

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 Модуль контролю робочого часу з алкотестером для прохідних підприємства

Виконав: студент(ка) 4 курсу, групи РТ-118сп

Спеціальності 172 «Телекомунікації та радіотехніка»

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Радіоелектронні апарати та засоби»

Кучеров І.В.

(прізвище та ініціали)

Керівник

Малий О.Ю.

(прізвище та ініціали)

Рецензент

Мороз Г. В.

(прізвище та ініціали)

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

В.О. Завідувач кафедри ІТЕЗ

Овреніч Є.В.

« 27 » 05

2021 року

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

Кучеров Іван Володимирович

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Модуль контролю часу з алкотестером для прохідних на підприємствах

керівник проєкту (роботи) Малий Олександр Юрійович, к.т.н.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «26» квітня 2021 року №161

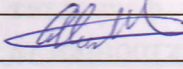
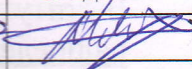
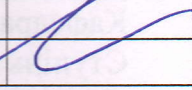
2. Строк подання студентом проєкту (роботи) 7 червня 2021 року

3. Вихідні дані до проєкту (роботи) вимоги до сучасних СКД, таблиця перерахунку концентрації парів алкоголю, допускові медичні рівні алкоголю в крові людини

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Огляд області розробки, постановка задач кваліфікаційної роботи, розробка структури модулю, вибір елементів, розробка конструкції модулю, розробка програмного забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) 26 рисунків, 19 таблиць, 18 слайдів

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1 - 3	Малий О.Ю., доцент		
Нормоконтроль	Поспеева 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 побудов СКД та основи вимірювання парів алкоголю	26.04.21	
2	Постановка технічного завдання	28.04.21	
3	Розробка структурної схеми та вибір елементів	29.04.21	
4	Розробка конструкції	03.05.21	
5	Написання тексту програмного забезпечення	29.05.21	
6	Оформлення ПЗ та захист дипломного проекту	01.06.21	

Студент(ка)

(підпис)

Кучеров І.В.

(прізвище та ініціали)

Керівник проекту (роботи)

(підпис)

Малий О.Ю..

(прізвище та ініціали)

РЕФЕРАТ

ПЗ: 59 с., 26 рис., 19 табл., 6 джерел, 18 слайдів.

Об'єкт розробки: система контролю доступом на підприємство.

Предмет розробки: електронний модуль та програмне забезпечення для контролю пропускового режиму на прохідних підприємства з можливістю контролю рівня алкоголю в диханні працівників.

Метою роботи: розробка апаратного модуля контролю доступу з алкотестером, написання керуючої програми мікроконтролера та програми для фіксації часу проходів через прохідну та рівень алкоголю в диханні робітників.

Для здійснення поставленої мети необхідно було вирішити наступні задачі:

- ознайомитись з особливостями побудови систем контролю доступу для підприємств;

- ознайомитись з принципами тестування на вміст алкоголю в диханні людини;

- провести розробку структури модуля контролю доступу та робочого часу;

- обрати електронні компоненти та елементи модуля;

- розробити конструкцію модулю;

- розробити програму керуючого мікроконтролеру модуля;

- розробити базу даних для зберігання даних щодо проходів на підприємства робітників;

- розробити програмне забезпечення обліку робочого часу та рівня алкоголю.

СИСТЕМА КОНТРОЛЮ ДОСТУПУ, ДАТЧИК АЛКОГОЛЮ,
МІКРОКОНТРОЛЕР, БАЗА ДАНИХ, МОДУЛЬ, ПЛАТА, КОНСТРУКЦІЯ,
ПРОГРАМА, ПРОТОКОЛ ОБМІНУ.

ЗМІСТ

Вступ.....	6
1. Огляд області розробки та постановка задач	8
1.1 Використання СКД на підприємствах	8
1.2 Огляд аналогів	9
1.3 Постановка задач.....	13
2. Розробка структури та конструкції модулю.....	16
2.1 Розробка структурної схеми	16
2.2 Вибір елементів	18
2.3 Розробка конструкції модулю.....	26
3. Написання програмного забезпечення.....	32
3.1 Розробка бази даних	32
3.2 Протокол обміну даними між апаратною частиною та ПК.....	45
3.3 Розробка програми мікроконтролера.....	48
3.4 Розробка програми для ПК	52
Висновки	58
Перелік літератури	59
Додаток А – Програма керування модулем.....	60
Додаток Б – Основний модуль програми для ПК.....	71
Додаток В – Презентація	77

ВСТУП

Нещасні випадки на виробництві часто відбуваються з вини нетверезих працівників. Людина в стані алкогольного сп'яніння ставить під загрозу своє життя і здоров'я. Також помилкові дії п'яних працівників можуть призвести до серйозних матеріальних втрат.

Традиційні методи алкотестування малоефективні і мають суттєві недоліки:

- медичний працівник, і співробітник служби безпеки - це люди, які можуть провести огляд неуважно або пропустити нетверезого співробітника цілком усвідомлено - з симпатії, з корисливих мотивів, просто «пошкодувати»;

- огляд кожного співробітника займає чимало часу, тому на великих підприємствах подібні методи незручні (призводять до появи заторів на прохідній, затримок початку робочої зміни, і т.д.).

Використання алкотестеру дозволить виключити помилки і суб'єктивність при оцінці стану алкогольного сп'яніння людини.

Система контролю доступу з алкотестером - це ефективна комплексна система безпеки об'єкта для вирішення проблеми виробничого травматизму з вини нетверезих співробітників. Інтеграція алкотестерів в СКУД - ідеальне рішення, що дозволяє повністю автоматизувати процес перевірки вхідної людини на предмет алкогольного сп'яніння, а також наявності прав доступу на об'єкт.

В кваліфікаційній роботі проводиться розробка модуля контролю робочого часу з алкотестером, що дозволить підприємствам контролювати час входу на підприємство та виходу з підприємства з додатковим обмеженням доступу працівників, що знаходяться в стані алкогольного оп'яніння.

Метою роботи є розробка апаратного модуля контролю доступу з алкотестером, написання керуючої програми мікроконтролера та програми

для фіксації часу проходів через прохідну та рівень алкоголю в диханні робітників.

Об'єкт розробки: система контролю доступом на підприємство.

Предмет розробки: електронний модуль та програмне забезпечення для контролю пропускового режиму на прохідних підприємства з можливістю контролю рівня алкоголю в диханні працівників.

1 ОГЛЯД ОБЛАСТІ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Використання СКД на підприємствах

Звичайна система контролю доступу вирішує завдання ідентифікації відвідувачів та обліку робочого часу персоналу. На багатьох підприємствах недостатньо фіксувати лише час приходу і відходу співробітників. Для допуску до робіт на високоточних верстатах, з дорогими виробами та інших робіт з високим рівнем відповідальності, необхідно контролювати працівників на стан алкогольного сп'яніння.

Наявність алкоголю в крові призводить до зниження критичного ставлення до власних дій, до поверхневого мислення, неточностей рухів і багатьох інших факторів, індивідуальним для кожної людини.

Алкотестери для систем контролю доступу - це особливий клас приладів, які ставляться на прохідній підприємства або організації і подають сигнал на дозвіл проходу, тільки якщо концентрація алкоголю в видиху знаходиться в межах норми ..

На відміну від інших алкотестерів, ці прилади працюють без участі оператора - або автономно, або під управлінням контролера СКУД.

З метою запобігання виходу на роботу співробітників в стані сп'яніння система контролю доступу (СКУД) може включати автоматизований контроль стану співробітників.

При виборі алкотестера для контролю стану співробітників при вході на територію підприємства, установи вимагає враховувати:

- кількість співробітників, що проходять через точку контролю;
- швидкість відновлення аналізатора;
- можливість і ступінь інтеграції з встановленою системою контролю доступу.

1.2 Огляд аналогів

Для постановки задач кваліфікаційної роботи було проведено огляд аналогів з метою визначення позитивних та негативних сторін їх роботи.



Рисунок 1.1 – Алкотестер АлкоФор s40

Спеціальний алкотестер АлкоФор s40 призначений для застосування на підприємствах з великою кількістю співробітників. Дана модель алкотестера має найбільшу пропускну спроможність в хвилину. Максимальна кількість вимірювань в хвилину до 12 тестів. Індикація результатів вимірювань відображається на світлодіодних індикаторах з лицьового боку приладу.

Характеристики алкотестера:

- тип сенсору – електрохімічний;
- діапазон вимірювань 0,00 - 4,00 "проміле"
- габаритні розміри 206x108x37;
- маса 674г;
- живлення – мережа 220В;
- час виходу в режим готовності – менше 30 сек;
- робоча температура 0 - 40 С
- внутрішня пам'ять – відсутня;

- підключення до ПК – відсутнє.

Недоліками алкотестера АлкоФор s40 є недостатня верхня межа вимірювань та відсутність можливості підключення до ПК



Рисунок 1.2 – Алкотестер Дінго В-02

Алкотестер для прохідної Дінго В-02 нова модифікація алкотестера застосовується в передрейсовому та передзмінному контролі, а також для контролю співробітників на вході на підприємство. Прилад оснащений високоточною електрохімічним сенсором, розрахованим на постійну роботу на великому потоці людей. Пластиковий корпус алкотестера Дінго В-02 захищає пристрій від вологи і пилу. Дінго в-02 може працювати як автономно, так і в складі системи контролю доступу. Різноманітні порти роблять можливим з'єднання приладу з іншими системами і пристроями. Прилад може підключатися до персонального комп'ютера через роз'єм USB (кабель поставляється в комплекті).

Технічні характеристики Алкотестер для прохідних Дінго В-02:

- тип датчика – електрохімічний;

- тип продування – через мундштук-воронку (можливо продування через соломку);

- час підготовки приладу – до 30 сек;
- очищення сенсора – до 60 сек;
- час до отримання результату тесту – до 10 сек;
- температура експлуатації від +5°C до + 40°C;
- температура зберігання від 0°C до + 50°C;
- додаткові опції лічильник тестів;
- міжкалібровочний інтервал 10 000 тестів або 6 місяців;
- пам'ять приладу – відсутня;
- USB-вихід і зовнішній індикатор;
- спосіб живлення – зовнішнє живлення 12V;
- габарити 108 x 97 x 206 мм;
- вага 674 г.

Обладнання, що використовується в типовій схемі:

- мережевий контролер доступу NC-8000;
- мережевий зчитувач PR-EH03;
- турнікет Praktika-t-01;
- алкотестер Дінго В-02;
- плата розширення NI-TW;
- плата інтеграції алкотестерів Ghost;
- базове ПО PNSoft-08;
- модуль обліку робочого часу PNSoftAR;
- настільний зчитувач PR-EH08.

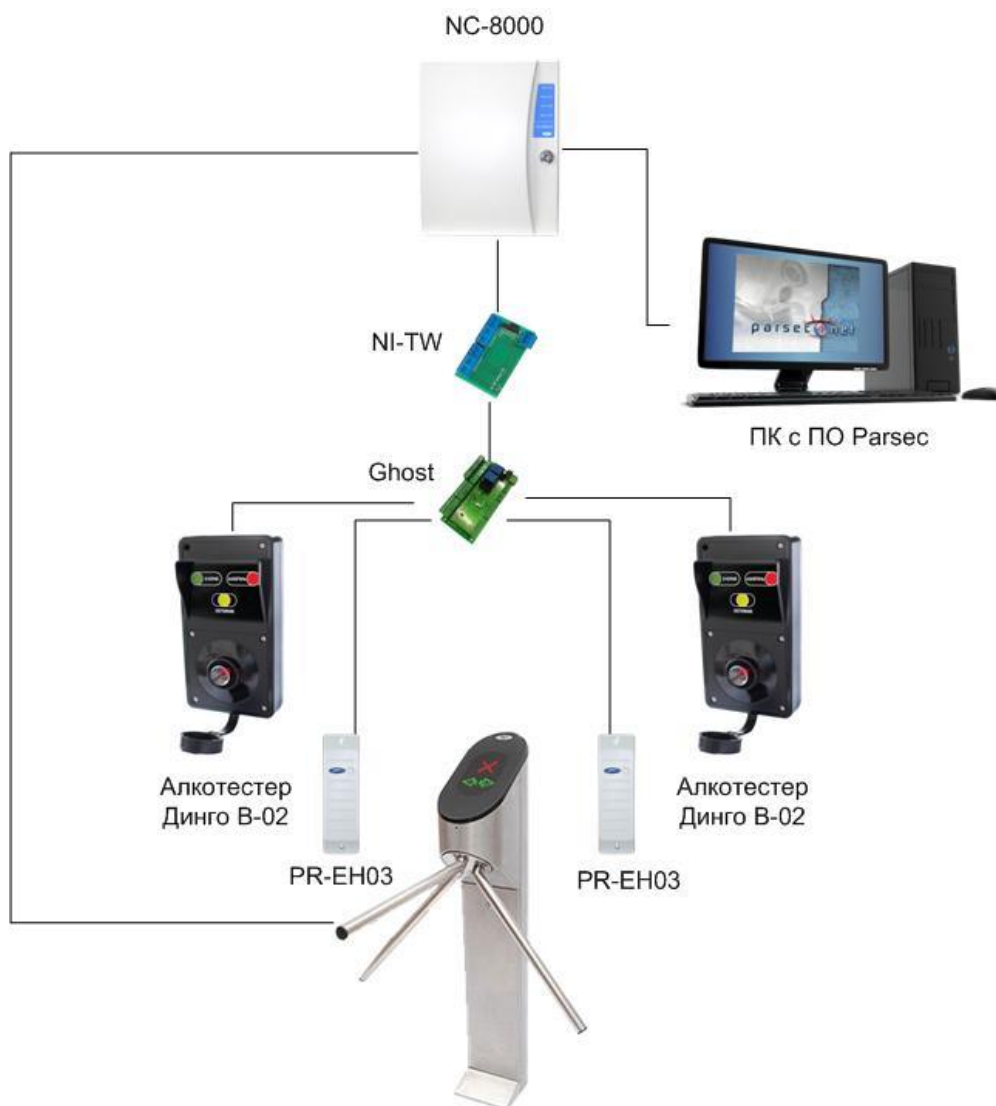


Рисунок 1.3 – Типова схема реалізації СКД з алкотестером на базі Дінго В-02

Механізм роботи:

- співробітник або гість прикладає картку до зчитувача, після цього "дує" в алкотестер;
- у систему Parsec потрапляє інформація про проміллях і дані користувача картки;
- система реагує по запрограмованому алгоритму: заборона / дозвіл проходу, sms або email повідомлення керівництва;
- дані зберігаються в журналі подій, можливе формування різних звітів .

