad (3.8)$$

Цифровий диференціальний фільтр було перевірено на стійкість, при $H(z) = 0$ виконується умова $z < 1$. На наступному етапі, виконується підсилення сигналу до квадрату, що значно збільшує відношення “сигнал-шум” (SNR)

відфільтрованого сигналу. На рис. 3.16, *а* показано приклад виконання операції підсилення сигналу до квадрату.

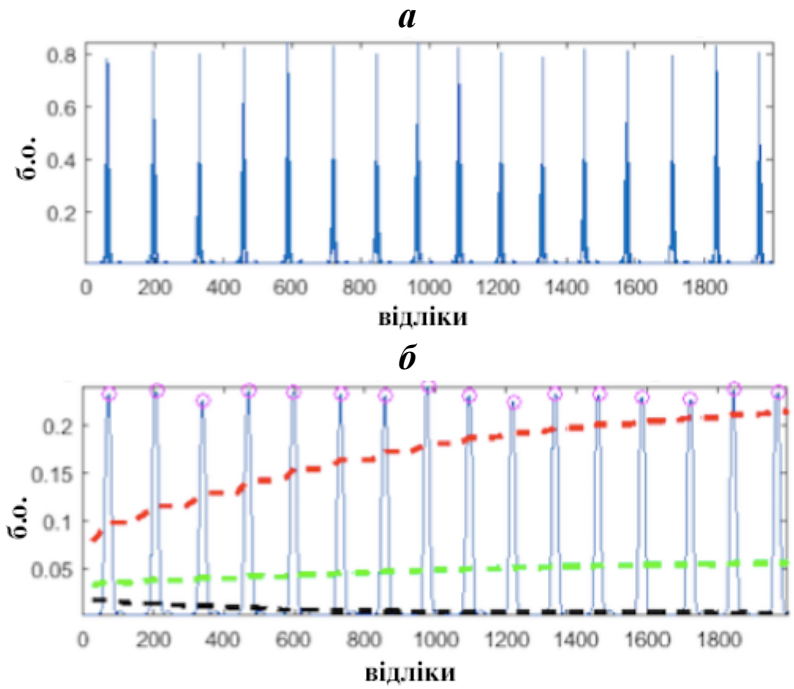


Рисунок 3.16 – Результат фільтрації вхідного ЕКГ-сигналу за допомогою тракту цифрової фільтрації: *а* – квадратичний нормалізований ЕКГ сигнал після стадії фільтрації; *б* – згладжений сигнал за допомогою фільтру MWI та відповідні детектовані R-піки ЕКГ-сигналу (кружечки) і рівні сигналу: рівень сигналу – красна пунктирна лінія; рівень шуму – чорна пунктирна лінія; адаптивний поріг – зелена пунктирна лінія. Вісь ординат – амплітуда нормалізованого сигналу у безрозмірних одиницях (б. о.). Вісь абсцис – номер семплу ЕКГ-сигналу

На останньому етапі відфільтровані дані пропускають через віконний фільтр середнього (MWI). Сигнал усереднюється для фільтрації від високочастотних завад за допомогою вікна довжиною 0.150 с:

$$Y(nT)=\frac{1}{N}[x(nT-(N-1)T)+x(nT-(N-2)T)+\ldots+x(nT)\,], \tag{3.9}$$

де Y – вихідний сигнал фільтра;

nT – номер відліку ЕКГ-сигналу;

N – загальна кількість семплів у вікні усереднення;

x – значення амплітуди ЕКГ-сигналу в кожний момент часу.

Результат проходження сигналу через згладжуючий фільтр показано на рис. 3.16, *б*. Програмний код фільтрації та спайкового кодування вхідного ЕКГ-сигналу у середовищі Matlab наведено у Додатку Е.

3.4.3 Програмний алгоритм детектування QRS- і P-QRS-T комплексів

Валідація та дослідження параметрів точності роботи розробленого методу та апаратно-програмного комплексу можливі лише за умови порівняння роботи засобу із методом, характеристики якого добре досліджено та точність підтверджена попередніми дослідженнями. Саме тому здійснена розробка валідаційного програмного алгоритму (або референтного алгоритму) для детектування P-QRS-T. Головна вимога до алгоритму: універсальний швидкодіючий алгоритм для виявлення комплексу P-QRS-T із можливістю подальшого детектування окремих зубців та інтервалів на вхідному ЕКГ-сигналі. Для вирішення цієї проблеми розроблена модифікація програмного алгоритму детектування P-QRS-T запропонованого у роботах Pan та Tompkins [56].

На першому етапі вхідний необроблений ЕКГ-сигналів надходив до тракту попереднє оброблення та фільтрації, роботу якого описано у розділі 3.4.1. У результаті роботи алгоритму на виході було отримано сигнал MWI (рис. 3.16 *б*). Локальні максимуми згладженого MWI-сигналу відповідають положенню P-QRS-T комплексу у вхідному сигналі. На другому етапі відбувається пошук максимумів MWI-сигналу. Висновок про те, чи відповідає імпульс комплексу QRS, приймається з варіацією адаптивного порогу. Для аналізу амплітуди виходу MWI алгоритм використовує два порогові значення (THR_{SIG} і THR_{NOISE}, відповідним чином ініціалізовані під час короткого другого етапу тренувань), які постійно адаптуються до зміни якості ЕКГ-сигналу. Перший прохід використовує ці пороги для класифікації кожного ненульового зразка (*CURRENT_PEAK*) як сигналу, так і шуму:

- якщо $CURRENT\_PEAK > THR_{SIG}$, це місце ідентифікується як кандидат на комплекс QRS і рівень сигналу (SIG_{LEV}) оновлюється: $SIG_{LEV} = 0.125 \cdot CURRENT\_PEAK + 0.875 \cdot SIG_{LEV}$.

- Якщо $THR_{NOISE} < CURRENT\_PEAK < THR_{SIG}$, то це місцезнаходження ідентифікується як пік шуму і рівень шуму ($NOISE_{LEV}$) оновлюється: $NOISE_{LEV} = 0.125 \cdot CURRENT\_PEAK + 0.875 \cdot NOISE_{LEV}$.

Виходячи з нових оцінок рівнів сигналу і шуму (SIG_{LEV} *і* $NOISE_{LEV}$) в поточній точці ЕКГ, пороги регулюються наступним чином: $THR_{SIG} = NOISE_{LEV} + 0.25 \cdot (SIG_{LEV} - NOISE_{LEV})$. $THR_{NOISE} = 0.5 \cdot THR_{SIG}$.

Ці коригування поступово знижують поріг в сегментах сигналів, які вважаються більш низькими. На кожному кроці оновлення порогового значення, відбувається порівняння $CURRENT\_PEAK < THR_{SIG}$, якщо ця нерівність правдива, то пік не вважається результатом комплексу QRS. Якщо минув невиправдано тривалий період без порогового піку, алгоритм буде вважати, що QRS було пропущено, і виконано пошук назад. Мінімальний час, який використовують для запуску пошуку, – це час, у 1.66 рази більший від поточного значення RR-інтервалу. Крім того, через існування фізіологічного рефрактерного періоду вважається неможливим виявити комплекс QRS, якщо він лежить в межах 200 мс після виявленого комплексу. Після визначення положення максимуму QRS-комплексу та відповідно R-зубця, який має максимальну складову потужності QRS, виконувався пошук та детектування P- та T-піків за допомогою незгладженого сигналу, отриманого на виході фільтра верхніх частот. Коли QRS-комплекс виділено, визначається максимальний кут нахилу QRS-хвилі та виконується пошук хвилі після QRS на інтервалі часу, що дорівнює 200 мс (менше можливого RR інтервалу). Після знаходження хвилі та положення її локального максимуму, визначається максимальний кут нахилу хвилі. Якщо він менший, або дорівнює половині кута нахилу хвилі QRS, то максимум хвилі анотується як T-пік. Для пошуку та детектування P-піків було застосовано таку ж саму процедуру на інтервалі часу 200 мс перед QRS-

комплексом. Програмний код референтного метода детектування P-QRS-T-комплексів ЕКГ-сигналу наведено у Додатку Є.

Висновки до 3-го розділу

1. Результати моделювання структури спайкового шифратора для оптимального і точного кодування ЕКГ сигналів підтвердили адекватність запропонованого підходу до формування необхідних і достатніх передумов для побудови імпульсної штучної мережі.

2. Експериментальним шляхом встановлено, що нейронні структури з підвищеною кількістю стабільних станів спроможні генерувати більш складні патерни вихідної активності, що обґрунтувало можливість надання нейронній мережі додаткового обчислювального ресурсу підвищеної складності.

3. Схемотехнічна реалізація апаратного забезпечення засобу сприяла новому розумінню універсальної схеми реєстрації ЕКГ-сигналу (ЕКГ-фронтенду), що керується вбудованим програмним забезпеченням мікроконтролера в режимі реального-часу.

4. Показана необхідність первинної фільтрації та оброблення вхідного ЕКГ-сигналу для подальшої більш точної обробки даних мережею. Для вирішення цієї проблеми було застосовано та реалізовано апаратні аналогові та програмні цифрові засоби фільтрації вхідного сигналу.

Результати експериментальних досліджень даного розділу наведено в таких публікаціях:

[1] Y. M. Snizhko, O. O. Boiko, N. P. Botsva, D. V. Chernetchenko, and M. M. Milykh, “Methods for increasing the accuracy of recording the parameters of the cardiovascular system in double-beam photoplethysmography”, *Regulatory Mechanisms in Biosystems,* vol. 9(3), pp. 335-339, 2018.

[2] М. М. Милых, Е. М. Снежко, И. В. Тимченко, и Д. В. Чернетченко, “Реализация алгоритмов преобразования АДМ-ИКМ в системах автоматики и

передачи данных”, *Гірнича електромеханіка та автоматика,* № 2 (93), с. 72-76, 2014.

[3] Є. М. Сніжко, М. М. Мілих, М. П. Моцний, та Д. В. Чернетченко, “Мобільна система для вимірювання кольорових характеристик об’єктів”, *Електромагнітна сумісність та безпека на залізничному транспорті,* № 14, с. 102-106, 2017.

[4] Д. В. Чернетченко, М. П. Моцний, Н. П. Боцьва, О. В. Єліна та М. М. Мілих, “Автоматизована система реєстрації біоелектричних потенціалів”, *Вісник Дніпропетровського університету. Біологія, екологія*, т. 21(2), с. 70-75, 2013.

[5] T. Tkachenko, N. Botsva, T. Komendar, and D. Chernetchenko, “Autonomous hardware-software complex for biosignals registration and processing”, на *III Всеукр. наук.-практ. конф. Перспективні напрямки сучасної електроніки, інформаційних і комп’ютерних систем (MEICS – 2018)*, Дніпро, 2018, с. 95-96.

[6] A. Lavrenjuk, D. Chernetchenko, N. Botsva, and K. Pustova, “Blood pressure monitoring”, in *Proc. XIII Międzyn. nauk.-prakt. konf. Naukai Inowacja* – 2017, Przemyśl, 2017, pp. 53-55.

[7] K. Pustova, D. Chernetchenko, N. Botsva, and A. Lavrenjuk, “Non-invasive monitoring tools for automatic stress detection”, in *Proc. XIII Mezin. věd.-prakt. konf. Zprěvy VědeckéIdeje–2017*, Praha, 2017, pp. 21-23.

[8] Д. В. Чернетченко, Д. І. Золотова, Н. П. Боцьва, Е. М. Гасанов, та В. С. Олійник, “Автономний апаратно-програмний комплекс реєстрації та обробки кардіографічних сигналів”, in *Proc. XIIInt. scientific and pract. conf. Cutting-Edge Science – 2016*, Sheffield, 2016, pp.113-116.

[9] М. П. Моцний, Д. В. Чернетченко, Н. П. Боцьва, та О. В. Єліна, “Реєстрація біоелектричних потенціалів засобами комплексної автоматизованої системи”, *in Proc. X Mezin. věd.-prakt. konf. Veda a technologie: krok do budouchnosti– 2014,* Praha, 2014, pp. 102-105.

РОЗДІЛ 4
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АПРОБАЦІЯ АПАРАТНО-ПРОГРАМНОГО ЗАСОБА І МЕТОДА

4.1 Вибір та обґрунтування методики валідації даних

Даний розділ присвячено експериментальному дослідженню та апробації референтного метода та апаратно-програмного засобу (АПЗ). Розглянуто питання оцінювання ефективності запропонованого SNN апарату для виявлення P-QRS-T комплексів ЕКГ та детектування P-, R-, T-, Q-, S-піків в реальному часі та порівняння отриманих результатів з референтним опорним алгоритмом, що працює за принципом кінцевого автомата. При оцінюванні точності розробленої системи для кожної пари детектованих піків запропонованим референтним методом обчислюється похибка визначення точної часової мітки зареєстрованого піка, визначається параметр середнього проценту похибки відхилення (MAPE) детектованих піків. Параметр MAPE є різницею між фактичною міткою часу a_i певної ознаки сигналу (P, Q, R, S, T) і розрахунковою міткою часу p_i, що була отримана за допомогою референтного методу,

$$MAPE = \frac{1}{N}\sum_{i=0}^{N-1}\left(\frac{|a_i - p_i|}{a_i}\right)\cdot 100 \qquad (4.1)$$

де N – загальне число P-QRS-T комплексів у даному записі ЕКГ.

Розрахунок помилок визначення піків для розробленого та референтного методів проводився відносно заздалегідь достовірно анотованих ЕКГ-сигналів. Було проведено оцінку виникнення похибок першого (ложно-позитивних) та другого рода (ложно-негативних) за допомогою розрахунку відсоткового співвідношення ложно-позитивних та ложно-негативних похибок до загальної кількості похибок.

Оцінка середньої похибки SNN порівнюється з результатами роботи детермінованого алгоритму, основаного на модифікованому алгоритмі Pen та Tompkins [26], [27], який було описано в розділі 3.4.2.

4.1.1 Валідація результатів амплітудно-частотного детектування QRS-комплексу ЕКГ-сигналу з використанням SNN і референтного методу

За допомогою отриманого пристрою реєструвався аналоговий ЕКГ-сигнал, який перетворювався у послідовність спайків, що надходила на внутрішній шар нейронної мережі. Часова діаграма спайків для вхідного, внутрішнього та вихідного шарів, яка отримана з нейромодуля та відповідає даним симуляції VHDL-моделі показана на рис. 4.1.

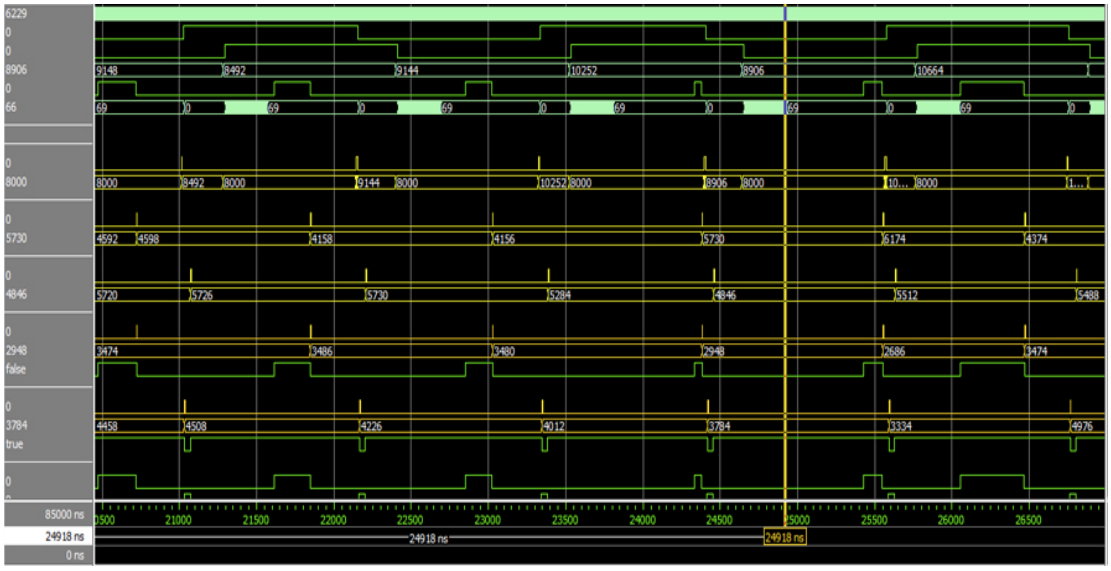


Рисунок 4.1 – Часова діаграма моделювання VHDL-структури

Для кожного валідаційного експерименту проводилось по дві серії експериментів, перша включала в себе перевірку роботи пристрою на готових записах з відкритої бази данних MIT-BIH. Вибірка записів з бази даних фізіосигналів включала в себе ЕКГ-записи тривалістю не менше 300 секунд, з різних пацієнтів без наявних патологій, та вираженими патологіями: тахікардія, брадікардія, аритмії, атріальні фібріляції, тощо. При цьому дані в цифровому форматі одразу надходили до нейромодуля за допомогою мікроконтролерного інтерфейсу UART. Друга серія експериментів виконувалась безпосередньо на волонтерах, при безперервній реєстрації ЕКГ-сигналу тривалістю 300 секунд для різних станів: у стані спокою у положенні стоячи, у положенні сидячи, у положенні лежачи, після фізичного навантаження та під час дії стрессових умов

–розв’язання складних математичних задач. Приклад часової діаграми співвідношення вихідної генерації мережі нейромодуля із часом та вхідного ЕКГ-сигналу на вході нейронального шару, показано на рис. 4.2.

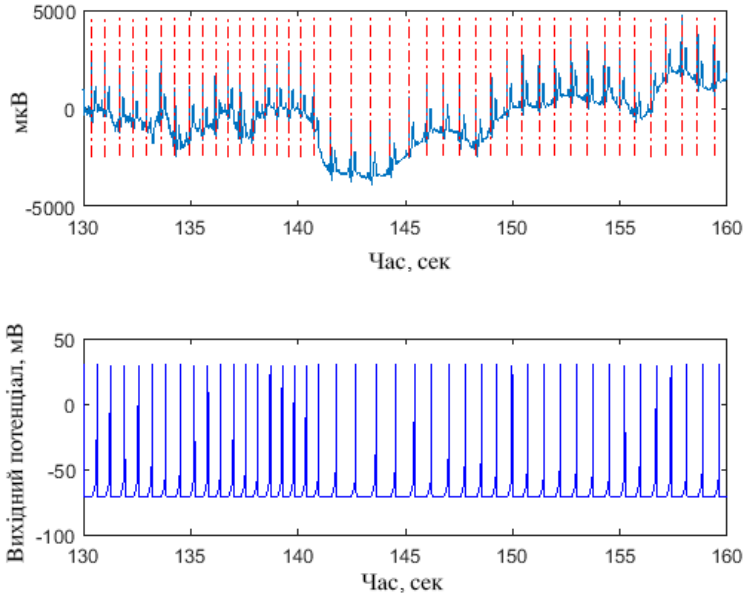


Рисунок 4.2 – Залежність імпульсів просторово-часового кодування нейрону вихідного шару (нижній графік) від вхідного ЕКГ-сигналу (верхній графік). Вісь ординат – амплітуда сигналу, мкВ (верхній графік), мВ (нижній графік). Вісь абсцис – час, с

Збільшення концентрації вхідних спайків на дендритній структурі нейрона вихідного шару призводить до генерації вихідного спайку, що пояснюється наявністю максимуму P-QRS-T комплексу в певний момент часу у вхідному сигналі.

На рис. 4.3 показана вихідна спайкова послідовність у співставленні із синхронізованим фрагментом вхідного ЕКГ-сигналу, який подається на вхід SNN і кодується до спайкового формату. З графіку залежності видно, що вихідні спайки досить чітко корелюють з часовими мітками QRS-комплексів вхідного ЕКГ-сигналу.

Розвиток нейронної активності SNN у часі показано на рис. 4.3, *а* для тривалості початкової фази навчання мережі $T_{\text{і}} = 10$ с, та на рис. 4.3, *б* для початкової фази навчання $T_{\text{і}} = 30$ с.

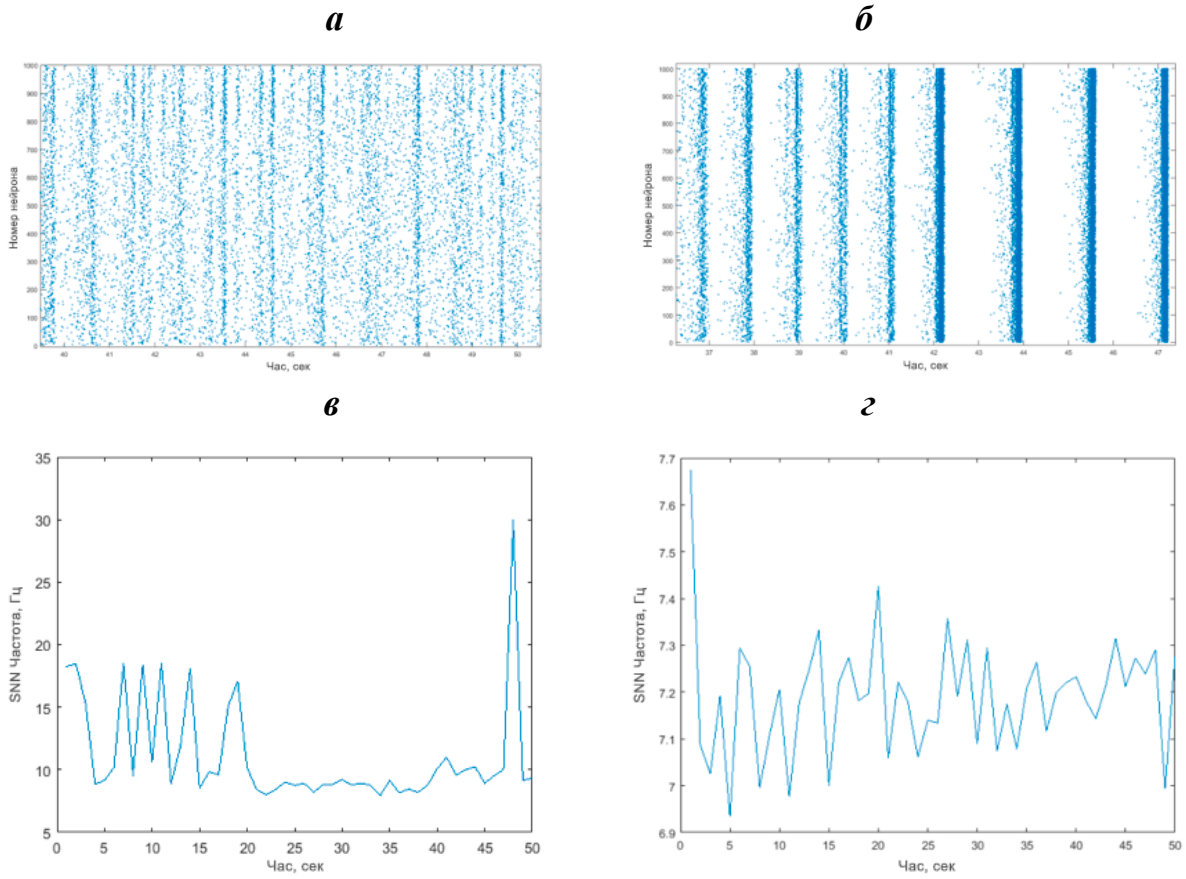


Рисунок 4.3 – Розвиток вихідної активності нейронів внутрішнього шару після фази початкового тренування мережі T_i та відповідна середня частота генерації спайків мережі із часом: *а, в* – тривалість навчальної фази 10 с; *б, г* – тривалість навчальної фази 30 с; Вісь абсцис: *а, б* – номер нейрона внутрішнього нейронного шару; *в, г* – частота генерації вихідних спайків, Гц. Вісь ординат – час, с.

Додатково можна проаналізувати навчальний процес мережі за допомогою частоти генерації спайків у мережі. Рис. 4.3, *в, г* ілюструють частоту генерації всіх спайків мережі з часом для випадків рис. 4.3, *а*, *б* відповідно. Отриманий результат легко пояснити: коли мережеві нейрони тренуються до генерації спайків після специфічного моменту, що виявляється у вхідному сигналі (QRS-комплекс) - основна частота генерації спайків в мережі є близькою до частоти ознаки у вхідного сигналу, що класифікується. Цей процес успішно спостерігається на рис. 4.3, *в*, *г*. Також встановлено, що час попереднього

тренування повинен бути не менший за $T_i = 30$ с для кожного паттерну ЕКГ-сигналу із набором індивідуальних властивостей.

Розглянемо приклад ЕКГ-запису тривалістю 5 хвилин, при якому сигнал безперервно реєструвався з волонтера. Під час реєстрації волонтеру було запропоновано вирішувати складні математичні задачі. Діаграма часової залежності сигналів на входах та виході для вихідного нейронального шару показана на рис. 4.4, *а*.

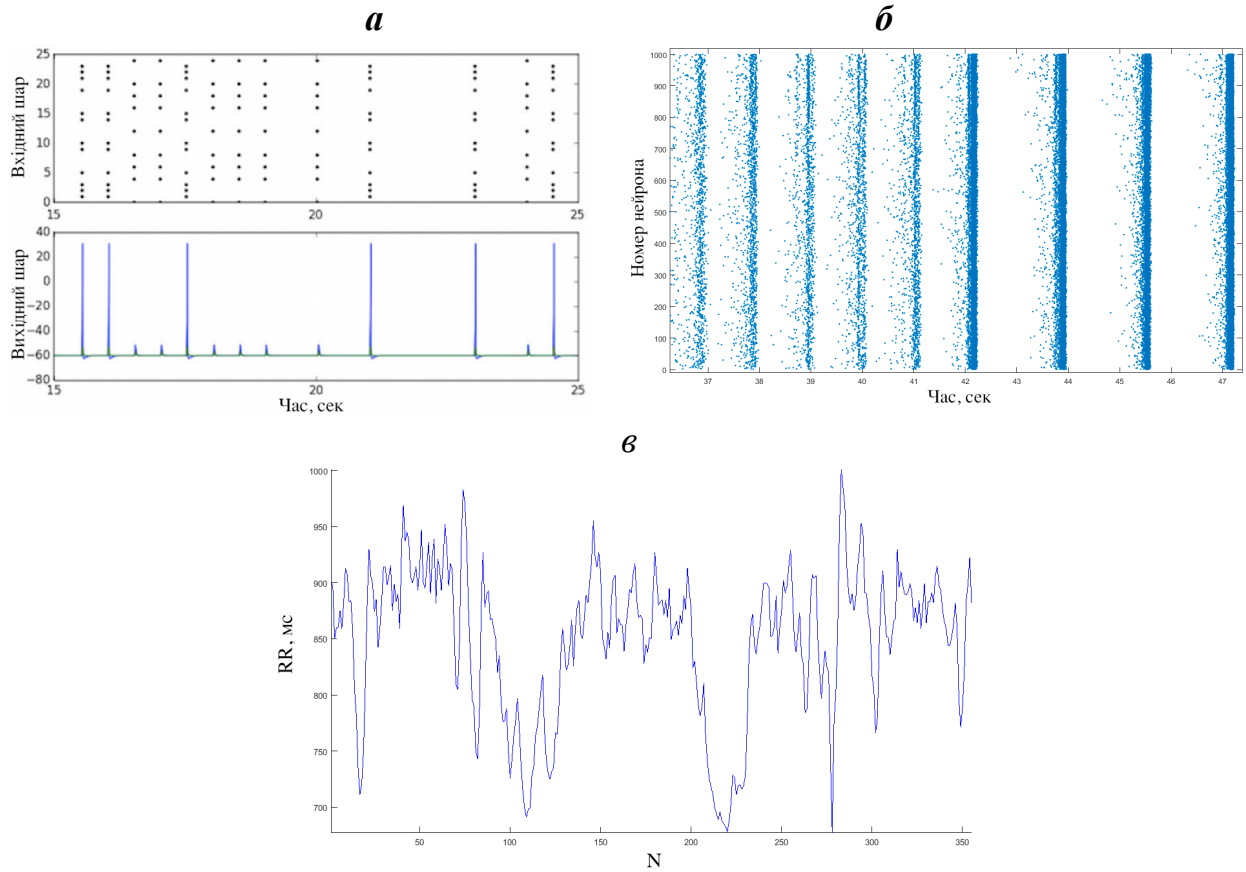


Рисунок 4.4 – Діаграма часової залежності сигналів на входах та виході: *а* – активність вихідного нейрона, зареєстрована у формі дендритних входів (верхній графік) та активність спайкового виходу (нижній графік); *б* – загальна активність нейронів SNN протягом 10 с; *в* – послідовність RR-інтервалів, які отримані з відповідної послідовності спайків на виході SNN

Необхідно додати, що наведену діаграму можна вважати випадком добре навченої мережі, результати якої можна легко пояснити. Збільшення концентрації вхідних спайків на дендритній структурі вихідного нейрона

приводить до генерації вихідного спайку, що пояснюється наявністю R-піку в певний момент часу у вхідному сигналі. Спайкова активність SNN з часом показана на рис. 4.4, *б*. Отриману послідовність RR-інтервалів на базі детектованих QRS-комплексів показано на рис. 4.4, *в*. Таким чином, результати роботи апаратно-програмного комплексу для виявлення QRS було отримано в трьох серіях експериментів. У першій серії доведено ефективність виявлення QRS з використанням оригінальних алгоритмів попереднього оброблення сигналу у порівнянні з детермінованим програмним алгоритмом. У другій серії була оцінена стійкість і ефективність шифратора ЕКГ-сигналів за допомогою запропонованого алгоритму. У третій серії оцінена ефективність запропонованого SNN за наявності та відсутності вхідного попереднього оброблення. Результати валідації точності детектування R-піків ЕКГ-сигналу для референтного методу наведені в таблиці 4.1. Для довільно обраних записів: найкращий випадок MAPE становить 0.32 %, а найгірший – 5.0 % відповідно.

