МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВТАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ, ЕЛЕКТРОНІКИ ТА ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА ЕЛЕКТРОНІКИ, РОБОТОТЕХНІКИ ТЕХНОЛОГІЙ
МОНІТОРИНГУ ТА ІНТЕРНЕТУ РЕЧЕЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____Володимир ШУТКО
«___ »_____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА ОСВТІНЬОГО СТУПЕНЯ БАКАЛАВРА
ЗТ СПЕЦІАЛЬНОСТІ 153 «МІКРО- ТА НАНОСИСТЕМНА ТЕХНІКА»
ОПП «ФІЗИЧНА ТА БІОМЕДИЧНА ЕЛЕКТРОНІКА»

Тема: «Пристрій для виміру пульсу та насичення крові киснем в польових умовах.»

Виконавець
студент групи МН-405Б _____Шкут Денис Сергійович

Керівник
д.т.н., старший викладач _____Бурцева Наталія Вікторівна

Нормоконтролер
к.т.н., доцент _____ Сініцин Р.Б.

КИЇВ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій

Кафедра електроніки, робототехніки технологій моніторингу та інтернету речей

Напрям (спеціальність, ОПП): 153 «Фізична та біомедична електроніка»

(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри
_____Володимир ШУТКО
«____»_____2023 р.

ЗАВДАННЯ

на виконання дипломної роботи

Шкут Денис Сергійович

(П.І.Б., випускника)

1. Тема дипломної роботи: «Пристрій для виміру пульсу та насичення крові киснем в польових умовах»
затверджена наказом ректора від «23» березня 2019 р. № 387/ ст

2. Термін виконання роботи: з 8.05.2023 р. по 9.06.2023 р.

3. Вихідні дані роботи: технічне завдання, спеціалізоване програмне забезпечення.

4. Зміст пояснювальної записки: огляд технічної та довідкової літератури за темою проекту, розробка схемотехнического проекту портативного пульсоксиметра, створення програмного коду, список використаної літератури.

5. Перелік обов’язкового ілюстративного матеріалу: таблиці, рисунки, схеми.

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Підпис керівника
1.	Вступ	8.05 – 12.05	
2.	Обробка матеріалів за темою дипломної роботи: підручники, Інтернет-ресурси	15.05 – 26.05	
3.	Розробка схемотехнічного проекту портативного пульсоксиметра	29.05 – 6.06	
4.	Подання на кафедру. Усунення недоліків. Оформлення пояснювальної записки.	7.06 – 9.06	
5.	Електронна версія доповіді, ілюстративний матеріал доповіді	12.06 – 13.06	

7. Дата видачі завдання:

Керівник дипломної роботи _____ _____
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ _____
(підпис випусника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи: "Пристрій для виміру пульсу та насичення крові киснем в польових умовах.": 51 с., 32 рис., 11 літературних джерел.

Мета роботи: розробити схемотехнічний проект та зібрати пристрій для виміру пульсу та насичення крові киснем в польових умовах.

Методи дослідження: комп’ютерне моделювання, програмування, створення прототипу.

Пристрій як результат бакалаврської роботи можна використовувати для виявлення проблем із здоров’ям, контролю параметрів організму та надання швидкої медичної допомоги у польових умовах.

ПУЛЬСОКСИМЕТРІЯ, ПУЛЬС, САТУРАЦІЯ, МІКРОКОНТРОЛЛЕР, ДАТЧИК, ВИМІРЮВАЛЬНИЙ ПРИСТРІЙ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

RHb – дезоксигемоглобін, гемоглобін, який не містить кисню

HbO_2 – оксигемоглобін, гемоглобін, кожна молекула якого містить чотири молекули кисню

SpO_2 – сатурація, ступінь насичення гемоглобіну крові киснем

АЦП – аналого-цифровий перетворювач

МК – мікроконтролер

ПК – персональний комп’ютер

ФПГ – фотоплетізмограма

Даташит - це технічний документ, що містить детальну інформацію про конкретний електронний компонент або пристрій.

I_2C – послідовна шина даних для зв'язку інтегральних схем, розроблена фірмою Philips на початку 1980-х як проста шина внутрішнього зв'язку для створення керуючої електроніки. Використовується для з'єднання низькошвидкісних периферійних компонентів з материнською платою, вбудовуваними системами та мобільними телефонами. Назва є абревіатурою слів Inter-Integrated Circuit.

ВСТУП

Медициною доведено, що рівень кисню в крові має значний вплив на ефективність та стан здоров'я людини. Кількість кисню, що присутня в артеріальній крові, є одним з найважливіших факторів, який впливає на обмін речовин у людини і, відповідно, на її життєву активність. Недостатній рівень кисню призводить до зниження життєдіяльності, тоді як нормальні показники є свідченням гарного самопочуття.

Багато людей не приділяють уваги насиченості крові киснем, так званій сатурації (від англ. «saturation»), і не контролюють її, але низькі показники цього параметру можуть свідчити про розвиток небезпечних та серйозних захворювань, таких як діабет, утворення тромбів, загальні порушення кровообігу, а також проблеми з роботою серця та нервової системи.

Першими ознаками недостатнього рівня насичення є часті головні болі, запаморочення, непритомність, знижена концентрація та погіршення роботи мозку. Зрозуміло, що з такими симптомами якість життя суттєво погіршується, такі ознаки заважають ефективній праці, відпочинку та насолоди життям, оскільки хворій людині може знадобитися медична допомога в будь-який момент, наприклад у польових умовах.

Пульсоксиметри широко використовуються терапевтами та польовими медиками, оскільки їх показники є важливими для точної діагностики. Крім того, вони популярні серед спортсменів та всіх, хто займається важкими фізичними навантаженнями.

В польових умовах, де доступ до медичних засобів може бути обмеженим, пульсоксиметр дозволяє медичному персоналу швидко оцінити стан поранених та виявити потенційні проблеми з диханням або циркуляцією крові. Наприклад, низький рівень кисню може вказувати на крововтрату або респіраторну недостатність. Також пристрій дозволяє медикам контролювати ефективність лікування, виявляти погіршення стану пацієнта та приймати необхідні заходи для запобігання важких наслідків.

Метою даної дипломної роботи є розробка портативного пристрою для вимірювання пульсу та рівня кисню в крові, який може використовуватися для контролю цих показників в польових умовах.

РОЗДІЛ 1
ЗАГАЛЬНІ ВІДОМОСТІ

1.1. Основи пульсоксиметрії

До виникнення пульсоксиметрії та створення технічних пристроїв що дозволяють вимірювати пульс, людство користувалось пальпацією – прикладання пальця до певної частини тіла й оцінки тактильної інформації, найчастіше – кількість поштовхів (рис.1.1). Назва цього ефекту пішла від латинського «pulsus» - удар/поштовх. Частоту серцебиття вираховували рахуючи кількість поштовхів за короткий проміжок часу – хвилину або 30 секунд. [1]

Рис. 1.1

Оптична пульсоксиметрія (оксігемометрія, гемоксіметрія) – неінвазивний метод визначення ступеня насичення крові киснем та пульсу за допомогою світлових сигналів.

Повністю насичена киснем кров (100% сатурація) це та, молекули гемоглобіну якої з’єднані з чотирма молекулами кисню. Насичена киснем кров циркулює по тілу, живлячи органи людини.

Пульсоксиметрія використовує два ключових фізіологічних явища:

1. Гемоглобін в залежності від насиченості крові киснем по різному поглинає світло певної довжини хвилі при проходженні світла через довільну частину тканини.
2. Артерії та артеріоли пульсують відповідно до ударного ритму серця.

Принцип роботи оксиметрії полягає в особливостях поглинання світла дезоксигемоглобіном (RHb) і оксигемоглобіном (HbO2). RHb інтенсивно поглинає червоне світло, майже не затримуючи інфрачервоне. З іншого боку, HbO2 добре поглинає інфрачервоне випромінювання, але слабо поглинає червоне світло. За допомогою фотодетектора, який реєструє пропускання світла через ділянку тканини (наприклад, мочку вуха або палець), вимірюється співвідношення червоного (R) і інфрачервоного (IR) потоків, що проходять через кров. Це співвідношення використовується для визначення ступеня насиченості гемоглобіну киснем, відображеного як сатурація (SpO2). Значення сатурації гемоглобіну крові киснем відображається на екрані пристрою (рис. 1.2). За допомогою цього методу, оксиметр дозволяє неінвазивним способом швидко та зручно виміряти насиченість киснем в крові пацієнта.[1][3]

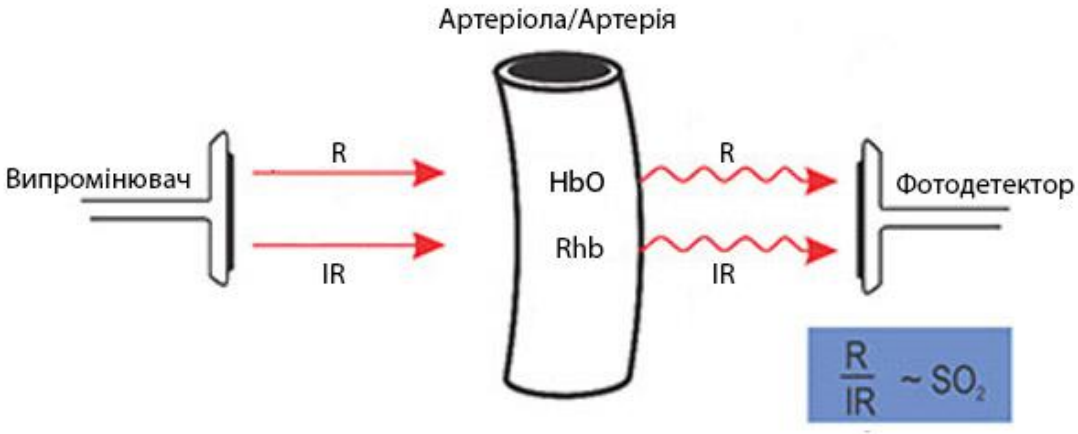


Рис.1.2

В результаті пульсації артерій і артеріол утворюється пульсова хвиля. Пульсації є наслідком викиду певного об'єму крові в аорту лівим шлуночком.

Результатом реєстрації таких коливань кровонаповнення є фотоплетизмограми (ФПГ). Аналіз ФПГ дозволяє визначити частоту серцевих скорочень і оцінити якість периферичного кровотоку (рис. 1.3). [1]

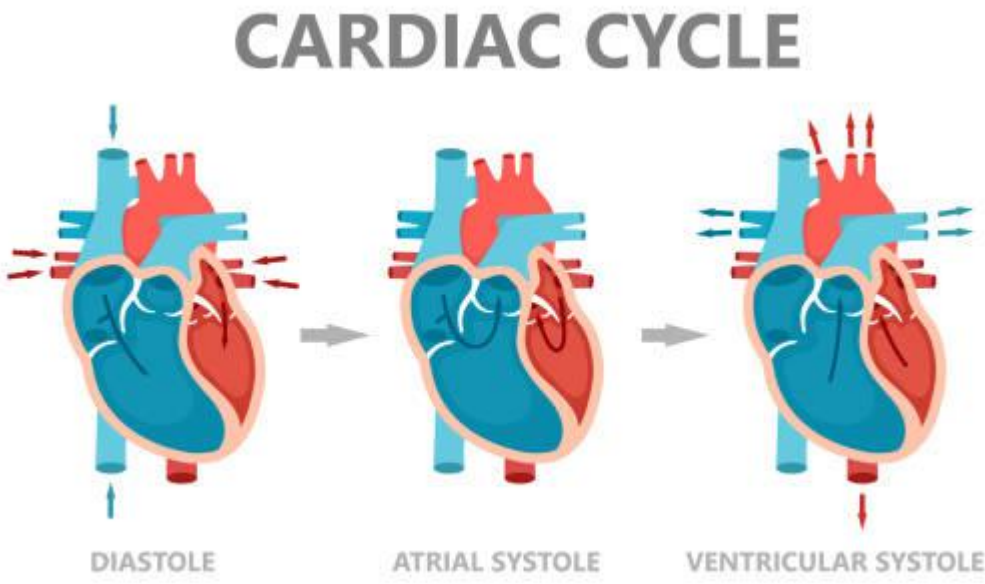


Рис.1.3

1.2. Історія оксиметрії

Метод оксиметрії, який використовується для вимірювання насичення киснем в крові, має свою історію, що починається з досліджень, проведених у кінці 19 століття і продовжується до сьогодення. Оскільки я не маю інформації про події після 1990 року, згаданий в тексті обсяг продажів пульсоксиметрів досяг 65 тисяч одиниць.

У 1874 році Вірордт спостерігав, що потік червоного світла, яке проходило через кисть, слабшав після накладення джгута. Це була перша спроба гемоксиметрії, або вимірювання кількості кисню в крові.[4]

Протягом 1930-1960-х років було багато спроб створити пристрій для швидкого виявлення гіпоксемії (недостатньої оксигенації організму). Карл Меттес у Лейпцигу (1936 рік) та Глен Міллікан у Кембриджі (1940 рік) створили пристрої, спрямовані на діагностику гіпоксії у пілотів. Проте, ці

пристрої були великими та вимагали складного обслуговування, тому їх застосування було обмеженим.[4]

У 1972 році Такуо Аоягі, японський інженер, виявив, що коливання абсорбції світла, викликані пульсацією артеріол, можна використовувати для розрахунку насичення артеріальної крові киснем. Завдяки розвитку технологій, вже в 1975 році був випущений перший мобільний неінвазивний пульсоксиметр, який дозволяв тривалий моніторинг за насиченням крові киснем. Цей прилад не потребував калібрування, але використовував систему світлофільтрів як джерело світла. Хоча він не знайшов значного успіху на ринку, цей прилад положив початок використанню оксиметрії.[4]

Пізніше Скотт Вілбер, американський дослідник, використав принцип Т. Аоягі, але замінив систему світлофільтрів на світлодіоди, що дозволило створити легкий та компактний вушний датчик для вимірювання насичення крові киснем.