Недоліком СКД з алкотестором Дінго В-02 є необхідність об'єднання елементів системи засобами дротового зв'язку та досить велика ціна впровадження за рахунок розгалуженої структури.

Розгляд аналогів та огляд області розробки дозволив розробити задачі проектування. Пропонується розробити модуль, що буде мати наступний функціонал:

- використання RFID міток у якості ідентифікації робітників при проході через прохідну;
- використання датчику алкоголю для фіксування рівня алкоголю в диханні робітника, що проходить через прохідну;
- відображення рівня алкоголю на дисплеї;
- зберігання часу проходку та рівня алкоголю в базі даних програми на ПК;

- звукова та світова індикація права проходу та відповідності рівня алкоголю в крові нормальному значенню;
- можливість керування турнікетом або електронним замком для регулювання доступу на основі даних отриманих з картки та рівня алкоголю;
- можливість перегляду звітів, щодо часу проходів через прохідну та рівня алкоголю за будь який період;
- з'єднання модуля з ПК засобами бездротового зв'язку;
- можливість відправлення звітів та окремих вимірювань (наприклад з перевищенням рівня алкоголю) на Telegram.

Принцип роботи системи

Працівник підійшовши до модулю підносить RFID картку або брелок, система фіксує ПІБ робітника якому належить картка/брелок або відбиток та на протязі 30 секунд зчитує рівень концентрації парів алкоголю, відображає на рідинно-кристалічному індикаторі отримане значення та реєструє у базі даних з висвітленням на комп'ютері. У випадку, якщо значення концентрації алкоголю перевищує критичне (встановлене в налаштуваннях системи) або картка не внесена до бази даних модуль видає звуковий попереджувальний сигнал та у програмі відповідне вимірювання підсвічується червоним кольором. Якщо користувач (картка або) є в системі та рівень алкоголю не перевищено – подається сигнал на реле, що може бути підключено до електронного замка або турнікета – таким чином працівник може пройти на територію або вийти з неї.

Таблиця 1.1 – Основні параметри системи

№	Назва параметру	Од.виміру	Значення
1	Напруга живлення	В	220±5%
2	Час вимірювання	с	30
3	Діапазон вимірювання рівня алкоголю	мг/л	0,05...10

Користувач системи має змогу проводити фільтрацію проведених вимірювань за часом, типом проходу (вхід на територію / вихід з території), по робітнику, по рівню алкоголю.

2 РОЗРОБКА СТРУКТУРИ ТА КОНСТРУКЦІЇ МОДУЛЮ

2.1 Розробка структурної схеми

Першим етапом розробки модуля була розробка структурної схеми (рис.2.1).

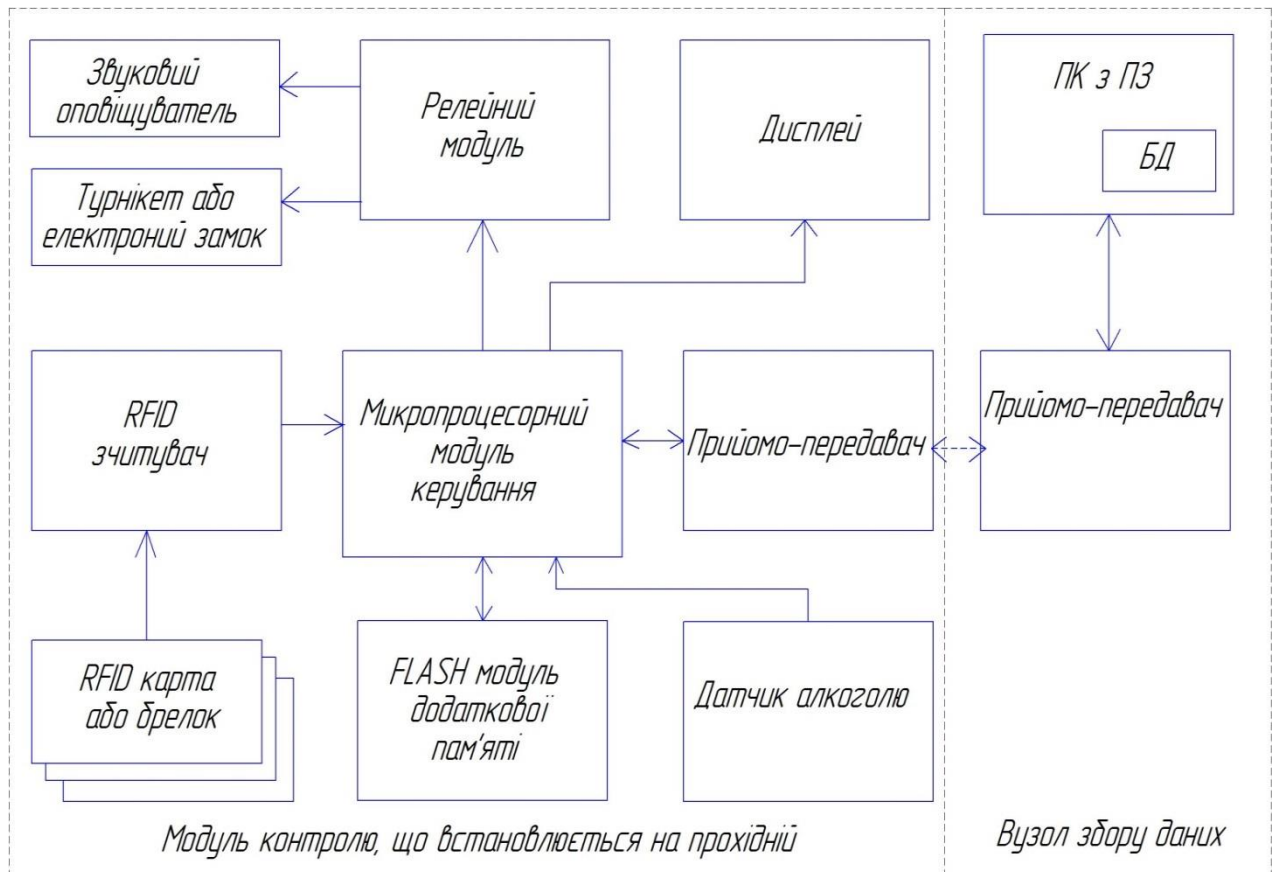


Рисунок 2.1 – Структурна схема системи контролю робочого часу з алкотестером

Система складається з наступних компонентів:

- модуль зчитування RFID карт та рівня алкоголю у крові з реле для керування електронних замком або турнікетом;
- набір RFID карт або RFID брелоків у кількості, що відповідає кількості працівників підприємства (підрозділу). Базовий комплект постачання включає 30 RFID карт.

- програмне забезпечення для реєстрації входів та виходів співробітників з фіксуванням рівня алкоголю.

Працівник підходить до модулю контролю часу на прохідній прикладує картку або брелок, зчитувач отримує код картки та на дисплеї з'являється ідентифікаційний номер картки або брелока. Ідентифікаційний номер картки передається через прийомом передавача по радіоканалу на радіо модуль, що з'єднаний з комп'ютером.

На комп'ютері встановлено програмне забезпечення з базою даних. Якщо картка є в базі програма отримує дані щодо працівника та записує час проведення вимірювання.

У працівника є 30 секунд для проведення заміру рівня алкоголю. Якщо за 30 секунд працівник не подув на датчик буде виведено повідомлення про це на дисплеї та замкнеться реле, що керує звуковим оповіщувателем і буде подано звуковий сигнал на протязі 2 секунд та через радіоканал інформація про це буде передана на комп'ютер, щоб повідомити охорону що працівник з якоїсь причини не проводив замір алкоголю.

Якщо працівник дув в область датчика та було отримано значення алкоголю – його буде виведено на дисплей та передано на комп'ютер. При перевищенні значення алкоголю (налаштовується в програмі) буде подано довгий сигнал через реле та звуковий оповіщувач (довжина сигналу також налаштовується в програмі).

Якщо ключ є в базі даних та рівень алкоголю не перевищує межовий спрацює реле яке відкриє електронний замок або турнікет та працівник зможе зайти або вийти з території.

В модулі передбачено додаткову пам'ять, що зберігає перелік ключів та до 200 вимірів щоб у випадку коли віддалений комп'ютер вимкнено можна було проводити заміри та забезпечувати пропускний режим.

Зчитувачі RFID працюють на частоті 13.56 МГц. Для доступу можуть використовуватись RFID карти або RFID брелоки за бажанням підприємства (або робітників).



Рисунок 2.2 – RFID карти та RFID брелоки

2.2 Вибір елементів

Після розробки структурної схеми було проведено вибір електронних компонентів для реалізації модулю.

У якості мікроконтролерного модулю було обрано модуль Arduino Nano – він має невеликий розмір та ціну при тому забезпечує необхідну кількість портів вводу-виводу та має достатню пам'ять програм, оперативну пам'ять та пам'ять даних.

Додатковою перевагою модулю є можливість його перепрограмування без застосування зовнішнього програматора – достатньо здійснити підключення через USB шнур до роз'єму комп'ютера. Середовище розробки програмного забезпечення безкоштовне та має велику кількість безкоштовних бібліотек для різних датчиків, пристроїв виводу. Мікроконтролер на базі якого побудований модуль має відкриту архітектуру, що дозволяє створювати власні бібліотеки модулів та вносити зміни в існуючі прискорюючи та оптимізуючи при цьому кінцевий програмний код пристрою.

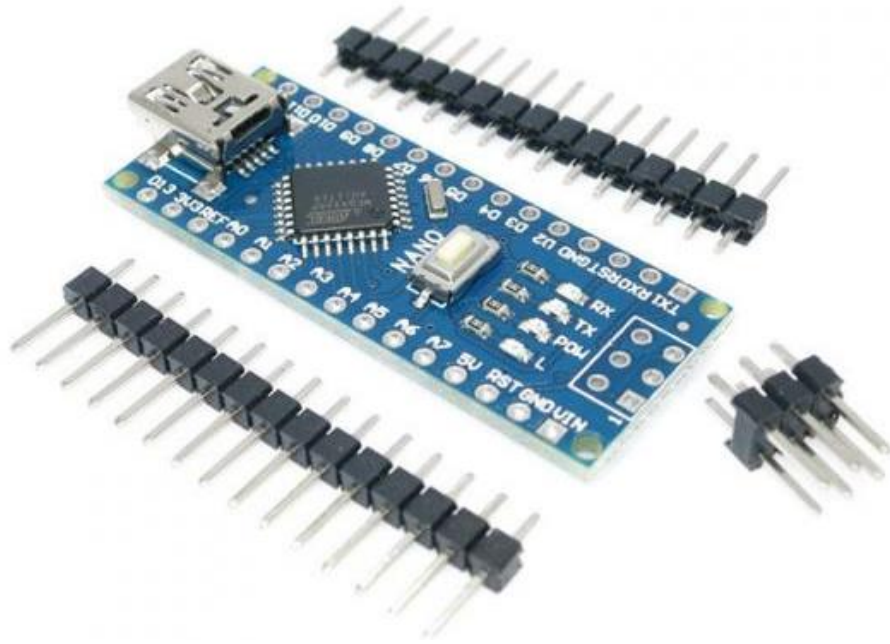


Рисунок 2.3 – Модуль Arduino Nano

Радіочастотна ідентифікація (RFID) - це технологія автоматичної безконтактної ідентифікації об'єктів за допомогою радіочастотного каналу зв'язку. Базова система RFID складається з:

- радіочастотної мітки;
- зчитувача інформації (рідера);
- комп'ютера або мікропроцесорного модулю для обробки інформації.