Таблиця 4.1

Точність референтного методу

Номер запису	Всього R-піків	Позитивні похибки, кількість R-піків	Негативні похибки, кількість R-піків	MAPE, %
1	321	0	0	0.0
2	425	0	6	1.4
3	375	0	0	0.0
4	272	0	5	1.8
5	290	1	3	1.3
6	285	2	2	1.4
7	304	1	0	0.32
8	398	4	2	1.5
9	361	0	4	1.1
10	326	13	1	5.0

Проведено успішну валідацію алгоритма нейромодуля для ЕКГ-записів з табл. 4.1, тривалість яких обмежена t = 300 секундами. Звіт про точність і результати перевірки SNN для кожного запису наведено в табл. 4.2. Найкраще значення MAPE становить 0.10 %, а найгірше – 3.0 %.

Таблиця 4.2

Показники точності роботи SNN

Номер запису	Всього R-піків	Позитивні похибки, кількість R-піків	Негативні похибки, кількість R-піків	MAPE, %
1	311	0	1	0.32
2	270	1	0	0.10
3	354	2	8	3.00
4	298	1	6	2.61
5	367	1	3	1.15
6	405	1	4	2.02
7	434	2	2	1.11
8	305	2	1	0.98
9	321	1	2	0.87
10	287	1	2	1.30

На рис. 4.5 показано порівняльне співвідношення середньої точності оцінки RR-інтервалів з використанням SNN підходу та опорним алгоритмом для довільно оброних п'яти записів MIT-BIH і п'яти записів внутрішньої бази даних (intdb), які були отримані при записі з волонтерів. Як видно з рисунку, середнє значення MAPE, що використовує мережевий підхід, становить менше 2 % для всіх записів. Для бази даних MIT-BIH, MAPE змінюється між 0.1 і 2.0 %, в середньому дорівнює – 1.05 %. Для внутрішньої бази даних ЕКГ (intdb) MAPE змінюється між 0.87 і 2.02 %, при середньому MAPE – 1.4 %. MAPE для

референтного методу коливається в межах від 0.3 до 3.50 % при середньому значенні 1.9 %.

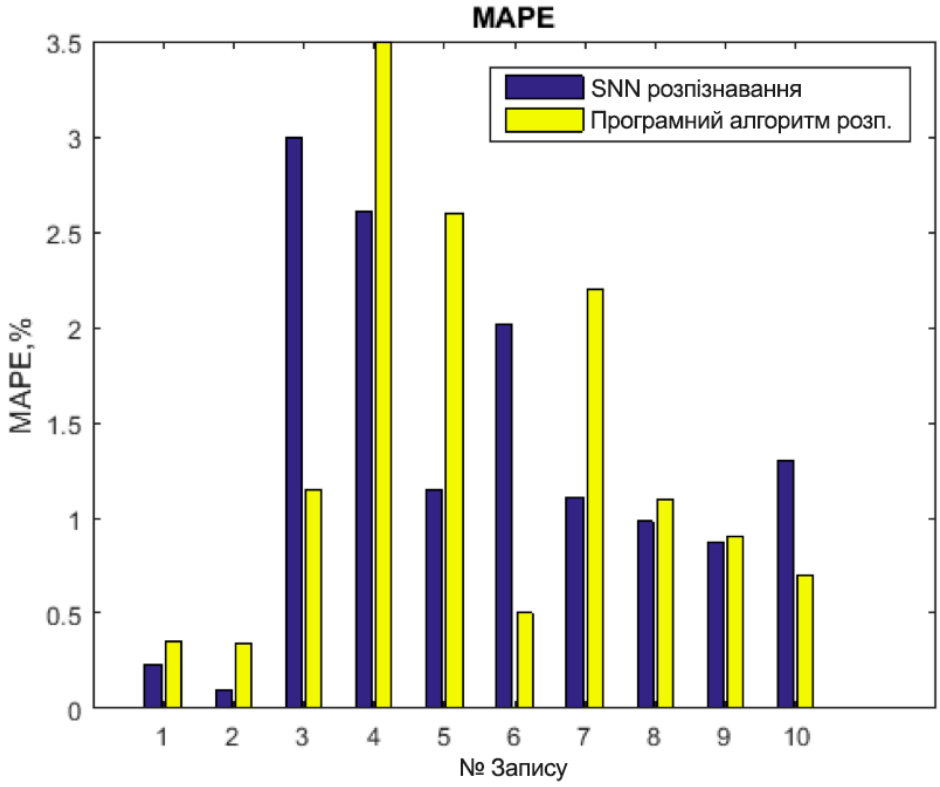


Рисунок 4.5 – Значення MAPE, розраховані для десяти різних ЕКГ записів (п'ять з яких обрані випадково з бази даних ЕКГ MIT-BIH, п'ять – з внутрішньої ЕКГ бази даних intdb): фіолетові смуги – середня похибка розпізнавання SNN після етапу навчання; жовті смуги – середня похибка детектування за допомогою референтного методу. Вісь ординат – значення MAPE, %. Вісь абсцис – номер запису ЕКГ-сигналу

Необхідно звернути увагу на те, що більш високий рівень шумів і артефактів у вхідному сигналі призводить до більшого значення MAPE як для SNN, так і для референтного методу. Водночас, слід відзначити, що для SNN середнє значення похибки виявилось меншим за похибку референтного методу, навіть для зашумленого сигналу.

Послідовність RR-інтервалів, отримана за допомогою SNN, порівнювалась з масивом інтервалів, який визначався за допомогою детермінованого чіткого алгоритмічного підходу (рис. 4.6). Середня різниця між інтервалами масиву RR

на виході SNN і відповідними RR інтервалами референтного методу не перевищувала 0.5%.

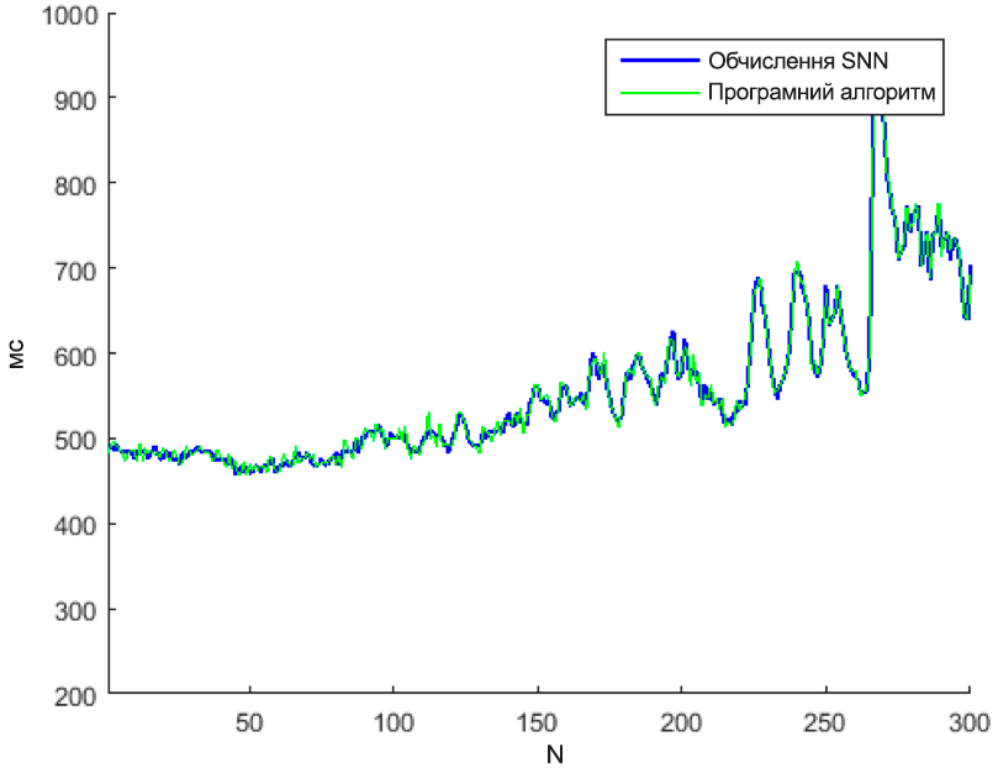


Рисунок 4.6 – Послідовність отриманих RR-інтервалів для випадково обраного запису ЕКГ тривалістю 300 с за допомогою нейро-модуля на базі SNN(синя суцільна лінія) та за допомогою референтного алгоритму (зелена суцільна лінія). Вісь ординат – розмір RR-інтервалу, мс. Вісь абсцис – номер RR-інтервалу

Порівняння показників точності оцінки середніх RR-інтервалів, отриманих для довільно обраних п'яти записів MIT-BIH і п'яти внутрішніх записів (intdb) за наявності та відсутності стадії попереднього оброблення та фільтрації первинних даних із застосуванням підходу SNN та детермінованим референтним методом (рис. 4.7), свідчить, що MAPE при використанні запропонованого здобувачем підходу становить менше 2 % для всіх записів. Значення MAPE для внутрішньої бази даних варіюється від 0.5 до 2.1 % та в середньому становить 1.2 %. Для бази даних MIT-BIH – від 0.53 до 5.3 %, при середньому значенні 2.32 %. Без стадії фільтрації роздільна здатність SNN критично знижується, а MAPE змінюється

між 2 і 12.2 % при середньому 5 %. Більш високий рівень шумів і артефактів у вхідному сигналі приводить до більш суттєвого зростання MAPE для SNN.

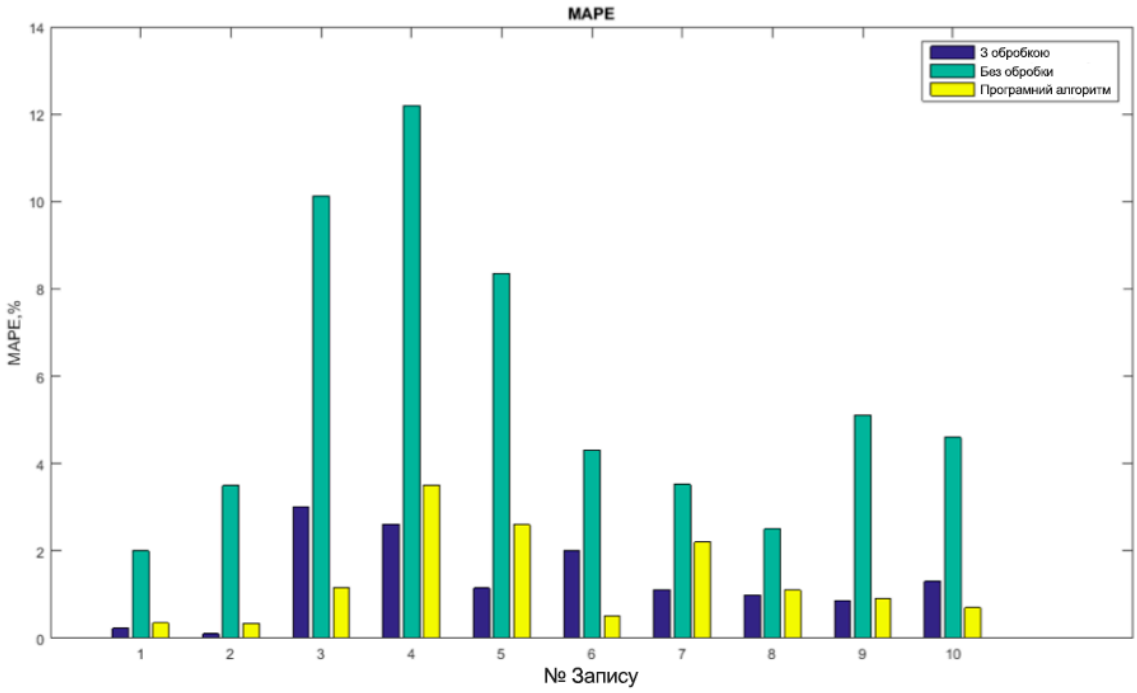


Рисунок 4.7 – Значення MAPE, розраховані для трьох основних випадків: для SNN після етапу навчання і без застосування етапу фільтрації (фіолетові смуги); SNN з етапом попереднього оброблення вхідних даних (зелені смуги); детермінований референтний алгоритм (жовті смуги). Вісь ординат – значення MAPE, %. Вісь абсцис – номер запису ЕКГ-сигналу

4.1.2 Валідація результатів класифікації головних параметрів P-QRS-T-комплексу ЕКГ-сигналу

За допомогою системи, яку описано в розділі 2, проведено валідаційне оцінювання показників точності детектування основних позитивно-амплітудних ЕКГ-піків (P, R, T-піків).

Перш за все, постійна часу і рефрактерний період внутрішніх нейронів SNN були налаштовані відносно просторово-часових параметрів вхідного сигналу. Для виявлення основних параметрів вхідного сигналу розраховано усереднений ЕКГ-паттерн, заснований на інформації про 328 комплексів P-QRS-T, які виявлено з запису ЕКГ довжиною 300 с (рис. 4.8).

Просторово-часові параметри карти ЕКГ дають важливу інформацію про мінімальний рефрактерний період мережевих нейронів, постійну часу синапсів і максимальне значення затримки передачі даних.

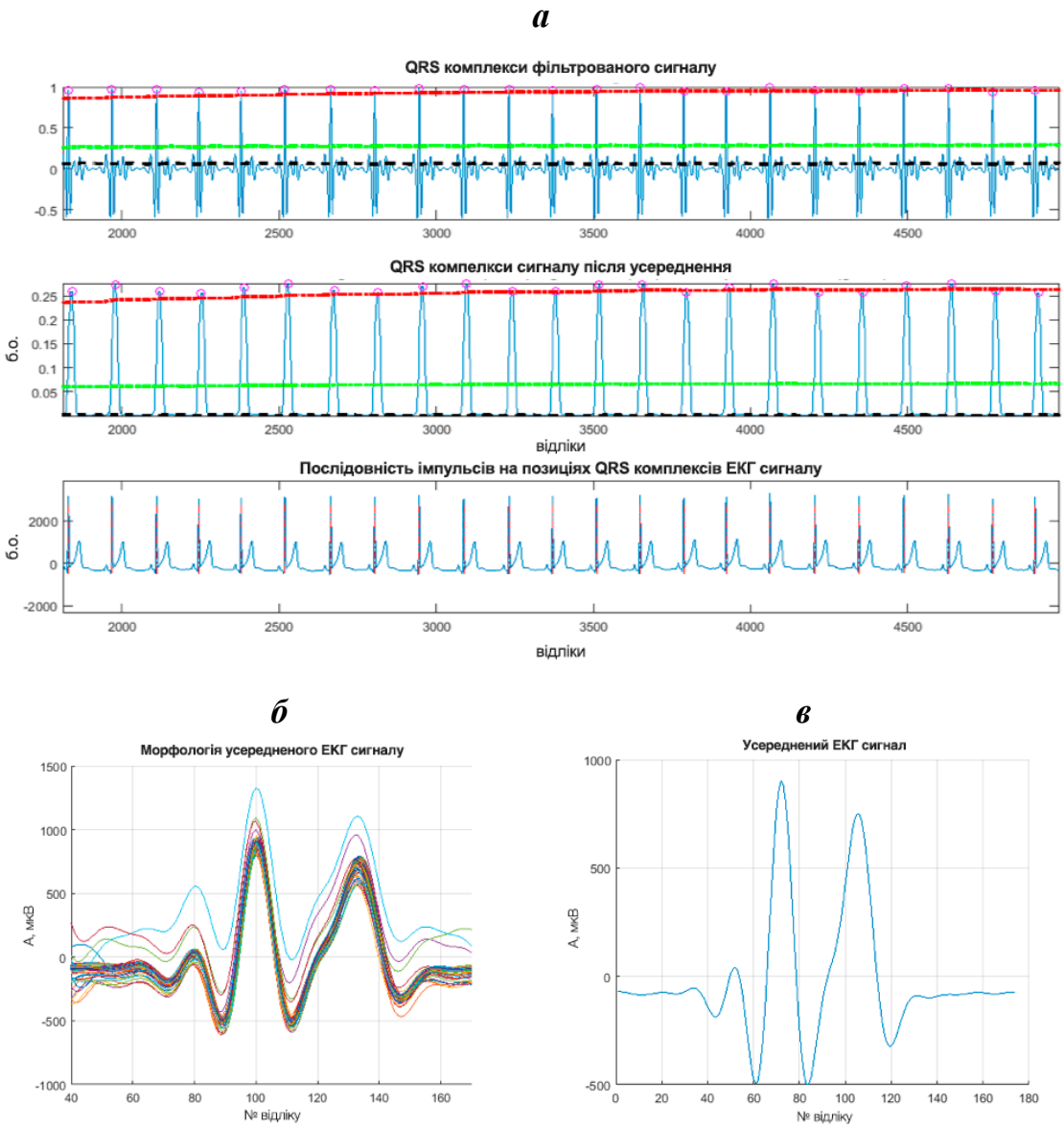


Рисунок 4.8 – Етапи розрахунку усередненого ЕКГ-паттерну: *а* – етапи оброблення ЕКГ-сигналу (зверху вниз): необроблені дані, квадрат і згладжений сигнал з виявленими позиціями QRS, виявлена позиція QRS на вхідному сигналі; *б* – ансамблі P-QRS-T комплексів; *в* – загальний вигляд усередненого ансамблю P-QRS-T-комплексу ЕКГ-сигналу

Найбільш значуща частотна складова для Т-піків дорівнює $f = 1.47$ Гц; для Р-піків $f = 3.67$ Гц і для R-піків $f = 6.6$ Гц. Постійна часу системи була обрана до τ = 100 мс відносно сигнальної складової з максимальною частотою (R-пік);

рефрактерний період генерації спайків дорівнює 10 мс; час затримки розповсюдження попереднього синаптичного сигналу дорівнює 1 мс.

4.1.3 Валідація результатів детектування P-QRS-T-комплексу ЕКГ-сигналу за допомогою SNN

Приклад залежності вхідних та вихідних генерацій із часом для вихідного нейронального шару показано на рис. 4.9. На рисунку вихідні нейронні пачки спайків показано у порівнянні з синхронізованим зразком вхідного сигналу ЕКГ. З часових діаграм видно, що вихідні спайки достатньо сильно корелюють з відповідними піками ЕКГ (сині – P-піки; червоний – R-піки; зелений – T-піки). Для позначення цього явища пікові моменти були виділені на верхньому графіку (рис. 4.9) як пунктирні вертикальні лінії відповідних кольорів (синій – P-піки; червоний – R-піки; зелений – T-піки).

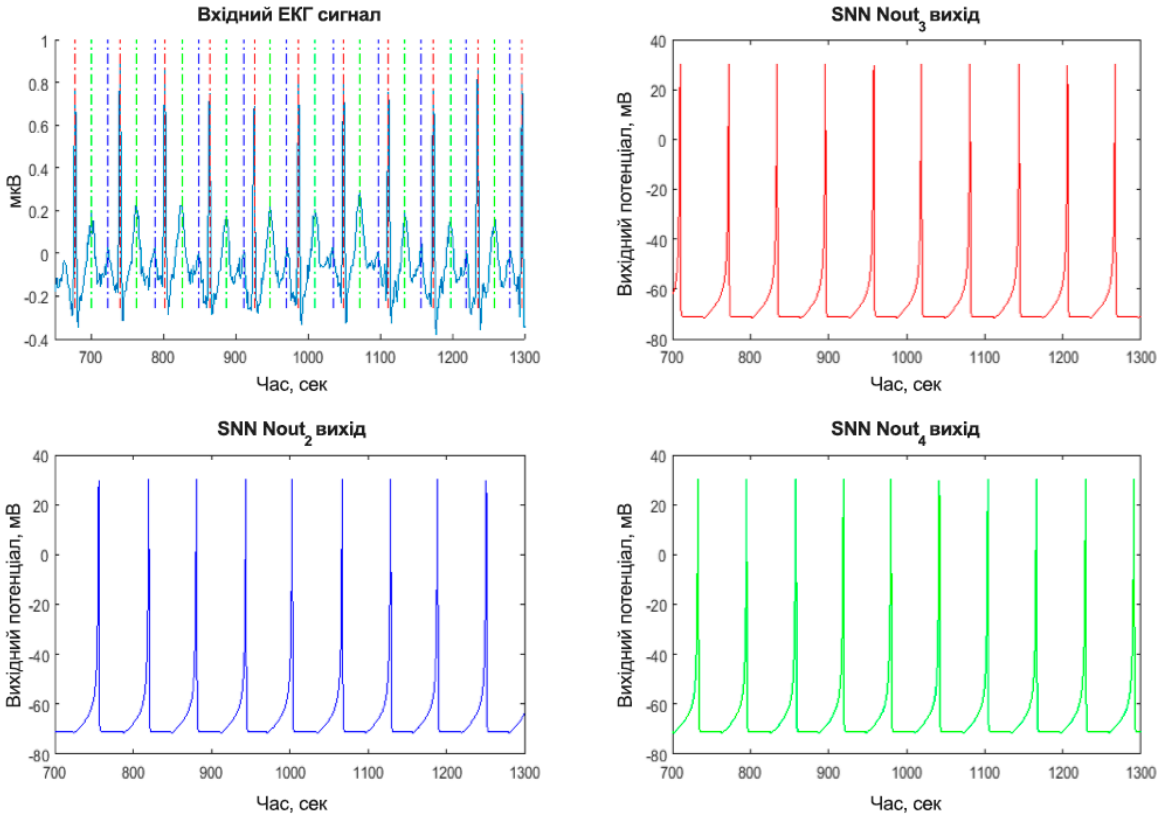


Рисунок 4.9 – Залежність вихідного просторово-часового коду від вхідного ЕКГ-сигналу. Вісь ординат: амплітуда сигналу мВ (вихідні спайкові відповіді), мкВ (вхідний сигнал). Вісь абсцис – час, с

Часова залежність нейронної активності SNN показана на рис. 4.10, *а*. Крім того, процес навчання мережі може бути проаналізований за допомогою загальної частоти генерації спайків у мережі. Рис. 4.10, *б* ілюструє зміну середньої частоти генерації рекурсивно пов'язаних нейронів з часом. Коли мережеві нейрони успішно навчаються до генерації спайків після появи специфічного паттерну у вхідному сигналі, загальна частота “стрільби” наближається до частоти піків вхідного сигналу.

а ***б***

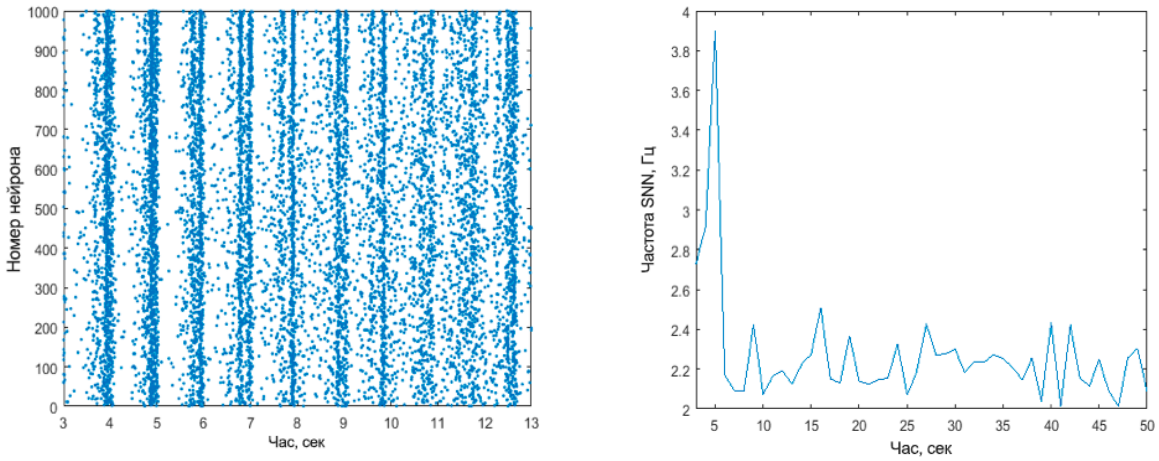


Рисунок 4.10 – Розвиток вихідної активності нейронів внутрішнього шару після фази початкового тренування мережі 30 с: *а* – спайкова активність нейронів мережі; *б* – середня частота генерації спайків. Вісь ординат: *а* – номер нейрона; б – частота генерації, Гц. Вісь абсцис – час, с

Алгоритм успішно випробуваний на десяти різних записах ЕКГ, тривалість яких скоротилася на 300 с. Звіт про точність і результати валідації для кожного запису наведено у таблиці 4.3. Діаграма (рис. 4.11) ілюструє результати точності класифікації P, R і T-піків з використанням підходу SNN для п'яти записів MIT-BIH і п'яти внутрішніх записів бази даних (intdb) у порівнянні з детермінованим референтним методом.

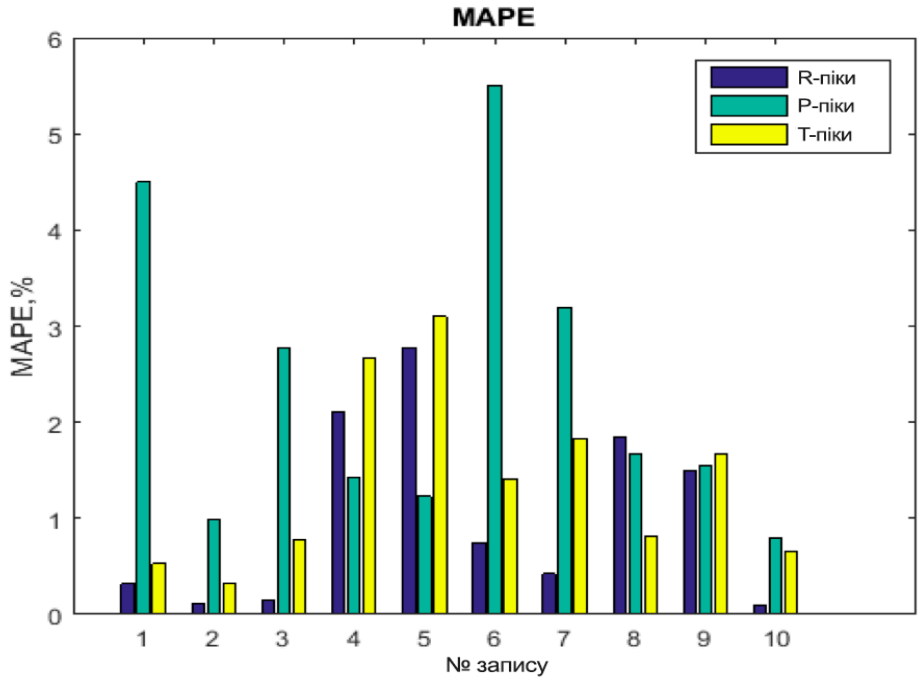


Рисунок 4.11 – Значення MAPE, розраховані для десяти різних записів (п'ять записів, вибраних випадково з бази даних ЕКГ MIT-BIH та п'ять з внутрішньої бази даних ЕКГ): при виявленні R-піків (фіолетові смуги); при виявленні P-піків (зелені смуги); при виявленні T-піків (жовті смуги). Вісь ординат – значення MAPE, %. Вісь абсцис – номер запису ЕКГ-сигналу

Таблиця 4.3

Точність визначення піків

Номер запису	Всього серцевих скорочень	MAPE R-піків, %	MAPE P-піків, %	MAPE T-піків, %
1	311	0.32	4.50	0.53
2	270	0.12	0.98	0.32
3	354	0.15	2.78	0.78
4	298	2.10	1.43	2.67
5	367	2.78	1.23	3.10
6	405	0.75	5.50	1.41
7	434	0.42	3.20	1.83
8	305	1.85	1.67	0.81

Продовж. Таблиця 4.3

Номер запису	Всього серцевих скорочень	MAPE R-піків, %	MAPE P-піків, %	MAPE T-піків, %
9	321	1.50	1.55	1.67
10	287	0.10	0.80	0.65

Отримані результати свідчать:

- для R-піків найкращий випадок MAPE – 0.10 % і найгірший – 3.10 %, середнє значення MAPE – 1.0 %;
- для P-піків найкращий випадок MAPE – 0.80 %, найгірший – 5.5 %, середнє значення MAPE – 2.36 %;
- для T-піків найкращий випадок MAPE – 0.32 %, найгірший – 2.78 % і середній MAPE – 1.37 %.

4.2 Порівняльний аналіз розробленого АПЗ та аналогів

Щільність даних і енергоспоживання визначено для шести об'єктів з бази даних (табл. 4.4). Максимальна частота таймера шифратора спайків варіювалась та набувала значень 100, 500 і 1000 Гц. Для кожної позиції рядка в таблиці зареєстровано середню швидкість генерації спайків у мережі, щільність даних (бітів на спайк) і споживання енергії. Середня швидкість генерації спайків – це загальна кількість спайків, що утворюються в SNN, усереднених для запису ЕКГ тривалістю 300 с. Щільність даних при цьому визначає ефективність роботи спайкового шифратора та обчислюється як відношення кількості бітів ($АЦП_{біт}$), необхідних для перетворення інформації аналогового ЕКГ-сигналу у цифровий формат за допомогою АЦП до загального числа спайків ($N_{спайків}$), що генеруються на виході нашого спайкового генератора та здатні шифрувати туж саму інформацію без втрат (на вході в SNN):

$$\text{Щільність даних} = \frac{\text{АЦП}_{\text{біт}}}{N_{\text{спайків}}} \tag{4.2}$$

Витрати енергії запропонованого підходу оцінюють з використанням методології оцінки загального споживання енергії для структури ПЛІС, описаної в [21], при постійних нормальних умовах навколишнього середовища (температура, тиск, вологість).

На рис. 4.12 показана частина сигналу ЕКГ і відповідні згенеровані спайкові послідовності. Процес генерації спайку з сегмента зразка ЕКГ показаного на нижньому графіку.

