Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет радіоелектроніки та телекомунікацій
(повне найменування інституту, факультету)

Кафедра інформаційних технологій електронних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему Створення низькочастотного навчального осцилографа на базі Arduino UNO R3 та персонального комп'ютера

Виконав: студент(ка) 4 курсу, групи РТз-117

Спеціальності 172 «Телекомунікації та радіотехніка»
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Радіоелектронні апарати та засоби»

Асютін В.О.
(прізвище та ініціали)

Керівник Оніщенко В.Ф.
(прізвище та ініціали)

Рецензент Мороз Г. В.
(прізвище та ініціали)

2021

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет __Інститут інформатики та радіоелектроніки, Факультет радіоелектроніки та телекомунікацій_____
Кафедра__ Інформаційних технологій електронних засобів_____
Ступінь вищої освіти __бакалавр_____
Спеціальність _172 «Телекомунікації та радіотехніка»_____
(код і найменування)
Освітня програма (спеціалізація)_ Радіоелектронні апарати та засоби
_____
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТЕЗ
_____ к.т.н. Огренич Є.В.
«27» 05 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Асютін Владислав Олегович
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи)__Створення низькочастотного навчального осцилографа на базі Arduino UNO R3 та персонального комп'ютера ____
_____
керівник проєкту (роботи)__Оніщенко Вадим Федорович, к.ф.-м.н._____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «26» квітня 2021 року №161
2. Строк подання студентом проєкту (роботи)_7 червня 2021 року_____
3. Вихідні дані до проєкту (роботи)__плата Arduino UNO, частотний діапазон від 0 до 3000Гц _____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)__Огляд області розробки, постановка задач кваліфікаційної роботи, розробка апаратної частини осцилографа, написання програмного забезпечення для компютера
_____
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
__11 рисунків, 2 таблиць, 18 слайдів
_____

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1 - 3	Оніщенко В.Ф., доцент		
Нормоконтроль	Поспеєва І.Є., ст. викладач		

7. Дата видачі завдання «26» квітня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд принципів побудови осцилографів	26.04.21	
2	Постановка технічного завдання	28.04.21	
3	Розробка програми Arduino	29.04.21	
4	Розробка протоколу обміну між Arduino та ПК	03.05.21	
5	Написання тексту програмного забезпечення для ПК	29.05.21	
6	Оформлення ПЗ та захист дипломного проекту	01.06.21	

Студент(ка) _____ Асютін В.О.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Оніщенко В.Ф..
(підпис) (прізвище та ініціали)

РЕФЕРАТ

ПЗ: 40 с., 11 рис., 2 табл., 14 джерел, 18 слайдів.

Об'єкт розробки: осцилограф.

Предмет розробки: електронний осцилограф на основі плати Arduino UNO та персонального компютера.

Мета роботи: розробка програмного забезпечення для плати Arduino UNO та програми для персонального компютера для реалізації осцилографа.

Для здійснення поставленої мети необхідно було вирішити наступні задачі:

- ознайомитись з застосуванням осцилографів та їх принципом дії;
- розглянути апаратне забезпечення для створення осцилографу;
- розробити протокол обміну даними між платою Arduino UNO та компютером;
- написати програму керування мікроконтролером;
- написати програму відображення осцилограм для персонального комп’ютера.

ОСЦИЛОГРАФ, АЦП, ДЖЕРЕЛО ОПОРНОЇ НАПРУГИ, ШАГ ДИСКРЕТИЗАЦІЇ, ARDUINO UNO, МІКРОКОНТРОЛЕР, ПЛАТА, ПРОГРАМА, ПРОТОКОЛ ОБМІНУ.

ЗМІСТ

ВСТУП

Осцилограф до ПК - це пристрій, який дозволяє графічно спостерігати електричний сигнал.

У наш час використання різних вимірювальних пристроїв, побудованих на базі взаємодії з персональним комп'ютером, досить багато. Значною перевагою їх використання є можливість збереження отриманих значень досить великого обсягу в пам'яті пристрою, з подальшим їх аналізом.

Комп'ютер в ролі осцилографа, спектроаналізатора, частотоміра і генератора Сучасна вимірювальна апаратура давно зрослася з цифровими і процесорними засобами управління і обробки інформації. Стрілочні покажчики вже стають нонсенсом навіть в дешевих побутових приладах. Аналітичне обладнання все частіше підключається до звичайних ПК через спеціальні плати-адаптери. Таким чином, використовуються інтерфейси і можливості програм додатків, які можна модернізувати і нарощувати без заміни основних вимірювальних блоків, плюс обчислювальна потужність настільного комп'ютера. Подібні засоби для модернізації комп'ютерів випускаються багатьма фірмами. Однак ціна і вузьконаправлена специфіка не роблять це обладнання поширеним в наших умовах.

Цифровий USB осцилограф з комп'ютера, розробка якого проводиться у кваліфікаційній роботі є одним з варіантів подібних вимірювальних інструментів та може використовувати в навчальних цілях. Його можна застосувати в якості осцилографа і пристрою записуючого електричні сигнали в оперативну пам'ять і на жорсткий диск комп'ютера. Схема нескладна і містить мінімум компонентів, в результаті чого вдалося домогтися гарної компактності пристрою.

1 ОГЛЯД ОБЛАСТІ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Використання осцилографів

Всі, хто працює з електрикою, знають, що напруга вимірюють вольтметром, а струм амперметром. Але ці прилади показують тільки те значення струму, яке є в момент вимірювань. Навіть при вимірюванні змінних за значенням і знаку величин ви отримуєте якесь усереднене за певними алгоритмами або законам значення.

Але за допомогою вольтметра можна стежити за тим, як вимірюється величина, правда, з похибками. У стрілочних приладів вони обумовлені конструктивними особливостями, а у цифрових також, але додаються ще й частота дискретизації і інші програмні проблеми.

Осцилограф являє прилад, який використовується для дослідження тимчасових і амплітудних параметрів електричного сигналу, який подається на його вхід, або безпосередньо на екрані, або записується на фотострічці. На сьогоднішній день це один з найбільш поширених типів контрольно-вимірювальних приладів, який поряд з мультиметром дозволяє виробляти виробничі та наукові дослідження.

Усього є кілька типів осцилографів, які розрізняються за характеристиками:

- аналогові;
- аналогово-цифрові;
- цифрові запам'ятовуючі;
- пристрої змішаних сигналів;
- віртуальні пристрої.

За кількістю променів осцилограф може бути:

- однопроменевий;
- двопроменевий і так далі.

Число променів може бути 16 і більше (n-променевої прилад має n сигнальних входів, в тому числі може відображати на екрані одночасно n графіків вхідних сигналів).

Прилади також класифікуються за принципом дії:

- електронний: аналоговий і цифровий.

- електромеханічний: електродинамічний, випрямний, електростатичний, термоелектричний, електромагнітний, електромагнітний.

За розгорткою їх можна поділити:

- спеціальний;

- запам'ятовуючий;

- стробоскопічний;

- швидкісний;

- універсальний.

Є також прилади, які сумісні з іншими вимірювальними пристроями. Це може бути не тільки автономний пристрій, але і приставка, наприклад, комп'ютер, карта розширення або зовсім підключення до зовнішнього порту.

Осцилограф являє вимірювальний прилад, за допомогою нього можна:

- визначити значення напруги сигналу (амплітуду) і тимчасові параметри;

- вимірявши тимчасові характеристики сигналу, вдасться визначити його частоту;

- спостерігати зрушення фаз, що відбувається при проходженні різних ділянок ланцюга;

- з'ясувати змінну (AC) і постійну (DC), які складають сигнал;

- спостерігати спотворення сигналу, який вносить певну ділянку ланцюга;

- з'ясувати співвідношення сигнал / шум, визначити стаціонарність шуму або його зміна за часом;

- зрозуміти процеси, які відбуваються в електричному ланцюзі;

- з'ясувати частоту коливань і так далі.

Ці пристрої переважно застосовуються в електроніці і радіотехніці. Особливо важливим елементом прилад використовується в електромеханічних сферах виробництва. Цей пристрій виступає в якості фіксуючого приладу, який наочно відображає всі коливання електричного струму, що відбуваються в певному електричному механізмі. За допомогою приладу можна знайти перешкоди, а також спотворення проходження електричного імпульсу в самих різних вузлах схеми.

1.2 Принципи роботи осцилографу

Конструкція аналогових пристроїв базується на застосуванні систем аналогової горизонтальної розгортки і електронно-променевих трубок. Одним з головних блоків даних приладів є генератори лінійно мінливого напруги пилкоподібної форми.

Відхилення променя на екрані визначається напруга пластин. Трубки виділяються великим діапазоном частоти. Горизонтальна розгортка функціонує від напруги горизонтальних пластин по лінійної залежності. Верхня межа частоти визначається підсилювачем і ємністю пластин. Нижня межа відповідає 10 герц.

Для візуалізації характеристик і форми в аналогово-цифрових приладах досліджуваного сигналу використовуються системи аналогової горизонтальної розгортки, електронно-променеві трубки, в тому числі генератори лінійно змінюється напруги. До того ж в конструкції приладів є вбудовані запам'ятовують модулі, які використовуються для зберігання зображення.

Запам'ятовуючі цифрові прилади застосовують високошвидкісну оцифровку аналогових сигналів, забезпечують їх зберігання і виводять на рідкокристалічний індикатор, який застосовується замість електронно-променевої трубки. Цифровий осцилограф має перетворювач аналогового

сигналу, підсилювач, дільник, блок управління, пам'ять і блок виведення на РК-панелі.

Пристрої змішаних сигналів швидко оцифровує аналогові сигнали, в тому числі мають функцію введення цифрових послідовностей. Вся необхідна інформація зберігається в модуль, що запам'ятовує і виводиться на рідкокристалічний монітор при необхідності.

Принцип дії

Аналогові пристрої для створення зображення на екрані застосовують електронно-променеву трубку. У ній напругу, яка подається на осі X і Y, змушує точку пересуватися по екрану. На горизонталі можна спостерігати залежність від часу, тоді як по вертикалі йде відображення пропорційне вхідного сигналу. В цілому ж сигнал посилюється і прямує на електроди, які відхиляють по осі Y електронно-променевої трубки із застосуванням аналогової технології.

Цифровий осцилограф працює дещо по-іншому.

Виконується модифікація входить аналогового сигналу в цифрову форму;

Потім відбувається його збереження. Швидкість збереження залежить від керуючого пристрою. Верхня межа визначається швидкістю перетворювача, при цьому біля нижньої межі немає обмежень.

Перетворення сигналу в цифровий код дозволяє підвищити стійкість відображення, зробити масштаб і розтяжку простіше, зберегти дані в пам'ять.

Використання дисплею замість електронної трубки дає можливість відображати будь-які дані, в тому числі виконувати керування приладом. У дорогих приладів встановлені кольорові екрани, завдяки чому вони дають можливість виділяти кольором різні місця, розрізняти курсори і сигнали інших каналів.

Синхронізацію можна спостерігати прямо перед включенням розгортки. Використовувані процесори обробки сигналу дозволяють обробляти сигнал за допомогою аналізу перетворенням Фур'є.

Інформація в цифровому вигляді дає можливість записати екран з підсумками вимірювання в пам'ять, в тому числі роздрукувати на принтері. Більшість приладів мають накопичувачі, щоб можна було записати зображення в архів і надалі провести їх обробку.

