КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра радіотехніки та радіоелектронних систем

До захисту допущено: «На правах рукопису»
Завідувач кафедри _____ Ігор АНІСІМОВ
19 грудня 2022 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Апаратно-програмний комплекс контролю змін периферійного кровотоку та динаміки трансформації молекул ксигемоглобіну при використанні стоматологічного лазеру»

Виконав:
студент 2-го курсу магістратури
денної форми навчання
спеціальності 172 Телекомунікації та радіотехніка
ОПП «Захист інформації в телекомунікаціях»
Ілюха Микола Романович _____

Науковий керівник:
к.ф.-м. н., доц. Бех Ігор Іванович _____

Рецензент:
к.ф.-м. н., с.н.с. Соколов Володимир Олександрович _____

Засвідчую, що у цій магістерській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____

Робота допущена до захисту в ЕК рішенням кафедри радіотехніки та радіоелектронних систем від 19 грудня 2022 р., протокол № 9.

Завідувач кафедри радіотехніки та радіоелектронних систем,
доктор фіз.-мат. наук, професор
Анісімов Ігор Олексійович _____

Київ – 2022

РЕФЕРАТ

Робота складається: ст. 48, рис. 21, вступ, 4 розділи, висновки, список використаних джерел та три додатки.

Ключові слова: кровоток, оксигемоглобін, терапевтичний ефект, стоматологічний лазер.

Дана робота присвячена розробці та виготовленню прототипу апаратно-програмного комплексу для контролю змін периферійного кровотоку та динаміки трансформації молекул оксигемоглобіну при вивченні терапевтичного ефекту при застосуванні стоматологічних лазерів. Для побудови макету використано сучасні компонентну базу цифрової та аналогової електроніки та програмне забезпечення.

Макет апаратно-програмного комплексу виконує такі функції - реєстрація двохвильової фотоплетизмограми, виділення однієї фотоплетизмограми та визначення відносного рівня оксигемоглобіну в крові.

Показано, що запропоновані в роботі елементна база та програмне забезпечення цілком придатні для побудови апаратно-програмного комплексу для контролю динаміки трансформації молекул оксигемоглобіну, а відносний рівень сатурації крові киснем, виміряний за допомогою виготовленого прототипу, співпадає із значенням сатурації крові киснем, визначеним за допомогою побутового пульсоксиметра, з точністю до ±0.5%.

ЗМІСТ

ВСТУП

Фізика з кожним днем стає складовою все більшої кількості сфер нашого життя. Одне з останніх таких нововведень – лазерне випромінювання у медицині.

Лазери в медицині використовуються для лабораторної діагностики, профілактики і лікування різних захворювань. У цих цілях використовуються два типи лазерів: терапевтичні лазери (низькоенергетичні) і хірургічні лазери (високоенергетичні) [1, 2]. Зокрема, терапевтичні лазери використовують в комплексному лікуванні хронічних захворювань, з метою мобілізації організму, активізації його імунної системи за рахунок серії біологічних ефектів [3, 4].

В останні роки лазери знайшли своє застосування і в стоматології [5, 6]. В стоматології хірургічні лазери застосовується з метою розсічення для дренажу абсцесів, висічення фібром, оголення імплантату, абляції площинних захворювань слизової оболонки порожнини рота, корекції високо прикріпленої вуздечки губи, язика і щоки, резекції верхівки кореня зуба, тощо.

Терапевтичний ефект стоматологічних лазерів, скоріш за все, пов'язаний з фотодисоціацією оксигемоглобіну та виділенням вільного кисню [7]. Це пояснює нестабільність терапевтичних результатів, оскільки величина фотодисоціації оксигемоглобіну залежить від величини сатурації артеріальної крові киснем [8].

Для вивчення терапевтичного ефекту при застосуванні стоматологічних лазерів було запропоновано провести ряд досліджень змін периферійного кровотоку та динаміки трансформації молекул оксигемоглобіну. Для проведення даних досліджень необхідний аналізатор, який буде компактним та високоточним. Аналізатор повинен: проводити реєстрацію двохвильової фотоплетизмограми, виділяти одну фотоплетизмограму та визначати відносний рівень оксигемоглобіну в крові.

У зв’язку зі сказаним вище переді мною було поставлено задачу створити із використанням сучасної елементної бази апаратно-програмний комплекс для контролю змін периферійного кровотоку та динаміки трансформації молекул оксигемоглобіну при вивченні терапевтичного ефекту при застосуванні стоматологічних лазерів.

ОГЛЯД ЛІТЕРАТУРИ

РОЗДІЛ 1

ВЗАЄМОДІЯ ЛАЗЕРНОГО ВИПРОМІНЮВАННЯ ІЗ БІОЛОГІЧНИМИ ТКАНИНАМИ

Попри широке використання хірургічних лазерів у стоматології, їх терапевтична дія залишається поза увагою. Однією з причин цього може бути те, що механізм терапевтичної дії лазерів до кінця не вивчений. При застосуванні лазерних технологій, навіть для одного і того ж пацієнта, але в різний час доби, можуть бути різні параметри [6]. Лікарі, які застосовують терапевтичні лазери багато років, володіють великим досвідом, підбирають індивідуальні параметри для кожного пацієнта. Слід зауважити, що оскільки довжина хвилі більшості стоматологічних лазерів знаходиться в червоній та ближній інфрачервоній ділянках спектру, а ясна є біологічною тканою з дуже великою кількістю кровоносних судин, то випромінювання попадає в область ефективної фотодисоціації оксигемоглобіну [7]. Припускається, що терапевтичний ефект стоматологічних лазерів, скоріш за все, пов'язаний з фотодисоціацією оксигемоглобіну та виділенням вільного кисню. Це пояснює нестабільність терапевтичних результатів, оскільки величина фотодисоціації оксигемоглобіну залежить від величини сатурації артеріальної крові киснем [8].

Ключову ідею у дослідженні залежності фотодисоціації оксигемоглобіну від рівня сатурації крові киснем зіграла вперше запропонована і розроблена гіпотеза про роль лазерно-індукованої фотодисоціації оксигемоглобіну (HbO_2) в кровоносних судинах у механізмі біостимулюючого та терапевтичного ефекту низькоінтенсивного лазерного випромінювання. Суть ідеї полягає в тому, що при дії лазерним випромінюванням через шкірний покрив, частина випромінювання неминуче поглинається оксигемоглобіном в кровоносних судинах [9]. При цьому варто

очікувати, що з імовірністю ~ 10% відбувається фотодисоціація з вивільненням кисню і відновленням гемоглобіну(Hb): $[HbO_2] \rightarrow [Hb] + [O_2]$

Ця гіпотеза експериментально була підтверджена і узагальнена в [9]. Стосовно комплексу гемоглобіну з киснем, лазерно-індукована фотодисоціація дозволяє додатково вивільнити кисень в зоні опромінення і, таким чином, усунути тканинну гіпоксію. Тим самим стимулювати аеробний метаболізм клітин і досягти бажаного терапевтичного ефекту шляхом впливу низькоінтенсивного лазерного випромінювання.

Основою запропонованого методу впливу низькоінтенсивного лазерного випромінювання для стимуляції аеробного метаболізму клітин і досягнення бажаного терапевтичного ефекту є опромінення кровоносних судин і капілярів оптичним випромінюванням певної довжини хвилі. При цьому значна частина енергії, що поглинається карбоксигемоглобіном, витрачається на фотодисоціацію.

Для проведення досліджень змін периферійного кровотоку та динаміки трансформації молекул оксигемоглобіну при вивченні терапевтичного ефекту при застосуванні стоматологічних лазерів планується провести ряд безпосередніх досліджень. Для цього слід мати компактний та високоточний аналізатор, який дозволить реєструвати двохвильовий фотоплетизмографічний сигнал, виділяти поодиноку фотоплетизмограму та визначати відносну концентрацію оксигемоглобіну в крові. Створення прототипу апаратно-програмного комплексу для контролю змін периферійного кровотоку та динаміки трансформації молекул оксигемоглобіну при вивченні терапевтичного ефекту при застосуванні стоматологічних лазерів і є метою даної роботи.

РОЗДІЛ 2
СХЕМА ПРИСТРОЮ ТА ВИМОГИ ДО АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ

1.

На рис. 1.1 наведено структурну схему апаратно-програмного комплексу для контролю змін периферійного кровотоку та динаміки трансформації молекул оксигемоглобіну при вивченні терапевтичного ефекту при застосуванні стоматологічних лазерів.

Рис. 1.1. Структурна схема апаратно-програмного комплексу для контролю змін периферійного кровотоку та динаміки трансформації молекул оксигемоглобіну [10].

Вона складається із оптичного датчика (1) з вбудованими світлодіодами та фотоприймачем, мікропроцесорного блоку керування (2), що включатиме схему формування імпульсів живлення світлодіодів, комутатор, вхідний підсилювач сигналу фотоприймача, АЦП та інтерфейс для зв'язку з персональним комп’ютером (3), на якому буде встановлено відповідне програмне забезпечення [10] для отримання, оброблення, візуалізації та збереження даних, які надходитимуть від мікропроцесорного блоку керування.

Схема інформаційних потоків у апаратно-програмному комплексі наведена на рис. 1.2.

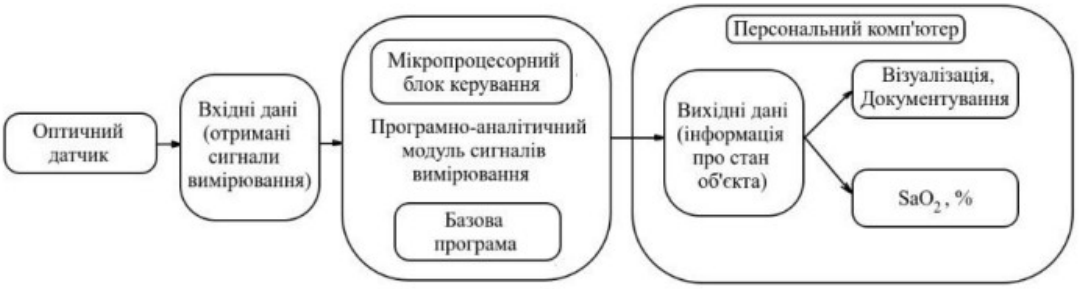

Рис. 1.2. Схематичне зображення інформаційних потоків у аналізаторі змін периферійного кровотоку та динаміки трансформації молекул оксигемоглобіну [10].

На даному етапі проектування пристрою та створення програмного забезпечення, керування роботою мікропроцесорного блоку керування та отримання даних від нього виконується за допомогою послідовного інтерфейсу передачі даних UART через провідне з'єднання з використанням USB.

В подальшому для забезпечення більш зручного обміну даними між мікропроцесорним блоком керування та персональним комп'ютером може бути використаний інший протокол передачі даних, наприклад, Bluetooth або Wi-fi. Це дозволить покращити безпеку каналів зв'язку для запобігання витоку конфіденційної інформації.

Для проведення досліджень змін периферійного кровотоку та динаміки трансформації молекул оксигемоглобіну при вивченні терапевтичного ефекту при застосуванні стоматологічних лазерів апаратно-програмний комплекс повинен:

- проводити калібрування системи перед початком кожного циклу вимірювання – встановлення необхідних струмів через світлодіоди. Після калібрування встановлені параметри не змінюються протягом всього циклу роботи приладу з окремим пацієнтом;
- робити вимірювання рівня сатурації крові у реальному часі;
- візуалізувати дані у вигляді графіку;

- давати можливість зберігати дані кожного дослідження у файлі текстового формату;
- давати можливість перегляду результатів попередніх досліджень.