Ідентифікація об'єктів проводиться за унікальним цифровому коду, який зчитується з пам'яті електронної мітки, що прикріплюється до об'єкта ідентифікації. Зчитувач містить в своєму складі передавач і антену, за допомогою яких випромінюється електромагнітне поле певної частоти. Потрапили в зону дії пристрою, що зчитує поля радіочастотні мітки «відповідають» власним сигналом, що містить інформацію (ідентифікаційний номер товару, призначені для користувача дані і т.д.). Сигнал вловлюється антеною зчитувача, інформація розшифровується і передається в комп'ютер для обробки.

Справжній захист від копіювання та підробки забезпечують такі ідентифікатори, в чіпах яких реалізовано криптографічний захист. Це

безконтактні смарт-карти, що працюють на частоті 13,56 МГц, найбільш поширеними з них є карти Mifare®. У картах цих стандартів криптозащита організована на високому рівні, і підробка таких карт практично неможлива.

У якості зчитувача RFID обрано модуль RC522, що працює на частоті 13,56МГц (рис.2.4) у першу чергу за рахунок його роботи по SPI ынтерфейсу.



Рисунок 2.4 – Модуль зчитувача RC522

Модуль RC522 - RFID-модуль 13,56 МГц з SPI-інтерфейсом. Модуль RFID RC522 заснований на мікросхемі MFRC522 від NXP. Поставляється з двома мітками, RFID-карткою і брелоком з об'ємом пам'яті 1 КБ.

Модуль RFID RC522 створює електромагнітне поля з частотою 13,56 МГц, яке використовується для зв'язку з RFID мітками (стандартні мітки ISO 14443A). Для взаємодії з контролерами, модуль використовує 4-х контактний інтерфейс SPI. Так само, модуль підтримує протоколи зв'язку I2C і UART.

Додатково, виведений контакт переривання IRQ, якій дозволяє опитувати модуль тільки тога, коли приклали картку.

Основні характеристики:

- заснований на мікросхемі MFRC522;
- напруга живлення: 3,3 В;
- споживаний струм: 13-26 мА;
- робоча частота: 13,56 МГц;
- дальність зчитування: 0 ~ 60 мм;
- інтерфейс: SPI, максимальна швидкість передачі 10 Мбіт / с;
- розмір: 40 × 60 мм;
- читання і запис RFID-міток.

Одним з основних компонентів модулю є датчик алкоголю. У якості датчику алкоголю було обрано датчик MQ-3 (рис.2.5).



Рисунок 2.5 – Датчик алкоголю

Модуль MQ-3 являє собою датчик навколишнього середовища серії MQ, який призначений для виявлення етанолу і парів алкоголю в навколишньому середовищі. Так само датчик має невисоку чутливість до парам бензину. Модуль може використовуватися в проектах, де необхідно отримувати інформацію про знаходження шкідливих парів у навколишньому середовищі або видихуваному повітрі.

Плата датчика спроектована на основі газоаналізатора MQ-3 і компаратора LM393. Датчик MQ-3 являє собою нагрівальний елемент,

покритий спеціальним напиленням. При великій концентрації шкідливої речовини в датчику відбувається реакція, в результаті якої передається інформація про концентрацію газу. За допомогою компаратора LM393 відбувається перетворення імпульсів датчика в цифровий та аналоговий сигнали.

Підключається датчик до Мікроконтролерні пристрою через 4 виведення. Для подачі живлення напругою 5 В на модуль використовуються контакти VCC і GND. Передача даних здійснюється за допомогою аналогового (A0) або цифрового (B0) висновків.

На платі розміщений потенціометр, який необхідний для регулювання чутливості датчика до шкідливих речовин. Даний змінний резистор дозволяє підлаштовувати датчик до необхідних параметрів газу для конкретного проекту. Так само на платі є 2 світлодіода. Індикатор червоного кольору PWR-Led загоряється при подачі напруги на модуль, а світлодіод D0-Led сигналізує зеленої індикацією про перевищення або зменшення граничних значень шкідливих речовин.

Так як датчик володіє нагрівальним елементом, модуль споживає досить велику кількість енергії, тому рекомендується подавати напругу від зовнішнього джерела живлення. Перед першим використанням датчика необхідно залишити його включеним на кілька годин, завдяки цьому вихідні параметри MQ-3 стануть точнішими.

MQ-3 ставитися до напівпровідникових приладів. Принцип роботи датчика заснований на зміні опору тонкоплівкового шару діоксиду олова SnO_2 при контакті з молекулами визначається газу. Чутливий елемент датчика складається з керамічної трубки з покриттям Al_2O_3 і нанесеного на неї чутливого шару діоксиду олова. Всередині трубки проходить нагрівальний елемент, який нагріває чутливий шар до температури, при якій він починає реагувати на визначений газ. Чутливість до різних газів досягається варіюванням складу домішок в чутливому шарі.

Технічні характеристики модуля:

- напруга живлення, В – 5;
- струм, мА – 150;
- діапазон вимірювань, мг / л 0,05 ... 10;
- час прогріву при включенні, хв – 1;
- робоча температура, ° С -20 ... +50.

Для відображення даних про ідентифікаційний номер картки та рівень алкоголю обираємо дисплей LCD QC1602A.



Рисунок 2.6 – Дисплей QC1602A

LCD QC1602A дисплей для підключення до Arduino. Має два рядки по 16 символів в кожній. Працює зі стандартною бібліотекою LiquidCrystal з поставки Arduino IDE.

Характеристики:

- розміри 80 x 36 мм;
- робоча температура 0 ~ 50 ° С;
- підсвітка блакитна;
- колір символів білий;

- розмір символу 4.35 x 2.95мм;
- формат 16 x 2;
- розміри точки 0.5 x 0.5мм;
- інтерфейс HD44780;
- видима область 64.5 x 13.8мм;
- живлення 5В.

Для передавання даних від модуля до комп'ютеру було обрано передавачі NRF24L01 (рис. 2.7) з підсилювачем для збільшення відстані передавання.



Рисунок 2.7 – Радіомодуль NRF24L01

Передача сигналу відбувається на частоті 2,4 ГГц не вимагає дозволу. Швидкість передачі може бути налаштована, а максимальна = 256 KBytes / s.

Може працювати на одному з 126-та незалежних каналів, тому відразу кілька пристроїв можуть спілкуватися між собою не заважаючи один одному. Для передачі даних використовується власний протокол.

Додатково встановлено підсилювач потужності сигналу на мікросхемі RFX2401C і SMA гніздо для зовнішньої антени. Це дозволяє домогтися дальності прийому на відкритому просторі без перешкод до 1000 м.

Може бути застосований для зв'язку двох мікроконтролерів / arduino між собою. Підключення модуля до мікроконтролерного / мікропроцесорного пристрою відбувається по протоколу SPI. Зібраний на базі мікросхеми nRF24L01 +. У модулі встановлена вбудована антена.

Основні характеристики:

- напруга живлення: 1,9 ... 3,6 В;
- споживання струму:
 - робочий режим: 15 мА (залежить від швидкості передачі);
 - режим очікування: 26 мкА;
- швидкість передачі:
 - 31,25 KB / s;
 - 128 KB / s (1 Mbit / s);
 - 256 KB / s (2 Mbit / s);
- інтерфейс: SPI;
- радіус передачі: до 10 ... 100 м (залежить від перешкод).

Для приєднання радіомодуля на боці комп'ютера було обрано конвертер NRF-USB (рис.2.8).



Рисунок 2.8 – Перехідник "RF-USB"

Перехідник "RF-USB" створює на комп'ютері COM-порт, робота з яким забезпечує обмін даними через радіомодуль NRF24L01. Записані в COM-порт дані передаються в ефір, а прийняті з ефіру дані надходять в COM-порт.

Радіомодуль NRF24L01 проводиться в Китаї, легко доступний для придбання як в звичайних так і в інтернет-радіомагазинах. Робоча частота 2,4 ГГц.

Після установки віртуального COM-порту, перехідник не вимагає додаткових налаштувань і готовий до роботи зі швидкістю 9600 біт / с.

2.3 Розробка конструкції модулю

Наступним етапом проектування була розробка конструкції модуля. Розроблена у програмі SprintLayout друкована плата наведена на рис. 2.9.

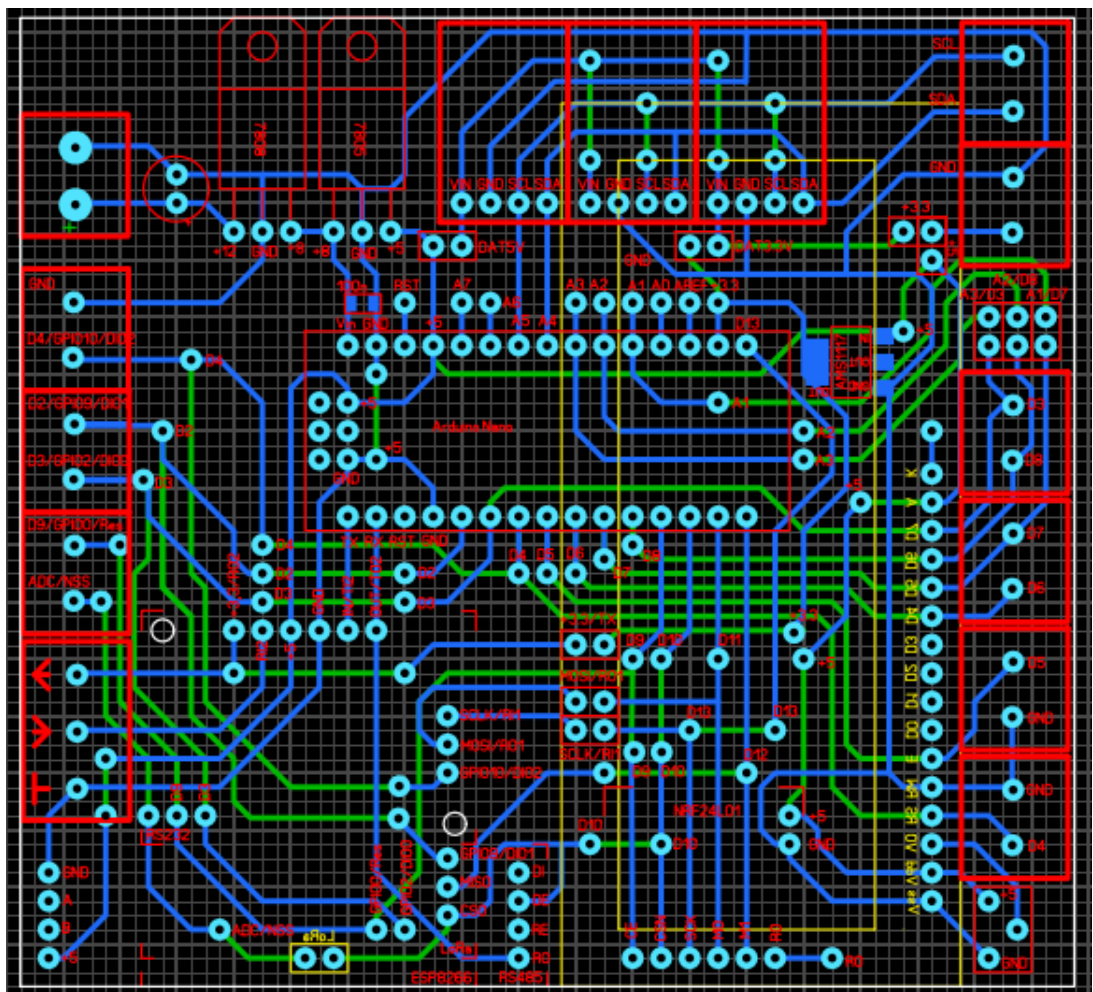


Рисунок 2.9 – Конструкція основної плати керування

Оскільки рідиннокристалічний індикатор приєднується на спеціально розведені контакти плати через роз'єм – для нього задіяні контакти наведені в табл.2.1. Катод підсвітки дисплею з'єднано з виводом A2 оскільки це дозволяє курувати під світкою програмно (блмати) для того, щоб звернути уваги користувача на якусь подію – наприклад перевищення вмісту алкоголю у крові або заборонений дамп карти доступу (чи не внесений в базу відбиток пальця).

Таблиця 2.1 – Підключення контактів дисплею

№	Контакт Arduino	Контакт дисплею (HD44780)
1	D4	RS
2	GND	RW
3	D5	E
4	D6	D4
5	D7	D5
6	D8	D6
7	D3	D7
8	+5	A
9	A2	K

Для приєднання RFID зчитувача використовуються з'єднання як наведено у таблиці 2.2. Контакти MOSI, MISO, SCK зчитувача відповідають контактам шини SPI Arduino. Контакти SDA, RST приєднані окремо для кожного зі зчитувачів, щоб можна було програмно вказувати з яким зі зчитувачів йде робота.

Оскільки контакти шини SPI плати керування виведені окремою контактною групою вниз плати – з'єднання MOSI, MISO, SCK та SDA, RST зчитувача на вході проводиться саме туди, а контакти SDA, RST зчитувача на виході проводиться до групи контактів A0, A1 над Arduino.