1.3 Постановка задач

В кваліфікаційній роботі необхідно розробити навчальний осцилограф на базі плати Arduino UNO.

Максимальна частота вхідного вимірюваного гармонійного сигналу 3 кГц.

Параметри осцилографа:

- максимальна смуга частотного спектра вхідного сигналу повинна лежати в діапазоні [1 Гц, 3000 Гц];
- екран відображення осцилограми в програмі 1280x1024.
- програма повина мати можливість налаштовувати параметри осцилографу;
- для сигналів вище 400 Гц для відображення сигналів необхідно застосовувати зворотнє перетворення Фур’є для отримання осцилограми.

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ ОСЦИЛОГРАФУ

2.1 Опис апаратного забезпечення

Для реалізації осцилографа було обрано плату Arduino UNO та для подальшої модернізації осцилографу графічний дисплей TFT 2,4 дюйми діагоналлю.

Загалом запропонована схема реалізації осцилографа наведена на рис.2.1.

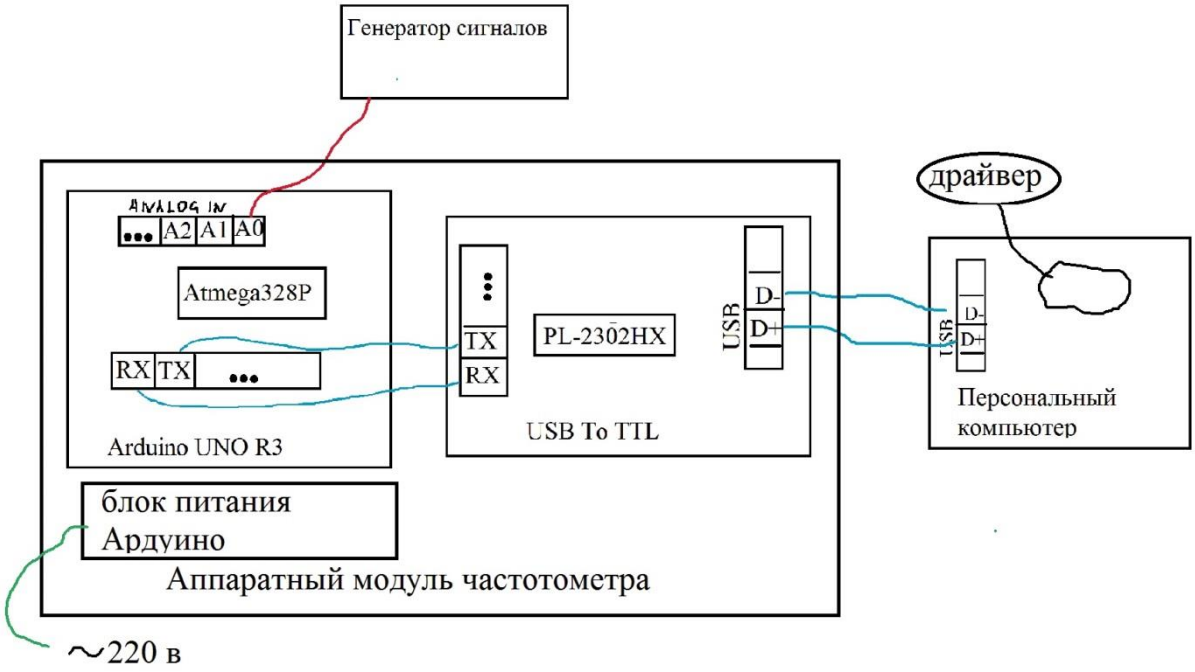

Рисунок 2.1 – Запропонована структурна схема осцилографа

Arduino Uno - плата від компанії Arduino, побудована на мікроконтролері ATmega 328.

Плата має на борту 6 аналогових входів, 14 цифрових висновків загального призначення (можуть бути як входами, так і виходами), кварцовий генератор на 16 МГц, два роз'єми: силовий і USB, роз'єм ISCP для внутрисхемного програмування і кнопку гарячої перезавантаження

пристрою. Для стабільної роботи плату необхідно підключити до харчування або через вбудований USB Роз'єм, або підключивши роз'єм живлення до джерела від 7 до 12В. Через перехідник живлення плата також може працювати і від батареї формату Крона.

Основна відмінність плати від попередніх - для взаємодії з USB Arduino Uno використовує окремий мікроконтролер ATmega8U2. Минулі версії Arduino використовували для цього мікросхему програматора FTDI.

Платформа Arduino Uno має на борту мікроконтролер ATmega328, який володіє Flash, SRAM і EEPROM пам'яттю.

FLASH - 32kB, з яких 0.5kB використовується для зберігання завантажувача.

SRAM (ОЗУ) - 2kB.

EEPROM - 1kB (доступна за допомогою бібліотеки EEPROM).

На платі виведені 14 цифрових пинов (контактів), кожний з яких може працювати як на висновок інформації, так і на введення.

Arduino Uno має на своїй платформі 6 аналогових входів з дозволом 10 Біт на кожен вхід. Даний дозвіл говорить нам про те, що сигнал, що приходить на нього, можна оцифрувати в діапазоні від 0 до 1024 умовних значень.

Згідно загальної схеми осцилограф складається з плати Arduino UNO на базі мікроконтролера Atmega 328P перетворювача інтерфейсів TTL-USB для перетворення сигналу UART мікроконтролера Atmega 328P у шину USB. Перетворювач базується на мікросхемі PL-2302HX. Однак в платі Arduino UNO такий перетворювач вже вмонтовано в плату і відповідно не має потреби ставити додатково.

Мікроконтролер через аналоговий вхід вбудованого АЦП отримує сигнал та перетворює його в цифровий вигляд (10 біт). Основним елементом відображення є персональний комп’ютер, який отримає виміряні дані пакетами та вирисовує в графічному вигляді.

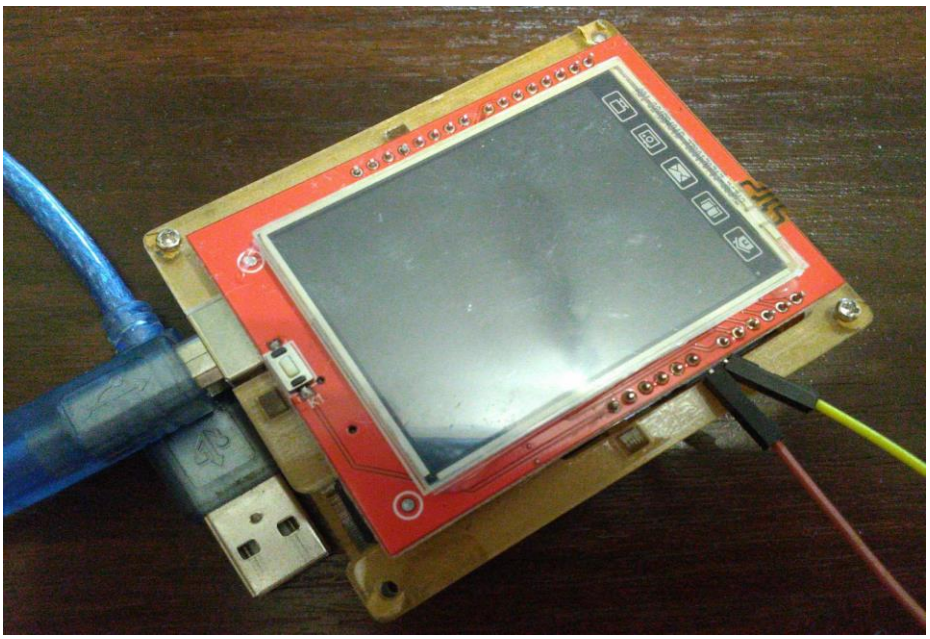

Рисунок 2.2 – Фотографія плати Arduino UNO з модулем дисплею та шнур з’єднання з ПК

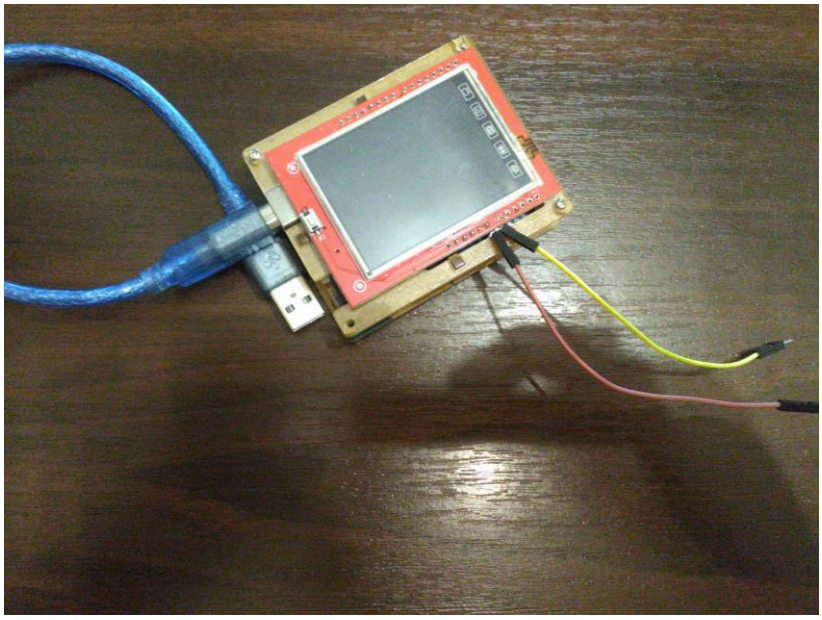

Рисунок 2.3 – Фотографія плати Arduino UNO з модулем дисплею та шнур з’єднання з ПК

Майже кожен мікроконтролер має на борту універсальний послідовний інтерфейс - UART. AVR тут не виняток і підтримує цей протокол в повному обсязі повністю апаратно. За структурою це звичайний асинхронний послідовний протокол, тобто сторона, яка передає по черзі видає в лінію 0 і 1, а приймаюча відстежує їх і запам'ятовує. Синхронізація йде по часу -

приймач і передавач заздалегідь домовляються про те на якій частоті буде йти обмін.

2.2 Протокол обміну даними між апаратною частиною та ПК

Перш ніж перейти безпосередньо до написання коду керуючої програми осцилографа нами було розроблено протокол обміну даними між платою Arduino UNO та персональним компютером.

Дані між платою та комп’ютером передаються по UART інтерфейсу оскільки це найпростіший спосіб зєднання між собою враховуючи, що в платі Arduino UNO вже вмонтовано перетворювач TTL-USB.

Передача даних в UART здійснюється по одному біту в рівні проміжки часу. Цей часовий проміжок визначається заданою швидкістю UART і для конкретного з'єднання вказується в бодах (що в даному випадку відповідає бітам в секунду).

Існує загальноприйнятий ряд стандартних швидкостей: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 бод.

Крім власне інформаційного потоку, UART автоматично вставляє в потік синхронизирующие мітки, так звані стартовий і стоповий біти. При прийомі ці зайві біти видаляються з потоку. Зазвичай стартовий і стоповий біти обрамляють один байт інформації (8 біт), проте зустрічаються реалізації UART, які дозволяють передавати по 5, 6, 7, 8 або 9 біт.

