Міністерство освіти і науки України
ДВНЗ «Прикарпатський національний університет імені Василя Стефаника»
Кафедра комп'ютерної інженерії та електроніки
(повна назва кафедри)

Воробій Віталій Дмитрович
Vorobii Vitalii

УДК 004:681.5

Спеціальність 123 «комп'ютерна інженерія»
(шифр та назва спеціальності)

Кваліфікаційна робота
на здобуття освітньо-кваліфікаційного рівня бакалавр
(бакалавр, спеціаліст, магістр)

Мікроконтролерна система оперативного детектування електричних сигналів серця
Microcontroller operating system of cardiac electrical signal detection

Науковий керівник:
кандидат технічних наук,
доцент Голота В.І.

Рецензент:
Доктор фіз.-мат. наук, професор
кафедри матеріалознавства і
новітніх технологій
Рачій Б.І..

Івано-Франківськ
2021

Формат	Зона	Поз.	Позначення	Найменування	К-ть	Прим.
			123.КІ-41.05	Пояснювальна записка	1	

Змн.	Арк.	№ докум.	Підпис	Дата	*123.КІ-41.05*		
Розробив		Воробій В.Д.			Специфікація	*Літ.*	*Арк.* 2 / *Аркушів* 69
Перевірив		Голота В.І.					
Н. Контр.							
Затвердив							

АНОТАЦІЯ

У бакалаврській роботі реалізовано систему оперативного детектування електричних сигналів серця.

За основу взято систему на кристалі ESP-32. У якості аналогового інтерфейсу використано мікросхему AD8232.

Розроблено блок-схему алгоритму, функціональну і електричну принципову схеми пристрою. Також, реалізовано друковану плату цього пристрою і проведено аналіз його переваг і недоліків.

Обґрунтовано економічну доцільність розробленої системи.

SUMMARY

A microcontroller operating system of cardiac electrical signal detection is implemented in the diploma project.

It is based on ESP-32 system on chip. As an analog interface, AD8232 chip was leveraged.

A block diagram of the algorithm, functional and electrical wiring digrams of the device were developed. Printed circuit board of the device was implemented as well as a through analisys of its advantages and disadvantages.

The economic feasibility of developed system was justified.

Міністерство освіти і науки України
Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Фізико-технічний факультет
Кафедра «Комп'ютерної інженерії та електроніки»

Пояснювальна записка

до кваліфікаційної роботи на тему:

Мікроконтролерна система оперативного детектування електричних сигналів серця

					123.КІ-41.05			
Змін.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розробив		Воробій В.Д.			*Пояснювальна записка*	*Арк.*	*Аркуш*	*Аркушів*
Перевірив		Голота В.І.					*5*	*69*
Н. Контр.								
Затверд.								

ЗМІСТ

ВСТУП

Серце – найважливіший орган серцево-судинної системи. Без нього існування організму людини є неможливим, оскільки скорочення серцевого м’яза забезпечує постійний кровотік і життєдіяльність в усіх органах.

Серцево-судинні захворювання це одна з найсерйозніших проблем нашого суспільства. Статистика інфарктів та інсультів показує, що зі зростанням темпу життя пріоритет піклування про здоров’я асимптотично падає до нуля.

Кожна людина врешті решт стикається з цими захворюваннями з віком. Обмінні порушення, вікові зміни органів та тканин породжують гіпертонію, інсульт, серцеву недостатність, гіпертрофію міокарду тощо [1, 13].

Найкращий спосіб уникнути розвиток хвороби - це ліквідація її причини. На щастя, серед факторів ризику є такі, на які людина може вплинути.

Нормалізація маси тіла, оптимізація харчування, формування здорового способу життя, боротьба зі зловживанням алкоголем і палінням суттєво мінімізують ризик захворювання серця.

Головними симптомами такого типу захворювання є:

1. Біль у грудній клітці
2. Задишка
3. Пришвидшене серцебиття
4. Провали в пам’яті [1, 11]

Існує багато методів дослідження стану серця людини. Пацієнт, згідно з проявленими симптомами і його анамнезом, повинен пройти відповідну процедуру за рекомендацією лікаря.

За допомогою такого методу як рентгеноскопія можна з достатньою точністю визначити розмір і форму серця. При цій процедурі реєструють контури серцевої тіні. Для отримання запису зміни границь серцевої тіні в процесі скорочення або під впливом зміни положення тіла застосовують рентгенокімографію. Тобто перед рентгенівською плівкою розміщують

металеву решітку і при кожному скороченні серця її зміщують на ширину щілини. У результаті отримують зубчате відображення серця, що відображає зміну границь серця під час скорочення. Паралельно відбувається опромінення організму.

З використанням ехокардіографії можна отримати морфологічну характеристику серця та окремих його структуру. Наприклад його розмір, товщину стінок, рух клапанів тощо. Це досягається за рахунок запису ультразвукових коливань, що відбиваються від різних поверхонь серця, внутрішніх і зовнішніх поверхонь стінок та клапанів [2, 2].

Серед цих і багатьох інших інструментальних методів дослідження, що наразі використовуються в кардіології, панівне місце посідає електрокардіографія (далі - ЕКГ). Вона незамінна в повсякденній клінічній практиці і допомагає лікарю-кардіологу вчасно діагностувати порушення серцевої провідності та ритму, ішемічну хворобу серця, інфаркт міокарда, гіпертрофію міокарда передсердь і шлуночків тощо. Також ЕКГ застосовують для функціонального дослідження серця.

Часом людина, яка має проблеми з серцем, не має можливості звернутись до лікаря коли відчуває погіршення. А на наступний день знімок ЕКГ може не виявити відхилень, так-як стандартна електрокардіограма фіксує серцеву діяльність протягом відносно короткого часового інтервалу.

Також, за рахунок малорухливого способу життя, симптоми деякого серцевого захворювання можуть малою мірою або взагалі не проявляються. Тут у нагоді може стати мікроконтролерна система оперативного діагностування серця.

Ця робота передбачає розробку пристрою для проведення оперативної діагностики серця.

1. ПРЕДМЕТ ДОСЛІДЖЕННЯ МІКРОКОНТРОЛЕРНОЇ СИСТЕМИ ОПЕРАТИВНОГО ДЕТЕКТУВАННЯ ЕЛЕКТРИЧНИХ СИГНАЛІВ СЕРЦЯ

1.1. Огляд сучасного стану систем і засобів діагностики серця.

На ринку є багато засобів для проведення діагностики серця різного цінового діапазону, точності, енергоефективності тощо. Нижче наведено кілька найпоширеніших варіантів.

1. Електрокардіограф Heaco 100G ECG100G

Це 12-канальний простий і портативний апарат, що користується попитом серед кардіологів та терапевтів. Дає можливість попереднього перегляду кардіограми і друку результату на термопапері у подальшому. Вивід передається на кольоровий сенсорний дисплей сигналу, що заздалегідь обробляється системою фільтрів від завад. Також містить слот під SD карту, на яку можна записати до 100 проведених досліджень. Ціна такого пристрою сягає близько 20000 гривень.

Недоліком цього пристрою є висока ціна.

Рисунок 1.1. Вигляд пристрою Heaco 100G ECG100G

2. Пристрій ЕКГ PIMA GM-10

Цей пристрій призначений для самостійного слідкування за порушеннями серцевого ритму та збереження результату проведення ЕКГ для його подальшого аналізу. Сигнал серця під час дослідження відображається на невеликому дисплеї. Також є можливість передавати сигнал по мережі Bluetooth. Ціна пристрою сягає близько 3000 гривень.

Недоліком цього пристрою є відсутність додаткової системи фільтрів для усунення завад.

Рисунок 1.2. Вигляд пристрою PIMA GM-10

3. Цифровий електрокардіограф “Біомед” iE 3

Це триканальний кардіограф, що оснащений 7-дюймовим дисплеєм високої роздільної здатності, термопринтером, системою фільтрів тощо. Термопринтер забезпечує одночасний друк трьох відведень ЕКГ і значення пульсу в конкретний момент часу. Окрім друку на папір, пристрій здатний передавати сигнал по таким безпровідним технологіям як WiFi та 3G. Формат даних, що описує сигнал, може бути обраним користувачом з-поміж таких варіантів як: ECG, XML, JPEG, DICOM та PDF. Також можна передавати сигнал через такі провідні інтерфейси як LAN і USB та безпосередньо зберігати

результат на SD карту. Вбудований акумулятор забезпечує автономну роботу пристрою не менше ніж як на 2.5 години. Ціна цього пристрою сягає близько 16000 гривень.

Недоліком цього пристрою є висока ціна.

Рисунок 1.3. Вигляд пристрою “Біомед” iE 3

1.2. Опис ЕКГ серця. Норма і патології

Людина, як біологічний об’єкт, є джерелом електромагнітного сигналу. Коли серце людини б’ється – відбувається зміна електричного потенціалу серця, що можна легко представити у вигляді залежності напруги від часу. Цю функцію можна отримати, приклавши кілька електродів до тіла людини за деяким законом. І така крива, що демонструє електричну активність серця зветься *електрокардіограмою* (ЕКГ). [12]

Першим, кому вдалося записати ЕКГ ще в 1906 році був нідерландський вчений *Вілем Ейнтховен*, який також сконструював струнний гальванометр для цього. [11]

Серцевий цикл є розділеним на інтервали та зубці, кожен з яких відображає фазу розповсюдження хвилі збудження у міокарді. При нормі ЕКГ людини повинна мати форму, тотожну рисунку нижче.

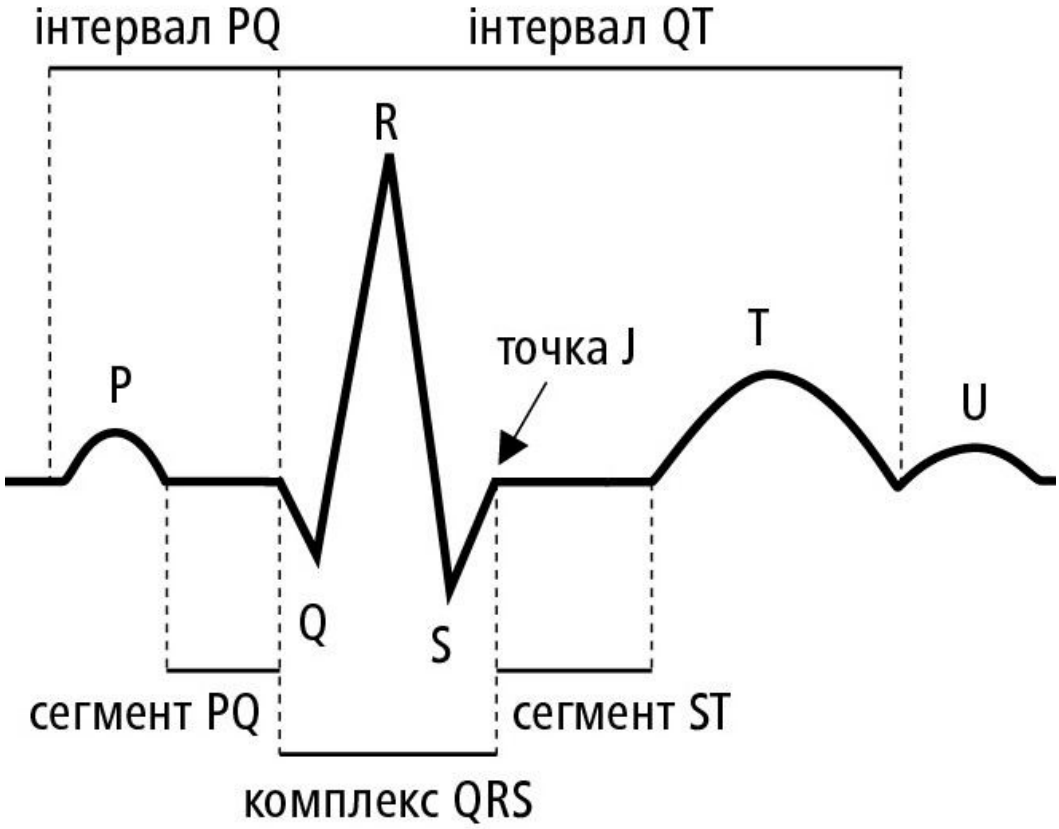


Рисунок 1.4. Нормальна форма ЕКГ

Ритм серця людини за вищенаведеним графіком можна легко з’ясувати за наступною формулою.

$$\beta = \frac{60}{R_1 - R_0} \tag{1.1}$$

де R_1 - R_0 інтервал часу між сусідніми позначками R

У нормі ритм серця досягає значення 60-90 ударів в хвилину у дорослого, 70-80 у дитини і від 135 у новонародженого. [6, 22]

Якщо ритм серця нижчий норми, то таку патологію серця називають *брадикардією , я*кщо вище норми - *тахікардією*. [2]

PR інтервал - це інтервал часу між позначкою P і початком QRS комплексу. У нормі це значення повинно бути в межах від 0.12 до 0.2 секунди. [2, 23]

P хвиля описує деполяризацію передсердь і її тривалість не повинна перевищувати 0.1 секунду. [2, 24]

QRS комплекс описує розподіл збудження тканиною шлуночків. Складається з трьох зубців:

1. Q - негативне відхилення від ізолінії

2. R - позитивне відхилення від ізолінії

3. S - негативне відхилення після зубця R

У нормі тривалість цього комплексу повинна бути в межах від 0.1 до 0.12 секунди.

Хвиля U проявляється у пацієнтів з певними електролітними порушеннями. Наприклад, при *гіпокаліємії*. [12]

Хвиля ST починається з точки J і закінчується початком хвилі T. Вона описує ізолектричний період між деполяризації і реполяризацією шлуночків. ЇЇ тривалість лежить в межах від 5 до 150 мс. У нормі повинна мати невелику увігнутість вгору. В іншому випадку пацієнту може бути притаманна *коронарна недостатність*, *гіпокаліємія* або *отруєння діоксинами*. [15]

1.3. Джерела випромінювань що впливають на засоби діагностики серця

Рівень сигналу ЕКГ є достатньо низьким, відповідно є чутливим до різного типу завад і наведень. Більшість артефактів, що проявляються на ЕКГ, спричинені самою людиною. Наприклад, при недостатньо закріплених електродах вигляд ЕКГ може набути наступної форми. [14]

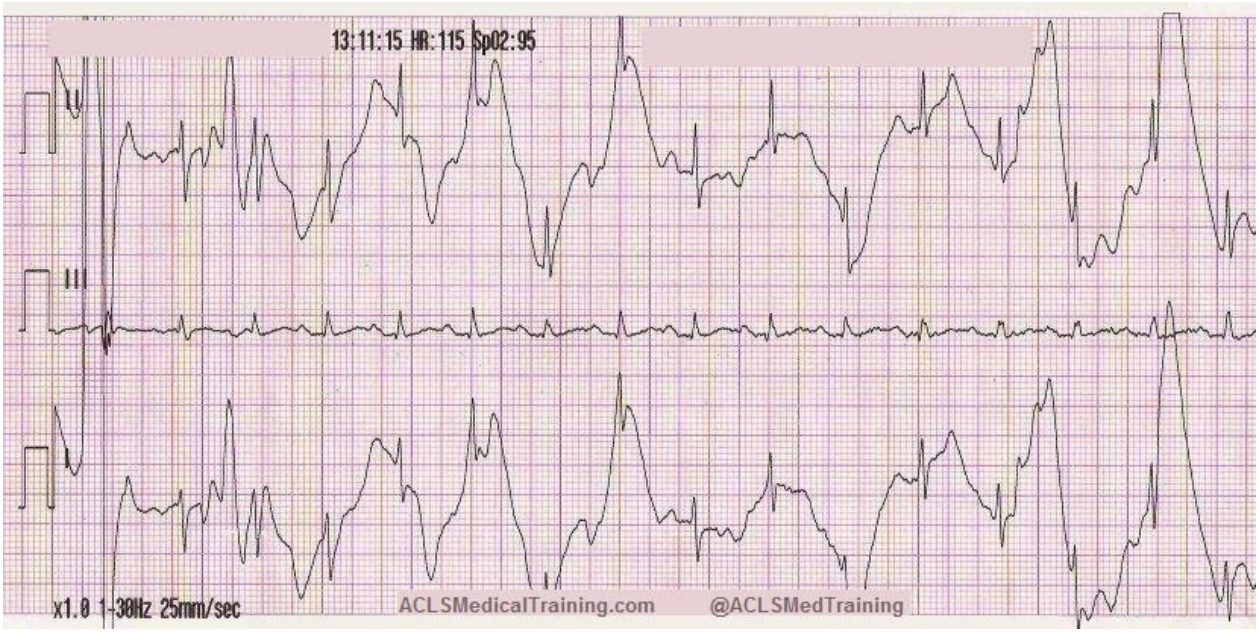


Рисунок 1.5. Вигляд ЕКГ при недостатньо закріплених електродах

Проте, також артефакти на ЕКГ можуть бути спричинені електромагнітними перешкодами.

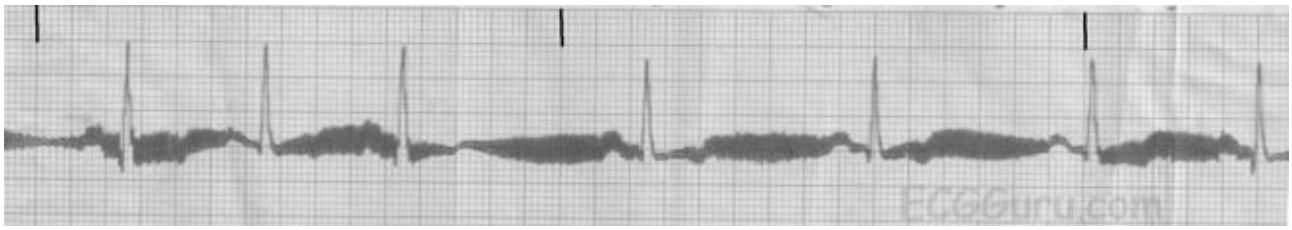


Рисунок 1.6. Вплив ліній електропередачі на ЕКГ

Найбільш суттєвий вплив на вигляд ЕКГ мають лінії електропередачі, електричне обладнання і мобільні телефони. [17]

Найбільш очевидним підходом до усунення цієї проблеми є виділення корисного сигналу методом фільтрації, встановлюючи режим 0.5 - 40 Гц на ЕКГ пристрої. Якщо є необхідність в наявності гармонік вищої частоти в результаті, можна застосувати ежекторний фільтр 50-60 Гц.