РОЗДІЛ 3
МАКЕТ АПАРАТНОЇ ЧАСТИНИ АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ

3.1. Апаратне забезпечення та середовище розробки

3.1.1. Оптичний датчик

Для виконання поставленої і роботі задачі було використано інтегрований пульсоксиметричний модуль та монітор серцевого ритму MAX30102 від компанії Maxim Integrated (США). Він включає вбудовані світлодіоди, фотодетектор, оптичні елементи та електроніку з малим рівнем власних шумів і захистом від впливу навколишнього світла. Його характеристики [11], повністю задовольняють умовам технічного завдання.

На рис. 3.1 наведено фото даного датчика.

Рис. 3.1. Фото датчика MAX30102.

Даний датчик при його налаштуванні дає можливість вибору:

- тривалості імпульсів струму через світлодіоди;

- частоти засвічування світлодіодів;
- струму через світлодіоди;
- розрядності АЦП;
- режиму роботи датчика: червоний або червоний та інфрачервоний світлодіод;
- усереднення вихідних значень.

3.1.2. Плата мікроконтролера

В цій роботі був використаний мікроконтролер ARDUINO UNO REV3 [12], який зображений на рис. 3.2.

Рис. 3.2. Фото мікроконтролера ARDUINO UNO REV3.

Використання мікроконтролера ARDUINO UNO REV3 обумовлене:

- його досить малими енергозатратами;
- достатнім об’ємом внутрішньої пам’яті, що дозволяє робити первинну обробку даних;
- невеликими розмірами плати;
- гнучкістю програмування;
- швидкістю опитування;
- дешевизною плати.

На рис. 3.3 зображено схему підключення оптичного датчика до мікроконтролера ARDUINO UNO REV3 по інтерфейсу I^2C.

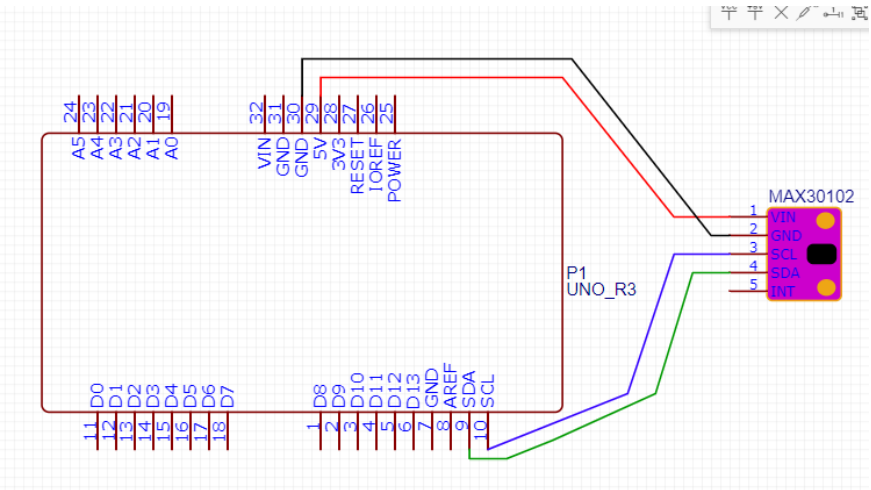

Рис. 3.3. Схема підключення датчика до мікроконтролера.

А також на рис. 3.4 наведено фото зібраного макету.

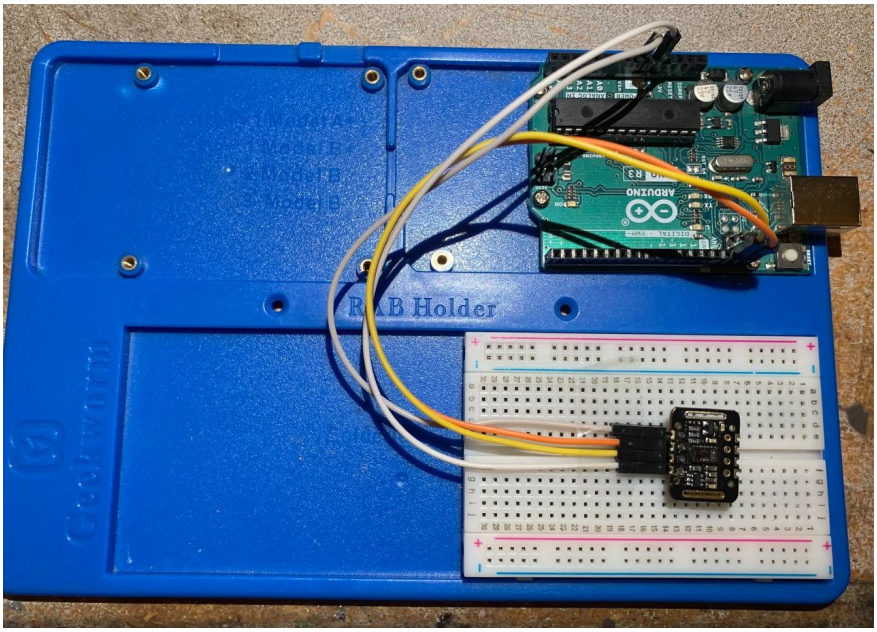

Рис. 3.4. Фото зібраного макету.

На фото видно, що датчик під’єднаний до мікропроцесорного блоку керування довгими дротами. Це зроблено для того, щоб зручніше проводити вимірювання до того, як оптичний модуль буде винесено на спеціальний тримач для зручного використання у стоматології.

3.1.3. Середовище розробки

У даній роботі було використано середовище програмування Arduino IDE та бібліотеку для датчика MAX30102 під назвою SparkFun_MAX3010x_Sensor_Library [13]. Бібліотека для датчика дозволяє легко використовувати його основні функції та задавати початкові налаштування.

Було написано програму для реєстрації даних з каналів червоного та інфрачервоного випромінювання датчика MAX30102.

Програма виконує такий алгоритм дій:

- підключення потрібних бібліотек;
- ініціалізація послідовного з'єднання із заданою швидкістю передачі даних;
- ініціалізація оптичного датчика за допомогою I^2C інтерфейсу із стандартною швидкістю опитування 400 кГц;
- задаються параметри роботи датчика;
- передання датчику заданих налаштувань;
- переведення датчика в режим очікування;
- проведення калібрування струму, що проходить через світлодіоди (на цьому етапі дослідження не використовується);
- зчитування з фотодіоду сигналу та його передача за допомогою послідовного інтерфейсу передачі даних UART через провідне з'єднання з використанням USB.

Код програми для зчитування даних з датчика наведений в додатку А.

3.2. Визначення оптимальних налаштувань оптичного датчика

Для вибору оптимальних налаштувань оптичного датчика було проведено дослідження впливу на його вихідний сигнал різних параметрів. В кожному досліді до датчика був прикладений палець.

На початку дослідження було обрано середні значення для усіх параметрів:

- струм через світлодіоди = 12 мА;
- тривалість імпульсу струму через світлодіоди = 118 мкс;
- частота засвічення світлодіодів = 400 Гц;
- розрядність АЦП = 14 біт.

На першому етапі досліджувалась залежність вихідного сигналу оптичного датчика від значення струму через світлодіоди. На рис. 3.5 та рис. 3.6 зображені фотоплетизмограми червоного та інфрачервоного каналів відповідно при струмі через світлодіод у 0,2 мА, 25 мА та 50 мА.

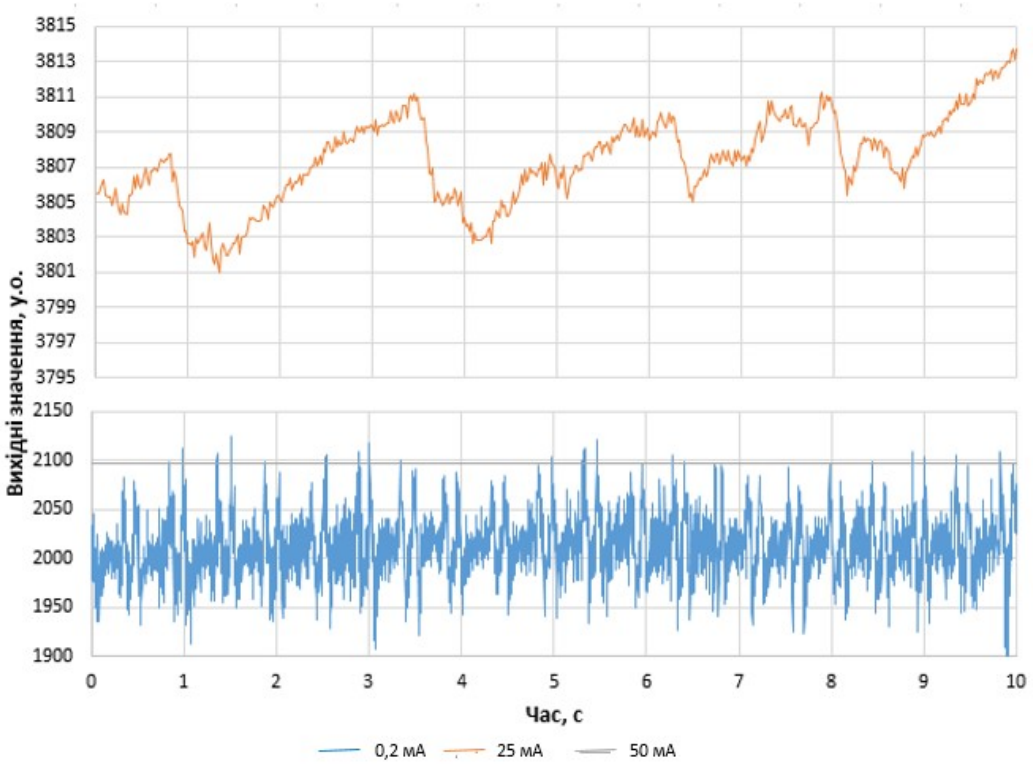

Рис. 3.5. Фотоплетизмограми червоного каналу при різних значеннях струму через світлодіод.

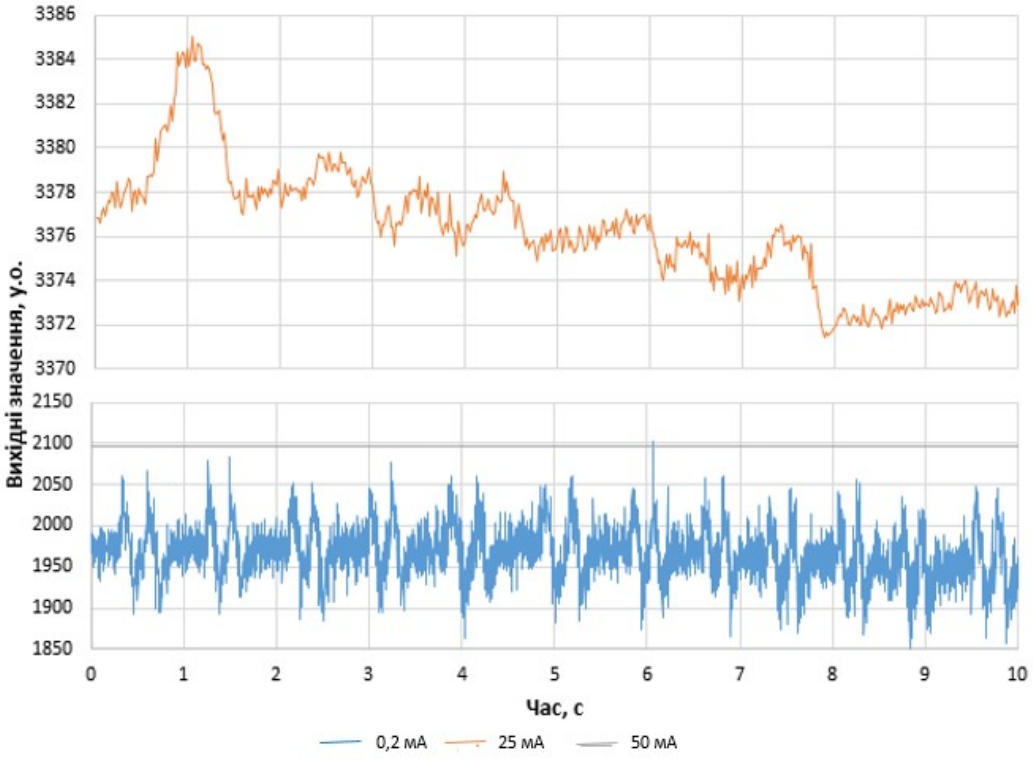

Рис. 3.6. Фотоплетизмограми інфрачервоного каналу при різних значеннях струму через світлодіод.