Обрамлені стартом і стопом біти є мінімальною посилкою. Деякі реалізації UART дозволяють вставляти два степових бита при передачі для зменшення ймовірності рассинхронизации приймача і передавача при щільному трафіку. Приймач ігнорує другий стоповий біт, сприймаючи його як коротку паузу на лінії.

Прийнято угоду, що пасивним (за відсутності потоку даних) станом входу і виходу UART є логічна 1.

Стартовий біт завжди логічний 0, тому приймач UART чекає перепаду з 1 в 0 і відраховує від нього часовий проміжок в половину тривалості біта (середина передачі стартового біта). Якщо в цей момент на вході все ще 0, то запускається процес прийому мінімальної посилки. Для цього приймач відраховує 9 бітових тривалостей поспіль (для 8-бітних даних) і в кожен момент фіксує стан входу. Перші 8 значень є прийнятими даними, останнє значення перевірочне (стоп-біт).

Значення стоп-біта завжди 1, якщо реально прийняте значення інше, UART фіксує помилку.

Для формування часових інтервалів передавальний і приймальний UART мають джерело точного часу (тактирования). Точність цього джерела повинна бути такою, щоб сума похибок (приймача і передавача) установки тимчасового інтервалу від початку стартового імпульсу до середини стопового імпульсу не перевищувала половини (а краще хоча б чверті) бітового інтервалу. Для 8-бітної посилки 0,5 / 9,5 = 5% (в реальності не більше 3%). Оскільки ця сума помилок приймача і передавача плюс можливі спотворення сигналу в лінії, то рекомендований допуск на точність тактирования UART - не більше 1,5%.

Оскільки синхронізуючі біти займають частину бітового потоку, то результуюча пропускна здатність UART не дорівнює швидкості з'єднання. Наприклад, для 8-бітних посилок формату 8-N-1 синхронизирующие біти займають 20% потоку, що для фізичної швидкості 115 200 бод дає бітову швидкість даних 92 160 біт / с або 11 520 байт / с.

Частота тактового сигналу на платі Ардуіно становить 16 МГц. Для використання 10 розрядів АЦП частота визначається роботу АЦП повинна бути не більше 200 кГц. Для цього при налаштуванні АЦП необхідно використовувати поділ 16 МГц на 128. При цьому отримуємо частоту тиків АЦП 125 кГц.

При даній частоті тиків АЦП максимальна частота вибірки і оцифровки вхідного сигналу буде дорівнювати:

$$f = \frac{\text{частота АЦП}}{\text{час отримання та оцифрування 1 вибірки}} = \frac{125000\ \text{періодів/с}}{14\ \text{періодів/вибірку}} \quad (2.1)$$

f=8925 = вибірки/сек

Оскільки кожна вибірка містить 10 біт даних (вбудовано АЦП має 10 розрядів). Отже необхідно передавати в персональний комп'ютер 8928 * 10 = 89280 біт / с корисної інформації. Фрейм UART містить в середньому 11 біт - повний розмір і 8 біт даних. Отже, один відлік можна передати одним UART фреймом. Треба накопичувати 4 вибірки, а потім їх передавати 5 UART фреймами. Одна вибірка відбувається за 1 / 8,928кГц = 112 мкс. Отже, 4 вибірки займуть 4 * 112 = 448мкс

Щоб передача не вплинула на максимальну частоту сигналу час передачі має становити не більше одного інтервалу взяття вибірки Тобто воно дорівнює 112 мкс. За цей час необхідно передати 5 * 11 = 55 біт. Тобто швидкість UART повинна бути не менше 55 біт / 112 мкс = 0,492 Мбіт / с. Обираємо найближчу в більшу сторону стандартну швидкість 921600 бод.

Задачею протокола передавання даних є визначення послідовності передавання значень отриманих на аналоговому вході Arduino до персонального комп’ютера та команд з налаштувань роботи плати від комп’ютера.

З боку Arduino реалізуться наступні функції:

- отримання даних з персонального комп'ютера і налаштування АЦП, що знаходиться в мікроконтролері;

- отримання запиту з персонального комп'ютера на початок вимірів і запуск вимірів АЦП з автоматичною передачею даних вимірювань по порту UART (також знаходиться в мікроконтролері) в персональний комп'ютер;

- отримання запиту від персонального комп'ютера на UART про закінчення вимірювання, зупинка процесу вимірювання АЦП і очищення всіх буферів пам'яті мікроконтролера.

Таблиця 2.1 – Протокол обміну між Arduino UNO та ПК

№	Команда з боку ПК	Відповідь з боку Arduino	Опис команди
1	7A 5A (“zZ”)	“OK”	Запит на початок вимірів
2	61 41 (“aA”)	“OK”	Запит про закінчення вимірювань
3	Відсутня	XX XX XX XX XX	Отримати поточні показання
4	3F 3F (“??”)	3F 2B (“?+“) якщо внутрішнє 3F 41 (“?A”) Якщо зовнішнє	Отримати поточні налаштування (тип джерела опорної напруги, що використовують як основний)
5	21 40 (“!@”)	21 40	Задати у якості основного внутрішнє джерело опорної напруги
6	21 5E (“!^”)	21 5E	Задати у якості основного зовнішнє джерело опорної напруги

Значення вимірювань XX XX XX XX XX представляє собою 4 вимірювання по 10 біт які групіровані в 40 біт тобто 5 байт.

З боку програми на комп’ютері реалізуються такі функції:

- посилання даних мікроконтроллеру на для настройки АЦП;

- надсилання запиту на початок вимірів микроконтроллеру і починає приймати дані з мікроконтролера зі зберіганням їх для подальшої обробки (або в оперативній пам'яті, або на диску, якщо отримані дані не поміщаються в оперативну память);

- посилання Arduino запиту про закінчення вимірювань і початок обробки даних з використанням необхідного алгоритму (в залежності від частоти вимірюваних сигналів).

Виміряні дані передаються пакетами по 5 байтів щоб можно було зробити пересилання 4 вимірювань.

У якості налаштувань висилається використання зовнішнього або внутрішнього джерела опорної напруги. Значення напруги зовнішнього джерела опорної напруги задається лише в комп’ютерній програмі оскільки програмно всередині ардуіно ми не може отримати це значення. Задання значення використовується для перерахунку вихідного сигналу АЦП у значення напруги для побудови осцилограми.

2.3 Написання тексту програми Arduino

Для зручної роботи з послідовним портом розробники Arduino написали цілу бібліотеку, яка значно спрощує роботу з портом, абстрагуючись кінцевого користувача від простої, «залізної» роботи з регістрами.

Головною частиною програми для Arduino UNO є блок, що буде відповідати за аналогово-цифрове перетворення з подальшею передачею через порт на персональний комп’ютер.

На платі Arduino Uno аналогові входи мають буквено-цифрові позначення A0, A1, ..., A5 (знизу ліворуч).

Після виклику функції analogRead, мікроконтролер виміряє рівень аналогового сигналу на заданому контакті, і збереже результат роботи АЦП в змінну «результат». При цьому результатом функції analogRead буде число від 0 до 1023.

В Arduino Uno встановлений 10-бітний АЦП, і це означає, що будь-яка напруга на аналоговому вході в діапазоні від 0 до 5 вольт буде перетворено в число з точністю 1/1024 вольта. На графіку буде складно зобразити стільки сходинок. Маючи таку точність, 10-бітний АЦП може «відчути» зміна напруга на вході розміром усього 5 мілівольт (4,9мВ).

У загальному випадку цей діапазон виглядає інакше:

від 0 до опорного напруги

Ця зміна потягне за собою зміну формули розрахунок точності АЦП:

точність = опорна напруга / тисячу двадцять чотири

Опорна напруга визначає межу діапазону, з яким буде працювати АЦП.

У нашому прикладі опорна напруга буде дорівнює напрузі харчування Arduino Uno, яке дав USB порт комп'ютера. У моєму конкретному випадку це напруга була 5.02 Вольта, і я можу сміливо заявити, що виміряв заряд батарейки з високою точністю.

Нарешті розглянемо випадок, коли ми живимо Arduino Uno однією напругою, а в якості опорного хочемо встановити зовсім інше, наприклад, 3.3 Вольта.

Щоб вирішити проблему, нам потрібно подати на цей контакт напруга 3.3 Вольта, і дозволити використання зовнішнього джерела опорного напруги функцією:

analogReference (EXTERNAL);

яку слід викликати всередині функції setup нашої програми.

Також слід враховувати, що результат вимірювання значення напруги не може перевищувати межі діапазону. Якщо ми вибираємо в якості опорного напруги 3.3 Вольта, а надходить сигнал буде з великим напруженням, то ми отримаємо неправильне значення напруги, оскільки АЦП «не знає» про наявність більш високої напруги.

Зміншивши верхню границю напруги АЦП ми зменшима шаг квантування тим самом піднимемо точність.

Для того, щоб повноцінно використовувати АЦП в Arduino Uno, необхідно зробити наступні речі:

analogRead (pin);
analogReference ();

analogReadResolution (bits);

Перш за все необхідно відзначити що канали АЦП Arduino Uno мають за замовчуванням опорне значення 5 В (опорна напруга). Це означає, що максимальна вхідна значення напруги для кожного каналу АЦП Arduino становить 5 В. Але деякі датчики мають вихідну напругу в діапазоні 0-2,5 В, тому якщо ми будемо використовувати опорну напругу за замовчуванням (5 В), то ми втратимо в точності вимірювань. У зв'язку з цим корисно мати можливість зміни значення опорного напруги, для Arduino Uno це робиться за допомогою команди "analogReference ();".

Функція визначає опорна напруга щодо якого відбуваються аналогові вимірювання. Функція analogRead () повертає значення з дозволом 10 біт пропорційно вхідному напрузі на аналоговому вході, і в залежності від опорного напруги.

Можливі настройки:

- DEFAULT: стандартна опорна напруга 5 В (на платформах з напругою живлення 5 В) або 3.3 В (на платформах з напругою живлення 3.3 В);

- INTERNAL: вбудоване опорна напруга 1.1 В на мікроконтролерах ATmega168 і ATmega328, і 2.56 В на ATmega8.

- EXTERNAL: зовнішнє джерело опорного напруги, підключений до висновку AREF

За замовчуванням ми маємо роздільну здатність АЦП, рівну 10 біт, дозвіл АЦП ми також можемо змінити використовуючи команду "analogReadResolution (bits);". Це може бути корисно в ряді випадків.

Тепер, якщо всі установки параметрів роботи АЦП нами зроблені, ми можемо вважати значення АЦП з каналу "0" просто використовуючи інструкцію "analogRead (pin);", де "pin" означає контакт (висновок), на який ми подаємо аналоговий сигнал, в нашому випадку це буде контакт "A0". Значення з виходу АЦП може бути перетворено в число типу integer,

наприклад, за допомогою інструкції "int ADCVALUE = analogRead (A0);", в результаті виконання цієї інструкції значення з використовуваного каналу АЦП після проведення перетворення (тобто АЦП) зберігається в змінній цілого типу (integer) під назвою "ADCVALUE".

Оскільки ми використовуємо стандартні методи для роботи з АЦП та UART підключення додаткових бібліотек для цих модулів не треба. Однак налаштування використання зовнішнього чи внутрішнього джерела опорної напруги для задавання верхньої межі бажано зберігати у EEPROM, а отже треба бібліотека для роботи з EEPROM (<avr/eeprom.h>).