Згаданий обсяг продажів пульсоксиметрів до 1990 року свідчить про широке застосування цього методу та інтерес до нього з боку багатьох фірм і лікарів. Завдяки розвитку технологій оксиметрія стала широко доступним інструментом для неінвазійного вимірювання насичення крові киснем, що використовується в клінічній практиці, вдома та у багатьох медичних сферах.[4]

1.3. Механізм роботи датчика

Основним принципом дії пульсоксиметра є вимірювання поглинання світла кров'ю.

Пульсоксиметр складається з трьох основних компонентів: датчика зі світлодіодами, мікропроцесора і дисплея. Датчик зі світлодіодами випромінює світло на двох різних хвилях - інфрачервоній і червоній (рис.1.4.). Це світло проходить через тканини, а кров, що містить гемоглобін, поглинає певну частину світла. [2]

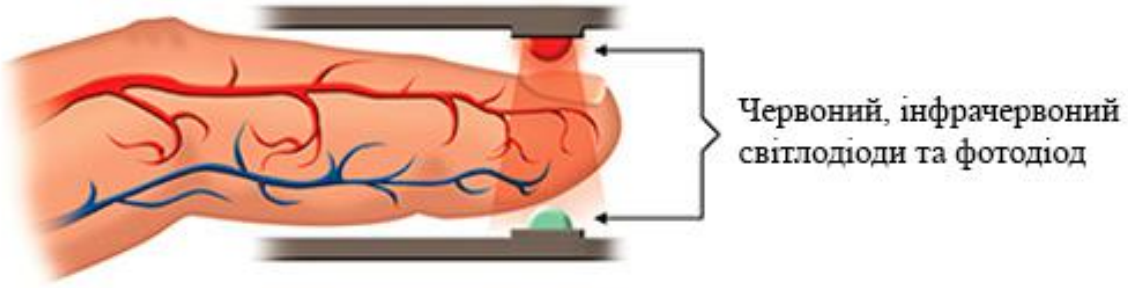


Рис.1.4

Ступінь поглинання світла гемоглобіном залежить від ступеня насиченості киснем в крові. Коли кров багата на кисень, світло поглинається меншим чином, а коли кисень у крові менш насичений, поглинання світла збільшується. Мікропроцесор обробляє отримані дані про поглинання світла і розраховує показник насиченості киснем (сатурацію) крові.[3]

Показники сатурації та частоти пульсу відображаються на дисплеї пульсоксиметра у вигляді цифрових значень або графіків. Показник сатурації зазвичай виводиться протягом 5-20 секунд після початку вимірювання. Частота пульсу розраховується на основі кількості світлодіодних циклів та впевнених сигналів, що спостерігаються за одиницю часу.

Нормальним показником сатурації киснем вважається 95-98%. За допомогою пульсоксиметра можна оцінювати функції дихальних органів та виявляти дихальну недостатність, коли показник сатурації падає нижче 95%.[3]

Для правильного використання пульсоксиметра важливо дотримуватися інструкцій, які зазвичай детально описані в посібнику з використання приладу. Рекомендується проводити виміри в затемнених приміщеннях, а пацієнт, якому проводяться виміри, повинен перебувати в стані спокою. [3]

1.4. Сфера застосування пульсоксиметрії

Пульсоксиметри застосовують для постійного контролю, та для одноразового вимірювання рівня насичення гемоглобіну артеріальної крові киснем (сатурації) і пульсу.

Неінвазивність методу дозволяє швидко вимірювати необхідні параметри, що полегшує роботу медичного персоналу. Прилад можна використовувати незалежно від притомності людини, що допомагає відслідковувати стан пацієнта навіть під час анестезії або ж у разі втрати свідомості у результаті хвороби.

Сфери застосування пульсоксиметрії:

- ✓ Кардіологія: використовується для моніторингу пульсу та насичення крові киснем у пацієнтів з серцевими захворюваннями, такими як хвороба серця, ішемічна хвороба серця, аритмії тощо.
- ✓ Анестезіологія: засіб контролю застосування анестезії та моніторингу пульсу та рівня насичення крові киснем під час операцій та процедур з укладанням пацієнтів в загальний наркоз.
- ✓ Екстрена медицина: У невідкладних ситуаціях та транспортуванні хворих пульсоксиметри використовуються для швидкого визначення пульсу та рівня насичення крові киснем, що допомагає в оцінці стану пацієнта та виборі подальшої тактики лікування.
- ✓ Педіатрія: застосовується в педіатрії для моніторингу пульсу та насичення крові киснем у новонароджих та дітей з різними захворюваннями.
- ✓ Неврологія: У деяких неврологічних захворюваннях, таких як цереброваскулярні події (інсульт), пульсоксиметри використовуються для моніторингу пульсу та рівня насичення крові киснем для оцінки стану кровопостачання мозку.

- ✓ Спорт: контроль пульсу та насичення крові киснем у спортсменів під час тренувань та змагань для оцінки фізичної витривалості та ефективності тренувань.

Застосування пульсоксиметра в дослідженнях має незаперечні переваги порівняно з альтернативними методами, які раніше є ведучими у даній області. Порівняно з інвазивним методом забору крові через прокол шкіри та використанням СО-оксиметрів та газових аналізів, пульсоксиметр забезпечує більш високу точність результатів. Крім того, в порівнянні з традиційним СО-оксиметром, пульсоксиметр є більш доступним за ціною. У випадках, коли швидкість отримання результатів має велике значення, це особливо зручно, оскільки пульсоксиметр не вимагає контрольного забору крові, що може бути незручним. Зокрема, це особливо важливо для пацієнтів, які страждають від дихальної недостатності та потребують негайного отримання результатів.[1]

Важливо мати на увазі, що пульсоксиметр є чутливим до сильного освітлення, тремтіння та рухів, оскільки ці фактори можуть спричинити штучне формування пульсоподібної кривої та вплинути на показники насичення крові. Надмірна зовнішня інтерференція може призвести до неточних результатів.[3]

1.5. Види пульсоксиметрів

Наразі існують наступні види пульсоксиметрів:

- ✓ пальцеві;
- ✓ стаціонарні;
- ✓ монітори сну;
- ✓ поясні.

Найпопулярніша модель пульсоксиметра – пальцеві або портативні пульсоксиметри невелика вага при цьому габаритах що дозволяють

використовувати їх будь-де будь-коли. За своїми можливостями майже нічим не поступається стаціонарним приладам. [5]

Портативний пульсоксиметр (рис.1.5.) складається з датчика, електроніки та дисплею, що включають світлодіоди, фотодетектори, аналогові та цифрові фільтри, підсилювачі, АЦП та мікропроцесорні чіпи. Корпус служить для захисту компонентів та забезпечення зручного використання пристрою.

Рис.1.5.

Також існує пристрій датчик якого, у вигляді прищіпки, кріплять на вухо. Такий варіант більше використовується для разових замірів, або для недовгих сесій замірів.

Існує ще одна популярна версія пульсоксиметра - поясна модель (рис.1.6.). Ці пульсоксиметри є незалежними від джерела живлення завдяки своїм компактним розмірам та низькому енергоспоживанню. Вони мають значний обсяг пам'яті, що дозволяє зберігати дані для подальшого аналізу фахівцем. Крім того, зручною особливістю цих приладів є вбудована тривожна сигналізація, яка попереджає пацієнта, якщо вимірювані показники виходять за допустимі межі. Поясні моделі пульсоксиметрів також дозволяють передавати вимірювальні дані на комп'ютер для подальшої обробки.[5]

Рис. 1.6.

Монітор сну (рис.1.7.) є важливим інструментом для проведення тривалої оксиметрії, включаючи періоди сну. Завдяки високій дискретності вимірювання, ці прилади здатні записувати дані з великою швидкістю для подальшого аналізу. Саме в період сну найкраще виявити ознаки дихальної недостатності. Моніторинг сну дозволяє отримати точний діагноз та призначити відповідне лікування.

Рис. 1.7.

Стаціонарні пульсоксиметри - це медичні пристрої, призначені для нагляду за пульсом та рівнем кисню в крові у стаціонарних умовах, таких як лікарня, клініка або домашнє оточення(рис.1.8.).

Такі пристрої мають кілька переваг порівняно з портативними версіями, зокрема вони забезпечують вищу точність вимірювань, стабільну фіксацію на

пацієнті, додаткові функціональні можливості для детального аналізу даних та надійну роботу завдяки живленню з мережі або довготривалій батарейці, що робить їх незамінними при медичному нагляді у стаціонарних умовах.[5]

Рис. 1.8.

РОЗДІЛ 2
РОЗРОБКА СТРУКТУРНОЇ СХЕМИ ПРИСТРОЮ

2.1. Процедура пульсової оксиметрії

У 70-х роках була розроблена методика пульсової оксиметрії, яка ґрунтується на принципах фотоплетізмографії. Ця методика передбачає розміщення спеціального датчика між джерелом світла та фотоприймачем на досліджуваній ділянці тканини, де відбувається вимірювання кровотоку.

Законом В-Л встановлено, що абсорбція світла прямопропорційна товщині поглинаючого шару речовини. Застосовуючи цей закон до дослідження кровотоку, можна визначити розмір судини або об'єм крові, що протікає через досліджувану ділянку тканин. Під дією артеріальної пульсації, судини звужуються або розширюються, що призводить до відповідної зміни амплітуди сигналу, отриманого з фотоприймача. Шляхом посилення та обробки цього сигналу фотоприймача можна отримати фотоплетізмограму (ФПГ), яка відображає стан кровотоку на місці розташування датчика.

Для незінвазивного вимірювання рівня сатурації крові, використовується фотоплетизмографічний датчик, який розміщується на певній ділянці тканини з артеріальними судинами. У такому випадку сигнал з датчика, який пропорційний поглинанню світла, що проходить через тканини, містить дві компоненти: пульсуючу складову, що залежить від зміни об'єму артеріальної крові під час кожного серцевого скорочення, і постійну "базову" складову, що визначається оптичними властивостями шкіри, венозної і капілярної крові, а також інших тканин на досліджуваній ділянці.

Для вимірювання сатурації крові застосовується метод двопроменевої спектрофотометрії. В процесі вимірювання абсорбції світла застосовується два джерела випромінювання з різними спектральними характеристиками. Вимірювання проводиться у моменти систолічного викиду, коли амплітуда сигналу датчика досягає максимального значення для двох довжин хвиль випромінювання.

Для досягнення найвищої точності вимірювання насиченості киснем, необхідно вибирати довжини хвиль випромінювання джерел, де спостерігається найбільша різниця у поглинанні світла оксигемоглобіном і гемоглобіном. Ці критерії задовольняють червона і ближня інфрачервона області спектра випромінювання.

При використанні довжини хвилі 660 нм (червона область) гемоглобін поглинає близько 10 разів більше світла, ніж оксигемоглобін, тоді як при хвилі 940 нм (інфрачервона область) поглинання оксигемоглобіну перевищує поглинання гемоглобіну.

З метою покращення точності виміру сатурації методом пульсової оксиметрії застосовується нормування сигналів поглинання світла, для чого ми вимірюємо постійну складову під час діастоли, а також знаходиться амплітуда

Для підвищення точності визначення сатурації методом пульсової оксиметрії використовується нормування сигналів поглинання світла, для чого вимірюється постійна складова в моменти діастоли $A_{пост}$ і знаходиться відношення амплітуди пульсуючого складової $A_{пуьс}$ до величини $A_{пост}$:

$$A_{норм} = \frac{A_{пульс}}{A_{пост}}$$

Для кожної довжини хвилі випромінювання виконується ця процедура. Нормоване значення поглинання не залежить від інтенсивності світлодіодів, а визначається виключно оптичними властивостями живої тканини.

За наступною формулою розраховується значення сатурації шляхом обчислення відношення нормованих величин поглинання світла для обох довжин хвиль.

$$R = \frac{\left(\frac{A_{пульс}}{A_{пост}}\right)_{ч}}{\left(\frac{A_{пульс}}{A_{пост}}\right)_{іч}}$$

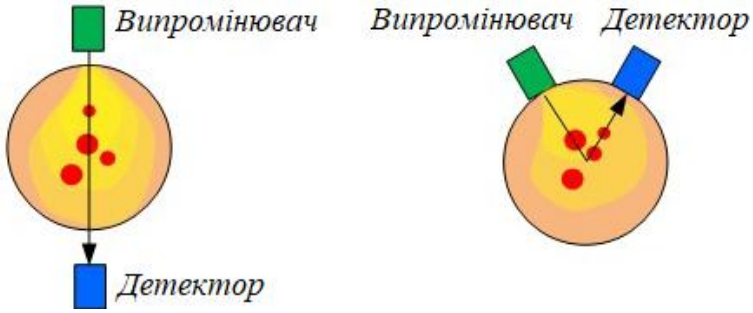
де індекс ч – поглинання в червоній, іч – в інфрачервоній області спектра.

Величина відношення R не залежить від оптичних характеристик шкіри, належних тканин, а визначається оптичними властивостями артеріального викиду крові, що визначає високу точність вимірювання сатурації в пульсоксиметрії. [5]

2.2. Особливості побудови датчиків пульсоксиметрів

Датчик пульсоксиметра на основі фотоплетізмографії складається з двох світловипромінюючих діодів, які працюють у "червоному" та "інфрачервоному" спектральних діапазонах, а також широкосмугового фотоприймача. Конструктивно датчик розташовується на поверхні тіла людини, і світло, випромінене діодами, проходить через тканини, які включають артеріальну судину, і потім потрапляє на фотоприймач після зменшення інтенсивності.