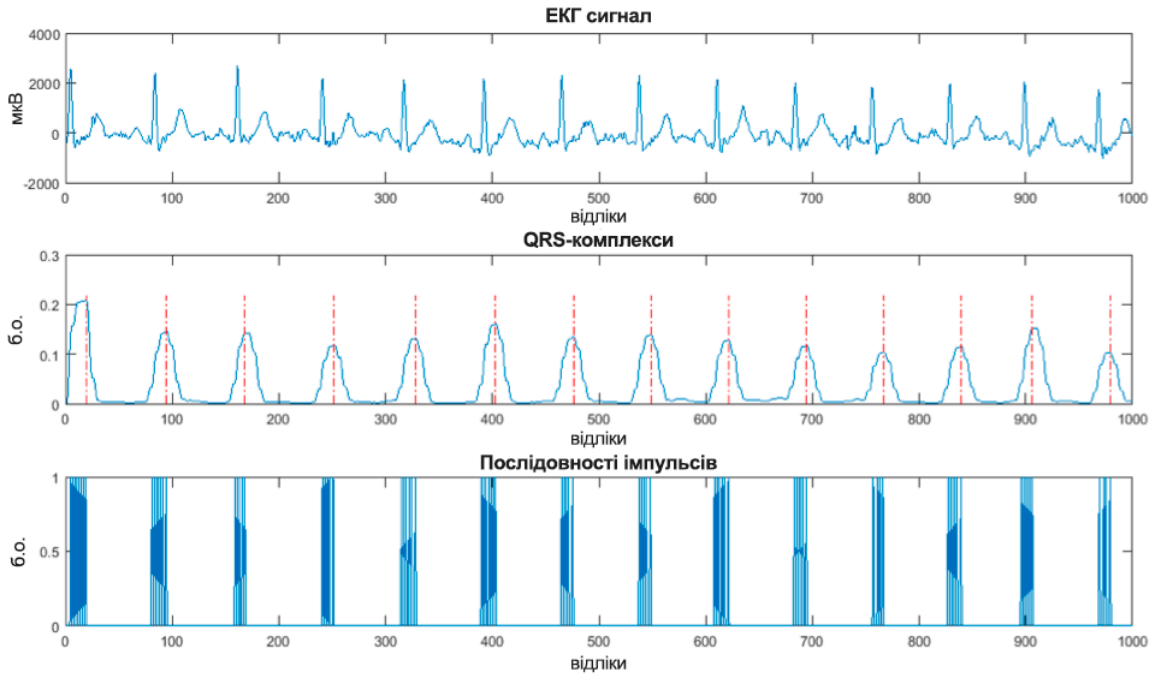


Рисунок 4.12 – Збільшена частина сигналу ЕКГ і відповідні згенеровані спайкові пачки, що кодують вхідний сигнал

Як видно з таблиці, середня щільність даних (усереднена по всіх суб'єктах) для MIT-BIH становить 52.0 при $F_{\text{CLK\_MAX}}$ = 100 Гц. Цей результат може бути інтерпретований таким чином: необхідно передати з датчика в середньому 52.0 біт даних ЕКГ для аналогово-цифрового перетворення та подальшого виявлення P-QRS-T з використанням стандартних методів QRS-детектування, тоді як запропонований спайковий шифратор для такого детектування передає лише один спайк.

Для трьох значень частот середнє споживання енергії суб'єктами становить 1.1, 1.25 і 1.40 мкВт відповідно. Таким чином, стиснення даних економить енергію і пропускну здатність у системі.

Таблиця 4.4

Загальне споживання енергії системи

№ п/п	F_{CLK_MAX}, Гц	Середня частота спайків, Гц	Щільність даних, бітів/спайк	Енергоспоживання, мкВт
1	100	5.3	53.6	1.09
2	100	5.9	50.4	1.11
3	500	7.5	37.2	1.24
4	500	7.2	40.3	1.27
5	1000	10.4	25.53	1.41
6	1000	9.8	27.3	1.39

Чим вище щільність, тим вища економія. Треба відзначити, що середнє споживання енергії з використанням за допомогою нашого підходу (усередненого за всіма суб'єктами) для записів з бази даних MIT-BIH на 1.25 мкВт нижче, ніж у роботах [25], [26], що доводить ефективність підходу, запропонованого здобувачем для використання у портативних пристроях із високою енергоефективністю.

Висновки до 4-го розділу

1. Пріоритетність та оригінальність технічних рішень нейроморфного модуля і розробленого на його основі апаратно-програмного засобу підтверджені застосуванням результатів дослідженнь у двох патентах США та експертною оцінкою комплекту конструкторської документації на виготовлення дослідного зразка засобу.

2. Попереднє оброблення і фільтрація на вході нейронної мережі обґрунтували необхідність розроблення методу оброблення ЕКГ-сигналу в SNN-

мереж, новизна якого зумовлена суттєвим зниженням щільності даних, набуттям мережею стабільних станів і зменшенням помилки розпізнавання в 4 і більше разів.

3. Апробація апаратно-програмного засобу і методики оцінювання валідації показників точності розпізнавання QRS- і P-QRS-T-патернів на реальних ЕКГ-сигналах показали такі результати: для R-зубця – 1.0 %; для P-зубця – 2.36 % і для T-зубця – 1.37 %, що є суттєво кращими ніж у діючих аналогів.

Результати експериментальних досліджень даного розділу наведено в таких публікаціях:

[1] S. M. Korogod, and D. V. Chernetchenko, “Nature of electrical tristability in a neuron model with bistable asymmetrical dendrites”, *Neurophysiology*, vol. 40, no. 5/6, pp. 412-416, 2008.

[2] Є.М. Сніжко, Д. В. Чернетченко, “Динаміка електричних потенціалів моделі мережі нейронів із нелінійними функціями активації”, *Вісник Дніпропетровського університету. Фізика. Радіоелектроніка*, т. 20, № 2(19), с. 50-57, 2012.

[3] Y. M. Snizhko, O. O. Boiko, N. P. Botsva, D. V. Chernetchenko, and M. M. Milykh, “Methods for increasing the accuracy of recording the parameters of the cardiovascular system in double-beam photoplethysmography”, *Regulatory Mechanisms in Biosystems,* vol. 9(3), pp. 335-339, 2018.

[4] Д. В. Чернетченко, М. М. Мілих, та К. В. Луданов, “Апаратна реалізація імпульсної штучної нейронної мережі для детектування параметрів електрокардіографічного сигналу (ЕКГ)”, *Вісник Хмельницького національного університету. Технічні науки*, № 4(275), с. 126-133, 2019.

[5] D. V. Chernetchenko, “A Novel Method of Preprocessing and Spike Encoding of Electrocardiographic Signal for Multi-stable Spiking Neuronal Networks Application”, *Вчені записки Таврійського національного університету ім. В.І. Вернадського, Серія технічні науки*, т. 30(69), № 3, ч. 1, с. 191-199, 2019.

ВИСНОВКИ

У дисертаційній роботі розв’язано актуальну науково-практичну задачу визначення головних клінічно-значущих ознак стану серцево-судинної системи пацієнта в режимі реального часу шляхом аналізу його ЕКГ-сигналу, зокрема детектування P-QRS-T-комплексів. Дана задача була вирішена за допомогою моделі штучної імпульсної нейронної мережі з мультистабільними нейронами, що забезпечує ефективний механізм навчання під час класифікації основних складових кардіосигналу. Усі розроблені методи та моделі лягли в основу апаратної реалізації портативного пристрою для аналізу ЕКГ-сигналів пацієнта. Основні наукові та практичні результати роботи полягають у наступному:

1. За результатами аналізу літературного контенту виявлено об’єктивні і суб’єктивні фактори, що впливають на якість оброблення ЕКГ-сигналів із застосуванням нейронних мереж та проведено оцінювання наявних апаратно-програмних засобів для аналізу ЕКГ-сигналів, результати якого покладені в системну і схемотехнічну реалізацію апаратного забезпечення розробленого засобу.

2. Комплексне застосування спайкового шифратора вхідного сигналу, рекурентних нейронів внутрішнього шару та вихідних нейронів мережі зумовило побудову імпульсної штучної нейронної мережі у вигляді системи класифікації, яка самонавчається і здатна автоматично адаптуватися в режимі реального часу до змін вхідного сигналу, що забезпечило оброблення складних клінічно-значущих випадків ЕКГ-сигналу.

3. Попереднє оброблення і фільтрація на вході нейронної мережі обґрунтували необхідність розроблення методу оброблення ЕКГ-сигналу в SNN-мереж, новизна якого зумовлена суттєвим зниженням щільності даних, набуттям мережею стабільних станів і зменшенням помилки розпізнавання в 4 і більше разів.

4. Надання властивостей адаптивності моделі імпульсного штучного нейрона досягнуто забезпеченням його електричної мультистабільності та здатності відтворювати патерни електричної активності біологічних об’єктів з

одночасним збільшенням обчислювальної потужності, що обґрунтувало використання моделі в якості базового компонента нейроморфного модуля.

5. Результати моделювання структури спайкового шифратора для оптимального і точного кодування ЕКГ-сигналів підтвердили адекватність запропонованого підходу до формування необхідних і достатніх передумов для побудови імпульсної штучної мережі.

6. Експериментальним шляхом встановлено, що нейронні структури з підвищеною кількістю стабільних станів спроможні генерувати більш складні патерни вхідної активності, що обґрунтувало можливість надання нейронній мережі додаткового обчислювального ресурсу підвищеної складності.

7. Схемотехнічна реалізація апаратного забезпечення засобу сприяла новому розумінню універсальної схеми реєстрації ЕКГ-сигналу як ЕКГ-фронтенду, що підтримується вбудованим програмним забезпеченням мікроконтролера.

8. Пріоритетність та оригінальність технічних рішень нейроморфного модуля і розробленого на його основі апаратно-програмного засобу підтверджені двома патентами США та експертною оцінкою комплекту конструкторської документації на виготовлення дослідного зразка засобу.

9. Розроблено методику і проведено валідацію показників точності розпізнавання QRS- і P-QRS-T-комплексів на реальних електрокардіограмах. Проведено валідаційні випробування та тестування отриманого приладу, з оцінки точності класифікації основних амплітудно-часових ознак ЕКГ-сигналу на вибірці записів з відкритої бази даних (MIT-BIH), а також в лабораторних умовах, на замірах з волонтерів за допомогою розробленого пристрою. В результаті яких встановлено, що середня похибка класифікації R-піків – 1.00 %; P-піків – 2.36 %; T-піків – 1.37 %, що є значно кращим показником ніж у сучасних діючих аналогів. Випробування енергоспоживання показали, що реалізоване рішення має у 1.25 менше споживання енергії, ніж у діючих сучасних рішень на базі спеціалізованих інтегральних схем (ASIC).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Статистичні доклади ВОЗ - World Health Organization, 2019 р. [Електронний ресурс] – Режим доступу до ресурсу: https://www.who.int/countries/ukr/ru/

[2] Міністерство охорони здоров'я (МОЗ) України, 2019 р. [Електронний ресурс] – Режим доступу до ресурсу: http://moz.gov.ua

[3] Скнар И.И., "Концепция построения биологически правдоподобной искусственной нейронной сети*", Нейро-нечіткі технології моделювання в економіці*, Київ: КНЕУ, Т.3., с. 188-218, 2014.

[4] О.К. Колесницький, С.М. Богатчук, М.В. Крещенецька та С.С. Яремчук "Моделювання імпульсної нейронної мережі у задачі розпізнавання багатовимірних імпульсних послідовностей", *Вісник Вінницького політехнічного інституту*. — Вінниця: УНІВЕРСУМ-Вінниця, №5, с. 62-66, 2008.

[5] О. К. Колесницький, "Аналітичний огляд апаратних реалізацій спайкових нейронних мереж", *Математичні машини і системи*, № 1, с. 3-19, 2015.

[6] Тимощук П. В. "Імпульсна нейронна мережа типу "K-WINNERS-TAKE-ALL", *Вісник Національного університету "Львівська політехніка". Серія: Автоматика, вимірювання та керування*, Львів : Видавництво Львівської політехніки, № 880, с. 41-43, 2017.

[7] Тимощук П. "Аналогова нейронна схема ідентифікації найбільших за величиною з множини сигналів з невідомого діапазону", *Вісник Національного університету "Львівська політехніка". Серія: Комп'ютерні науки та інформаційні технології : збірник наукових праць*, № 826, с. 3-8, 2015.

[8] S. Shatnyi and P Tymoshchuk, "Hardware implementation of discrete-time neural circuit of largest/smallest signal identification", in *Proc. of the XIIIth Int. Conf. "The Expirence of Designing and Application of CAD Systems in Microelectronics"*, L'viv-Polyana, Ukraine, pp. 226-230, 2015.

[9] P. Tymoshchuk and S. Shatnyi, "A hardware implementation of neural circuit of maximal/minimal value discrete-time signal identification", *Вісник*

Національного університету "Львівська політехніка". Серія: Комп'ютерні системи проектування теорія і практика: збірник наукових праць, №828. с. 27-34, 2015.

[10] В.Ф. Бардаченко, О.К. Колесницький та С.А.Василецький, "Перспективи застосування імпульсних нейронних мереж з таймерним представленням інформації для розпізнавання динамічних образів", *Управляющие системы и машины*, №6, с. 73-82, 2003.

[11] О. К. Колесницкий та Самра Муавия Хамо, "Метод распознавания многомерных временных рядов при помощи импульсных нейронных сетей", *Інформаційні технології та комп'ютерна інженерія*, №2(6), с. 86-93, 2006.

[12] А. И. Галушкин А.И., "Нейрокомпьютеры: учебн. пособ. для вузов" – М.: ИПРЖР, 2000. – Кн. 3. – 528 с.

[13] Бардаченко В.Ф., Колесницький О.К. та Василецький С.А., "Таймерні нейронні елементи та структури", Вінниця: УНІВЕРСУМ-Вінниця, с. 126, 2005.

[14] Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning [Електронний ресурс] – Режим доступу до ресурсу: http://bigdata-madesimple.com/machine-learning-explained-understanding-supervisedunsupervised-and-reinforcement-learning/.

[15] Cybenko G. "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals and Systems*, p. 117, 1989.

[16] С. И. Лукаш, О. К. Колесницкий и И. Д. Войтович "Техника и технология анализа объектов для экологической и медицинской диагностики по запаху", *Комп'ютерні засоби, мережі та системи*, №5, с.141-148, 2006.

[17] Maass W. "Networks of spiking neurons: the third generation of neural network models", *Neural Networks*, N 10., pp. 1659-1671, 1997.

[18] Wilson, P., Metcalfe, B., Graham-Harper-Cater, J., and Bailey, J. A., "A reconfigurable architecture for real-time digital simulation of neurons" in *Intelligent Systems Conference*, 2017.

[19] Kappel D., Nessler B., and Maass W. “STDP Installs in Winner-Take-All Circuits an Online Approximation to Hidden Markov Model Learning” *PLoS Comput. Biol*., Vol. 10., № 3, 2014.

[20] R.R. Borges et.al., “Effects of the spike timing- dependent plasticity on the synchronization in a random Hodgkin-Huxley neuronal network”, *Communications in Nonlinear Science and Numerical Simulation,* Elsevier B.V., pp. 12-22, 2015.

[21] J. A. Henderson, T. A. Gibson, and J. Wiles. *Spike Event Based Learning in Neural Networks*, 2015.

[22] Bishop C. “Neural Networks for Pattern Recognition”, 1995.

[23] Ayodele T. “Types of Machine Learning Algorithms”, *New Advances in Machine Learning*, 2010.

[24] S. M. Korogod, and D. V. Chernetchenko, “Nature of electrical tristability in a neuron model with bistable asymmetrical dendrites”, *Neurophysiology*, vol. 40, no. 5/6, pp. 412-416, 2008.

[25] Є. М. Сніжко, та Д. В. Чернетченко, “Динаміка електричних потенціалів моделі мережі нейронів із нелінійними функціями активації”, *Вісник Дніпропетровського університету. Фізика. Радіоелектроніка*, т.20, № 2(19), с. 50-57, 2012.

[26] Diehl P.U., Neil D., Binas, J., Cook M., Liu S.C., and Pfeiffer M. “Fast-Classifying, High-Accuracy Spiking Deep Networks Through Weight and Threshold Balancing”, *Proc. of IEEE International Joint Conference on Neural Networks (IJCNN)*, 2015.

[27] E. M. Izhikevich, “Simple model of spiking neurons”, *IEEE Transactions on neural networks,* 14 (6), 1569-1572, 2003.

[28] Y. Loewenstein, S. Mahon, P. Chadderton, K. Kitamura, H. Sompolinsky, and Y. Yarom, M. Häusser, “Bistability of cerebellar Purkinje cells modulated by sensory stimulation”, *Nat Neurosci.*, 8, No.2, pp. 202-211, 2005.

[29] M. Mynlieff, and K. G. Beam, “Characterization of voltage-dependent calcium currents in mouse motoneurons”, *J. Neurophysiol.*, 68, No. 1, pp. 85-92, 1992.

[30] P.C. Schwindt, and W. E. Crill, “Amplification of synaptic current by persistent sodium conductance in apical dendrite of neocortical neurons”, *J. Neurophysiol.*, 74, No. 5, pp. 2220-2224, 1995.

[31] S.M. Korogod, I.B. Kulagina, J. Durand, and S. Tyc-Dumont, “Electrical multistability of the dendritic arborization receiving tonic NMDA activation: a simulation study on reconstructed abducens motoneuron,” *3rd Forum of European Neuroscience, Paris, Palais des Congrus, July 13-17, FENS Abstr*, V.1, A058.9, 2002.

[32] Smon Haykin, Neural Network: A Comprehensive Foundation, 2nd Ed, Pearson Publication, 2005.

[33] S.M. Korogod, and I.B. Kulagina, “Electrical bistability in a neuron model with monostable dendritic and axosomatic membrane,” *Neirofiziologija/ Neurophysiology*, 32, No. 2, pp. 98-101, 2000.

[34] M. Hines, “NEURON – a program for simulation of nerve equations”, *in: Neural Systems: Analysis and Modeling, F. Eeckman (ed.), Kluwer Acad. Publ., Norwell MA*, pp. 127-136, 1993.

[35] L. Brodin, H. G. C. Traven, and A. Lansner, “Computer simulation of N-methyl-D-aspartate (NMDA) receptor-induced membrane properties in a neuron model”, J. *Neurophysiol.*, 66, pp. 473-484, 1991.

[36] Destexhe, A. Babloyantz, A., and Sejnowski, T. J. “Ionic mechanisms for intrinsic slow oscillations in thalamic relay neurons”, *Biophys. J.* 65, pp. 1538-1552, 1993.

[37] Rinzel, J., and Ermentrout, B. “Methods in Neuronal Modelling: From Synapses to Networks”, 1998.

[38] Victoria Booth, John Rinzel, and Ole Kiehn “Compartmental Model of Vertebrate Motoneurons for Ca2/-Dependent Spiking and Plateau Potentials Under Pharmacological Treatment”, *J Neurophysiol*, 78, pp. 3371-3385, 1997.

[39] Rinzel J., and Ermentrout B. “Methods in Neuronal Modelling: From Synapses to Networks”, *Cambridge: MIT Press*, pp. 40-62, 1998.

[40] Booth V., Rinzel J., and Kiehn O. “Compartmental Model of Vertebrate Motoneurons for Ca2+-Dependent Spiking and Plateau Potentials Under Pharmacological Treatment”, *J. Neurophysiology*, V.78, pp. 3371-3385, 1998.

[41] Alaburda A., Alaburda M., Baginskas A., Gutman A., and Svirskis, G., “Criteria of bistability of the cylindrical dendrite with variable negative slope of N-shaped current-voltage (I-V) membrane characteristic” *Biophysics-USSR*, V.46, pp. 337-340, 2001.

[42] Brodin L., Traven H. G. C., and Lansner A., “Computer simulation of N-methyl-D -aspartate (NMDA) receptor-induced membrane properties in a neuron model” *J. Neurophysiol*, 66(2), pp. 473-484, 1991.

[43] Svirskis G., Svirskene N., and Gutman A., “The theoretical analysis of multistability of pyramidal neurons”, *Biofizika*, pp. 87-91, 1998.

[44] Bailey J., Wilson P. R., Brown A. D., and Chad J., “Behavioural Simulation of Biological Neuron Systems using VHDL and VHDL-AMS”, in *IEEE Behavioural Modeling and Simulation*, San Jose, USA. pp. 153-158, 2007.

[45] Bailey, J., Wilson, P. R., Brown, A. D. and Chad, J., “Behavioural Simulation and Synthesis of Biological Neuron Systems using VHDL”, in *IEEE Behavioural Modeling and Simulation*, Sep 2008, San Jose, USA. pp. 7-12, 2008.

[46] G. W. Gross, “The Use of Neuronal Networks of Multielectrode Array as Biosensors”, *Biosens. Bioelectron.*, V.10, pp. 553-567, 1995.

[47] E. T. Claverol, “An event-driven approach to biologically realistic simulation of neural aggregates”, *Doctor of Philosophy, Electronics & Computer Science Dept., University of Southampton, UK, Southampton*, 2000.

[48] E.T. Claverol, “Discrete simulation of large aggregates of neurons”, *Neurocomputing*, V. 47, pp. 277-297, 2002.

[49] E.T. Claverol, “A Large-Scale Simulation of the Piriform Cortex by a Cell Automaton-Based Network Model”, in *IEEE Trans. on Biomedical Engineering*, V.49, pp. 921-934, 2002.

[50] W. Rall, “Theoretical significance of dendritic trees for neuronal input-output relations”, *Stanform*, CA: Stanford University Press., 1964.

[51] I. Segev, and R. E. Burke, “Compartmental Models of Complex Neurons”, in *Methods in Neuronal Modeling: From Ions to Networks*, *Second edition Cambridge*, Massachusetts: MIT Press, 1998.

[52] I. Segev, et al., “Modeling the electrical behavior of anatomically complex neurons using a network analysis”, 53(1), pp. 41-56, 1985.

[53] Peter Tino, “Artificial Neural Network Models”, *Springer handbook of computational intelligence*, V.8, I.3, pp. 456- 457, 1997.

[54] John D. Zakis and Brian J. Lithgow, “Neuron Modeling using VHDL”, 1999.

[55] Xilinx ISE. [Електронний ресурс] – Режим доступу до ресурса: https://www.xilinx.com/products/design-tools/ise-design-suite.html.

[56] JIAPU Pan, and Wills J.Tompkins, “A real Time QRS Detection Algorithm” *IEEE Transactions on biomedical Engineering*, 32(3), 1985.

[57] W. Sandham, D. Hamilton, P. Laguna, and M. Cohen “Advances in Electrocardiogram Signal Processing and Analysis”, *Hindawi Publishing Corporation EURASIP Journal on Advances in Signal Processing Volume*, 2007.

[58] Ashish S., “Hardware Implementation of Real Time ECG Analysis Algorithm”. [Електронний ресурс] – Режим доступу до ресурсу: http://scholarspace.manoa.hawaii.edu/handle/10125/20576.

[59] Chris F. Zhang and Tae-wuk, Bae “VLSI Friendly ECG QRS Complex Detector for Body Sensor Network”, V.2, No.1, 2012.

[60] F.Zhang and Y.Lin “Novel QRS Detection by CWT for ECG sensor”, in *IEEE Biomedcircuit system montrael*, Canada, 2012.

[61] S. Sumathi, and M.Y. Sanavullah, “Comparative Study of QRS Complex Detection in ECG” *International Journal of Recent Trends in Engineering*, V., No. 5, 2009.

[62] M. Millis “Advances in Electrocardiograms Clinical Applications”, 2012.

[63] Sameer K. Salih, S. A. Aljunid, Abid Yahya, and Khalid Ghailan “A Novel Approach for Detecting QRS Complex of ECG signal”, I*JCSI International Journal of Computer Science Issues*, V.9, I.6, No 3, 2012.

[64] J. Lazzaro, J. Wawrzynek, M. Mahowald, M. Sivilotti, and D. Gillespie, “Silicon auditory processors as computer peripherals*,”* in *IEEE Trans. Neural Netw.*, vol. 4, no. 3, pp. 523-528, 1993.

[65] Neha Joshi and Preet Jain “ECG Based Heart Rate Monitoring System Implementation Using FPGA For Low Power Devices And Applications” *International Journal of Research in Engineering and Technology*.

[66] Inan Güler and Elif Derya Übeyl, “ECG Beat Classifier Designed by Combined Neural Network Model”, *The Journal of Pattern Recognition Society*, 2004.

[67] Atena Sajedin, “A Trainable Neural Network Ensemble for ECG Beat Classification”, *World Academy of Science, Engineering and Technology*, No.69, 2010.

[68] GARMIN: офіційна сторінка [Електронний ресурс] – Режим доступу до ресурсу: http://garmin.com.ua.

[69] V. Varshavsky, V. Marakhovsky, and H. Saito, “CMOS Implementation of an Artificial Neuron Training on Logical Threshold Functions”, *WSEAS Transaction on Circuits and Systems*, No. 4, V. 8, p. 370-390, 2009.

[70] S. Barro, M. Fernandez-Delgado, J.A. Vila-Sobrino, C. V. Regueiro and E. Sanchez, “Classifying Multichannel ECG Patterns with an Adaptive Neural Network”, *IEEE engineering in medicine and biology*, 1998.

[71] M. Rossman, A. Buhlmeier, G. Manteuffe and K. Goser, “Short- and Long-Term Dynamics in a Stochastic Pulse Stream Neuron Implementation on FPGA”, *in Artificial Neural Networks: 6th international conference; proceedings (ICANN 96)*, 1997.

[72] H. Gholam Hosseini, D. Luo and K.J. Reynolds, “The comparison of different feed forward neural network architectures for ECG signal diagnosis”, *Medical Engineering & Physics*, 2005.

[73] I. D. Castro, C. Varon, T. Torfs, S. Van Huffel, R. Puers, and C. Van Hoof, “Evaluation of a multichannel non-contact ecg system and signal quality algorithms for sleep apnea detection and monitoring,” *Sensors*, V.18, N. 2, p. 577, 2018.

[74] M. Nappi, V. Piuri, T. Tan, and D. Zhang, “Introduction to the special section on biometric systems and applications,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, V. 44, N. 11, pp. 1457-1460, 2014.

[75] K. A. Sidek, I. Khalil, and H. F. Jelinek, “Ecg biometric with abnormal cardiac conditions in remote monitoring system,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, V. 44, N. 11, pp. 1498-1509, 2014.

[76] S. Kiranyaz, T. Ince, and M. Gabbouj, “Real-time patient-specific ecg classification by 1-d convolutional neural networks”, *IEEE Transactions on Biomedical Engineering*, V. 63, N. 3, pp. 664-675, 2016.

[77] B. Pourbabaee, M. J. Roshtkhari, and K. Khorasani, “Deep convolution neural networks and learning ecg features for screening paroxysmal atrial fibrillatio patients”, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

[78] A. Szczepanski and K. Saeed, “A mobile device system for early warning of ecg anomalies”, *Sensors*, V. 14, N. 6, pp. 11031-11044, 2014.

[79] Z. Zhao, L. Yang, D. Chen, and Y. Luo, “A human ecg identification system based on ensemble empirical mode decomposition”, *Sensors*, V. 13, N. 5, pp. 6832-6864, 2013.

[80] A. De Gaetano, S. Panunzi, F. Rinaldi, A. Risi, and M. Sciandrone, “A patient adaptable ECG beat classifier based on neural networks “, *Applied Mathematics and Computation,* pp. 243-249, 2009.

[81] Shanxiao Yang, and Guangying Yang, “ECG Pattern Recognition Based on Wavelet Transform and BP Neural Network”, *Proc. of the Sec. Inter. Symposium on Network Security*, 2010.

[82] B. Abibullaev, and H.D. Seo, “A New QRS Detection Method Using Wavelets and Artificial Neural Networks”, *Springer Science Business Media*, 2010.

[83] I. Guler, and E. D. Ubeyli, “Ecg beat classifier designed by combined neural network model”, *Pattern recognition*, V. 38, N. 2, pp. 199-208, 2005.

[84] S. Mitra, M. Mitra, and B. B. Chaudhuri, “A rough-set-based inference engine for ecg classification”, *IEEE Transactions on instrumentation and measurement*, V. 55, N. 6, pp. 2198-2206, 2006.

[85] T. Mar, S. Zaunseder, J. P. Mart´ınez, M. Llamedo, and R. Poll, "Optimization of ecg classification by means of feature selection," *IEEE transactions on Biomedical Engineering*, vol. 58, no. 8, pp. 2168-2177, 2011.

[86] W. Li, J. Li, and Q. Qin, "Set-based discriminative measure for electrocardiogram beat classification," *Sensors*, vol. 17, no. 2, p. 234, 2017.

[87] Y. Wu, R. M. Rangayyan, Y. Zhou, and S.-C. Ng, "Filtering electrocardiographic signals using an unbiased and normalized adaptive noise reduction system", *Medical Engineering & Physics*, vol. 31, no. 1, pp. 17-26, 2009.

[88] J. Yan, Y. Lu, J. Liu, X. Wu, and Y. Xu, "Self-adaptive model-based ecg denoising using features extracted by mean shift algorithm", *Biomedical Signal Processing and Control*, vol. 5, no. 2, pp. 103-113, 2010.

[89] M. Blanco-Velasco, B. Weng, and K. E. Barner, "Ecg signal denoising and baseline wander correction based on the empirical mode decomposition", *Computers in biology and medicine*, vol. 38, no. 1, pp. 1-13, 2008.

[90] V. Bhateja, S. Urooj, R. Mehrotra, R. Verma, A. Lay-Ekuakille, and V. D. Verma, "A composite wavelets and morphology approach for ecg noise filtering", in *International Conference on Pattern Recognition and Machine Intelligence. Springer*, pp. 361-366, 2013

[91] W. Jenkal, R. Latif, A. Toumanari, A. Dliou, O. El Bcharri, and F. M. Maoulainine, "An efficient algorithm of ecg signal denoising using the adaptive dual threshold filter and the discrete wavelet transform," *Biocybernetics and Biomedical Engineering*, vol. 36, no. 3, p. 499-508, 2016.

[92] S. Poungponsri and X.-H. Yu, "An adaptive filtering approach for electrocardiogram (ecg) signal noise reduction using neural networks," *Neurocomputing*, vol. 117, pp. 206-213, 2013.

[93] Y. Xu, M. Luo, T. Li, and G. Song, "Ecg signal de-noising and baseline wander correction based on ceemdan and wavelet threshold," *Sensors*, vol. 17, no. 12, pp. 2754, 2017.

[94] V. X. Afonso, W. J. Tompkins, T. Q. Nguyen, and S. Luo, "Ecg beat detection using filter banks", in *IEEE transactions on biomedical engineering*, vol. 46, no. 2, pp. 192-202, 1999.

[95] N. Zeng, Z. Wang, and H. Zhang, "Inferring nonlinear lateral flow immunoassay state-space models via an unscented kalman filter", *Science China Information Sciences*, vol. 59, no. 11, pp. 112-204, 2016.

[96] M. Javadi, R. Ebrahimpour, A. Sajedin, S. Faridi, and S. Zakernejad, "Improving ecg classification accuracy using an ensemble of neural network modules", *PLoS one*, vol. 6, no. 10, pp. 243-286, 2011.