Отже на початку програми проводимо ініціалізацію цієї бібліотеки:

#include <avr/eeprom.h>

Для зберігання поточного вимірювання 4х точок в 5ти байтах задаємо масив OscBuf типу без знакового однобайтного цілого:

byte OscBuf[5]; //буфер для зберігання 4х вимірювань.

Для зберігання поточного налаштування щодо типу джерела опорної напруги, що використовується як основний задаємо булеву змінну VrefInt, що буде приймати значення true, то використовуєть вбудоване джерело опорної напруги, а якщо false – зовнішнє.

bool VrefInt=true; //ознака використання внутрішнього джерела опорної напруги

Також задаємо додаткові змінні:

- char c –змінна для збереження отриманого по UART байту чи байту, що треба на UART відправити;

- byte UARTCnt=0 – лічильник кількості отриманих по UART байт;

- String FromUart="" – отримана від ПК команда

bool NeedMeas=false;//треба проводити вимірювання та відсилання на ПК

Після того як ми обявили всі змінні переходимо до мблоку налаштувань, що виконується одн раз при старті setup().

У цьому блоку ми читаэмо налаштування щодо типу джерела опорноъ напруги, що використовуэться як основний. Це налаштування зберігається за адресою 0x00. Оскільки ознака булева читання проводимо цілого байта з подальшою перевіркою на рівність його 0.

byte btemp=eeprom_read_byte(0x00); //читання байту в якому зберігається ознака використання типу джерела опорної напруги

if (btemp=0) {VrefInt=true;} else {VrefInt=false;}

Відповідно до значення ознаки підключаємо відповідне джерело опорної напруги

if (!VrefInt) {analogReference (EXTERNAL);} else {analogReference (DEFAULT);}

Для того, щоб дані не накладувалися при читанні робимо очистку буфера, що приймає значення з АЦП

ClearBuf();//очищення буферу читання

Ця процедура робуть присвоєння нуля всім 5ти елементам буферу.

Запускаємо роботу UART для передавання даних на ПК та отримання команд від ПК:

Serial.begin(921600);//запускаємо UART

Зкидаємо лічильник прийнятих від ПК байт:

UARTCnt=0;

Далі переходимо до основної процедуру програми – блоку loop.

Перевіряємо чи прийнято байт від ПК

if (Serial.available())

Якщо байт прийнято, то зберігаємо його у змінній c та збільшуємо значення лічильника прийнятих байт.

c = Serial.read();

UARTCnt++;

Якщо кількість отриманих байт дорівнює 2 – обнуляємо лічильник, очищаємо буфер прийому та перевіряємо отримані 2 байти на ідентичність з командами.

```
if (UARTCnt==2) {
UARTCnt=0;
CleanRXBuffer();
```

Якщо команда «zZ» значить був запит на початок вимірів, відповідно встановлюємо змінну признак та відсилаємо відповідь для ПК, що ми команду отримали та прийняли к виконанню.

```
if (FromUart=="zZ") {NeedMeas=true; Serial.print("OK");}
```

Якщо команда «aA» значить був запит на завершення вимірювань, відповідно зкидаємо змінну признак та відсилаємо відповідь для ПК, що ми команду отримали та прийняли к виконанню.

```
if (FromUart=="aA") {NeedMeas=false; Serial.print("OK");}
```

Якщо з боку ПК була отримана команда «??» – відповідно прийшов запит на отримання налаштування щодо типу джерела опорної напруги. Одже в залежності від значення змінної VrefInt відправляємо або 21 40 або 21 5E.

```
if (FromUart=="??") {
    if (VrefInt) {Serial.print("?+");}
    else {Serial.print("?A");}
  }
```

У випадку отримання комбінації "!@" у якості основного джерела опорної напруги встановлюємо внутрішнє джерело, зберігаємо налаштування в EEPROM та задаємо відповідне значення змінні-ознаці.

```
if (FromUart=="!@") {
    VrefInt=true;
    eeprom_write_byte(0x00,0);//зберігаємо налаштування в EEPROM
    analogReference (DEFAULT);
    Serial.print("!@");} //був запит на зміну джерела на внутрішнє
```

При отриманні команди "!^" робимо схожі дії за винятком, що основним джерелом стає зовнішнє джерело опорної напруги.

```
if (FromUart=="!^") {
```

```
VrefInt=false;
eeprom_write_byte(0x00,1);//зберігаємо налаштування в EEPROM
analogReference (EXTERNAL);
Serial.print("!^");}
```

В незалежності чи була розпізнана команда чи ні зкидаємо лічильник та змінну, що зберігала команду ПК.

```
FromUart="";
UARTCnt=0;
```

У випадку коли було отримано від ПК команду про початок вимірювань та відповідно змінна/ознака прийняла значення true починаємо читати данні та зберігати в буфері для подальшого відправлення.

```
int meas1=analogRead(A0);
OscBuf[0]=lowByte(meas1);
int meas2=analogRead(A0);
meas2=meas2<<2;
OscBuf[1]=highByte(meas1)||lowByte(meas2);
int meas3=analogRead(A0);
meas3=meas3<<4;
OscBuf[2]=highByte(meas2)||lowByte(meas3);
int meas4=analogRead(A0);
meas4=meas4<<6;
OscBuf[3]=highByte(meas3)||lowByte(meas4);
OscBuf[4]=highByte(meas4);
Serial.print(OscBuf[4]);
Serial.print(OscBuf[3]);
Serial.print(OscBuf[2]);
Serial.print(OscBuf[1]);
 Serial.print(OscBuf[0]);
```

Дані 10 біт від АЦП записуються в змінні типу int що має 16 біт, а потім починаючи з другого вимірювання зміщаються на відповідно 2, 4 та 6 бітів вліво (рис.2.4).

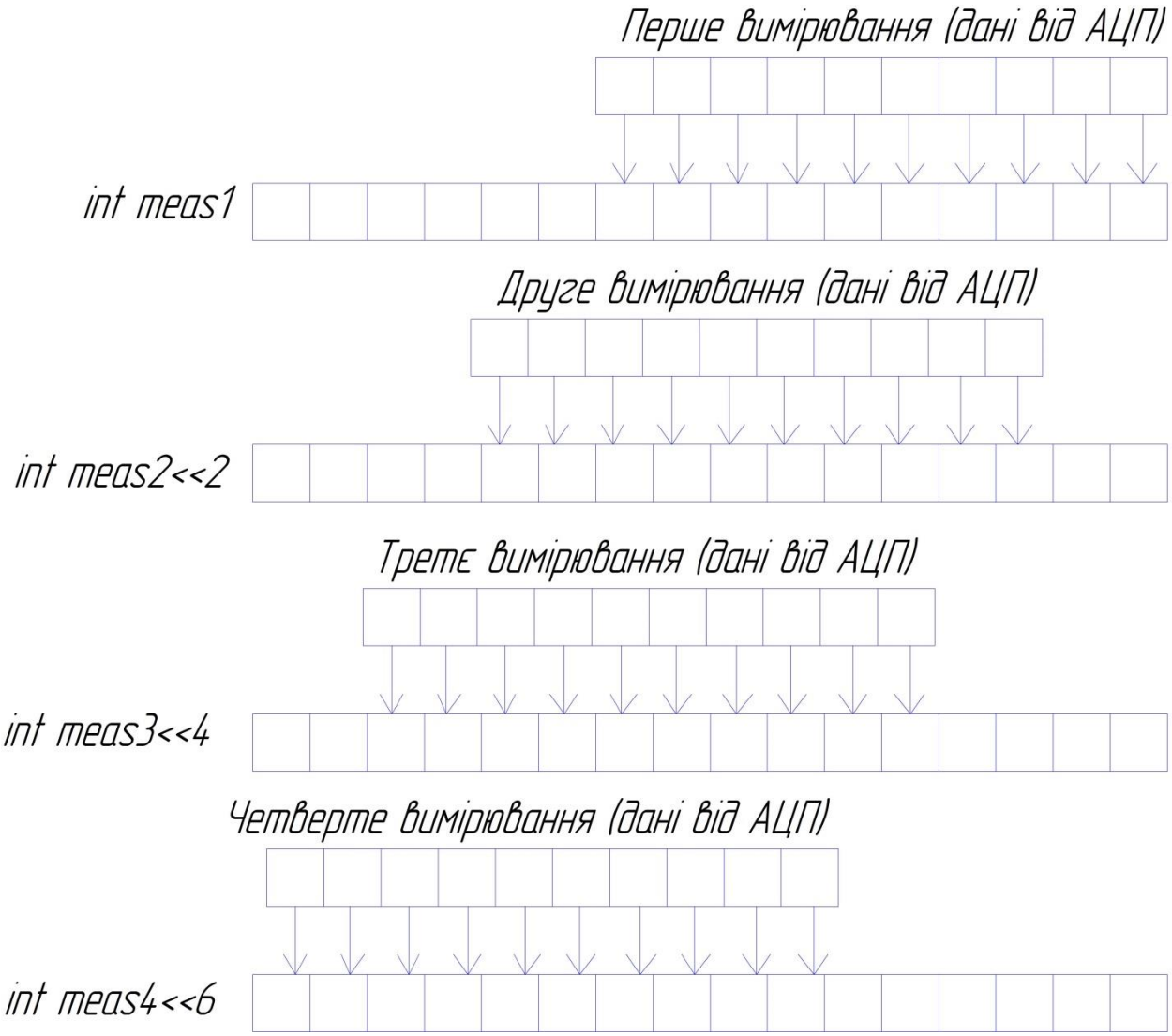

Рисунок 2.4 – Збереження даних від АЦП

Далі за допомогою функцій highByte() та lowByte() 40 бітний простір упаковується в 5 байтів (масив OscBuf) та передається на ПК.