На рис. 3.5 та рис. 3.6 фотоплетизмограми при струмах 12 мА та 25 мА нормовані за струмом на величину струму для того, щоб зменшити до мінімуму вплив від зміни потужності сигналу на результати вимірювань. З рисунків видно, що при струмі 0,2 мА дуже велика шумова компонента, при струмі 12 мА ця шумова компонента вже немає значного впливу на вихідний сигнал. Є припущення, що це тепловий шум, який при малих значеннях потужності сигналу, що надходить на фотодетектор, вносить дуже суттєву похибку у вимірювання. При струмі 25 мА сигнал виходить за динамічний діапазон АЦП при даних налаштуваннях, через це фотоплетизмограма має вигляд прямої паралельної до осі абсцис. Пульсову хвилю чітко видно тільки при струмі 12 мА. В порівнянні з червоним каналом, на інфрачервоному каналі амплітуда пульсової хвилі значно менша, але її все ж можна визначити.

З цього можна зробити висновок, що для отримання чіткого вихідного сигналу з мінімальним впливом шумових компонент краще обирати середні значення струму через світлодіод.

На другому етапі, при значенні струму через світлодіоди 12 мА, було проведено вимірювання залежності вихідного сигналу червоного та інфрачервоного каналів оптичного датчика при значеннях тривалості імпульсу струму через світлодіодів у 69 мкс, 118 мкс та 411 мкс. На рис. 3.7 та рис. 3.8 зображено фотоплетизмограми червоного та інфрачервоного каналів при різних значеннях тривалості імпульсу струму через світлодіоди.

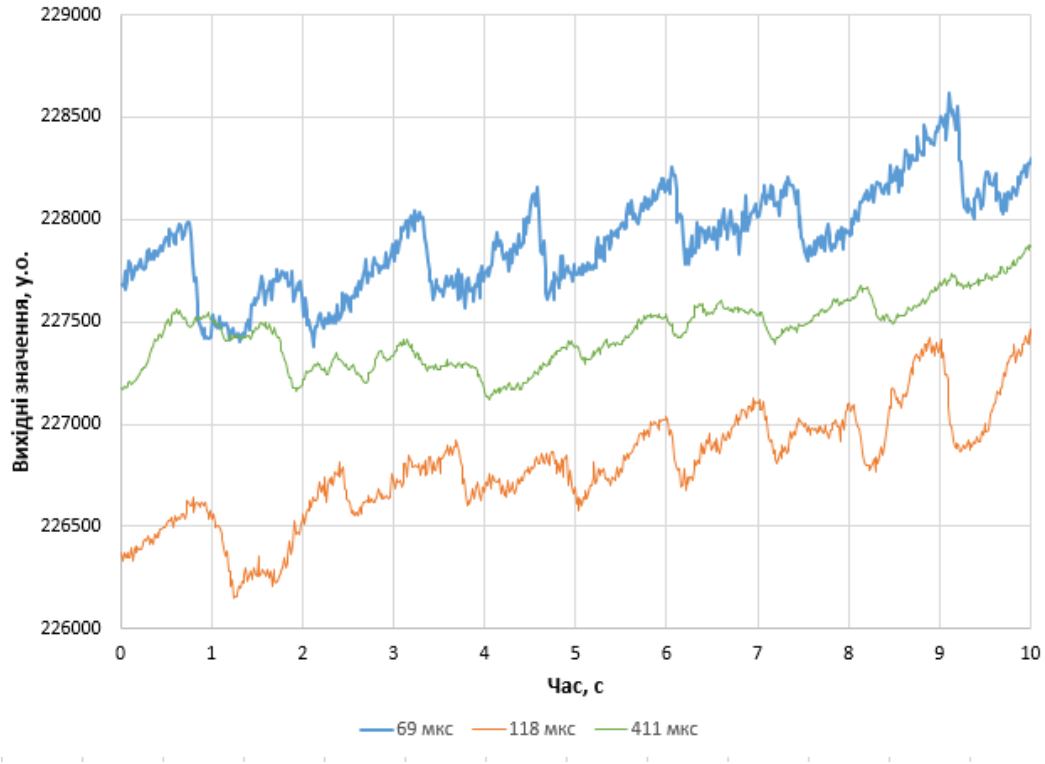

Рис. 3.7. Фотоплетизмограми червоного каналу при різних значеннях тривалості імпульсу струму через світлодіод.

З даного рисунка видно, що зі збільшенням тривалості імпульсу струму через світлодіоди зменшується амплітуда пульсової хвилі. Також на червоному каналі пульсова хвиля доходить до стабільного рівня швидше, якщо тривалість імпульсу струму через світлодіоди менша. Пульсову хвилю

чітко видно при тривалості імпульсу струму через світлодіоди у 69 мкс та 118 мкс.

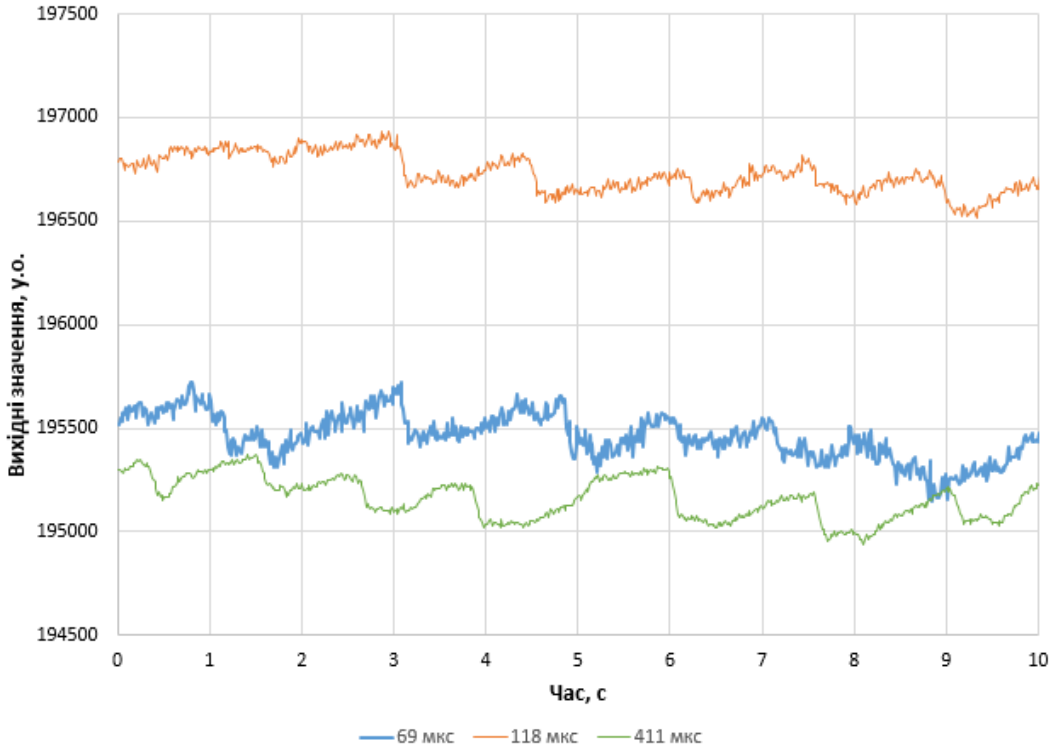

Рис. 3.8. Фотоплетизмограми інфрачервоного каналу при різних значеннях тривалості імпульсу струму через світлодіод.

З даного рисунка можна побачити, що при низьких значеннях тривалості імпульсу струму через світлодіоди пульсова хвиля погано розрізняється серед усього сигналу. Починаючи зі значення тривалості імпульсу струму через світлодіоди у 118 мкс пульсова хвиля розрізняється краще, а при значенні 411 мкс менш чітко виражена та швидко виходить на стабільний рівень, має не велику, але достатню для розрізнення серед іншого сигналу, амплітуду.

Аналіз графіків, наведених рис. 3.7 та рис. 3.8, дозволяє припустити, що краще використовувати середні значення тривалості імпульсу струму через світлодіоди, бо вони показують кращу результативність на двох каналах (червоному та інфрачервоному) порівняно з іншими.

На третьому етапі при струмі через світлодіоди у 12 мА та тривалості імпульсу струму через світлодіоди у 118 мкс було проведено дослідження впливу різних значень частоти засвічення світлодіодів (50 Гц, 200 Гц, 400 Гц та 800 Гц) на вихідний сигналу оптичного датчика. На рис. 3.9 та 3.10 зображено фотоплетизмограми червоного та інфрачервоного каналів при різних значеннях частоти засвічення світлодіодів.

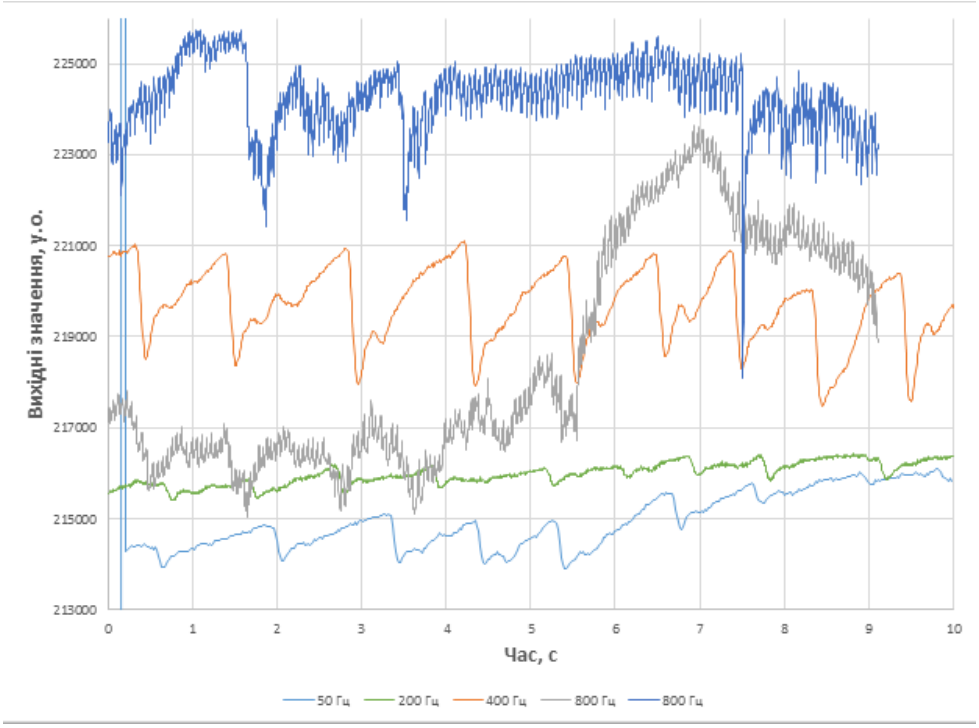

Рис. 3.9. Фотоплетизмограми червоного каналу при різних значеннях частоти засвічення світлодіоду.