Також є інший ряд радіосигналів, що також негативно впливають на результат ЕКГ, і вони наведені нижче та згруповані по діапазону частот в межах від 1 Гц до 5 МГц. [18]

Таблиця 1.1. Поділ діапазонів радіочастот

Номер	Назва діапазону	Діапазон частот	Діапазон довжини хвилі	Приклади застосування
1	Екстремально довгі хвилі	3-30 Гц	100000-10000 км	Комунікація з субмаринами
2	Супердовгі хвилі	30-300 Гц	10000-1000 км	Комунікація з субмаринами
3	Ультра низькі хвилі	300-3000 Гц	1000-100 км	Комунікація між субмаринами, підземний зв’язок
4	Наддовгі хвилі	3-30 кГц	100-10 км	Навігація, геофізика, безпровідні монітори частоти серцевого ритму
5	Довгі хвилі	30-300 кГц	10-1 км	RFID, аматорське радіо
6	Середні хвилі	300-3000 кГц	1000-100 м	Лавинний детектор, середньо-смугові трансляції

1.4. Засіб проектування фільтрів AD8232 BAND PASS DESIGN

AD8232 BAND PASS DESIGN – засіб для зручного проектування фільтрів розроблений компанією *Analog Devices*, що працює під операційною системою Windows. Умовно інтерфейс програми можна розділити на кілька блоків.

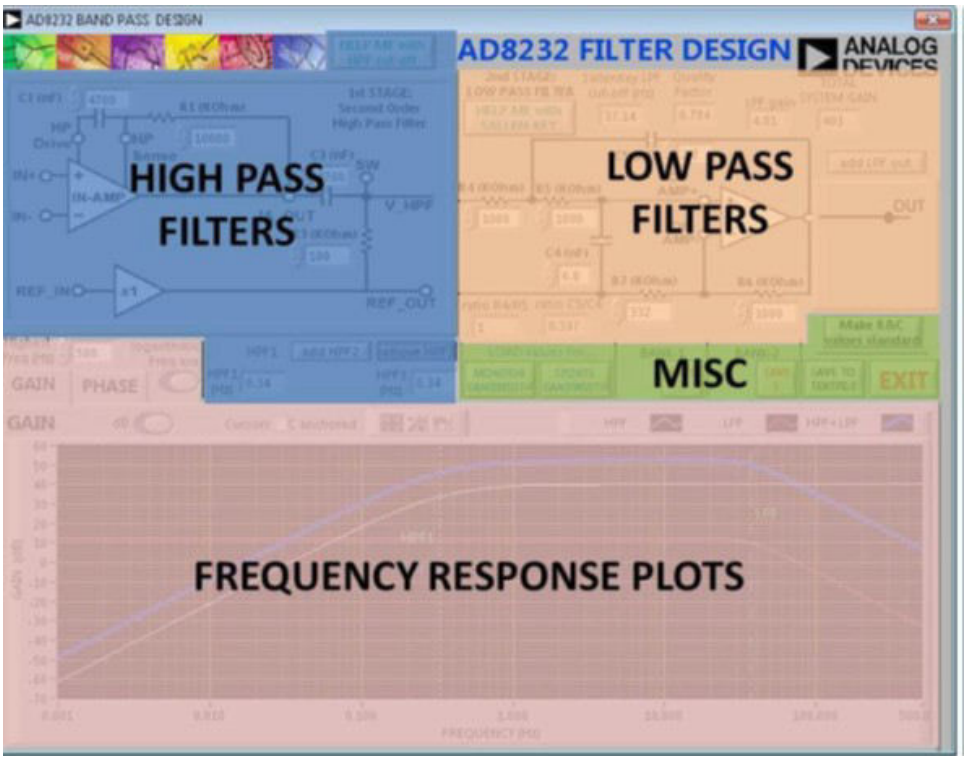


Рисунок 1.7. Структура інтерфейсу програми AD8232 BAND PASS DESIGN

Програма дозволяє користувачу змінювати номінали пасивних елементів, додавати додаткові фільтри високих/низьких частот, зберігати поточну конфігурацію, завантажувати збережену конфігурацію, зберігати конфігурацію у текстовий файл, встановити параметри за замовчуванням.

При зміні параметрів цієї системи знизу відображається графік амплітудно-частотної та/або фазо-частотної характеристики. Додатково динамічно відображаються параметри системи: добротність, коефіцієнт підсилення тощо. [16]

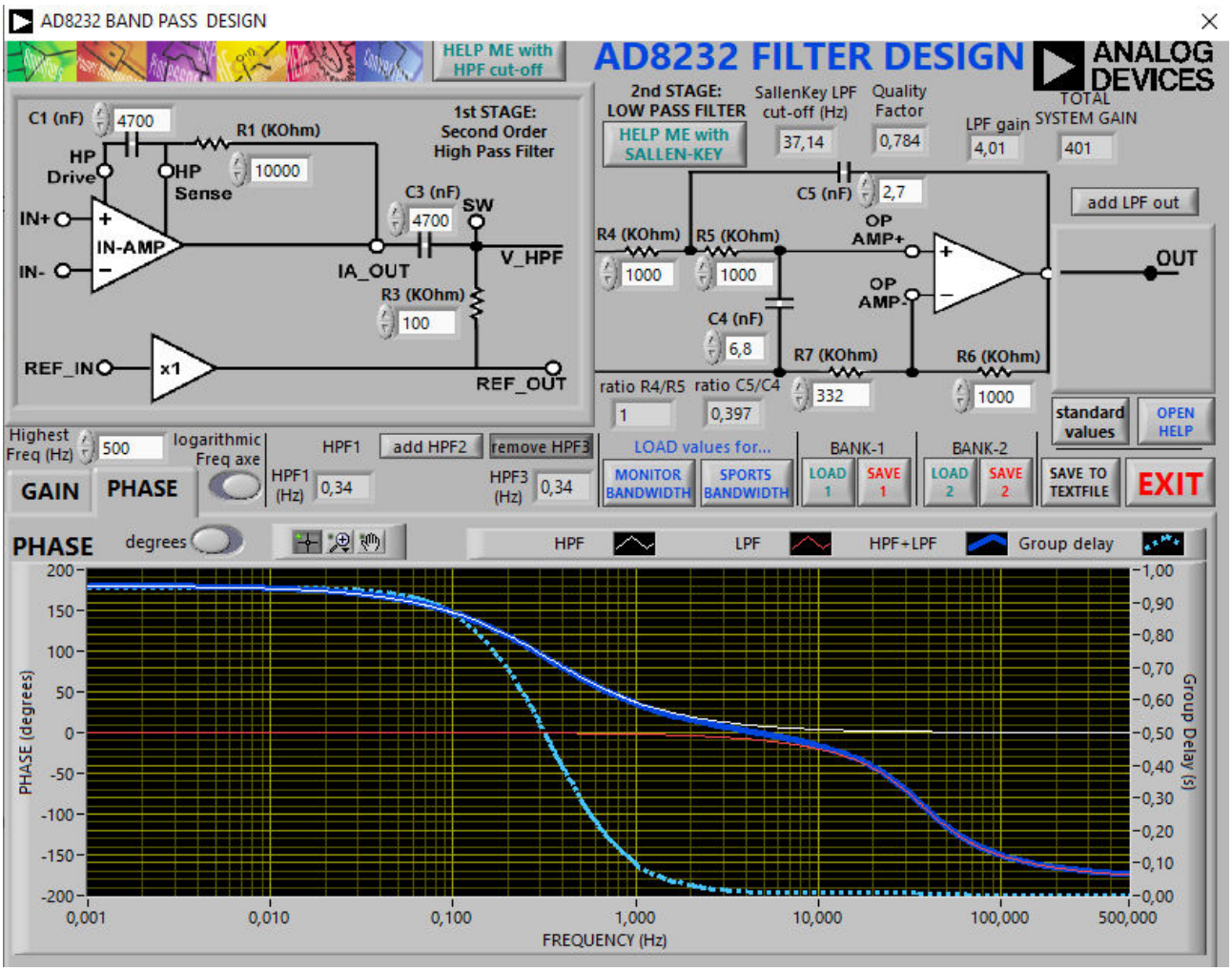


Рисунок 1.8. Інтерфейс програми A8232 BAND PASS DESIGN

Висновки до розділу

У цьому розділі я описав коротко засоби діагностування серця, що доступні на сьогоднішній день. Також, я навів опис ЕКГ і патології, що можна виявити з її допомогою. Розглянуто джерела випромінювання, що можуть спричинити артефакти на ЕКГ і способи мінімізації їх впливу. Здійснено опис засобу проектування фільтрів AD8232 BAND PASS DESIGN.

2. ОПИС ЦИФРОВИХ ФІЛЬТРІВ І ЇХ ЗАСТОСУВАННЯ

2.1. Перетворення Фур'є

Практично все на цьому світі можна представити у вигляді сигналу – звук як приклад механічного коливання що може сприймати людина, електрокардіограма як функція напруги від часу тощо.

Перетворення Фур'є дозволяє поглянути на ці функції з іншого боку, розбиваючи їх на ряд найпростіших періодичних функцій – синусів. Таким чином, перетворення Фур'є повертає функцію як аргумент приймає частоту (зсув фази) і повертає амплітуду відповідного синуса.

Вчений, що довів те що будь-яку функцію можна представити у вигляді нескінченної суми синусів був Жан Батист Жозеф Фур'є. Аналітично це можна представити наступною формулою:

$$g(\omega)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(x)e^{-ix\omega}dx \qquad (2.1)$$

Однак, так-як комп'ютер може оперувати тільки дискретними (скінченними) величинами вищенаведену формулу розрахувати неможливо. У такому випадку слід використовувати модифікацію цього перетворення – дискретне перетворення Фур'є. На вхід цього перетворення подається дискретизована функція, яка часто створюється методом дискретизації (вибірка значень з неперервної функції). [18]

Формула що описує дискретне перетворення Фур'є задана наступним чином:

$$X_k=\sum_{n=0}^{N-1} x_n \cdot e^{\frac{-i2\pi}{N}kn} \qquad (2.2)$$

Користуючись відомою формулою Ейлера, що пов'язує комплексну експоненту з тригонометричними функціями:

$$e^{ix}=\cos(x)+i\cdot\sin(x) \qquad (2.3)$$

Можна описати дискретне перетворення Фур’є наступним чином:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(2\pi kn / N\right) - i \cdot \sin\left(2\pi kn / N\right)\right] \qquad (2.4)$$

де N – кількість значень сигналу, що було виміряно за період, x_n – виміряне значення сигналу в дискретній точці n, X_k – комплексна амплітуда синусоїдального сигналу.

Дискретне перетворення Фур’є відносно легко реалізувати на ЕОМ незалежно від процесора, проте існують спеціалізовані процесори, що здатні швидко виконувати обрахунки цифрової обробки сигналів (Digital Signal Processor).

Рисунок 2.1. Процесор TMS320 від компанії Texas Instruments спеціально призначений для цифрової обробки сигналів

Нижче наведено реалізацію дискретного перетворення Фур'є мовою програмування *Java*:

```
public static double[] discreteFourierTransform(
      double[] samples, int N, boolean forwardPropagation) {
   final double[] X = new double[2 * N];
   final double omega = forwardPropagation ?
          2.0 * Math.PI / N :
         -2.0 * Math.PI / N;
   for (int i = 0; i < N; i++) {
      for (int j = 0; j < N; j++) {
         X[2 * i] +=
                  samples[2 * j] * Math.cos(omega * j * i) +
                        samples[2 * j + 1] * Math.sin(omega * j * i);
         X[2 * i + 1] +=
                 -samples[2 * j] * Math.sin(omega * j * i) +
                        samples[2 * j + 1] * Math.cos(omega * j * i);
      }
   }
   if ( forwardPropagation ) {
      for (int i = 0; i < N; i++) {
         X[2 * i] /= N;
         X[2 * i + 1] /= N;
      }
   }
   return X;
}
```

Існує також видозміна дискретного перетворення – швидке перетворення Фур'є. Асимптотична складність обрахунку дискретного перетворення Фур'є для *N* точок сягає $O(n^2)$, в той час як у випадку швидкого перетворення Фур'є – $O(n \log(n))$.

Нижче наведено реалізацію швидкого перетворення Фур'є мовою програмування *Java*:

```
public static void fastFFT(
double[] samples, int N, boolean forwardPropagation) {
   double xtemp, xr, xi;
   for(int i = 0, j = 0, k = 0; i < N - 1; i++, j += k) {
      if (i < j) {
         double tempr = samples[2 * i];
```

```
            double tempi = samples[2 * i + 1];
            samples[2 * i] = samples[2 * j];
            samples[2 * i + 1] = samples[2 * j + 1];
            samples[2 * j] = tempr;
            samples[2 * j + 1] = tempi;
        }
        k = N / 2;
        while (k <= j) {
            j -= k;
            k >>= 1;
        }
    }
    final double omega = forwardPropagation ?   2.0 * Math.PI / N :
                                                -2.0 * Math.PI / N;
    for(int M = 2; M < 2 * N; M *= 2 ) {
        double sin = Math.sin(omega);
        double cos = Math.cos(omega) - 1.0;
        xr = 1.0;
        xi = 0.0;
        for (int m = 0; m < M - 1; m += 2) {
            for (int i = m; i < 2 * N; i += M * 2) {
                int j = i + m ;
                double tempr = xr * samples[j] – xi * samples[j + 1];
                double tempi = xr * samples[j + 1] + xi * samples[j];
                samples[j] = samples[i] - tempr;
                samples[j + 1] = samples[i + 1] - tempi;
                samples[i] += tempr;
                samples[i + 1] += tempi;
            }
            xtemp = xr;
            xr = xr + xr*cos - xi*sin;
            xi = xi + xtemp*sin +xi*cos;
        }
    }
    if ( forwardPropagation ) {
        for (int i = 0; i < N; i++) {
            samples[2 * i] /= N;
            samples[2 * i + 1] /= N;
        }
    }
}
```

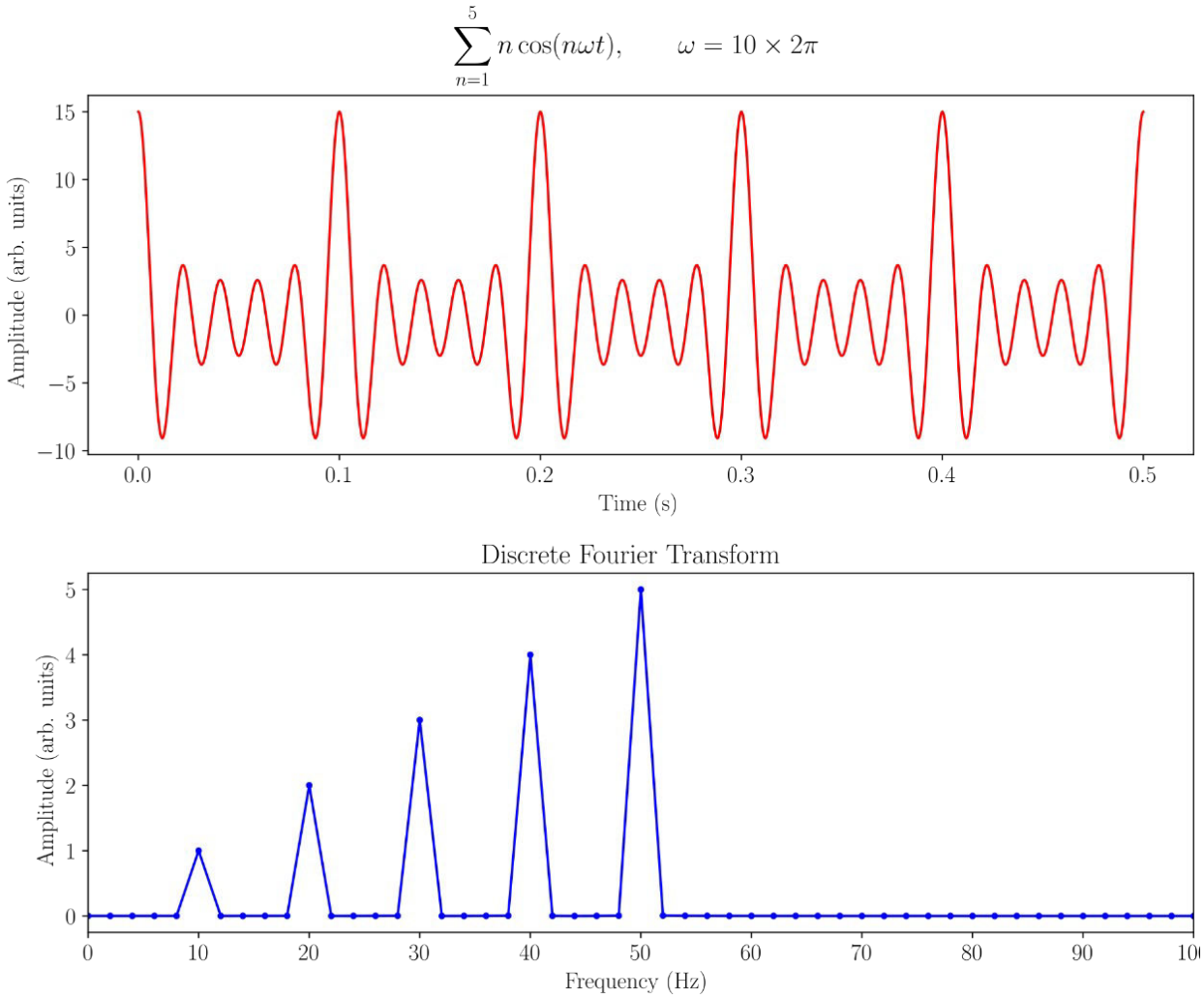


Рисунок 2.2. Дискретне перетворення Фур'є функції суми косинусів

Користь перетворення Фур'є важко переоцінити. Воно широко використовується у багатьох областях науки і техніки, таких як фізика, комбінаторика, криптографія, статистика тощо. [18]

2.2. Побудова і оптимізація цифрових фільтрів

У цифровій обробці сигналів фільтр призначений для усунення деяких небажаних завад, як наприклад випадковий шум, або виділення корисної частини сигналу у межах деякого діапазону частот. Класичний приклад усунення шуму 50-60 Гц від ліній електропередачі можна здійснити за допомогою відповідного фільтра.