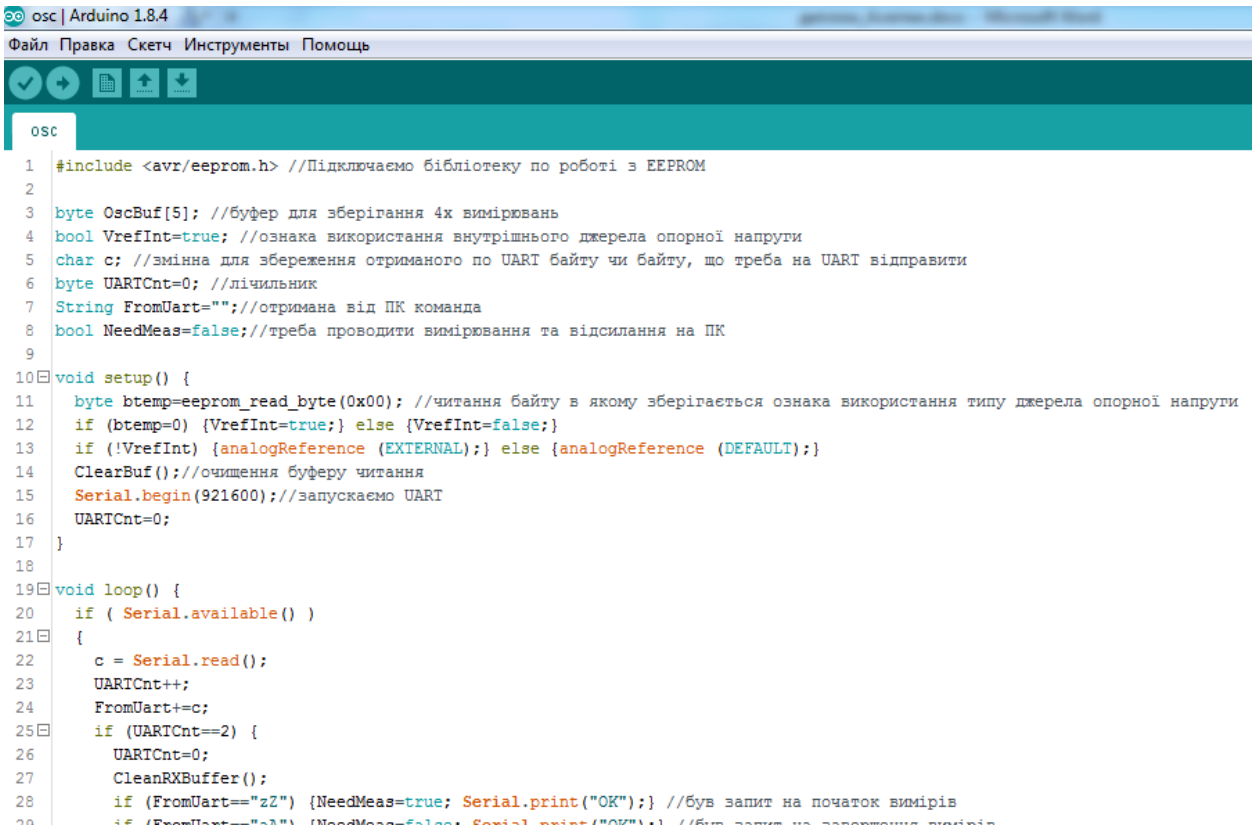

Рисунок 2.5 – Написання програми мікроконтролера у середовищі Arduino IDE

Повний текст програми мікроконтролеру наведено у додатку А.

3 НАПИСАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОМП'ЮТЕРА

Для написання програми відображення даних, що отримуються на осцилографі використовувалося середовище розробки Delphi (рис.3.1).

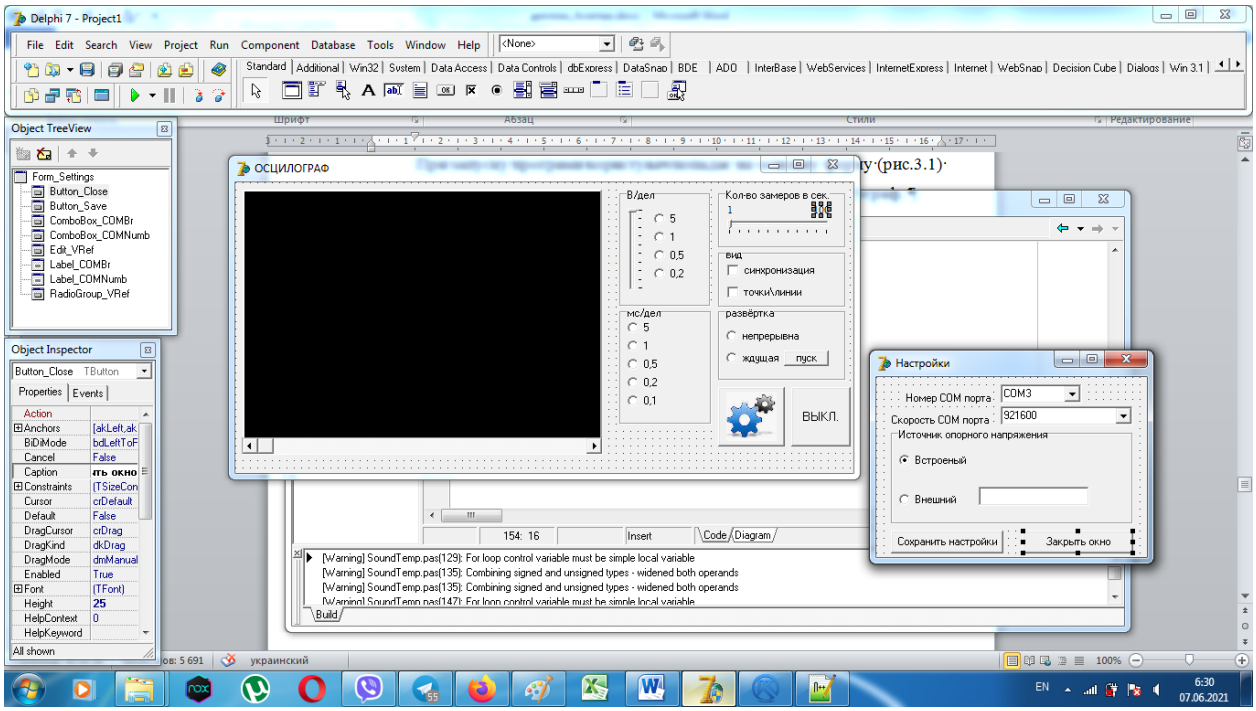

Рисунок 3.1 – Розробка програми для ПК у середовищі Delphi

Delphi - імперативний, структурований, об'єктно-орієнтована мова програмування, діалект Object Pascal. Починаючи з середи розробки Delphi 7.0, в офіційних документах Borland стала використовувати назву Delphi для позначення мови Object Pascal. Починаючи з 2007 року вже мова Delphi (похідний від Object Pascal) почав жити своїм самостійним життям і зазнавав різні зміни, пов'язані з сучасними тенденціями (наприклад, з розвитком платформи .NET) розвитку мов програмування: з'явилися class helpers, перевантаження операторів і інше.

На сьогоднішній день, поряд з підтримкою розробки 32 і 64-розрядних програм для Windows, реалізована можливість створювати додатки для Apple Mac OS X (починаючи з Embarcadero Delphi XE2), IOS (включаючи симулятор, починаючи з XE4 за допомогою власного компілятора), а також, в

Delphi XE5, додатки для Google Android (безпосередньо виконувані на ARM процесорі).

Незалежна, стороння реалізація середовища розробки проектом Lazarus (Free Pascal, компіляція в режимі сумісності з Delphi) дозволяє використовувати його для створення додатків на Delphi для таких платформ, як Linux, Mac OS X і Windows CE. Також робилися спроби використання мови в проектах GNU (наприклад, Notepad GNU) і написання компілятора для GCC.

При створенні мови не ставилося завдання забезпечити максимальну продуктивність виконуваного коду, або лаконічність вихідного коду для економії оперативної пам'яті. Спочатку, мова ставив на перше місце стрункість і високу читаність, оскільки був призначений для навчання дисципліни програмування. Ця початкова стрункість, в подальшому, як у міру зростання апаратних потужностей, так і в результаті появи нових парадигм, спростила розширення мови новими конструкціями.

Так, складність об'єктного C ++, в порівнянні з C, зросла вельми істотно і ускладнила його вивчення в якості першої мови програмування, чого не можна сказати про Object Pascal щодо Pascal.

Програма на комп’ютері реалізує такі функції:

- відправляє данні мікроконтроллеру на для настройки параметрів АЦП;

- відправляє запити про необхідність початку вимірювань мікроконтроллеру плати Arduino та приймає дані з мікроконтролера зі зберіганням їх для подальшої обробки в масиві або у файлі;

- відправляє Arduino запити про необхідність закінчення вимірювань і початок обробки даних з відображенням осцилограми;

- формує якусь графічну лицьову панель подібно лицьовій панелі осцилографа і виводить графіки напруг вхідного вимірюваного сигналу;

- реалізує зміну масштабування осціллограми як з часової осі, так і по осі величин и напруги.

Для відправлення даних мікроконтролеру та отриманню даних у відповідь, а також даних щодо вимірювань було створено окремий потік TCommThread, що обробляє асинхронні події (отримання інформації від плати).

```
type TCommThread = class(TThread)
  private
    procedure QueryPort; //процедура, занимающаяся опросом порта
    function COM_Init(numb:byte; br:longint):boolean;//инициализация
    procedure COM_Close(port_numb:byte);//закрытие порта
    function ReadCOM(var Buf; Size: Word): Integer; //чтение порта
    function WritePort(var Buf; Size: Word): Integer;
    function send_string(s_temp:string):boolean; //отправка строки в порт
  protected
    procedure Execute; override;//переопределим метод запуска потока
  end;
```

Потік TCommThread має такі методи:

- QueryPort – процедура, що проводить опитування порту на наявність нових даних та обробляє їх якщо вони зявилися;

- COM_Init – функція ініціалізації порту з параметрами його номеру та швидкості обміну даних;

- COM_Close – процедура, що закриває роботу з портом номер якого зазначений у якості параметру;

- ReadCOM – функція читання даних з порта заповнює програмний буфер у випадку коли системний буфер має отримані данні;

- WritePort – функція запису програмного буферу у порт для відправлення;

- send_string – функція відправлення строки у порт.

Додатково для визначення наявних в системі портів використовується функція COMList, що шляхом читання з реєстру записів в розділі HKEY_LOCAL_MACHINE\ HARDWARE\DEVICEMAP\SERIALCOMM\

визначає перелік COM портів, що наявні в системі. Ця функція використовується для перевірки наявності виставленого в налаштуваннях порта та для підказки користувачеві про перелік.

Ініціалізація порта проводиться за допомогою системної функції CreateFile, а читання та запис в порт відповідно за допомогою системних функцій ReadFile та WriteFile. При створенні потоку опросу порту та відповідно ініціалізації порта викликається ця функція та отримується унікальний системний заголовок.

FComHandle:=CreateFile(PChar('\\.\'+FCOM), GENERIC_READ or GENERIC_WRITE, 0, nil, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);

Функція CreateFile створює або відкриває каталог, фізичний диск, тому, буфер консолі (CONIN $ або CONOUT $), пристрій на магнітній стрічці, комунікаційний ресурс, поштовий слот або іменований канал. Функція повертає дескриптор, який може бути використаний для доступу до об'єкта.

```
HANDLE CreateFile(
    LPCTSTR  lpFileName,                    // ім'я файлу
    DWORD  dwDesiredAccess,                 // режим доступу
    DWORD  dwShareMode,                     // сумісний доступ
    LPSECURITY_ATTRIBUTES  lpSecurityAttributes, // SD (дескриптор захисту)
    DWORD  dwCreationDisposition,           // порядок дій
    DWORD  dwFlagsAndAttributes,            // атрибути файлу
    HANDLE  hTemplateFile                   // дескриптор шаблону файлу
);
```

У програмі було створено 2 форми які реалізуються інтерфес користувача.

При запуску програми користувач попадає на основну форму (рис.3.1) та бачить зовнішній вид, що нагадує повністю апаратний осцилограф.