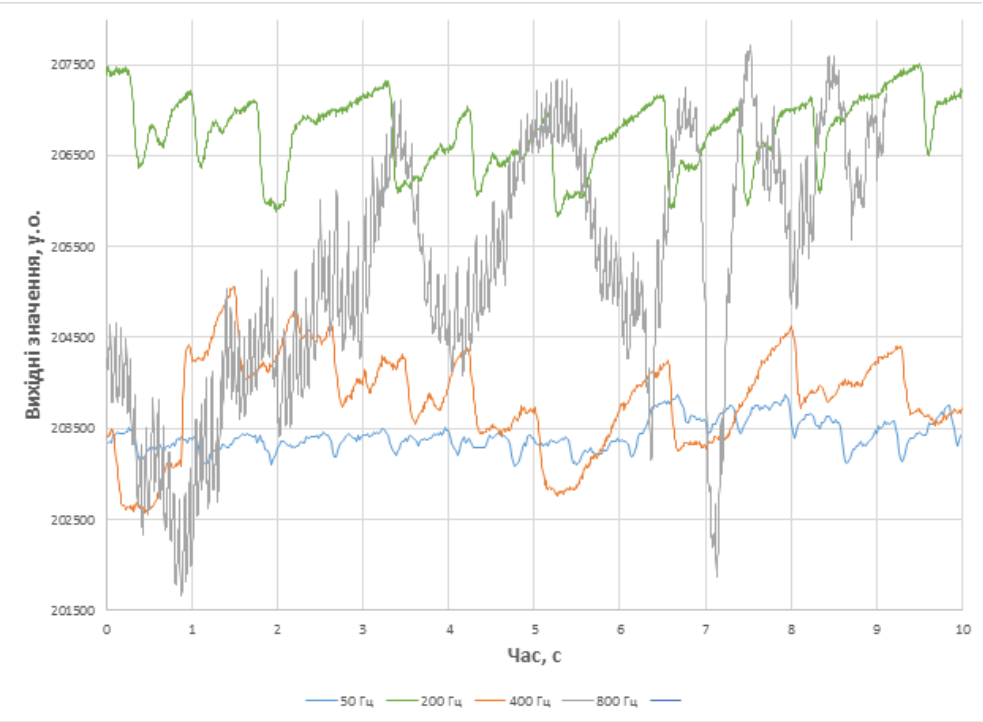

Рис. 3.10. Фотоплетизмограми інфрачервоного каналу при різних значеннях частоти засвічування світлодіоду.

Як видно з рис. 3.9, на фотоплетизмограмах червоного каналу при частоті засвічення світлодіодів 50 Гц та 400 Гц чітко видно пульсову хвилю з достатньою для розрізнення з поміж іншого сигналу амплітудою та малим впливом шумової компоненти. При частоті засвічення світлодіодів 200 Гц пульсова хвиля має дуже малу амплітуду, тому можуть виникнути складнощі у аналізі даних при використанні даної частоти. При частоті засвічення світлодіодів 800 Гц дуже великий вплив шумової компоненти, тому це пульсову хвилю не завжди можна розрізнити. Різкі перепади сигналу зумовлені рухами пальця під час вимірювання.

Фотоплетизмограми з інфрачервоного каналу, наведені на рис. 3.10, показують, що пульсова хвиля дуже добре розрізнюється та має досить велику амплітуду при частотах засвічення світлодіодів 200 Гц та 400 Гц. На частоті засвічення світлодіодів 50 Гц у вихідного сигналу дуже мала амплітуда пульсової хвилі, а при 800 Гц ситуація така ж сама, як і на червоному каналі, тобто дуже великий вплив шумової компоненти, з якої важко виділити корисний сигнал.

Після аналізу усіх отриманих результатів було обрано оптимальні налаштування для оптичного датчика, при яких якість вихідних даних є найвищою: струм через світлодіоди – 12 мА, тривалість імпульсу струму через світлодіоди – 118 мкс, частота засвічення світлодіодів – 400 Гц.

Ці параметри якнайкраще підходять для аналізу даних з мого пальця. І це дуже важливо, оскільки для кожної людини через біологічні особливості знадобляться різні налаштування. В першу чергу це стосується струму через світлодіоди, саме для цього необхідна процедура калібрування перед кожним новим вимірюванням. Усі інші налаштування є оптимальними та універсальними для усіх людей.

РОЗДІЛ 4
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Графічний інтерфейс

Для створення зручного графічного інтерфейсу, інтуїтивно зрозумілого користувачу, було обрано програмне середовище Qt Designer [14].

Qt Designer - кросплатформене вільне середовище для розробки графічних інтерфейсів (GUI) програм, що використовують бібліотеку Qt . Входить до складу Qt framework [14].

Це середовище дуже зручне, бо воно використовує стандартні елементи інтерфейсу - віджети Windows, що звичні та зрозумілі для багатьох користувачів. Кожен віджет має свій набір властивостей, відповідним класом бібліотеки Qt. Властивості віджету можуть бути змінені за допомогою "Редактора влади". Для кожного класу влади віджету існує спеціалізований редактор. Характерною особливістю Qt Designer є підтримка візуального редагування сигналів та слотів. Так, наприклад, можна зв'язати сигнал, що генерується з перемикання стану віджету CheckBox зі слотом, що відповідає доступності іншого віджету.

Використовуючи дане середовище було створено графічний інтерфейс з двома вкладками. На рис. 4.1 - 4.2 зображено знімки екрану графічного інтерфейсу створеної за допомогою Qt Designer програми.

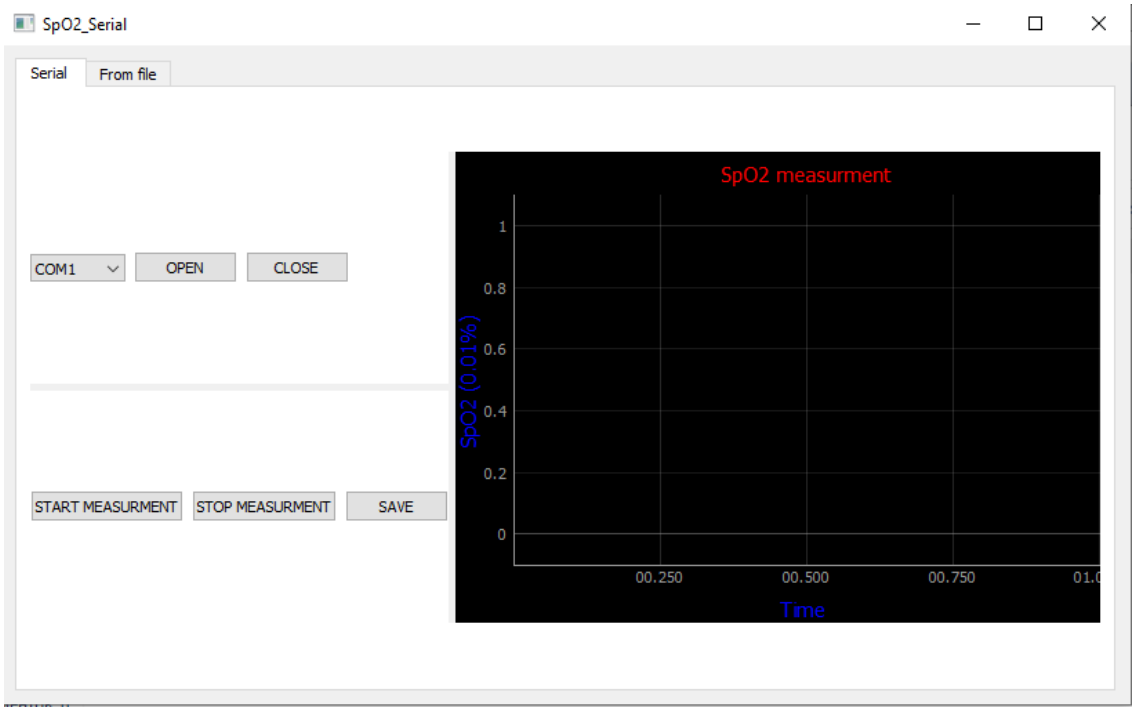

Рис. 4.1. Знімок екрану графічного інтерфейсу створеної програми.

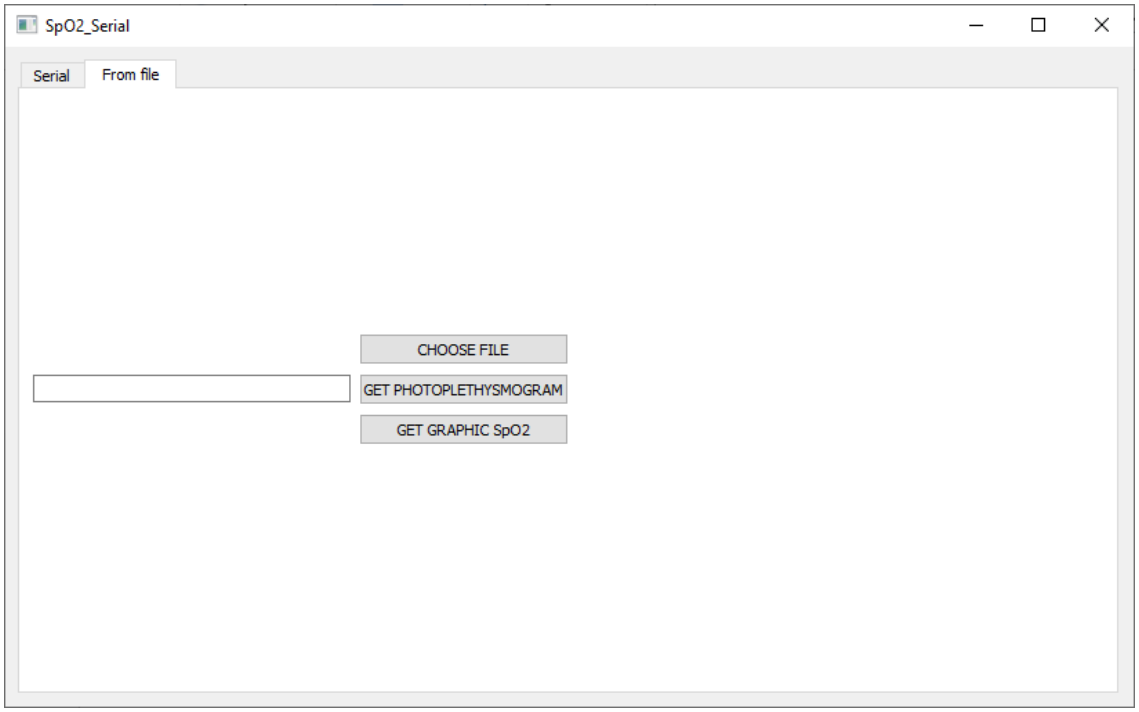

Рис. 4.2. Знімок екрану графічного інтерфейсу створеної програми.

На рис. 4.1 зображено знімок екрану графічного інтерфейсу, призначеного для вимірювань рівня сатурації крові у реальному часі. У даному інтерфейсі можна побачити три секції.

У першій секції у лівому верхньому куті знаходиться три кнопки, призначені для організації каналу зв’язку між персональним комп’ютером та мікропроцесорним блоком керування. Віджет ComboBox призначений для вибору порту до якого підключений мікропроцесорний блок керування, а кнопки «OPEN» та «CLOSE» призначені для ініціалізації з’єднання та закінчення сесії відповідно.

У другій секції у лівому нижньому куті знаходять три кнопки для контролю вимірювань у реальному часі. Кнопки «START MEASURMENT» та «STOP MEASURMENT» призначені для передачі команди до мікропроцесорного блоку керування для початку та зупинки вимірювання відповідно. А кнопка «SAVE» призначена для збереження даних, отриманих під час останнього вимірювання, у форматі обраному користувачем.