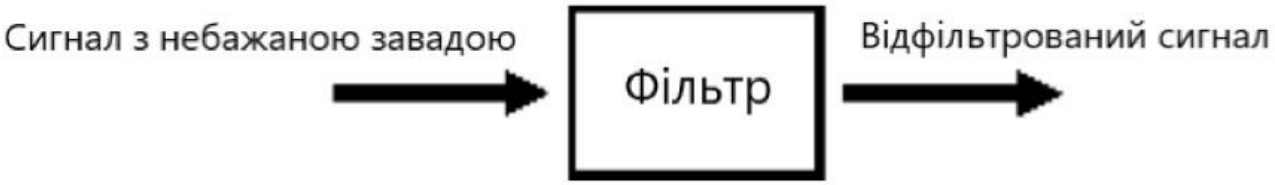


Рисунок 2.3. Ілюстрація процесу фільтрування сигналу

Існує два типи фільтрів – аналогові та цифрові. Відмінність між ними колосальна. Аналоговий фільтр досягається використанням таких компонентів як резистор, конденсатор та операційний підсилювач для отримання відповідного ефекту. Аналогові фільтри часто використовуються у системах відео трансляції, для усунення шуму, у Hi-Fi системах тощо.

На відмінну від аналогових фільтрів, цифрові фільтри виконують математичні обрахунки над дискретизованими значеннями на процесорі для отримання бажаного ефекту. Так-як вхідний сигнал переважно по своїй природі є неперервним, то перед застосуванням фільтру виконується дискретизація сигналу. Це можна здійснити за допомогою аналого-цифрового перетворювача (АЦП). Після виконання обрахунків результат можна перетворити на сигнал електричної величини - струм або напругу. Це досягається за допомогою цифро-аналогового перетворювача (ЦАП). [19]

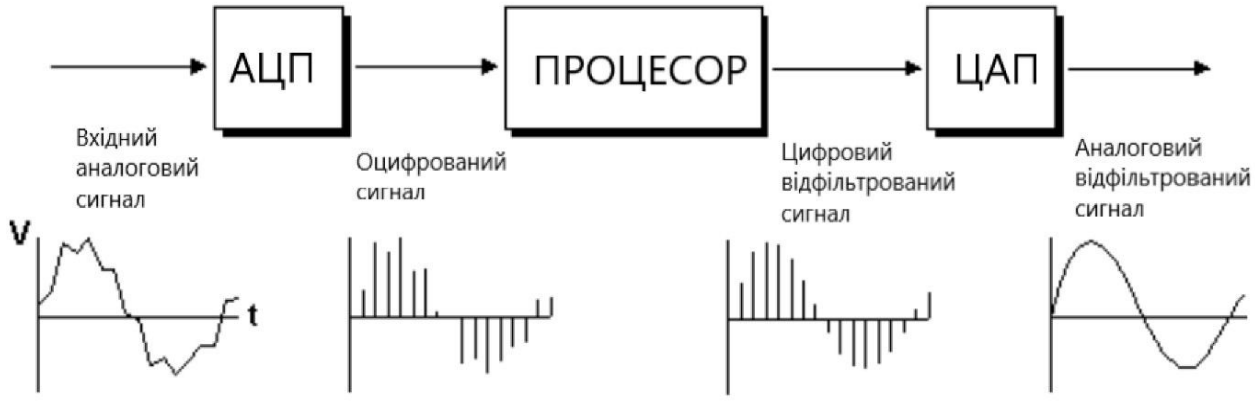


Рисунок 2.4. Ілюстрація фільтрування сигналу цифровим методом

Переваги використання цифрових фільтрів:

1. Цифрові фільтри є програмованими, операції є чітко визначені програмою

збереженою у пам’яті процесора. Таким чином зміна параметрів фільтру не призведе до зміни структури електричної схеми. У випадку аналогових фільтрів – схема кожного разу змінюється при модифікації фільтру

2. Цифрові фільтри значно простіше спроектувати, реалізувати і протестувати.

3. Характеристики аналогових фільтрів залежать від температури (особливо ті що містять активні елементи). Цифровим фільтрам ця проблема не притаманна, вони працюють стабільно незалежно від часу і температури.

4. Цифрові фільтри на відмінну від аналогових значно точніше обробляють низькі частоти. А оскільки швидкість процесорів продовжує зростати – вони все частіше застосовуються для обробки радіо частот.

5. Сучасні цифрові фільтри дозволяють комбінувати фільтри паралельно або послідовно, що значно спрощує розробку електричної схеми.

Найпростіші типи фільтрів:

1. Буферний фільтр (*Unity Gain Filter*)

$$y_n = x_n \tag{2.4}$$

Інакше кажучи, вихідне значення фільтра ідентичне вхідному.

2. Простий підсилювальний фільтр

$$y_n = K \cdot x_n \tag{2.5}$$

Вихідне значення множиться на деякий сталий коефіцієнт. Відповідно, якщо $K > 1$ то фільтр підсилює вхідний сигнал, якщо ж $K < 0$ то послаблює вхідний сигнал.

3. Фільтр, що реалізує затримку вхідного сигналу

$$y_n = x_{n-1} \tag{2.6}$$

Вихідне значення фільтру в час $t = n\,h$ - це вхідне значення фільтру в час $t = (n - 1)\,h$.

4. Диференціальний фільтр

$$y_n = x_n - x_{n-1} \tag{2.7}$$

Кожне вихідне значення фільтру в час $t = n\ h$ – різниця значення сигналу в $t = n\ h$ і $t = (n - 1)\ h$. Ідентичний ефект досягається з використанням *диференціального підсилювача*.

5. Фільтр середнього значення другого порядку

$$y^n = \frac{x_n + x_{n-1}}{2} \tag{2.8}$$

Вихід фільтру є середнім арифметичним поточного і попереднього вхідного значення. Цей фільтр є простим фільтром нижніх частот, що згладжує високочастотні гармоніки у вхідному сигналі.

Цифрові фільтри можна розділити на два класи – фільтри з нескінченною імпульсною характеристикою (НІХ-фільтри) і фільтри з скінченною імпульсною характеристикою (СІХ-фільтри). У стандартній формі ці фільтри можна відобразити через коефіцієнти його імпульсної характеристики *h(k)*. Вхідний і вихідний сигнал пов’язані з допомогою функції згортки. Даний зв’язок приведений в формулі (2.9) для СІХ-фільтрів і в формулі (2.10) для НІХ-фільтрів.

$$y(n) = \sum_{k=0}^{\infty} h(k) \cdot x(n-k) \tag{2.9}$$

$$y(n) = \sum_{k=0}^{N-1} h(k) \cdot x(n-k) \tag{2.10}$$

З вищенаведених зрозуміло, що імпульсна характеристика СІХ-фільтрів має нескінченну тривалість, тоді як для НІХ-фільтрів є чітко детермінована, так-як *h(k)* для НІХ-фільтра може приймати тільки *N* значень. На практиці обчислювання вихідного значення за допомогою формули вище для СІХ-фільтрів є неможливим, так-як тривалість імпульсного виходу є занадто

великою (у теорії нескінченною). Замість цього переписують рівняння наступним чином.

$$y(n)=\sum_{k=0}^{\infty} h(k)\,x(n-k)=\sum_{k=0}^{N} b_k\,x(n-k)-\sum_{k=1}^{M} a_k\,y(n-k) \qquad (2.11)$$

де a_k і b_k коефіцієнти фільтра.

Вибір між ними залежить від їх окремих переваг:

1. СІХ-фільтри мають виключно лінійну фазову характеристику. Це означає, що фільтр не вводить фазового спотворення на виході, що важливо в таких сферах як: передача даних, цифрова аудіо обробка, біомедицина або обробка зображень. У випадку НІХ-фільтрів, ФЧХ є нелінійною, особливо на межах смуги.
2. СІХ-фільтри є завжди стійкими за рахунок нерекурсивної реалізації.
3. Для реалізації фільтрів застосовується скінченне число бітів. Відповідно, шуми округлення і помилки квантування є більш значущими для НІХ-фільтрів.
4. СІХ-фільтр потребує більше коефіцієнтів для реалізації характеристики з різким зрізом частот, ніж НІХ-фільтр. З цього випливає, що для СІХ-фільтру потрібно більше обчислювальної потужності і пам’яті, а це може грати ключову роль при виконанні алгоритму на пристрою з обмеженими апаратними ресурсами.
5. Аналогові фільтри значно легше перетворити на еквівалентний НІХ-фільтр, ніж на СІХ-фільтр.
6. Синтез СІХ-фільтрів математично складніший. Особливо без використання комп’ютерної підтримки для розробки.

2.3. Приклад перетворення Фур’є для електрокардіограми

Не зважаючи на складність математики, що лежить в основі перетворення Фур’є, це перетворення досить легко здійснити з допомогою сучасних програмних пакетів. Сучасні програмні пакети *інкапсулюють* складність реалізації математичних функцій, надаючи кінцевим користувачам простий

інтерфейс для взаємодії. Таким чином, можна працювати на значно вищому рівні абстракції.

Мова програмування, що підтримує найбільшу кількість пакетів є безсумнівно Python. В прикладі, наведеному нижче я скористався пакетом для векторних обрахунків NumPy, пакетом для чисельних алгоритмів SciPy і пакетом для візуалізації Matplotlib.

Заздалегідь я підготував вхідні дані — запис *16265* з бази даних *MIT-BIH Normal Sinus Rhythm Database.* Частота дискретизації у цьому випадку сягає 128 Гц. Формат представлених даних — Comma Seperated Values (CSV).

Роздрук програми:

```
import numpy as np
from scipy.fft import fft
import matplotlib.pyplot as plt

# Шлях до файлу, що містить зафіксовані значення ЕКГ
ecg_file_name = 'ecg.csv'

# Роздільник у файлі
delimiter = ','

# Частота дискретизації
sampling_frequency = 128

# Отримання масиву значень з файлу
ecg_data = np.loadtxt(ecg_file_name, delimiter=delimiter)

# Знаходження довжини масиву
N = len(ecg_data)

# Отримання періоду
T = 1.0 / sampling_frequency

# Часовий вектор
x = np.linspace(0.0, N*T, N)

# Обрахунок перетворення Фур'є
yf = fft(ecg_data)
```

```
# Знаходження частотного вектора
xf = np.linspace(0.0, 1.0/(2.0*T), N//2)

fig, axs = plt.subplots(2, 1)

# Візуалізація отриманого результату

axs[0].plot(x, ecg_data)
axs[0].set_ylabel("Напруга (Вольт)")
axs[0].set_xlabel("Час (мсек)")

axs[0].grid()

axs[1].plot(xf, 2.0/N * np.abs(yf[0:N//2]))

axs[1].set_ylabel("Амплітуда")
axs[1].set_xlabel("Частота (Гц)")

axs[1].grid()

fig.tight_layout()
plt.show()
```

Результат отриманий під час виконання програми:

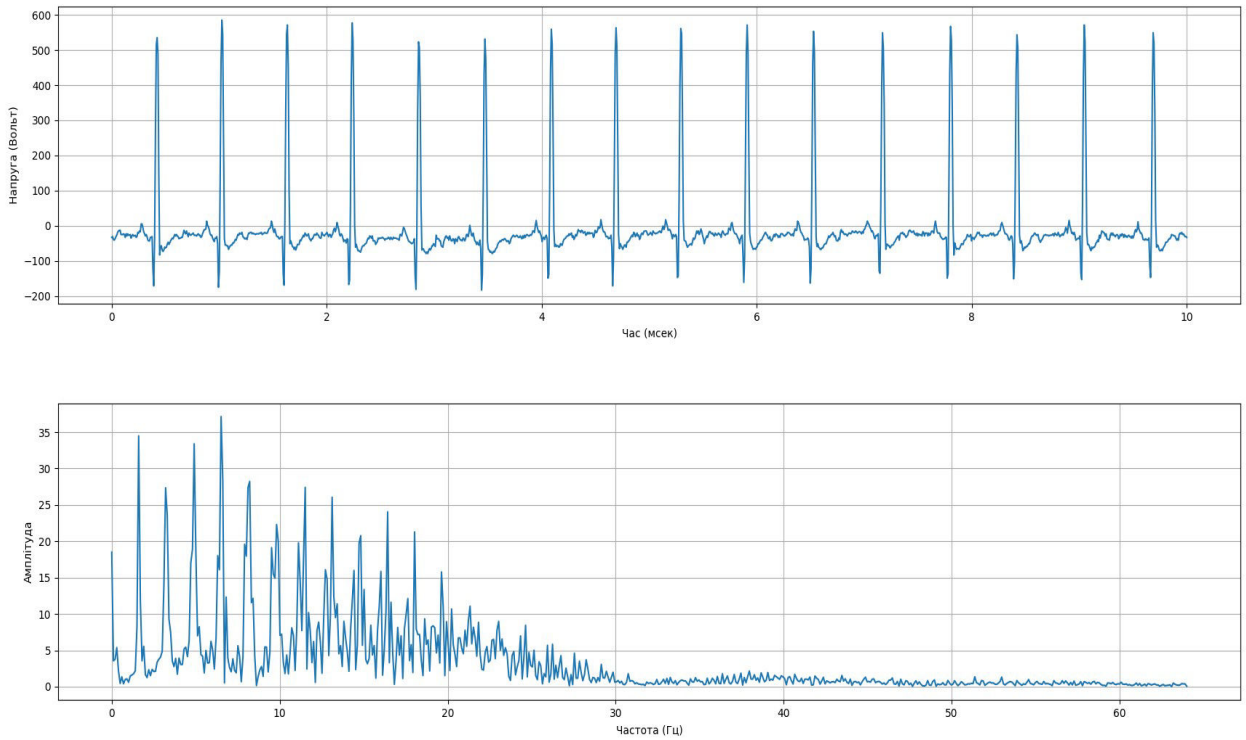


Рисунок 2.5. Результат виконання

Верхній графік відображає вхідні дані, нижній відображає результат перетворення Фур’є.

Висновки до розділу

У цьому розділі я описав математичний апарат перетворення Фур'є і навів приклад перетворення Фур'є сигналу електрокардіограми мовою Python з допомогою допоміжних пакетів. Також, розглянув цифрові фільтри та порівняв їх з аналоговими фільтрами.

3. РЕАЛІЗАЦІЯ СТРУКТУРНОЇ ТА ФУНКЦІОНАЛЬНОЇ СХЕМ

3.1. Розроблення структурної схеми ЕКГ пристрою

Структурна схема - це схема, що відображає основні функціональні компоненти пристрою та взаємозв’язки між ними без деталізації. [20]

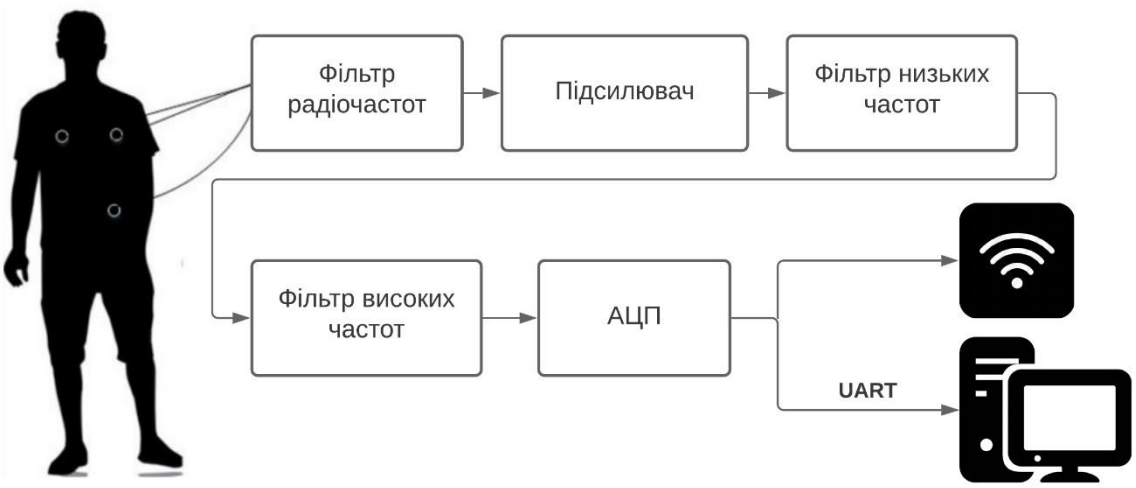


Рисунок 3.1. Структурна схема ЕКГ пристрою

У нашому випадку над сигналом, що надходить з електродів, застосовується радіочастотний фільтр для усунення небажаних радіочастотних шумів. Після фільтрації сигнал підсилюється, бо сигнал, який надходить від тіла людини є надзвичайно слабким. Після для виділення корисного сигналу з-поміж шуму застосовуються фільтри низьких і високих частот. У результаті сигнал проходить через аналого-цифровий перетворювач і паралельно транслюється по UART протоколу і Wi-Fi мережі.