Таблиця 2.2 – Підключення контактів зчитувачів

№	Контакт Arduino	Контакт RC522 (Вхід)	Контакт RC522 (Вихід)
1	D11	MOSI	MOSI
2	D12	MISO	MISO
3	D13	SCK	SCK
4	+3.3	+3.3	+3.3
5	GND	GND	GND
6	D10	SDA	–
7	D9	RST	–
8	A0	–	SDA
9	A1	–	RST

Модуль встановлюється у пластиковий корпус, що виконано на основі електричної коробки з подальшим вирізанням отворів для дисплею та датчику алкоголю. Встановлення дисплею та датчику алкоголю виконуються у отвори в натяг та додатково закріплюється сіліконом. Зчитувач плоскою стороною заклеюється до кришку (рис.2.11).



Рисунок 2.11 – Кріплення елементів на кришку корпусу

Плата керування закріплюється в корпус за допомогою сіліконового клею. З'єднання з елементами відбувається за допомогою шлейфових з'єднань.

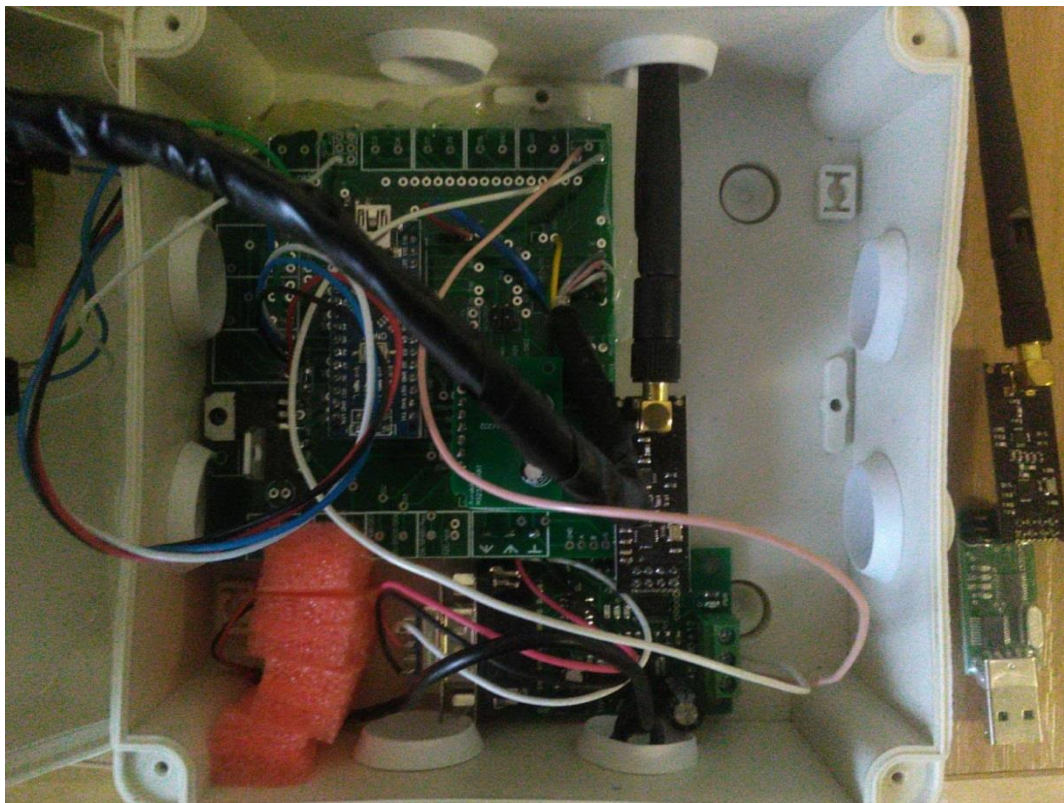


Рисунок 2.12 – Встановлення плати керування в корпус



Рисунок 2.13 – Модуль з закритою кришкою та модулем що вставляється в комп'ютер

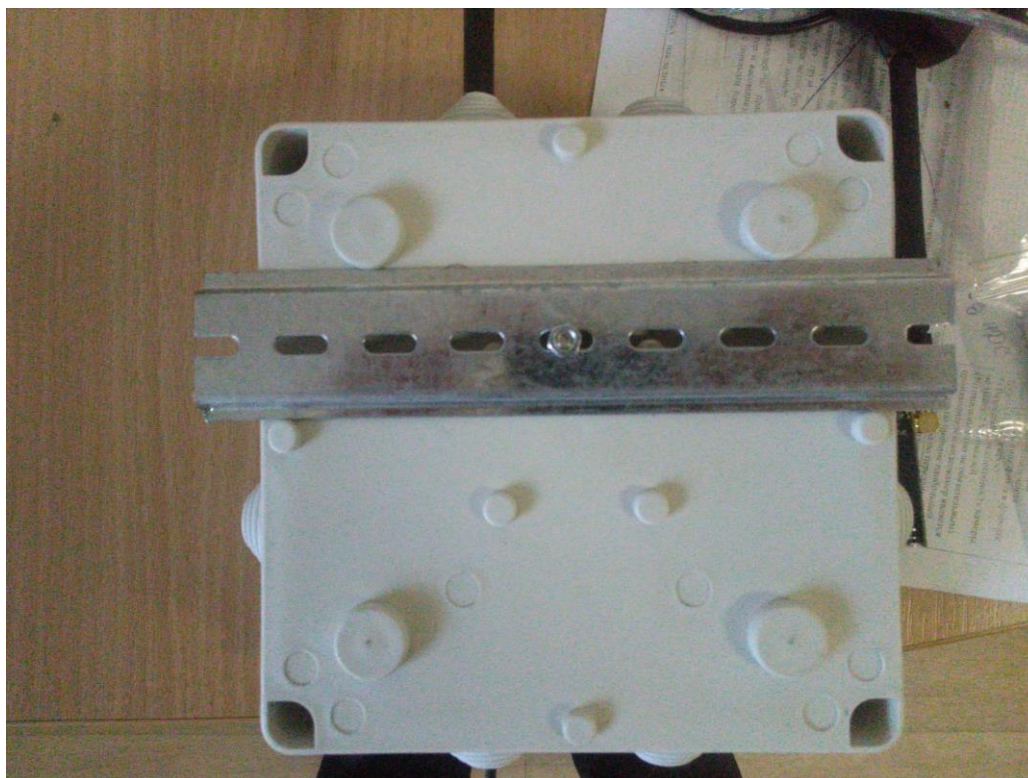


Рисунок 2.14 – Корпус модуля зі зворотної сторони з пластиною для кріплення до стіни

3 НАПИСАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка бази даних

Після розробки апаратного забезпечення ми перейшли до розробки програмного забезпечення. Для зберігання даних, щодо вимірювань та відповідності ключів доступу робітникам підприємства було створено базу даних в яку програма на комп'ютері буде зберігати отримані дані.

Таблиця 3.1 – Таблиця БД поточної версії програми Versions

№	Назва стовпця	Що зберігається	Тип даних	Приклад заповнення
1	Id	Номер запису (ключ таблиці)	INT	1
2	VNumber	Номер версії	INT	1
3	VDate	Дата створення версії	DATETIME	«2021-03-12 21:00:00»

Поточна таблиця використовується для контролю версії бази даних, яка зараз використовується програмою. При оновленні програми часто виникає ситуація коли треба проводити оновлення структури бази даних але при цьому зберегти записи, що були зроблені до цього. Програма при запуску перевіряє необхідність оновлення бази даних, оновлює її в разі необхідності та без втрат переносить записи до нової структури. Якщо подібна таблиця буде відсутня оновлена програма оновлена програма не зможе працювати з базою даних (оскільки та має стару структуру) або буде створюватися порожня база даних оновленої структури та відповідно попередньо збережені данні втрачатися, а це у більшості випадків неприпустимо. Програма при старті перевіряє запис з індексом 1 (це єдиний запис який завжди вказує на поточну версію бази даних та змінюється при її оновленні).

SQL запит для створення таблиці: CREATE TABLE Versions (Id INT, VNumber INT, VDate DATETIME, PRIMARY KEY (Id));

SQL запит для створення першого (та у випадку цієї таблиці єдиного) запису в таблицю: INSERT INTO Versions VALUES (1, 1, "2021-03-12 21:00:00");

Таблиця 3.2 – Таблиця описів Descriptions

№	Назва стовпця	Що зберігається	Тип даних	Приклад заповнення
1	Id	Номер запису (ключ таблиці)	INT	1
2	DescrText	Текст опису	TEXT	«керівник відділу охорони»

Поточна таблиця використовується для зменшення об'єму кінцевої бази даних. Тексти опису різних параметрів часто можуть повторюватись, а збереження текстової інформації у основних таблицях збільшує загальний обсяг та зменшує швидкість обробки запитів. Використання індексації описів дозволить використовувати в основних таблицях індекси текстів з посиланням та поточну таблицю. При створенні нової бази даних поточна таблиця залишається порожньою, а записи в неї додаються при необхідності при додаванні записів в основні таблиці.

SQL запит для створення таблиці: CREATE TABLE Descriptions (Id INT, DescrText TEXT, PRIMARY KEY (Id));

Наступною створена таблиця постів (табл. 3.3). Поточна таблиця використовується для зберігання назв постів контролю у випадках коли на одному підприємстві більше одного поста, а данні передаються на один комп'ютер з встановленою програмою керування та обліку. Це зручно коли треба переглядати потік робітників через різні пости охорони для отримання

загальної картини руху робітників через прохідні підприємства. В таблиці вказується внутрішня назва пункту пропуску, апаратний ідентифікатор модуля читання та додатковий опис посту. Назва та опис постів задається адміністратором довільно (як правило за внутрішньою класифікацією підприємства), апаратний ідентифікатор додається на основі наданої розробником інформації при постачанні обладнання.

Таблиця 3.3 – Таблиця постів охорони Posts

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id	Номер запису (ключ таблиці)	INT	—
2	PostID	Ідентифікатор апаратної частини	CHAR(5)	—
3	EquipType	Тип обладнання (0 – алкотестер, 1 – термометр)	INT	—
4	PostName	Назва посту (внутрішня на підприємстві)	VARCHAR(100)	—
5	Decsr	Додатковий опис посту	INT	Description(Id)
6	Activity	Стан запису (для запобігання повного видалення з БД: 1-запис активний, 0 – запис умовно видалений)	INT	

SQL запит для створення таблиці: CREATE TABLE Posts (Id INT, PostID CHAR(5), EquipType INT, PostName VARCHAR(100), Descr INT, Activity INT, PRIMARY KEY (Id), FOREIGN KEY (Descr) REFERENCES Descriptions(id));

Окремо було створено таблицю мов (табл.3.4). Поточна таблиця використовується для забезпечення можливості використання багатомовності інтерфейсу програми. Написи на елементах інтерфейсу програми, описи параметрів та ПІБ робітників та користувачів можуть бути надані на різних мовах це забезпечить можливість продажу програмного продукту у будь-яку країну світу. В таблиці зберігається скорочення мови згідно міжнародного стандарту ISO 639-3 та її назва на самій мові, щоб користувачеві було зручно знайти свою мову з наявного переліку.

Таблиця 3.4 – Таблиця мов Languages

№	Назва стовпця	Що зберігається	Тип даних	Приклад заповнення
1	Id	Номер запису (ключ таблиці)	INT	1
2	LangName	Скорочення мови в міжнародній класифікації ISO 639-3	VARCHAR(5)	“ukr”
3	FullLangName	Назва мови на ній самій	VARCHAR(50)	“українська”

SQL запит для створення таблиці: CREATE TABLE Languages (Id INT, LangName VARCHAR(5), FullLangName VARCHAR(50), PRIMARY KEY (Id));

SQL запит для створення української мови: INSERT INTO Languages VALUES (1, "ukr", "українська");

SQL запит для створення російської мови: INSERT INTO Languages VALUES (2, "rus", "русский");

Для того щоб максимально про індексувати основну таблицю вимірювань було створено таблиці прізвищ, імен та по батькові (табл.3.5-3.7). Поточна таблиця призначена для зберігання текстового написання прізвищ. Вона, як і таблиця описів (Descriptions), покликана зменшити обсяг кінцевої

бази даних та проіндексувати її для підвищення швидкості обробки SQL запитів.

Таблиця 3.5 – Таблиця текстів прізвищ SurNames

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id	Номер запису (ключ таблиці)	INT	–
2	MName	Текстове написання прізвища	VARCHAR(50)	–
3	Lang	Посилання на мову на якій прізвище написано	INT	Languages(Id)

На підприємствах з великою кількістю робітників часто зустрічається більш ніж один працівник з одноковим прізвищем. Крім того, таблиця зберігає посилання на мову на якій це прізвище написано – зручно при необхідності перекладання даних БД на іншу мову – достатньо лише провести переклад цієї таблиці. Або, навіть, створити копії записів з перекладом на додаткову мову, що надасть можливості швидкого перекладу даних у звітності. При створенні таблиця порожня, а записи з’являються при додаванні користувачів або робітників. Крім того, наявність даної таблиці дозволяє створити програмний механізм підказування правильного та швидкого написання прізвища при пошуку.