[97] W. Liang, Y. Zhang, J. Tan, and Y. Li, "A novel approach to ecg classification based upon two-layered hmms in body sensor networks", *Sensors*, vol. 14, no. 4, pp. 5994-6011, 2014.

[98] S. Osowski, L. T. Hoai, and T. Markiewicz, "Support vector machine based expert system for reliable heartbeat recognition," *IEEE transactions on biomedical engineering*, vol. 51, no. 4, pp. 582-589, 2004.

[99] M. Barni, P. Failla, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Privacy-preserving ecg classification with branching programs and neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 2, pp. 452-468, 2011.

[100] J.-S. Wang, W.-C. Chiang, Y.-L. Hsu, and Y.-T. C. Yang, "Ecg arrhythmia classification using a probabilistic neural network with a feature reduction method", *Neurocomputing*, vol. 116, pp. 38-45, 2013.

[101] Q. Li, C. Liu, J. Oster, and G. D. Clifford, "Signal processing and feature selection preprocessing for classification in noisy healthcare data", *Machine Learning for Healthcare Technologies*, vol. 2, p. 33, 2016.

ДОДАТКИ

Додаток А

Список публікацій здобувача за темою дисертації

[1] Є.М. Сніжко, Д. В. Чернетченко, “Динаміка електричних потенціалів моделі мережі нейронів із нелінійними функціями активації”, *Вісник Дніпропетровського університету. Фізика. Радіоелектроніка*, т. 20, № 2(19), с. 50-57, 2012.

[2] М. М. Милых, Е. М. Снежко, И. В. Тимченко, и Д. В. Чернетченко, “Реализация алгоритмов преобразования АДМ-ИКМ в системах автоматики и передачи данных”, *Гірнича електромеханіка та автоматика,* № 2 (93), с. 72-76, 2014.

[3] Є. М. Сніжко, М. М. Мілих, М. П. Моцний, та Д. В. Чернетченко, “Мобільна система для вимірювання кольорових характеристик об’єктів”, *Електромагнітна сумісність та безпека на залізничному транспорті,* № 14, с. 102-106, 2017.

[4] Y. M. Snizhko, O. O. Boiko, N. P. Botsva, D. V. Chernetchenko, and M. M. Milykh, “Methods for increasing the accuracy of recording the parameters of the cardiovascular system in double-beam photoplethysmography”, *Regulatory Mechanisms in Biosystems,* vol. 9(3), pp. 335-339, 2018.

[5] Д. В. Чернетченко, М. М. Мілих, та К. В. Луданов, “Апаратна реалізація імпульсної штучної нейронної мережі для детектування параметрів електрокардіографічного сигналу (ЕКГ)”, *Вісник Хмельницького національного університету. Технічні науки*, № 4(275), с. 126-133, 2019.

[6] D. V. Chernetchenko, “A Novel Method of Preprocessing and Spike Encoding of Electrocardiographic Signal for Multi-stable Spiking Neuronal Networks Application”, *Вчені записки Таврійського національного університету ім. В.І. Вернадського, Серія технічні науки*, т. 30(69), № 3, ч. 1, с. 191-199, 2019.

[7] Д. В. Чернетченко, М. П. Моцний, Н. П. Боцьва, О. В. Єліна та М. М. Мілих, “Автоматизована система реєстрації біоелектричних потенціалів”, *Вісник Дніпропетровського університету. Біологія, екологія*, т. 21(2), с. 70-75, 2013.

[8] S. M. Korogod, and D. V. Chernetchenko, “Nature of electrical tristability in a neuron model with bistable asymmetrical dendrites”, *Neurophysiology*, vol. 40, no. 5/6, pp. 412-416, 2008.

[9] N. Botsva, I. Naishtetik, L. Khimion, and D. Chernetchenko, “Predictors of aging based on the analysis of heart rate variability”, *Pacing Clinical Electrophysiology*, vol. 40(11), pp. 1269-1278, 2017.

[10] D. V. Chernetchenko, R. S. Romaniuk, D. Sawicki, and G. Yusupova, “Analysis of electrical patterns activity in artificial multi-stable neural networks”, *Proceedings of SPIE –The International Society for Optical Engineering*, 7 p., 2019.

[11] T. Tkachenko, N. Botsva, T. Komendar, and D. Chernetchenko, “Autonomous hardware-software complex for biosignals registration and processing”, на *III Всеукр. наук.-практ. конф. Перспективні напрямки сучасної електроніки, інформаційних і комп’ютерних систем (MEICS – 2018)*, Дніпро, 2018, с. 95-96.

[12] A. Lavrenjuk, D. Chernetchenko, N. Botsva, and K. Pustova, “Blood pressure monitoring”, in *Proc. XIII Międzyn. nauk.-prakt. konf. Naukai Inowacja* – 2017, Przemyśl, 2017, pp. 53-55.

[13] K. Pustova, D. Chernetchenko, N. Botsva, and A. Lavrenjuk, “Non-invasive monitoring tools for automatic stress detection”, in *Proc. XIII Mezin. věd.-prakt. konf. ZprěvyVědeckéIdeje–2017*, Praha, 2017, pp. 21-23.

[14] Д. В. Чернетченко, Д. І. Золотова, Н. П. Боцьва, Е. М. Гасанов, та В. С. Олійник, “Автономний апаратно-програмний комплекс реєстрації та обробки кардіографічних сигналів”, in *Proc. XIIInt. scientific and pract. conf. Cutting-Edge Science – 2016*, Sheffield, 2016, pp.113-116.

[15] Д. В. Чернетченко, Е. М. Гасанов, Д. І. Золотова, та В. С. Олійник, “Апаратно-програмний комплекс реєстрації міографічних сигналів на базі мікроконтролера”, in *Proc. XII Mezin. věd.-prakt. konf. Efektivní nástroje moderníchvěd – 2016*, Praha, 2016, pp. 23-26.

[16] М. П. Моцний, Д. В. Чернетченко, Н. П. Боцьва, та О. В. Єліна, “Реєстрація біоелектричних потенціалів засобами комплексної автоматизованої

системи”, *in Proc. X Mezin. věd.-prakt. konf. Veda a technologie: krok do budouchnosti– 2014,* Praha, 2014, pp. 102-105.

[17] С. В. Тимчик, С. М. Злепко, І. О. Криворучко, та Д. В. Чернетченко, “Професійне здоров’я людини і психічна, фізіологічна і клінічна складові”, на *Міжнар. наук.-практ. конф. Сучасні наукові дослідження у психології та педагогіці – прогресс майбутнього,* Одеса, 2019, с. 45-48.

[18] Д. В. Чернетченко, Д. І. Золотова, Е. М. Гасанов, та В. С. Олійник, “Комплекс реєстрації та обробки біопотенціалів на базі мікроконтролера”, in *XII Międzynarodowej naukowo-praktycznej konferencji Perspektywiczne Opracowania Sa Nauka i technikami – 2016*, Przemysl, 2016, pp. 77-80.

[19] D. V. Chernetchenko, T. A. Botsva, “Method of registering the intervals between adjacent R-peaks of the ECG signal with the one hand in order to diagnose and assess the state of the human body and Heart Rate Variability wearable monitoring device”, *U.S. Patent and Trademark Office, US20180242858A1*, 2018.

[20] D. V. Chernetchenko D.V., T. A. Botsva, “Method and apparatus for cuff less blood pressure monitoring based on simultaneously measured ECG and PPG signals designed in wristband form for continuous wearing”, *U.S. Patent and Trademark Office, US20190059752A1*, 2019.

Додаток Б

Акти впровадження

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Дніпровського національного університету
імені Олеся Гончара, д.х.н., проф.
_____ С. І. Оковитий
«10» 04 2019 р.

АКТ
про впровадження результатів дисертаційної роботи
Чернетченка Дмитра Володимировича
на тему: "Метод та апаратно-програмний засіб обробки електрокардіографічних сигналів за допомогою штучних мультистабільних нейронних мереж"
у навчальний процес

Результати дисертаційного дослідження Д. В. Чернетченка з теми "Метод та апаратно-програмний засіб обробки електрокардіографічних сигналів за допомогою штучних мультистабільних нейронних мереж" впроваджувалися в освітній процес на кафедрі експериментальної фізики та фізики металів Дніпровського національного університету ім. Олеся Гончара впродовж 2016-2019 років. Результати дослідження були впроваджені у процес підготовки студентів напряму 172 "Телекомунікації та радіотехніка", 6.050902 "Радіоелектронні апарати", спеціальності 8.05090204 "Біотехнічні та медичні апарати і системи" та напряму 6.040203" Фізика" на факультеті фізики, електроніки та комп'ютерних систем Дніпровського національного університету ім. Олеся Гончара.

За програмою дисертанта було впроваджено нові практичні та лабораторні заняття для наступних курсів: "Програмування вбудованих систем", "Комплексна автоматизація фізичного експерименту", "Інженерна та комп'ютерна графіка", "Мікроконтролерні системи реального часу". Теоретичні та практичні результати роботи використовуються викладачами кафедри в процесі викладання зазначених курсів.

Зважаючи на актуальність і наукову новизну результатів, описаних в роботі здобувача колективом кафедри експериментальної фізики та фізики металів були зроблені висновки про доцільність подальшого впровадження результатів дослідження в навчальний процес.

Науковий керівник, доц., к.т.н. Є. М. Сніжко

Зав. каф. експериментальної
фізики та фізики металів, проф., д.ф-м.н. С. І. Рябцев

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ
«НАУКОВО-ВИРОБНИЧЕ ПІДПРИЄМСТВО «СМД»
49033, Україна, м. Дніпропетровськ, вул. Краснопільська, б.9, офіс.406
Тел./факс: (056) 763-50-88

«11» Вересня 2018 року

АКТ
про впровадження результатів наукових досліджень в практику діяльності організації

Ця довідка надана Чернетченку Дмитру Володимировичу, громадянину України, який зареєстрований за адресою: Тополя-3, буд. 20, блок 2, кв. 26, м. Дніпропетровськ, Україна, паспорт МЕ №552083 (*надалі – Виконавець*), здобувача звання кандидата технічних наук, що займає посаду асистента кафедри експериментальної фізики та фізики металів Дніпровського національного університету ім. Олеся Гончара, про те що у період з 2016 по 2017 рік він здійснював наукове консультування ТОВ "Науково-виробниче підприємство "СМД" (*надалі – Замовник*) з питань створення винаходу в рамках своєї дисертаційної тематики: ***«Пристрій для діагностики, що здійснюється на основі аналізу варіабельності ритму серця, та спосіб розшифровки, тлумачення сигналів, спосіб здійснення аналізу варіабельності ритму серця за допомогою отриманих пристроєм сигналів, спосіб передачі й відображення результатів діагностики на дисплеї мобільного пристрою»*** (*надалі – Винахід*),а саме: повного циклу розробки завершеного технічного рішення за Технічним Завданням (*надалі – Прилад*); концепції; доробки, спрямовані на покращення Винаходу; промислові зразки; відкриття; торгівельні марки та ін. В результаті співпраці за Договірними зобов'язаннями було встановлено, що за Винахідником залишається право авторства на Винахід і немайнові Об'єкти інтелектуальної власності, пов'язані із Винаходом, які охороняються безстроково.

В процесі роботи Виконавець виконав на користь Заявника наступні роботи:

1. розробка методу отримання R-R інтервалів з зап'ястя за допомогою датчику Приладу;
2. підготовка технічних рішень прототипу Приладу до серійного випуску Приладу, в тому числі, але не обмежуючись:
   - проведення робіт з розрахунку серійного зразка;
   - оптимізація технічних рішень для зниження собівартості випуску Приладу в якості серійного продукту;
3. підготовка інформації щодо вартості випуску прототипу Приладу на всіх етапах, включаючи, але не обмежуючись:
   - проведення тендерної процедури обрання підрядника/підрядників на кожний з етапів виробництва Приладу;
   - узгодження комерційних умов співробітництва з затвердженими Заявником підрядниками;
   - логістика виробничого процесу;
4. складання необхідної щодо Приладу технічної документації;
5. здійснення контролю якості Приладів, які виготовляються;
6. здійснення контролю за дотриманням всіма учасниками процесу виготовлення Приладів охорони прав інтелектуальної власності на Винахід;
7. виконання робіт, пов'язаних з процедурою сертифікації:
   - підготовка Приладів до сертифікації,

- пропонування Заявнику сертифікаційних площадок;

8. виконання інших робіт в межах кваліфікації Виконавця.

Практичну цінність мають наступні рішення запропоновані та розроблені Виконавцем:

- метод реєстрації електрокардіографічних сигналів (ЕКГ) з людини;
- апаратно-програмний комплекс, включаючи датчик Приладу, додаток для мобільного пристрою, що дозволяє реєструвати ЕКГ сигнали з зап'ястя людини та візуалізувати в неперервному режимі;
- програмні алгоритми обробки та фільтрації сирих даних;
- програмні інтелектуальні алгоритми на базі алгоритмів машинного навчання та штучних нейронних мереж, що дозволяють отримувати дані про форму ЕКГ сигналу в режимі реального часу та визначати головні ознаки сигналу (зубці, інтервали);
- математичні алгоритми обробки отриманих даних (R-R інтервалів), для подальшого статистичного аналізу та для використання методики варіабельності ритму серця (ВРС).

Результати роботи, впроваджені у виробничий процес Замовника:

- розроблено та опубліковано 2 патенти на корисні модель в закордонному патентному відомстві (США), а саме:

D.V. Chernetchenko, T. A. Botsva, Method of registering the intervals between adjacent R-peaks of the ECG signal with the one hand in order to diagnose and assess the state of the human body and Heart Rate Variability wearable monitoring device, United States Patent and Trademark Office, US20180242858A1, 2018.

D.V. Chernetchenko D.V., T. A. Botsva, Method and Apparatus for Cuff Less Blood Pressure Monitoring Based on Simultaneously Measured ECG and PPG Signals Designed in Wristband Form for Continuous Wearing, United States Patent and Trademark Office, US20190059752A1, 2019.

- розроблено та впроваджено у виробництво Приладу для реєстрації та аналізу ЕКГ з зап'ястя людини, що отримало комерційну назву "*SenceBand*";
- розроблено та підготовлено пакет конструкторської документації для виготовлення Приладу;
- виготовлена партія приладів 2200 примірників Приладу;
- виконано авторський нагляд та супутня технічна підтримка в процесі виробництва та після завершення виробництва Приладу.

ВИКОНАВЕЦЬ	**ЗАМОВНИК**
Чернетченко Дмитро Володимирович	**ТОВ «НВП «СМД»**
Фізична адреса: 49041, ж/м Тополя-3, буд. 20, корп.2, кв. 26. Паспорт: МЕ №552083, виданий Бабушкінським РВ ДМУ УМВС України в Дніпропетровській області, 29.09.2004 року Тел./факс: +380504208604	Юридична адреса: 49033, м. Дніпропетровськ, вул. Краснопільська, буд. 9, оф. 406 р/р 26007060775252 в ПАТ «КБ «ПРИВАТ БАНК» м. Дніпропетровськ, МФО 305299 ЄДРПОУ 39005006 ІПН 10000000072035 Тел./факс: 8(056)7635088
	Директор ТОВ «НВП «СМД»
_____ Д.В. Чернетченко "11" 09 2018р.	_____ В. С. Шевченко "11" 09 2018р. М.П.

ПОГОДЖЕНО
Проректор з наукової роботи
Дніпровського національного
університету імені Олеся Гончара

_____ С.І. Оковитий

«_____» _____ 2018 р.

ЗАТВЕРДЖЕНО
Проректор з науково-педагогічної
роботи Дніпровського національного
університету імені Олеся Гончара

_____ В.А. Куземко

«_____»_____ 2018 р.

Акт

впровадження результатів науково-дослідної роботи в освітній процес Дніпровського національного університету імені Олеся Гончара

ФФЕКС 56-16 № 0116U003588
«Дослідження принципів обробки інформації та управління в біомедичних та технічних системах», термін виконання: 01.01.2016 – 31.12.2018

1 «20» листопада 2018 р. вчена рада факультету фізики, електроніки та комп'ютерних систем у складі 20 з 25 осіб заслухали повідомлення д.б.н., Шугурова Олега Олеговича та к.т.н. Сніжко Євгена Матвійовича. про результати виконання робіт за темою.

2 Стисла характеристика результатів дослідження: досліджені характеристики потенціалів дорсального корінця викликаних одиночними або ритмічними серіями імпульсів, наведені залежності частоти та фази вихідних сигналів від вхідних сигналів стану об'єкту та характеристик моделі; досліджені біопотенціали рослин в умовах впливу холодової, теплової, фото- та електростимуляції, а також в умовах впливу різноманітних комбінацій вказаних факторів; проведено дослідження поглинання оптичних сигналів в червоному та інфрачервоному спектрі. Дослідження дозволили визначити концентрацію кисню у крові, зареєструвати пульсові хвилі та визначити частоту пульсу.

3 Використання у освітньому процесі: матеріали НДР використані у навчальних курсах «Біофізика», «Біомедтехніка», «Апаратура біомедичних досліджень та її експлуатація», «Сучасні технології обробки медико-біологічної інформації», «Розподілені системи збирання, аналізу та захисту даних» «Автоматизація досліджень та випробувань», «Комплексна автоматизація експериментальних досліджень», «Методи обробки багатовимірної інформації», «Програмування вбудованих систем», «Управління у біологічних та медичних системах», «Мікропроцесори в електронних апаратах», «Візуалізація біомедичних сигналів», «Методи та засоби управління інформаційними ресурсами», «Програмування вбудованих медико-біологічних систем», «Апаратні та програмні засоби електронних біомедичних систем» та у курсових і дипломних роботах, а також у навчальному посібнику «Практикум з біофізичних методів дослідження» (М.П.Моцний, О.В.Єліна, С.О.Кочубей., Д.: 2016). Матеріали НДР використані при написанні дипломнихробіт (КФ-14-1 (наук. керівник – доц. Моцний М.П.: Власенко А. Г. «Дослідження біоелектричної активності рослин при стимуляції різними за параметрами світловими імпульсами», Геворкян А.А. «Дослідження дії електромагнітного поля на електричну активність рослин», Гелєвий Т.Т. «Дослідження біопотенціалів рослин при стимуляції світлом різного спектрального складу», Зінченко Т.О. «Вплив світлової стимуляції на біологічну активність рослин», наук. керівник – Сніжко Є.М.: Цвід Р.В.

«Бездротові сенсорні мережі для фізичних досліджень на основі ОС Contiki»), КФ-16м-1 (наук. керівник – доц. Моцний М.П.: Акобян А.А. «Дослідження дії електромагнітних коливань на біопотенціали рослин» , Бессараб В.І. «Дослідження світлоіндукованих потенціалів рослин, вирощених в умовах іонодефіциту», Гаркуш О.С. «Дослідження впливу температури на параметри метаболічних потенціалів рослин», Маслак І.В. «Дослідження дії зовнішнього освітлення на електричну активність рослин», Сливець А.О. «Дослідження біопотенціалів рослин при імпульсній світловій стимуляції»), КА-16м-1 (наук. керівник – Сніжко Є.М.: «Моніторинг стану людини за допомогою двопроменевої фотоплетизмографіїї», Будяк В.С. «Система управління чотирьохопорною платформою», наук. керівник – Боцьва Н.П.: Лавренюк А.С. «Комплекс неінвазивного моніторингу артеріального тиску»).

4 Захищено (заплановано) дисертацій за результатами НДР: немає.

5 Відомості про розроблені об'єкти права інтелектуальної власності по розглянутій науково-дослідній роботі: всього по темі 36 публікацій, найвагоміші з них:

Шугуров О.О. Распределение вызванных потенциалов поперек дорсальной поверхности спинного мозга при транссекции дорсальных корешков // ScienceRise: Biological Science.– 2017.– №1(4).– С.45–50;

Мобільна система для вимірювання кольорових характеристик об'єктів / Є.М.Сніжко, М.М.Мілих, М.П.Моцний, Д.В.Чернетченко // Електромагнітна сумісність та безпека на залізничному транспорті. – Д.: Вид-во ДНУЗТ, 2017. – № 13. – С.57-60.

6 Досягнення студентів за результатами НДР: Бойко О.А. – Диплом за найкращу доповідь на XIX Міжнародній молодіжній науково-практичній конференції «Людина і космос», 2017 р. (наук. керівник доц. Сніжко Є.М.); Якименко В.О. – Диплом III ступеню Всеукраїнського конкурсу студентських наукових робіт зі спеціальності «Радіотехніка» у 2017/2018 н.р.; участь у підсумковій науково-практичній конференції ХНУРЕ, Харків, 26 квітня 2018 р. (наук. керівник Сніжко Є.М.); Паламарчук Ю.А. – Доповідь на II Всеукраїнській науково-практичній конференції «Перспективні напрямки сучасної електроніки, інформаційних і комп'ютерних систем, Дніпро 2017 р. (наук. керівник доц. Сніжко Є.М.); Скрипльов Д. Д. Доповідь на III Всеукраїнській науково-практичній конференції «Перспективні напрямки сучасної електроніки, інформаційних і комп'ютерних систем, Дніпро 2018 р. (наук. керівник ас. Чернетченко Д.В.); Ткаченко Т.О. Доповідь на III Всеукраїнській науково-практичній конференції «Перспективні напрямки сучасної електроніки, інформаційних і комп'ютерних систем, Дніпро 2018 р. (наук. керівник доц. Боцьва Н.П.).

7 Пропозиції вченої ради факультету фізики, електроніки та комп'ютерних систем: продовжити дослідження принципів та методів обробки інформації у біомедичних та технічних системах.

Заст. зав. науково-аналітичного відділу		Т.В. Безуглая
Зав. відділу з питань інтелектуальної власності		Н.Є. Голик
Зав. навчального відділу		О.В. Верба
Голова Ради		О.В. Коваленко
Зав. кафедри		С.І. Рябцев
Науковий керівник		О.О. Шугуров

Додаток В

Патенти на корисні моделі

US 20180242858A1

(19) **United States**

(12) **Patent Application Publication**

BOTSVA et al.

(10) **Pub. No.: US 2018/0242858 A1**

(43) **Pub. Date: Aug. 30, 2018**

(54) **METHOD OF REGISTERING THE INTERVALS BETWEEN ADJACENT R-PEAKS OF THE ECG SIGNAL WITH THE ONE HAND IN ORDER TO DIAGNOSE AND ASSESS THE STATE OF THE HUMAN BODY AND HEART RATE VARIABILITY WEARABLE MONITORING DEVICE**

(71) Applicants: **Tetiana BOTSVA**, Dnipro (GB); **Dmytro CHERNETCHENKO**, Dnipro (GB)

(72) Inventors: **Tetiana BOTSVA**, Dnipro (GB); **Dmytro CHERNETCHENKO**, Dnipro (GB)

(21) Appl. No.: **15/442,631**

(22) Filed: **Feb. 25, 2017**

Publication Classification

(51) **Int. Cl.**

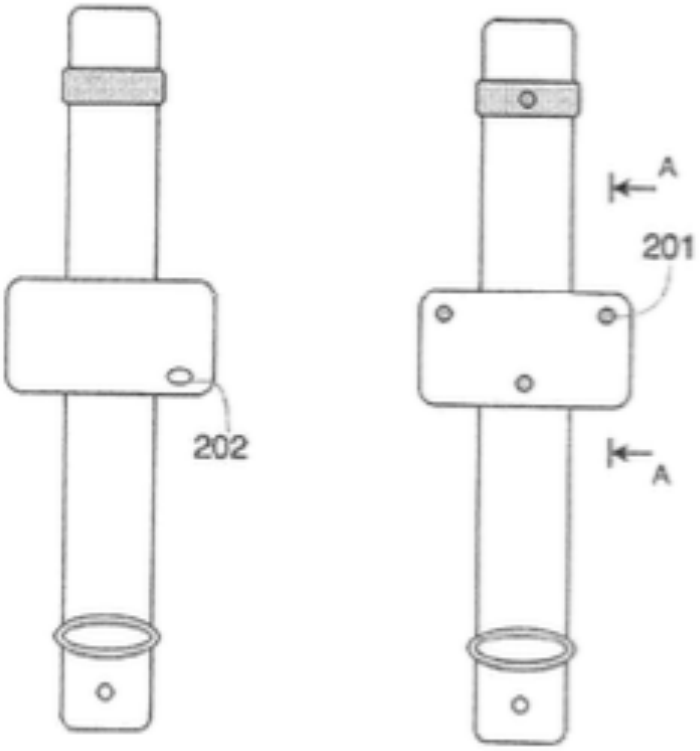
A61B 5/0245	(2006.01)
A61B 5/00	(2006.01)
A61B 5/024	(2006.01)
A61B 5/0402	(2006.01)
A61B 5/0456	(2006.01)
A61B 5/046	(2006.01)
A61B 5/0468	(2006.01)

(52) **U.S. Cl.**

CPC ***A61B 5/0245*** (2013.01); ***A61B 5/0006*** (2013.01); ***A61B 5/02405*** (2013.01); ***A61B 5/0402*** (2013.01); *A61B 5/0408* (2013.01); ***A61B 5/046*** (2013.01); ***A61B 5/0468*** (2013.01); ***A61B 5/7285*** (2013.01); *A61B 5/024* (2013.01); ***A61B 5/0456*** (2013.01)

(57) **ABSTRACT**

The invention relates to a method, system and program product to obtain R-R-intervals. The described herein wearable device is intended for recording the electrical signal caused by the work of the heart. The signal which is a potential difference is recorded from the one hand, by two dry electrodes relative "reference" electrode. This method allows to record the signal in motion. The accuracy of the R-R-intervals values calculated by using this method is sufficient for HRV analysis. The method comprises the following steps: registration of the ECG signal, amplification, filtering, AD-conversion, R-peak detection and the calculation of R-R-intervals. The resulting array of data is stored in device Flash memory and transmitted via wireless communication to a mobile device. Using the application installed on the mobile device and the R-R-intervals array HRV analysis is performed for example to analyze and assess the emotional state of the user.



US 20190059752A1

(19) **United States**

(12) **Patent Application Publication**

BOTSVA et al.

(10) **Pub. No.: US 2019/0059752 A1**

(43) **Pub. Date: Feb. 28, 2019**

(54) **METHOD AND APPARATUS FOR CUFF LESS BLOOD PRESSURE MONITORING BASED ON SIMULTANEOUSLY MEASURED ECG AND PPG SIGNALS DESIGNED IN WRISTBAND FORM FOR CONTINUOUS WEARING**

(71) Applicant: **PLANEXTA, INC.**, Santa Monica, CA (US)

(72) Inventors: **Tetiana BOTSVA**, Dnipro (GB); **Dmytro CHERNETCHENKO**, Dnipro (GB)

(21) Appl. No.: **15/687,678**

(22) Filed: **Aug. 28, 2017**

Publication Classification

(51) **Int. Cl.**

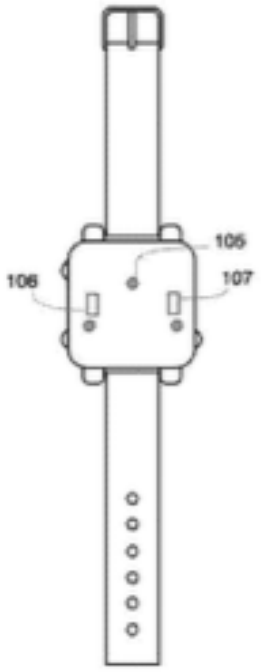
A61B 5/021	(2006.01)
A61B 5/0404	(2006.01)
A61B 5/0408	(2006.01)
A61B 5/0456	(2006.01)
A61B 5/1455	(2006.01)
A61B 5/00	(2006.01)
A61B 5/0205	(2006.01)

(52) **U.S. Cl.**

CPC ***A61B 5/02125*** (2013.01); *A61B 5/02427* (2013.01); ***A61B 5/0404*** (2013.01); ***A61B 5/04085*** (2013.01); ***A61B 5/0456*** (2013.01); ***A61B 5/14552*** (2013.01); ***A61B 5/681*** (2013.01); ***A61B 5/6843*** (2013.01); ***A61B 5/7221*** (2013.01); ***A61B 5/0006*** (2013.01); ***A61B 5/7475*** (2013.01); ***A61B 5/0205*** (2013.01); *A61B 2560/0223* (2013.01); *A61B 2562/0209* (2013.01); *A61B 2562/0238* (2013.01); *A61B 2560/0214* (2013.01); *A61B 2562/066* (2013.01); ***A61B 5/02116*** (2013.01)

(57) **ABSTRACT**

The present invention relates to the method and apparatus for cuffless measuring of blood pressure value from one limb of a subject. Proposed herein method of measuring blood pressure based on pressure calculation method from pulse wave velocity (PWV) value, which does not require any external device for calibration. PWV is calculated from three main signals: the heart electrical activity signal (ECG) and two photoplethysmographic signals (PPG). The described herein device comprises a set of ECG electrodes, optical and electronic sensors, a microcontroller for signal processing, a power supply unit, user interface means, and a wireless communication unit. The device is designed to monitor blood pressure and can be used for medical purposes or self-monitoring in everyday life.