3.2. Розроблення функціональних схем структурних блоків

На функціональній схемі містяться функціональні частини пристрою з можливою деталізацією.

3.2.1. Сенсорний блок

У дев’ятнадцятому сторіччі вчені виявили, що серцю притаманний деякий електричний потенціал і при своїй роботі воно виділяє електричну енергію певної величини. Таким чином, встановивши електроди, що проводять струм, на тілі людини, можна реєструвати значення напруги серця в певний момент часу.

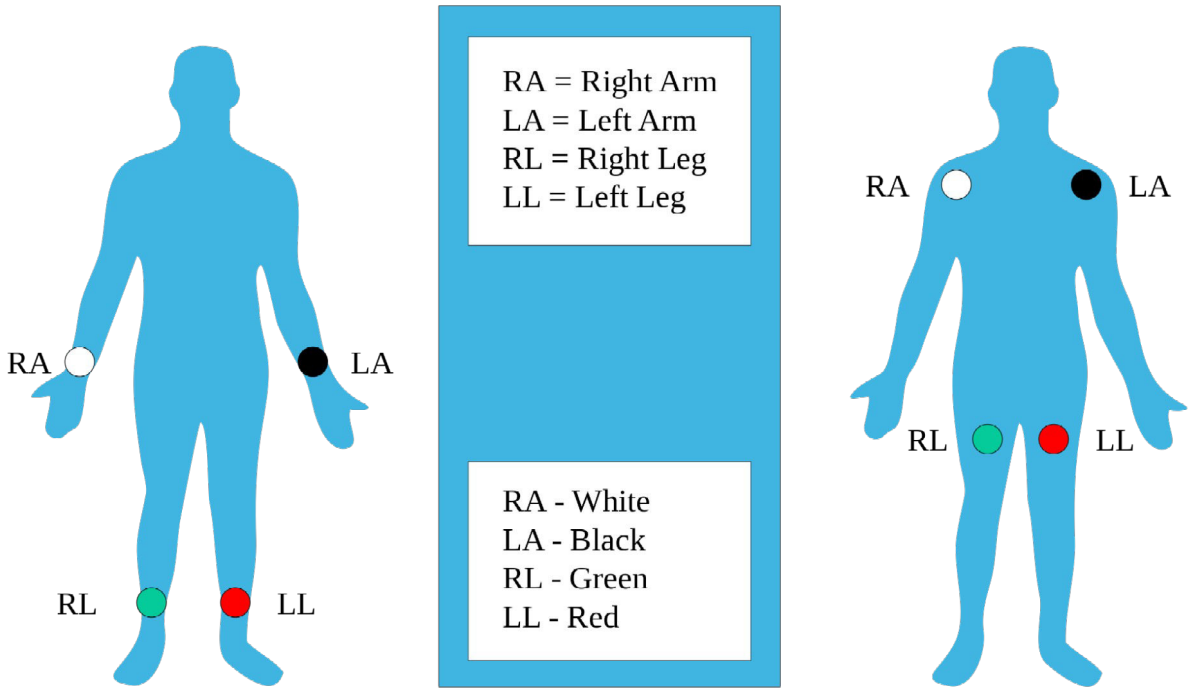


Рисунок 3.2. Схема накладення електродів на тіло людини

Також, на ринку є кабелі, що містять електроди на одному кінці і TRS-роз’єм на іншому. Виглядають вони схожим чином. [12]

Рисунок 3.3. ЕКГ кабель

У нашому пристрої можна виділити інтерфейс для TRS-роз’єму, на який буде надходити сигнал людини. Це забезпечує портативність при транспортуванні і зберіганні пристрою, а також це дозволить значно легше полагодити пошкодження кабелю. Схематично реалізувати це можна наступним чином.

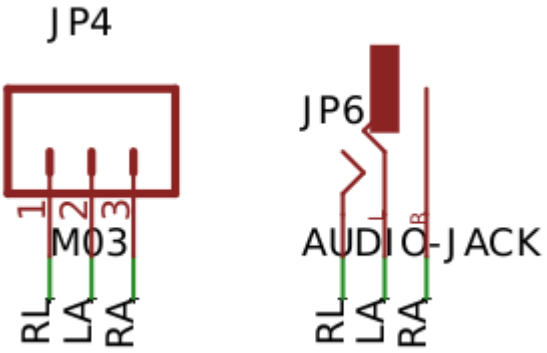


Рисунок 3.4. Інтерфейс для TRS — роз’єму

3.2.2. Блок підсилення та фільтрування сигналу

Сигнал, що надходить від тіла людини є надзвичайно слабким. Тому, необхідна схема, яка фіксує, фільтрує, підсилює і перетворює цей слабкий аналоговий сигнал. У цьому проекті в якості такої схеми я використав аналоговий процесор AD8232 від фірми *Analog Devices*.

AD8232 – інтегральний блок перетворення сигналів для ЕКГ або будь-яких інших біологічних сигналів. Він, головним чином, складається з інструментального та операційного підсилювачів. На людське тіло, як завада, наводяться електромагнітні сигнали з ліній електропередачі частотою 50-60 Гц, тому в AD8232 використаний метод диференційного зменшення впливу синфазних завад. Також вбудована система швидкого відновлення схеми зі стану насичення. Присутні незадіяні виходи операційних підсилювачів, що можна використати для додаткового фільтрування сигналу. [16]

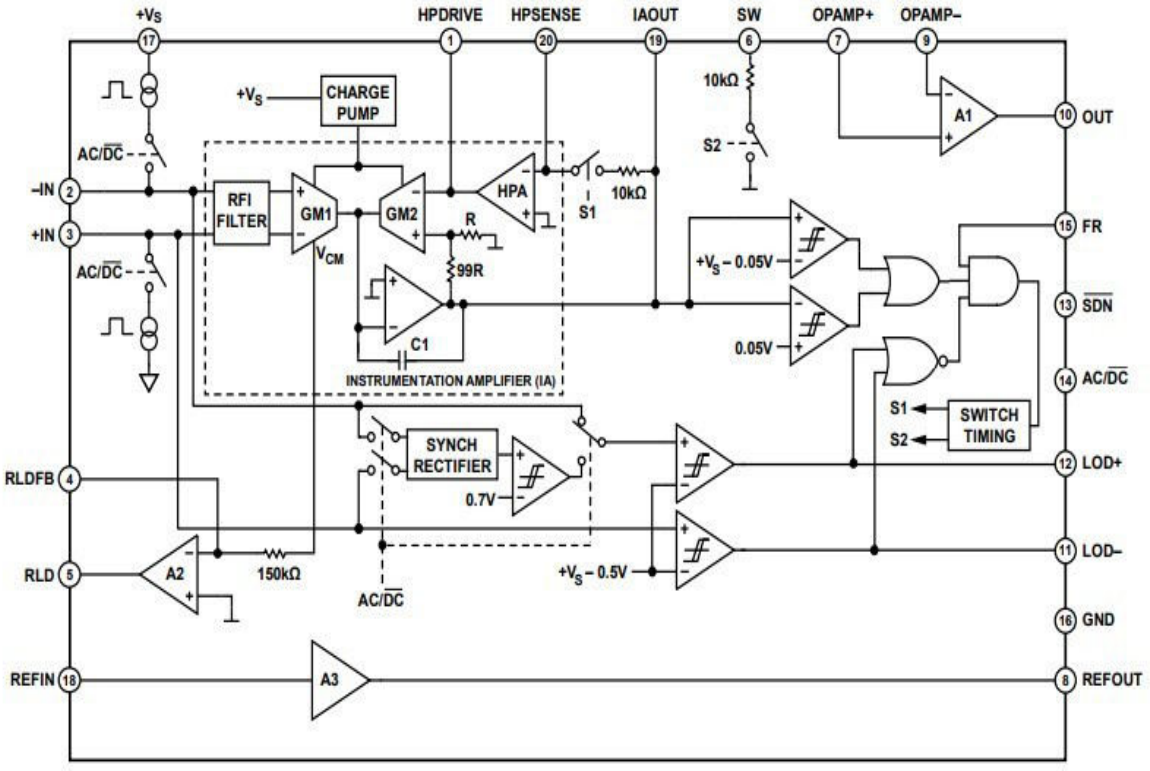


Рисунок 3.5. Функціональна діаграма AD8232

Вхідний інструментальний підсилювач можна використовувати як підсилювач і як фільтр високих частот.

Коефіцієнт підсилення інструментального підсилювача є сталим.

$$K_1 = 100 \tag{3.1}$$

Для того, щоб інструментальний підсилювач фільтрував занадто малі значення напруги на вході потрібно утворити *RC* коло між виводами *HPSENSE*, *HPDRIVE* і безпосередньо виходом інструментального підсилювача.

Реалізація наведена на рисунку нижче.

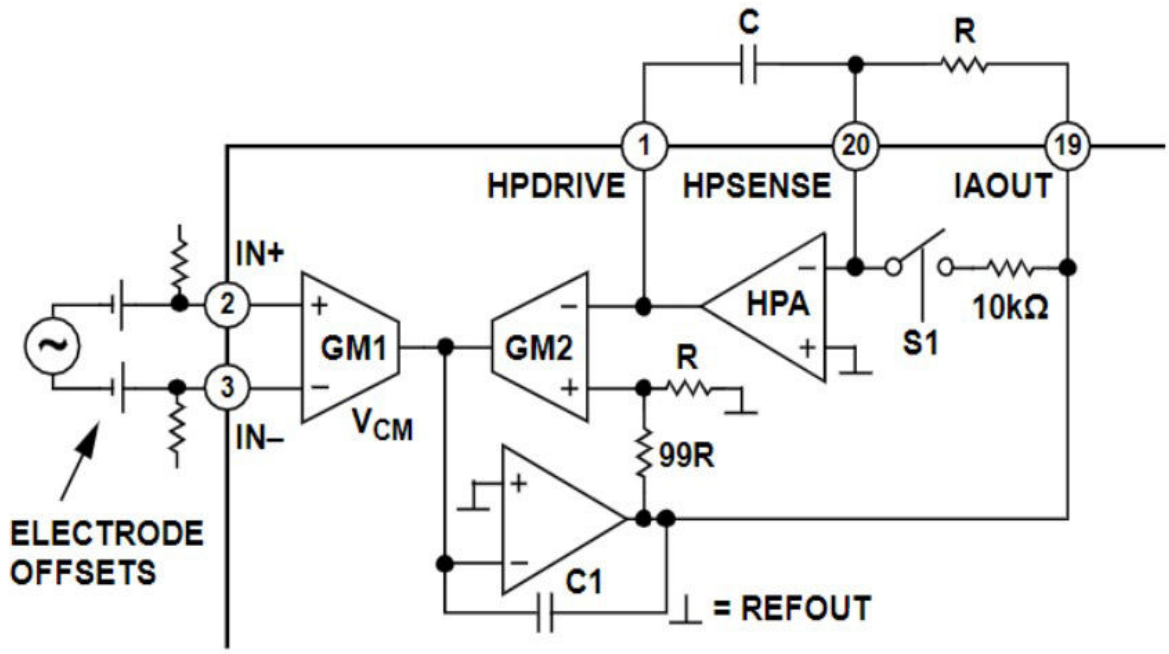


Рисунок 3.6. Схема фільтру високих частот першого порядку

RC-коло формує інтегратор, що передає не бажані сигнал назад у інструментальний підсилювач, фільтруючи при цьому сигнал і підтримуючи коефіцієнт підсилення сталим у спроектованому діапазоні частот. Значення ємності та опору визначають смугу пропускання даного активного фільтра.

Додатково інтегратор мінімізує ефект повільно наростаючих сигналів, таких як *“Блукання осі”* , що виникає при різких рухах людини або ж через механічної деформації електронних провідників, що викликає неочікувану зміну електродних потенціалів.

Частота зрізу при цьому задається формулою:

$$f_1 = \frac{K1}{2\pi RC} = \frac{100}{2\pi RC} \quad (3.2)$$

Як раніше зазначалось, мікросхема дозволяє реалізовувати додаткові фільтри. У фільтрах вищих порядків нахил АЧХ з більш різкими фронтами, проте ми отримуємо значно вищий рівень фазових спотворень, які виникають за рахунок наявності реактивного опору ємності. Ще одним недоліком є наявність більшої кількості елементів на друкованій платі.

Фільтр другого порядку в нашому випадку реалізується досить просто. Достатньо добавити ще одну *RC-ланку* до виходу інструментального підсилювача. Приклад такої реалізації наведений нижче.

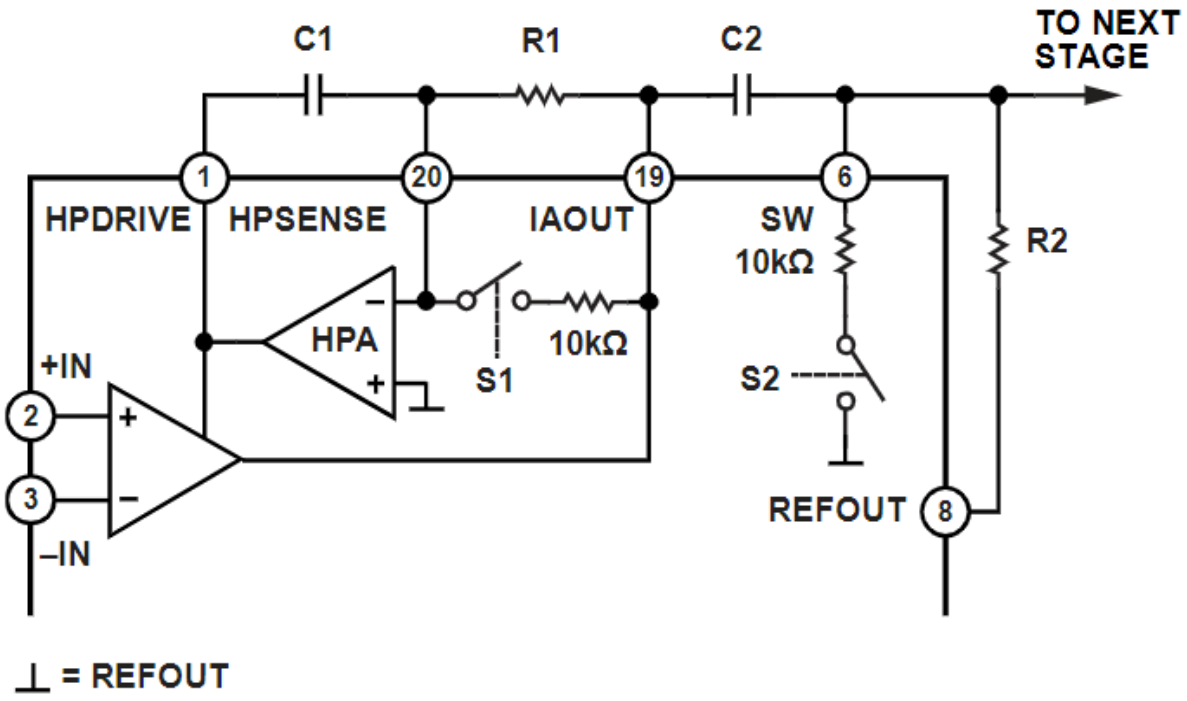


Рисунок 3.7. Схема фільтру високих частот другого порядку

Також мікросхема містить некомутований операційний підсилювач, що також здатний додатково підсилити сигнал і, відповідно, його відфільтрувати.

У найпростішому випадку коли фільтр високих частот взагалі не потрібен, простого *RC*-фільтра низьких частот цілком вистачить.

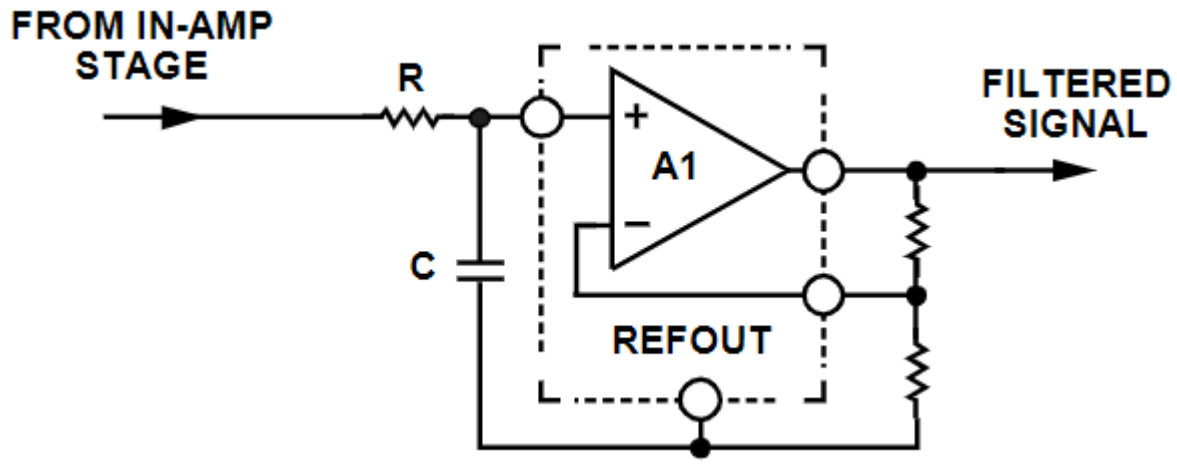


Рисунок 3.8. Схема найпростішого фільтру низьких частот

Якщо ж потрібно задати порогове значення частоти – є можливість реалізувати *фільтр Саллена-Кі*, що представляє собою активний електронний фільтр другого порядку.