У приладах зазвичай використовується один із двох типів датчиків, перший аналізує випромінювання світлодіодів, що проходять через тканини (рис.2.1,а), а другий - випромінювання, відбите від досліджуваних тканин (рис.2.1,б).



а) б)

Рис.2.1

Перший тип датчиків закріплюється на кінчику пальця руки/ноги або мочки вуха пацієнта. Другий тип прикладають до периферійних ділянок тіла, таких як палець, вухо або нога. Найпоширенішим місцем прикладання є кінчик пальця, зазвичай вказівного або середнього пальця. Це регіон має достатньої товщини тканини і є відносно доступним для прикладання датчика пульсоксиметра.

Для досягнення точного вимірювання насиченості крові киснем та пульсу необхідно використовувати світлодіоди з малою розкидом центральної довжини хвилі в області червоного і інфрачервоного випромінювання. Вимоги до цих довжин хвиль полягають у знаходженні червоного діапазону в межах 660 ± 5 нм і інфрачервоного діапазону в межах 940 ± 10 нм (за рис. 40). Однак, варто зазначити, що використання світлодіодів з такими характеристиками є необхідним для забезпечення високої чутливості та точності пульсоксиметра при вимірюванні насиченості крові киснем та пульсу.

Датчики пульсоксиметрів використовують кремнієві фотодіоди як фотоприймачі. Ці фотодіоди мають високу чутливість до світла в області "червоного" і "інфрачервоного" діапазонів випромінювання, а також характеризуються швидким реагуванням та низьким рівнем шуму.[5]

2.3. Схема пульсоксиметра

Для побудови пристрою нам знадобляться наступні компоненти:

1. Датчик пульсу та насиченості крові киснем MAX30102: з його допомогою пристрій зможе вимірювати необхідні парметри.
2. Плата Arduino NANO із мікроконтролером ATmega328p: Займається обробкою сигналів та виведенням результатів на дисплей.

3. Індикатор OLED 0.96 128x64: Оброблені дані виводяться на цей дисплей.
4. Кнопка тактильна: Використовується для керування роботи пристрою.

Тепер розглянемо підключення елементів пристрою до Arduino Nano. Плата має різноманітні піни та інтерфейси, які дозволяють підключати різні елементи та периферійні пристрої. Основні методи підключення елементів включають використання цифрових та аналогових входів, цифрових виходів, а також спеціалізованих інтерфейсів, таких як UART, SPI та I_2C.[7]

Піни SLC та SDA датчика MAX30102 та OLED індикатору підключаємо до аналогових входів плати Ардуіно A5 та A4 відповідно. Через них, мікроконтролер буде по-перше, приймати дані з датчика серцебиття та насиченості крові киснем та по-друге, відправляти сигнал на дисплей що буде виводити необхідні отримані дані користувачу.

Під’єднаємо піни VCC та GND до дисплею та датчику, тим самим, плата зможе живити наші елементи. Також заземляємо тактильну кнопку.

Кнопка під'єднується до Ардуіно за допомогою цифрового входу D3, оскільки цифрові входи працюють з дискретними значеннями логічних 0 або 1. Кнопка може змінювати свій стан між двома можливими значеннями: вона може бути натиснута (що відповідає логічному 1) або не натиснута (що відповідає логічному 0)(рис.2.2.). Прототип зібраний(рис.2.3.)

Пристрій можна живити двома основними способами:

1. За допомогою портативного акумулятора: звичайні портативні акумулятори, що не оснащені функцією QuickCharge (функція швидкої зарядки за рахунок підвищеної напруги), видають 5.0-5.1 В, що достатньо для живлення пристрою.
2. Від батарейки: якщо підключити виходи батарейки до виводів VIN та GND на платі Arduino Nano, то пристрій буде працювати. Важливо

пам'ятати, що напруга батарейок повинна бути від 7 до 12 В, щоб успішно живити пристрій.

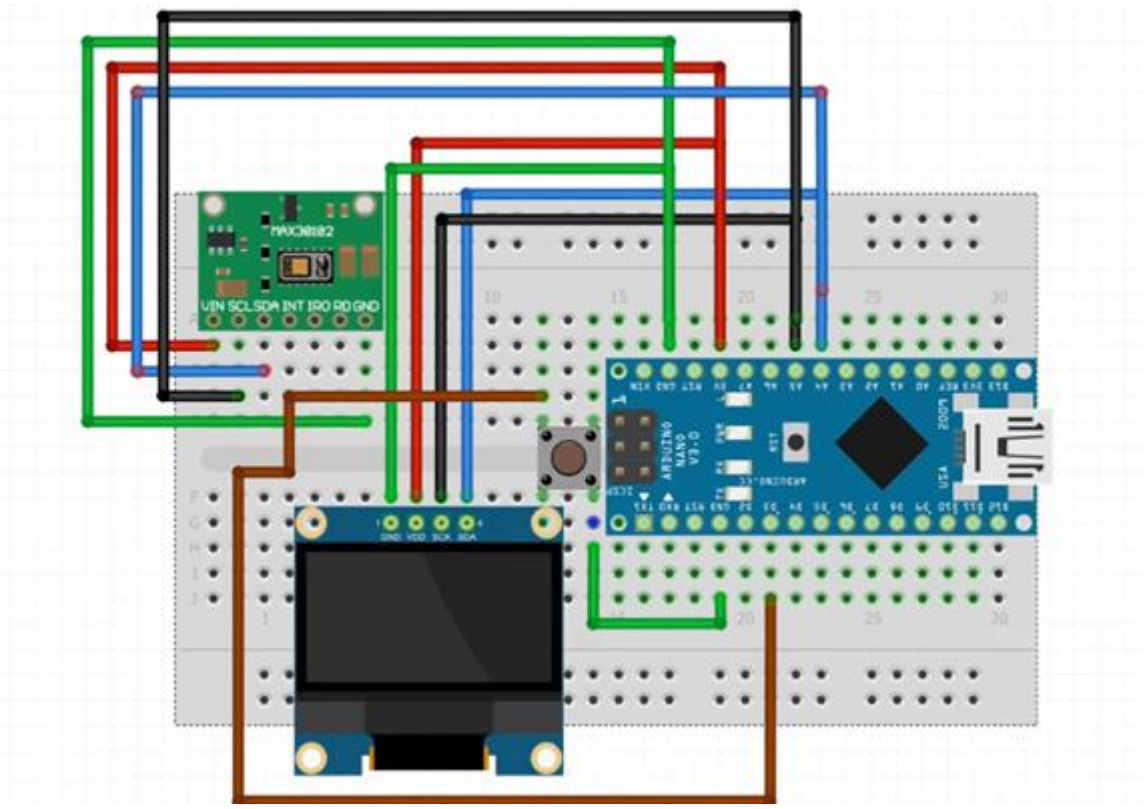


Рис. 2.2. Схема підключеня елементів пристрою

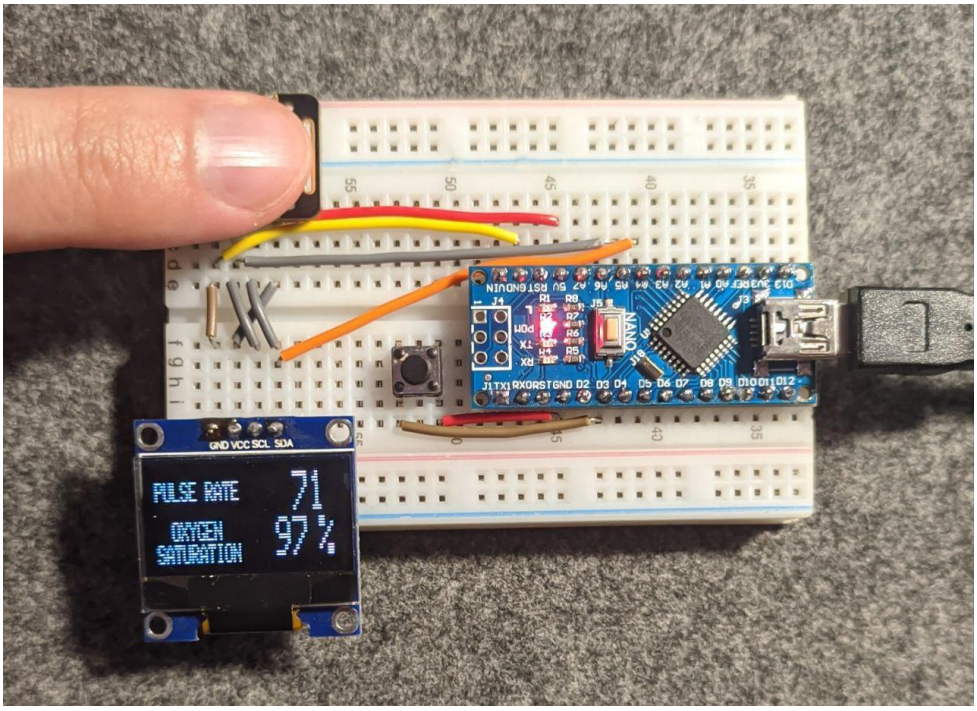


Рис.2.3. Фото зібраного прототипу

2.4. Датчик MAX30102

MAX30102 (рис.2.4.) - це інтегрований пульсоксиметрійний модуль та моніторингу серцевого ритму. Він включає в себе внутрішні світлодіоди червоного та інфрачервоного випромінювання, фотодетектор, оптичні елементи та електроніку з фільтрами навколишнього світла. MAX30102 забезпечує комплексне системне рішення для полегшення процесу розробки для мобільних і натільних пристроїв. MAX30102 працює від одного джерела живлення 1,8 В і окремого джерела живлення 3,3 В для внутрішніх світлодіодів.[6]

Зв'язок здійснюється через стандартний I_2C-сумісний інтерфейс. Модуль може бути вимкнений за допомогою програмного забезпечення з нульовим струмом очікування, що дозволяє шинам живлення завжди залишатися під напругою.

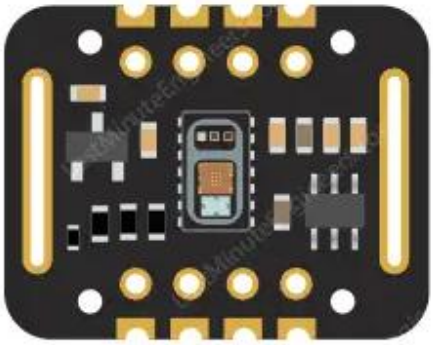


Рис.2.4. Зовнішній вигляд датчика

Функціональна схема датчика(рис.2.5.):

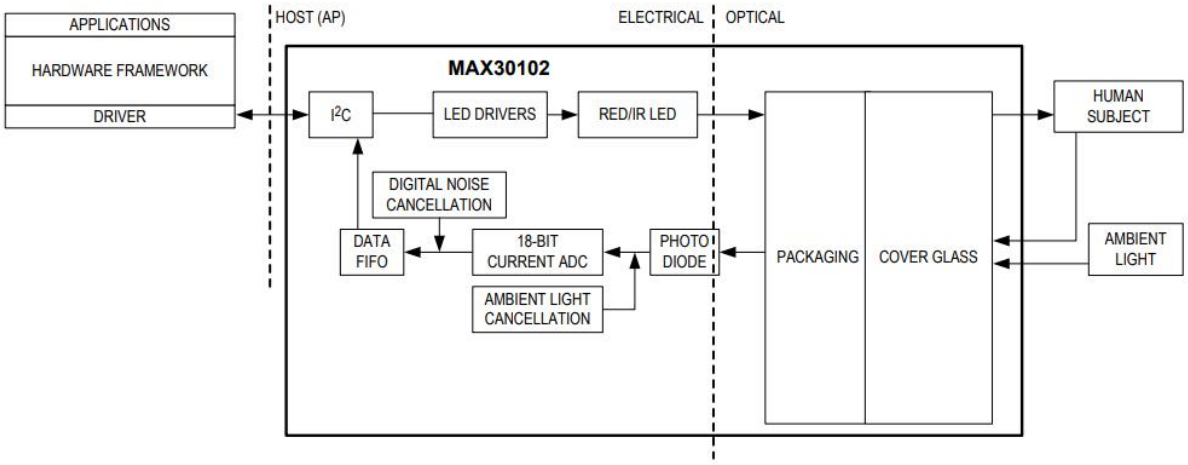


Рис. 2.5.

Процес вимірювання починається зі світлодіодів, які випромінюють світло через шкіру користувача. Частина світла поглинається гемоглобіном в крові, а решта світла розсіюється або відбивається від навколишніх тканин. Фотодіод на датчику сприймає світло, яке пройшло через тканини, і генерує електричний сигнал, який залежить від кількості поглинутого світла. На рис. 2.6. показано приклад синхронізації роботи світлодіодів для режиму роботи датчика MAX30102 з частотою дискретизації 1 кГц.[6]

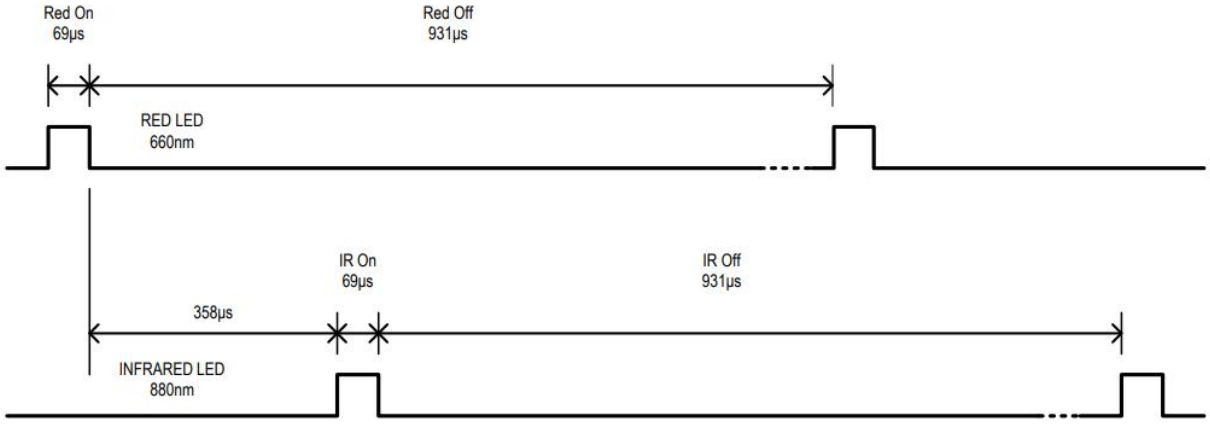


Рис. 2.6.