Додаток Г

Проект програмного забезпечення мікроконтролеру STM32L151C8T6

comm_protocol.c – програмний протокол комунікації із ПК за допомогою інтерфейсу UART

```
/**
  ******************************************************************************
  * File Name          : comm_protocol.c
  * Description        : MCU communication protocol initialization
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "uartio.h"
#include "stm32f4xx_hal.h"
#include "comm_protocol.h"
#include "modules_emul.h"
#include "bio_impedance.h"

uint8_t runningcrc = 0;
extern uint32_t timerCounter;
extern uint32_t timerCounterMax;
extern ADC_HandleTypeDef hadc;

uint32_t measureCounter = 0;
uint8_t ledTrigger = 0;
uint32_t packageStreamCounter = 0;

uint8_t isCommandReceived = 0;          //new command received
CommPackage inCommPackage;              //communication command package
CommPackage outCommPackage;             //communication command package
streamPackage_t stream;                 //stream data package
uint8_t commandReceivedArray[COMMAND_PACKAGE_SIZE]; //received package
uint8_t commandResponseArray[COMMAND_PACKAGE_SIZE]; //send packet

uint8_t crcTable[256] = {0x00, 0x07, 0x0e, 0x09, 0x1c, 0x1b, 0x12, 0x15, 0x38,
                         0x3f, 0x36, 0x31, 0x24, 0x23, 0x2a, 0x2d, 0x70, 0x77,
                         0x7e, 0x79, 0x6c, 0x6b, 0x62, 0x65, 0x48, 0x4f, 0x46,
                         0x41, 0x54, 0x53, 0x5a, 0x5d, 0xe0, 0xe7, 0xee, 0xe9,
                         0xfc, 0xfb, 0xf2, 0xf5, 0xd8, 0xdf, 0xd6, 0xd1, 0xc4,
                         0xc3, 0xca, 0xcd, 0x90, 0x97, 0x9e, 0x99, 0x8c, 0x8b,
                         0x82, 0x85, 0xa8, 0xaf, 0xa6, 0xa1, 0xb4, 0xb3, 0xba,
                         0xbd, 0xc7, 0xc0, 0xc9, 0xce, 0xdb, 0xdc, 0xd5, 0xd2,
                         0xff, 0xf8, 0xf1, 0xf6, 0xe3, 0xe4, 0xed, 0xea, 0xb7,
                         0xb0, 0xb9, 0xbe, 0xab, 0xac, 0xa5, 0xa2, 0x8f, 0x88,
                         0x81, 0x86, 0x93, 0x94, 0x9d, 0x9a, 0x27, 0x20, 0x29,
                         0x2e, 0x3b, 0x3c, 0x35, 0x32, 0x1f, 0x18, 0x11, 0x16,
                         0x03, 0x04, 0x0d, 0x0a, 0x57, 0x50, 0x59, 0x5e, 0x4b,
                         0x4c, 0x45, 0x42, 0x6f, 0x68, 0x61, 0x66, 0x73, 0x74,
```

```
                0x7d, 0x7a, 0x89, 0x8e, 0x87, 0x80, 0x95, 0x92, 0x9b,
                0x9c, 0xb1, 0xb6, 0xbf, 0xb8, 0xad, 0xaa, 0xa3, 0xa4,
                0xf9, 0xfe, 0xf7, 0xf0, 0xe5, 0xe2, 0xeb, 0xec, 0xc1,
                0xc6, 0xcf, 0xc8, 0xdd, 0xda, 0xd3, 0xd4, 0x69, 0x6e,
                0x67, 0x60, 0x75, 0x72, 0x7b, 0x7c, 0x51, 0x56, 0x5f,
                0x58, 0x4d, 0x4a, 0x43, 0x44, 0x19, 0x1e, 0x17, 0x10,
                0x05, 0x02, 0x0b, 0x0c, 0x21, 0x26, 0x2f, 0x28, 0x3d,
                0x3a, 0x33, 0x34, 0x4e, 0x49, 0x40, 0x47, 0x52, 0x55,
                0x5c, 0x5b, 0x76, 0x71, 0x78, 0x7f, 0x6a, 0x6d, 0x64,
                0x63, 0x3e, 0x39, 0x30, 0x37, 0x22, 0x25, 0x2c, 0x2b,
                0x06, 0x01, 0x08, 0x0f, 0x1a, 0x1d, 0x14, 0x13, 0xae,
                0xa9, 0xa0, 0xa7, 0xb2, 0xb5, 0xbc, 0xbb, 0x96, 0x91,
                0x98, 0x9f, 0x8a, 0x8d, 0x84, 0x83, 0xde, 0xd9, 0xd0,
                0xd7, 0xc2, 0xc5, 0xcc, 0xcb, 0xe6, 0xe1, 0xe8, 0xef,
                0xfa, 0xfd, 0xf4, 0xf3};

uint8_t crcCalc(uint8_t* array, uint16_t length);
uint8_t crcByte(uint8_t oldcrc, uint8_t byte);
/*  Communication Protocol Part */
/* Get command received flag */
uint8_t isProtocolCommandReceived() {
        return isCommandReceived;
}
/* Command received flag set */
void setProtocolCommandReceived(uint8_t* array) {
     for(uint8_t i = 0; i < COMMAND_PACKAGE_SIZE; i++) {
           commandReceivedArray[i] = array[i];
     }
     isCommandReceived = 1;
}
/* Clear protocol package */
void clearProtocolPackage(CommPackage* comm){
  for(uint8_t i = 0; i < PACKAGE_COMM_SIZE; i++){
    comm->command[i] = 0;
  }
  for(uint8_t i = 0; i < PACKAGE_PARAMS_SIZE; i++){
    comm->parameters[i] = 0;
  }
  for(uint8_t i = 0; i < (PACKAGE_COMM_SIZE + PACKAGE_PARAMS_SIZE); i++){
    comm->crc_packet[i] = 0;
  }
  for(uint8_t i = 0; i < COMMAND_PACKAGE_SIZE; i++) {
    commandReceivedArray[i] = 0;
  }
}
/* Fill protocol package structure */
void setProtocolPackage(CommPackage* comm, uint8_t* array){
  runningcrc = 0;
  comm->crc_packet[0] = commandReceivedArray[2];
  comm->crc_packet[1] = commandReceivedArray[3];
  comm->crc_packet[2] = commandReceivedArray[4];
  for(uint8_t i = 0, k = 5; i < PACKAGE_PARAMS_SIZE; i++,k++){
```

```
    comm->crc_packet[i+PACKAGE_COMM_SIZE] = commandReceivedArray[k];
  }
uint8_t crc = crcCalc(comm->crc_packet,
(PACKAGE_COMM_SIZE+PACKAGE_PARAMS_SIZE));
  debugPrintLine("CRC=%d",crc);
  if((crc == commandReceivedArray[COMMAND_PACKAGE_SIZE-3]) ||
     (commandReceivedArray[4] == COMMAND_STOP_STREAM)){
    debugPrintLine("CRC is Ok");
    /* Fill protocol package with data */
    comm->command[0] = commandReceivedArray[2];
    comm->command[1] = commandReceivedArray[3];
    comm->command[2] = commandReceivedArray[4];
    for(uint8_t i = 0, k = 5; i < PACKAGE_PARAMS_SIZE; i++,k++){
      comm->parameters[i] = commandReceivedArray[k];
    }
  } else {
    debugPrintLine("CRC is Bad");
  }
}

void setResponsePackage(CommPackage* comm, uint32_t command, uint8_t* params, uint8_t
length){
  comm->command[0] = (command & 0xFF0000);
  comm->command[1] = (command & 0x00FF00);
  comm->command[2] = (command & 0x0000FF);
  for(uint8_t i = 0; i < length; i++){
    comm->parameters[i] = params[i];
  }
  comm->crc_packet[0] = comm->command[0];
  comm->crc_packet[1] = comm->command[1];
  comm->crc_packet[2] = comm->command[2];
  for(uint8_t i = 0; i < length; i++){
    comm->crc_packet[i+PACKAGE_COMM_SIZE] = comm->parameters[i];
  }
  for(uint8_t i = length+PACKAGE_COMM_SIZE; i < (PACKAGE_PARAMS_SIZE-length);
i++){
    comm->crc_packet[i] = 0x00;
  }
}

/* Callback for commands parsing */
void receiveCommandCallback(CommPackage* comm){
  setProtocolPackage(comm,(uint8_t*)&commandReceivedArray);
  uint16_t frequency = (comm->parameters[0] << 8) | comm->parameters[1];
  if(frequency < MIN_MEASURE_FREQUENCY) frequency = MIN_MEASURE_FREQUENCY;
  if(frequency > MAX_MEASURE_FREQUENCY) frequency =
MAX_MEASURE_FREQUENCY;
  uint16_t duration = (comm->parameters[2] << 8) | comm->parameters[3];
  if(duration < MIN_MEASURE_DURATION) duration = MIN_MEASURE_DURATION;
  if(duration > MAX_MEASURE_DURATION) duration = MAX_MEASURE_DURATION;
  uint16_t vref = 0;
  uint16_t gain = 0;
```

```
  uint8_t* params;
  /* Parse Commands And parameters */
  uint32_t comm_command = (comm->command[0] << 16) | (comm->command[1] << 8) | comm-
>command[2];
  switch (comm_command){
    case COMMAND_PPG_STREAM:
        uint8_t leds_ctrl = comm->parameters[4];
        uint8_t current_led = comm->parameters[5];
        debugPrintLine("PPG_STREAM");
        debugPrintLine("Fs=%d; Duration=%d; LedsFlag=%d ",frequency,duration,leds_ctrl);
        startStream();
        startPPG(frequency,duration,leds_ctrl,current_led, TRUE);
    break;
    case COMMAND_IMPEDANCE_STREAM:
        uint32_t Istim_f = (comm->parameters[4] << 32) | (comm->parameters[5] << 16) | (comm-
>parameters[6] << 8) | comm->parameters[7];
        uint16_t Istim = (comm->parameters[8] << 8) | commandReceivedArray[9];
        vref = (comm->parameters[10]) | comm->parameters[11];
        gain = (comm->parameters[12] << 8) | comm->parameters[13];
        debugPrintLine("IMPEDANCE_STREAM");
        debugPrintLine("Fs=%d; Duration=%d; Istim_f=%d; Istim=%d; vref=%d; gain=%d",
frequency,duration,Istim_f,Istim,vref,gain);
        startStream();
        startImpedance(frequency,duration,Istim_f,Istim,vref,gain,TRUE);
      break;
    case COMMAND_ECG_STREAM:
        vref = (comm->parameters[4]) | comm->parameters[5];
        gain = (comm->parameters[6] << 8) | comm->parameters[7];
        debugPrintLine("ECG_STREAM");
        debugPrintLine("Fs=%d; Duration=%d; vref=%d; gain=%d",frequency,duration,vref,gain);
        startStream();
        startECG(frequency,duration,vref,gain,TRUE);
      break;
    case COMMAND_STOP_STREAM:
      debugPrintLine("STREAM_STOP");
      /* Send the rest of samples of the frame */
      stopImpedance();
      stopStream();
      break;
    case COMMAND_GET_BATTERY_CHARGE:
      //get battery charge value
      //BAT_Monitor_Init();
      uint8_t battery_level = 100; //BAT_Monitor_GetLevel();
      debugPrintLine("GET_BATTERY_CHARGE");
      debugPrintLine("Battery level:%d%",battery_level);
      params = (uint8_t*)calloc(1,sizeof(uint8_t));
      params[0] = battery_level;

setResponsePackage(&outCommPackage,COMMAND_GET_BATTERY_CHARGE,params,1);
      sendCommandResponse(&outCommPackage);
      break;
    case COMMAND_GET_CONTACT_STATUS:
```

```
      debugPrintLine("GET_CONTACT_STATUS");
      uint8_t contact_status = 0x01;
      params = (uint8_t*)calloc(3,sizeof(uint8_t));
      /* Electrode-skin impedance level */
      params[0] = (0x01 & 0xFF00) >> 8;
      params[1] = (0x01 & 0x00FF);
      /* Contact status */
      params[2] = contact_status;

setResponsePackage(&outCommPackage,COMMAND_GET_CONTACT_STATUS,params,3);
      sendCommandResponse(&outCommPackage);
      break;
    case COMMAND_GET_DEVICE_ID:
      debugPrintLine("GET_DEVICE_ID");
      params = (uint8_t*)calloc(14,sizeof(uint8_t));
      params[0] = (STM32_UUID[0] & 0xFF000000) >> 32;
      params[1] = (STM32_UUID[0] & 0x00FF0000) >> 16;
      params[2] = (STM32_UUID[0] & 0x0000FF00) >> 8;
      params[3] = (STM32_UUID[0] & 0x000000FF);
      params[4] = (STM32_UUID[1] & 0xFF000000) >> 32;
      params[5] = (STM32_UUID[1] & 0x00FF0000) >> 16;
      params[6] = (STM32_UUID[1] & 0x0000FF00) >> 8;
      params[7] = (STM32_UUID[1] & 0x000000FF);
      params[8] = (STM32_UUID[2] & 0xFF000000) >> 32;
      params[9] = (STM32_UUID[2] & 0x00FF0000) >> 16;
      params[10] = (STM32_UUID[2] & 0x0000FF00) >> 8;
      params[11] = (STM32_UUID[2] & 0x000000FF);
      params[12] = DEVICE_TYPE;
      params[13] = FIRMWARE_VERSION;
      setResponsePackage(&outCommPackage,COMMAND_GET_DEVICE_ID,params,14);
      sendCommandResponse(&outCommPackage);
      break;
    default:
      debugPrintLine("UNKNOWN COMMAND GET");
      //loop back unknown command
      sendResponsePackage(commandReceivedArray);
      break;
    }
  /* Clear Communication Package */
  clearProtocolPackage(comm);
  clearProtocolPackage(&outCommPackage);
}

/*
void sendDelayedResponce(){
  uint8_t currentState = SM_Get_State();
  uint8_t* params;
  switch (currentState){
    case STATE_CONTACT_CHECK:
      uint8_t contact_status = getContactStatus();
      params = (uint8_t*)calloc(3,sizeof(uint8_t));
      params[0] = (currentContactLevel & 0xFF00) >> 8;
```

```
      params[1] = (currentContactLevel & 0x00FF);
      params[2] = contact_status;

setResponsePackage(&outCommPackage,COMMAND_GET_CONTACT_STATUS,params,3);
      sendCommandResponse(&outCommPackage);
    break;
  }
  clearProtocolPackage(&outCommPackage);
}
*/

void sendCommandResponse(CommPackage* comm){
  commandResponseArray[0] = COMM_FIRST_BYTE;
  commandResponseArray[1] = COMM_SECOND_BYTE;
  for(uint8_t i = PACKAGE_COMM_SIZE-1, k=0; k < PACKAGE_COMM_SIZE; i++,k++){
    commandResponseArray[i] = comm->command[k];
  }
  for(uint8_t i = PACKAGE_COMM_SIZE+2, k=0; k < PACKAGE_PARAMS_SIZE; i++,k++){
    commandResponseArray[i] = comm->parameters[k];
  }
  runningcrc = 0;
  uint8_t crc = crcCalc(comm-
>crc_packet,(PACKAGE_COMM_SIZE+PACKAGE_PARAMS_SIZE));
  debugPrintLine("CRC=%d",crc);
  commandResponseArray[COMMAND_PACKAGE_SIZE-3] = crc;
  commandResponseArray[COMMAND_PACKAGE_SIZE-2] = COMM_SECOND_BYTE;
  commandResponseArray[COMMAND_PACKAGE_SIZE-1] = COMM_FIRST_BYTE;
  sendResponsePackage(commandResponseArray);
}

void initStream(streamPackage_t *stream){
  stream->framePackageSamples = 0;
  stream->framePackageSize = 0;
  stream->streamMode = IDLE_STREAM;
  stream->streamBuffer = NULL;
}

void clearStream(streamPackage_t *stream){
  free(stream->streamBuffer);
  stream->streamMode = IDLE_STREAM;
  stream->streamBuffer = NULL;
}

void setStreamParameters(streamPackage_t *stream, streamMode_t stream_mode){
  /* Settling stream parameters */
  switch (stream_mode){
    case PPG_STREAM:
      stream->streamMode = PPG_STREAM;
      stream->framePackageSize = STREAM_PPG_PACKAGE_SIZE;
      stream->framePackageSamples = STREAM_PACKAGE_SAMPLES_L;
      break;
    case IMPEDANCE_STREAM:
```

```
      stream->streamMode = IMPEDANCE_STREAM;
      stream->framePackageSize = STREAM_IMP_PACKAGE_SIZE;
      stream->framePackageSamples = STREAM_PACKAGE_SAMPLES_L;
      break;
    case ECG_STREAM:
      stream->streamMode = ECG_STREAM;
      stream->framePackageSize = STREAM_ECG_PACKAGE_SIZE;
      stream->framePackageSamples = STREAM_PACKAGE_SAMPLES_L;
      break;
}
  cleanDataStream();
  setDataStreaming(TRUE);
  stream->streamBuffer = (uint8_t*)calloc((stream->framePackageSamples * stream-
>framePackageSize), sizeof(uint8_t));
  if(stream->streamBuffer == NULL){
    debugPrintLine("Allocation Memory Error");
  }
}

/*****************************************************************************
* Start Stream
*****************************************************************************/
void startStream(){
  measureCounter = 0;
  packageStreamCounter = 0;
}

/*****************************************************************************
* Stop streaming
*****************************************************************************/
void stopStream(){
  //send the rest of buffer
  if(packageStreamCounter > 0) {
       sendStreamPackage(stream.streamBuffer, packageStreamCounter*stream.framePackageSize);
       packageStreamCounter = 0;
  }
  setDataStreaming(FALSE);
  clearStream(&stream);
}

/* UART streaming function */
void Send_Stream(){
  if(getMeasureTimerTicks() < timerCounterMax){
      incMeasureTimerTicks();
      if(measureCounter > 100){
        measureCounter = 0;
      } else {
        measureCounter++;
      }
      if(stream.streamMode == PPG_STREAM){
        uint16_t spo2 = 0;
        uint16_t channel_0 = 0;
```

```
        uint16_t channel_1 = 0;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+0]=COMM_FIRST_BYTE;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+1]=(COMM_SECOND_BYTE & 0xF0) | ((timerCounter &
0xF0000)>>16);
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+2]=(timerCounter
& 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+3]=(timerCounter
& 0x00FF);
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+4]=(channel_0 &
0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+5]=(channel_0 &
0x00FF);
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+6]=(channel_1 &
0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+7]=(channel_1 &
0x00FF);
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+8]=(spo2 &
0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+9]=(spo2 &
0x00FF);
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+10]=(spo2 &
0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+stream.framePackageSize-2]=COMM_SECOND_BYTE;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+stream.framePackageSize-1]=COMM_FIRST_BYTE;

    } else if(stream.streamMode == IMPEDANCE_STREAM){

        uint16_t channel_0 = measureCounter;
        uint16_t channel_1 = measureCounter;

        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+0]=COMM_FIRST_BYTE;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+1]=(COMM_SECOND_BYTE & 0xF0) | ((timerCounter &
0xF0000)>>16);
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+2]=(timerCounter
& 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+3]=(timerCounter
& 0x00FF);
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+4]=(channel_0 &
0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+5]=(channel_0 &
0x00FF);
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+6]=(channel_1 &
0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+7]=(channel_1 &
0x00FF);
```

```
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+8]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+9]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+10]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+11]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+stream.framePackageSize-2]=COMM_SECOND_BYTE;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+stream.framePackageSize-1]=COMM_FIRST_BYTE;


    } else if(stream.streamMode == ECG_STREAM){

        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+0]=COMM_FIRST_BYTE;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+1]=(COMM_SECOND_BYTE & 0xF0) | ((timerCounter &
0xF0000)>>16);
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+2]=(timerCounter
& 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+3]=(timerCounter
& 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+4]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+5]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+6]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+7]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+stream.framePackageSize-2]=COMM_SECOND_BYTE;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+stream.framePackageSize-1]=COMM_FIRST_BYTE;


    } else if(stream.streamMode == MOTION_STREAM){

        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+0]=COMM_FIRST_BYTE;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+1]=(COMM_SECOND_BYTE & 0xF0) | ((timerCounter &
0xF0000)>>16);
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+2]=(timerCounter
& 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter * stream.framePackageSize)+3]=(timerCounter
& 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+4]=(measureCounter & 0xFF00) >> 8;
```

```
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+5]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+6]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+7]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+8]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+9]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+10]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+11]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+12]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+13]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+14]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+15]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+16]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+17]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+18]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+19]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+20]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+21]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+22]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+23]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+24]=(measureCounter & 0xFF00) >> 8;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+25]=(measureCounter & 0x00FF);
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+stream.framePackageSize-2]=COMM_SECOND_BYTE;
        stream.streamBuffer[(packageStreamCounter *
stream.framePackageSize)+stream.framePackageSize-1]=COMM_FIRST_BYTE;
}

      /* Send Stream Package with data via UART */
      if(stream.streamMode > IDLE_STREAM){
        if(packageStreamCounter < (stream.framePackageSamples - 1)) {
          packageStreamCounter++;
```

```
      } else {
        sendStreamPackage(stream.streamBuffer, (stream.framePackageSamples *
stream.framePackageSize));
        packageStreamCounter = 0;
      }
    }
   /* Measure end */
   } else {
    stopImpedance();
    stopStream();
   }

}

/* CRC calculation */
uint8_t crcCalc(uint8_t* array, uint16_t length){
      for(uint8_t i = 0; i < length; i++){
         runningcrc = crcByte(runningcrc, array[i]);
      }
      return runningcrc;
}

uint8_t crcByte(uint8_t oldcrc, uint8_t byte){
      uint8_t res = crcTable[oldcrc & 0xFF ^ byte & 0xFF];
      return res;
}
```

core.c – функції для низькорівневих налаштуваннь та ініціалізації апаратних засобів мікроконтролера

```
/**
 ******************************************************************************
 * File Name          : core.c
 * Description        : MCU core hardware initialization
 ******************************************************************************
 */
/* Includes ------------------------------------------------------------------*/
#include "core.h"
#include "main.h"
#include "stm32f4xx_hal.h"

/* Main Timer */
TIM_HandleTypeDef htim2;
/* Debug UART */
UART_HandleTypeDef huart3;
/* DAC module */
DAC_HandleTypeDef hdac;
/* DMA/DAC peripheral system */
DMA_HandleTypeDef hdma_dac1;
/* TIM8 for generation */
```

```
TIM_HandleTypeDef htim8;
/* ADC for analog signal */
ADC_HandleTypeDef hadc;
/* ADC/DMA to memory system */
DMA_HandleTypeDef hdma_adc1;

/** System Clock Configuration
*/
void SystemClock_Config(void)
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;

    /**Configure the main internal regulator output voltage
    */
  __HAL_RCC_PWR_CLK_ENABLE();

  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 168;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }

    /**Configure the Systick interrupt time
    */
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);
```

```
   /**Configure the Systick
   */
  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}

void MX_ADC_Init(ADC_mode mode) {
   /**Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of
conversion)
   */
  hadc.Instance = ADC1;
  hadc.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
  hadc.Init.Resolution = ADC_RESOLUTION12b;
  hadc.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc.Init.EOCSelection = EOC_SINGLE_CONV;
  hadc.Init.ContinuousConvMode = DISABLE;
  hadc.Init.DiscontinuousConvMode = DISABLE;
  hadc.Init.DMAContinuousRequests = ENABLE;

  switch(mode){

    case SINGLE_MODE:   //single conversion
      hadc.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T8_TRGO;
      hadc.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
      hadc.Init.ScanConvMode = DISABLE;
      hadc.Init.NbrOfConversion = 1;
      break;
    case SEQUENCE_MODE:         //sequence of channels
      hadc.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T2_TRGO;
      hadc.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
      hadc.Init.ScanConvMode = ENABLE;
      hadc.Init.NbrOfConversion = 2;
      break;
  }
HAL_ADC_Init(&hadc);
}

void ADC_Start(ADC_channel adc_channel, IMP_CH* imp){
  ADC_ChannelConfTypeDef sConfig;
  ADC_InjectionConfTypeDef sConfigInjected;

  /**Configure for the selected ADC regular channel its corresponding rank in the sequencer and its
sample time.
  */
  if(adc_channel == AN5){
    sConfig.Channel = ADC_CHANNEL_5;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_15CYCLES;
    HAL_ADC_ConfigChannel(&hadc, &sConfig);
```

```
    HAL_ADC_Start_DMA(&hadc, (uint32_t*)imp->adc_dma_buf, (uint32_t)imp->dma_size);
  } else if(adc_channel == AN1_AN2){

    sConfigInjected.InjectedChannel = ADC_CHANNEL_1;
    sConfigInjected.InjectedRank = 1;
    sConfigInjected.InjectedSamplingTime = ADC_SAMPLETIME_3CYCLES;
    sConfigInjected.InjectedNbrOfConversion = 2;
    sConfigInjected.InjectedOffset = 0;
    sConfigInjected.InjectedDiscontinuousConvMode = ENABLE;
    sConfigInjected.ExternalTrigInjecConv = ADC_EXTERNALTRIGINJECCONV_T2_TRGO;
    sConfigInjected.ExternalTrigInjecConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
    HAL_ADCEx_InjectedConfigChannel(&hadc, &sConfigInjected);

    sConfigInjected.InjectedChannel = ADC_CHANNEL_2;
    sConfigInjected.InjectedRank = 2;
    HAL_ADCEx_InjectedConfigChannel(&hadc, &sConfigInjected);
    HAL_ADCEx_InjectedStart(&hadc);
  }
}

void ADC_Stop(){
    //HAL_ADC_Stop(&hadc);
    HAL_ADC_Stop_DMA(&hadc);
}

/* DAC init function */
void MX_DAC_Init(void)
{

  DAC_ChannelConfTypeDef sConfig;

    /**DAC Initialization
    */
  hdac.Instance = DAC;
  if (HAL_DAC_Init(&hdac) != HAL_OK)
  {
    Error_Handler();
  }

    /**DAC channel OUT1 config
    */
  sConfig.DAC_Trigger = DAC_TRIGGER_T8_TRGO;
  sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_DISABLE;
  if (HAL_DAC_ConfigChannel(&hdac, &sConfig, DAC_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }

}

/* TIM8 init function */
void MX_TIM8_Init(void)
```

```
{

  TIM_ClockConfigTypeDef sClockSourceConfig;
  TIM_MasterConfigTypeDef sMasterConfig;
  htim8.Instance = TIM8;
  htim8.Init.Prescaler = 0; //0 , 1
  htim8.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim8.Init.Period = 1; //1 , 15
  htim8.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim8.Init.RepetitionCounter = 0;
  if (HAL_TIM_Base_Init(&htim8) != HAL_OK)
  {
    Error_Handler();
  }

  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim8, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim8, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * Enable DMA controller clock
  */
void MX_DMA_Init(void)
{
  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();
  __HAL_RCC_DMA2_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Stream5_IRQn interrupt configuration */
  //HAL_NVIC_SetPriority(DMA1_Stream5_IRQn, 0, 0);
  //HAL_NVIC_EnableIRQ(DMA1_Stream5_IRQn);
  /* DMA2_Stream0_IRQn interrupt configuration */
  //HAL_NVIC_SetPriority(DMA2_Stream0_IRQn, 0, 0);
  //HAL_NVIC_EnableIRQ(DMA2_Stream0_IRQn);

}

/* TIM2 init function */
void MX_TIM2_Init(uint8_t sampling_rate)
{
  TIM_ClockConfigTypeDef sClockSourceConfig;
  TIM_MasterConfigTypeDef sMasterConfig;
```

```
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 16000-1;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = sampling_rate*2 - 1;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }

  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
}

/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
*/
void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD, LED4_Pin|LED3_Pin|LED5_Pin|LED6_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pins : LED4_Pin LED3_Pin LED5_Pin LED6_Pin */
  GPIO_InitStruct.Pin = LED4_Pin|LED3_Pin|LED5_Pin|LED6_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
  /*Configure GPIO pin : BTN_USER_Pin */
  GPIO_InitStruct.Pin = BTN_USER_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
```

```
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(BTN_USER_GPIO_Port, &GPIO_InitStruct);
  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI0_IRQn);
 }
}
```

main.c – головна функція проекту

```
/**
  ******************************************************************************
  * File Name          : main.c
  * Description        : Main program body
  ******************************************************************************
  */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_hal.h"
#include "bio_impedance.h"
#include "core.h"
#include "comm_protocol.h"
#include "uartio.h"
#include "timers.h"

extern uint8_t clickFlag;          //button clicked flag
extern uint8_t isCommandReceived;  //command receiving flag
extern CommPackage inCommPackage;  //input command structure
extern CommPackage outCommPackage; //output response/command structure
extern streamPackage_t stream;     //stream structure

/* MAIN funcrion goes here */
int main(void)
{

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

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

  /* Initialize all configured peripherals */
  MX_GPIO_Init();

  /* UART initialize */
  UARTInit();

  /* DMA system initialize */
  MX_DMA_Init();

  /* DAC module inialization */
  MX_DAC_Init();
```

```
  /* ADC module initialization */
  MX_ADC_Init(SINGLE_MODE);

  debugPrintLine("<---------- Starting device ---------->");
  debugPrintLine("Device code:%d",DEVICE_TYPE);
  debugPrintLine("Firmware version:%d",FIRMWARE_VERSION);
  debugPrintLine("Hardware revision:%d",HARDWARE_REVISION);
  /* Init blink of LED */
  blinkLED(3);
  /* Prepare protocol for communication */
  clearProtocolPackage(&inCommPackage);
  clearProtocolPackage(&outCommPackage);
  initStream(&stream);

  while (1)
  {
    /* Check For Incoming Commands */
    if(isProtocolCommandReceived()){
      isCommandReceived = 0;
      /* Processing incoming command */
      receiveCommandCallback(&inCommPackage);
    }

    /* Button Interrupt. For Testing */
    if(clickFlag == 1){
      clickFlag = 0;
      HAL_Delay(100);
      if(isDataStreaming() == FALSE){
        startECG(STD_F_S,MEASURE_DURATION,STD_F_D,STD_S_I,STD_BIAS,STD_GAIN,
TRUE);
        LED3On();
      } else if(isDataStreaming() == TRUE){
        LED3Off();
        stopImpedance();
        stopStream();
      }
    }
  }
}

void blinkLED(uint8_t times){
  uint8_t toggles = (times * 2);
  for(int i=0;i<toggles;i++){
    HAL_GPIO_TogglePin(LED3_GPIO_Port, LED3_Pin);
    HAL_Delay(100);
  }
  HAL_GPIO_WritePin(LED3_GPIO_Port, LED3_Pin, GPIO_PIN_RESET);
}

/* Turn off all leds */
void LedsOff(){
```

```
  HAL_GPIO_WritePin(LED3_GPIO_Port, LED3_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(LED4_GPIO_Port, LED4_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(LED5_GPIO_Port, LED5_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(LED6_GPIO_Port, LED6_Pin, GPIO_PIN_RESET);
}

/* Control LED3 */
void LED3On(){
  HAL_GPIO_WritePin(LED3_GPIO_Port, LED3_Pin, GPIO_PIN_SET);
}

void LED3Off(){
  HAL_GPIO_WritePin(LED3_GPIO_Port, LED3_Pin, GPIO_PIN_RESET);
}

/* Control LED4 */
void LED4On(){
  HAL_GPIO_WritePin(LED4_GPIO_Port, LED4_Pin, GPIO_PIN_SET);
}

void LED4Off(){
  HAL_GPIO_WritePin(LED4_GPIO_Port, LED4_Pin, GPIO_PIN_RESET);
}

/* Control LED5 */
void LED5On(){
  HAL_GPIO_WritePin(LED5_GPIO_Port, LED5_Pin, GPIO_PIN_SET);
}

void LED5Off(){
  HAL_GPIO_WritePin(LED5_GPIO_Port, LED5_Pin, GPIO_PIN_RESET);
}

/* Control LED6 */
void LED6On(){
  HAL_GPIO_WritePin(LED6_GPIO_Port, LED6_Pin, GPIO_PIN_SET);
}

void LED6Off(){
  HAL_GPIO_WritePin(LED6_GPIO_Port, LED6_Pin, GPIO_PIN_RESET);
}
```