SQL запит для створення таблиці: CREATE TABLE SurNames (Id INT, MName VARCHAR(50), Lang INT, PRIMARY KEY (Id), FOREIGN KEY (Lang) REFERENCES Languages(Id));

Таблиця текстів імен призначена для зберігання текстового написання імен. Призначення її те саме, що й у таблиці прізвищ (SurNames). На додачу треба сказати, що імена людей співпадають ще частіше ніж прізвища, а

реалізація підказки правильного написання при пошуку прискорить роботу з програмою.

SQL запит для створення таблиці: CREATE TABLE MNames (Id INT, MName VARCHAR(50), Lang INT, PRIMARY KEY (Id), FOREIGN KEY (Lang) REFERENCES Languages(Id));

Таблиця 3.6 – Таблиця текстів імен MNames

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id	Номер запису (ключ таблиці)	INT	–
2	MName	Текстове написання імені	VARCHAR(50)	–
3	Lang	Посилання на мову на якій ім'я написано	INT	Languages(Id)

Таблиця текстів по батькові призначена для зберігання текстового написання по батькові. Призначення її те саме, що й у таблиць прізвищ (SurNames) та імен (Names).

Таблиця 3.7 – Таблиця текстів по батькові FNames

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id	Номер запису (ключ таблиці)	INT	–
2	MName	Текстове написання по батькові	VARCHAR(50)	–
3	Lang	Посилання на мову на якій по батькові написано	INT	Languages(Id)

SQL запит для створення таблиці: CREATE TABLE FNames (Id INT, MName VARCHAR(50), Lang INT, PRIMARY KEY (Id), FOREIGN KEY (Lang) REFERENCES Languages(Id));

Таблиця 3.8 – Таблиця користувачів Users

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id	Номер запису (ключ таблиці)	INT	–
2	UserLogin	Login користувача	VARCHAR(50)	–
3	MD5Hash	MD5 хеш паролю	CHAR(32)	–
4	AccLvl	Рівень доступу (0 – розробник, 1- адміністратор/власник підприємства, 2 – оператор, 3 – користувач (робітник охорони))	INT	–
5	UserName	Ім'я користувача (посилання)	INT	MNames(Id)
6	UserFName	По батькові користувача (посилання)	INT	FNames(Id)
7	UserSName	Прізвище користувача (посилання)	INT	SurNames(Id)
8	Descr	Посилання на додатковий опис (наприклад посада)	INT	Description(Id)
9	Activity	Стан запису (1-запис активний, 0 – запис умовно видалений)	INT	–

Таблиця користувачів (табл..3.8) використовується для ідентифікації користувачів при вході в програму з метою недопускання стороннього доступу, розподілу прав доступу в залежності від типу користувача («розробник», «адміністратор», «оператор», «користувач») та обліку дії, виконаних з програмою, у випадках недостовірної інформації чи неправильного використання програми. В таблиці зберігаються ПІБ користувача, логін користувача, його рівень доступу та пароль у вигляді хешу (для запобігання прямого зчитування при отриманні доступу напряму до таблиць бази даних). Використовується хеш с алгоритмом MD5 (найпростіший алгоритм хешування, оскільки нема потреби сильного захисту від адміністратора системи, який і так має можливості зміни паролів).

Як вже було описано, розроблено 4 рівні доступу:

- користувач (найнижчий рівень) – може лише переглядати поточні вимірювання за поточну добу;
- оператор – крім можливостей користувача може додавати нові картки працівників (додавання але не зміна та не видалення записів з довідника працівників) та переглядати вимірювання за будь-який період (ймовірніше це начальник охорони підприємства);
- адміністратор – крім можливостей оператора може видаляти та змінювати картки працівників (усі зміни та видалення зберігаються у базі даних), змінювати налаштування з'єднань апаратної частини, проводити калібрування датчиків, змінювати налаштування передавання даних;
- розробник (найвищий рівень) – крім можливостей адміністратора може переглядати протоколу обміну з апаратною та серверною частиною для виявлення несправностей у випадку їх виникнення.

Для кінцевого підприємства доступні лише перші 3 рівня доступу оскільки потреби у відлагодженні апаратної та серверної частини можуть виникати лише розробника.

SQL запит для створення таблиці: CREATE TABLE Users (Id INT, UserLogin VARCHAR(50), MD5Hash CHAR(32), AccLvl INT, UserName INT, UserFName INT, UserSName INT, Descr INT, Activity INT, PRIMARY KEY (Id), FOREIGN KEY (UserName) REFERENCES MNames(id), FOREIGN KEY (UserFName) REFERENCES FNames(id), FOREIGN KEY (UserSName) REFERENCES SurNames (id), FOREIGN KEY (Descr) REFERENCES Descriptions(id));

SQL запити для створення первинних користувачів та паролів (користувача з кожним рівнем доступу по одному) при створенні нової бази даних:

- створення користувача з рівнем «розробник» з логіном «Avrobit» та паролем «avrobit» INSERT INTO Users VALUES (1, "Avrobit" , "7380665E6A61AD97EC6AC4D11F2BEF11", 0, NULL, NULL, NULL, NULL,1);

- створення користувача з рівнем «адміністратор» з логіном «admin» та паролем «admin» INSERT INTO Users VALUES (2, "admin" , "21232F297A57A5A743894A0E4A801FC3", 0, NULL, NULL, NULL, NULL,1);

- створення користувача з рівнем «оператор» з логіном «operator» та паролем «operator» INSERT INTO Users VALUES (3, "operator", "4B583376B2767B923C3E1DA60D10DE59", 0, NULL, NULL, NULL, NULL,1);

- створення користувача з рівнем «користувач» з логіном «user» та паролем «user» INSERT INTO Users VALUES (4, "user", "EE11CBB19052E40B07AAC0CA060C23EE", 0, NULL, NULL, NULL, NULL,1);

Наступною таблицею є таблиця робітників Workers, що зберігає дані про працівників підприємства, що безпосередньо є учасниками контролю на прохідній та яким вибаються картки доступу. Оскільки попередньо створено таблиці індексації імен – ПІБ працівників вказуються у вигляді посилань на ці таблиці.

Таблиця 3.9 – Таблиця працівників Workers

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id	Номер запису (ключ таблиці)	INT	–
2	WName	Ім'я робітника (посилання)	INT	Names(Id)
3	WName	По батькові робітника (посилання)	INT	FNames(Id)
4	WSName	Прізвище робітника (посилання)	INT	SurNames(Id)
5	Descr	Додатковий опис робітника	INT	Description(Id)
6	Activity	Стан запису (для запобігання повного видалення з БД: 1- запис активний, 0 – запис умовно видалений)	INT	–

SQL запит для створення таблиці: CREATE TABLE Workers (Id INT, WName INT, WName INT, WSName INT, Descr INT, Activity INT, PRIMARY KEY (Id), FOREIGN KEY (WName) REFERENCES MNames(id), FOREIGN KEY (WName) REFERENCES FNames(id), FOREIGN KEY (WSName) REFERENCES SurNames (id), FOREIGN KEY (Descr) REFERENCES Descriptions(id));

Розмір UID RFID 13,56МГц, дамп (код збереженого зображення) відбитка пальця з модуля AS608 та UID NFC мітки смартфона мають однакову довжину – 4 байти, що зберігається в типі даних INT SQL структури.

SQL запит для створення таблиці: CREATE TABLE Accesses (Id INT, IdentityType INT, UID INT, PRIMARY KEY (Id));

Таблиця 3.10 – Таблиця методів ідентифікації (RFID карток/відбитків пальців/NFC міток) Accesses

№	Назва стовпця	Що зберігається	Тип даних	Приклад заповнення
1	Id	Номер запису (ключ таблиці)	INT	1
2	IdentityType	Тип ідентифікації (0 – RFID картка, 1 – відбиток пальця, 2 – NFC мітка)	INT	0
3	UID	UID карти/відбитка/NFC	VARCHAR(10)	“4157163874”

Таблиця 3.11 – Таблиця відповідності методів ідентифікації працівнику AccWrks

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id	Номер запису (ключ таблиці)	INT	–
2	WId	Посилання на робітника	INT	Workers(Id)
3	AId	Посилання на метод доступу	INT	Accesses(Id)
	Activity	Стан запису (для запобігання повного видалення з БД: 1- запис активний, 0 – запис умовно видалений)	INT	–

SQL запит для створення таблиці: CREATE TABLE AccWrks (Id INT, WId INT, AId INT, Activity INT, PRIMARY KEY (Id), FOREIGN KEY (WId) REFERENCES Workers(Id), FOREIGN KEY (AId) REFERENCES Accesses(Id));

Таблиця 3.12 – Таблиця вимірювань Measures

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id	Номер запису (ключ таблиці)	INT	–
2	MDate	Дата вимірювання	DATE	
3	MTime	Час вимірювання	TIME	
4	EuipId	Посилання на обладнання (пост) яким було зчитане значення	INT	Posts(Id)
5	MeasValue	Зчитане з датчика значення параметру	FLOAT	
6	MeasType	Тип параметру який вимірюється (0 – алкоголь, 1 – температура, 2 – відстань)	INT	
7	Worker	Ссылка на работника которому производится измерение	INT	Workers(Id)
8	Activity	Стан запису (для запобігання повного видалення з БД: 1-запис активний, 0 – запис умовно видалений)	INT	–
9	Descr	Дополнительное описание	INT	Descriptions(Id)

SQL запит для створення таблиці: CREATE TABLE Measures (Id INT, MDate DATE, MTime TIME, EuipId INT, MeasValue FLOAT, MeasType INT, Worker INT, Activity INT, Descr INT, PRIMARY KEY (Id), FOREIGN KEY (EuipId) REFERENCES Posts(Id), FOREIGN KEY (Worker) REFERENCES Workers(Id), FOREIGN KEY (Descr) REFERENCES Descriptions(Id));

Таблиця 3.13 – Таблиця дій користувача Events

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id	Номер запису (ключ таблиці)	INT	–
2	EvType	Тип дії (0 – вхід в програму, 1 – вихід з програми, 2 – зміна користувача)	INT	
3	EvDate	Дата дії	DATE	
4	EvTime	Час дії	TIME	
5	EvUser	Посилання на користувача	INT	Users(Id)
6	Descr	Посилання на додатковий опис (в текстовому вигляді сама дія для журналу)	INT	Description(Id)

SQL запит для створення таблиці: CREATE TABLE Events (Id INT, EvType INT, EvDate DATE, EvTime TIME, EvUser INT, Descr INT, PRIMARY KEY (Id), FOREIGN KEY (EvUser) REFERENCES Users(Id), FOREIGN KEY (Descr) REFERENCES Descriptions(id));

Таблиця 3.14 – Таблиця написів Strings

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id	Номер запису (ключ таблиці)	INT	–
2	ElementName	Ім'я елемента	VARCHAR(150)	
3	Lang	Мова напису	INT	Languages(Id)
4	Inscr	Напис (посилання на таблицю описів)	INT	Description(Id)

SQL запит для створення таблиці: CREATE TABLE Strings (Id INT, ElementName VARCHAR(150), Lang INT, Inscr INT, PRIMARY KEY (Id), FOREIGN KEY (Lang) REFERENCES Languages(Id), FOREIGN KEY (Inscr) REFERENCES Descriptions(Id));

3.2 Протокол обміну даними між апаратною частиною та ПК

Після розробки бази даних було розроблено протокол обміну між елементами системи.

Модуль, що встановлюється на прохідну передає значення, що характеризують стан модулю (зчитувачів, датчику алкоголю), а також ідентифікаційні номери картки та виміряні рівні алкоголю.

Таблиця 3.15 – Операції, що ініціюються ПК (кінець передавання даних байт 0x03)

№	Операція	Команда від ПК	Відповідь Arduino	Змінна в Arduino
1	Встановити рівень нормального дихання (Вхід)	SetZIn=XXX	AlcZeroIn=XXX	AlcZero[0] (0x02 int 2 байта)
2	Встановити рівень нормального дихання (Вихід)	SetZOut=XXX	AlcZeroOut=XXX	AlcZero[1] (0x04 int 2 байта)
3	Встановити крок на 0,05 проміле (Вхід)	SetAStepIn=X XX	AlcStepIn=XXX	AlcStep[0] (0x06 float 4 байта)
	Встановити крок на 0,05 проміле (Вихід)	SetAStepOut=XXX	AlcStepOut=XXX	AlcStep[1] (0x16 float 4 байта)
4	Ініціювати калібрування	StartCal		—

Продовження таблиці 3.15

№	Операція	Команда від ПК	Відповідь Arduino	Змінна в Arduino
5	Отримати поточні значення калібрування	GetCal	<XX#XX% XXX\$XXX &XXX+XX X>	–
6	Встановити час звукового оповіщення про перевищення рівня алкоголю (в секундах)	SetBTm=XXX	BuzzerTime =XXX	BuzzerTime (0x01 byte)
7	Встановити час очікування дихання після прикладування карти (в секундах)	SetWTm=XX X	MTm=XXX	MTm (0x00 byte)
8	Відкрити замок	LockON	Lock Opened	
9	Закрити замок	LockOFF	Lock Closed	
10	Встановити напис на дисплеї	Display1="XX XX" Display2="XX XX"		
11	Вмикнути оповіщувач на встановлений в налаштуваннях час	BuzzerON	Buzzer ON	
12	Вимкнути оповіщувач	BuzzerOFF	Buzzer OFF	
13	Зчитати ідентифікатор обладнання	GetID	IDXXX	EquipID (0x0A) (String 5 байт)

Продовження таблиці 3.15

№	Операція	Команда від ПК	Відповідь Arduino	Змінна в Arduino
14	Встановити ідентифікатор обладнання	SetID=XXXXX	EquipID=XX XXX	EquipID (0x0A) (String 5 байт)
15	Отримати поточні покази датчика алкоголю (1 – вхід, 2 – вихід)	GetAlcoX	AlcoX:YYY /ZZZ (YYY показання АЦП, ZZZ – розраховані платою промілле)	
16	Отримати останній номер зчитаної карти (1 – вхід, 2 – вихід)	GetLastCardX		
17	Перевірити зв'язок з платою	TestConnect	Connect OK	
18	Встановити/Скинути необхідність перевірки відповіді ПК	SetPCTest ClearPCTest	TestSet TestClear	NeedPCTest (0x20 byte)

В табл.3.15 вказані команди які надсилає комп'ютер, тобто вини є асинхронними для модуля та основною метою їх є налаштування параметрів модулю (включаючи калібрування датчика) та керування виконуючими механізмами.