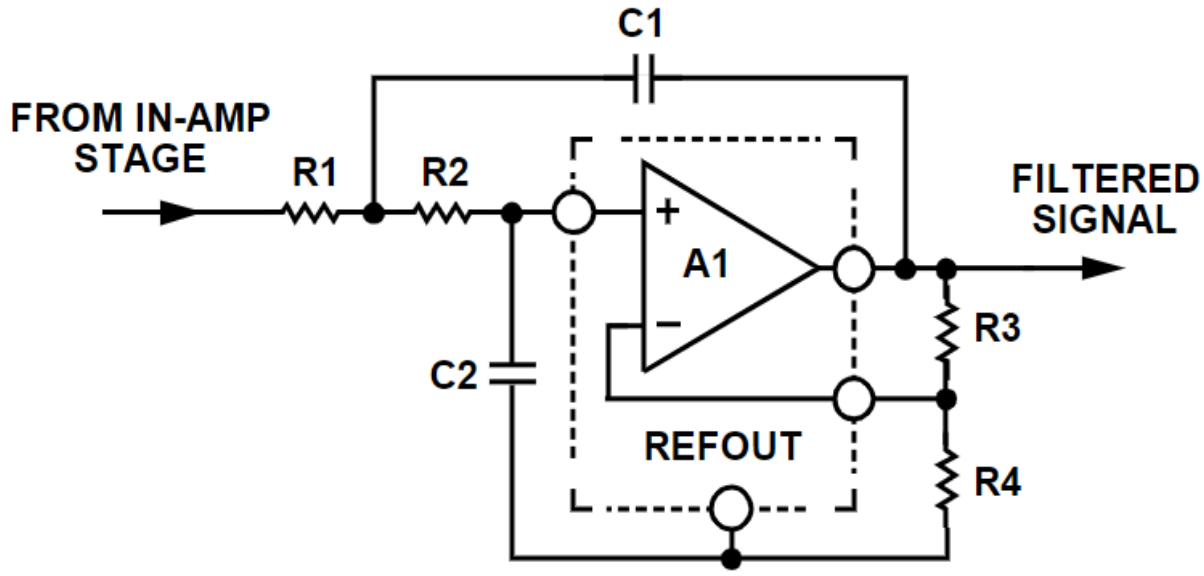


Рисунок 3.9. Схема фільтру Саллена-Кі

Наступні рівняння описують частоту зрізу, коефіцієнт підсилення і добротність цього фільтру.

$$f_2 = \frac{1}{2\pi\sqrt{R_1 C_1 R_2 C_2}} \qquad (3.3)$$

$$K_2 = 1 + \frac{R_3}{R_4} \tag{3.4}$$

$$Q = \frac{\sqrt{R_1 C_1 R_2 C_2}}{R_1 C_2 + R_2 C_2 + R_1 C_1 (1 - K_2)} \tag{3.5}$$

З цих рівнянь можна помітити, що змінюючи коефіцієнт підсилення ми впливаємо на добротність і навпаки.

Також при знятті сигналу ЕКГ необхідно врахувати вплив радіочастотного випромінювання. Конкретно проблема проявляється у напрузі зміщення на виході вхідного інструментального підсилювача. Для вирішення цієї проблеми AD8232 на двох входах цього інструментального підсилювача містить конденсатор ємністю близько 15 пФ і резистор опором 10 кОм. Таким чином формується фільтр низьких частот, що очевидно зменшує вплив випромінювання.

Таким чином частота зрізу РЧ-фільтру:

$$f_C = \frac{1}{2\pi RC} \approx 1.06\ МГц \tag{3.6}$$

Варто зауважити, що реальна частота зрізу буде значно нижчою за рахунок вищого вхідного опору, який необхідний для захисту пацієнта і захисту від перевантаження.

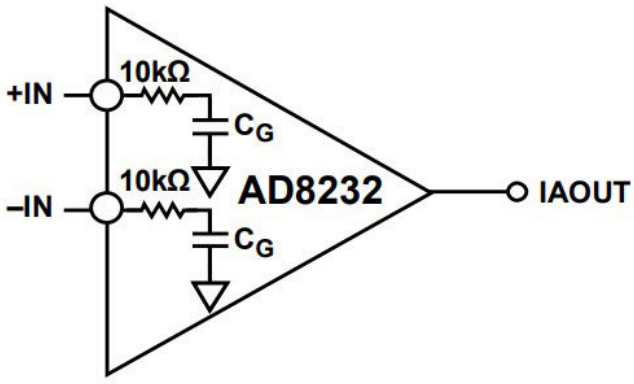


Рисунок 3.10. Фільтр радіочастотного випромінювання

Як бачимо у таблиці 1.1., джерела радіовипромінювання можуть спричинити завади у роботі нашого пристрою на відстані від 100 м до 100000 км. Проте, як раніше зазначено, ліва межа цього діапазону буде вища за рахунок додаткового зовнішнього резистора.

У проекті я сконфігурував AD8232 фільтром низьких частот другого порядку з частотою зрізу 40 Гц і фільтром високих частот другого порядку з частотою зрізу 0.5 Гц. Додатково, коли сигнал проходить через операційний підсилювач, він підсилюється у 11 разів. Таким чином, загальний коефіцієнт підсилення:

$$M = K_1 \cdot K_2 = 1100 \frac{B}{B} \tag{3.7}$$

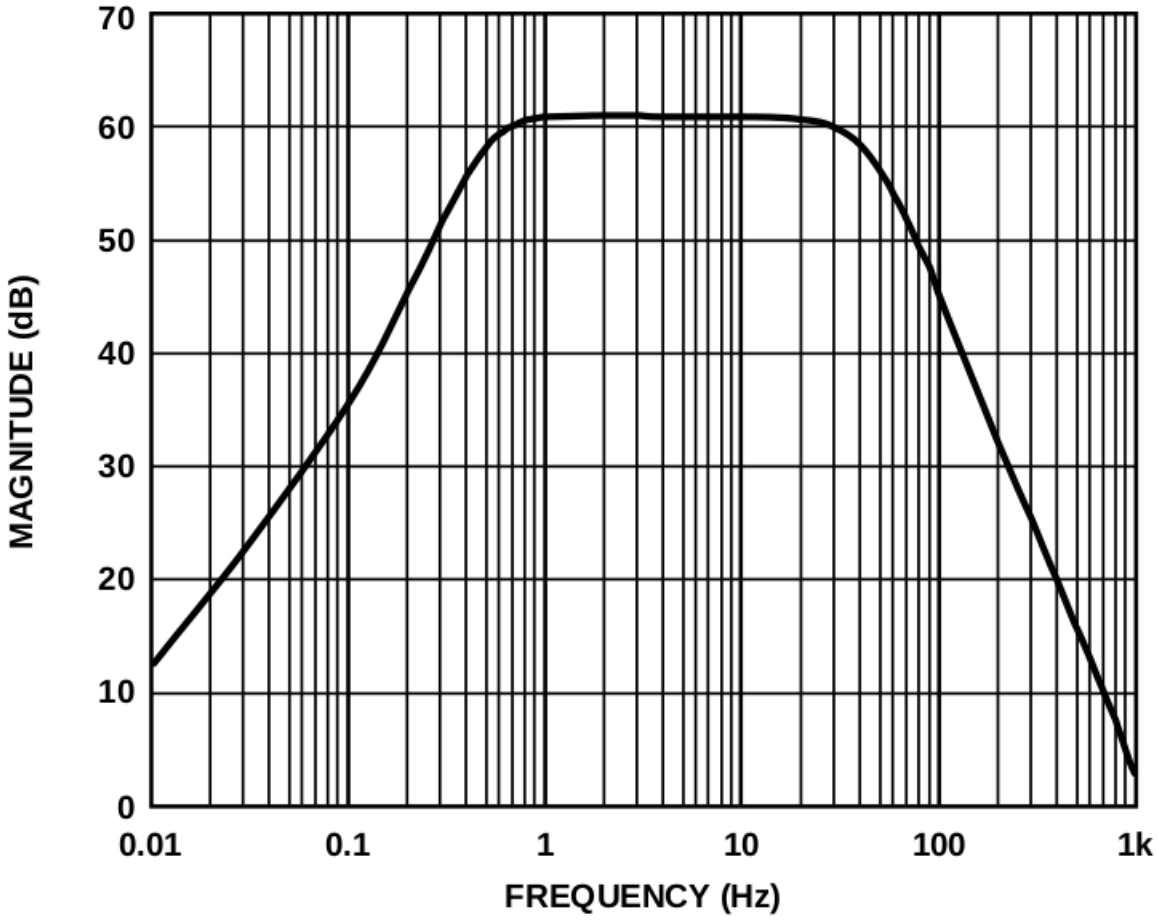


Рисунок 3.11. Амплітудна-частотна характеристика AD8232

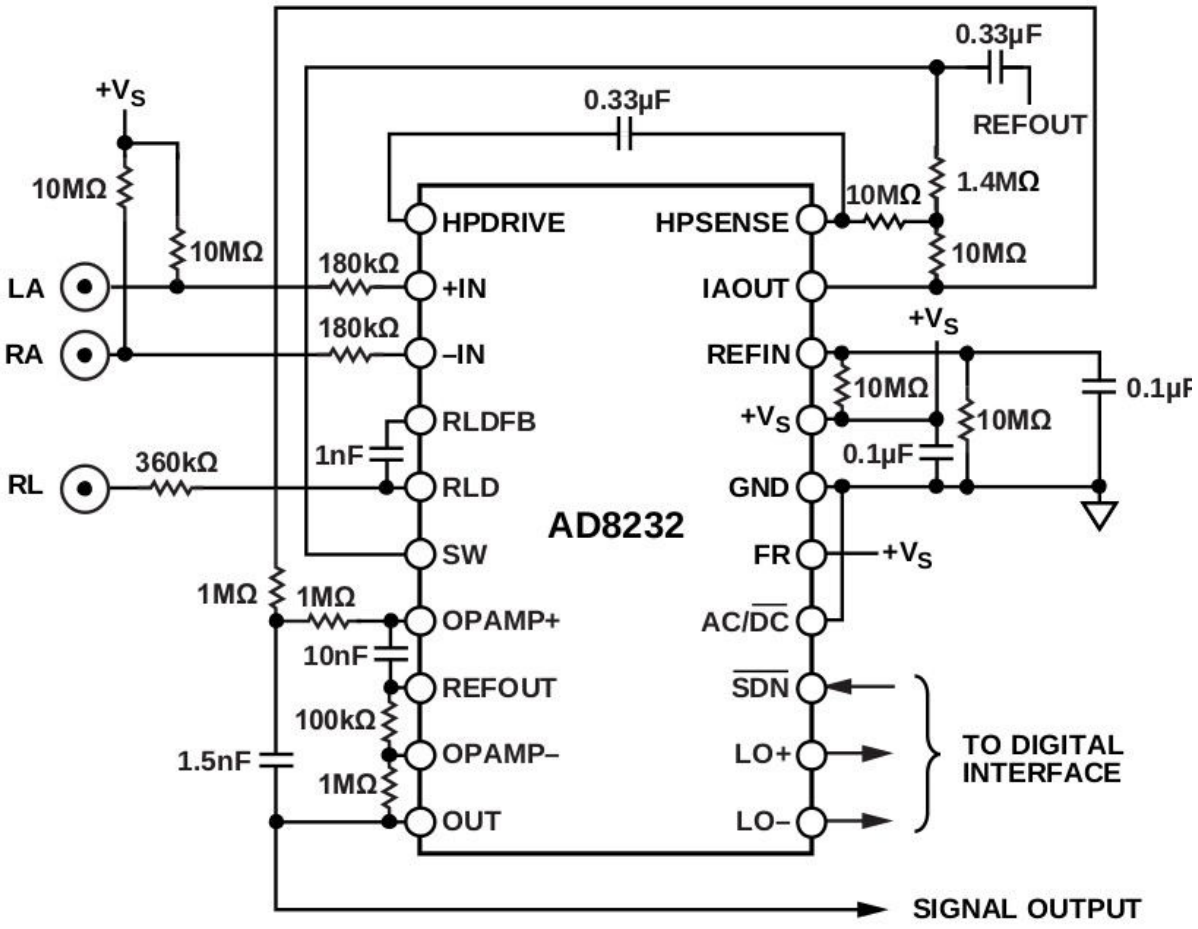


Рисунок 3.12. Конфігурація AD8232

3.2.3. Блок цифрування сигналу

У своїй роботі я вирішив інтегрувати систему на кристалі, що буде слугувати для отримання сигналу, його перетворення й передачі по інтерфейсах. Для цього проекту наступні компоненти повинні міститись у цій системі :

1. WiFi

2. Аналоговий-цифровий перетворювач

3. Багато ядерний процесор

Найбільш популярним і надійним варіантом такої системи є ESP32-WROOM-32, розроблена компанією EspressIf. [22]

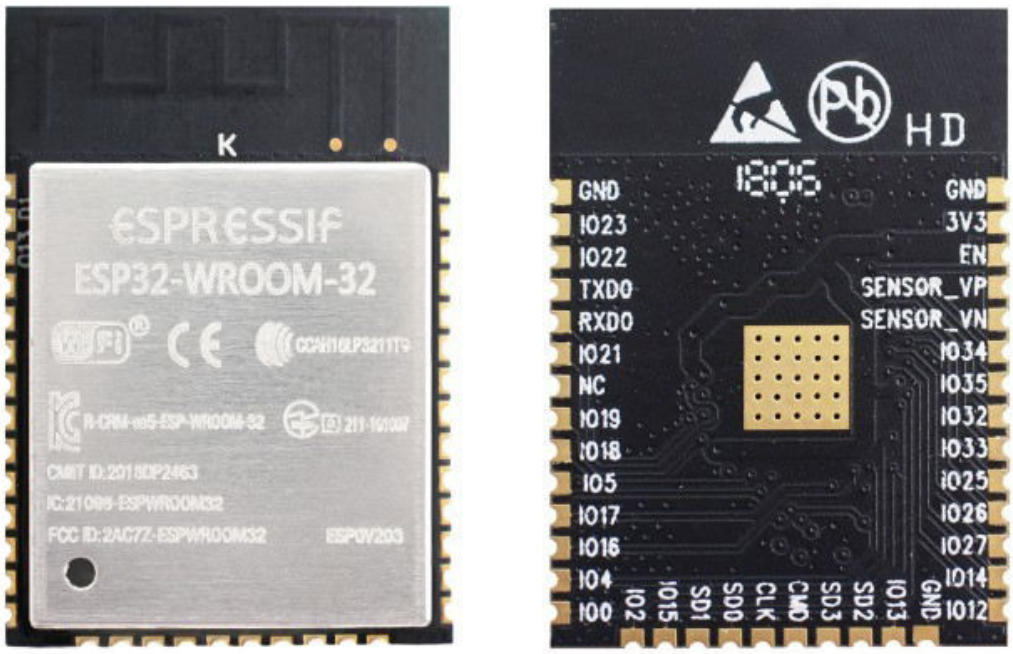


Рисунок 3.13. Система на кристалі ESP32-WROOM-32

Так-як сигнал, що надходить з AD8232 є аналоговим і він повинен бути збереженим у проміжне сховище для подальшої передачі, то він повинен бути представленим у цифровій формі. Для цього використовується аналоговий-цифровий перетворювач. Обрана система містить два 12-бітних аналого-цифрових перетворювачів. Проте, так-як другий АЦП використовується компонентом WiFi, то в нашому розпорядженні залишається тільки перший АЦП.

Тип цього АЦП — SAR (Successive Approximation ADC), принцип якого полягає в перетворенні вхідного аналогового сигналу в цифровий, використовуючи бінарний пошук по всіх рівнях квантування. [23]

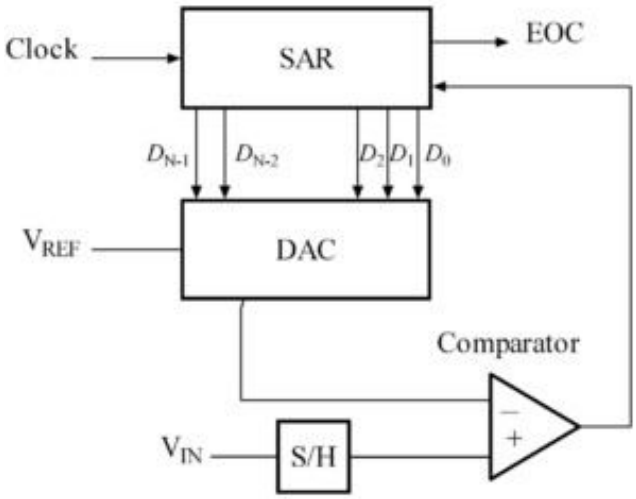


Рисунок 3.14. Функціональна діаграма АЦП SAR типу

3.2.4. Інтерфейсний блок Wi-Fi

Як раніше зазначено, Wi-Fi модуль є вбудованим у систему на кристалі ESP-32. Його характеристика наведена нижче.

Таблиця 3.1. Характеристика Wi-Fi модуля ESP32-WROVER-32

Параметр	Режим	Мін. Значення	Тип. значення	Макс. значення	Од. виміру
Частота	-	2412	-	2484	МГц
Вихідний імпеданс	-	-	50	-	Ом
Потужність TX	11n, MCS7	12	13	14	дБм
	11b mode	17.5	18.5	20	дБм
Чутливість	11b, 1 Mbps	-	–98	-	дБм
	11b, 11 Mbps	-	–89	-	дБм
	11g, 6 Mbps	-	-92	-	дБм
	11g, 54 Mbps	-	–74	-	дБм
	11n, HT20, MCS0	-	–91	-	дБм

Продовження Таблиця 3.1.