Далі, сигнал з фотодіода проходить через внутрішній аналого-цифровий перетворювач (ADC), який перетворює аналоговий сигнал у цифровий формат для подальшої обробки. Вбудований алгоритм датчика

аналізує цифрові дані, отримані з ADC, і визначає пульс та рівень кисню в крові. Результати можуть передаватися до зовнішнього пристрою або мікроконтролера через спеціальні інтерфейси, такі як I_2C або SPI.

Датчик MAX30102 є надзвичайно ефективним та точним вимірювальним пристроєм, здатним працювати навіть при низьких рівнях сигналу та в умовах руху. Цей датчик широко використовується в медичних пристроях, фітнес-трекерах, домашніх моніторах здоров'я та інших пристроях, що вимагають нагляду за пульсом та рівнем кисню в крові користувача.[6]

Основні переваги датчика перед аналогами на ринку:

- Розмір: Мініатюрний 14-контактний оптичний модуль 5,6 мм x 3,3 мм x 1,55 мм та інтегроване захисне скло для оптимальної та надійної роботи.
- Енергоспоживання: Наднизьке енергоспоживання для мобільних пристроїв. Програмована частота дискретизації і струм світлодіода для енергозбереження. Монітор серцевого ритму з низьким енергоспоживанням (< 1 мВт). Наднизький струм вимкнення (0,7 мкА, типовий).
- Швидкість роботи: виведення даних через інтерфейс I_2C.
- Висока частота дискретизації: забезпечує актуальність вимірюваних даних.
- Точність: вбудовані фільтри навколишнього світла та цифрового шуму забезпечують більшу точність результатів.
- Робочий діапазон температур: від -40°C до +85°C.

2.5. Негативні фактори що погіршують точність вимірювання

Точність вимірювання може бути позначена різними перешкодами, які виникають через такі електричні, оптичні та фізіологічні фактори.

Рух: Рух тіла є одним з основних факторів, які можуть спотворити сигнал пульсу та рівня кисню в крові. Переміщення частин тіла під час вимірювання може призвести до зміни контакту між датчиком і шкірою, що може призвести до артефактів у сигналі. Неправильне надання достатнього тиску на датчик або недостатня контактність можуть впливати на якість сигналу і призводити до неточностей у вимірюваннях. Це може статися, наприклад, якщо датчик недостатньо притискається до шкіри або якщо є проміжок повітря між датчиком і шкірою. Рух також може створювати додаткові шуми, які ускладнюють аналіз отриманого сигналу.

Освітлення: Сильне освітлення, особливо безпосереднє сонячне світло або штучне освітлення, може впливати на світлочутливі компоненти датчика, такі як фотодіод. Це може призводити до неправильного вимірювання пульсу та насичення крові киснем, оскільки сильне світло може спотворити отримані значення.

Вплив навколишнього середовища: Навколишнє середовище може бути джерелом інших джерел світла, електромагнітних перешкод або інших пристроїв, які можуть впливати на точність вимірювання. Наявність інших джерел світла може призводити до спотворення сигналу датчика. Електромагнітні перешкоди можуть впливати на електронні компоненти датчика і призводити до помилок. Окрім того, інші пристрої в навколишньому середовищі можуть створювати електромагнітні шуми, які можуть спотворити сигнал датчика.

2.6. Індикатор OLED SSD 1306

SSD1306 є контролером OLED-дисплеїв, розробленим компанією Solomon Systech Limited (рис.2.7.). Цей контролер використовується для управління дисплеями з OLED-панелями розміром 0,96 дюйма. Він підтримує різні функції, включаючи відображення тексту, графіки, анімації та інших елементів на дисплеї.

Контроллер працює з різними мікроконтролерами і має інтерфейси, такі як I_2C (Inter-Integrated Circuit) і SPI (Serial Peripheral Interface), для зручного підключення до мікроконтролерних пристроїв. Контролер також підтримує керування яскравістю дисплея, режимами сну і має можливість вибору кольору фону та шрифту. Є досить популярним контролером для OLED-дисплеїв розміром 0,96 дюйма через свою простоту використання і гнучкість. [8]

OLED (Organic Light Emitting Diode) є технологією дисплеїв, яка використовує органічні речовини, щоб створити світло. Ці органічні речовини випромінюють світло при проходженні електричного струму через них. OLED-дисплеї відрізняються високою яскравістю, контрастністю і широким кутом огляду, а також є енергоефективними, оскільки світло випромінюється тільки з активних пікселів. [8]

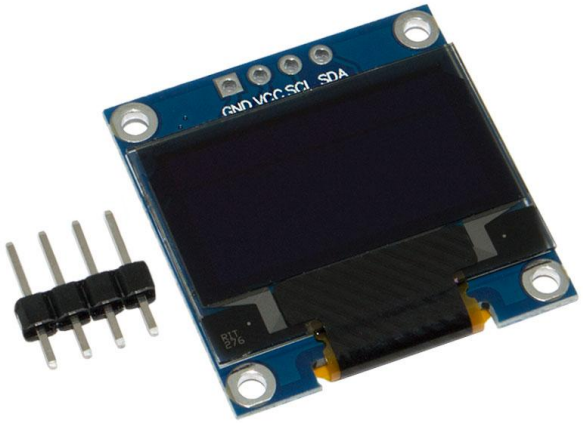


Рис.2.7.

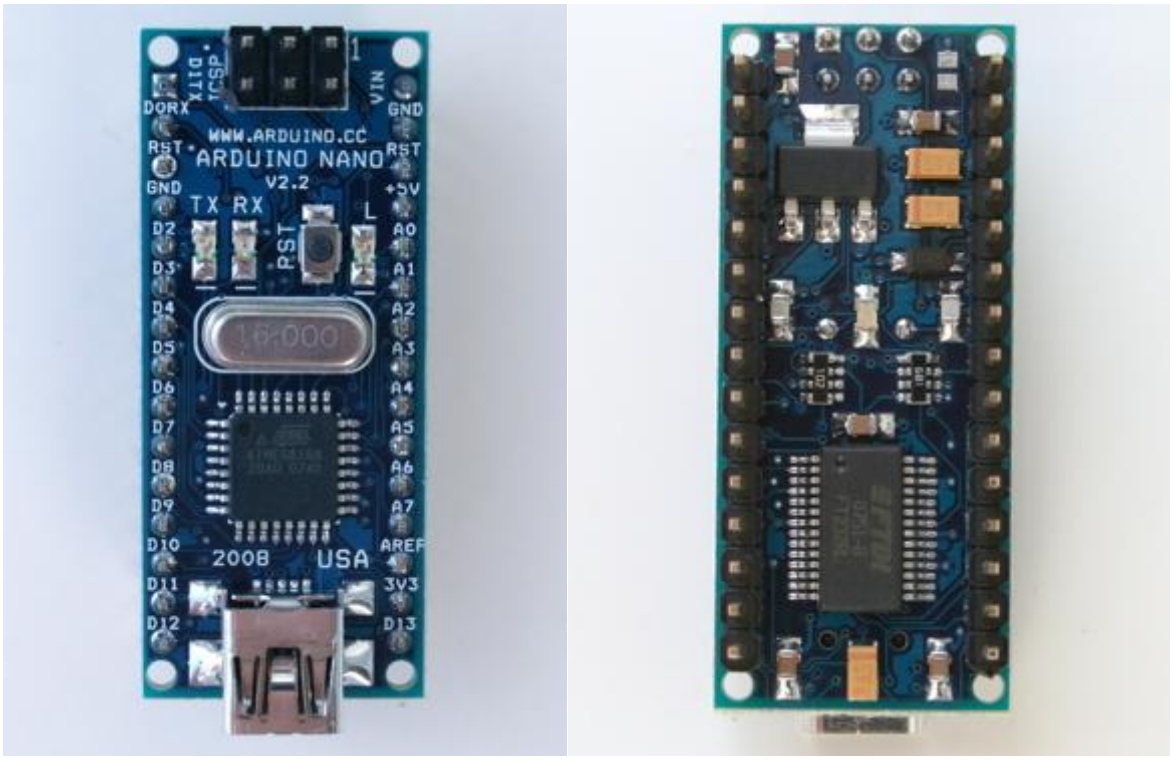
Розглянувши датчик та індикатор, можна перейти до розгляду мікроконтролера що буде виконувати обчислення та виводити інформацію.

РОЗДІЛ 3
МІКРОКОНТРОЛЕР

3.1. Arduino Nano

3.1.1 Основні характеристики

Arduino Nano - це повнофункціональний мініатюрний пристрій на базі мікроконтролера ATmega328 (Arduino Nano 3.0) адаптований для використання з макетними платами (рис.3.1(а), (б)). За функціональністю пристрій схожий на Arduino Duemilanove, і відрізняється від нього розмірами, відсутністю роз'єму живлення, а також іншим типом (Mini-B) USB-кабелю. Arduino Nano розроблено і випускається фірмою Gravitech.[7]



(а) (б)

Рис. 3.1. Вигляд плати згори(а) та знизу(б)

Основні характеристики плати наведено в табл.3.1.

Таблиця 3.1

Мікроконтролер	ATmega328
Робоча напруга	5В
Напруга живлення (рекомендована)	7-12В
Напруга живлення (максимальна)	6-20В
Цифрові входи та виходи	14 (з яких 6 можуть використовуватися як ШИМ-виходи)
Аналогові входи	8
Максимальний струм одного виходу	40 мА
Flash-пам’ять	32 КБ (2 з яких використовує завантажувач)
EEPROM	1 КБ
Тактова частота	16 МГц
Розміри плати	1.85 см x 4.3 см

3.1.2 Живлення

Arduino Nano може живитися через кабель Mini-B USB, від зовнішнього джерела живлення з нестабілізованою напругою 6-20В (через вивід 30) або зі стабілізованою напругою 5В (через вивід 27). Пристрій автоматично вибирає джерело живлення з найбільшою напругою.

Напруга на мікросхему FTDI FT232RL подається тільки в разі живлення Arduino Nano через USB. Тому при живленні пристрою від інших зовнішніх джерел (не USB), вихід 3.3В (сформований мікросхемою FTDI) буде неактивним, унаслідок чого світлодіоди RX і TX можуть мерехтіти за наявності високого рівня сигналу на виводах 0 і 1.[7]

3.1.3 Входи та виходи

З використанням функцій pinMode(), digitalWrite() і digitalRead() кожен із 14 цифрових виводів Arduino Nano може працювати як вхід або вихід. Робоча напруга виводів - 5В. Максимальний струм, який може віддавати або споживати один вивід, становить 40 мА. Усі виводи пов'язані з внутрішніми підтягувальними резисторами (за замовчуванням вимкненими) номіналом 20-50 кОм. Крім основних, деякі виводи Ардуіно можуть виконувати додаткові функції:

Послідовний інтерфейс: виводи D0 (RX) і D1 (TX). Використовуються для отримання (RX) і передавання (TX) даних за послідовним інтерфейсом. Ці виводи з'єднані з відповідними виводами мікросхеми-перетворювача USB-UART від FTDI.

Зовнішні переривання: виводи D2 і D3. Ці виводи можуть бути налаштовані як джерела переривань, що виникають за різних умов: за низького рівня сигналу, за фронтом, за спадом або в разі зміни сигналу. Для отримання додаткової інформації див. функцію attachInterrupt().

ШІМ: виводи D3, D5, D6, D9, D10 і D11. За допомогою функції analogWrite() можуть виводити 8-бітові аналогові значення у вигляді ШІМ-сигналу.

Інтерфейс SPI: виводи D10 (SS), D11 (MOSI), D12 (MISO), D13 (SCK). Ці виводи дають змогу здійснювати зв'язок за інтерфейсом SPI. У пристрої реалізовано апаратну підтримку SPI, однак наразі мова Ардуіно поки що її не підтримує. Світлодіод: вивід 13. Вбудований світлодіод, приєднаний до цифрового виводу 13. Під час надсилання значення HIGH світлодіод вмикається, під час надсилання LOW - вимикається.