У табл.3.16 наведені команди, що ініціюються модулем. Ці події є асинхронними для комп'ютера і відбуваються коли виникає якась технічна подія пов'язана з модулем або робота з працівником, що проходить через прохідну.

Таблиця 3.16 – Операції, що ініціюються модулем

№	Операція	Команда від Arduino	Відповідь ПК
1	Плата запущена	Module Started	MSOK
2	Відсутність з'єднання зі зчитувачем входу	No RFID IN	BuzzerON
3	Відсутність з'єднання з зчитувачем виходу	No RFID OUT	BuzzerON
4	Отримання номеру картки від плати	RFID1:XXXXX	RFID OK
5	Отримання рівня алкоголю від плати	AlcoX:YYY/ZZZ	Alco OK

3.3 Розробка програми мікроконтролера

Наступним етапом було написання тексту програми керуючого мікроконтролера.

Оскільки у якості мікропроцесорного модуля було використано Arduino Nano програма була написана у середовищі Arduino IDE на мові програмування C.

Спочатку були підключені бібліотеки:

- бібліотека по роботі з шиною I2C `#include <Wire.h>` ;
- бібліотека по роботі з дисплеєм `#include <LiquidCrystal_I2C.h>`;
- бібліотека по роботі з ППЗП `#include <avr/eeprom.h>` ;
- бібліотека по роботі зі сторожовим таймером `#include <avr/wdt.h>`;
- бібліотека по роботі з шиною SPI `#include <SPI.h>`;
- бібліотека для роботи зі зчитувачем RFID міток `#include <MFRC522.h>`

Далі було проведено ініціалізацію виводів мікроконтролеру їх використанню.

```
#define AlcoPin1 17 //A3 аналоговий вивод першого датчика алкоголю
(вхід - без дисплею)

#define AlcoPin2 16 //A2 аналоговий вивод другого датчика алкоголю
(вихід - з дисплеєм)

#define RST1_PIN 7//RST контакт 1го зчитувача (вхід - без дисплею)
#define SS1_PIN 8//SDA контакт 1го зчитувача (вхід - без дисплею)
#define RST2_PIN 9 //RST контакт 2го зчитувача (вихід - з дисплеєм)
#define SS2_PIN 10 //SDA контакт 2го считывателя (вихід - з дисплеєм)
#define Buzzer 4 //контакт реле оповіщувача перевищення рівня
#define Lock 5 //контакт реле електронного замка або турнікету
#define HLImpRead 6 //світлодіод коробки входу - попередження що
```

треба дути

Далі було створено об'єкти – ініціалізація

LiquidCrystal_I2C lcd(0x27, 16, 2); – ініціалізація дисплею з назвою lcd з адресою 0x27 по шині I2C.

MFRC522 RFID_IN(SS1_PIN, RST1_PIN); – створення об'єкту зчитувача входу.

MFRC522 RFID_OUT(SS2_PIN, RST2_PIN); – створення об'єкту зчитувача виходу.

Наступним етапом було створення змінних.

int TIME_UNTIL_WARMUP = 900; //час нагріву датчика алкоголю для входу в робочий режим.

int AlcoLevel[2]={0,0}; //масив виходів АЦП датчиків алкоголю (вхід [0], вихід [1])

byte CardNumb=0; //номер зчитувача до якого прикладено картку (0 - не прикладена; 1 – до зчитувача входу; 2 - до зчитувача виходу).

unsigned long uidDec, uidDecTemp; // для зберігання номеру мітки в десятичному форматі.

unsigned long LastuidDec[2]={0,0}; //останні зчитанні ІД карток з входу та виходу (вхід [0], вихід [1]).

unsigned long StartSRead,StopSRead;//засікаємо час початку та кінця вимірювання алкоголю.

byte MTm=15;//час (в секундах) вимірювання концентрації алкоголю після прикладування картки.

byte BuzzerTime=10;//час (в секундах) включення оповіщувача при перевищенні рівня алкоголю.

unsigned int AlcZero[2]={95,140};//максимальний рівень показів АЦП для нормального дихання (вхід [0], вихід [1]).

float ro[2]={6.3,6.3}; //опір датчика при чистому повітрі (вхід [0], вихід [1]).

float AlcStep[2]={10,10};//крок показань АЦП на 0,05 проміле (вхід [0], вихід [1]).

bool NeedPCTest=false; //необхідність перевіряти відповідь ПК.

char EquipID[6]="ID001";//ідентифікаційний номер обладнання Первинні налаштування робимо в блоці setup.

Налаштування напрямків роботи виводів портів:

pinMode(Buzzer,OUTPUT);

pinMode(Lock, OUTPUT);

pinMode(HLInpRead, OUTPUT);

Читаємо поточні налаштування в ППЗП: ReadSetings();

Запуск серійного порта: Serial.begin(9600);

SPI.begin(); – ініціалізація шини SPI;

RFID_IN.PCD_Init(); – ініціалізація зчитувача входу;

RFID_OUT.PCD_Init(); – ініціалізація зчитувача виходу;

Запуск дисплею и вивід вітального напису

lcd.init(); – ініціалізуємо дисплей;

lcd.backlight(); – включаємо підсвітку;

lcd.setCursor(0, 0);// Установлюємо курсор в начало 1 строки

lcd.print("Module Stated"); // Выводим текст

Serial.println("Module Started");

```

delay(500);
BuzzerOFF;
LockOFF;
digitalWrite(HLInpRead, LOW); //тушим світлодіод
AlcoLevel[0]=ReadAlcoADC(1);
AlcoLevel[1]=ReadAlcoADC(2);

```

Перевіряємо чи надійшла команда від комп'ютера:

```
TestPC();
```

Поки рівень на датчиках більший ніж встановлений нульовий рівень пишемо на дисплеї, що йде нагрів та в циклі читаємо показання датчиків.

```

while ((AlcoLevel[0]>AlcZero[0])||(AlcoLevel[1]>AlcZero[1])) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Warming");
    lcd.setCursor(8, 0);
    lcd.print(AlcoLevel[0]);
    lcd.setCursor(12, 0);
    lcd.print(AlcoLevel[1]);
    lcd.setCursor(0, 1);
    lcd.print("Please wait!");
}

```

Читаємо рівень нуля виходу АЦП для датчиків алкоголю:

```

    AlcoLevel[0]=ReadAlcoADC(1);
    AlcoLevel[1]=ReadAlcoADC(2);
}

```

Розрахунок первинного опору для нульового рівня датчиків:

```

    ro[0]=calculateResistance(AlcZero[0])/ 60.0;
    ro[1]=calculateResistance(AlcZero[1])/ 60.0;
}

```

Повний текст програми мікроконтролерного модуля наведено у додатку А.

3.4 Розробка програми для ПК

Після написання програми керуючого мікроконтролера була написана програма для ПК. Для написання програми використовувалося середовище розробки Delphi.

В програмі було реалізовано 7 форм користувача та 18 модулів – перелік форм наведено на рис.3.1, а перелік модулів на рис.3.2.

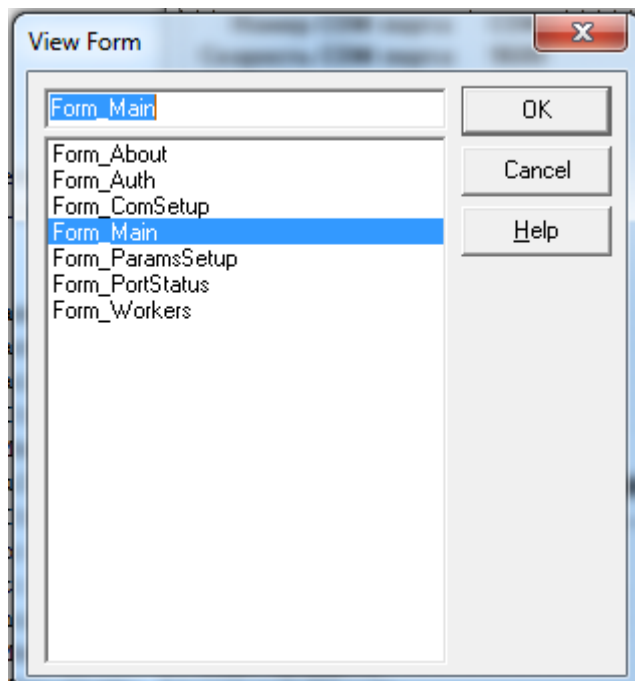


Рисунок 3.1 – Структура форм програми.

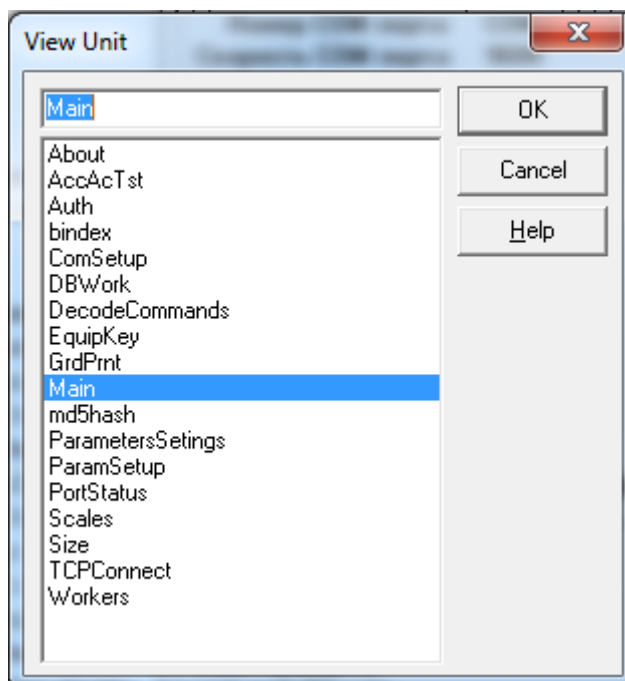


Рисунок 3.2 – Структура модулів програми

При запуску програми з'являється форма входу як наведено на рис.3.3. Існує можливість додавати будь яку кількість користувачів з власними паролями, при цьому користувачі мають 3 рівні доступу:

- адміністратор;
- оператор;
- охоронець.

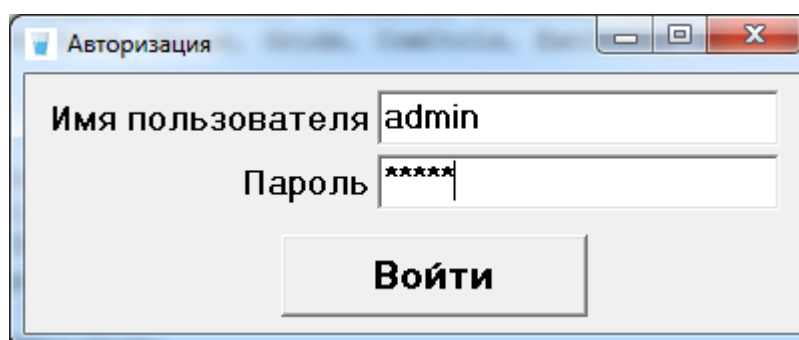


Рисунок 3.3 – Форма входу в програму

Після правильного вводу паролю запускається основне робоче вікно в якому вказані поточні налаштування та відображається таблиця входів та вимірювань рівня алкоголю за поточну добу (рис.3.4).