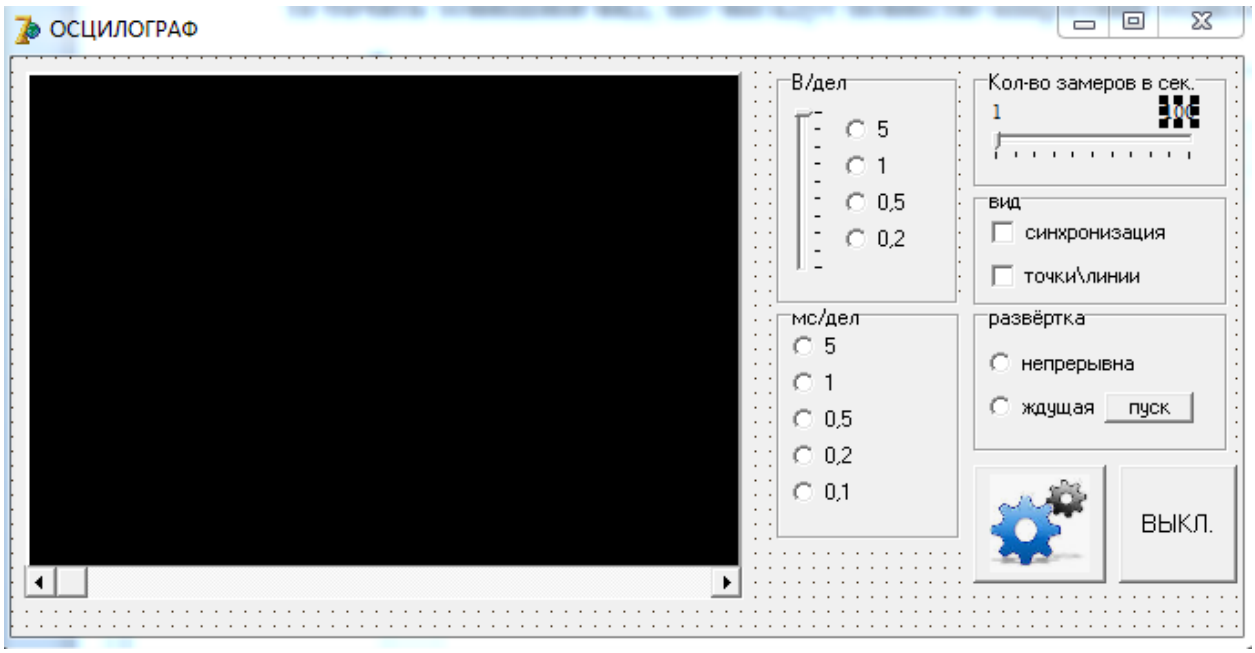

Рисунок 3.2 – Основна програми

На формі є поле виводу зображення, перемикачі масштабу по вертикалі та горизонталі, перемикачі режимів роботи.

Для відображення осцилограми було розміщено елемент інтерфейсу Panel якому задано чоний фон що прорисовування проводити зеленим коліром та було добре видно.

Для перемикачів режимів обрано елементи GroupBox з RadioButton в середині.

Користувач натискаючи на кнопку пуск ініціює процес вимірювань (відповідно програма ПК відправляє в порт команду початку вімірювань та переходить в режим очікування отримання даних).

Отримані дані зберігаються у динамічному масиві OscBuf, що має розмірність елементів 5 байтів. В нього переміщуються отримані пакети для подальшої обробки.

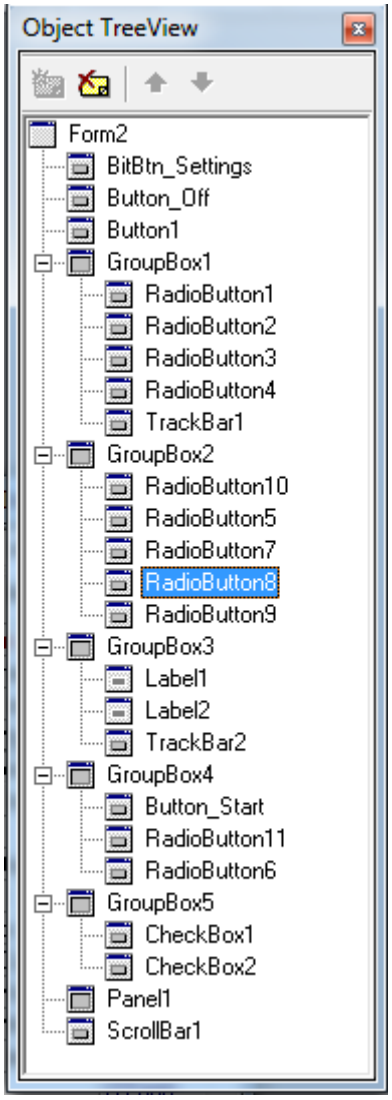

Рисунок 3.3 – Ієрархія об’єктів інтерфейсу головної форми.

Для відмальовування сигналу написана функція grafsignal:

```
function
grafsignal(vid:integer;mash:integer;kontec:integer;X0:integer;Xn:integer;Ykf:integer;Ycm:integer;colP:integer;colL
:integer):integer;
var pen1,pen2,X,n:integer;
begin
  DeleteObject(pen1);
  DeleteObject(pen2); //îñâîáîäèòü ïåðüÿ
  pen2:=CreatePen(PS_SOLID,1,colP);//ñîçäàòü ïåðî
  pen1:=CreatePen(PS_SOLID,1,colL);
  if mash=0 then mash:=1; //ìàøòàá

  releaseDC(kontec,hwd); // îñâîáîäèòü êîíòåêñò
  hwd:= getdc(kontec);  // ñîçäàòü êîíòåêñò
//releaseDC(kontec,hwd); //
```

```
 if OSCHILOGRAF.Form2.CheckBox1.Checked then  n:=faza else n:=0;
 i:=0;
 for i:=n+X0 to Xn+n do begin       //
   s1:= DMemBlock[i*4+2];//ïëàäøèé áàéò
   if s1>$80 then s1:=($ff-s1);
   s2:= DMemBlock[(i*4+3)];
   if s2 >127 then  ss:=-(($ff-s2)*255)+s1 else ss:=s1+(s2*255);

 if mash<0 then X:=(i-n-X0)*(-mash) else X:= (i-n-X0)div mash;
    if vid=0 then begin
               setpixelV(hwd,X,((ss div Ykf)+Ycm),colP); //íàðèñîâàòü òî÷êó
                 end;
     if vid=1 then begin
          SelectObject(hwd,pen1);//ïîäêëþ÷èòü ïåðî
          lineto(hwd,X,((ss div Ykf)+Ycm)); //íàðèñîâàòü ëèíèþ
        end;
       end;
 ////////////////////////////////////////////////////////////////////////////////
 movetoex(hwd,0,0,nil);
 i:=0;
 for i:=n+X0 to Xn+n do begin
             s3:= DMemBlock[i*4+0];//ïëàäøèé áàéò
             if s3>$80 then s3:=($ff-s3);
            s4:= DMemBlock[(i*4+1)];
           if s4 >127 then  ss1:=-(($ff-s4)*255)+s3 else ss1:=s3+(s4*255);
 if mash<0 then X:=(i-n-X0)*(-mash) else X:= (i-n-X0)div mash;
    if vid=0 then begin
                 setpixelV(hwd,X,((ss1 div Ykf)+Ycm),colL);//
                 end;
     if vid=1 then begin
              SelectObject(hwd,pen2);  lineto(hwd,X,((ss1 div Ykf)+Ycm));
                 end;


       end;
        ss:=0; ssl:=0;
          result:=0;
DeleteObject(pen1);DeleteObject(pen2); //îñâîáîäèòü ïåðüÿ
 end;
```

Для налаштувань зєднання та типу джерела опорної напруги, що використовується платою, а також щоб задачати верхню межу переводу даних АЦП в значення напруги створено спеціальну форму (рис.3.4). У

формі також вказується значення напруги у випадку коли джерело опорної напруги завнішнє. Цей параметр необхідний для визначення дійсного значення напруги при перерахунку отриманих даних АЦП з плати.

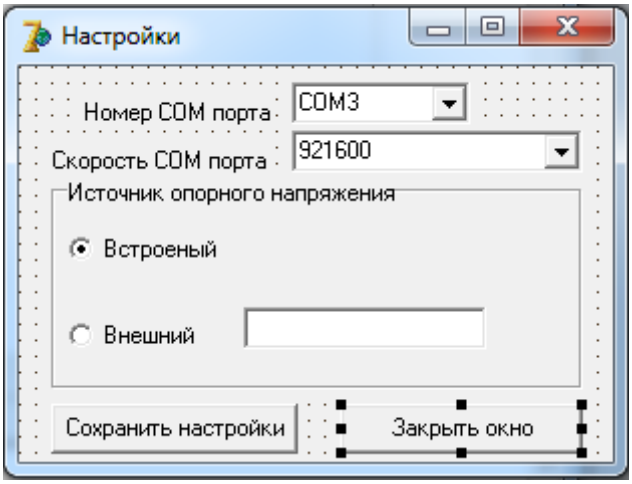

Рисунок 3.4 – Форма налаштувань

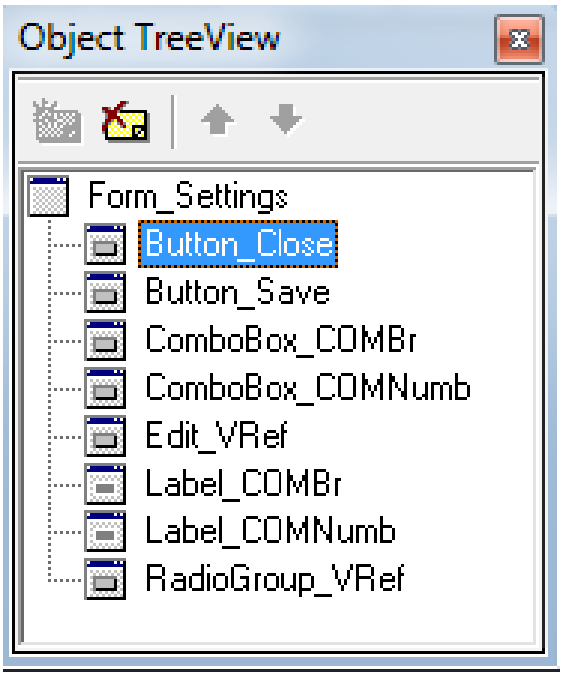

Рисунок 3.5 – Ієрархія об’єктів інтерфейсу форми налаштувань

Для зберігання поточних налаштувань було створено файл налаштувань Sets.dat, що має структуру наведену у табл.3.1.

Таблиця 3.1 – Структура файлу налаштувань Sets.dat

№	Параметр	Опис	Приклад
1	COMNumb	Номер порту до якого приєднана плата Arduino UNO	COMNumb=3
2	COMBr	Швидкість обміну даними по порту	COMBr=19600
3	VRefType	Тип джерела опорної напруги, що використовується як базова	VRefType=0
4	VRefVal	Значення напруги джерела опорної напруги (при внутрішньому значення дорівнює 5)	VRefVal=5
5	XDiv	Коефіцієнт відображення по горизонталі	XDiv=0.5
6	YDiv	Коефіцієнт відображення по вертикалі	YDiv=1

Робота програми при подаванні вимірювального сигналу зображена на рис.3.6.