В Arduino є 8 аналогових входів (A0-A7), кожен з яких може представити аналогову напругу у вигляді 10-бітного числа (1024 різних значення). За замовчуванням, вимірювання напруги здійснюється щодо діапазону від 0 до 5 В. Проте верхню межу цього діапазону можна змінити, використовуючи вивід AREF і функцію analogReference(). Крім цього, деякі з виводів мають додаткові функції:

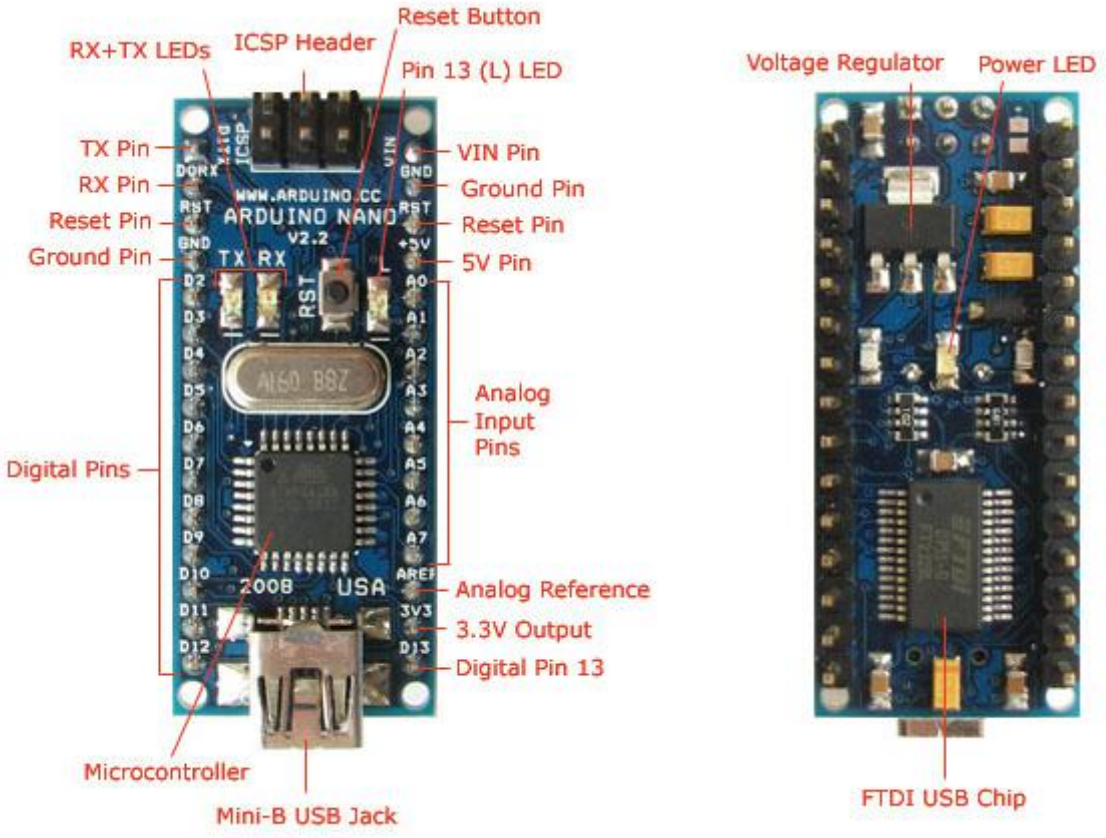
I_2C: виводи 4 (SDA) і 5 (SCL). З використанням бібліотеки Wire (документація на веб-сайті Wiring) ці виводи можуть здійснювати зв'язок по інтерфейсу I_2C (TWI).[7]

Крім перерахованих на платі існує ще кілька виводів:

AREF. Опорна напруга для аналогових входів. Може задіюватися функцією analogReference().

Reset. Формування низького рівня (LOW) на цьому виводі призведе до перезавантаження мікроконтролера. Зазвичай цей вивід слугує для функціонування кнопки скидання на платах розширення

Конфігурація виводів представлена на рис. 3.2(а),(б)



(а) (б)

Рис. 3.2

3.1.4 Зв’язок

Arduino Nano надає низку можливостей для здійснення зв'язку з комп'ютером, ще одним Ардуіно або іншими мікроконтролерами. У ATmega328 є приймач UART, що дає змогу здійснювати зв'язок через послідовні інтерфейси за допомогою цифрових виводів 0 (RX) і 1 (TX). Мікросхема FTDI FT232RL забезпечує зв'язок приймача з USB-портом комп'ютера, і під час під'єднання до ПК дає змогу Ардуіно визначатися як віртуальний COM-порт (драйвери FTDI включено в пакет програмного забезпечення Ардуіно). До пакета програмного забезпечення Ардуіно також входить спеціальна програма, що дає змогу зчитувати і надсилати на Ардуіно прості текстові дані. Під час передавання даних комп'ютеру через USB на платі блиматимуть світлодіоди RX і TX. (Під час послідовного передавання даних за допомогою виводів 0 і 1 ці світлодіоди не задіюються).[7]

Бібліотека SoftwareSerial дає змогу реалізувати послідовний зв'язок на будь-яких цифрових виводах Arduino Nano.

У мікроконтролері ATmega328 також реалізовано підтримку послідовних інтерфейсів I_2C (TWI) і SPI. У програмне забезпечення Ардуіно входить бібліотека Wire, що дає змогу спростити роботу з шиною I_2C; для отримання детальнішої інформації зверніться до документації. Для роботи з інтерфейсом SPI використовуються даташити мікроконтролеру ATmega328.

3.1.5 Програмування

Arduino Nano програмується за допомогою програмного забезпечення Ардуіно – Arduino IDE. Arduino IDE (Integrated Development Environment) - це середовище розробки для програмування мікроконтролерів Arduino. Воно надає зручний інтерфейс, що допомагає створювати і налагоджувати програми для Arduino-плат.

Програма забезпечує візуальне програмування на основі мови C / C++. Вона має вбудований текстовий редактор, де можна написати код програми. Після написання коду його можна компілювати та завантажувати на мікроконтролер Arduino.

Arduino IDE також надає доступ до бібліотек, які розширюють можливості програмування Arduino. Ці бібліотеки містять готові функції та процедури для використання різних компонентів, таких як датчики, дисплеї, мотори і т. д.[9]

Для цього з меню Tools > Board необхідно вибрати "Arduino Duemilanove or Nano w/ ATmega328" (залежно від мікроконтролера на вашій платі). Для отримання детальнішої інформації див. довідку та приклади.

ATmega328 в Arduino Nano випускається з прошитим завантажувачем, що дає змогу завантажувати в мікроконтролер нові програми без необхідності використання зовнішнього програматора. Взаємодія з ним здійснюється за оригінальним протоколом STK500 (довідка, заголовки C-файлів).

Проте мікроконтролер можна прошити і через роз'єм для внутрішньосхемного програмування ICSP (In-Circuit Serial Programming), не звертаючи уваги на завантажувач.

3.1.6 Автоматичне (програмне) скидання

Щоб щоразу перед завантаженням програми не потрібно було натискати кнопку скидання, Arduino Nano спроектовано таким чином, що дає змогу здійснювати його скидання програмно з під'єднаного комп'ютера. Один із виводів мікросхеми FT232RL, що бере участь в управлінні потоком даних (DTR), з'єднаний із виводом RESET мікроконтролера ATmega168 або ATmega328 через конденсатор номіналом 100 нФ. Коли на лінії DTR з'являється нуль, вивід RESET також переходить у низький рівень на час, достатній для перезавантаження мікроконтролера. Цю особливість

використовують для того, щоб можна було прошивати мікроконтролер лише одним натисканням кнопки в середовищі програмування Ардуїно. Така архітектура дає змогу зменшити таймаут завантажувача, оскільки процес прошивки завжди синхронізований зі спадом сигналу на лінії DTR. Така архітектура дає змогу зменшити таймаут завантажувача, оскільки процес прошивки завжди синхронізований зі спадом сигналу на лінії DTR.

Однак ця система може призводити й до інших наслідків. Під час під'єднання Arduino Nano до комп'ютерів, що працюють на Mac OS X або Linux, його мікроконтролер буде скидатися при кожному з'єднанні програмного забезпечення з платою. Після скидання на Arduino Nano активізується завантажувач на час близько пів секунди. Незважаючи на те, що завантажувач запрограмований ігнорувати сторонні дані (тобто всі дані, що не стосуються процесу прошивання нової програми), він може перехопити кілька перших байт даних з посилки, яку надсилають платі відразу після встановлення з'єднання. Відповідно, якщо в програмі, що працює на Ардуіно, передбачено отримання від комп'ютера будь-яких налаштувань або інших даних під час першого запуску, переконайтеся, що програмне забезпечення, з яким взаємодіє Ардуіно, здійснює відправлення через секунду після встановлення з'єднання.[7]

3.2. Мікроконтролер ATmega328

Основні характеристики мікроконтролера перелічені в таблиці 3.2

Таблиця 3.2

Архітектура	8 біт
Частота роботи	16 МГц
Кількість таймерів/лічильників	3
АЦП	10-біт, 6ти канальний
Пам’ять (Flash)	32 КБ

SRAM	2КБ
EEPROM	1 КБ
Інтерфейси	UART, SPI, I_2C
Кількість пінів	32
Кількість входів/виходів (I/O) (загального призначення)	23
Кількість аналогових входів	6

Схема роботи мікроконтролера ATmega328 може бути узагальнено описана наступним чином:

1. Завантаження програми: Починається з завантаження програмного коду в програмну пам'ять (Flash) мікроконтролера. Це може виконуватися через спеціальні програматори або за допомогою інтегрованих рішень, таких як Arduino IDE. Програма зберігається в пам'яті Flash і буде виконуватися після включення живлення.

2. Ініціалізація: При включенні живлення мікроконтролер виконує початкову ініціалізацію. Це може включати налаштування таких параметрів, як режими таймерів, встановлення початкових значень регістрів, налаштування переривань та інше.

3. Виконання програми: Після ініціалізації мікроконтролер починає виконувати програму збережену в пам'яті Flash. Він починає виконувати інструкції послідовно, одну за одною. Це включає обчислення, розгалуження, взаємодію з периферійними пристроями та інші дії, необхідні для функціонування програми.

4. Обробка переривань: Мікроконтролер може приймати переривання від зовнішніх пристроїв або внутрішніх подій. При спрацюванні переривання виконання основної програми призупиняється, а виконується обробка переривання. Це дозволяє реагувати на події в реальному часі та виконувати

певні дії, наприклад, зчитування даних з датчиків або взаємодію зі зовнішніми пристроями.

5. Взаємодія з периферійними пристроями: ATmega328 має різні периферійні пристрої, такі як таймери, зовнішні інтерфейси (UART, SPI, I_2C), аналого-цифровий перетворювач (ADC) та багато інших. Мікроконтролер може взаємодіяти з цими пристроями, налаштовувати їх параметри, передавати та отримувати дані.

6. Виведення результатів: Мікроконтролер може виводити результати своєї роботи на зовнішні пристрої, такі як дисплеї, світлодіодні індикатори або інші виводи. Він може керувати станом виводів і передавати необхідні дані для відображення або керування зовнішніми пристроями.

7. Цикл виконання: Мікроконтролер продовжує виконувати програму в циклі, повторюючи кроки від 3 до 6, поки живлення залишається увімкненим. Він неперервно виконує програму та реагує на зміни у вхідних сигналах, події та переривання для забезпечення необхідного функціонування системи.

Це загальна схема роботи пристрою. Залежно від конкретного застосування та програми, можуть бути додаткові кроки та функції, які додаються програмістом для вирішення конкретних завдань.[10]

Внутрішня структура мікроконтролера складається із багатьох компонентів. Структурну схему мікроконтролера приведена на рис.3.3..

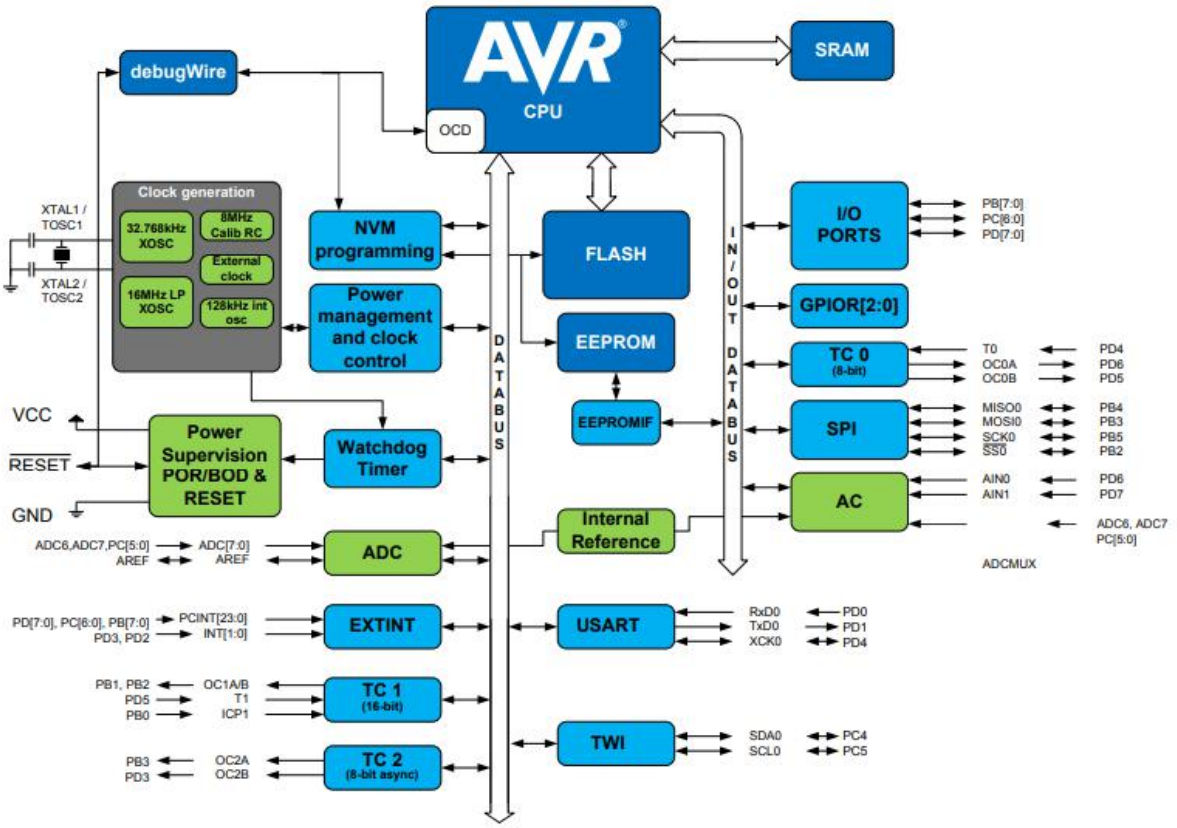


Рис. 3.3. Структура мікроконтролера

Мікроконтролер ATmega328 має велику кількість регістрів, з якими він взаємодіє для керування та контролю різних функцій. Регістри ATmega328 можна розділити на декілька категорій:

- Регістри загального призначення (General Purpose Registers): ATmega328 має 32 регістри загального призначення, позначені як R0-R31. Ці регістри використовуються для зберігання даних та проміжних результатів під час виконання програми. Вони широко використовуються для обчислень та зберігання тимчасових даних.
- Регістри вказівників (Instruction Register): пристрій має два регістри вказівників, позначені як X та Y. Вони використовуються для доступу до даних в пам'яті. Регістр X зазвичай використовується для доступу до даних у програмній пам'яті (Flash), а регістр Y - для доступу до даних у SRAM.