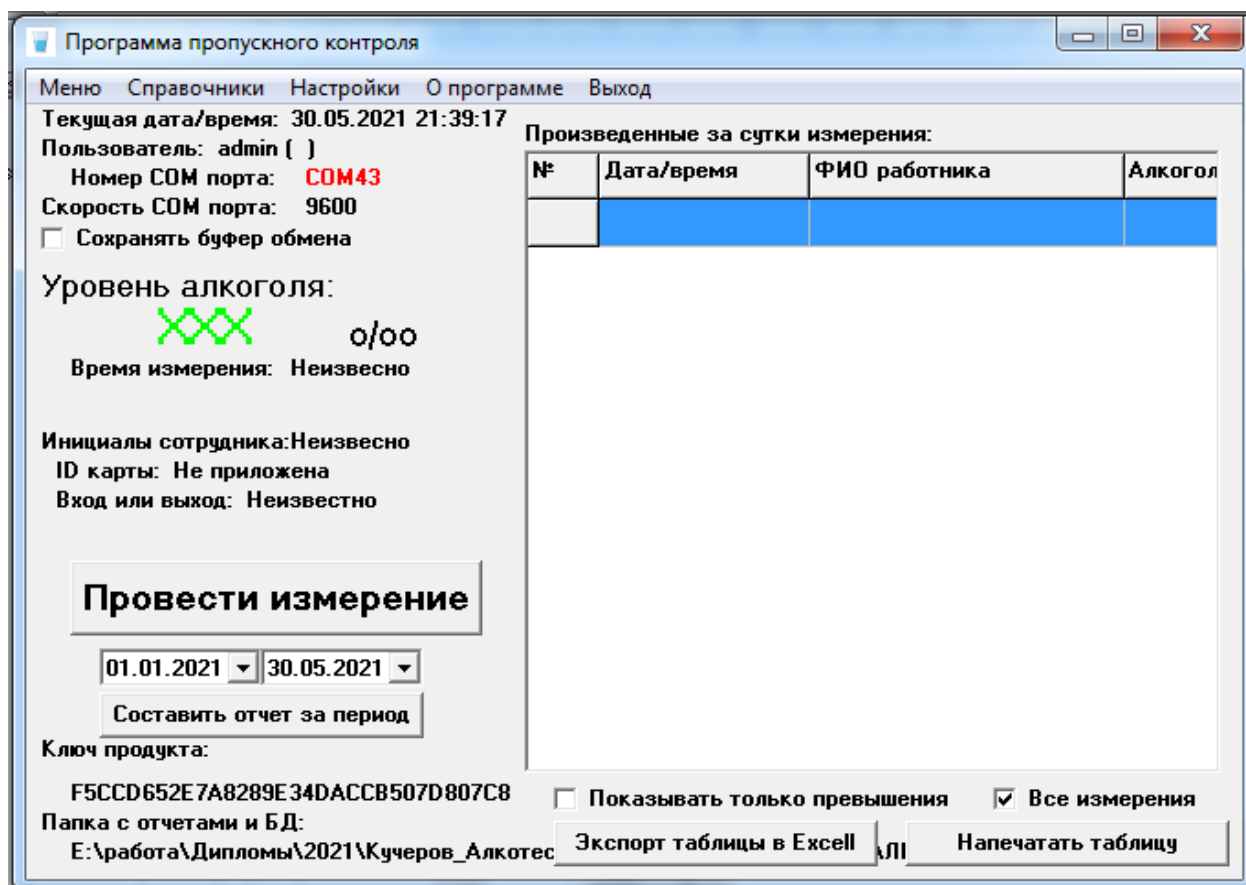


Рисунок 3.4 – Головне вікно програми

Для того щоб внести картки (брелоки) до бази даних програми користувач програми з рівнем не менше адміністратора має запустити довідник співробітників (рис.3.5). для відкривання довідника співробітників в меню «Довідники» треба обрати відповідний пункт.

У цьому довіднику користувач може давати унікальні ідентифікаційні дані ключів доступу. Для спрощення процедури при перевіреному зв'язку з модулем достатньо піднести картку яку треба зберегти до зчитувача та натиснути кнопку «Зчитати картку». При цьому ідентифікаційний код з'явиться у відповідному полі та користувачеві достатньо буде вказати прізвище, ім'я та по батькові робітника, якому ця картка буде видана.

Справочник сотрудников

Список сотрудников:

№	Фамилия	Имя	Отчество	ID карты
8	5			1462122595
9	6			133177955
10	7			1204695395
11	8			394866275
12	9			3343921251
13	10			2269848675

ID карты:

Фамилия: Имя: Отчество:

Рисунок 3.5 – Довідник співробітників

Оскільки немало важливим є організація з'єднання апаратної частини з комп'ютером – окремо було створено форму налаштувань з'єднання (рис.3.6).

Настройки соединения

Тип соединения: ☒ COM порт ☐ TCP соединение

Номер COM порта:

Скорость COM порта:

LabeledEdit_IPAddr:

LabeledEdit_IPPort:

Рисунок 3.6 – Форма налаштувань з'єднання

Модуль, що було обрано для реалізації зв'язку та описано у п.2.2 кваліфікаційної роботи (NRF-USB перехідник) встановлює віртуальний COM порт, а отже найбільш розповсюджене буде з'єднання ао COM порту. Однак існує варіант коли запис NRF-USB встановлюється NRF-COM перехідник

та COM-Ethernet перехідник та далі з'єднання з комп'ютером з програмою відбувається за допомогою Ethernet з'єднання. Це найчастіше буває у випадках коли через бетонні перекриття та велику відстань між модулем та комп'ютером радіосигнал не якісний. Тоді перехідник ставлять не дуже далеко від модуля, а далі дані передаються через локальну мережу.

Крім того модуль має досить велику кількість налаштувань, що описують його роботу, як з точки зору калібрування датчиків алкоголю так і з точки зору часу вимірювання, часу подавання звукових сигналів і т.і. Всі ці налаштування винесені у форму налаштувань параметрів (рис.3.7).

Рисунок 3.7 – Форма налаштувань параметрів

Крім перегляду вимірювань за поточну добу користувача може створювати звіти за будь який період як для повного переліку працівників так і по одному робітнику, а також переглядати вимірювання рівень яких перевищує межовий. При цьому отримані звіти можуть бути експортовані в таблицю Excel або надруковані безпосередньо з програми (рис.3.8).

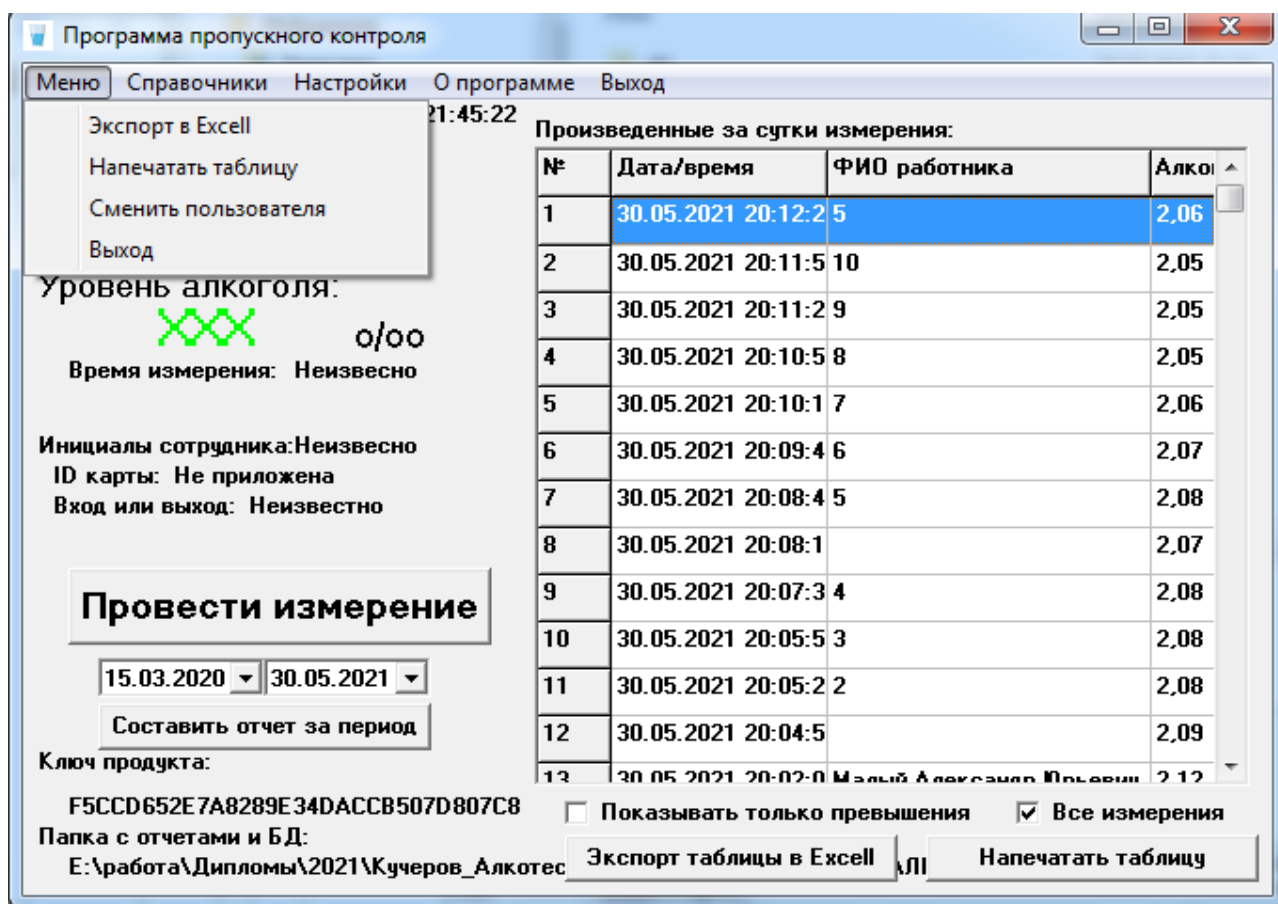


Рисунок 3.8 – Формування та експорт звітів

Повний текст основного модулю (Main) програми для ПК наведено у додатку Б.

ВИСНОВКИ

У кваліфікаційній роботі бакалавра було розроблено модуль контролю робочого часу з вбудованим алкотестером. Було складено структурну схему, обрано електронні компоненти, розроблено конструкцію плати та корпусу модуля, розроблена база даних, протокол обміну між платою та комп'ютером, написані програми керуючого мікроконтролера та персонального комп'ютера.

Особливостями розробленого модуля у порівнянні з існуючими аналогами є:

- передавання даних бездротовим способом;
- наявність внутрішньої пам'яті для проміжного зберігання даних;
- програмне забезпечення, що дозволяє зберігати дані та проводити створення звітів за будь які періоди;
- при необхідності можливе налаштування передавання вимірів на меседжер Telegram;
- можливість внесення змін в усі параметри роботи з програми;
- можливість проведення віддаленої калібровки;
- невелика вартість.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Ворона В. А. Система контроля и управления доступов / В. А. Ворона, В. А. Тихонов. – М.: Горячая линия – Телеком, 2010. – 272с.
2. Рыжова В. А. Проектирование и исследование комплексных систем безопасности //СПб: НИУ ИТМО. – 2012.
3. Юрьев Н. Н. Система контроля и управления доступом / Н. Н. Юрьев, Т. А. Васяева, С. Д. Бельков, Н. С. Суббота // Информатика, управляющие системы, математическое и компьютерное моделирование. – 2017. – № 7. – С. 601–604.
4. Волхонский В.В. Основные положения концепции обеспечения безопасности объектов // Научнотехнический вестник СПбГУ ИТМО. – 2011. – № 3(73). – С. 116–121.
5. Носков А.Н., Чечулин А.А., Тарасова Д.А. Исследование эвристических подходов к обнаружению атак на телекоммуникационные сети на базе методов интеллектуального анализа данных // Труды СПИИРАН. 2014. Вып. 37. С. 208–224.
6. Волхонский В.В. Подход к оценке вероятности пресечения несанкционированных действий в объединенной системе безопасности // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. – Київ, 2000. – С. 77–81.