І третя секція у якій знаходиться віджет для відображення графіку залежності сатурації крові киснем від часу.

На рис. 4.2 зображено знімок екрану графічного інтерфейсу, що призначений для побудови графіків фотоплетизмограм та рівня сатурації крові на основі попередніх вимірювань, збережених у форматі текстового файлу з даними.

У даному інтерфейсі знаходяться три кнопки:

- «CHOOSE FILE» - призначена для взаємодії з файловою системою, за її допомогою можна обрати файл з даними попередніх вимірювань для подальших дій;
- «GET PHOTOPLETHYSMOGRAM» - дозволяє побудувати фотоплетизмограми червоного та інфрачервного каналів за даними, зазначеними в обраному файлі, та взаємодіяти з ними;
- «GET GRAPHIC SpO2» - дозволяє побудувати графік сатурації крові киснем на основі даних, отриманих з обраного файлу.

Програмний код інтерфейсу програми наведений в додатку Б.

4.2. Інструменти для відображення графіків

Для відображення у графічному інтерфейсі графіків, побудованих за отриманими з мікропроцесорного блоку керування даними, було використано бібліотеку PyQtGraph [15].

PyQtGraph — це бібліотека графіки та графічного інтерфейсу на чистому Python, побудована на PyQt/PySide та numpy. Вона призначена для використання в математичних/наукових/інженерних додатках. Незважаючи на те, що бібліотека повністю написана на Python, вона дуже швидка завдяки потужній бібліотеці NumPy для роботи з числовими даними і платформі Qt GraphicsView для швидкого відображення [15].

Ця бібліотека була обрана для роботи через високу кількість переваг, а саме:

- базується на мові програмування Python;
- лінійні та точкові графіки;
- дані можна панорамувати/масштабувати за допомогою миші;
- швидке малювання для відображення та взаємодії даних у реальному часі;
- відображає більшість типів даних (int або float; будь-яка бітова глибина; RGB, RGBA або яскравість);
- у 2D-графіці використовують фреймворк Qt GraphicsView, який є високоздатним і досконалим;
- управління багатьма процесами, що дозволяє віддалено створювати графіки, підключати сигнал Qt між процесами та дуже просте вбудоване розпаралелювання [15].

На рис. 4.3 зображено знімок екрану вікна для відображення графіків, створеного за допомогою даної бібліотеки, в якому будуть відображатися графіки.

Рис. 4.3. Знімок екрану вікна для відображення графіків.

З рисунку можна побачити, що вікно виглядає мінімалістично, нічого зайвого, але його легко можна змінити до вигляду, який буде зручний для аналізу.

4.3. Алгоритм роботи програми

Для виконання поставлених у роботі цілей було створено графічні інтерфейси, зображені на рис. 4.1 та 4.2. Для їх правильної роботи необхідно було написати код та зробити усі компоненти взаємопов’язаними.

На рис. 4.4 зображено діаграму алгоритму роботи інтерфейсу для вимірювань сатурації крові киснем у режимі реального часу.

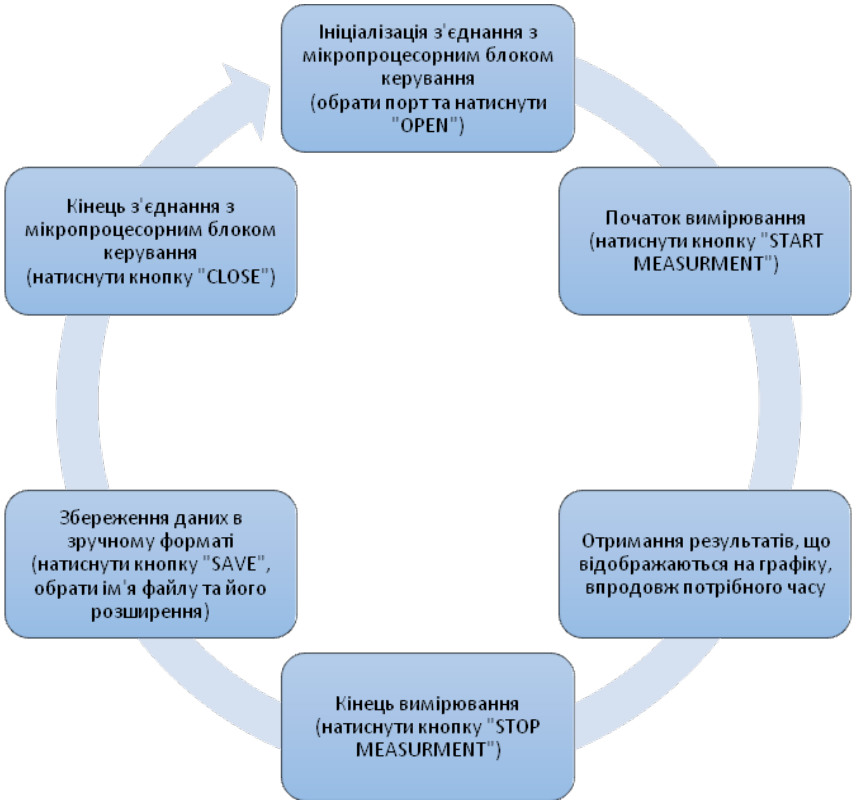

Рис. 4.4. Діаграма алгоритму роботи інтерфейсу для вимірювань сатурації крові киснем у режимі реального часу.

Робота починається з ініціалізації з'єднання з мікропроцесорним блоком керування, це запускає процес відправлення/отримання інформації між персональним комп’ютером та мікропроцесорним блоком керування.

Потім починається вимірювання, при натисканні на кнопку «START MEASURMENT» по послідовному каналу зв’язку відправляється старт-команда від персонального комп’ютера до мікропроцесорного блоку, він в свою чергу при надходженні даної команди починає процес роботи оптичного датчика та збору даних отриманих від нього. В свою чергу блок керування відправляє на комп’ютер дані з оптичного датчика, вимір за виміром. Коли програма на комп’ютері отримує певну задану кількість вимірів (кількість залежить від частоти вимірів оптичного модуля) вона робить обрахунок рівня сатурації крові киснем та записує це значення до масиву даних, що повинні відображатися на графіку. Після кожного такого обрахунку виконується команда, яка оновлює вікно програми, що дозволяє

оновити графік. Графік має обмеження в десять значень, щоб мати можливість бачити лише результати за останні 10 секунд і швидше реагувати на зміни, що відбуваються на ньому.

На рис. 4.5 зображено знімок екрану програми під час вимірювання в реальному часі.

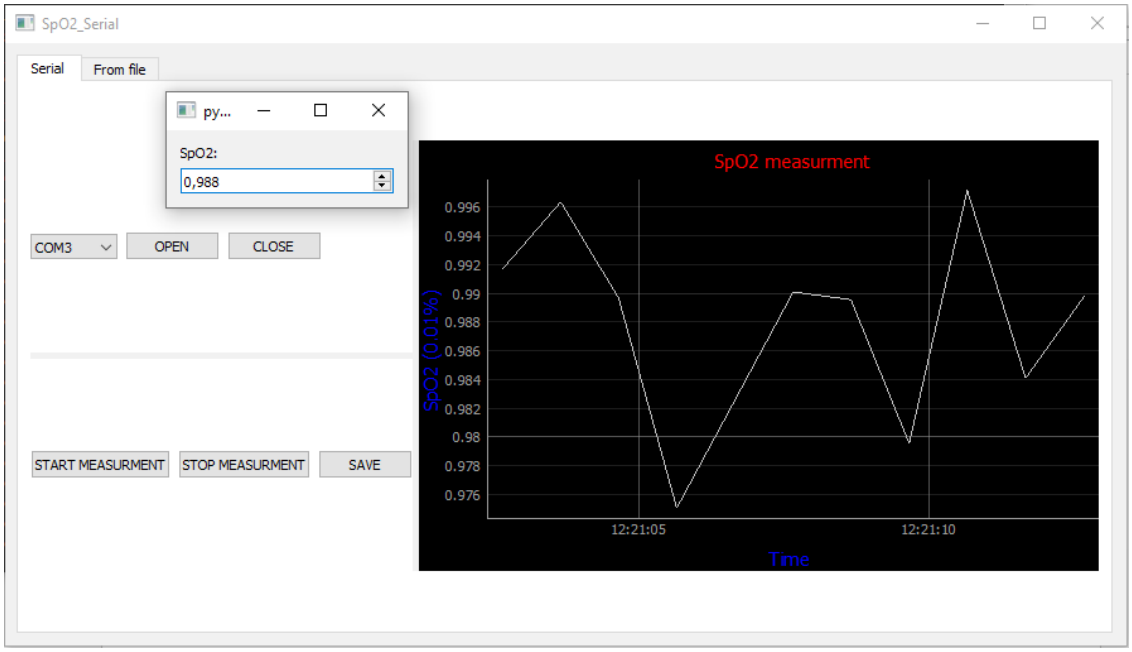

Рис. 4.5. Знімок екрану програми під час вимірювання в реальному часі.

На даному рисунку також видно маленьке вікно, воно показує середнє значення сатурації крові киснем за останні п’ять значень, обраховане як середнє арифметичне.

Щоб закінчити вимірювання необхідно натиснути кнопку «STOP MEASURMENT». Після натискання кнопки з персонального комп’ютера до мікропроцесорного блоку керування надсилається стоп-команда по послідовному каналу зв’язку. У цей момент блок керування зупиняє збір даних з оптичного модуля та переводить його в сплячий режим до наступного вимірювання. Після надсилання команди програма останній раз оновлює графік у робочому вікні.

Наступним кроком є збереження даних, отриманих під час вимірювання. Це можна зробити за допомогою кнопки «SAVE». Натискання кнопки викликає провідник для обрання місця збереження даних, імені файлу та його розширення. Після задання цих параметрів дані переписуються до створеного файлу.

Для завершення роботи необхідно натиснути кнопку «CLOSE», це призведе до закриття послідовного каналу зв’язку між персональним комп’ютером та мікропроцесорним блоком керування.

На рис. 4.6 зображено діаграму алгоритму роботи інтерфейсу для побудови графіків фотоплетизмограм та рівня сатурації крові киснем на основі попередніх вимірювань, збережених у форматі текстового файлу з даними.

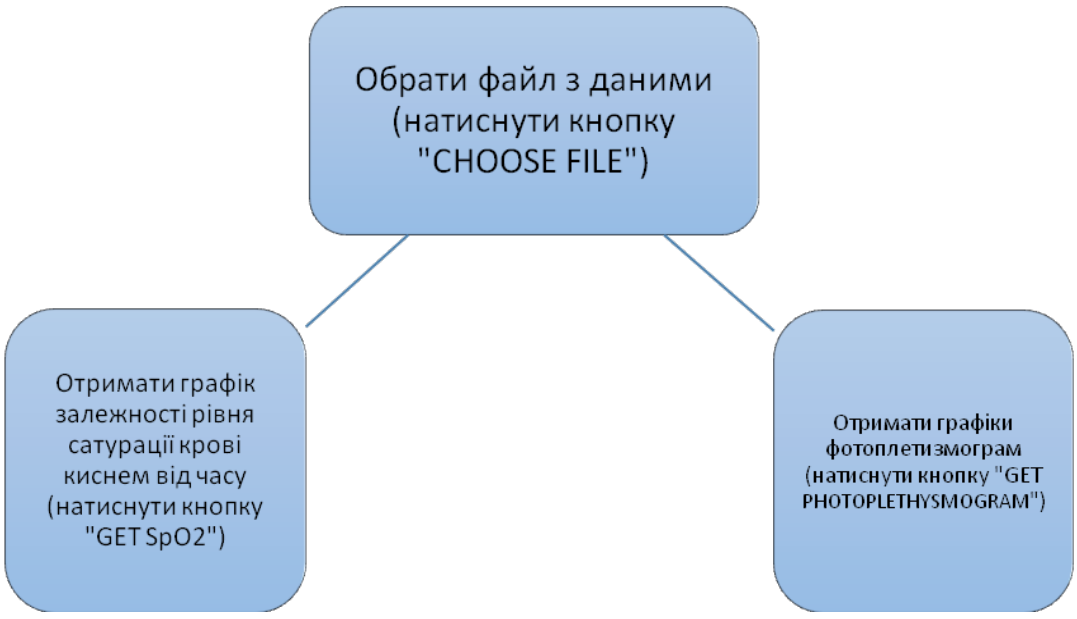

Рис. 4.6. Діаграма алгоритму роботи інтерфейсу для побудови графіків фотоплетизмограм та рівня сатурації крові киснем на основі попередніх вимірювань.