- Регістр вказівника стеку (Stack Pointer): Stack Pointer (SP) вказує на поточну позицію стеку в SRAM. Він використовується для збереження тимчасових даних та адрес повернення під час виклику та повернення з підпрограм.
- Регістр статусу (Status Register): Регістр статусу (SREG) використовується для зберігання різних прапорців та прапорців переривань, які впливають на виконання програми. Наприклад, прапорець переносу (Carry) вказує на перенос або запозичення під час виконання арифметичних операцій.
- Регістри керування таймерами та зовнішніми перериваннями (Program Counter): ATmega328 має регістри, що використовуються для налаштування та керування таймерами та зовнішніми перериваннями. Наприклад, регістр TCCR1A використовується для налаштування режиму таймера 1, а регістр PCICR дозволяє увімкнути зовнішні переривання для певних виводів мікроконтролера.

Ці регістри можна зчитувати та записувати безпосередньо у програмі на мові програмування, такій як C або C++, за допомогою спеціальних інструкцій для роботи з регістрами. Взаємодія з регістрами дозволяє програмістам контролювати різні функції мікроконтролера та налаштовувати його роботу відповідно до потреб пристрою що проектується.[10]

Схема структури пам’яті процесора наведена на рис. 3.4., вона відображає внутрішню взаємодію між елементами пам’яті мікроконтролера.

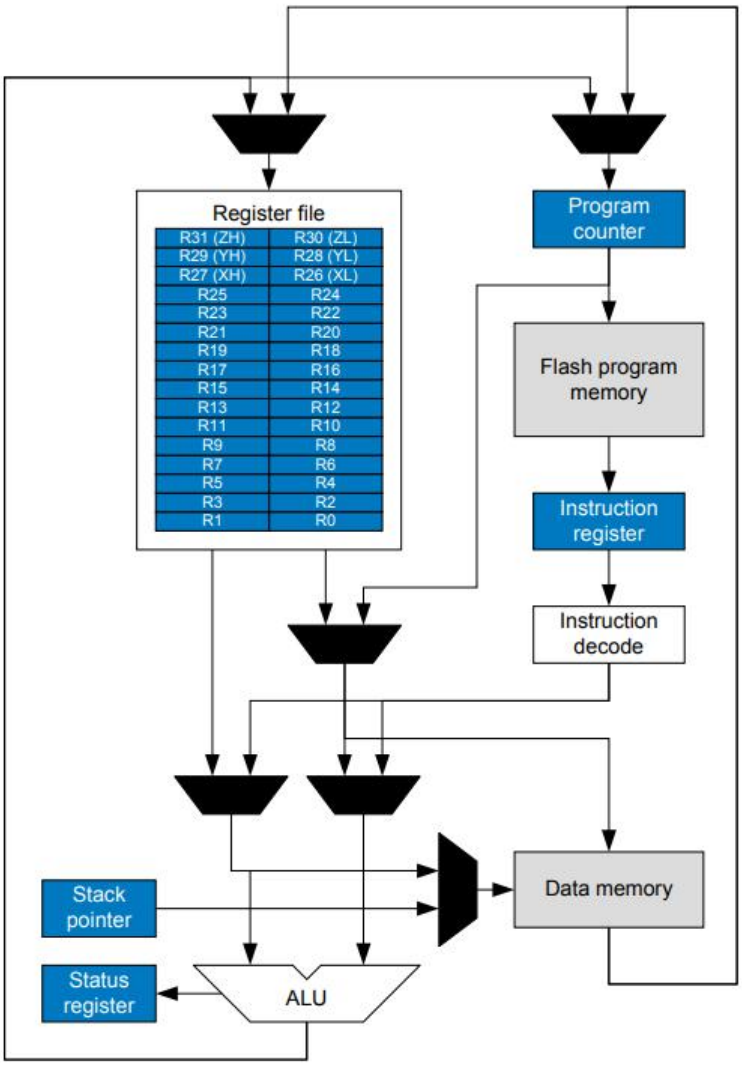


Рис. 3.4. Структура пам’яті мікроконтролера

РОЗДІЛ 4 РОЗРОБКА ПРОГРАМНОГО КОДУ

4.1. Середовище розробки Arduino IDE

Arduino IDE (Integrated Development Environment) - це середовище розробки, призначене для програмування і взаємодії з платформою Arduino. Воно надає зручний інтерфейс для написання коду, завантаження його на плату Arduino і відладку програм.

Основні особливості Arduino IDE включають:

- Редактор коду: має простий текстовий редактор, де можна написати програмний код мовою Arduino. Редактор надає функції підсвічування синтаксису, автоматичного завершення коду і відступів для полегшення роботи з кодом.
- Бібліотеки поставляється з великою кількістю стандартних бібліотек, які містять функції і класи для роботи з різними пристроями і модулями. Ці бібліотеки значно спрощують розробку проектів, оскільки містять готові функції для роботи з датчиками, дисплеями, моторами та іншими пристроями.
- Компіляція і завантаження: в наявності вбудований компілятор, який перетворює код мовою Arduino в машинний код, зрозумілий платі Arduino. Після компіляції можна завантажити програму на плату Arduino через USB-порт або інші доступні засоби підключення.
- Відладка: надає прості засоби для відладки програм. Є можливість виводу повідомлень на консоль, а також можна використовувати вбудований засіб монітора порту для перегляду і відладки даних, що передаються між Arduino і комп'ютером.
- Підтримка різних плат: підтримує не тільки оригінальні плати Arduino, але й багато інших сумісних плат, таких як ESP8266, ESP32, STM32 і

багато інших. Для кожної платформи доступні відповідні бібліотеки і налаштування.

- Відкрите програмне забезпечення: Arduino IDE є відкритим програмним забезпеченням, що означає, що його вихідний код доступний для перегляду та модифікації. Це дозволяє розробникам вносити зміни в середовище розробки або створювати власні розширення і бібліотеки.

Загалом, Arduino IDE - це потужний інструмент, що допомагає розробникам швидко та зручно програмувати проекти на базі платформи Arduino і взаємодіяти з підключеними пристроями та датчиками. Його простота використання і підтримка широкої спільноти роблять його популярним серед початківців і досвідчених розробників. [9]

Для написання коду, слід зазначити алгоритм роботи пристрою(рис.4.1.).

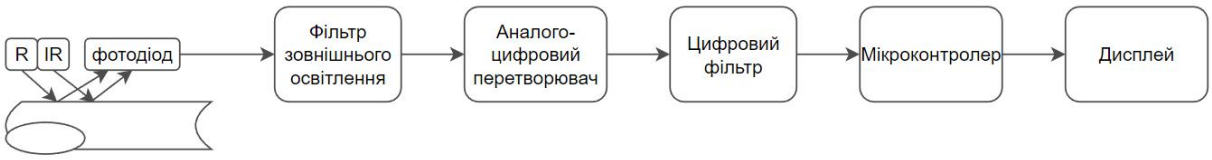


Рис.4.1. Алгоритм роботи пульсоксиметра

4.2. Бібліотеки

Для початку, розглянемо бібліотеки що будуть використані для написання програмного забезпечення пульсоксиметра (рис. 4.2.).

```
1  #include "ssd1306h.h"
2  #include "MAX30102.h"
3  #include "Pulse.h"
4  #include <EEPROM.h>
5  #include <avr/sleep.h>
6
```

Рис. 4.2.

1. "ssd1306h.h" – Використовується для роботи з OLED дисплеєм, за допомогою функцій що знаходяться в цій бібліотеці можна виводити текст, числа, символи та малювати просту графіку.
2. "MAX30102.h" – бібліотека для роботи із датчиком пульсу та насиченості крові киснем. За допомогою функцій з бібліотеки, можемо звертатись до датчика отримувати з нього інформацію для подальшої обробки. [11]
3. "Pulse.h" – використовується для фільтрації даних отриманих з датчика та реєстрації серцебиття.
4. <EEPROM.h> (англ. Electrically Erasable Programmable Read-Only Memory) – функції з бібліотеки використовуються для доступу до одного з типів пам’яті Arduino. Ця пам’ять є енергонезалежною і після перезавантаження пристрою та від’єднання його від живлення зберігає значення в собі.[11]
5. <avr/sleep.h> - для функцій що дозволяють використовувати режим очікування Arduino, що знижує енергоспоживання пристрою коли їм не користуються. [11]

4.3 Основні функції

В цьому розділі буть розглянуті та описані основні функції що відповідають за коректну роботу пристрою.

```
void draw_oled(int msg) {
    oled.firstPage();
    do{
    switch(msg){
        case 0:  oled.drawStr(10,0,F("Device error"),1);
                 break;
        case 1:  oled.drawStr(0,0,F("PLACE YOUR"),2);
                 oled.drawStr(25,18,F("FINGER"),2);
                 break;
        case 2:  print_digit(86,0,beatAvg);
                 oled.drawStr(0,3,F("PULSE RATE"),1);
                 oled.drawStr(11,17,F("OXYGEN"),1);
                 oled.drawStr(0,25,F("SATURATION"),1);
                 print_digit(73,16,SPO2f,' ',3,2);
                 oled.drawChar(116,16,'%',2);
                 break;
        case 3:  oled.drawStr(33,0,F("Pulse"),2);
                 oled.drawStr(17,15,F("Oximeter"),2);
                 break;
        case 4:  oled.drawStr(28,12,F("OFF IN"),1);
                 oled.drawChar(76,12,10-sleep_counter/10+'0');
                 oled.drawChar(82,12,'s');
                 break;
        }
    } while (oled.nextPage());
}
```

Рис.4.3.

За допомогою функції зображеної на рис. 4.3. ми заготовлюємо 5 сценаріїв виводу інформації на екран.

1. (case 0) (рис.4.4.) При некоректному підключенні або пошкодженню датчика, помилка про неможливість доступу до інформації з MAX30102 буде виведена на екран для користувача.

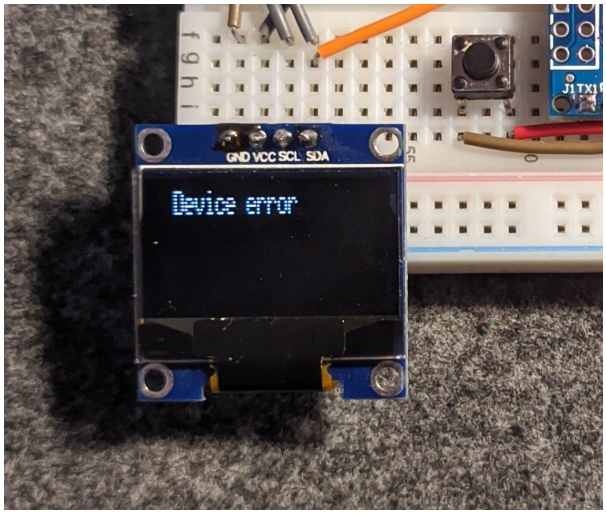


Рис.4.4.

2. (case 3) (рис.4.5.) Початковий екран. Виводиться при підключенні пристрою до джерела живлення або після виходу з режиму очікування. Під час виведення цього екрану користувачеві, пристрій ініціалізує дані.

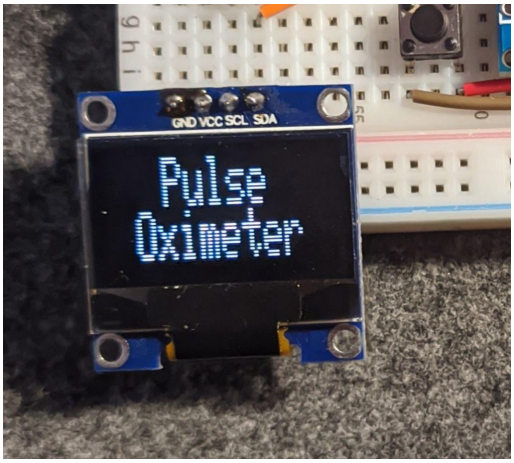


Рис.4.5.

3. (case 1) (рис.4.6.) Інформація про готовність роботи пристрою із закликом до притискання пальця до датчика із ціллю виміру необхідних даних.

Рис.4.6.

4. (case 2) (рис.4.7.) Стандартний екран на який виводиться актуальна інформація про частоту серцебиття та насиченості крові киснем в відсотках.

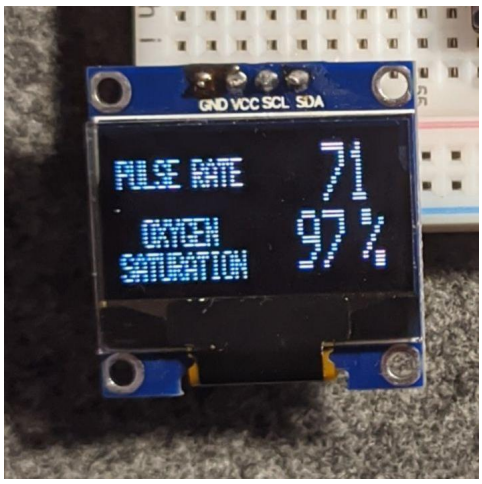


Рис.4.7.