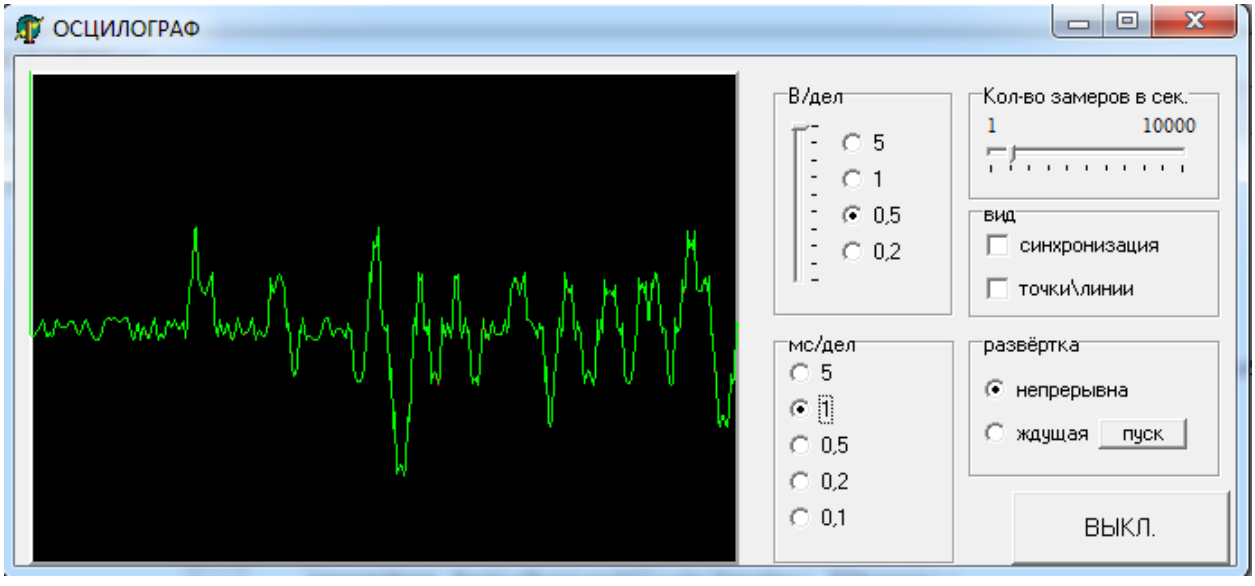

Рисунок 3.6 – Робота програми осцилографа

ВИСНОВКИ

В кваліфікаційній роботі було розроблено електроний осцилограф на базі плати Arduino UNO. Для створення цього осцилографа було написано програму мікропроцесорного модуля Arduino UNO на мові програму вання C, а також програму відображення даних для персонального комп’ютера у середовищі розробки Delphi.

Найбільшою перевагою розробленого осцилографа є його простота та невелика ціна.

Подальша робота над розробкою дозволить задіяти наявний модуль графічного дисплею та в залежності від режиму який обрав користувач можна буде або виводити на ПК або на невеликий графічний дисплей, що зробить пристрій більш мобільним.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Кожемяко В. П., Тарновський М. Г., Павлов С. В. Схемотехніка сучасного приладобудування. Частина IV. –Вінниця: ВНТУ, 2003. –136 с.

2. Кожемяко В. П., Павлов С. В., Тарновський М. Г. Оптоелектронна схемотехніка. Навчальний посібник. –Вінниця: УНІВЕРСУМ-Вінниця, 2008. –189 с.

3. Проекты с использованием контроллера Arduino. —СПб.: БХВ-Петербург, 2014. —400 с.: ил. —(Электроника)

4. Bas Wijnen, G. C. Anzalone and Joshua M. Pearce, Open-source mobile water quality testing platform. Journal of Water, Sanitation and Hygiene for Development, 4(3) pp. 532–537 (2014). doi:10.2166/washdev.2014.137

5. xoscillo –A software oscilloscope that acquires data using an arduino or a parallax (more platforms to come). –Google Project Hosting". Code.google.com. Retrieved 2013-01-18.

6. Давыдкин Максим Николаевич, Дистанционный курс «Мехатроника и робототехникаArduino», https://remote.misis.ru/enroll/XPE6RH

7. Джереми Блум Изучаем Arduino. Инструменты и методы технического волшебства. - СПб.:БХВ-Петербург, 2016.

8. Кривонос О.М. Огляд платформи Arduino Nano 3.0 та перспективи використання під час навчального процесу /О.М.Кривонос, Є.В.Кузьменко, С.В.Кузьменко// Інформаційні технології і засоби навчання. Том 56, No 6. - Київ, 2016.-С.79-80

9. В.Петин. Проекты с использованием контроллера Arduino. –БХВ-Петербург, 2015, 448с.

10. Дж.Блум. ИзучаемArduino: инструменты и методы технического волшебства. –БХВ-Петербург, 2015, 336 с.

11. Саймон Монк Программируем Arduino. –СПб. : Питер, 2017. –252 с.

12. Мікроконтролери AVRсімейства Mega/ Євстіфеєв А.В. –Москва: Видавничий дім «Додека -XXI», 2007.-595с.

13. ArduinoUnoR3. Відкритий електроний ресурс. Режим доступу: https://hcomp.ru/downloads/arduino/UNOr3/arduino_uno_r3_RUS.pdf. Дата доступу: 19.05.2021

14. Arduino Cookbook / Michael Margolis -NewYorkO'Reilly Media -M., 2011. -648 с.

ДОДАТОК А – ПРОГРАМА ОСЦИЛОГРАФА ДЛЯ ARDUINO UNO

```
#include <avr/eeprom.h> //Підключаємо бібліотеку по роботі з EEPROM

byte OscBuf[5]; //буфер для зберігання 4х вимірювань
bool VrefInt=true; //ознака використання внутрішнього джерела опорної напруги
char c; //змінна для збереження отриманого по UART байту чи байту, що треба на UART відправити
byte UARTCnt=0; //лічильник
String FromUart="";//отримана від ПК команда
bool NeedMeas=false;//треба проводити вимірювання та відсилання на ПК

void setup() {
  byte btemp=eeprom_read_byte(0x00); //читання байту в якому зберігається ознака використання типу
джерела опорної напруги
  if (btemp=0) {VrefInt=true;} else {VrefInt=false;}
  if (!VrefInt) {analogReference (EXTERNAL);} else {analogReference (DEFAULT);}
  ClearBuf();//очищення буферу читання
  Serial.begin(921600);//запускаємо UART
  UARTCnt=0;
}

void loop() {
  if ( Serial.available() )
  {
    c = Serial.read();
    UARTCnt++;
    FromUart+=c;
    if (UARTCnt==2) {
      UARTCnt=0;
      CleanRXBuffer();
      if (FromUart=="zZ") {NeedMeas=true; Serial.print("OK");} //був запит на початок вимірів
      if (FromUart=="aA") {NeedMeas=false; Serial.print("OK");} //був запит на завершення вимірів
      if (FromUart=="??") {//був запит налаштувань
        if (VrefInt) {Serial.print("?+");}
        else {Serial.print("?A");}
      }
      if (FromUart=="!@") {
          VrefInt=true;
          eeprom_write_byte(0x00,0);//зберігаємо налаштування в EEPROM
          analogReference (DEFAULT);
          Serial.print("!@");} //був запит на зміну джерела на внутрішнє
```

```
    if (FromUart=="!^") {
       VrefInt=false;
       eeprom_write_byte(0x00,1);//зберігаємо налаштування в EEPROM
       analogReference (EXTERNAL);
       Serial.print("!^");} //був запит на зміну джерела на зовнішнє
    FromUart="";
    UARTCnt=0;
   }
  }
  if(NeedMeas) {//якщо треба проводити вимірювання
    ClearBuf();//очищення буферу читання
    int meas1=analogRead(A0);
    OscBuf[0]=lowByte(meas1);
    int meas2=analogRead(A0);
    meas2=meas2<<2;
    OscBuf[1]=highByte(meas1)||lowByte(meas2);
    int meas3=analogRead(A0);
    meas3=meas3<<4;
    OscBuf[2]=highByte(meas2)||lowByte(meas3);
    int meas4=analogRead(A0);
    meas4=meas4<<6;
    OscBuf[3]=highByte(meas3)||lowByte(meas4);
    OscBuf[4]=highByte(meas4);
    Serial.print(OscBuf[4]);
    Serial.print(OscBuf[3]);
    Serial.print(OscBuf[2]);
    Serial.print(OscBuf[1]);
    Serial.print(OscBuf[0]);
  }
}

//очищення буферу приема UART
void CleanRXBuffer()
{
  delay(1);
  while(Serial.available())
  {
    Serial.read();
  }
}

//очищення буферу читання АЦП
```

```
void ClearBuf()
{
  OscBuf[0]=0;
  OscBuf[1]=0;
  OscBuf[2]=0;
  OscBuf[3]=0;
  OscBuf[4]=0;
}
```