Для початку роботи необхідно натиснути кнопку «CHOOSE FILE», це відкриває файлову систему та дозволяє обрати потрібний для аналізу файл.

Після обрання файлу його розташування записується у строку (можливо написати розташування вручну).

Щоб отримати графіки фотоплетизмограм необхідно натиснути кнопку «GET PHOTOPLETHYSMOGRAM». Після натискання на кнопку зі строки витягується розташування файлу, тобто текст, трохи корегується для коректної роботи та файл за цим розташуванням відкривається. Далі програма рядок за рядком розбиває дані за правилом. Окремі частини кожного рядка, а саме дані червоного та інфрачервоного каналів записуються в окремі масиви даних. Використовуючи ці масиви будуються фотоплетизмограми. На рис. 4.7 зображено знімок екрану з фотоплетизмограмами побудованими за даними з файлу.

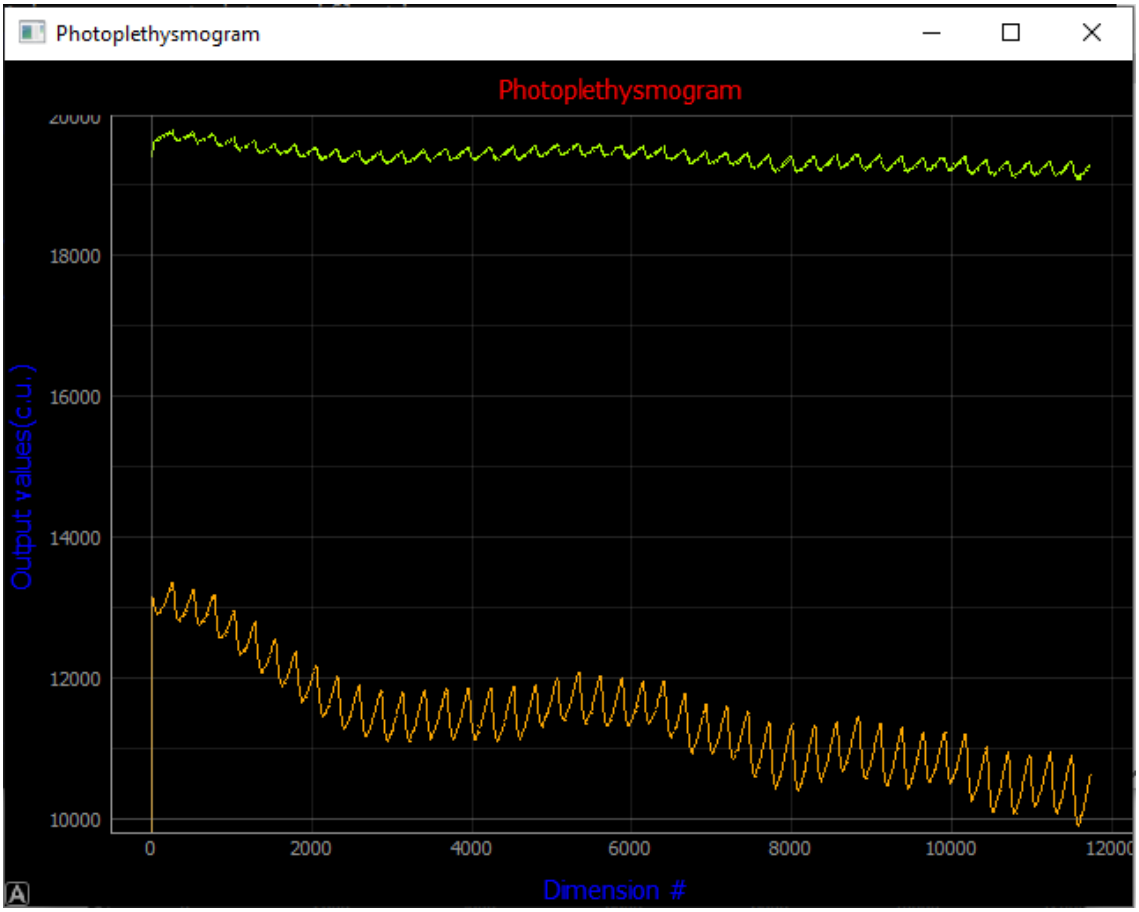

Рис. 4.7. Знімок екрану з фотоплетизмограмами побудованими за даними з файлу.

Також у цьому робочому вікні є можливість обирати область побудови графіку, робити різноманітні перетворення, а також за необхідності можна зберегти графік у форматі зображення.

Для того, щоб отримати графік залежності рівня сатурації крові киснем від часу необхідно натиснути кнопку «GET SpO_2». Після натискання на кнопку з рядка витягується розташування файлу, тобто текст, трохи корегується для коректної роботи та файл за цим розташуванням відкривається. Далі програма рядок за рядком розбиває дані за правилом. Ці дані записуються в окремі масиви для червоного та інфрачервоного каналів. Далі програма виділяє певну задану кількість вимірів (кількість залежить від частоти вимірів оптичного модуля), обчислює для них значення рівня сатурації крові киснем, додає це значення до масиву та переходить до наступного виміру. Програма проводить виміри доки не доходить до кінця масиву з вимірами. За даними отриманими в результаті обчислення будується графік. На рис. 4.8 знімок екрану з графіком рівня сатурації крові киснем побудованими за даними з файлу.

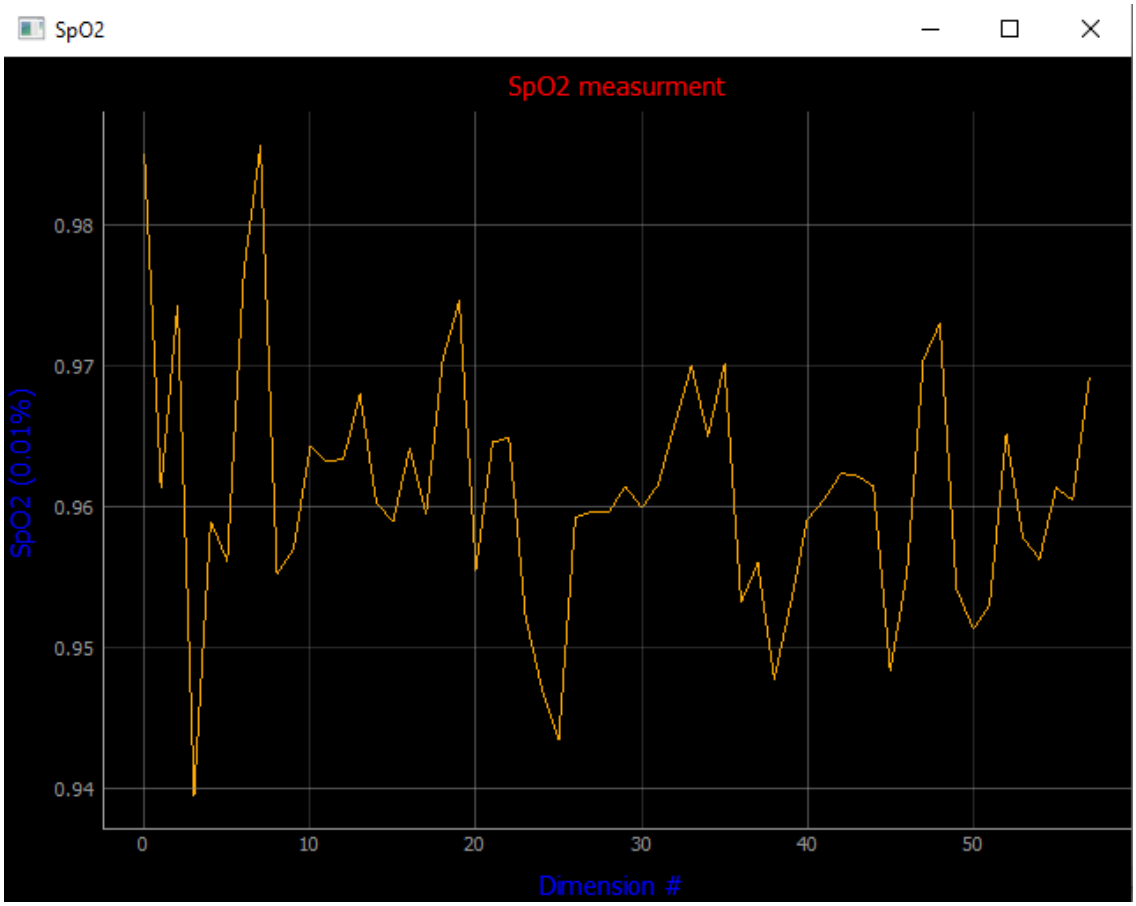

Рис. 4.8. Знімок екрану з графіком рівня сатурації крові киснем побудованими за даними з файлу.

На графіку видно багато значень та вони мають хаотичний характер. Кількість значень залежить від того, який коефіцієнт кореляції взяти за основу, в даному випадку на графік виводяться усі значення коефіцієнт кореляції яких більше 0,9.

Повний код програми наведено в додатку В.

Для перевірки працездатності було проведено порівняння роботи створеного апаратно-програмного комплексу у реальному часі з побутовим пульсоксиметром. Результати порівняння зображені на рис. 4.9.

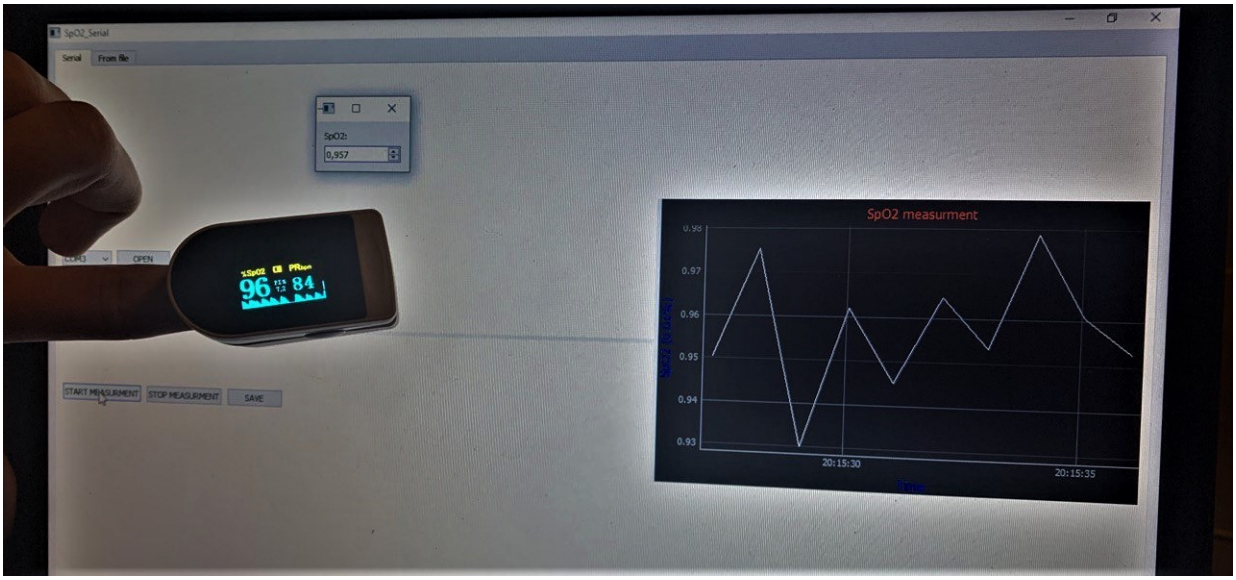

Рис. 4.9. Порівняння роботи апаратно-програмного комплексу у реальному часі з побутовим пульсоксиметром.