Селективність	11g, 6 Mbps	-	31	-	дБ
	11g, 54 Mbps	-	14	-	дБ
	11n, HT20, MCS0	-	31	-	дБ
	11n, HT20, MCS7	-	13	-	дБ

3.2.5. Блок передавання сигналу на комп'ютер по протоколу UART

UART у перекладі з англійської мови є універсальним асинхронним приймачем і передавачем. Кожний біт передається у чітко відведений проміжок часу. За замовчуванням розмір даних у пакеті, що передається рівний 8 бітам, але окрім цих даних пакет містить додатково службову інформацію. [24]

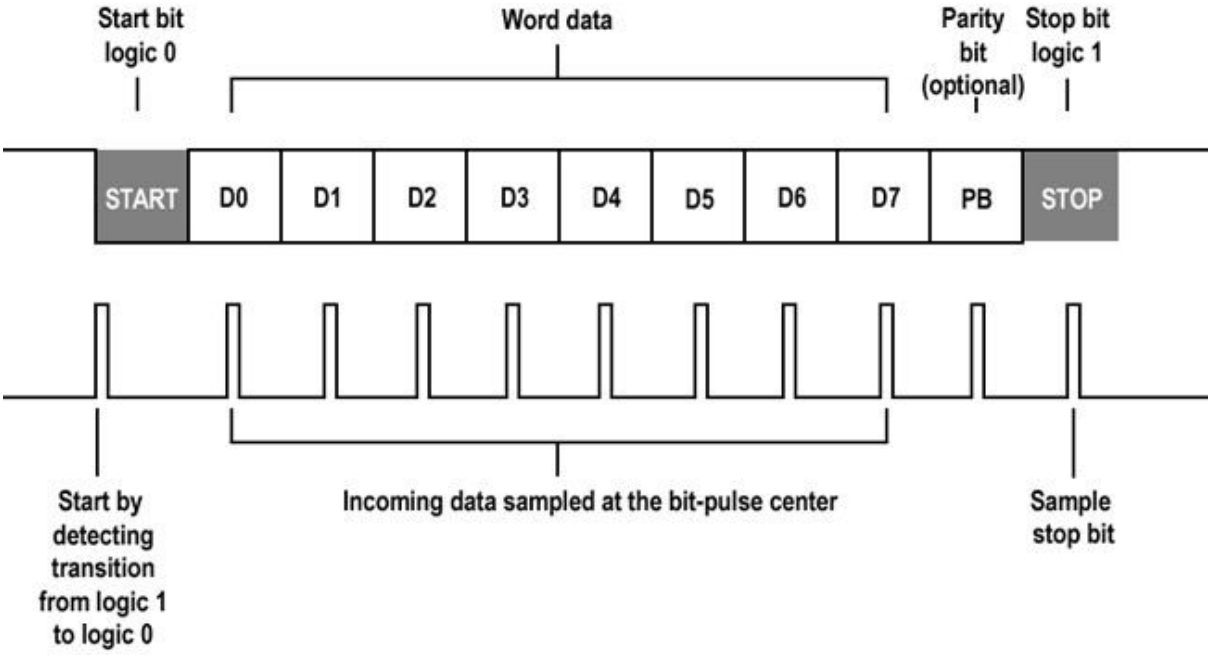


Рисунок 3.15. Візуалізація UART

А саме - стартовий біт, кінцевий біт, біт парності. Потрібно також зауважити, що оскільки вибрано асинхронний режим роботи, тому швидкість обробки даних повинна бути встановлена у передавача і приймача однаковою.

Таблиця 3.2. Часові характеристики UART

Швидкість передачі (бод)	Час передачі одного біта (мкс)	Час передачі одного байту (мкс)
4800	208	2083
9600	104	1042
19200	52	521
38400	26	260
57600	17	174
115200	8.7	87

UART може бути реалізованим як у напівдуплексному режимі так і в повнодуплексному режимі, тому що лінії приймання та передачі є розділеними. Лінія TXD є відповідальною за передачу, в той час як RXD — за приймання.

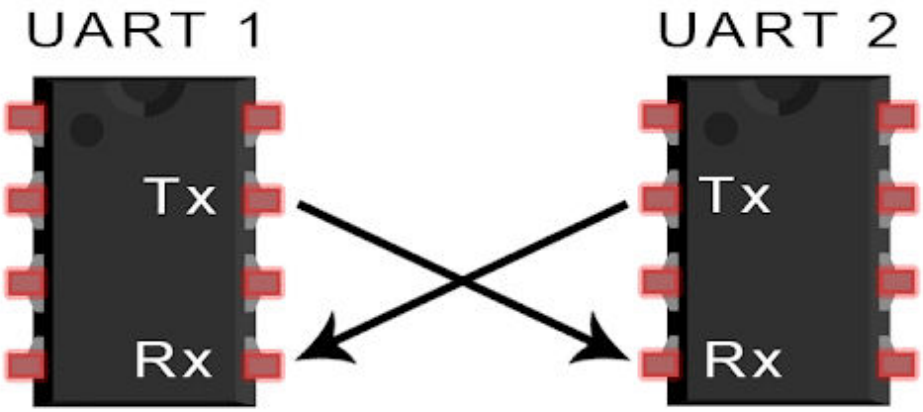


Рисунок 3.16. UART з’єднання

Як бачимо, цей інтерфейс чудово підійде для передачі ЕКГ сигналу з високою швидкістю і без втрати даних. Але, оскільки більшість сучасних комп'ютерів не дозволяють безпосередньо приєднати інший пристрій по цьому інтерфейсу, то для цього стане у нагоді конвертор USB-UART. Його призначення — перетворення сигналу зі стандарту USB у стандарт UART і навпаки.

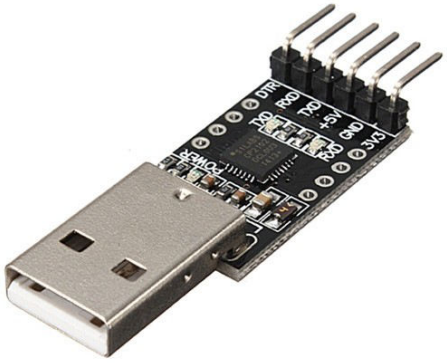


Рисунок 3.17. Вигляд USB-UART конвертора

Схематично це можна представити наступним чином.

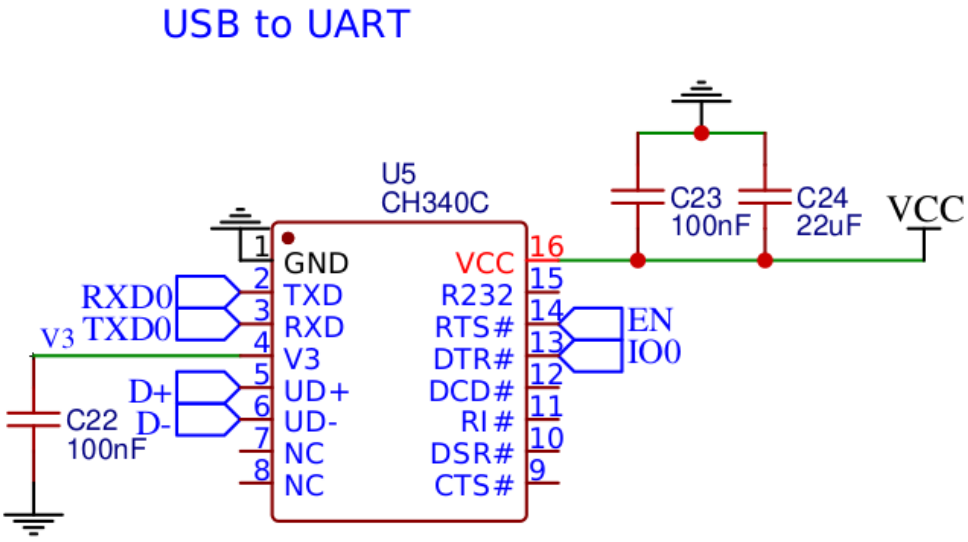


Рисунок 3.18. Схематичне представлення USB-UART конвертора

3.2.6. Блок візуалізації сигналу у браузері

Основною вимогою до блоку візуалізації сигналу у браузері є оновлення даних для візуалізації у режимі реального часу. Для цього сигнал будемо передавати по мережі, у нашому випадку мережі WiFi з використанням певного мережевого протоколу. У цьому випадку я вважаю потрібно дотриматись клієнт-серверної архітектури при виборі протоколу.

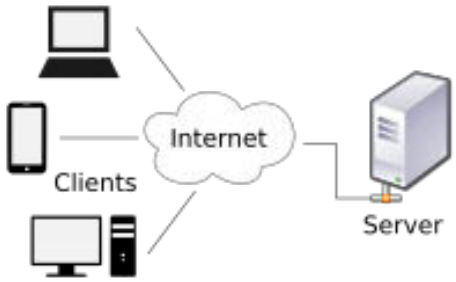


Рисунок 3.19. Діаграма клієнт-серверної архітектури

Сервер відповідальний за надання інформації або інших послуг клієнтам. Клієнти певним чином використовують сервіси надані сервером. Мережа встановлює взаємодію між сервером і клієнтом. [25]

Необхідно зауважити, що сервер працює з клієнтом без ідентифікації. Якщо запит, що надійшов на сервер від клієнта має правильну структуру згідно використовуваного протоколу — то сервер змушений надати відповідь.

При цьому ми отримуємо надзвичайну гнучкість — можна з легкістю припустимо реалізувати клієнтську програму для мобільного телефону або ж реалізувати клієнтську програму для комп’ютера, що певним чином аналізує отриманий сигнал.

Найкращим протоколом для нашого випадку є протокол WebSocket, що якраз призначений для обміну інформацією між клієнтом і сервером у режимі реального часу. Цей протокол реалізує двонаправлений повнодуплексний канал передачі через єдиний TCP-сокет. Окрім цього, цей протокол є повністю стандартизований і є підтримуваний усіма сучасними браузерами. [26]

Реалізувавши HTTP сервер на ESP32, я також розробив застосунок для браузера, що отримує сигнал і його динамічно відображає.

Рисунок 3.20. Вигляд браузерного застосунку

3.3. Розроблення алгоритмів і програм функціональних блоків

Як раніше було зазначено, в якості системи на кристалі я обрав ESP32. Компанія EspressIf для спрощення програмування з цією системою розробили фреймворк. Називається він ESP-IDF, і програмувати під нього потрібно мовою C або Python. Я вирішив обрати перший варіант, бо це дозволить досягти максимальної швидкодії а також програмний інтерфейс для мови C ширший і краще підтримується командою розробників. Як наведено вище, ESP32 містить двоядерний процесор. Тому є можливість запускати кілька потоків виконання паралельно, що можна легко здійснити з допомогою операційної системи FreeRTOS, на якій власне ESP-IDF побудований. Окрім можливості запуску потоків, FreeRTOS наділений функціоналом синхронізації, лічильниками, чергами й іншими компонентами для реалізації багатопотокової програми. [27]

Програма для цієї системи була розроблена згідно з алгоритмом, що наведений у Додатку 1. Хочу зауважити, що кожен крок алгоритму був реалізований виключно з допомогою вбудованих компонентів ESP-IDF, що

підкреслює достатню функціональність цієї платформи для реалізації вельми складних завдань.

Як можна побачити на діаграмі, програма на мікроконтролері виконується розподілено. Існує потік, що зчитує оцифрований сигнал з АЦП і записує в попередньо проініціалізовану матрицю розміром *N* на *M*. Додатково, при повному запису деякого рядка він передає індекс цього рядка в чергу повідомлень. Також існує потік, що зчитує значення з деякого рядка матриці й паралельно передає його “слухачам”. Слухачами в нашому випадку виступають мережеві клієнти і UART інтерфейс. Інформацію про те, який саме рядок матриці доцільно передавати надходить з черги повідомлень. Черга повідомлень є реалізованою у FreeRTOS як компоненту Queue. Перевага такого підходу полягає в тому, що потік, що зчитує сигнал, працює незалежно і має змогу робити виміри з АЦП значно швидше й точніше, що є ключовим для ЕКГ сигналу.

Щодо передачі сигналу по інтерфейсу UART то все досить просто. Єдиний параметр, що був сконфігурований це швидкість, значення якої я встановив 115 200 бод. Щодо передачі по мережі WebSocket, то все значно складніше. У програмі окремо виконується потік, що прослуховує 80 порт і при отриманні запиту записує його в чергу. Інший потік, зчитує з цієї черги запити і обробляє їх. Якщо запит задовольняє нашим вимогам, то ми додаємо нового клієнта у список, який у майбутньому отримуватиме сигнал. Це нагадує шаблон проектування Observer (Спостерігач). Програмний код апаратної частини проекту наведений у Додатку 2.

Щодо клієнтської програми для візуалізації сигналу в браузері, то вона була написана мовою програмування *JavaScript.* Додатково я використав бібліотеку *React.js,* розроблену компанією *Facebook* для побудови компонентних веб-застосунків, і бібліотеку для відображення графіків — *Recharts*.

3.4. Розроблення електричної принципової схеми

Електрична принципова схема, на відмінну від функціональної схеми, дає повне уявлення про принцип роботи пристрою та його структуру. Але на відмінну від друкованої плати, не показує взаємного розміщення елементів, але лише дає вказівку які з'єднання присутні.

Під час розробки електричної принципової схеми пристрою я скористався інструментом EasyEDA.

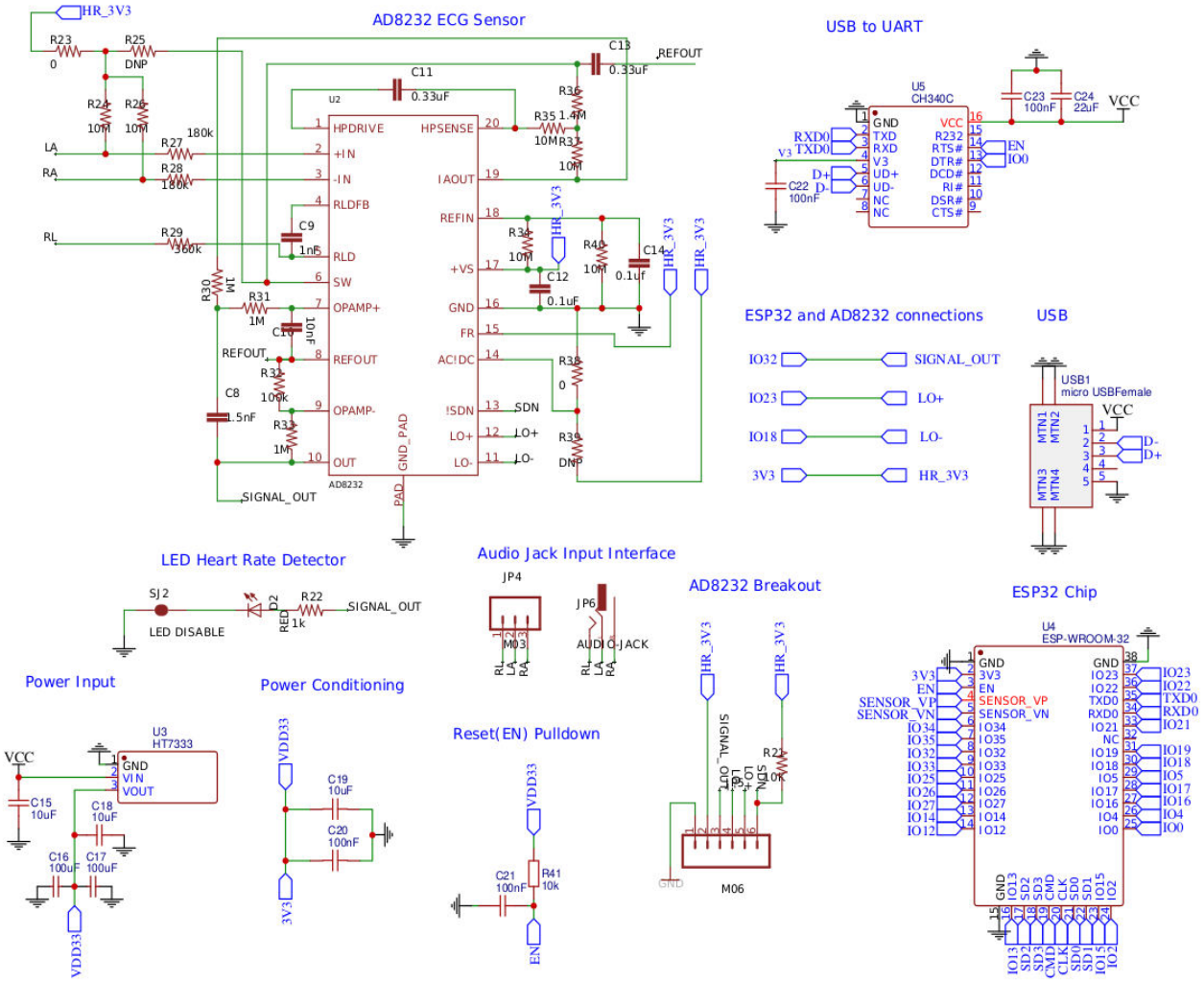


Рисунок 3.21. Принципова електрична схема ЕКГ пристрою

Висновки до розділу

У цьому розділі я реалізував схеми, що описують принцип роботи розробленого пристрою. Починаючи з структурної схеми, на якій було

наведено основні функціональні компоненти пристрою і закінчуючи принциповою електричною схемою, що найдетальніше описує його. Також у цьому розділі я описав роботу пристрою з програмної точки зору, навівши алгоритми і програмний код.

4. РОЗРОБЛЕННЯ І ТЕСТУВАННЯ АПАРАТНО-ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Вибір компонент для реалізації експериментального зразка та їх компонування

Для реалізації експериментального зразка, я вирішив використати так звану відладочну плату, що містить мікроконтролер, USB-UART адаптер, стабілізатор живлення тощо. Це дозволило спростити розробку експериментального зразка і його власне тестування.

Оскільки система на кристалі ESP-32 є вельми популярна, то й відповідно кількість відладочних плат, що на ній базуються є великою. Основними критеріями щодо вибору конкретної відладочної плати були:

1. Низька ціна
2. Наявність доступу до всіх виводів мікроконтролера
3. Компактність

Плата, що повністю задовольняє вищенаведені вимоги, виробляється компанією LilyGo і має назву TTGO T8 1.7. Окрім цього, плата містить вбудовану 3D антену, що значно покращує якість переданого сигналу, і слот для SD-картки.