5. (case 4) (рис.4.8.) Після прибирання пальця з датчику пристрій переходить у режим очікування після 5 секунд. Під час зворотнього відліку, інформація про перехід до режиму очікування та час що залишився виводиться на екран.

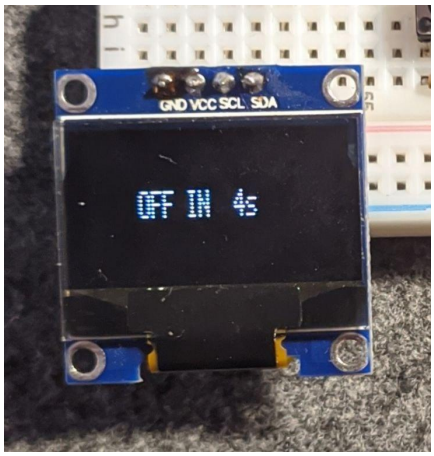


Рис.4.8.

Функція go_sleep(рис.4.9) відповідає за перехід пристрою в режим очікування, вона вимикає екран та сенсор, знижує енергоспоживання пульсоксиметра. Також функція чекає сигналу з кнопки, щоб «розбудити» пристрій запустивши ПО.

```
void go_sleep() {
    oled.fill(0);
    oled.off();
    delay(10);
    sensor.off();
    delay(10);
    cbi(ADCSRA, ADEN);  // disable adc
    delay(10);
    pinMode(0,INPUT);
    pinMode(2,INPUT);
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);
    sleep_mode();  // sleep until button press
    // cause reset
    setup();
}
```

Рис.4.9.

Код на рис.4.10. представляє собою частину програми, яка вимірює та визначає частоту серцебиття на основі різниці часу між ударами серця.

1. Перевіряється умова `draw_Red ? beatRed : beatIR`. Залежно від цієї умови виконується вимірювання серцебиття для червоного (`beatRed`) або інфрачервоного (`beatIR`) сигналу.

2. Обчислюється час, що пройшов між попереднім ударом серця та поточним моментом. Це робиться за допомогою різниці між поточним часом, отриманим за допомогою `millis()`, та попереднім значенням часу удару серця `lastBeat`.

3. На основі розрахованого часу між ударами серця, обчислюється частота ударів серця на хвилину (`beatsPerMinute`). Для цього використовується формула `60 / (btpm / 1000.0)`, де `btpm` - це розрахована різниця часу між ударами серця.

4. Якщо значення `beatsPerMinute` знаходиться в діапазоні від 20 до 255, то воно зберігається в масиві `rates` за допомогою змінної `rateSpot`. Значення `rateSpot` інкрементується, а потім оновлюється залишком від ділення на `RATE_SIZE`. Це забезпечує циклічне зберігання значень у масиві.

5. Після збереження нового значення в масиві `rates`, обчислюється середнє значення частоти ударів серця (`beatAvg`). Для цього сумуються всі значення `rates` у циклі, а потім результат ділиться на `RATE_SIZE`, щоб отримати середнє значення.

Загалом, цей код дозволяє виміряти частоту серцебиття, зберегти кілька останніх значень в масиві та обчислити середнє значення цих вимірів.

```
// check IR or Red for heartbeat
if (draw_Red ? beatRed : beatIR){
    long btpm = millis() - lastBeat;
    beatsPerMinute = 60 / (btpm / 1000.0);
    lastBeat = millis();
  if (beatsPerMinute < 255 && beatsPerMinute > 20)
  {
    rates[rateSpot++] = (byte)beatsPerMinute; //Store this reading in the array
    rateSpot %= RATE_SIZE; //Wrap variable

    //Take average of readings
    beatAvg = 0;
    for (byte x = 0 ; x < RATE_SIZE ; x++)
      beatAvg += rates[x];
    beatAvg /= RATE_SIZE;
  }
```

Рис.4.10.

Наступний код (рис.4.11.) використовується для обчислення значення насиченості крові киснем (SpO2) на основі даних, отриманих з датчика.

Перші два рядки коду обчислюють чисельник і знаменник для обрахунку відношення SpO2 (RX100). Вони використовують середні значення, отримані з двох датчиків пульсового червоного світла (pulseRed) і пульсового інфрачервоного світла (pulseIR).

У третьому рядку відбувається обчислення відношення RX100, яке використовується в наступному рядку для обчислення SPO2f (насиченості крові киснем за формулою). Значення SPO2f обчислюється за допомогою простої формули, в якій RX100 множиться на певні коефіцієнти та виконуються математичні операції.

Останні два рядки коду використовуються для отримання значення насиченості крові киснем (SPO2) з таблиці. Якщо значення RX100 потрапляє

в певний діапазон (від 0 до 183), то воно використовується як індекс для отримання відповідного значення насиченості крові киснем з таблиці (spo2_table). Значення SPO2 зберігається у змінній SPO2.

Загалом, цей код використовує отримані дані з датчика пульсу червоного і інфрачервоного світла, обчислює відношення RX100 та на основі цього обчислює значення насиченості крові киснем (SPO2) за допомогою формули або таблиці.

```
// compute SpO2 ratio
long numerator   = (pulseRed.avgAC() * pulseIR.avgDC())/256;
long denominator = (pulseRed.avgDC() * pulseIR.avgAC())/256;
int RX100 = (denominator>0) ? (numerator * 100)/denominator : 999;
// using formula
SPO2f = (10400 - RX100*17+50)/100;
// from table
if ((RX100>=0) && (RX100<184))
  SPO2 = pgm_read_byte_near(&spo2_table[RX100]);
```

Рис.4.11.

Загальна блок схема роботи коду(рис.4.12.) виглядає наступним чином:

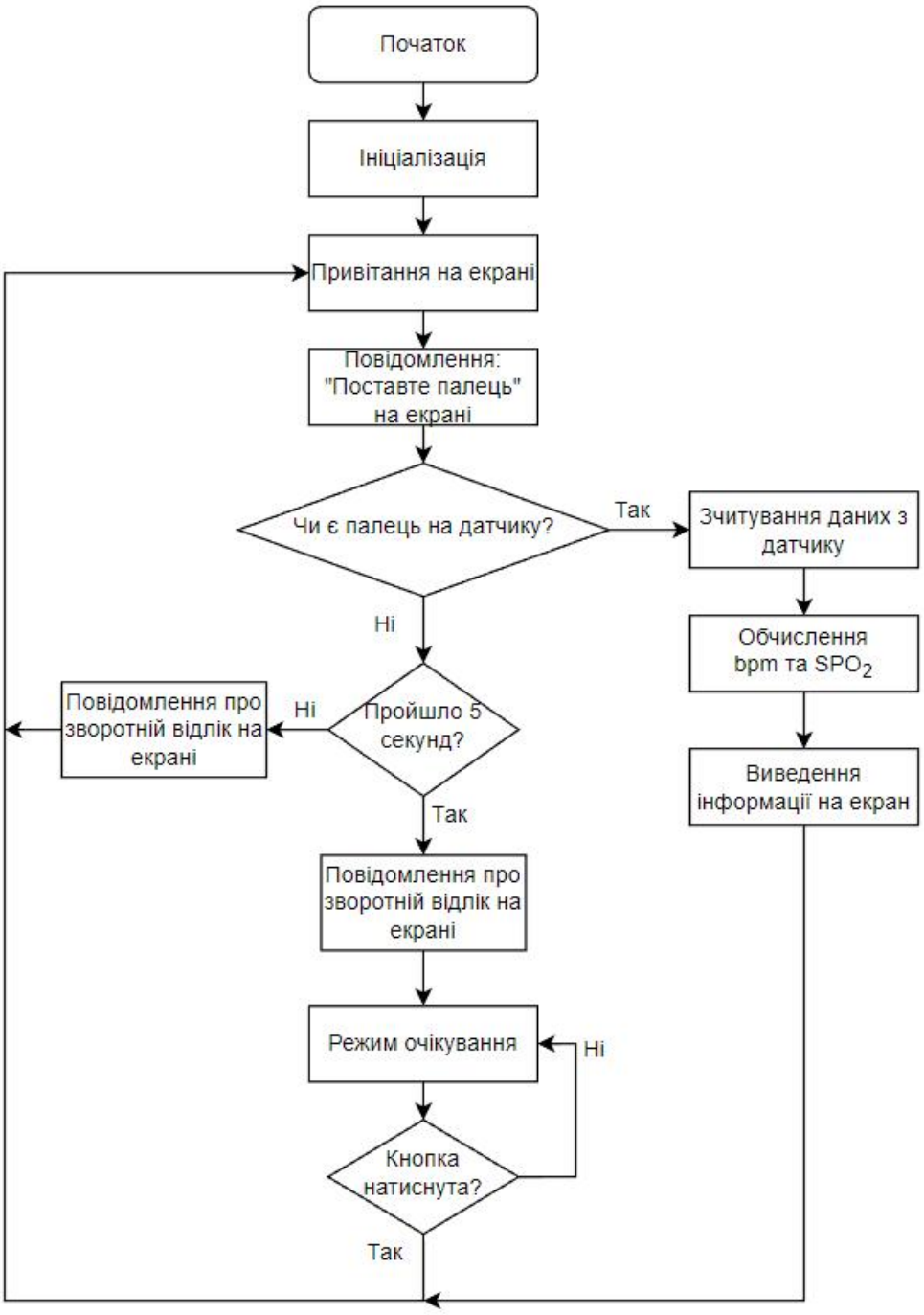


Рис. 4.12.

Повний код програмного забезпечення мікроконтролера можна знайти у додатку А.

ВИСНОВКИ

У даній роботі була показана актуальність пульсоксиметра в сучасному світі. Були розглянуті випадки використання пристрою у різних медичних галузях, таких як кардіологія, анестезіологія, екстрена медицина, педіатрія, неврологія та спорт. Використання пульсоксиметра в сучасному світі веде до багатьох переваг та можливостей. Окрім медичних галузей, в яких пристрій застосовується для діагностики та моніторингу пацієнтів, його використання також поширене у спорті. Спортсмени використовують пульсоксиметри для контролю за своїм фізичним станом під час тренувань та змагань. Вимірювання пульсу та насиченості крові киснем дозволяють спостерігати за змінами в організмі та вчасно реагувати на перевантаження або погіршення фізичного стану.

Наявність портативних пульсоксиметрів, зокрема тих, які засновані на платформі Arduino Nano, робить їх доступними та зручними для широкого кола користувачів. Вони можуть бути використані як професіоналами в медичних закладах, так і невід'ємною частиною домашнього медичного обладнання. Це дозволяє пацієнтам з хронічними захворюваннями або післяопераційним пацієнтам здійснювати самостійний контроль за своїм станом та вчасно реагувати на будь-які зміни.

Збір та аналіз даних, здійснюваних пульсоксиметром, може бути посилено за допомогою інтеграції пристрою у систему моніторингу життєдіяльності. Це дозволить створити повноцінну систему збору, зберігання та аналізу даних, що отримуються протягом тривалого періоду часу. Такі дані можуть бути використані для вивчення трендів у здоров'ї людини, виявлення патологій або ефективності лікування.

У роботі також були розглянуті різні види пульсоксиметрів, включаючи портативні пристрої, які можуть бути використані як медичним персоналом у медичних закладах, так і вдома пацієнтами для самостійного контролю. Були проаналізовані принципи роботи сучасних датчиків

пульсоксиметрів, які базуються на інфрачервоному світлі та фотодіоді, що дозволяє точно виміряти пульс та рівень кисню в крові.

Був зібраний пульсоксиметр на основі Arduino Nano, що є потужною платформою з відкритим кодом для розробки електронних пристроїв. Використовуючи датчик пульсу та рівня кисню в крові, а також дисплей та тактильну кнопку для взаємодії з пристроєм, було успішно спроектовано та зібрано пристрій, який може надавати точні вимірювання обраних параметрів.

Зібраний пульсоксиметр на основі Arduino Nano виявився не тільки ефективним вимірювальним пристроєм, але й гнучкою платформою для подальших розширень та розробок. Його можна легко налаштувати для збору додаткових даних та виконання розрахунків, що дозволяє розширити можливості пристрою в галузях, які вимагають більш детального аналізу.

Цей пристрій виявився особливо корисним у польових умовах, оскільки його можна легко налаштувати для збору додаткових даних та виконання розрахунків. Він може бути доповнений різноманітними сенсорами, такими як термометр, акселерометр та гіроскоп, що розширює його функціональні можливості в галузях спорту, медицини та реабілітації. Завдяки своїй енергоефективності та простоті, пульсоксиметр на Arduino Nano стає незамінним інструментом у польових умовах, де точність, мобільність та надійність є важливими факторами.

Для роботи пристрою був розроблений відповідний код, що дозволяє отримувати дані з датчика та виводити їх на екран. Цей код не лише може бути використаний для подальшої обробки даних або взаємодії з іншими пристроями, але також може бути інтегрований у систему моніторингу життєдіяльності. Така система може бути використана для дослідження стану здоров'я людини, збору та аналізу даних на тривалому проміжку часу.

Код, розроблений для роботи пульсоксиметра, може бути налаштований для передачі даних на зовнішні сервери або зберігання на локальному пристрої. Це дає можливість створювати інтелектуальні системи з аналізом даних в реальному часі, машинним навчанням та алгоритмами

штучного інтелекту. Така інтеграція може бути особливо корисною в галузі медичного моніторингу, де можливість автоматичного виявлення аномалій або попередження про критичні стани може врятувати життя пацієнта.