З рис. 4.9 можна побачити, що результати вимірювань сатураіцй крові киснем співпадають, враховуючи те, що ми не знаємо, який алгоритм округлення використовується у побутовому пульсоксиметрі. Враховуючи це, вважаю розбіжність до 0.5% допустимою.

Отже створений мною апаратно-програмний комплекс працює правильно та має дуже багато можливостей для удосконалення, починаючи від винесення оптичного модуля на тримач для більш зручного використання у стоматології та закінчуючи використанням інших технологій передавання сигналів, таких як Bluetooth або Wi-Fi.

ВИСНОВКИ

1. Проведений огляд літератури за темою дипломної роботи показав, що:
    - при інтенсивному використанні хірургічних лазерів у стоматології, через відсутність чітких уявлень про механізм їх терапевтичного впливу на біологічні тканини, ця дія залишається поза увагою;
    - терапевтичний ефект стоматологічних лазерів може бути пов’язаний з виділенням вільного кисню внаслідок фотодисоціаціїї оксигемоглобіну, бо більша частина стоматологічних лазерів працюють з довжинами хвиль червоної та ближньої інфрачервоної ділянок спектру, що припадає на область ефективної фотодисоціації оксигемоглобіну;
    - для вивчення терапевтичного ефекту при застосуванні стоматологічних лазерів необхідно провести ряд досліджень змін периферійного кровотоку та динаміки трансформації молекул оксигемоглобіну, що потребує спеціального аналізатора, який буде компактним та високоточним, тому завдання по створенню комплексу для контролю змін периферійного кровотоку та динаміки трансформації молекул оксигемоглобіну є актуальним.

2. Можливість застосування обраної елементної бази для побудови апаратно-програмного комплексу для контролю динаміки трансформації молекул оксигемоглобіну продемонстрована на прикладі макету, створеного в ході виконання роботи.

3. Значення рівня сатурації крові киснем, який визначається за допомогою апаратно-програмного комплексу, співпадає із значенням рівня, визначеного за допомогою побутового пульсоксиметра, з точністю до ±0.5%.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. M. Jovicic, L. Konstantinovic, M. Lazovic, and V. Jovicic, “Clinical and functional evaluation of patients with acute low back pain and radiculopathy treated with different energy doses of low level laser therapy” *Vojnosanitetski Pregled*, vol. 69, no. 8, pp. 656–662, 2012.

2. R. Antonić, “The effects of low level laser therapy on the management of chronic idiopathic orofacial pain: trigeminal neuralgia, temporomandibular disorders and burning mouth syndrome,” *Medicina Fluminensis*, vol. 53, no. 1, pp. 61–67, 2017.

3. Y.-F. Huang, J.-C. Lin, H.-W. Yang, Y.-H. Lee, and C.-H. Yu, “Clinical effectiveness of laser acupuncture in the treatment of temporomandibular joint disorder,” *Journal of the Formosan Medical Association*, vol. 113, no. 8, pp. 535–539, 2014.

4. A. M. Shirani, N. Gutknecht, M. Taghizadeh, and M. Mir, “Low-level laser therapy and myofacial pain dysfunction syndrome: a randomized controlled clinical trial,”*Lasers in Medical Science*, vol. 24, no. 5, pp. 715–720, 2009.

5. R. A. Convissar, *Principles and Practice of Laser Dentistry*, Elsevier Health Sciences, New York, NY, USA, 2nd edition, 2015.

6. Rittinghouse J. W. Business continuity and disaster recovery for infosec managers / John W. Rittinghouse, James F. Ransome. – Oxford: Elsevier, 2015. – 408 p.

7. https://bolinet.com.ua/lazeryvmedicine.htm (дата звернення: 13.10.2022).

8. S.S.Yesman, S.O.Mamilov, D.V.Veligotsky, A.I.Gisbrecht "Local changes in arterial oxygen saturation induced by visible and near infrared light radiation" Lasers in Medical Science, (2016) DOI: 10.1007/s10103-015-1838-y. ISSN: 0268-8921 (Print) 1435-604X (Online).

9. Experimental study of the laser-induced oxyhemoglobin photodissociation in cutaneous blood vessels. A. Gisbrecht, S. Mamilov. Acta Medica Bulgarica, Vol. XLII, 2015, № 2, pp. 42-48

10. Остапенко О.С., Янішевскій К.А., Мамілов С.О., Бех І.І. Програмно-апаратний комплекс для контролю змін периферійного кровотоку та

динаміки трансформації молекул оксигемоглобіну при використанні стоматологічного лазеру: матеріали X Міжнародної конференції «Медична фізика – сучасний стан, проблеми, шляхи розвитку. Новітні технології» (Київ, 22–24 вересня, 2021 рік). Київ. С. 102-106.

11. Технічна документація на датчик MAX30102. URL: https://pdf1.alldatasheet.com/datasheet-pdf/view/859400/MAXIM/ MAX30102.html (дата звернення: 10.10.2022).

12. Технічна документація на мікроконтролер ARDUINO UNO REV3. URL: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf (дата звернення: 10.10.2022).

13. Бібліотека SparkFun_MAX3010x_Sensor_Library. URL: https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library (дата звернення: 10.10.2022).

14. Qt Designer: https://www.qt.io/ (дата звернення: 20.09.2022).

15. PyQtGraph: https://www.pyqtgraph.org/ (дата звернення: 22.09.2022).

ДОДАТОК А

КОД ПРОГРАМИ ДЛЯ ЗЧИТУВАННЯ ДАНИХ З ДАТЧИКА

```
#include <Wire.h>
#include "MAX30105.h"

MAX30105 particleSensor;

#define MAX_BRIGHTNESS 255
const int numReadings = 5;
int calibrateReadings[numReadings];

void setup()
{

 //MAX30105 init
 Serial.begin(115200);
 particleSensor.begin(Wire, I2C_SPEED_FAST);
 particleSensor.shutDown();

}

void loop()
{

 byte incoming = Serial.read();

 if (incoming == 48)
 {
  byte ledMode = 2;
  byte ledBrightness = 30;
  int pulseWidth = 118;
  int sampleRate = 400;
  int adcRange = 8192;
  byte sampleAverage = 1;
  long average_8192 = 131000;
  long average_min_8192 = 124000;
  long average_max_8192 = 138000;
            particleSensor.setup(ledBrightness, sampleAverage, ledMode,
sampleRate, pulseWidth, adcRange);

  while(true)
  {
```

```
    delay(1000);
    long red_read = particleSensor.getRed();
    if (red_read <= average_min_8192)
    {
      ledBrightness = ledBrightness + 2;
      particleSensor.setup(ledBrightness);
      continue;
    }
    else if (red_read >= average_max_8192)
    {
      ledBrightness = ledBrightness - 2;
      particleSensor.setup(ledBrightness);
      continue;
    }
    else
    {
      break;
    }
  }


  while(true)
  {

    byte incoming = Serial.read();
    if (incoming == 49)
    {
      break;
    }

    Serial.print(particleSensor.getRed());
    Serial.print(" "); Serial.println(particleSensor.getIR());
    delay(1);

  }

  particleSensor.shutDown();

 }
}
```

ДОДАТОК Б

КОД ІНТЕРФЕЙСУ ПРОГРАМИ

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(590, 477)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout_3 = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout_3.setObjectName("gridLayout_3")
        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
        self.tabWidget.setObjectName("tabWidget")
        self.Serial = QtWidgets.QWidget()
        self.Serial.setObjectName("Serial")
        self.gridLayout = QtWidgets.QGridLayout(self.Serial)
        self.gridLayout.setObjectName("gridLayout")
        self.splitter_2 = QtWidgets.QSplitter(self.Serial)
        self.splitter_2.setOrientation(QtCore.Qt.Horizontal)
        self.splitter_2.setObjectName("splitter_2")
        self.splitter = QtWidgets.QSplitter(self.splitter_2)
        self.splitter.setOrientation(QtCore.Qt.Vertical)
        self.splitter.setObjectName("splitter")
        self.widget = QtWidgets.QWidget(self.splitter)
        self.widget.setObjectName("widget")
        self.horizontalLayout = QtWidgets.QHBoxLayout(self.widget)
        self.horizontalLayout.setContentsMargins(0, 0, 0, 0)
        self.horizontalLayout.setObjectName("horizontalLayout")
```

```
        self.com_list = QtWidgets.QComboBox(self.widget)
        self.com_list.setObjectName("com_list")
        self.horizontalLayout.addWidget(self.com_list)
        self.open_com = QtWidgets.QPushButton(self.widget)
        self.open_com.setObjectName("open_com")
        self.horizontalLayout.addWidget(self.open_com)
        self.close_com = QtWidgets.QPushButton(self.widget)
        self.close_com.setObjectName("close_com")
        self.horizontalLayout.addWidget(self.close_com)
            spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout.addItem(spacerItem)
        self.widget1 = QtWidgets.QWidget(self.splitter)
        self.widget1.setObjectName("widget1")
        self.horizontalLayout_3 = QtWidgets.QHBoxLayout(self.widget1)
        self.horizontalLayout_3.setContentsMargins(0, 0, 0, 0)
        self.horizontalLayout_3.setObjectName("horizontalLayout_3")
        self.start_measurment = QtWidgets.QPushButton(self.widget1)
        self.start_measurment.setObjectName("start_measurment")
        self.horizontalLayout_3.addWidget(self.start_measurment)
        self.stop_measurment = QtWidgets.QPushButton(self.widget1)
        self.stop_measurment.setObjectName("stop_measurment")
        self.horizontalLayout_3.addWidget(self.stop_measurment)
        self.file_save = QtWidgets.QPushButton(self.widget1)
        self.file_save.setObjectName("file_save")
        self.horizontalLayout_3.addWidget(self.file_save)
            spacerItem1 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_3.addItem(spacerItem1)
        self.Graph = PlotWidget(self.splitter_2)
        self.Graph.setMaximumSize(QtCore.QSize(548, 341))
```

```
        self.Graph.setObjectName("Graph")
        self.gridLayout.addWidget(self.splitter_2, 0, 0, 1, 1)
        self.tabWidget.addTab(self.Serial, "")
        self.From_file = QtWidgets.QWidget()
        self.From_file.setObjectName("From_file")
        self.gridLayout_2 = QtWidgets.QGridLayout(self.From_file)
        self.gridLayout_2.setObjectName("gridLayout_2")
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        self.chosen_file = QtWidgets.QLineEdit(self.From_file)
        self.chosen_file.setObjectName("chosen_file")
        self.horizontalLayout_2.addWidget(self.chosen_file)
        self.verticalLayout_2 = QtWidgets.QVBoxLayout()
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.choose_file = QtWidgets.QPushButton(self.From_file)
        self.choose_file.setObjectName("choose_file")
        self.verticalLayout_2.addWidget(self.choose_file)
        self.get_graph = QtWidgets.QPushButton(self.From_file)
        self.get_graph.setObjectName("get_graph")
        self.verticalLayout_2.addWidget(self.get_graph)
        self.get_graph_SpO2 = QtWidgets.QPushButton(self.From_file)
        self.get_graph_SpO2.setObjectName("get_graph_SpO2")
        self.verticalLayout_2.addWidget(self.get_graph_SpO2)
        self.horizontalLayout_2.addLayout(self.verticalLayout_2)
        self.gridLayout_2.addLayout(self.horizontalLayout_2, 0, 0, 1, 1)
            spacerItem2 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Minimum)
        self.gridLayout_2.addItem(spacerItem2, 0, 1, 1, 1)
        self.tabWidget.addTab(self.From_file, "")
        self.gridLayout_3.addWidget(self.tabWidget, 0, 0, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget)
```