ДОДАТОК Б – ОСНОВНІ МОДУЛІ ПРОГРАМИ ДЛЯ ПК

```
unit OSCHILOGRAF;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,soundTemp, StdCtrls,mmsystem, ExtCtrls, ComCtrls, Buttons;

type
  TForm2 = class(TForm)
    Button1: TButton;
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    TrackBar1: TTrackBar;
    GroupBox2: TGroupBox;
    RadioButton5: TRadioButton;
    RadioButton7: TRadioButton;
    RadioButton8: TRadioButton;
    RadioButton9: TRadioButton;
    RadioButton10: TRadioButton;
    GroupBox3: TGroupBox;
    TrackBar2: TTrackBar;
    GroupBox4: TGroupBox;
    RadioButton11: TRadioButton;
    Button_Start: TButton;
    RadioButton6: TRadioButton;
    GroupBox5: TGroupBox;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
```

```
    Label1: TLabel;
    Label2: TLabel;
    Button_Off: TButton;
    BitBtn_Settings: TBitBtn;
    ScrollBar1: TScrollBar;
    procedure Button1Click(Sender: TObject);
    procedure MMOutOpen(var msg: Tmessage); message MM_WIM_DATA;//ñîîáùåíèå çâ.êàðòû
    procedure FormCreate(Sender: TObject);
    procedure RadioButton1Click(Sender: TObject);
    procedure TrackBar1Change(Sender: TObject);
    procedure Button_StartClick(Sender: TObject);
    procedure RadioButton5Click(Sender: TObject);
    procedure TrackBar2Change(Sender: TObject);
    procedure RadioButton6Click(Sender: TObject);
    procedure Button_OffClick(Sender: TObject);
    procedure BitBtn_SettingsClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

uses Settings;

{$R *.dfm}
var vtemp1,vtemp, vdel,mash,sdvig:integer;
  sx,ln:integer;
//îáðàáîòêà ñîîáùåíèé çâ.êàðòû
procedure TForm2.MMOutOpen(var msg: Tmessage);
begin
```

```
if  RadioButton6.Checked then begin
      if CheckBox1.Checked then sx:=0 else sx:=1;
      if CheckBox2.Checked then ln:=0 else ln:=1;
      clearHDC(Panel1.Handle);
      grafsignal(ln,mash,panel1.Handle,sdvig,vdel+sdvig,vtemp,Panel1.Height div 2,$00ff00,$0000ff);
       sleep(300); //÷àñòîòà îáíîâëåíèÿ ýêðàíà
       msgwavIn;//ïîãðîìîâêà çâ.êàðòû ê ñëåäóþùåìó öèêëó
       end;

end;
//ñòàðò ðàáîòû îñöèëîãðàôà
procedure TForm2.Button1Click(Sender: TObject);
begin
//2-êàíàëà, 44100Ãö-÷àñòîòà, 176400-ðàçìåð áóôåðà, 4áàéò-áëîê, 16áèò-ïðåîáðàçîâ, 40000 áèò/ñåê
  createsound(2,44100,176400,4,16,40000);
  startInsound(form2.Handle); //
end;
////////////////////////////////////////////////////////////////////////////////

procedure TForm2.FormCreate(Sender: TObject);
begin
  RadioButton1.Checked :=true;
  RadioButton6.Checked:=true;
  RadioButton5.Checked:=true;
  vtemp:=1000;
  button1.Click;
  TrackBar2.Position:=1000;
end;

procedure TForm2.RadioButton1Click(Sender: TObject);
begin
  if RadioButton1.Checked then  vtemp1:=1000;
  if RadioButton2.Checked then  vtemp1:=100;
  if RadioButton3.Checked then  vtemp1:=10;
  if RadioButton4.Checked then  vtemp1:=1;
  TrackBar1.Position:=1;
```

```
  vtemp:=vtemp1;
end;

procedure TForm2.TrackBar1Change(Sender: TObject);
begin
  vtemp:= vtemp1 div TrackBar1.Position;
end;

procedure TForm2.Button_StartClick(Sender: TObject);
begin
  clearHDC(Panel1.Handle);
  application.ProcessMessages;
  msgwavIn;
  grafsignal(ln,mash,panel1.Handle,sdvig,vdel+sdvig,vtemp,Panel1.Height div 2,$00ff00,$0000ff);
end;

procedure TForm2.RadioButton5Click(Sender: TObject);
begin
  if RadioButton5.Checked then begin  vdel:=10000; mash:=10; end;
  if RadioButton7.Checked then begin  vdel:=1000; mash:=1; end;
  if RadioButton8.Checked then begin  vdel:=1000; mash:=-5; end;
  if RadioButton9.Checked then begin  vdel:=100; mash:=-20; end;
  if RadioButton10.Checked then begin  vdel:=100; mash:=-40; end;
end;

procedure TForm2.TrackBar2Change(Sender: TObject);
begin

sdvig:= TrackBar2.Position;
      clearHDC(Panel1.Handle);
      application.ProcessMessages;
      grafsignal(ln,mash,panel1.Handle,sdvig,vdel+sdvig,vtemp,Panel1.Height div 2,$00ff00,$0000ff);

end;
```

```
procedure TForm2.RadioButton6Click(Sender: TObject);
begin
  if form2.Active then msgwavIn;
end;

procedure TForm2.Button_OffClick(Sender: TObject);
begin
  form2.Close;
end;

procedure TForm2.BitBtn_SettingsClick(Sender: TObject);
begin
  Form_Settings.Show;
end;

end.

unit Scales;
//Модуль работы с портом

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, Registry, Sockets;

//определим тип TComThread - поток для порта первых весов
type TCommThread = class(TThread)
  private
    procedure QueryPort; //процедура, занимающаяся опросом порта
    function COM_Init(numb:byte; br:longint):boolean;//инициализация
    procedure COM_Close(port_numb:byte);//закрытие порта
    function ReadCOM(var Buf; Size: Word): Integer; //чтение порта
 //   function WritePort(var Buf; Size: Word): Integer;
 //   function send_string(s_temp:string):boolean; //отправка строки в порт
  protected
```

```
    procedure Execute; override;//переопределим метод запуска потока
  end;

var
  CommThread:TCommThread;
  COMPortList:TStringList;
  TcpClientScales:TTcpClient;
  COMPortNumb:byte;//номер порта для работы
  COMPortBr:longint;//скорость COM порта для работы
  Parity:string[1];//          Проверка на четность
  Bits:integer; //   Количество бит в символе
  StopBits:integer;//          Количество стоповых бит
  FCOMHandle:THandle; //заголовок работы с портом
  FCOM:string; //строка с именем порта "COM1" например
  WghtConnected:boolean;//признак что связь с COM портом установлена
  ScaleName:String;//название весового терминала
  ScaleBuf:String;//накопленный буфер данных
  function COMList:TStringList;//чтение из реестра списка подключенных COM портов
  function IsCOMInSystem(n:byte):boolean; //проверяем есть ли COM порт с номером n в системе

implementation

uses Main,ParametersSetings,HBMDigital,Settings,ScalesReq;

//чтение из реестра списка подключенных COM портов
//при вызове создается COMPortList в котором храняться последний считанный списко портов
function COMList:TStringList;
var
  Reg : TRegistry; //переменная для работы с реестром
  reg_parameters:TStringList;
  Subkey:String;
  i:integer;
begin
  Subkey:='HARDWARE\DEVICEMAP\SERIALCOMM\';
  reg_parameters := TStringList.Create;//создаем пустой список строк
  COMPortList:=TStringList.Create;
```

```
  reg := TRegistry.Create(); //создаем элемент работы с реестром
  reg.RootKey := HKEY_LOCAL_MACHINE; //указываем раздел реестра для работы
  reg.Access := KEY_QUERY_VALUE;
  reg.OpenKeyReadOnly(Subkey);
  reg.GetValueNames(reg_parameters); //извлекаем список параметров в подразделе
  if reg_parameters.Count>0 then for i:=0 to reg_parameters.Count-1 do
     COMPortList.add(reg.ReadString(reg_parameters[i]));
  Result:=COMPortList;
  reg.Free;
  reg_parameters.free;
end;

//проверяем есть ли COM порт с номером n в системе
function IsCOMInSystem(n:byte):boolean;
var
  SLTemp:TStringList;
  s_temp:string;
  i:integer;
begin
  Result:=false;
  SLTemp:=TStringList.Create;
  SLTemp:=COMList;
  if SLTemp=nil then Exit;
  s_temp:='COM'+IntToStr(n);
  for i:=0 to SLTemp.Count-1 do
    if SLTemp[i]=s_temp then begin
      Result:=true;
      //FreeAndNil(SLTemp);
      Exit;
      end;
  //FreeAndNil(SLTemp);
end;

//инициализация потока для весов
procedure StartComThread;
begin
```

```
  ScaleBuf:='';//очищаем буфер полученных с порта данных
  WghtNow:='XXXXX'; //поскольку неизвестно будет ли вес - задаем необределенность
  //пытаемся инициализировать поток
  CommThread := TCommThread.Create(False);
  //проверяем получилось или нет
  if CommThread = nil then
  begin
    //ошибка, все выключаем
    SysErrorMessage(GetLastError);
    Exit;
  end;
end;

//Реализуем асинхронный опрос порта для первых весов и чтение из него данных
procedure TCommThread.QueryPort;
var
  InputBuffer: array[0..255] of char;  // приёмный буфер размером 256 символов (или сколько надо)
  ErrorCode: integer;  // переменная для приёма кода ошибки
  s_temp,s_temp2:string;//временная переменная
begin
  //!! Очищаем буфер перед приемом
  FillChar(InputBuffer, Length(InputBuffer),0);//!!
  //читаем порт
  ErrorCode:= ReadCOM(InputBuffer, Length(InputBuffer));  // здесь собственно приём пакета
  if ErrorCode = 0 then WghtNow:='XXXXX' //если пакет пустой запрещаем работу с портом
  else
    begin
    // здесь из InputBuffer забираем принятые символы (байты)
    s_temp:=string(InputBuffer);
    ScaleBuf:=ScaleBuf+s_temp;
    s_temp2:=ScaleBuf; //!!!!
    if length(s_temp)=0 then Exit;
    //расшифровываем вес
    //if length(s_temp)>20 then

    WghtNow:=DecodeWght(ScaleName, s_temp, s_temp2); //ScaleBuf);
```

```
    if WghtNow<>'' then begin  //если вес распознался - обнуляем буфер
      ScaleBuf:='';
      end
      else
      WghtNow:='XXXXX';
      end;
end;

//=========================================================================
//ПОТОК ДЛЯ первых весов - закрытие COM порта
procedure TCommThread.COM_Close(port_numb:byte);
begin
   // Закрытие порта
  CloseHandle(FCOMHandle);
end;

//=========================================================================
//ПОТОК ДЛЯ первых  весов - чтение пакета из COM порта
function TCommThread.ReadCOM(var Buf; Size: Word): Integer;
var
  NumberOfBytesReaded: Cardinal;
begin
  Result:=0;
  if not ReadFile(FCOMHandle, Buf, Size, NumberOfBytesReaded, nil) then exit;
  Result:=NumberOfBytesReaded;
end;

//=========================================================================
//ПОТОК ДЛЯ первых  весов - инициализация COM порта
function TCommThread.COM_Init(numb:byte; br:longint):boolean;
var
  OldDCB:TDCB;
  Timeouts:TCommTimeouts;
  s_temp:string;
begin
  FCOM:='COM'+inttostr(numb);
```

```
  Result:=False;
  //Создание и инициализация порта
  FComHandle:=CreateFile(PChar('\\.\'+FCOM), GENERIC_READ or GENERIC_WRITE,
        0, nil, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
  if (FComHandle=INVALID_HANDLE_VALUE) or (FComHandle=0) then Exit;
  if not(Windows.GetCommState(FComHandle,OldDCB)) then begin
    s_temp:='';
    Exit;
    end;
  //GetCommState извлекает данные о текущих настройках управляющих сигналов для указанного
коммуникационного устройства.
  //Структура DCB определяет настройки управления последовательным коммуникационным устройством.
  OldDCB.BaudRate:=br;
  OldDCB.Parity:=NOPARITY; //!!!!добавить заполнение из настроек
  // бывает EVENPARITY, MARKPARITY, NOPARITY, ODDPARITY, SPACEPARITY
  OldDCB.ByteSize:=8; //!!!!добавить заполнение из настроек
  OldDCB.StopBits:=ONESTOPBIT;  //!!!!добавить заполнение из настроек
  //бывает ONESTOPBIT, ONE5STOPBIT, TWOSTOPBIT
  if not(Windows.SetCommState(FComHandle,OldDCB)) then Exit;//если нет портов - выход
  if Not SetupComm(FComHandle,256,256) then Exit; //размеры буфера приема и передачи
  if GetCommTimeouts(FComHandle,Timeouts) then
    begin
      Timeouts.ReadIntervalTimeout:=50;//50
      Timeouts.ReadTotalTimeoutMultiplier:=0;
      Timeouts.ReadTotalTimeoutConstant:=50; //50
      Timeouts.WriteTotalTimeoutMultiplier:=1;
      Timeouts.WriteTotalTimeoutConstant:=10;//10
      if not SetCommTimeouts(FComHandle,Timeouts) then Exit;
    end
  else
    Exit;
  // Сброс порта
  if not(PurgeComm(FComHandle, PURGE_TXABORT or PURGE_RXABORT or PURGE_TXCLEAR or
PURGE_RXCLEAR)) then Exit;
  EscapeCommFunction(FComHandle, CLRRTS);
  if not SetCommMask(FComHandle, EV_RXCHAR) then Exit;
```

```
  Result:=True;
end;

//Запускаем процедуру опроса порта в нашем потоке для первых весов.
procedure TCommThread.Execute;
begin
if COM_Init(COMPortNumb, COMPortBr)=true then begin
  WghtConnected:=true;//признак что связь с COM портом установлена
  repeat
    QueryPort; //процедура опроса порта будет производиться пока поток не будет прекращен
  until Terminated;
  end
  else begin
    WghtConnected:=false;//признак что связь с COM портом не установлена
    Terminate;
  end;
end;
```

ДОДАТОК В – ПРЕЗЕНТАЦІЯ