Отже, в результаті даної роботи було показано, що пульсоксиметр є актуальним пристроєм для моніторингу пульсу та рівня кисню в крові. Зібраний пульсоксиметр на основі Arduino Nano є прикладом вдалого поєднання технологій, що дозволяють надавати точні вимірювання та розширюють можливості медичного моніторингу. Його потенціал для використання у багатьох галузях, від медицини до спорту, робить його цінним інструментом для стеження за здоров'ям та поліпшення якості життя.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Руководство ВОЗ по пульсоксиметрии. [Електронний ресурс] – Електрон. дан. – Режим доступу: https://csma.org.ua/wp-content/uploads/2018/06/%D0%9F%D0%A3%D0%9B%D0%AC%D0%A1%D0%9E%D0%9A%D0%A1%D0%98%D0%9C%D0%95%D0%A2%D0%A0%D0%86%D0%AF.pdf
2. Методика визначення кисню в артеріальній крові людини, яка займається фізичними вправами: метод.рекомеменд. / [Чичкан О.А., Довганик М.С., Кмицяк М.Г., Костовський М.Г.]; "Львівський держ. ун-т внутр. справ", ін-т з підг. фах. для підр. Нац. поліції, 2023. – 18 с.
3. Мікропроцесорна техніка [Електронний ресурс] : підручник для студентів спеціальності «Електроніка» / В.Я.Жуйков, Т.О.Терещенко, Ю.С.Ямненко, А.В.Заграничний; НТУУ «КПІ»; ред. О.В.Борисов. – Електронні текстові дані (1 файл: 6,28 Мбайт). – Київ: НТУУ «КПІ», 2016. – 440 с.https://ela.kpi.ua/handle/123456789/18969.
4. Pulse Oximeters: The Invention That Changed the Paradigm of Patient Safety Around the World – A Japanese Perspective [Електронний ресурс] – Електрон. дан. – Режим доступу: https://www.apsf.org/article/pulse-oximeters-the-invention-that-changed-the-paradigm-of-patient-safety-around-the-world-a-japanese-perspective/
5. Ємчик Л.Ф. Основи біологічної фізики і медична апаратура. – Київ, Медицина, 2014. – 392 с./
6. MAX30102 --High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health datasheet. [Електронний ресурс] – Електрон. дан. – Режим доступу: https://www.analog.com/media/en/technical-documentation/data-sheets/max30102.pdf

7. Глухов О.В., Кравчук О.О., Левченко Є.В. Вивчення властивостей мікроконтролерів і електронних систем на базі платформи Ардуіно: навч. посібник для студентів ВНЗ. Харків: ХНУРЕ, 2019. – 192 с.
8. SSD1306 Advance Information [Електронний ресурс] – Електрон. дан. – Режим доступу: https://www.mini-tech.com.ua/download/datasheet/Display/SSD1306.pdf
9. Using the Arduino Software (IDE) [Електронний ресурс] – Електрон. дан. – Режим доступу: https://docs.arduino.cc/learn/starting-guide/the-arduino-software-ide
10. ATMEGA328P Datasheet. [Електронний ресурс] – Електрон. дан. – Режим доступу: https://pdf1.alldatasheet.com/datasheet-pdf/view/1425005/MICROCHIP/ATMEGA328P.html
11. Getting Started with Arduino [Електронний ресурс] – Електрон. дан. – Режим доступу: https://docs.arduino.cc/learn/starting-guide/getting-started-arduino#libraries

Додаток А

```
#include "ssd1306h.h"

#include "MAX30102.h"

#include "Pulse.h"

#include <EEPROM.h>

#include <avr/sleep.h>


#ifndef cbi

#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))

#endif

#ifndef sbi

#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))

#endif



SSD1306 oled;

MAX30102 sensor;

Pulse pulseIR;

Pulse pulseRed;

MAFilter bpm;
```

```
#define LED LED_BUILTIN

#define BUTTON 3

#define OPTIONS 7


static const uint8_t heart_bits[] PROGMEM = { 0x00, 0x00, 0x38, 0x38, 0x7c,
0x7c, 0xfe, 0xfe, 0xfe, 0xff,

                              0xfe, 0xff, 0xfc, 0x7f, 0xf8, 0x3f, 0xf0, 0x1f, 0xe0, 0x0f,

                              0xc0, 0x07, 0x80, 0x03, 0x00, 0x01, 0x00, 0x00, 0x00,
0x00,

                              0x00, 0x00 };


//spo2_table is approximated as  -45.060*ratioAverage* ratioAverage + 30.354
*ratioAverage + 94.845 ;

const uint8_t spo2_table[184] PROGMEM =

      { 95, 95, 95, 96, 96, 96, 97, 97, 97, 97, 97, 98, 98, 98, 98, 98, 99, 99, 99, 99,

       99, 99, 99, 99, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100,

       100, 100, 100, 100, 99, 99, 99, 99, 99, 99, 99, 99, 98, 98, 98, 98, 98, 98, 97,
97,

       97, 97, 96, 96, 96, 96, 95, 95, 95, 94, 94, 94, 93, 93, 93, 92, 92, 92, 91, 91,

       90, 90, 89, 89, 89, 88, 88, 87, 87, 86, 86, 85, 85, 84, 84, 83, 82, 82, 81, 81,

       80, 80, 79, 78, 78, 77, 76, 76, 75, 74, 74, 73, 72, 72, 71, 70, 69, 69, 68, 67,

       66, 66, 65, 64, 63, 62, 62, 61, 60, 59, 58, 57, 56, 56, 55, 54, 53, 52, 51, 50,
```

```
        49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 31, 30, 29,
        28, 27, 26, 25, 23, 22, 21, 20, 19, 17, 16, 15, 14, 12, 11, 10, 9, 7, 6, 5,
        3, 2, 1 } ;


int getVCC() {
  //reads internal 1V1 reference against VCC
  #if defined(__AVR_ATmega1284P__)
    ADMUX = _BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) |
_BV(MUX1);  // For ATmega1284
  #else
    ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);  // For
ATmega328
  #endif
  delay(2); // Wait for Vref to settle
  ADCSRA |= _BV(ADSC); // Convert
  while (bit_is_set(ADCSRA, ADSC));
  uint8_t low = ADCL;
  unsigned int val = (ADCH << 8) | low;
  //discard previous result
  ADCSRA |= _BV(ADSC); // Convert
  while (bit_is_set(ADCSRA, ADSC));
```

```
  low = ADCL;
  val = (ADCH << 8) | low;

  return (((long)1024 * 1100) / val)/100;
}

void print_digit(int x, int y, long val, char c=' ', uint8_t field = 3,const int BIG = 2)
    {
    uint8_t ff = field;
    do {
        char ch = (val!=0) ? val%10+'0': c;
        oled.drawChar( x+BIG*(ff-1)*6, y, ch, BIG);
        val = val/10;
        --ff;
    } while (ff>0);
}

/*
 *   Record, scale  and display PPG Wavefoem
 */
```

```
const uint8_t MAXWAVE = 72;


class Waveform {
  public:
    Waveform(void) {wavep = 0;}


      void record(int waveval) {
        waveval = waveval/8;         // scale to fit in byte
        waveval += 128;              //shift so entired waveform is +ve
        waveval = waveval<0? 0 : waveval;
        waveform[wavep] = (uint8_t) (waveval>255)?255:waveval;
        wavep = (wavep+1) % MAXWAVE;
      }


      void scale() {
        uint8_t maxw = 0;
        uint8_t minw = 255;
        for (int i=0; i<MAXWAVE; i++) {
          maxw = waveform[i]>maxw?waveform[i]:maxw;
          minw = waveform[i]<minw?waveform[i]:minw;
        }
```

```
      uint8_t scale8 = (maxw-minw)/4 + 1;  //scale * 8 to preserve precision
      uint8_t index = wavep;
      for (int i=0; i<MAXWAVE; i++) {
        disp_wave[i] = 31-((uint16_t)(waveform[index]-minw)*8)/scale8;
        index = (index + 1) % MAXWAVE;
      }
    }

void draw(uint8_t X) {
  for (int i=0; i<MAXWAVE; i++) {
    uint8_t y = disp_wave[i];
    oled.drawPixel(X+i, y);
    if (i<MAXWAVE-1) {
      uint8_t nexty = disp_wave[i+1];
      if (nexty>y) {
        for (uint8_t iy = y+1; iy<nexty; ++iy)
        oled.drawPixel(X+i, iy);
        }
        else if (nexty<y) {
          for (uint8_t iy = nexty+1; iy<y; ++iy)
          oled.drawPixel(X+i, iy);
```

```
        }
      }
    }
}

private:
    uint8_t waveform[MAXWAVE];
    uint8_t disp_wave[MAXWAVE];
    uint8_t wavep = 0;

} wave;

int  beatAvg;
int  SPO2, SPO2f;
int  voltage;
bool filter_for_graph = false;
bool draw_Red = false;
uint8_t pcflag =0;
uint8_t istate = 0;
uint8_t sleep_counter = 0;
```

```
void button(void){
   pcflag = 1;
}

void checkbutton(){
   if (pcflag && !digitalRead(BUTTON)) {
     istate = (istate +1) % 4;
     filter_for_graph = istate & 0x01;
     draw_Red = istate & 0x02;
     EEPROM.write(OPTIONS, filter_for_graph);
     EEPROM.write(OPTIONS+1, draw_Red);
   }
   pcflag = 0;
}

void go_sleep() {
   oled.fill(0);
   oled.off();
   delay(10);
   sensor.off();
   delay(10);
```

```
    cbi(ADCSRA, ADEN);  // disable adc
    delay(10);
    pinMode(0,INPUT);
    pinMode(2,INPUT);
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);
    sleep_mode();  // sleep until button press
    // cause reset
    setup();
}

void draw_oled(int msg) {
    oled.firstPage();
    do{
    switch(msg){
        case 0:  oled.drawStr(10,0,F("Device error"),1);
                 break;
        case 1:  oled.drawStr(0,0,F("PLACE YOUR"),2);
                 oled.drawStr(25,18,F("FINGER"),2);
                 break;
        case 2:  print_digit(86,0,beatAvg);
                 oled.drawStr(0,3,F("PULSE RATE"),1);
```

```
            oled.drawStr(11,17,F("OXYGEN"),1);
            oled.drawStr(0,25,F("SATURATION"),1);
            print_digit(73,16,SPO2f,' ',3,2);
            oled.drawChar(116,16,'%',2);
            break;
      case 3:  oled.drawStr(33,0,F("Pulse"),2);
            oled.drawStr(17,15,F("Oximeter"),2);
            break;
      case 4:  oled.drawStr(28,12,F("OFF IN"),1);
            oled.drawChar(76,12,10-sleep_counter/10+'0');
            oled.drawChar(82,12,'s');
            break;
      }
   } while (oled.nextPage());
}

void setup(void) {
  pinMode(LED, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);
  filter_for_graph = EEPROM.read(OPTIONS);
  draw_Red = EEPROM.read(OPTIONS+1);
```

```
  oled.init();
  oled.fill(0x00);
  draw_oled(3);
  delay(3000);
  if (!sensor.begin())  {
    draw_oled(0);
    while (1);
  }
  sensor.setup();
  attachInterrupt(digitalPinToInterrupt(BUTTON),button, CHANGE);
}

long lastBeat = 0;    //Time of the last beat
long displaytime = 0; //Time of the last display update
bool led_on = false;

void loop()  {
    sensor.check();
    long now = millis();   //start time of this cycle
    if (!sensor.available()) return;
    uint32_t irValue = sensor.getIR();
```

```
uint32_t redValue = sensor.getRed();
sensor.nextSample();
if (irValue<5000) {
    voltage = getVCC();
    checkbutton();
    draw_oled(sleep_counter<=50 ? 1 : 4); // finger not down message
    delay(200);
    ++sleep_counter;
    if (sleep_counter>100) {
      go_sleep();
      sleep_counter = 0;
    }
} else {
    sleep_counter = 0;
    // remove DC element
    int16_t IR_signal, Red_signal;
    bool beatRed, beatIR;
    if (!filter_for_graph) {
       IR_signal =  pulseIR.dc_filter(irValue) ;
       Red_signal = pulseRed.dc_filter(redValue);
       beatRed = pulseRed.isBeat(pulseRed.ma_filter(Red_signal));
```

```
  beatIR =  pulseIR.isBeat(pulseIR.ma_filter(IR_signal));
} else {
  IR_signal =  pulseIR.ma_filter(pulseIR.dc_filter(irValue)) ;
  Red_signal = pulseRed.ma_filter(pulseRed.dc_filter(redValue));
  beatRed = pulseRed.isBeat(Red_signal);
  beatIR =  pulseIR.isBeat(IR_signal);
}
// invert waveform to get classical BP waveshape
wave.record(draw_Red ? -Red_signal : -IR_signal );
// check IR or Red for heartbeat
if (draw_Red ? beatRed : beatIR){
   long btpm = 60000/(now - lastBeat);
   if (btpm > 0 && btpm < 200) beatAvg = bpm.filter((int16_t)btpm);
   lastBeat = now;
   digitalWrite(LED, HIGH);
   led_on = true;
   // compute SpO2 ratio
   long numerator   = (pulseRed.avgAC() * pulseIR.avgDC())/256;
   long denominator = (pulseRed.avgDC() * pulseIR.avgAC())/256;
   int RX100 = (denominator>0) ? (numerator * 100)/denominator : 999;
   // using formula
```

```
            SPO2f = (10400 - RX100*17+50)/100;
            // from table
            if ((RX100>=0) && (RX100<184))
              SPO2 = pgm_read_byte_near(&spo2_table[RX100]);
        }
        // update display every 50 ms if fingerdown
        if (now-displaytime>50) {
            displaytime = now;
            wave.scale();
            draw_oled(2);
        }

    }
    // flash led for 25 ms
    if (led_on && (now - lastBeat)>25){
        digitalWrite(LED, LOW);
        led_on = false;
     }
}
```