queue.c – драйвер для організації черги при передачі інформації за допомогою інтерфейсу UART у режимі реального часу

```
/*
 * queue.c
 *
*/
```

```
#include "queue.h"
#include <stdlib.h>

void InitByteQueue(ByteQueue* queue, uint32_t buffer_size) {

	queue->buffer = (uint8_t*)malloc(buffer_size * sizeof(uint8_t));
	queue->size = buffer_size;
	for(uint32_t i = 0; i < buffer_size; i++) {
		queue->buffer[i] = 0;
	}
	queue->current_size = 0;
	queue->front = 0;
	queue->rear = 0;
}

void DestroyByteQueue(ByteQueue* queue) {
	queue->current_size = 0;
	queue->size = 0;
	queue->front = 0;
	queue->rear = 0;
	free(queue->buffer);
}

void PushByteQueue(ByteQueue *queue, uint8_t byte) {

	queue->buffer[queue->rear] = byte;
	if(queue->rear < queue->size - 1) {
		queue->rear++;
	} else {
		queue->rear = 0;
	}
	if(queue->current_size <= queue->size) {
		queue->current_size++;
	} else {
		cleanQueue(queue);
	}
}

uint8_t PopByteQueue(ByteQueue* queue) {

	uint8_t result = queue->buffer[queue->front];
	if(queue->front < queue->size - 1) {
		queue->front++;
	} else {
		queue->front = 0;
	}
	if(queue->current_size > 0) {
		queue->current_size--;
	} else {
		cleanQueue(queue);
	}
	return result;
```

```
}

uint8_t ViewFrontByteQueue(ByteQueue* queue, int offset) {
	int index = queue->front + offset;
	uint8_t result = 0;
	if(index >= 0 && index < queue->size) {
		result = queue->buffer[index];
	} else if(index >= queue->size) {
		result = queue->buffer[index - queue->size];
	} else if(index < 0) {
		result = queue->buffer[index + queue->size];
	}
	return result;
}

uint8_t ViewRearByteQueue(ByteQueue* queue, int offset) {
	int index = queue->rear + offset;
	uint8_t result = 0;
	if(index >= 0 && index < queue->size) {
		result = queue->buffer[index];
	} else if(index >= queue->size) {
		result = queue->buffer[index - queue->size];
	} else if(index < 0) {
		result = queue->buffer[index + queue->size];
	}
	return result;
}

void cleanQueue(ByteQueue* queue) {
	queue->rear = 0;
	queue->front = 0;
	queue->current_size = 0;
	/*
	while(queue->current_size) {
		uint8_t trash = PopByteQueue(queue);
	}
	*/
}
```

timers.c – головний модуль управління апаратними таймерами мікроконтролера

```
/**
  ******************************************************************************
  * File Name          : timers.c
  * Description        : MCU core hardware initialization
  ******************************************************************************
  */
/* Includes ------------------------------------------------------------------*/
#include "timers.h"
#include "core.h"
#include "main.h"
#include "comm_protocol.h"
```

```
#include "stm32f4xx_hal.h"
const uint16_t sine_wave[SINE_RES] = {2048, 2060, 2072, 2084, 2096, 2108, 2120, 2132, 2144,
2156,
…
2011, 2023, 2035, 2047
};

uint32_t timerCounter = 0;
uint32_t timerCounterMax = 0;

extern TIM_HandleTypeDef htim2;
extern DAC_HandleTypeDef hdac;
extern DMA_HandleTypeDef hdma_dac1;
extern TIM_HandleTypeDef htim8;
extern uint8_t ledTrigger;

extern streamPackage_t stream;
extern uint16_t measureCounter;
extern uint32_t packageStreamCounter;

/* TIMER 2 INTERRUPT */
void TIM2_IRQHandler(void) {

  if(TIM2->SR & TIM_SR_UIF) {
    TIM2->SR &= ~TIM_SR_UIF;

    //send stream packages via UART
    Send_Stream();

  }

}

void setMeasureTimerCounts(uint16_t f_s, uint16_t duration){
  timerCounterMax = (duration*f_s);
}

void setMeasureTimerTicks(uint32_t ticks){
  timerCounter = ticks;
}

void incMeasureTimerTicks(){
  timerCounter++;
}

uint32_t getMeasureTimerTicks(){
  return timerCounter;
}

/* Starting Streaming Timer */
void TIM2_Start(){
```

```
  //debugPrintLine("Timer Starting");
  timerCounter = 0;
  measureCounter = 0;
  packageStreamCounter = 0;
  HAL_TIM_Base_Start_IT(&htim2);

  HAL_NVIC_SetPriority(TIM2_IRQn, 3, 3);
  HAL_NVIC_EnableIRQ(TIM2_IRQn);

}

void TIM2_Stop(){

  //debugPrintLine("Timer Stop");
  HAL_TIM_Base_Stop(&htim2);
  HAL_NVIC_DisableIRQ(TIM2_IRQn);

}

void TIM8Start(){
  TIM8->CNT = 0;
  TIM8->SR = 0;
  //HAL_TIM_Base_Start_IT(&htim8);
  //HAL_NVIC_EnableIRQ(TIM8_UP_TIM13_IRQn);
  HAL_TIM_Base_Start(&htim8);
}

void startSineGeneration(IMP_CH* imp){
  /* Prepare DMA array with sine samples */
  imp->dma_dac_buf = (uint16_t*)calloc(imp->dma_size,sizeof(uint16_t));

  if(imp->stim_mode == IMP_SINE_WAVE){
    uint16_t j = 0;
    for(uint16_t i=0;i<imp->dma_size;i++){
      imp->dma_dac_buf[i] = (uint16_t)(sine_wave[j]);
      j += imp->freq_div;
    }
  } else if(imp->stim_mode == IMP_MEANDR_WAVE){
    uint8_t pulse_sign = 0;
    for(uint16_t i = 0; i<imp->dma_size;i++){
      if(i%5 == 0){
        if(pulse_sign == 0){pulse_sign = 1;} else {pulse_sign = 0;}
      }
      if(pulse_sign == 0){
        imp->dma_dac_buf[i] = DAC_MIN + DAC_OFFSET;
      } else {
        imp->dma_dac_buf[i] = DAC_MAX - DAC_OFFSET;
      }
    }
  }

  /* Start DMA->DAC transferring, triggers with TIM9 cnt */
```

```
  HAL_DAC_Start_DMA(&hdac, DAC_CHANNEL_1, (uint32_t*)imp->dma_dac_buf, imp-
>dma_size, DAC_ALIGN_12B_R);
  MX_TIM8_Init();
  TIM8Start();
}

void stopSineGeneration(){
  HAL_DAC_Stop_DMA(&hdac,DAC_CHANNEL_1);
  HAL_TIM_Base_Stop(&htim8);
  //HAL_TIM_Base_Stop_IT(&htim8);
}

/* when interrupt is in enable state */
void TIM8_UP_TIM13_IRQHandler(void)
{/* TIM Update event */
  if(TIM8->SR & TIM_SR_UIF) {
    TIM8->SR &= ~TIM_SR_UIF;
     //do something
      if(ledTrigger == 1) {
         HAL_GPIO_WritePin(DAC_GPIO_Port, DAC_Pin, GPIO_PIN_SET);
         ledTrigger = 0;
      } else {
         HAL_GPIO_WritePin(DAC_GPIO_Port, DAC_Pin, GPIO_PIN_RESET);
         ledTrigger = 1;
      }
    }
}
```

uart.c – драйвер універсального послідовного інтерфейсу UART

```
/**
  ******************************************************************************
  * File Name          : uartio.c
  * Description        : MCU core hardware initialization
  ******************************************************************************
  */
/* Includes ------------------------------------------------------------------*/

#include "main.h"
#include "core.h"
#include "uartio.h"
#include "queue.h"
#include "comm_protocol.h"
#include "stm32f4xx.h"
#include "stm32f4xx_hal.h"
#include "stm32f4xx_hal_uart.h"
#include "stm32f4xx_hal_conf.h"
#include <stdarg.h>
#include <string.h>
uint8_t isStreaming = 0;
uint8_t dataTXRunning = 0;
uint32_t tempBuf = 0;
```

```
uint8_t findPacketStart = 0;

extern UART_HandleTypeDef huart3;

ByteQueue dataTXQueue;
ByteQueue dataRXQueue;

void config_USART3() {
        __USART3_CLK_ENABLE();
        huart3.Instance = USART3;
        huart3.Init.BaudRate = 115200;
        huart3.Init.WordLength = UART_WORDLENGTH_8B;
        huart3.Init.StopBits = UART_STOPBITS_1;
        huart3.Init.Parity = UART_PARITY_NONE;
        huart3.Init.Mode = UART_MODE_TX_RX;
        huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
        huart3.Init.OverSampling = UART_OVERSAMPLING_16;
        HAL_UART_Init(&huart3);
       __HAL_UART_ENABLE_IT(&huart3, UART_IT_RXNE | UART_IT_TC);
        HAL_NVIC_SetPriority(USART3_IRQn, 3, 0);
        HAL_NVIC_EnableIRQ(USART3_IRQn);
}

void UARTInit() {
        GPIO_USART3_Config();
       InitByteQueue(&dataTXQueue, 600);
        InitByteQueue(&dataRXQueue, 80);
        config_USART3();
}

void GPIO_USART3_Config(){

   /* Peripheral clock enable */
   GPIO_InitTypeDef GPIO_InitStruct;
   __HAL_RCC_USART3_CLK_ENABLE();
   /**USART3 GPIO Configuration
   PB10     ------> USART3_TX
   PB11     ------> USART3_RX
   */
   GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9;
   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
   GPIO_InitStruct.Pull = GPIO_PULLUP;
   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
   GPIO_InitStruct.Alternate = GPIO_AF7_USART3;
   HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

}

void USART3_IRQHandler() {
        //disableAllInterrupts();
        if(USART3->SR & USART_SR_CTS) {
                // Clear to send
```

```
        // Clean this flag:
        USART3->SR &= ~USART_SR_CTS;
}
if(USART3->SR & USART_SR_LBD) {
        // LIN break detection flag
        // Clean this flag:
        USART3->SR &= ~USART_SR_LBD;
}
if(USART3->SR & USART_SR_TXE) {
        // Transmit data register empty
        // Clean this flag:
        //USART3->DR = 0xFF;
}
if(USART3->SR & USART_SR_TC) {
        // Transmission complete
        // Clean this flag:
        USART3->SR &= ~USART_SR_TC;
        // Or
        // 1) Read SR again
        //uint32_t sr = USART3->SR;
        // 2) Write DR
        //USART3->DR = 0xFF;

        if(dataTXQueue.current_size > 0) {
                USART3->DR = PopByteQueue(&dataTXQueue);
        } else {
            if(!isDataStreaming()) cleanDataStream();
            dataTXRunning = 0;
        }
}

if(USART3->SR & USART_SR_RXNE) {
        // Read data register not empty
        // Clean this flag:
        // Read or Write DR
        uint32_t dr = USART3->DR;
      if(dr == COMM_SECOND_BYTE &&
        tempBuf == COMM_FIRST_BYTE &&
        findPacketStart == 0
       ){
        findPacketStart = 1;
        PushByteQueue(&dataRXQueue, tempBuf);
        //HAL_GPIO_WritePin(LED_USER_GPIO_Port, LED_USER_Pin, GPIO_PIN_SET);
      }

      if(findPacketStart == 1){
        PushByteQueue(&dataRXQueue, dr);
      }

      /* Stop stream commnad received */
      if(dataRXQueue.current_size == 5 && dr == COMMAND_STOP_STREAM){
        uint8_t byte0 = ViewFrontByteQueue(&dataRXQueue, 0);
```

```
        uint8_t byte1 = ViewFrontByteQueue(&dataRXQueue, 1);
        uint8_t byte4 = ViewFrontByteQueue(&dataRXQueue, 4);
        uint8_t array[COMMAND_PACKAGE_SIZE];
        if(byte0 == COMM_FIRST_BYTE &&
           byte1 == COMM_SECOND_BYTE &&
           byte4 == COMMAND_STOP_STREAM){
          for(uint8_t j = 0; j < COMMAND_PACKAGE_SIZE; j++) {
            array[j] = ViewFrontByteQueue(&dataRXQueue, j);
          }
          cleanDataStream();
          setProtocolCommandReceived((uint8_t*)&array);
          findPacketStart = 0;
          tempBuf = 0;
        }
      }

      if(dataRXQueue.current_size > COMMAND_PACKAGE_SIZE-1){
        uint8_t byte0 = ViewFrontByteQueue(&dataRXQueue, 0);
        uint8_t byte1 = ViewFrontByteQueue(&dataRXQueue, 1);
        uint8_t byte_last = ViewRearByteQueue(&dataRXQueue, -1);
        uint8_t byte_prev = ViewRearByteQueue(&dataRXQueue, -2);
        if(byte0 == COMM_FIRST_BYTE &&
           byte1 == COMM_SECOND_BYTE &&
           byte_last == COMM_FIRST_BYTE &&
           byte_prev == COMM_SECOND_BYTE){
          uint8_t array[COMMAND_PACKAGE_SIZE];
          for(uint8_t j = 0; j < COMMAND_PACKAGE_SIZE; j++) {
                array[j] = PopByteQueue(&dataRXQueue);
          }
          cleanDataStream();
          setProtocolCommandReceived((uint8_t*)&array);
          findPacketStart = 0;
          tempBuf = 0;
        } else {
          //debugPrintLine("Wrong Packet");
          cleanDataStream();
          findPacketStart = 0;
          tempBuf = 0;
        }
      }
      tempBuf = dr;
      // Or simple clean it
      // USART1->SR &= ~USART_SR_RXNE;
}

if(USART3->SR & USART_SR_IDLE) {
        // IDLE line detected
        // Clean this flag:
        // 1) Read SR again
        uint32_t sr = USART3->SR;
        // 2) Read or Write DR
        uint32_t dr = USART3->DR;
```

```
        }
        if(USART3->SR & USART_SR_ORE) {
                // Overrun error
                // Clean this flag:
                // 1) Read SR again
                uint32_t sr = USART3->SR;
                // 2) Read or Write DR
                uint32_t dr = USART3->DR;
        }
        if(USART3->SR & USART_SR_NE) {
                // Noise detected flag
                // Clean this flag:
                // 1) Read SR again
                uint32_t sr = USART3->SR;
                // 2) Read or Write DR
                uint32_t dr = USART3->DR;
        }
        if(USART3->SR & USART_SR_FE) {
                // Framing error
                // Clean this flag:
                // 1) Read SR again
                uint32_t sr = USART3->SR;
                // 2) Read or Write DR
                uint32_t dr = USART3->DR;
        }
        if(USART3->SR & USART_SR_PE) {
                // Parity error

                // Clean this flag:
                // 1) Wait for RXNE
                while(!(USART3->SR & USART_SR_RXNE));
                // 2) Read SR again
                uint32_t sr = USART3->SR;
                // 3) Read or Write DR
                uint32_t dr = USART3->DR;
        }
        //enableAllInterrupts();
}

void cleanDataStream() {
        cleanQueue(&dataTXQueue);
        cleanQueue(&dataRXQueue);
}

void sendStreamPackage(uint8_t* data, uint16_t length){
   if(isDataStreaming()){
     disableAllInterrupts();
     for(uint16_t i = 0; i < length; i++) {
       PushByteQueue(&dataTXQueue, data[i]);
     }
     if(!dataTXRunning) {
       USART3->DR = PopByteQueue(&dataTXQueue);
```

```
      dataTXRunning = 1;
    }
    enableAllInterrupts();
  }
}

void sendResponsePackage(uint8_t* data){
  disableAllInterrupts();
  for(uint16_t i = 0; i < COMMAND_PACKAGE_SIZE; i++) {
      PushByteQueue(&dataTXQueue, data[i]);
  }
  if(!dataTXRunning) {
      USART3->DR = PopByteQueue(&dataTXQueue);
     dataTXRunning = 1;
  }
  enableAllInterrupts();
}

void sendByteDataTX(uint8_t data) {
      USART3->CR1 &= ~USART_CR1_TCIE;
      if(dataTXRunning) {
            PushByteQueue(&dataTXQueue, data);
      } else {
            USART3->DR = data;
            dataTXRunning = 1;
      }
      USART3->CR1 |= USART_CR1_TCIE;
}

void printString(const char * string) {
 if(DEBUG_MODE == 0) return;
      disableAllInterrupts();
      uint16_t length = strlen(string);

      for(uint16_t i = 0; i < length; i++) {
            PushByteQueue(&dataTXQueue, string[i]);
      }
      if(!dataTXRunning) {
            USART3->DR = PopByteQueue(&dataTXQueue);
            dataTXRunning = 1;
      }
      enableAllInterrupts();
}

void printHex(uint32_t value, uint8_t bytes) {
 if(DEBUG_MODE == 0) return;
 if(bytes==2)
  debugPrintLine("0x%02x", value);
 else if(bytes==8)
  debugPrintLine("0x%08x", value);
 else
  debugPrintLine("0x%x", value);
```

```
}

int debugPrintLine(char *fmt, ...) {
  if(DEBUG_MODE == 0) return 0;
  char *buf = calloc(strlen(fmt)+40, sizeof(char));
  va_list aptr;
  int ret;
  va_start(aptr, fmt);
  ret = vsprintf(buf, fmt, aptr);
  va_end(aptr);
  strcat(buf, "\n");
  printString(buf);
  free(buf);
  return(ret);
}
uint8_t isDataTXRunning() {
        return dataTXRunning;
}
uint8_t isDataStreaming() {
        return isStreaming;
}
void setDataStreaming(uint8_t streaming) {
        isStreaming = streaming;
}
void disableUSART3() {
       USART3->CR1 &= ~USART_CR1_UE;
}
void enableUSART3() {
       USART3->CR1 |= USART_CR1_UE;
}
```