```
        self.retranslateUi(MainWindow)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)


    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "SpO2_Serial"))
        self.open_com.setText(_translate("MainWindow", "OPEN"))
        self.close_com.setText(_translate("MainWindow", "CLOSE"))
        self.start_measurment.setText(_translate("MainWindow", "START MEASURMENT"))
        self.stop_measurment.setText(_translate("MainWindow", "STOP MEASURMENT"))
        self.file_save.setText(_translate("MainWindow", "SAVE"))
          self.tabWidget.setTabText(self.tabWidget.indexOf(self.Serial), _translate("MainWindow",
"Serial"))
        self.choose_file.setText(_translate("MainWindow", "CHOOSE FILE"))
        self.get_graph.setText(_translate("MainWindow", "GET PHOTOPLETHYSMOGRAM"))
        self.get_graph_SpO2.setText(_translate("MainWindow", "GET GRAPHIC SpO2"))
                                    self.tabWidget.setTabText(self.tabWidget.indexOf(self.From_file),
_translate("MainWindow", "From file"))
from pyqtgraph import PlotWidget
```

ДОДАТОК В
ПОВНИЙ КОД ПРОГРАМИ

```
from PyQt5 import QtWidgets, QtGui
from PyQt5.QtSerialPort import QSerialPort, QSerialPortInfo
from PyQt5.QtCore import QIODevice
from Main_Window import Ui_MainWindow
from datetime import datetime

import os
import sys
import PyQt5
import time
import pyqtgraph as pg
import numpy as np
import math as mt


class main_window(QtWidgets.QMainWindow):
def __init__(self):
super(main_window, self).__init__()
self.ui = Ui_MainWindow()
self.ui.setupUi(self)

#Serial tab
def Open_COM():
serial.setPortName(self.ui.com_list.currentText())
serial.open(QIODevice.ReadWrite)

def Close_COM():
serial.close()

def setYRange():
self.SpO2_Graph.enableAutoRange(axis='y', enable = True)
self.SpO2_Graph.setAutoVisible(y=True)

def SpO2_average_show(start_value):
self.window_spo2 = PyQt5.QtWidgets.QWidget()
self.verticalLayout = PyQt5.QtWidgets.QVBoxLayout(self.window_spo2)

self.my_label = PyQt5.QtWidgets.QLabel()
self.verticalLayout.addWidget(self.my_label)
self.my_label.setText('SpO2:')

self.SpO2_value = PyQt5.QtWidgets.QDoubleSpinBox()
self.verticalLayout.addWidget(self.SpO2_value)
self.SpO2_value.setDecimals(3)

if start_value == 0:
```

```
self.window_spo2.show()
if start_value == 1:
self.window_spo2.close()

def Start_measurment():
data_file = open("DATA", "w+")
red_data = []
ired_data = []
start_value = "0"
print(start_value.encode())
serial.write(start_value.encode())
print("Start =", datetime.now())
SpO2_average_show(0)
while serial.waitForReadyRead(10000):
pg.QtGui.QApplication.processEvents()
rx_data = serial.readLine()
while not '\\n'in str(rx_data):
serial.waitForReadyRead(1)
cash = serial.readLine()
if not not str(cash, 'utf-8'):
rx_data = (str(rx_data, 'utf-8')+str(cash, 'utf-8')).encode()
rx_data_string = str(rx_data, 'utf-8').strip()
res_rx_data = rx_data_string.split(" ")
red_data.append(res_rx_data[0])
ired_data.append(res_rx_data[1])
data_file.write(rx_data_string+"\n")
if len(red_data) % 25 == 0:
X_X = np.empty(25, dtype = object)
Y_Y = np.empty(25, dtype = object)
X_Y = np.empty(25, dtype = object)
for i, item in enumerate(red_data):
red_data[i] = int(item)
for i, item in enumerate(ired_data):
ired_data[i] = int(item)
for i in range(25):
X_X[i] = (red_data[i])**2
Y_Y[i] = (ired_data[i])**2
X_Y[i] = red_data[i] * ired_data[i]
x_max = max(red_data[0:25])
y_max = max(ired_data[0:25])
X = np.sum(red_data[0:25])/x_max
Y = np.sum(ired_data[0:25])/y_max
XX = np.sum(X_X)/(x_max**2)
YY = np.sum(Y_Y)/(y_max**2)
XY = np.sum(X_Y)/(x_max * y_max)
A12 = (X*Y - 25*XY)/((X**2) - 25*XX)
SpO2 = 1.12 - (A12/3)
R1 = (XY - (X*Y)/25)/(mt.sqrt((XX - (X**2)/25)) * mt.sqrt((YY - (Y**2)/25)))
print(datetime.now(), R1)
if R1 >= 0.9:
if len(self.SpO2_data) == 10:
self.SpO2_data = self.SpO2_data[1:]
```

```
self.log_time = self.log_time[1:]
self.SpO2_data.append(SpO2)
self.log_time.append(time.time())
self.curve_position.setData(self.log_time, self.SpO2_data, clear = True)
if len(self.SpO2_data) % 5 == 0:
SpO2_average = np.sum(self.SpO2_data)/len(self.SpO2_data)
self.SpO2_value.setValue(SpO2_average)

red_data.clear()
ired_data.clear()

def Stop_measurment():
pg.QtGui.QApplication.processEvents()
SpO2_average_show(1)
stop_value = "1"
serial.write(stop_value.encode())
print("Stop =", datetime.now())
self.SpO2_data.clear()
self.log_time.clear()

def Save_file():
URL = os.getcwd()
str_URL = (str(URL)+'/DATA').replace("\\", "/")
print(str_URL)
name = QtWidgets.QFileDialog.getSaveFileName(self, 'Save File')
str_name = str(name)
str_file_name = str_name[2:-19]
file = open(str_file_name,'w')
with open(str_URL) as f:
for line_number, line in enumerate(f):
if line:
line_text = str(line)
file.write(line_text)
line_number += 1
f.close()
file.close()



#Serial and ComboBox
serial = QSerialPort()
serial.setBaudRate(115200)
port_list = []
ports = QSerialPortInfo().availablePorts()
for port in ports:
port_list.append(port.portName())
self.ui.com_list.addItems(port_list)

#Arrays with data
self.log_time = []
self.SpO2_data = []
```

```
#Plot Items
self.SpO2_Graph = self.ui.Graph
self.SpO2_Graph.setAxisItems(axisItems = {'bottom': pg.DateAxisItem()})
self.SpO2_Graph.setTitle("SpO2 measurment", color="r", size="15px")
self.SpO2_Graph.setLabel('left', "<span style=\"color:blue;font-size:15px\">SpO2
(0.01%)</span>")
self.SpO2_Graph.setLabel('bottom', "<span style=\"color:blue;font-size:15px\">Time</span>")
self.SpO2_Graph.showGrid(x=True, y=True)
self.SpO2_Graph.setYRange(0, 1)

self.SpO2_Graph.setAspectLocked(lock=False)
self.SpO2_Graph.setAutoVisible(y = True)
self.SpO2_Graph.enableAutoRange(axis = 'y', enable = True)
self.SpO2_Graph.setDefaultPadding(padding = 0.01)
self.SpO2_Graph.sigXRangeChanged.connect(setYRange)

#Curve
self.curve_position = self.SpO2_Graph.plot(self.log_time, self.SpO2_data)

#Buttons
self.ui.open_com.clicked.connect(Open_COM)
self.ui.close_com.clicked.connect(Close_COM)
self.ui.start_measurment.clicked.connect(Start_measurment)
self.ui.stop_measurment.clicked.connect(Stop_measurment)
self.ui.file_save.clicked.connect(Save_file)


#File tab
def Choose_file():
file_name = QtWidgets.QFileDialog.getOpenFileName()
str_file_name = str(file_name)
self.ui.chosen_file.setText(str_file_name[2:-19])

def Get_graph():
red_buf = []
ired_buf = []
file_name = self.ui.chosen_file.text()
with open(file_name, 'r') as f:
for line_number, line in enumerate(f):
if line:
line_text = str(line)
splitted_text = line_text.split()
red_buf.append(splitted_text[0])
ired_buf.append(splitted_text[1])
line_number += 1
for i, item in enumerate(red_buf):
red_buf[i] = int(item)
for i, item in enumerate(ired_buf):
ired_buf[i] = int(item)
length = len(red_buf)
x = np.arange(length)
graph = pg.plot(title = 'Photoplethysmogram')
```

```
graph.plot(x, red_buf, pen = 1)
graph.plot(x, ired_buf, pen = 2)

def Get_graph_SpO2():
red_buf = []
ired_buf = []
SpO2_buf = []
file_name = self.ui.chosen_file.text()
with open(file_name, 'r') as f:
for line_number, line in enumerate(f):
if line:
line_text = str(line)
splitted_text = line_text.split()
red_buf.append(splitted_text[0])
ired_buf.append(splitted_text[1])
line_number += 1
for i, item in enumerate(red_buf):
red_buf[i] = int(item)
for i, item in enumerate(ired_buf):
ired_buf[i] = int(item)
length = len(red_buf)
max_i = length//200
for i in range(1, max_i):
max_range = i*200
if i == 1:
continue
else:
X_X = np.empty(length, dtype = object)
Y_Y = np.empty(length, dtype = object)
X_Y = np.empty(length, dtype = object)
for i in range(max_range-200,max_range):
X_X[i] = (red_buf[i])**2
Y_Y[i] = (ired_buf[i])**2
X_Y[i] = red_buf[i] * ired_buf[i]
x_max = max(red_buf[max_range-200:max_range])
y_max = max(ired_buf[max_range-200:max_range])
X = np.sum(red_buf[max_range-200:max_range])/x_max
Y = np.sum(ired_buf[max_range-200:max_range])/y_max
XX = np.sum((X_X[max_range-200:max_range]))/(x_max**2)
YY = np.sum((Y_Y[max_range-200:max_range]))/(y_max**2)
XY = np.sum((X_Y[max_range-200:max_range]))/(x_max * y_max)
A12 = (X*Y - 200*XY)/((X**2) - 200*XX)
SpO2 = 1.12 - (A12/3)
R1 = (XY - (X*Y)/200)/(mt.sqrt((XX - (X**2)/200)) * mt.sqrt((YY - (Y**2)/200)))
if R1 >= 0.9: SpO2_buf.append(SpO2)
x = np.arange(len(SpO2_buf))
graph = pg.plot(title = 'SpO2')
graph.setTitle("SpO2 measurment", color="r", size="15px")
graph.setLabel('left', "<span style=\"color:blue;font-size:15px\">SpO2 (0.01%)</span>")
graph.setLabel('bottom', "<span style=\"color:blue;font-size:15px\">Dimension #</span>")
graph.showGrid(x=True, y=True)
graph.plot(x, SpO2_buf, pen = 1)
```

```
self.ui.choose_file.clicked.connect(Choose_file)
self.ui.get_graph.clicked.connect(Get_graph)
self.ui.get_graph_SpO2.clicked.connect(Get_graph_SpO2)

app = QtWidgets.QApplication([])
application = main_window()
application.show()
sys.exit(app.exec())
```