Таблиця 4.1. Характеристики TTGO T8 1.7

Назва параметру	Значення параметру
Робоче значення струму	37 мА
Значення струму у режимі очікування	3 мА
Габарити	63.34 мм x 25.96 мм x 4.95 мм
Ціна	7 доларів

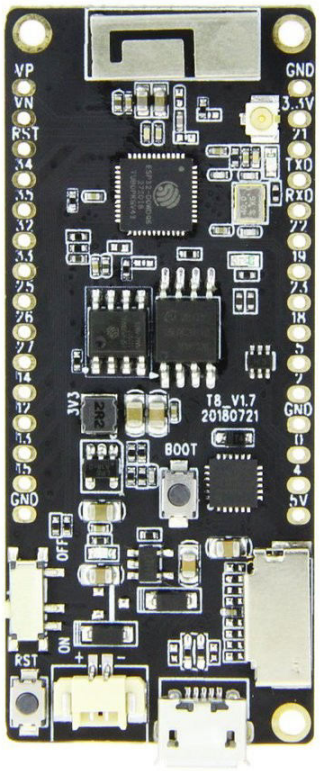


Рисунок 4.1. Вигляд TTGO T8 1.7

Як раніше зазначено, блок підсилення і фільтрування сигналу серця був реалізований з допомогою мікросхеми AD8232, у технічній документації до якої окрім параметрів і характеристик самої схеми додатково містились різні конфігурації. Я обрав конфігурацію, що дозволяє при цілковитому знерухомленні людини найточніше відтворити сигнал. Як виявилось, саме цю конфігурацію компанія SparkFun застосувала при розробці плати SparkFun AD8232. Використання цієї плати спростило мені розробку експериментального зразка.

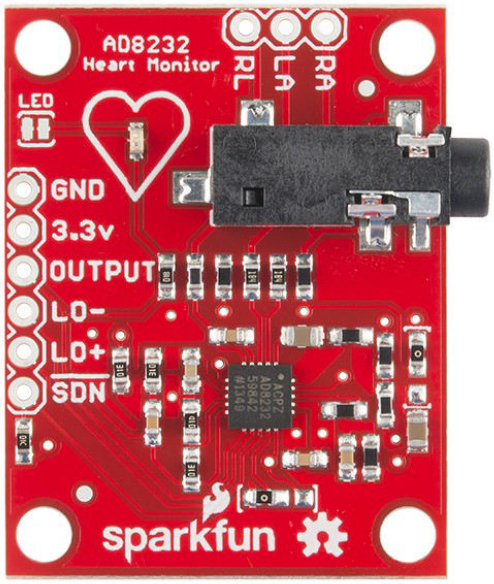


Рисунок 4.2. Плата SparkFun AD8232

Нижче наведено з’єднання двох вищенаведених компонентів.

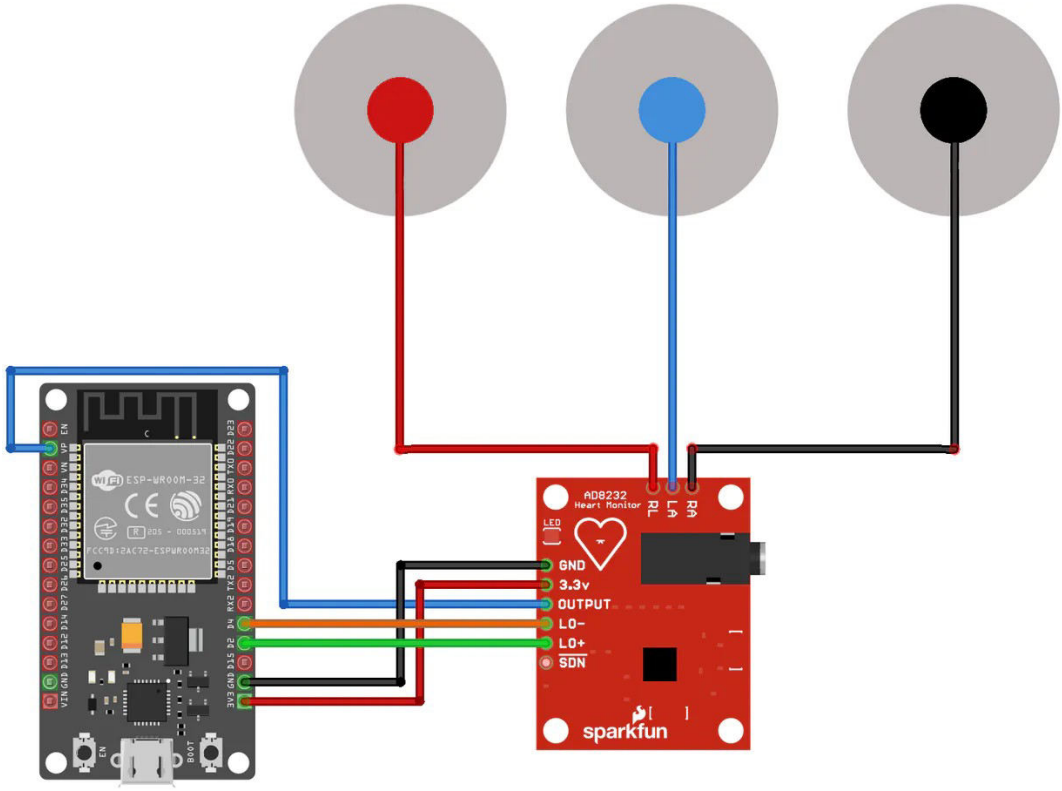


Рисунок 4.3. З’єднання компонентів

4.2. Розробка друкованої плати

Друкована плата це пластина, що зазвичай виробляється з діалектика і містить хоча б один шар провідних доріжок. Модель друкованої плати я розробив у середовищі EasyEDA. Вона була розроблена на основі наведеної принципової електричної схеми. Трасування доріжок виконано автоматично з допомогою застосунку AutoRouter.

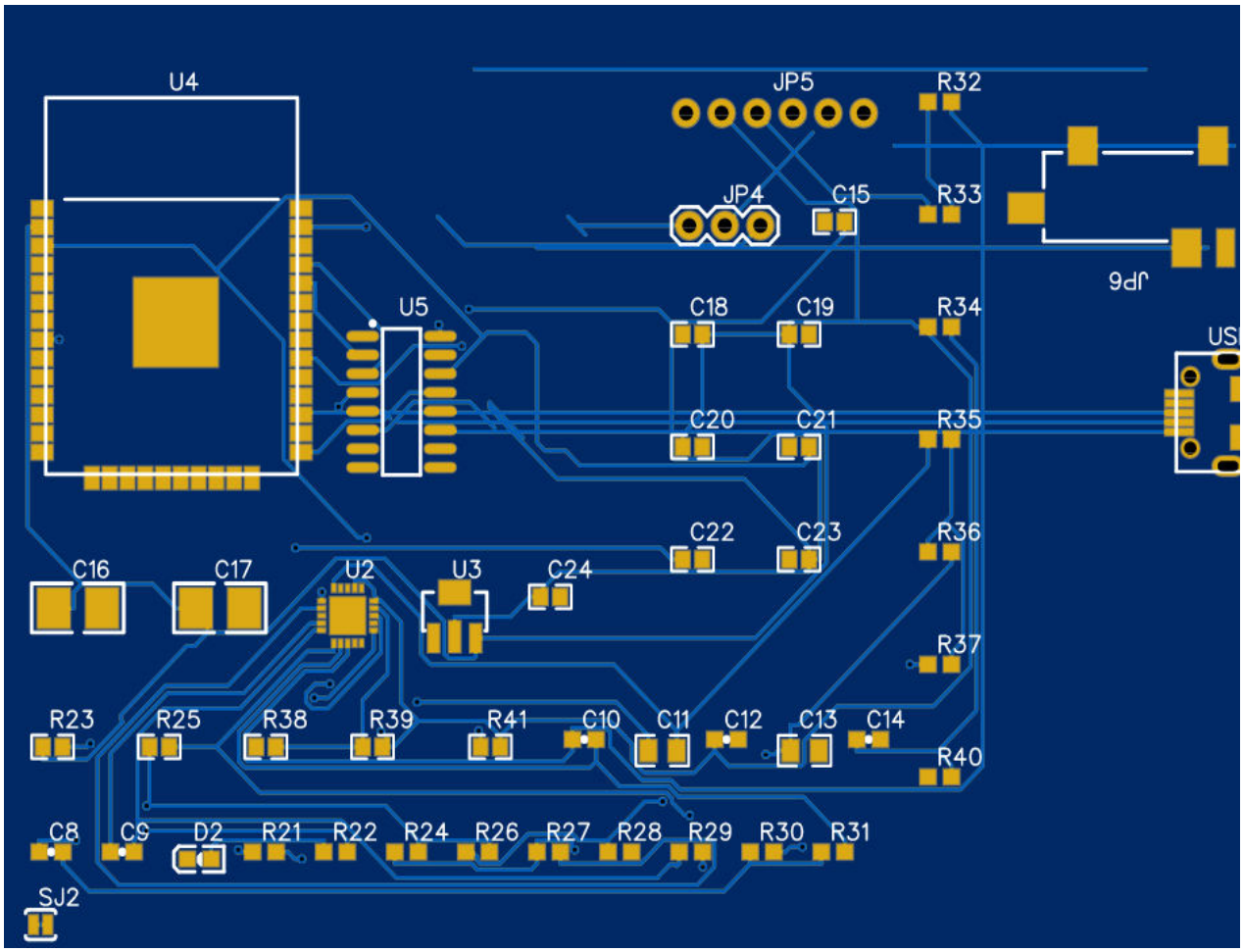


Рисунок 4.4. Друкована плата

4.3. Результати тестування розробленого експериментального зразка

Встановивши розроблене програмне забезпечення на систему на кристалі ESP-32, я розпочав тестування виконаного експериментального зразка. Тестував роботу пристрою я передаючи сигнал на комп'ютер через протокол UART. При цьому на комп'ютері був встановлений програмний пакет MATLAB, що містить вбудовані можливості як візуалізації сигналу в режимі реального часу, так і його отримання з допомогою пакету *serial*.

Під час проведення експерименту я окрім візуалізації сигналу додатково безпосередньо записував отриманні значення у файл. Після цього здійснив додаткове опрацювання отриманих значень. А саме — застосував цифровий ежекторний фільтр у цьому ж середовищі. Нижче наведено отриманий результат до застосування фільтру і, відповідно, після.

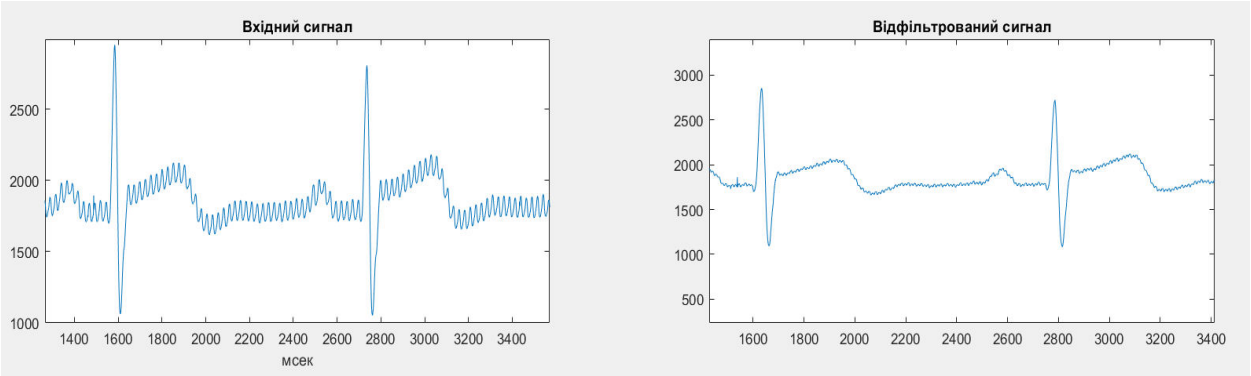


Рисунок 4.5. Результат експерименту

Висновки до розділу

У цьому розділі я описав компоненти, що були використані для виготовлення тестового зразка. Також, я презентував друковану плату і навів результат тестування виконаного зразка.

5. ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ

Загальна вартість розробленого пристрою сягає 11.77 доларів. У ціну входить плата для розробки ESP-32-WROVER вартістю 7 доларів і аналоговий інтерфейс SparkFun AD8232 ціною 4.77 доларів. Вартістю провідників і вартістю електродів можна знехтувати. На ринку фігурують схожі пристрої, починаючи з 3000 гривень. При цьому, деякі варіанти бракують функціоналом передачі сигналу безпосередньо на комп'ютер, деякі не містять вбудованої системи фільтрів тощо.

Також варто зазначити, що хоча для цього проекту я розробив браузерний застосунок для візуалізації сигналу в режимі реального часу, проте за рахунок дотримання клієнт-серверної архітектури можна легко реалізувати власний застосунок під будь-яку платформу будь-якого характеру, що отримує сигнал по достатньо простому протоколу WebSocket. Єдиною вимогою до реалізації є наявність WiFi інтерфейсу на пристрої, що власне виконує цей застосунок.

Я вважаю, що розроблена система є економічно вигідною і ретельно протестувавши, її можна використовувати не тільки у навчальних цілях, але й в комерційних.

ВИСНОВКИ

У цій роботі описано розроблений проект - систему оперативного детектування електричних сигналів серця.

Розглянуто існуючі засоби діагностування серця, підкреслюючи їх переваги та недоліки. Здійснено опис перетворення Фур'є, цифрових та аналогових фільтрів і їх застосування для покращення якості ЕКГ сигналу на прикладі програмної реалізації мовою Python.

Представлено опис розробленої системи, навівши структурну, функціональну та електричну принципову схеми. Детально описано та обґрунтовано вибір кожного компоненту та кожне інженерне рішення в даній системі.

Розроблено блок-схему алгоритму роботи пристрою та наведено відповідний програмний код мовою програмування C. Додатково реалізовано клієнтський браузерний застосунок мовою Javascript, що відображає отриманий сигнал з мікроконтролера у режимі реального часу.

Наведено компоненти, що були використані для реалізації тестового взірця та результат його тестування. Проаналізовано економічну доцільність виготовлення цього пристрою.

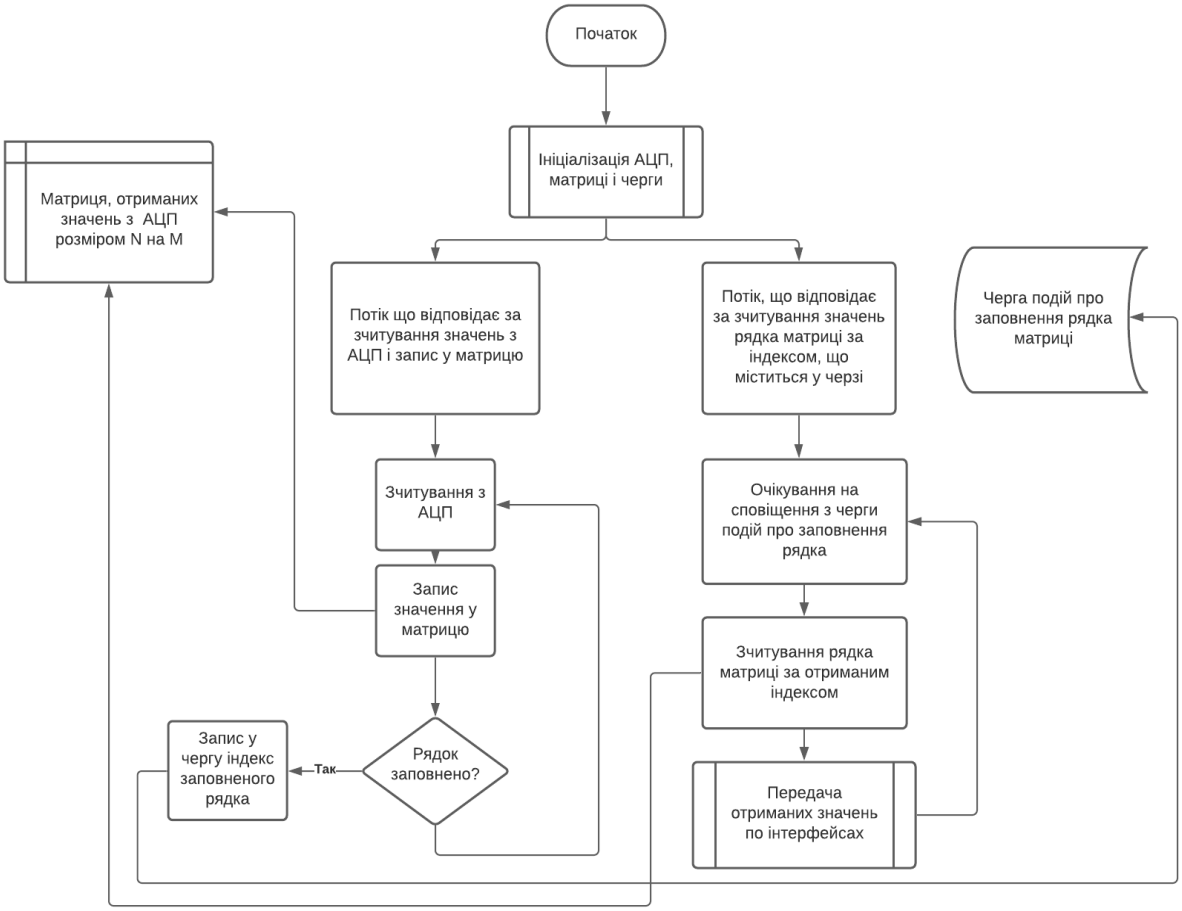
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Graham Jackson. Heart Health. Class Publishing - 2009
2. Antoni Bayés de Luna. Basic Electrocardiography: Normal and Abnormal ECG Patterns – 2007
3. Основи практичної електрокардіографії (навчальний посібник) / Швед М. І., Гребеник М. В. — Тернопіль, Укрмедкнига. 2000. — 128 с.
4. Клінічна електрокардіографія: навч. посібник / Люлька Н. О., Скрипник І. М., Потяженко М. М., Шкляренко В. М., Дубровінська Т. В.; Мін.освіти і науки України, Мін. охорони здоров'я України, ВДНЗУ «УМСА». — Полтава: ТОВ "Фірма «Техсервіс», 2009. — 152 с.
5. Steven W. Smith, The Scientist and Engineer’s Guide to Digital Signal Processing, Second Edition, 1999, California Technical Publishing
6. S. Haykin, Adaptive Filter Theory, 3rd Edition, Prentice-Hall, 1996.
7. Brian W. Kernighan and Dennis M. Ritchie: The C Programming Language, 2nd ed., Prentice Hall, 1988
8. Колонтаєвський Ю. П., Сосков А. Г. Електроніка і мікросхемотехніка : Підручник. 2-е вид. – 2009.
9. Буров Є. В. Комп'ютерні мережі: підручник / Євген Вікторович Буров. — Львів: «Магнолія 2006», 2010. — 262 с.
10. Eloquent JavaScript; 3rd Ed; Marijn Haverbeke; No Starch Press; 472 pages; 2018
11. Віллем Ейнтховен [Електронний ресурс] :
    Режим доступу: https://uk.wikipedia.org/wiki/Віллем_Ейнтховен
12. Електрокардіограма [Електронний ресурс] :
    Режим доступу: https://uk.wikipedia.org/wiki/ЕКГ
13. Методи дослідження серцево-судинної системи [Електронний ресурс] :
    Режим доступу: http://repository.ldufk.edu.ua:8080/bitstream/
14. Аритмії [Електронний ресурс] :
    Режим доступу: https://uk.wikipedia.org/wiki/Аритмії_серця