Додаток Д

Проект програмного забезпечення ПК для візуалізації та запису вхідних даних

```
from serial import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import QTimer
from multiprocessing import Queue
from pyqtgraph import QtCore
import pyqtgraph as pg
import numpy as np
import threading
import datetime
import queue
import glob
import time
import sys
import csv
import os
class crc8:
    def __init__(self):
        self.crcTable = (0x00, 0x07, 0x0e, 0x09, 0x1c, 0x1b, 0x12, 0x15, 0x38,
                         0x3f, 0x36, 0x31, 0x24, 0x23, 0x2a, 0x2d, 0x70, 0x77,
                         0x7e, 0x79, 0x6c, 0x6b, 0x62, 0x65, 0x48, 0x4f, 0x46,
                         0x41, 0x54, 0x53, 0x5a, 0x5d, 0xe0, 0xe7, 0xee, 0xe9,
                         0xfc, 0xfb, 0xf2, 0xf5, 0xd8, 0xdf, 0xd6, 0xd1, 0xc4,
                         0xc3, 0xca, 0xcd, 0x90, 0x97, 0x9e, 0x99, 0x8c, 0x8b,
                         0x82, 0x85, 0xa8, 0xaf, 0xa6, 0xa1, 0xb4, 0xb3, 0xba,
                         0xbd, 0xc7, 0xc0, 0xc9, 0xce, 0xdb, 0xdc, 0xd5, 0xd2,
                         0xff, 0xf8, 0xf1, 0xf6, 0xe3, 0xe4, 0xed, 0xea, 0xb7,
                         0xb0, 0xb9, 0xbe, 0xab, 0xac, 0xa5, 0xa2, 0x8f, 0x88,
                         0x81, 0x86, 0x93, 0x94, 0x9d, 0x9a, 0x27, 0x20, 0x29,
                         0x2e, 0x3b, 0x3c, 0x35, 0x32, 0x1f, 0x18, 0x11, 0x16,
                         0x03, 0x04, 0x0d, 0x0a, 0x57, 0x50, 0x59, 0x5e, 0x4b,
                         0x4c, 0x45, 0x42, 0x6f, 0x68, 0x61, 0x66, 0x73, 0x74,
                         0x7d, 0x7a, 0x89, 0x8e, 0x87, 0x80, 0x95, 0x92, 0x9b,
                         0x9c, 0xb1, 0xb6, 0xbf, 0xb8, 0xad, 0xaa, 0xa3, 0xa4,
                         0xf9, 0xfe, 0xf7, 0xf0, 0xe5, 0xe2, 0xeb, 0xec, 0xc1,
                         0xc6, 0xcf, 0xc8, 0xdd, 0xda, 0xd3, 0xd4, 0x69, 0x6e,
                         0x67, 0x60, 0x75, 0x72, 0x7b, 0x7c, 0x51, 0x56, 0x5f,
                         0x58, 0x4d, 0x4a, 0x43, 0x44, 0x19, 0x1e, 0x17, 0x10,
                         0x05, 0x02, 0x0b, 0x0c, 0x21, 0x26, 0x2f, 0x28, 0x3d,
                         0x3a, 0x33, 0x34, 0x4e, 0x49, 0x40, 0x47, 0x52, 0x55,
                         0x5c, 0x5b, 0x76, 0x71, 0x78, 0x7f, 0x6a, 0x6d, 0x64,
                         0x63, 0x3e, 0x39, 0x30, 0x37, 0x22, 0x25, 0x2c, 0x2b,
                         0x06, 0x01, 0x08, 0x0f, 0x1a, 0x1d, 0x14, 0x13, 0xae,
                         0xa9, 0xa0, 0xa7, 0xb2, 0xb5, 0xbc, 0xbb, 0x96, 0x91,
                         0x98, 0x9f, 0x8a, 0x8d, 0x84, 0x83, 0xde, 0xd9, 0xd0,
                         0xd7, 0xc2, 0xc5, 0xcc, 0xcb, 0xe6, 0xe1, 0xe8, 0xef,
                         0xfa, 0xfd, 0xf4, 0xf3)
```

```
    def crc(self, msg):
        runningcrc = 0
        for c in msg:
            runningcrc = self.crcByte(runningcrc, c)
        return runningcrc

    def crcByte(self, oldcrc, byte):
        res = self.crcTable[oldcrc & 0xFF ^ byte & 0xFF]
        return res

# class for logging data vlues
class Logger:
    def __init__(self, log_queue=None):
        self.log_queue = log_queue
        self.log_name = ''
        self.stop = False
        self.logging_thread = threading.Thread(target=self._write_log)
        self.first_run = True

    # set logging properties
    def set_logging(self, log_queue, log_name=''):
        # current date in string format for creating log file
        current_date = str(datetime.datetime.now())[:19]
        current_date = current_date.replace(':', '_')

        # create log_file name
        if log_name == '':
            self.log_name = 'Logs\\Data_log_%s.csv' % current_date
        else:
            self.log_name = 'Logs\\' + log_name + (', %s.csv' % current_date)
        os.makedirs(os.path.dirname(self.log_name), exist_ok=True)
        self.log_queue = log_queue

    # function that writes log in .csv file
    def _write_log(self):
        last_package_time = time.time()
        with open(self.log_name, 'a') as data_file:

            file_writer = csv.writer(data_file, delimiter=',', quotechar='|',
quoting=csv.QUOTE_MINIMAL)
            while not self.stop:
                # try get package
                package = []
                try:
                    package = self.log_queue.get(0)
                    last_package_time = time.time()
                except queue.Empty:
                    time.sleep(0.001)
                    # if there no packages for a while, we stop logging
                    if time.time() - last_package_time > 5:
                        self.stop = True
```

```
            # if package got successfully, an it's not empty, print it in log
            if package:
                file_writer.writerow(package)
        data_file.close()

    # function that starts logging
    def start_logging(self):
            self.logging_thread.start()

# class that manages device communication
class DeviceCommunicator:

    # initialize instance of communicator
    def __init__(self, data_queue=None):

        # data port
        self.port = Serial()

        # current state trackers
        self.is_streaming = False
        self.end_of_thread = False

        # thread for data reading
        self.data_thread = threading.Thread(target=self._get_data)
        self.buffer = b''
        self.package_chunk = []
        self.chunk_size = 10

        # shows, if data receiving thread was started
        self.first_run = True
        self.port_is_open = False

        # queue for parsed packages for data plotter
        self.queue = data_queue

        # timeout handler
        self.timeout = False
        self.time_last_data_received = 0

    # function for port configuration
    def port_configure(self, port, baudrate):
        self.port = Serial(port=port, baudrate=baudrate)

    # set data queue
    def set_queue(self, data_queue):
        self.queue = data_queue

    # this function gets data from device
    def _get_data(self):
        # function can read data until main plotting window is closed
        while not self.end_of_thread:
```

```
            # if there was command for streaming start, function try to read data
            if self.is_streaming:

                # read all bytes in device buffer
                data = b''
                try:
                    data = self.port.read(self.port.inWaiting())
                except:
                    pass

                self.buffer += data

                if data == b'':
                    if time.time() - self.time_last_data_received >= 5:
                        self.timeout = True

                else:
                    self.time_last_data_received = time.time()

                # search for start and end of package
                end_index = self.buffer.find(b'\x55\xAA')
                if end_index > -1:
                    candidate = self.buffer[:(end_index+2)]
                    self.buffer = self.buffer[(end_index+2):]

                    first = candidate[0:1]
                    second = candidate[1] >> 4

                    if (first == b'\xAA') & (second == 5):
                        self.package_chunk.append(candidate)

                    if (len(self.package_chunk) >= self.chunk_size) | (time.time() -
self.time_last_data_received >= 0.05):
                        self.queue.put(self.package_chunk)
                        self.package_chunk = []
                        time.sleep(0.001)

            else:
                time.sleep(0.001)
                if len(self.buffer) | len(self.package_chunk):
                    self.buffer = b''
                    self.package_chunk = []

    # this function encodes command, which are send to the device
    def _encode_command(self, command, parameters=None):
        if parameters:
            if parameters['frequency'] > 100:
                self.chunk_size = 1 + int(0.05 * parameters['frequency'])
            else:
                self.chunk_size = 1

        # get command code in bytes
```

```
        command_code = command.to_bytes(3, byteorder='big')
        # getting parameters code in bytes
        parameters_code = 0
        if command == 1:
            parameters_code = int.from_bytes(parameters['frequency'].to_bytes(2, byteorder='big') +
parameters['duration'].to_bytes(2, byteorder='big') +
                                             parameters['LED'].to_bytes(1, byteorder='big') + parameters['current
LED'].to_bytes(1, byteorder='big'), byteorder='little').to_bytes(32, byteorder='little')
        elif command == 2:
            parameters_code = int.from_bytes(parameters['frequency'].to_bytes(2, byteorder='big') +
parameters['duration'].to_bytes(2, byteorder='big') +
                                             parameters['stimul_frequency'].to_bytes(4, byteorder='big') +
parameters['stimul_current'].to_bytes(2, byteorder='big') +
                                             parameters['Vref_impedance'].to_bytes(2, byteorder='big') +
parameters['Gain_impedance'].to_bytes(1, byteorder='big'),
                                             byteorder='little').to_bytes(32, byteorder='little')

        elif command == 4:
            parameters_code = int.from_bytes(parameters['frequency'].to_bytes(2, byteorder='big') +
parameters['duration'].to_bytes(2, byteorder='big') +
                                             parameters['Vref_ECG'].to_bytes(2, byteorder='big') +
parameters['Gain_ECG'].to_bytes(1, byteorder='big'),
                                             byteorder='little').to_bytes(32, byteorder='little')

        elif command == 5:
            parameters_code = int.from_bytes(parameters['frequency'].to_bytes(2, byteorder='big') +
parameters['duration'].to_bytes(2, byteorder='big'),
                                             byteorder='little').to_bytes(32, byteorder='little')

        elif command == 6:
            parameters_code = int.from_bytes(parameters['frequency'].to_bytes(2, byteorder='big') +
parameters['duration'].to_bytes(2, byteorder='big') +
                                             parameters['scale'].to_bytes(1, byteorder='big'),
byteorder='little').to_bytes(32, byteorder='little')

        else:
            parameters_code = parameters_code.to_bytes(32, byteorder='big')

        # evaluation of crc8 for command and parameters
        crc = (crc8().crc(command_code + parameters_code)).to_bytes(1, byteorder='big')

        # adding bytes of start and end of command string
        result = b'\xAA\x55' + command_code + parameters_code + crc + b'\x55\xAA'
        return result

    # this function starts stream from device
    def start_stream(self, command_code, parameters):

        # send command to start stream
        command = self._encode_command(command_code, parameters)

        # reset port
```

```
        self.port.flushInput()
        self.port.flushOutput()
        self.port.close()
        time.sleep(0.01)
        self.port.open()

        self.port.write(command)

        # start time measure
        self.time_last_data_received = time.time()

        # clear buffers
        self.package_chunk = []
        self.buffer = b''

        # set streaming to true
        self.is_streaming = True
        self.timeout = False

        # if stream starts first time, we starting a data receiving thread
        if self.first_run:
            self.data_thread.start()
            self.first_run = False

    # function, that stops stream
    def stop_stream(self, write=True):

        # set streaming to false
        self.is_streaming = False
        self.timeout = False
        self.package_chunk = []
        self.buffer = b''

        # send stop command to device
        command = self._encode_command(3)

        if write:
            self.port.write(command)

        self.port.flushInput()
        self.port.flushOutput()
        self.port.close()
        time.sleep(0.01)
        self.port.open()


    def check_status(self, command_code):
        self.port.flushInput()
        self.port.flushOutput()
        time.sleep(0.1)

        try:
```

```
            data = self.port.read(self.port.inWaiting())
        except SerialException:
            pass

        command = self._encode_command(command_code)
        self.port.write(command)
        time_out = False
        status_got = False
        start_time = time.time()
        buffer = b''

        while not (time_out | status_got):

            data = self.port.read(self.port.inWaiting())
            buffer += data

            end_index = buffer.find(b'\x55\xAA')
            if end_index > -1:
                # buffer = buffer[:(end_index+2)]
                status_got = True

            if time.time() - start_time >= 5:
                time_out = True

        try:
            data = self.port.read(self.port.inWaiting())
        except SerialException:
            pass

        self.port.flushInput()
        self.port.flushOutput()

        if status_got:
            return buffer
        else:
            return 'timeout'

    # function that ends data receiving thread
    def end(self):
        self.end_of_thread = True

    # function, that gets list of available ports
    @staticmethod
    def serial_ports():
        if sys.platform.startswith('win'):
            ports = ['COM%s' % (i + 1) for i in range(256)]
        elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):
            # this excludes current terminal "/dev/tty"
            ports = glob.glob('/dev/tty[A-Za-z]*')
        elif sys.platform.startswith('darwin'):
            ports = glob.glob('/dev/tty.*')
        else:
```

```
            raise EnvironmentError('Unsupported platform')

        result = []
        for port in ports:
            try:
                s = Serial(port)
                s.close()
                result.append(port)
            except (OSError, SerialException):
                pass
        result.append('---')
        return result


# useful widget
class Scroller(QScrollArea):
    def __init__(self, parent=None):
        QScrollArea.__init__(self, parent)

    def wheelEvent(self, ev):
        if ev.type() == QtCore.QEvent.Wheel:
            ev.ignore()


# useful widget
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.exit_call = False

    def closeEvent(self, event):
        self.exit_call = True


# class for plots with local duration processing
class Canvas:
    def __init__(self, central_widget):

        self.plot = pg.PlotWidget(central_widget)
        self.is_tracking = False
        self.update_duration = False
        self.duration = 10
        self.button = QPushButton('set')
        self.button.clicked.connect(self._update_duration_init)

        checkbox = QCheckBox('Signal tracking', self.plot)
        checkbox.stateChanged.connect(self._change_tracking)
        label = QLabel('Local duration:', self.plot)
        self.entry = QLineEdit('10', self.plot)

        hbox = QHBoxLayout()
        hbox.addStretch(1)
```

```
        hbox.addWidget(checkbox)
        hbox.addWidget(label)
        hbox.addWidget(self.entry)
        hbox.addWidget(self.button)

        vbox = QVBoxLayout(self.plot)
        vbox.addLayout(hbox)
        vbox.addStretch(1)

    def _change_tracking(self, state=False):
        self.is_tracking = state

    def _update_duration_init(self):
        self.update_duration = True

# fast plotting class
class MultiLine(pg.QtGui.QGraphicsPathItem):
    def __init__(self, x, y, pencolor, penwidth=0.5):
        x = np.array(x)
        y = np.array(y)
        connect = np.ones(x.shape, dtype=bool)
        self.path = pg.arrayToQPath(x.flatten(), y.flatten(), connect.flatten())
        pg.QtGui.QGraphicsPathItem.__init__(self, self.path)
        self.setPen(pg.mkPen(color=pencolor, width=penwidth))

    def shape(self):  # override because QGraphicsPathItem.shape is too expensive.
        return pg.QtGui.QGraphicsItem.shape(self)

    def boundingRect(self):
        return self.path.boundingRect()


# window with graphs
class Window:
    # initialize plotting window
    def __init__(self, available_ports=None, data_queue=None, log_queue=None):
        # GENERAL PROPERTIES
        # main window
        self.MainWindow = MainWindow()
        self.central_widget = QWidget(self.MainWindow)
        self.MainWindow.setCentralWidget(self.central_widget)
        self.MainWindow.setWindowTitle('Data Plotter v.1.13')
        self.app = QApplication(sys.argv)
        self.screen = self.app.primaryScreen()
        self.screen_width = self.screen.size().width()
        self.screen_height = self.screen.size().height()
        self.MainWindow.setGeometry(int(0.01 * self.screen_width), int(0.05 * self.screen_height),
int(0.98 * self.screen_width), int(0.9 * self.screen_height))
        self.screen_width = int(0.98 * self.screen_width)
        self.screen_height = int(0.9 * self.screen_height)

        # main layout
```

```
self.main_horizontal_layout = QHBoxLayout(self.central_widget)

self.controls_frame = QFrame()
self.controls_frame.setFrameShape(QFrame.StyledPanel)
self.controls_frame.setFixedSize(int(0.15*self.screen_width), int(0.99*self.screen_height))
self.plotting_frame = QFrame()
self.plotting_frame.setFrameShape(QFrame.StyledPanel)
self.plotting_frame.setFixedSize(int(0.85*self.screen_width), int(0.99*self.screen_height))

# add plotting and controls frames into layout
self.main_horizontal_layout.addWidget(self.controls_frame)
self.main_horizontal_layout.addWidget(self.plotting_frame)

# plotting frame initialization
self.plotting_horizontal_layout = QHBoxLayout(self.plotting_frame)
self.scrollArea = Scroller(self.plotting_frame)
self.plotting_horizontal_layout.addWidget(self.scrollArea)
self.scrollAreaWidgetContents = QWidget()
self.scrollArea.setWidget(self.scrollAreaWidgetContents)
self.vertical_plotting_layout = QVBoxLayout(self.scrollAreaWidgetContents)

# controls frame initialization
self.vertical_controls_layout = QVBoxLayout(self.controls_frame)

# mode choice
self.mode_layout = QHBoxLayout()
self.mode_label = QLabel('Mode:')
self.mode_choice = QComboBox()
modes = ['PPG', 'Impedance', 'ECG', 'Motion', 'Body temp']
self.mode_choice.addItems(modes)
self.mode_choice.activated[str].connect(self._set_mode)
self.mode_layout.addWidget(self.mode_label)
self.mode_layout.addWidget(self.mode_choice)
self.mode_layout.addStretch(1)
self.vertical_controls_layout.addLayout(self.mode_layout)
self.vertical_controls_layout.addSpacing(int(0.01 * self.screen_height))

# logging parameters
self.log_layout = QHBoxLayout()
self.log_name_input = QLineEdit()
self.logging_checkbox = QCheckBox('Log')
self.logging_checkbox.setChecked(True)
self.log_layout.addWidget(self.log_name_input)
self.log_layout.addWidget(self.logging_checkbox)
self.log_layout.addStretch(1)
self.vertical_controls_layout.addLayout(self.log_layout)
self.vertical_controls_layout.addSpacing(int(0.01 * self.screen_height))

# device port
self.device_layout = QHBoxLayout()
self.device_label = QLabel('Device port:')
self.device_ports = QComboBox()
```

```
if not available_ports:
    available_ports = ['---']
self.device_ports.addItems(available_ports)
self.device_layout.addWidget(self.device_label)
self.device_layout.addWidget(self.device_ports)
self.device_layout.addStretch(1)
self.vertical_controls_layout.addLayout(self.device_layout)
self.vertical_controls_layout.addSpacing(int(0.01 * self.screen_height))

# control buttons
self.button_layout = QHBoxLayout()
self.start_button = QPushButton('Start')
self.button_layout.addWidget(self.start_button)
self.button_layout.addStretch(1)
self.stop_button = QPushButton('Stop')
self.button_layout.addWidget(self.stop_button)
self.vertical_controls_layout.addLayout(self.button_layout)
self.vertical_controls_layout.addSpacing(int(0.01 * self.screen_height))

# common parameters
self.common_layout = QHBoxLayout()

# frequency
self.frequency_layout = QVBoxLayout()
self.frequency_label = QLabel('Frequency:')
frequency_options = ['250', '500']
self.frequency_options = QComboBox()
self.frequency_options.addItems(frequency_options)
self.frequency_layout.addWidget(self.frequency_label)
self.frequency_layout.addWidget(self.frequency_options)
self.common_layout.addLayout(self.frequency_layout)

self.common_layout.addStretch(1)

# duration
self.duration_layout = QVBoxLayout()
self.duration_label = QLabel('Duration:')
self.duration_input = QLineEdit('100')
self.duration_input.setFixedWidth(int(0.05*self.screen_width))
self.duration_layout.addWidget(self.duration_label)
self.duration_layout.addWidget(self.duration_input)
self.common_layout.addLayout(self.duration_layout)

self.vertical_controls_layout.addLayout(self.common_layout)
self.vertical_controls_layout.addSpacing(int(0.01 * self.screen_height))

# OPTIONS FRAMES

# PPG
self.ppg_widget = QWidget()
self.ppg_layout = QVBoxLayout(self.ppg_widget)
```

```
# LEDs
self.led_layout = QHBoxLayout()
self.led_layout1 = QVBoxLayout()
self.led_layout2 = QVBoxLayout()

self.green_led = QRadioButton('Green LED')
self.green_led.setChecked(True)
self.red_led = QRadioButton('Red LED')
self.infro_led = QRadioButton('Infro LED')
self.spo2 = QRadioButton('SpO2\nOxygenation')

led_group = QButtonGroup(self.ppg_widget)
led_group.addButton(self.green_led)
led_group.addButton(self.red_led)
led_group.addButton(self.infro_led)
led_group.addButton(self.spo2)

self.led_layout1.addWidget(self.green_led)
self.led_layout1.addWidget(self.red_led)
self.led_layout1.addStretch(1)
self.led_layout2.addWidget(self.infro_led)
self.led_layout2.addWidget(self.spo2)
self.led_layout2.addStretch(1)

self.led_layout.addLayout(self.led_layout1)
self.led_layout.addStretch(1)
self.led_layout.addLayout(self.led_layout2)

self.ppg_layout.addLayout(self.led_layout)

self.led_edit_layout = QHBoxLayout()
self.led_edit_layout.addStretch(1)
vert_layout = QVBoxLayout()
self.led_label = QLabel('Current LED:')
self.led_input = QLineEdit('0')
self.led_input.setFixedWidth(int(0.05 * self.screen_width))
vert_layout.addWidget(self.led_label)
vert_layout.addWidget(self.led_input)
vert_layout.addStretch(1)
self.led_edit_layout.addLayout(vert_layout)
self.led_edit_layout.addStretch(1)

self.ppg_layout.addLayout(self.led_edit_layout)

# oxygenation and heart rate
self.ppg_params_layout = QHBoxLayout()

self.oxy_layout = QVBoxLayout()
self.oxy_label = QLabel('Oxygenation:')
self.oxy_info = QLineEdit()
self.oxy_info.setReadOnly(True)
self.oxy_info.setFixedWidth(int(0.05 * self.screen_width))
```

```
self.oxy_layout.addWidget(self.oxy_label)
self.oxy_layout.addWidget(self.oxy_info)
self.oxy_layout.addStretch(1)

self.heart_layout = QVBoxLayout()
self.heart_label = QLabel('Heart rate:')
self.heart_info = QLineEdit()
self.heart_info.setReadOnly(True)
self.heart_info.setFixedWidth(int(0.05 * self.screen_width))
self.heart_layout.addWidget(self.heart_label)
self.heart_layout.addWidget(self.heart_info)
self.heart_layout.addStretch(1)

self.ppg_params_layout.addLayout(self.oxy_layout)
self.ppg_params_layout.addStretch(1)
self.ppg_params_layout.addLayout(self.heart_layout)

self.ppg_layout.addLayout(self.ppg_params_layout)
self.ppg_layout.addStretch(1)

# IMPEDANCE
self.impedance_widget = QWidget()
self.impedance_layout = QVBoxLayout(self.impedance_widget)

# input options
self.stimulation_frequency_label = QLabel('Stimulation\nfrequency:')
self.stimulation_frequency_input = QLineEdit('0')
self.stimulation_frequency_input.setFixedWidth(int(0.05 * self.screen_width))

self.stimulation_current_label = QLabel('Stimulation\ncurrent:')
self.stimulation_current_input = QLineEdit('0')
self.stimulation_current_input.setFixedWidth(int(0.05 * self.screen_width))

self.ref_voltage_impedance_label = QLabel('Ref voltage:')
self.ref_voltage_impedance_input = QLineEdit('0')
self.ref_voltage_impedance_input.setFixedWidth(int(0.05 * self.screen_width))

self.gain_voltage_impedance_label = QLabel('Gain voltage:')
gain_options = ['0', '2', '4', '6', '8', '10', '16', '32']
self.gain_voltage_impedance_options = QComboBox()
self.gain_voltage_impedance_options.addItems(gain_options)

self.impedance_layout1 = QVBoxLayout()
self.impedance_layout1.addWidget(self.stimulation_frequency_label)
self.impedance_layout1.addWidget(self.stimulation_frequency_input)
self.impedance_layout1.addStretch(1)
self.impedance_layout1.addWidget(self.stimulation_current_label)
self.impedance_layout1.addWidget(self.stimulation_current_input)

self.impedance_layout2 = QVBoxLayout()
self.impedance_layout2.addStretch(1)
self.impedance_layout2.addWidget(self.ref_voltage_impedance_label)
```

```
self.impedance_layout2.addWidget(self.ref_voltage_impedance_input)
self.impedance_layout2.addStretch(1)
self.impedance_layout2.addWidget(self.gain_voltage_impedance_label)
self.impedance_layout2.addWidget(self.gain_voltage_impedance_options)

self.impedance_options_layout = QHBoxLayout()
self.impedance_options_layout.addLayout(self.impedance_layout1)
self.impedance_options_layout.addStretch(1)
self.impedance_options_layout.addLayout(self.impedance_layout2)

self.impedance_layout.addLayout(self.impedance_options_layout)

# parameter
self.abs_z_label = QLabel('Skin |Z|:')
self.abs_z_info = QLineEdit()
self.abs_z_info.setReadOnly(True)
self.abs_z_info.setFixedWidth(int(0.05 * self.screen_width))
self.impedance_parameter_layout = QHBoxLayout()
self.support_layout = QVBoxLayout()
self.impedance_parameter_layout.addStretch(1)
self.support_layout.addWidget(self.abs_z_label)
self.support_layout.addWidget(self.abs_z_info)
self.impedance_parameter_layout.addLayout(self.support_layout)
self.impedance_parameter_layout.addStretch(1)

self.impedance_layout.addLayout(self.impedance_parameter_layout)
self.impedance_layout.addStretch(1)

# ECG
self.ecg_widget = QWidget()
self.ecg_layout = QHBoxLayout(self.ecg_widget)

self.ref_voltage_ecg_label = QLabel('Ref voltage:')
self.ref_voltage_ecg_input = QLineEdit('0')
self.ref_voltage_ecg_input.setFixedWidth(int(0.05 * self.screen_width))
self.ref_layout = QVBoxLayout()
self.ref_layout.addWidget(self.ref_voltage_ecg_label)
self.ref_layout.addWidget(self.ref_voltage_ecg_input)
self.ecg_layout.addLayout(self.ref_layout)
self.ref_layout.addStretch(1)

self.ecg_layout.addStretch(1)

self.gain_voltage_ecg_label = QLabel('Gain voltage:')
self.gain_voltage_ecg_options = QComboBox()
self.gain_voltage_ecg_options.addItems(gain_options)
self.gain_layout = QVBoxLayout()
self.gain_layout.addWidget(self.gain_voltage_ecg_label)
self.gain_layout.addWidget(self.gain_voltage_ecg_options)
self.gain_layout.addStretch(1)

self.ecg_layout.addLayout(self.gain_layout)
```

```
# options layout
self.options_layout = QStackedLayout()
self.options_layout.addWidget(self.ppg_widget)
self.options_layout.addWidget(self.impedance_widget)
self.options_layout.addWidget(self.ecg_widget)
self.options_layout.addWidget(self.motion_widget)
self.options_layout.addWidget(self.body_widget)
self.vertical_controls_layout.addLayout(self.options_layout)
self.vertical_controls_layout.addStretch(1)

# check status layout

self.check_status_layout = QVBoxLayout()
horizontal_layout = QHBoxLayout()
self.status_label = QLabel('Check status of:')
self.status_options = QComboBox()
statuses = ['battery charge', 'contact status', 'device ID']
self.status_options.addItems(statuses)
self.status_button = QPushButton('check')

self.check_status_layout.addWidget(self.status_label)
horizontal_layout.addWidget(self.status_options)
horizontal_layout.addWidget(self.status_button)
self.check_status_layout.addLayout(horizontal_layout)
self.vertical_controls_layout.addLayout(self.check_status_layout)

# messages
self.messages_label = QLabel('Messages:')
self.messages = QPlainTextEdit()
self.messages.setReadOnly(True)
self.messages.setFixedWidth(int(0.14 * self.screen_width))
self.messages.setFixedHeight(int(0.35 * self.screen_height))
self.vertical_controls_layout.addWidget(self.messages_label)
self.vertical_controls_layout.addWidget(self.messages)

# OTHER PROPERTIES

self.canvases = []
self.trackers = [False] * 9
self.local_durations = [10] * 9

self.data_queue = data_queue
self.log_queue = log_queue
self.buffer = []
self.stop_signal = True
self.stop_call = False

default_data = {'name': 'Test'}
self.canvases_data = []
for i in range(0, 9):
    self.canvases_data.append(default_data.copy())
self.graphs_points = [[], [], [], [], [], [], [], [], [], [], [], []]
```

```
        self.indexes = [0] * 12
        self.just_started = True

        self.number_of_plots = 2
        self.last_update_time = 0
        self.last_package_time = 0
        self.timeout = False

        # options tracking variables
        self.stream_code = 0
        self.package_structure = []

        # lists for optional parameters
        self.parameters_indexes = []
        self.parameters_outputs = []

        # INITIALIZE DEFAULTS

        self.initial_frequency = 250
        self.duration = 100
        self.last_mode = 'PPG'
        self._set_mode('PPG')

    # remove old plots
    @staticmethod
    def _clear_layout(layout):
        while layout.count() > 0:
            item = layout.takeAt(0)
            if item:
                w = item.widget()
                if w:
                    w.deleteLater()

    # set frequencies for current stream
    def _set_frequencies(self, frequencies):
        self.frequency_options.clear()
        self.frequency_options.addItems(frequencies)

    # set current streaming mode
    def _set_mode(self, text):

        if text == 'PPG':
            self.options_layout.setCurrentIndex(0)
            self._set_frequencies(['250', '500'])
            self.stream_code = 1
            self.package_structure = [2] * 3 + [1]
            self.parameters_indexes = [3, 4]
            self.parameters_outputs = [self.oxy_info, self.heart_info]
            self.number_of_plots = 2
            self.canvases_data[0]['name'] = 'PPG channel 0'
            self.canvases_data[1]['name'] = 'PPG channel 1'
```

```
elif text == 'Impedance':
    self.options_layout.setCurrentIndex(1)
    self._set_frequencies(['250', '500'])
    self.stream_code = 2
    self.package_structure = [2] * 4
    self.parameters_indexes = [4]
    self.parameters_outputs = [self.abs_z_info]
    self.number_of_plots = 3
    self.canvases_data[0]['name'] = 'Impedance channel 0'
    self.canvases_data[1]['name'] = 'Impedance channel 1'
    self.canvases_data[2]['name'] = 'Detector channel'

elif text == 'ECG':
    self.options_layout.setCurrentIndex(2)
    self._set_frequencies(['250', '500', '1000'])
    self.stream_code = 4
    self.package_structure = [2] * 2
    self.parameters_indexes = []
    self.parameters_outputs = []
    self.number_of_plots = 2
    self.canvases_data[0]['name'] = 'ECG channel 0'
    self.canvases_data[1]['name'] = 'Detector channel 2'

elif text == 'Motion':
    self.options_layout.setCurrentIndex(3)
    self._set_frequencies(['10', '50', '100'])
    self.stream_code = 5
    self.package_structure = [2] * 11
    self.parameters_indexes = [10, 11]
    self.parameters_outputs = [self.velocity_info, self.distance_info]
    self.number_of_plots = 9
    self.canvases_data[0]['name'] = 'Accelerometer x-axis'
    self.canvases_data[1]['name'] = 'Accelerometer y-axis'
    self.canvases_data[2]['name'] = 'Accelerometer z-axis'
    self.canvases_data[3]['name'] = 'Gyro x-axis'
    self.canvases_data[4]['name'] = 'Gyro y-axis'
    self.canvases_data[5]['name'] = 'Gyro z-axis'
    self.canvases_data[6]['name'] = 'Magnetometer x-axis'
    self.canvases_data[7]['name'] = 'Magnetometer y-axis'
    self.canvases_data[8]['name'] = 'Magnetometer z-axis'

else:
    self.options_layout.setCurrentIndex(4)
    self._set_frequencies(['1', '2', '5'])
    self.stream_code = 6
    self.package_structure = [2] * 4
    self.parameters_indexes = []
    self.parameters_outputs = []
    self.number_of_plots = 4
    self.canvases_data[0]['name'] = 'Object Temperature'
    self.canvases_data[1]['name'] = 'Ambient Temperature'
    self.canvases_data[2]['name'] = 'Environment Temperature'
```

```
            self.canvases_data[3]['name'] = 'Atmospheric Pressure'

        self._set_plots()

    # set queues for window
    def set_queues(self, data_queue, log_queue):
        self.data_queue = data_queue
        self.log_queue = log_queue

    # clean up before new plotting
    def reset_window(self, clear_plots=True, clear_messages=True):
        self.duration = self._get_entry_value(self.duration_input, 100, 10, 600, 'duration', False)
        self.last_update_time = time.time()
        self.last_package_time = time.time()
        self.last_mode = str(self.mode_choice.currentText())
        self.initial_frequency = int(self.frequency_options.currentText())
        array = [0] * (self.duration * self.initial_frequency + 50)
        self.graphs_points = []
        for i in range(0, 12):
            self.graphs_points.append(array.copy())
        self.indexes = [0] * 12
        self.buffer = []
        self.just_started = True
        self.timeout = False
        if clear_plots:
            self._clear_plots()
        if clear_messages:
            self.messages.setPlainText('')

    # GETTERS

    # get values of parameters for stream
    def get_configuration(self):
        # get common parameters
        frequency = self.initial_frequency
        duration = self.duration

        # get optional parameters

        # PPG
        green = int(self.green_led.isChecked())
        red = int(self.red_led.isChecked())
        infro = int(self.infro_led.isChecked())
        led_config = 4 * infro + 2 * red + green
        current_led = self._get_entry_value(self.led_input, 0, 0, 255, 'current LED')

        # IMPEDANCE
        stimul_frequency = self._get_entry_value(self.stimulation_frequency_input, 0, 0, 1e5,
'stimulation frequency')
        stimul_current = self._get_entry_value(self.stimulation_current_input, 0, 100, 1e5, 'stimulation
current')
```

```
        vref_impedance = self._get_entry_value(self.ref_voltage_impedance_input, 0, 1, 3300,
'reference voltage')
        gain_impedance = int(self.gain_voltage_impedance_options.currentText())

        # ECG
        vref_ecg = self._get_entry_value(self.ref_voltage_ecg_input, 0, 1, 3300, 'reference voltage')
        gain_ecg = int(self.gain_voltage_ecg_options.currentText())

        # BODY TEMP
        if str(self.body_scales.currentText()) == 'Celsius':
            scale = 0
        else:
            scale = 1

        parameters = {'frequency': frequency, 'duration': duration, 'LED': led_config, 'current LED':
current_led, 'stimul_frequency': stimul_frequency,
                      'stimul_current': stimul_current, 'Vref_impedance': vref_impedance,
'Gain_impedance': gain_impedance, 'Vref_ECG': vref_ecg,
                      'Gain_ECG': gain_ecg, 'scale': scale}
        return parameters

    # this function gets value of text entry
    def _get_entry_value(self, entry, default_value, min_value=0, max_value=1e6, name='',
special_zero=True, form='int'):
        try:
            # we try to get number from entry
            if form == 'int':
                value = int(entry.text())
            else:
                value = float(entry.text())

            # if number was get nicely, we check it
            if not (special_zero & (value == 0)):
                if value < min_value:
                    value = min_value
                    self.messages.appendPlainText('Given %s value is too small. Least value is set.\n' %
name)

                elif value > max_value:
                    value = max_value
                    self.messages.appendPlainText('Given %s value is too big. Biggest value is set.\n' %
name)

        except ValueError:
            # if string is not a number, we process it
            self.messages.appendPlainText('Invalid %s. Value set to default.\n' % name)
            self.stop_call = True
            value = default_value

        entry.setText(str(value))

        return value
```

```
    # gets chunk of packages
    def _get_chunk(self):
        # get new package chunk from queue
        while 1:
            if time.time() - self.last_package_time >= 5:
                self.timeout = True

            try:
                self.buffer += self.data_queue.get(0)
                if self.buffer:
                    self.last_package_time = time.time()
            except queue.Empty:
                break

    # this function transforms sequence of bytes in parsed list-like package
    def _parse_package(self, package):

        structure = self.package_structure
        # useful counters
        n = len(package)
        parsed_package = []

        # checking, if package is valid
        last = package[n-2:].hex()
        first = package[0:1].hex()
        second = package[1] >> 4
        if (first == b'\xAA'.hex()) & (last == b'\x55\xAA'.hex()) & (second == 5):

            timestamp = ((package[1] % 16) * 65536 + int.from_bytes(package[2:4], byteorder='big')) /
self.initial_frequency
            parsed_package.append(timestamp)

            index = 4

            # if package is valid, we read it's data in a way how structure tells
            for i in range(0, len(structure)):
                current_data = b''
                for j in range(0, structure[i]):
                    current_data += package[(index + j):(index+j+1)]
                index += structure[i]
                current_data_value = int.from_bytes(current_data, byteorder='big')
                parsed_package.append(current_data_value)

            if parsed_package:

                for i in range(0, len(parsed_package)):
                    if self.indexes[i] < len(self.graphs_points[i]):
                        self.graphs_points[i][self.indexes[i]] = parsed_package[i]
                        self.indexes[i] += 1

                if self.logging_checkbox.isChecked():
```

```
                self.log_queue.put(parsed_package)

    # put new plots
    def _set_plots(self):

        duration = self.duration
        number_of_plots = self.number_of_plots

        if self.vertical_plotting_layout:
            self._clear_layout(self.vertical_plotting_layout)

        self.scrollAreaWidgetContents.setGeometry(QtCore.QRect(0, 0, int(0.82 * self.screen_width),
number_of_plots*int(self.screen_height / 2.15)))

        pg.setConfigOption('background', 'w')
        pg.setConfigOption('foreground', 'k')

        self.canvases = []
        for i in range(0, number_of_plots):
            canvas = Canvas(self.central_widget)

            canvas.plot.setXRange(0, duration, padding=0.01)
            canvas.plot.setTitle(self.canvases_data[i]['name'])
            canvas.plot.plot(pen=(i, 1))
            canvas.plot.showGrid(True, True, 0.2)
            canvas.entry.setFixedWidth(int(0.025*self.screen_width))
            canvas.button.setFixedWidth(int(0.025*self.screen_width))

            self.vertical_plotting_layout.addWidget(canvas.plot)
            self.canvases.append(canvas)

    # clear plots from data
    def _clear_plots(self):
        for i in range(0, len(self.canvases)):
            self.canvases[i].plot.clear()
            self.canvases[i].got_duration = False
            self.canvases[i].plot.setXRange(0, self.duration, padding=0.01)

    # update plots with new data
    def _update_plots(self):

        for i in range(0, len(self.canvases)):
            canvas = self.canvases[i]
            plot = canvas.plot
            plot.clear()
            x_points = self.graphs_points[0][:self.indexes[0]]
            y_points = self.graphs_points[i+1][:self.indexes[i+1]]
            lines = MultiLine(x_points, y_points, (i, len(self.canvases)))
            plot.addItem(lines)
            n = len(x_points)
            if canvas.is_tracking & n:
                last_x = x_points[n-1]
```

```
            if canvas.update_duration:
                canvas.duration = self._get_entry_value(canvas.entry, 10, 0.1, 30, 'local duration of
'+self.canvases_data[i]['name'], False, 'float')
                canvas.update_duration = False

            xmin = int(last_x / canvas.duration) * canvas.duration
            canvas.plot.setXRange(xmin, xmin + canvas.duration, padding=0.01)

            last_index = n-1
            ymin = y_points[last_index]
            ymax = y_points[last_index]

            while (last_index >= 0) & (x_points[last_index] >= xmin):
                last_index -= 1
                if last_index >= 0:
                    if y_points[last_index] > ymax:
                        ymax = y_points[last_index]

                    if ymin > y_points[last_index]:
                        ymin = y_points[last_index]

            canvas.plot.setYRange(ymin-1, ymax+1, padding=0.01)

    def draw_plots(self):

        while (len(self.buffer) > 0) & (not self.stop_signal):
            self._parse_package(self.buffer[0])
            del self.buffer[0]

            if time.time() - self.last_update_time >= 5:
                self._update_message_windows()
                self.last_update_time = time.time()

            if self.just_started:
                self.just_started = False
                self._update_message_windows()

        self._update_plots()

        if self.stop_signal:
            self.reset_window(False, False)
        else:
            self._get_chunk()
            QTimer.singleShot(1, lambda: self.draw_plots())

    def _update_message_windows(self):
        for i in range(0, len(self.parameters_indexes)):

            parameter_value =
self.graphs_points[self.parameters_indexes[i]][self.indexes[self.parameters_indexes[i]]-1]

            self.parameters_outputs[i].setText(str(parameter_value))
```

```
# class that manages the process
class DataPlotter:

    def __init__(self):
        # shared queue
        self.data_queue = Queue()
        self.log_queue = Queue()

        # initializing device communication
        self.device = DeviceCommunicator(data_queue=self.data_queue)

        # time tracking variable
        self.start_time = 0

        # get list of ports
        available_ports = self.device.serial_ports()

        # plotting frame
        self.window = Window(available_ports, self.data_queue, self.log_queue)

        # making start and stop functions
        self.window.start_button.clicked.connect(self._start)
        self.window.stop_button.clicked.connect(lambda: self._stop())
        self.window.status_button.clicked.connect(self._status_check)

        # logging object
        self.log = None
        self.end = False
        self.window.MainWindow.show()
        self._periodic_call()

    # get chosen device port
    def _get_port(self):
        return str(self.window.device_ports.currentText())

    # function, which tracks status of process
    def _periodic_call(self):
        passed_time = time.time() - self.start_time
        if (passed_time > self.window.duration+0.05) & (not self.window.stop_signal):
            self.window.messages.appendPlainText('Session ended.')
            self._stop(False)
        elif not self.window.stop_signal:
            self.window.duration_input.setText(str(int(self.window.duration - passed_time)))

        if (self.device.timeout | self.window.timeout) & (not self.window.stop_signal):
            self.window.messages.appendPlainText('5-sec timeout reached.')
            self._stop(False)

        if (self.window.last_mode != str(self.window.mode_choice.currentText())) & (not
self.window.stop_signal):
```

```
                self.device.stop_stream()
                self.window.messages.appendPlainText('Unexpected mode change.')
                self.window.last_mode = str(self.window.mode_choice.currentText())
                self._stop()
                self.window.reset_window(True, False)

        if self.window.MainWindow.exit_call:
            self._close_window()

        if not self.end:
            QTimer.singleShot(1, lambda: self._periodic_call())

    # function for clearing queues between different plotting sessions
    def _clear_queues(self):
        while not self.data_queue.empty():
            try:
                package = self.data_queue.get()
            except queue.Empty:
                break

        while not self.log_queue.empty():
            try:
                package = self.log_queue.get()
            except queue.Empty:
                break

    # get specified device status
    def _get_status(self, command_code, structure):

        parsed_status = []

        if self.device.port_is_open:

            status = self.device.check_status(command_code)

            if status != 'timeout':
                index = 5

                for i in range(0, len(structure)):
                    current_data = status[index:(index+structure[i])]
                    index += structure[i]
                    current_data_value = int.from_bytes(current_data, byteorder='big')
                    parsed_status.append(current_data_value)

                crc = status[-3:-2]
                encoded_command = command_code.to_bytes(3, byteorder='big')
                encoded_parameters = b''
                for i in range(0, len(structure)):
                    encoded_parameters += parsed_status[i].to_bytes(structure[i], byteorder='big')
                encoded_parameters = int.from_bytes(encoded_parameters,
byteorder='little').to_bytes(32, byteorder='little')
```

```
            if crc != (crc8().crc(encoded_command+encoded_parameters)).to_bytes(1,
byteorder='big'):
                answer = 'Device response is damaged. Please check connection.'
                parsed_status = []

            elif command_code == 7:
                answer = 'Device charge is %s percent(s).' % str(parsed_status[0])

            elif command_code == 8:
                answer = ('Current ESI level is %s' % (str(parsed_status[0]))) + (', contact status is %s.'
% str(parsed_status[1]))

            else:
                try:
                    device_type = {1: 'PPG', 2: 'Impedance', 3: 'ECG', 4: 'Motion', 5: 'Body
temp'}[parsed_status[1]]
                except:
                    device_type = 'unknown'

                answer = ('Device ID is %s.' % str(parsed_status[0])) + '\nDevice type: ' + device_type
+ ('\nFirmware version: %s' % str(parsed_status[2]))

        else:
            answer = '5-sec timeout is reached. Please, check device connection.'
    else:
        answer = 'Device port failed. Please, check connection and try again.'

    return {'answer': answer, 'status': parsed_status}

# check specified device parameter
def _status_check(self, need_return=False):

    if self.window.stop_signal:
        self.window.messages.setPlainText('')
        QApplication.processEvents()
        if not self.device.port_is_open:
            self.device.port_configure(port=self._get_port(), baudrate=115200)
            self.device.port_is_open = True

        variant = str(self.window.status_options.currentText())
        if variant == 'battery charge':
            command_code = 7
            structure = [1]
        elif variant == 'contact status':
            command_code = 8
            structure = [2, 1]
        else:
            command_code = 9
            structure = [12, 1, 1]

        self.window.messages.setPlainText(self._get_status(command_code, structure)['answer'])
```

```
        else:
            self.window.messages.appendPlainText('Session in process detected. Status check is
canceled.')

    # function that starts data receiving and plotting
    def _start(self):
        if not self.window.stop_signal:
            return

        # configure port, if stream starting first time in session
        if not self.device.port_is_open:
            port = self._get_port()
            if port == '---':
                self.window.messages.setPlainText('Invalid port. Please, check device connection and
choose proper device port.')
            else:
                self.device.port_configure(port=port, baudrate=115200)
                self.device.port_is_open = True

        if self.device.port_is_open:
            self.window.messages.setPlainText('')
            self.window.messages.appendPlainText('Checking device ID...')
            QApplication.processEvents()
            device_id = self._get_status(9, [12, 1, 1])
            self.window.messages.appendPlainText(device_id['answer'])

            if device_id['status']:
                self.window.messages.appendPlainText('Device connected properly.')
                self.window.messages.appendPlainText('Checking device charge...')
                QApplication.processEvents()
                device_charge = self._get_status(7, [1])
                self.window.messages.appendPlainText(device_charge['answer'])

                if device_charge['status']:
                    if device_charge['status'][0] < 10:
                        self.window.messages.appendPlainText('Current charge is too low. Please, charge
device before data streaming.')
                        QApplication.processEvents()
                    else:
                        self.window.messages.appendPlainText('Device charge is nominal.')
                        self.window.messages.appendPlainText('Checking skin contact...')
                        QApplication.processEvents()
                        device_contact = self._get_status(8, [2, 1])
                        self.window.messages.appendPlainText(device_contact['answer'])

                        if device_contact['status']:
                            next_step = not device_contact['status'][1]
                        else:
                            next_step = False

                        if next_step:
```

```
                    self.window.messages.appendPlainText("Contact with skin wasn't detected.
Please, put device on.")

                else:

                    if self.window.stop_signal:
                        self.start_time = time.time()
                        self.window.stop_signal = False

                        # reset window parameters
                        self.window.reset_window(True, False)

                        # start stream and data plotting
                        self.device.start_stream(command_code=self.window.stream_code,
parameters=self.window.get_configuration())

                        if self.window.stop_call:
                            self._stop()
                            self.window.stop_call = False
                        else:
                            if self.window.logging_checkbox.isChecked():
                                self.window.messages.appendPlainText('Creating log file...')
                                QApplication.processEvents()
                                self.log = Logger()
                                self.log.set_logging(self.log_queue,
str(self.window.log_name_input.text()))
                                self.log.start_logging()
                            self.window.messages.appendPlainText('Plotting started.')
                            self.start_time = time.time()
                            self.window.duration_label.setText('ETA:')
                            self.window.draw_plots()

    # function that stops plotting and data receiving
    def _stop(self, write=True):

        # stop window
        self.window.stop_signal = True
        self.window.messages.appendPlainText('Plotting stopped.')
        self.window.duration_label.setText('Duration:')
        self.window.duration_input.setText(str(self.window.duration))

        # stop device
        if self.device.port_is_open:
            self.device.stop_stream(write)

        # stop log
        if self.log:
            self.log.stop = True
            time.sleep(0.01)
            self.log = None

        time.sleep(0.01)
```

```
        self._clear_queues()

    # function that closes and stops everything
    def _close_window(self):
        self.device.end()
        self._stop()
        if self.device.port_is_open:
            self.device.stop_stream()
            self.device.port.close()
        self.window.MainWindow.close()
        self.end = True


# now we can run simulation of data receiving
if __name__ == '__main__':
    app = QApplication(sys.argv)
    client = DataPlotter()
    sys.exit(app.exec_())
```

Додаток Е

Проект програмного забезпечення в середовищі Matlab для моделювання спайкового кодування ЕКГ-сигналів та симуляції поведінки нейронної мережі

```
Holter_filename = './Holter/RAWdata/ID05099/060219173002.csv';
CSV = csvread(Holter_filename);
ecg = CSV(1:length(CSV), 2);
fs = 128;
mode = 1;
[qrs_amp_raw,qrs_i_raw,delay,ecg_filter] = pan_tompkin(ecg,fs,mode);
ecg_filter = ecg/max(ecg);
L = length(ecg_filter);
Uthr = 0;
Lthr = 0;
go_up = 1;
j = 0;
k = 0;
dur = 0;
n = 1;
max_value = 0;
max_idx = 0;
Uthr = ecg_filter(1);
state = 0;
spikes_level(1:L) = 0;
trig = 0;
peak_num = 0;
for i=2:L
   spikes(i) = 0;
   qrs(i) = 0;
   if(trig == 1)
      spikes_level(i) = 1;
   end
   if(state == 0)
      if(ecg_filter(i) > Uthr)
         Uthr = ecg_filter(i);
         state = 1;
      end
   elseif(state == 1)
      if(ecg_filter(i) < Uthr)
         state = 2;
      else
         Uthr = ecg_filter(i);
         if(ecg_filter(i) > ecg_filter(i-1) && ecg_filter(i) > 0.03)
            k = k + 1;
         else
            k = 0;
         end
         if(k>=0)
            if(spikes(i-1) == 0)
```

```
                spikes(i) = 1;
            else
                spikes(i) = 0;
            end
        end
        if(k>2)
            trig = 1;
        end
    end
    if(ecg_filter(i) > max_value)
        max_value = ecg_filter(i);
        max_idx = i;
    end
elseif(state == 2)
    if(ecg_filter(i) + 0.1 < Uthr)
        qrs_p(n) = max_idx;
        max_value = 0;
        max_idx = 0;
        n = n + 1;
        k = 0;
        state = 3;
    else
        state = 1;
    end
elseif(state == 3)
    j = j + 1;
    if(j > 5)
        j = 0;
        trig = 0;
        state = 0;
        Uthr = 0;
    end
end
end
figure;
% subplot(4,2,1);
% plot(ecg);
% title('ECG signal');
% xlabel('samples');
% ylabel('uV');
% xlim([5150 5480]);
subplot(2,1,1);
plot(ecg_filter);
title('ECG with Peaks Timestamps');
line(repmat(qrs_p,[2 1]),...
    repmat([min(ecg_filter)/2; max(ecg_filter)],size(qrs_p)),...
    'LineWidth',1,'LineStyle','-.','Color','r');
xlabel('smaples');
ylabel('a.u.');
xlim([5150 5480]);
subplot(2,1,2);
plot(spikes);
```

```
title('Spikes Train');
xlabel('samples');
ylabel('a.u.');
xlim([5150 5480]);

%% spiking neural network simulation routine
% Excitatory neurons Inhibitory neurons
Ne=800; Ni=200;
re=rand(Ne,1); ri=rand(Ni,1);
a=[0.02*ones(Ne,1); 0.02+0.08*ri];
b=[0.2*ones(Ne,1); 0.25-0.05*ri];
c=[-65+15*re.^2; -65*ones(Ni,1)];
d=[8-6*re.^2; 2*ones(Ni,1)];
S=[0.53*rand(Ne+Ni,Ne), -rand(Ne+Ni,Ni)];
v=-65*ones(Ne+Ni,1);    % Initial values of v
u=b.*v;                 % Initial values of u
firings=[];             % spike timings
f = 0;
spikes_num = 0;
for t=1:5000         % simulation of 10000 ms
I=[5*randn(Ne,1);2*randn(Ni,1)]; % thalamic input
fired=find(v>=30);                    % indices of spikes
L = length(fired);
spikes_num = spikes_num + L;
   if(t>0 && mod(t,1000)==0)
      f = f + 1;
      Freq(f) = spikes_num/1000;
      spikes_num = 0;
   end
firings=[firings; t+0*fired,fired];
v(fired)=c(fired);
u(fired)=u(fired)+d(fired);
I=I+sum(S(:,fired),2);
v=v+0.5*(0.04*v.^2+5*v+140-u+I); % step 0.5 ms
v=v+0.5*(0.04*v.^2+5*v+140-u+I); % for numerical
u=u+a.*(b.*v-u); % stability
end;
plot(firings(:,1)*8/1000,firings(:,2),'.');
ylabel('Íîìåð íåéðîíà');
xlabel('×àñ, ñåê');
xlim([3 13]);
figure;
plot(Freq-5);
ylabel('SNN frequency, Hz');
xlabel('Time, s');
xlim([3 50]);
```

Додаток Є

Проект програмного забезпечення в середовищі Matlab для фільтрації та детектування QRS-комплексів ЕКГ

```
function [qrs_amp_raw,qrs_i_raw,delay,ecg_filter]=pan_tompkin(ecg,fs,gr)

%% function [qrs_amp_raw,qrs_i_raw,delay]=pan_tompkin(ecg,fs)
% Complete implementation of Pan-Tompkins algorithm

%% Inputs
% ecg : raw ecg vector signal 1d signal
% fs : sampling frequency e.g. 200Hz, 400Hz and etc
% gr : flag to plot or not plot (set it 1 to have a plot or set it zero not
% to see any plots
%% Outputs
% qrs_amp_raw : amplitude of R waves amplitudes
% qrs_i_raw : index of R waves
% delay : number of samples which the signal is delayed due to the
% filtering
%% Method
% See Ref and supporting documents on researchgate.
%
https://www.researchgate.net/publication/313673153_Matlab_Implementation_of_Pan_Tompkins_
ECG_QRS_detector
%% References :
%[1] Sedghamiz. H, "Matlab Implementation of Pan Tompkins ECG QRS
%detector.",2014. (See researchgate)
%[2] PAN.J, TOMPKINS. W.J,"A Real-Time QRS Detection Algorithm" IEEE
%TRANSACTIONS ON BIOMEDICAL ENGINEERING, VOL. BME-32, NO. 3, MARCH 1985.

%% ============== Licensce ========================================== %%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS
% "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
% LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
% FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT
% OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
% SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED
% TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
OR
% PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF
% LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
% NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
% SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

```
%% =========== Update History ================== %%
% Feb 2018 :
%          1- Cleaned up the code and added more comments
%          2- Added to BioSigKit Toolbox

%% ================ Now Part of BioSigKit ==================== %%
if ~isvector(ecg)
  error('ecg must be a row or column vector');
end
if nargin < 3
    gr = 1;   % on default the function always plots
end
ecg = ecg(:); % vectorize

%% ====================== Initialize =============================== %
delay = 0;
skip = 0;                                          % becomes one when a T wave is detected
m_selected_RR = 0;
mean_RR = 0;
ser_back = 0;
ax = zeros(1,6);

%% ============ Noise cancelation(Filtering)( 5-15 Hz) =============== %%
if fs == 200
% ----------------- remove the mean of Signal -----------------------%
  ecg = ecg - mean(ecg);
%% ==== Low Pass Filter  H(z) = ((1 - z^(-6))^2)/(1 - z^(-1))^2 ==== %%
%%It has come to my attention the original filter doesnt achieve 12 Hz
%    b = [1 0 0 0 0 0 -2 0 0 0 0 0 1];
%    a = [1 -2 1];
%    ecg_l = filter(b,a,ecg);
%    delay = 6;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
   Wn = 12*2/fs;
   N = 3;                                                  % order of 3 less processing
   [a,b] = butter(N,Wn,'low');                             % bandpass filtering
   ecg_l = filtfilt(a,b,ecg);
   ecg_l = ecg_l/ max(abs(ecg_l));

 %% ====================== start figure ============================= %%
   if gr
    figure;
    ax(1) = subplot(321);plot(ecg);axis tight;title('Raw signal');ylabel('uV');xlabel('samples');
    ax(2)=subplot(322);plot(ecg_l);axis tight;title('Low pass filtered'); ylabel('a.u.');xlabel('samples');
   end

%% ==== High Pass filter H(z) = (-1+32z^(-16)+z^(-32))/(1+z^(-1)) ==== %%
%%It has come to my attention the original filter doesn achieve 5 Hz
%    b = zeros(1,33);
%    b(1) = -1; b(17) = 32; b(33) = 1;
```

```
%    a = [1 1];
%    ecg_h = filter(b,a,ecg_l);    % Without Delay
%    delay = delay + 16;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
   Wn = 5*2/fs;
   N = 3;                                              % order of 3 less processing
   [a,b] = butter(N,Wn,'high');                        % bandpass filtering
   ecg_h = filtfilt(a,b,ecg_l);
   ecg_h = ecg_h/ max(abs(ecg_h));
   if gr
    ax(3)=subplot(323);plot(ecg_h);axis tight;title('High Pass
Filtered');ylabel('a.u.');xlabel('samples');
   end
else
%%  bandpass filter for Noise cancelation of other sampling frequencies(Filtering)
 f1=5;                                        % cuttoff low frequency to get rid of baseline
wander
 f2=15;                                        % cuttoff frequency to discard high frequency
noise
 Wn=[f1 f2]*2/fs;                              % cutt off based on fs
 N = 3;                                        % order of 3 less processing
 [a,b] = butter(N,Wn);                         % bandpass filtering
 ecg_h = filtfilt(a,b,ecg);
 ecg_h = ecg_h/ max( abs(ecg_h));
 if gr
  ax(1) = subplot(3,2,[1 2]);plot(ecg);axis tight;title('Raw Signal');ylabel('uV');xlabel('samples');
  ax(3)=subplot(323);plot(ecg_h);axis tight;title('Band Pass Filtered');ylabel('a.u.');xlabel('samples');
 end
end
%% ==================== derivative filter ========================== %%
% ------ H(z) = (1/8T)(-z^(-2) - 2z^(-1) + 2z + z^(2)) --------- %
if fs ~= 200
 int_c = (5-1)/(fs*1/40);
 b = interp1(1:5,[1 2 0 -2 -1].*(1/8)*fs,1:int_c:5);
else
 b = [1 2 0 -2 -1].*(1/8)*fs;
end

 ecg_d = filtfilt(b,1,ecg_h);
 ecg_d = ecg_d/max(ecg_d);

 if gr
  ax(4)=subplot(324);plot(ecg_d);
  axis tight;
  title('Filtered with the derivative filter');
  ylabel('a.u.');xlabel('samples');
 end

%% ========== Squaring nonlinearly enhance the dominant peaks ========== %%
 ecg_s = ecg_d.^2;
 if gr
```

```
 ax(5)=subplot(325);
 plot(ecg_s);
 axis tight;
 title('Squared');
 ylabel('a.u.');xlabel('samples');
end

%% ============  Moving average ================== %%
%-------Y(nt) = (1/N)[x(nT-(N - 1)T)+ x(nT - (N - 2)T)+...+x(nT)]---------%
ecg_m = conv(ecg_s ,ones(1 ,round(0.150*fs))/round(0.150*fs));
delay = delay + round(0.150*fs)/2;

 if gr
 ax(6)=subplot(326);plot(ecg_m);
 axis tight;
 title('Averaged with 30 samples length,Black noise,Green Adaptive Threshold,RED Sig Level,Red
circles QRS adaptive threshold');
 axis tight;
 ylabel('a.u.');xlabel('samples');
end
ecg_filter = ecg_m;
%% ====================== Fiducial Marks ============================== %%
% Note : a minimum distance of 40 samples is considered between each R wave
% since in physiological point of view no RR wave can occur in less than
% 200 msec distance
[pks,locs] = findpeaks(ecg_m,'MINPEAKDISTANCE',round(0.2*fs));
%% ==================== Initialize Some Other Parameters =============== %%
LLp = length(pks);
% ---------------- Stores QRS wrt Sig and Filtered Sig ------------------%
qrs_c = zeros(1,LLp);          % amplitude of R
qrs_i = zeros(1,LLp);          % index
qrs_i_raw = zeros(1,LLp);      % amplitude of R
qrs_amp_raw= zeros(1,LLp);     % Index
% ------------------- Noise Buffers ---------------------------------%
nois_c = zeros(1,LLp);
nois_i = zeros(1,LLp);
% ------------------- Buffers for Signal and Noise ----------------- %
SIGL_buf = zeros(1,LLp);
NOISL_buf = zeros(1,LLp);
SIGL_buf1 = zeros(1,LLp);
NOISL_buf1 = zeros(1,LLp);
THRS_buf1 = zeros(1,LLp);
THRS_buf = zeros(1,LLp);


%% initialize the training phase (2 seconds of the signal) to determine the THR_SIG and
THR_NOISE
THR_SIG = max(ecg_m(1:2*fs))*1/3;                    % 0.25 of the max amplitude
THR_NOISE = mean(ecg_m(1:2*fs))*1/2;                 % 0.5 of the mean signal is
considered to be noise
SIG_LEV= THR_SIG;
NOISE_LEV = THR_NOISE;
```

```
%% Initialize bandpath filter threshold(2 seconds of the bandpass signal)
THR_SIG1 = max(ecg_h(1:2*fs))*1/3;                          % 0.25 of the max amplitude
THR_NOISE1 = mean(ecg_h(1:2*fs))*1/2;
SIG_LEV1 = THR_SIG1;                                        % Signal level in Bandpassed filter
NOISE_LEV1 = THR_NOISE1;                                    % Noise level in Bandpassed filter
%% ============ Thresholding and desicion rule ============= %%
Beat_C = 0;                                                 % Raw Beats
Beat_C1 = 0;                                                % Filtered Beats
Noise_Count = 0;                                            % Noise Counter
for i = 1 : LLp
  %% ===== locate the corresponding peak in the filtered signal === %%
   if locs(i)-round(0.150*fs)>= 1 && locs(i)<= length(ecg_h)
       [y_i,x_i] = max(ecg_h(locs(i)-round(0.150*fs):locs(i)));
     else
       if i == 1
        [y_i,x_i] = max(ecg_h(1:locs(i)));
        ser_back = 1;
       elseif locs(i)>= length(ecg_h)
        [y_i,x_i] = max(ecg_h(locs(i)-round(0.150*fs):end));
       end
   end
 %% ================= update the heart_rate ==================== %%
   if Beat_C >= 9
      diffRR = diff(qrs_i(Beat_C-8:Beat_C));                % calculate RR interval
      mean_RR = mean(diffRR);                               % calculate the mean of 8 previous R
waves interval
      comp =qrs_i(Beat_C)-qrs_i(Beat_C-1);                  % latest RR

      if comp <= 0.92*mean_RR || comp >= 1.16*mean_RR
    % ------ lower down thresholds to detect better in MVI -------- %
           THR_SIG = 0.5*(THR_SIG);
           THR_SIG1 = 0.5*(THR_SIG1);
      else
         m_selected_RR = mean_RR;                           % The latest regular beats mean
      end

   end

 %% == calculate the mean last 8 R waves to ensure that QRS is not ==== %%
     if m_selected_RR
        test_m = m_selected_RR;                             %if the regular RR availabe use it
     elseif mean_RR && m_selected_RR == 0
        test_m = mean_RR;
     else
        test_m = 0;
     end

   if test_m
      if (locs(i) - qrs_i(Beat_C)) >= round(1.66*test_m)    % it shows a QRS is missed
```

```
            [pks_temp,locs_temp] = max(ecg_m(qrs_i(Beat_C)+ round(0.200*fs):locs(i)-
round(0.200*fs))); % search back and locate the max in this interval
            locs_temp = qrs_i(Beat_C)+ round(0.200*fs) + locs_temp -1;      % location

            if pks_temp > THR_NOISE
             Beat_C = Beat_C + 1;
             qrs_c(Beat_C) = pks_temp;
             qrs_i(Beat_C) = locs_temp;
            % ------------ Locate in Filtered Sig ------------ %
             if locs_temp <= length(ecg_h)
               [y_i_t,x_i_t] = max(ecg_h(locs_temp-round(0.150*fs):locs_temp));
             else
               [y_i_t,x_i_t] = max(ecg_h(locs_temp-round(0.150*fs):end));
             end
            % ----------- Band pass Sig Threshold ------------------%
             if y_i_t > THR_NOISE1
               Beat_C1 = Beat_C1 + 1;
               qrs_i_raw(Beat_C1) = locs_temp-round(0.150*fs)+ (x_i_t - 1);% save index of bandpass
               qrs_amp_raw(Beat_C1) = y_i_t;                          % save amplitude of bandpass
               SIG_LEV1 = 0.25*y_i_t + 0.75*SIG_LEV1;                 % when found with the
second thres
             end

             not_nois = 1;
             SIG_LEV = 0.25*pks_temp + 0.75*SIG_LEV ;                 % when found with the
second threshold
           end
         else
            not_nois = 0;
         end
    end

    %% ====================  find noise and QRS peaks ================== %%
    if pks(i) >= THR_SIG
      % ------ if No QRS in 360ms of the previous QRS See if T wave ------%
       if Beat_C >= 3
          if (locs(i)-qrs_i(Beat_C)) <= round(0.3600*fs)
             Slope1 = mean(diff(ecg_m(locs(i)-round(0.075*fs):locs(i))));       % mean slope of the
waveform at that position
             Slope2 = mean(diff(ecg_m(qrs_i(Beat_C)-round(0.075*fs):qrs_i(Beat_C)))); % mean
slope of previous R wave
             if abs(Slope1) <= abs(0.5*(Slope2))                        % slope less then 0.5 of previous R
               Noise_Count = Noise_Count + 1;
               nois_c(Noise_Count) = pks(i);
               nois_i(Noise_Count) = locs(i);
               skip = 1;                                  % T wave identification
               % ----- adjust noise levels ------ %
               NOISE_LEV1 = 0.125*y_i + 0.875*NOISE_LEV1;
               NOISE_LEV = 0.125*pks(i) + 0.875*NOISE_LEV;
             else
               skip = 0;
             end
```

```
        end
      end
      %---------- skip is 1 when a T wave is detected ---------- %
      if skip == 0
        Beat_C = Beat_C + 1;
        qrs_c(Beat_C) = pks(i);
        qrs_i(Beat_C) = locs(i);

      %--------------- bandpass filter check threshold --------------- %
        if y_i >= THR_SIG1
           Beat_C1 = Beat_C1 + 1;
           if ser_back
             qrs_i_raw(Beat_C1) = x_i;                          % save index of bandpass
           else
             qrs_i_raw(Beat_C1)= locs(i)-round(0.150*fs)+ (x_i - 1);   % save index of bandpass
           end
           qrs_amp_raw(Beat_C1) =  y_i;                        % save amplitude of bandpass
           SIG_LEV1 = 0.125*y_i + 0.875*SIG_LEV1;                  % adjust threshold for
bandpass filtered sig
        end
       SIG_LEV = 0.125*pks(i) + 0.875*SIG_LEV ;                  % adjust Signal level
      end

   elseif (THR_NOISE <= pks(i)) && (pks(i) < THR_SIG)
       NOISE_LEV1 = 0.125*y_i + 0.875*NOISE_LEV1;          % adjust Noise level in filtered sig
       NOISE_LEV = 0.125*pks(i) + 0.875*NOISE_LEV;         % adjust Noise level in MVI
   elseif pks(i) < THR_NOISE
      Noise_Count = Noise_Count + 1;
      nois_c(Noise_Count) = pks(i);
      nois_i(Noise_Count) = locs(i);
      NOISE_LEV1 = 0.125*y_i + 0.875*NOISE_LEV1;              % noise level in filtered signal
      NOISE_LEV = 0.125*pks(i) + 0.875*NOISE_LEV;             % adjust Noise level in MVI
   end

   %% ================== adjust the threshold with SNR ============= %%
   if NOISE_LEV ~= 0 || SIG_LEV ~= 0
      THR_SIG = NOISE_LEV + 0.25*(abs(SIG_LEV - NOISE_LEV));
      THR_NOISE = 0.5*(THR_SIG);
   end

   %------ adjust the threshold with SNR for bandpassed signal -------- %
   if NOISE_LEV1 ~= 0 || SIG_LEV1 ~= 0
      THR_SIG1 = NOISE_LEV1 + 0.25*(abs(SIG_LEV1 - NOISE_LEV1));
      THR_NOISE1 = 0.5*(THR_SIG1);
   end


%--------- take a track of thresholds of smoothed signal ----------%
SIGL_buf(i) = SIG_LEV;
NOISL_buf(i) = NOISE_LEV;
THRS_buf(i) = THR_SIG;
```

```
%-------- take a track of thresholds of filtered signal ----------- %
SIGL_buf1(i) = SIG_LEV1;
NOISL_buf1(i) = NOISE_LEV1;
THRS_buf1(i) = THR_SIG1;
% ---------------------- reset parameters ------------------------- %
skip = 0;
not_nois = 0;
ser_back = 0;
end

%% ======================= Adjust Lengths ============================== %%
qrs_i_raw = qrs_i_raw(1:Beat_C1);
qrs_amp_raw = qrs_amp_raw(1:Beat_C1);
qrs_c = qrs_c(1:Beat_C);
qrs_i = qrs_i(1:Beat_C);

%% ======================= Plottings ================================== %%
if gr
  hold on,scatter(qrs_i,qrs_c,'m');
  hold on,plot(locs,NOISL_buf,'--k','LineWidth',2);
  hold on,plot(locs,SIGL_buf,'--r','LineWidth',2);
  hold on,plot(locs,THRS_buf,'--g','LineWidth',2);
 if any(ax)
  ax(~ax) = [];
  linkaxes(ax,'x');
  zoom on;
 end
end


%% ================== overlay on the signals ========================== %%
 if gr
   figure;
   axes('Units','normalized','Position',[0 0 1 1]);
   az(1)=subplot(311);
   ylabel('a.u.');xlabel('samples');
   plot(ecg_h);
   title('QRS on Filtered Signal');
   axis tight;
   hold on,scatter(qrs_i_raw,qrs_amp_raw,'m');
   hold on,plot(locs,NOISL_buf1,'LineWidth',2,'Linestyle','--','color','k');
   hold on,plot(locs,SIGL_buf1,'LineWidth',2,'Linestyle','-.','color','r');
   hold on,plot(locs,THRS_buf1,'LineWidth',2,'Linestyle','-.','color','g');
   az(2)=subplot(312);plot(ecg_m);
   title('QRS on MVI signal and Noise level(black),Signal Level (red) and Adaptive
Threshold(green)');axis tight;
   ylabel('a.u.');xlabel('samples');
   hold on,scatter(qrs_i,qrs_c,'m');
   hold on,plot(locs,NOISL_buf,'LineWidth',2,'Linestyle','--','color','k');
   hold on,plot(locs,SIGL_buf,'LineWidth',2,'Linestyle','-.','color','r');
   hold on,plot(locs,THRS_buf,'LineWidth',2,'Linestyle','-.','color','g');
```

```
  az(3)=subplot(313);
  plot(ecg-mean(ecg));
  title('Pulse train of the found QRS on ECG signal');
  ylabel('a.u.');xlabel('samples');
  axis tight;
  line(repmat(qrs_i_raw,[2 1]),...
     repmat([min(ecg-mean(ecg))/2; max(ecg-mean(ecg))/2],size(qrs_i_raw)),...
     'LineWidth',0.5,'LineStyle','-.','Color','r');
  linkaxes(az,'x');
  zoom on;
 end
end
```