15. ST Segment [Електронний ресурс] :
Режим доступу: https://en.wikipedia.org/wiki/ST_segment
16. Документація до AD8232 [Електронний ресурс]:
Режим доступу: https://www.analog.com/ru/products/ad8232.html#product-documentation
17. ECG Artifacts [Електронний ресурс]:
Режим доступу: https://www.aclsmedicaltraining.com/blog/guide-to-understanding-ecg-artifact/
18. Radio Spectrum [Електронний ресурс]:
Режим доступу: https://en.wikipedia.org/wiki/Radio_spectrum
19. Fourier Transform [Електронний ресурс]:
Режим доступу: https://www.thefouriertransform.com
20. Introduction to Digital Filters [Електронний ресурс]:
Режим доступу: https://ccrma.stanford.edu/~jos/filters/
21. Block Diagram [Електронний ресурс] :
Режим доступу: https://en.wikipedia.org/wiki/Block_diagram
22. TRS Connector [Електронний ресурс] :
Режим доступу: https://en.wikipedia.org/wiki/Phone_connector_(audio)
23. ESP32 [Електронний ресурс] :
Режим доступу: https://en.wikipedia.org/wiki/ESP32
24. Successive ApproximationADC [Електронний ресурс]:
Режим доступу: https://en.wikipedia.org/wiki/Successive-approximation_ADC
25. UART [Електронний ресурс]:
Режим доступу: https://uk.wikipedia.org/wiki/UART
26. Client-Server Model [Електронний ресурс]:
Режим доступу: https://en.wikipedia.org/wiki/Clien%E2%80%93server_model
27. WebSocket [Електронний ресурс]:
Режим доступу: https://uk.wikipedia.org/wiki/WebSocket
28. ESP-IDF Programming Guide [Електронний ресурс]:
Режим доступу: https://docs.espressif.com/projects/esp-idf/en/latest/esp32

ДОДАТОК 1

БЛОК-СХЕМА АЛГОРИТМУ РОБОТИ СИСТЕМИ



ДОДАТОК 2

ПРОГРАМНИЙ КОД ДЛЯ МІКРОКОНТРОЛЕРА

```
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "lwip/err.h"
#include "lwip/sys.h"
#include "driver/adc.h"
#include "driver/i2s.h"
#include "esp_adc_cal.h"
#include "soc/syscon_reg.h"
#include "soc/syscon_struct.h"
#include "websocket_server.h"
#include "esp32/clk.h"
#include "hal/adc_hal.h"
#include "esp_timer.h"
#include "esp_event_loop.h"

#define ESP_WIFI_SSID      CONFIG_ESP_WIFI_SSID
#define ESP_WIFI_PASS      CONFIG_ESP_WIFI_PASSWORD
#define ESP_WIFI_CHANNEL   CONFIG_ESP_WIFI_CHANNEL
#define MAX_STA_CONN       CONFIG_ESP_MAX_STA_CONN

static const char *TAG = "ECG_MONITOR";

// Параметри АЦП

#define ADC_CHANNEL ADC1_CHANNEL_4
#define ADC_UNIT ADC_UNIT_1

// Черга подій про заповнення буфера АЦП

static QueueHandle_t ADC_FILL_QUEUE;

// Черга WebSocket з'єднань

static QueueHandle_t WS_QUEUE;
```

```
// Параметри буфера

#define BUFFER_ROWS 8
#define BUFFER_COLS 1024

// Буфер, в який записуються і зчитуються дані з АЦП

static int BUFFERS [BUFFER_ROWS][BUFFER_COLS];

// Індекс останнього заповненого буфера

static int lastFilledBuffer;

// Функція-обробник подій пов'язаних WiFi
static esp_err_t event_handler(void* ctx, system_event_t* event) {
  const char* TAG = "EVENT_HANDLER";

  switch(event->event_id) {
    case SYSTEM_EVENT_AP_START:
      ESP_LOGI(TAG,"Access Point Started");
      break;
    case SYSTEM_EVENT_AP_STOP:
      ESP_LOGI(TAG,"Access Point Stopped");
      break;
    case SYSTEM_EVENT_AP_STACONNECTED:
      ESP_LOGI(TAG,"STA Connected, MAC=%02x:%02x:%02x:%02x:%02x:%02x AID=%i",
        event->event_info.sta_connected.mac[0],event->event_info.sta_connected.mac[1],
        event->event_info.sta_connected.mac[2],event->event_info.sta_connected.mac[3],
        event->event_info.sta_connected.mac[4],event->event_info.sta_connected.mac[5],
        event->event_info.sta_connected.aid);
      break;
    case SYSTEM_EVENT_AP_STADISCONNECTED:
      ESP_LOGI(TAG,"STA Disconnected, MAC=%02x:%02x:%02x:%02x:%02x:%02x AID=
%i",
        event->event_info.sta_disconnected.mac[0],event->event_info.sta_disconnected.mac[1],
        event->event_info.sta_disconnected.mac[2],event->event_info.sta_disconnected.mac[3],
        event->event_info.sta_disconnected.mac[4],event->event_info.sta_disconnected.mac[5],
        event->event_info.sta_disconnected.aid);
      break;
    case SYSTEM_EVENT_AP_PROBEREQRECVED:
      ESP_LOGI(TAG,"AP Probe Received");
      break;
    case SYSTEM_EVENT_AP_STA_GOT_IP6:
      ESP_LOGI(TAG,"Got IP6=%01x:%01x:%01x:%01x",
```

```
        event->event_info.got_ip6.ip6_info.ip.addr[0],event-
>event_info.got_ip6.ip6_info.ip.addr[1],
        event->event_info.got_ip6.ip6_info.ip.addr[2],event-
>event_info.got_ip6.ip6_info.ip.addr[3]);
      break;
    default:
      ESP_LOGI(TAG,"Unpredictable event=%i",event->event_id);
      break;
  }
  return ESP_OK;
}

// Ініціалізація TCP
static void tcp_init(void) {
    static const char * TAG = "TCP_INIT";

    ESP_LOGI(TAG,"Starting TCP/IP Adapter");
    tcpip_adapter_init();

    nvs_flash_init();
    ESP_ERROR_CHECK(tcpip_adapter_dhcps_stop(TCPIP_ADAPTER_IF_AP));

    // Параметри TCP адаптера
    tcpip_adapter_ip_info_t info;
    memset(&info, 0, sizeof(info));
    IP4_ADDR(&info.ip, 192, 168, 4, 1);
    IP4_ADDR(&info.gw, 192, 168, 4, 1);
    IP4_ADDR(&info.netmask, 255, 255, 255, 0);

    ESP_LOGI(TAG,"Setting getaway IP");
    ESP_ERROR_CHECK(tcpip_adapter_set_ip_info(TCPIP_ADAPTER_IF_AP, &info));

    ESP_LOGI(TAG,"Starting DHCPS Adapter");
    ESP_ERROR_CHECK(tcpip_adapter_dhcps_start(TCPIP_ADAPTER_IF_AP));

    ESP_LOGI(TAG,"Starting Event loop");
    ESP_ERROR_CHECK(esp_event_loop_init(event_handler, NULL));
}


// Функція запуску точки доступу WiFi
static void wifi_setup(void) {
    const char* TAG = "WIFI_SETUP";

    ESP_LOGI(TAG,"Initializing WIFI");
```

```
    wifi_init_config_t wifi_init_config = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&wifi_init_config));
    ESP_ERROR_CHECK(esp_wifi_set_storage(WIFI_STORAGE_RAM));
    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_AP));

    wifi_config_t wifi_config = {
        .ap = {
            .ssid = ESP_WIFI_SSID,
            .ssid_len = strlen(ESP_WIFI_SSID),
            .channel = ESP_WIFI_CHANNEL,
            .password = ESP_WIFI_PASS,
            .max_connection = MAX_STA_CONN,
            .authmode = WIFI_AUTH_WPA_WPA2_PSK
        },
    };
    if (strlen(ESP_WIFI_PASS) == 0) {
        wifi_config.ap.authmode = WIFI_AUTH_OPEN;
    }

    ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_AP, &wifi_config));
    ESP_ERROR_CHECK(esp_wifi_start());

    ESP_LOGI (TAG,
            "Initialization of Wifi Soft AP finished. SSID:%s password:%s channel:%d",
            ESP_WIFI_SSID,
            ESP_WIFI_PASS,
            ESP_WIFI_CHANNEL
    );
}

// Обробник подій пов'язаних з вебсокетами
static void ws_event_handler(uint8_t num, WEBSOCKET_TYPE_t type, char * msg, uint64_t
size) {
    const char * WS_TAG = "WEBCOCKET_SERVER";
    if (type == WEBSOCKET_CONNECT) {
        ESP_LOGI(WS_TAG, "Websocket client %i connected.", num);
    }
    else if (type == WEBSOCKET_DISCONNECT_EXTERNAL) {
        ESP_LOGI(WS_TAG, "Websocket client %i has disconnected.", num);
    }
    else if (type == WEBSOCKET_DISCONNECT_INTERNAL) {
        ESP_LOGI(WS_TAG, "Websocket client %i has been disconnected.", num);
    }
    else if (type == WEBSOCKET_DISCONNECT_ERROR) {
        ESP_LOGI(WS_TAG, "Websocket client %i was disconnected because of an error", num);
```

```
    }
}

// Функція-обробник HTTP запитів на сервер
static void http_handler(struct netconn * connection) {
    const char * HTTP_TAG = "HTTP_HANDLER";

    struct netbuf * input_buffer;
    static char * buffer;
    static uint16_t buffer_len;

    // Час затримки запиту - 1 секунда
    netconn_set_recvtimeout(connection, 1000);

    if (netconn_recv(connection, &input_buffer) == ESP_OK) {

        // Читання вмісту запиту в буфер
        netbuf_data(input_buffer, (void **) &buffer, &buffer_len);

        // Якщо тип вхідного запиту GET, то встановлюємо TCP/IP з'єднання
        if (strstr(buffer, "GET / ") && strstr(buffer, "Upgrade: websocket")) {
            ESP_LOGI(TAG, "Request on websocket endpoint");
            ws_server_add_client(connection, buffer, buffer_len, "/", ws_event_handler);
            netbuf_delete(input_buffer);
        }

        // В іншому випадку закриваємо з'єданння з користувачем
        else {
            ESP_LOGI(TAG, "Unknown request");
            netconn_close(connection);
            netconn_delete(connection);
            netbuf_delete(input_buffer);
        }

    }
    else {
        ESP_LOGI(TAG, "Error on reading request, closing connection");
        netconn_close(connection);
        netconn_delete(connection);
        netbuf_delete(input_buffer);
    }
}

// Потік сервера
```

```
static void server_task(void * arg) {
    const static char * SERVER_TAG = "SERVER_TASK";
    struct netconn * new_connection;

    const int WS_QUEUE_SIZE = 15, PORT = 80;

    // Ініціалізація черги вебсокет з'єднань
    WS_QUEUE = xQueueCreate(WS_QUEUE_SIZE, sizeof(struct netconn *));

    // Ініціалізація з'єднання TCP сервера
    struct netconn * connection = netconn_new(NETCONN_TCP);
    netconn_bind(connection, NULL, PORT);
    netconn_listen(connection);

    ESP_LOGI(SERVER_TAG, "Server is listening on PORT #%d", PORT);

    // Сервер прослуховує 80 порт, і при запиті делегує обробку запиту в окремому потоці
    while (netconn_accept(connection, &new_connection) == ESP_OK) {
        xQueueSendToBack(WS_QUEUE, &new_connection, portMAX_DELAY);
    }

    // При завершенні завдання відбувається очистка ресурсів і перезавантаження плати

    netconn_close(connection);
    netconn_delete(connection);

    ESP_LOGI(SERVER_TAG, "Task is ending, rebooting...");
    esp_restart();
}

// Потік обробки HTTP запитів
static void server_handle_task(void * arg) {
    const static char * SERVER_HANDLE_TAG = "SERVER_HANDLER";
    struct netconn * connection_to_process;
    while (true) {
        // Отримання з'єднання від іншого потоку
        xQueueReceive(WS_QUEUE, &connection_to_process, portMAX_DELAY);
        if (!connection_to_process) {
            continue;
        }
        // Його обробка
        http_handler(connection_to_process);
    }
    vTaskDelete(NULL);
}
```

```
// Ініціалізація АЦП в RTC режимі
// Attenuation встановлено значення 11 .Діапазон вхідних напруг 0 - 3.3 В
// Розрядність АЦП 12 біт. Діапазон значень: 0 - 4095
static void adc_rtc_init() {

   adc1_config_channel_atten(ADC_CHANNEL, ADC_ATTEN_DB_11);
   adc1_config_width(ADC_WIDTH_BIT_12);
}

// Функція ініціалізації черги подій про заповнення буфера АЦП
static void adc_filling_queue_init(void) {
   const int QUEUE_SIZE = 100;
   ADC_FILL_QUEUE = xQueueCreate(QUEUE_SIZE, sizeof(int));
   ESP_ERROR_CHECK( (ADC_FILL_QUEUE == NULL) ? ESP_ERR_NO_MEM : ESP_OK
);
}

// Функція, що зчитує сигнал з АЦП і записує значення в буфер.
static void adc_reading_task(void * arg) {

   // Затримка 10 мілісекунд між вимірюваннями
   const TickType_t DELAY_TICKS = 10 / portTICK_PERIOD_MS;

   while (true) {
      for (int i = 0; i < BUFFER_ROWS; i++) {

         for (int j = 0; j < BUFFER_COLS; j++) {

               // Запис значеня в буфер
            BUFFERS[i][j] = adc1_get_raw(ADC_CHANNEL);

            // Затримка
            vTaskDelay(DELAY_TICKS);
         }
         lastFilledBuffer = i;

         // Надсилання сповіщення іншій задачі про завершення запису в деякий буфер
         xQueueGenericSend(ADC_FILL_QUEUE, (void *) &lastFilledBuffer, (TickType_t) 0,
queueSEND_TO_BACK);
      }
   }
}

// Функція, що зчитує відповідний буфер за сповіщенням і передає значення по
```

```
інтерфейсах
static void trasmit_task(void * arg) {

        // Буфер для передавання
    char send_buffer[20];

    // Довжина повідомлення
    int len;

    // Формат цілочисельного числа
    const static char * template = "%d";

    // Індекс буфера
    int bufferIndex;

    while (true) {

        // Очікування на сповіщення
        if (xQueueGenericReceive(ADC_FILL_QUEUE, &bufferIndex, 10 /
portTICK_PERIOD_MS, false) == pdPASS) {
            for (int i = 0; i < BUFFER_COLS; i++) {

                    // Перетворення числа в текстове представлення
                len = sprintf(send_buffer, template, BUFFERS[bufferIndex][i]);

                // Надсилання значення всім користувачам по мережі WebSockets
                ws_server_send_text_all(send_buffer, len);

                // Передавання значення на UART
                printf("%d\n", BUFFERS[bufferIndex][i]);
            }
        }
    }
}


esp_err_t app_main(void)
{
ESP_LOGI(TAG, "Initializing TCP");
tcp_init();

ESP_LOGI(TAG, "Initializing WiFi Access Point");
wifi_setup();

 ESP_LOGI(TAG, "Initializing WebSocket Server");
```

```
    ws_server_start();

    ESP_LOGI(TAG, "Initializing ADC in RTC mode");
    adc_rtc_init();

    ESP_LOGI(TAG, "Initializing ADC Filling Queue");
    adc_filling_queue_init();

    ESP_LOGI(TAG, "Creating Server Tasks");
    xTaskCreatePinnedToCore(server_task, "Server task", 5000, NULL, 5, NULL, 0);
    xTaskCreatePinnedToCore(server_handle_task, "Server handler", 5000, NULL, 7, NULL, 0);

    ESP_LOGI(TAG, "Creating ADC reading && transmiting task");
    xTaskCreatePinnedToCore(adc_reading_task, "ADC reading", 5000, NULL, 10, NULL, 1);
    xTaskCreatePinnedToCore(trasmit_task, "Transmiting task", 5000, NULL, 8, NULL, 0);

    return ESP_OK;
}
```