ДОДАТОК А – ПРОГРАМА КЕРУВАННЯ МОДУЛЕМ

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <avr/eeprom.h> //Підключаєм бібліотеку по роботі з ППЗУ
#include <avr/wdt.h> //підключаєм бібліотеку по роботі со сторожевим таймером
#include <SPI.h> //Підключаєм бібліотеку для роботи по шині SPI
#include <MFRC522.h> //підключаєм бібліотеку для роботи со считывателем RFID RC522 13.56MHz
//-----
#define AlcoPin1 17 //A3 аналоговий вивід першого датчика алкоголю (вхід - без дисплея)
#define AlcoPin2 16 //A2 аналоговий вивід другого датчика алкоголю (вихід - с дисплеєм)
#define RST1_PIN 7//RST контакт 1го считывателя (вход - без дисплея)
#define SS1_PIN 8//SDA контакт 1го считывателя (вход - без дисплея)
#define RST2_PIN 9 //RST контакт 2го считывателя (выход - с дисплеем)
#define SS2_PIN 10 //SDA контакт 2го считывателя (выход - с дисплеем)
#define Buzzer 4 //контакт реле оповещателя привышения уровня
#define Lock 5 //контакт реле электронного замка
#define HLInpRead 6 //светодиод коробки входа - предупреждение что надо дуть
//-----
LiquidCrystal_I2C lcd(0x27, 16, 2);
MFRC522 RFID_IN(SS1_PIN, RST1_PIN); // Создаем объект считывателя входа
MFRC522 RFID_OUT(SS2_PIN, RST2_PIN); // Создаем объект считывателя выхода
//-----
int TIME_UNTIL_WARMUP = 900; //время нагрева датчика алкоголю для входа в рабочий режим

int AlcoLevel[2]={0,0}; //выход АЦП датчиков алкоголю (вход [0], выход [1])
byte CardNumb=0; //номер считывателя к которому приложена карта 0 - не приложена не куда; 1 - к
считывателю входа; 2 - к считывателю выхода
unsigned long uidDec, uidDecTemp; // для хранения номера метки в десятичном формате
unsigned long LastuidDec[2]={0,0}; //последние считанные ИД карт с входа и выхода (вход [0], выход [1])
unsigned long StartSRead, StopSRead; //засекаем время начала и конца измерения алкоголю

byte MTm=15; //время (в секундах) измерения концентрации алкоголю после прикладывания карты
byte BuzzerTime=10; //время (в секундах) включения оповещателя при привышении уровня алкоголю
unsigned int AlcZero[2]={95,140}; //максимальный уровень показаний АЦП для нормального дыхания (вход
[0], выход [1])
float ro[2]={6.3,6.3}; //опір датчика при чистому повітрі (вход [0], выход [1])

float AlcStep[2]={10,10}; //шаг показаний АЦП на 0,05 промиле (вход [0], выход [1])
bool NeedPCTest=false; //необходимость проверять ответ ПК
char EquipID[6]="ID001"; //ідентифікаційний номер обладнання

//-----
void setup() {
    //настройка направлений портов
    pinMode(Buzzer, OUTPUT);
    pinMode(Lock, OUTPUT);
    pinMode(HLInpRead, OUTPUT);
    //читаем текущие настройки в ППЗУ
    ReadSetings();
    //запуск порта
    Serial.begin(9600);
    //инициализация считывателей
    SPI.begin(); // Инициализация шины SPI
    RFID_IN.PCD_Init(); // Инициализация считывателя входа
    RFID_OUT.PCD_Init(); // Инициализация считывателя выхода
    //запуск дисплея и вывод приветственной надписи
    lcd.init(); // инициализируем экран
    lcd.backlight(); // включаем подсветку
    lcd.setCursor(0, 0); // Устанавливаем курсор в начало 1 строки
    lcd.print("Module Stated"); // Выводим текст
```

```

Serial.println("Module Started");
delay(500);
BuzzerOFF;
LockOFF;
digitalWrite(HLInpRead, LOW); //тушим светодиод
AlcoLevel[0]=ReadAlcoADC(1);
AlcoLevel[1]=ReadAlcoADC(2);
TestPC();
TestPC();
TestPC();
while ((AlcoLevel[0]>AlcZero[0])||(AlcoLevel[1]>AlcZero[1])) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Warming");
    lcd.setCursor(8, 0);
    lcd.print(AlcoLevel[0]);
    lcd.setCursor(12, 0);
    lcd.print(AlcoLevel[1]);
    lcd.setCursor(0, 1);
    lcd.print("Please wait!");
    AlcoLevel[0]=ReadAlcoADC(1);
    AlcoLevel[1]=ReadAlcoADC(2);
}
ro[0]=calculateResistance(AlcZero[0])/ 60.0;
ro[1]=calculateResistance(AlcZero[1])/ 60.0;
}
//-----
void loop() {
    CardNumb=0;//по умолчанию у нас не приложена никакая карта
    TestPC();//проверяем команды с ПК
    //убеждаемся что считыватели подключены корректно
    if (GetRFIDVer(1)==0) {ShowRFID_INErr(); }
    if (GetRFIDVer(2)==0) {ShowRFID_OUTErr(); }
    //проверяем есть ли карта на считывателях
    if (!RFID_IN.PICC_IsNewCardPresent())&&!RFID_OUT.PICC_IsNewCardPresent())
    { LCDWait();//выводим надпись ожидания карты
    return; } //если не поднесена не к одму из считывателей карта - переходим на начало программы
    if (RFID_IN.PICC_ReadCardSerial()) {
        // читаем карту с входа
        CardIDOut(1); //выводим в порт и на экран ИД карты
        CardNumb=1;
    }
    else if (RFID_OUT.PICC_ReadCardSerial()) {
        // читаем карту с выхода
        CardIDOut(2); //выводим в порт и на экран ИД карты
        CardNumb=2;
    } else {return;}
    if (CardNumb<1) {return;} //если карта не приложена не к одному из считывателей - идем на начало
    //если карта приложена к входу зажигаем светодиод
    if (CardNumb==1) {digitalWrite(HLInpRead, HIGH);}
    //читаем датчик алкоголя на протяжении MTm секунд и выводим максимальное значение в зависимости от
    считывателя
    int AlcoNow=0;//показания АЦП перед задувом
    int MaxAlco=0;
    bool WasFlow=false;
    if (CardNumb==1) {AlcoNow=ReadAlcoADC(1);}
    if (CardNumb==2) {AlcoNow=ReadAlcoADC(2);}
    StartSRead=millis();
    StopSRead=millis();
    while ((StopSRead-StartSRead)<(MTm*1000)) {
        if (CardNumb==1) {
            AlcoLevel[0]=ReadAlcoADC(1);
            if (AlcoLevel[0]>MaxAlco) {MaxAlco=AlcoLevel[0];}

```

```

    if ((AlcoLevel[0]-AlcoNow)>2) {WasFlow=true;}
    }
    if (CardNumb==2) {
        AlcoLevel[1]=ReadAlcoADC(2);
        if (AlcoLevel[1]>MaxAlco) {MaxAlco=AlcoLevel[1];}
        if ((AlcoLevel[1]-AlcoNow)>2) {WasFlow=true;}
    }
    StopSRead=millis();
    if (StartSRead>StopSRead) {break;}
}
digitalWrite(HLInpRead, LOW); //гушим светодиод
//показания датчика алкоголя выводим в порт и на экран
if (!WasFlow) {
    Serial.println("No Flow!");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("No Flow!");
    BuzzerON();
    delay(1000);
    BuzzerOFF();
    return;
}
//показания датчика алкоголя выводим в порт и на экран
OutAlco(CardNumb, MaxAlco);
//если уровень алкоголя превышен - включаем буззер
if (MaxAlco>(AlcZero[CardNumb]+5)){
    BuzzerON();
    delay(BuzzerTime*1000);
    BuzzerOFF();
}
delay(3000);
}
//-----
//расчет промилле
float CalcPromille(int SNumb, int ADCIn){
    if (ADCIn<0) {return 0;}
    // float CTemp=readAlcoholPpm(SNumb, ADCIn);
    float CTemp=((ADCIn-AlcZero[SNumb-1])/AlcStep[SNumb-1])*0.05;
    if (ADCIn<=AlcZero[SNumb-1]) {CTemp=0;}
    return CTemp;
}
//-----
//показания датчика алкоголя выводим в порт и на экран
void OutAlco(byte numb, int tempAlc){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Alcogol Level:");
    lcd.setCursor(0, 1);
    float AlcPrm=CalcPromille(numb, tempAlc);
    if (tempAlc>-1) {
        lcd.print(tempAlc);
        lcd.setCursor(3, 1);
        lcd.print(":");
        lcd.setCursor(5, 1);
        lcd.print(AlcPrm);
    }
    else {lcd.print("Sensor Error");}
    Serial.print("Alco");
    Serial.print(numb);
    Serial.print(":");
    Serial.print(tempAlc);
    Serial.print("/");
    Serial.println(AlcPrm);
}

```

```

if (NeedPCTest) {
    for (byte i=0; i<5; i++) {
        if (IsPCAnswer("Alco OK")) {return;}
        Serial.print("Alco");
        Serial.print(numb);
        Serial.print(":");
        Serial.print(tempAlc);
        Serial.print("/");
        Serial.println(AlcPrm);
    }
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("No PC");
    BuzzerON;
    delay(2000);
    BuzzerOFF; }
return;
}
//-----
//выводим в порт и на экран ИД карты для ридера numb
//numb=1 вход; numb=2 выход
void CardIDOut(byte numb) {
    if ((numb<1)||numb>2)) {return;}
    uidDec = 0;
    if (numb==1) {
        for (byte i = 0; i < RFID_IN.uid.size; i++) {
            uidDecTemp = RFID_IN.uid.uidByte[i];
            uidDec = uidDec * 256 + uidDecTemp;
        }
        LastuidDec[0]=uidDec;
        CardIDtoLCD(uidDec); //выводим ИД карты на дисплей
        Serial.print("RFID1:");
        Serial.println(uidDec);
        if (NeedPCTest) {
            for (byte i=0; i<5; i++) {
                if (IsPCAnswer("RFID OK")) {return;}
                Serial.print("RFID1:");
                Serial.println(uidDec);
            }
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("No PC");
            BuzzerON;
            delay(2000);
            BuzzerOFF;
        }
        return;
    }
    if (numb==2) {
        for (byte i = 0; i < RFID_OUT.uid.size; i++) {
            uidDecTemp = RFID_OUT.uid.uidByte[i];
            uidDec = uidDec * 256 + uidDecTemp;
        }
        LastuidDec[1]=uidDec;
        CardIDtoLCD(uidDec); //выводим ИД карты на дисплей
        Serial.print("RFID2:");
        Serial.println(uidDec);
        if (NeedPCTest) {
            for (byte i=0; i<5; i++) {
                if (IsPCAnswer("RFID OK")) {return;}
                Serial.print("RFID2:");
                Serial.println(uidDec);
            }
        }
    }
}

```

```

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("No PC");
    BuzzerON;
    delay(2000);
    BuzzerOFF;
}
return;
}
return;
}

//-----
//проверяем отвечает ли комп
bool IsPCAnswer(String NeedAnsw)
{
    String s_temp="";
    char rc_ch;
    static int PCDeadTime=200;//максимальное время которое могут ждем команды от ПК
    unsigned long last=millis();
    while(millis()-last<=PCDeadTime) //пока есть время делаем попытки считывая
    {
        while(Serial.available())
        {
            rc_ch=Serial.read();
            s_temp+=rc_ch;
            if(rc_ch==0x03){ //0x03 байт завершения команды
                CleanRXBuffer();//очищаем буффер приема
                if (NeedAnsw==s_temp) {return true;} else {return false;}
            }
        }
    }
    return false;
}

//-----
//выводим на дисплей ИД считанной карты
void CardIDtoLCD(unsigned long cid) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Card ID:");
    lcd.setCursor(0, 1);
    lcd.print(cid);
    return;
}

//-----
//отображаем на дисплее и выводим в порт ошибку подключения считывателя на входе
void ShowRFID_INErr () {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("No RFID IN");
    Serial.println("No RFID IN");
    delay(2000);
    return;
}

//-----
//отображаем на дисплее и выводим в порт ошибку подключения считывателя на входе
void ShowRFID_OUTErr () {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("No RFID OUT");
    Serial.println("No RFID OUT");
    delay(2000);
    return;
}

```

```

}
//-----
//считать усредненное значение выхода АЦП датчика алгоголя с №numb
int ReadAlcoADC(byte numb) {
    byte AlcoPin;
    if (numb==1) {AlcoPin=AlcoPin1;}
    if (numb==2) {AlcoPin=AlcoPin2;}
    if ((numb<1)||numb>2)) {return -1;}
    int val[5];
    int valSum=0;
    for (int i=0;i<5;i++) {
        val[i]=analogRead(AlcoPin);
        valSum+=val[i];
        delay(20);
    }
    return valSum/5;
}
//-----
//проверить соединение со считывателем - получить его версию 1 или 2 либо
//3 - версия не известна; 0 - если соединение отсутствует
//numb - номер считывателя 1 - вход, 2 - выход
byte GetRFIDVer(byte numb) {
    if ((numb<1)||numb>2)){return 0;}
    byte v;
    // получить версию ПО MFRC522
    if (numb==1) {v = RFID_IN.PCD_ReadRegister(RFID_IN.VersionReg);}
    if (numb==2) {v = RFID_OUT.PCD_ReadRegister(RFID_OUT.VersionReg);}
    // Если 0x00 или 0xFF - соединение отсутствует
    if ((v == 0x00) || (v == 0xFF)) {
        return 0;
    }
    if (v == 0x91)
        return 1;
    else if (v == 0x92)
        return 2;
    else
        return 3;
}
//-----
//читаем ИД из ППЗУ
void ReadID(){
    EquipID[0]=eeprom_read_byte(0x0A);
    EquipID[1]=eeprom_read_byte(0x0B);
    EquipID[2]=eeprom_read_byte(0x0C);
    EquipID[3]=eeprom_read_byte(0x0D);
    EquipID[4]=eeprom_read_byte(0x0E);
    if ((EquipID[0])!='I') {
        EquipID[0]='I';
        EquipID[1]='D';
        EquipID[2]='0';
        EquipID[3]='0';
        EquipID[4]='1';
        EquipID[5]='\0';
    }
    return;
}
//-----
//пишем ИД в ППЗУ
void SaveID(){
    eeprom_update_byte(0x0A, EquipID[0]);
    eeprom_update_byte(0x0B, EquipID[1]);
    eeprom_update_byte(0x0C, EquipID[2]);
    eeprom_update_byte(0x0D, EquipID[3]);
}

```

```

    eeprom_update_byte(0x0E, EquipID[4]);
    eeprom_update_byte(0x0F, EquipID[5]);
    return;
}
//-----
//читаем настройки из ППЗУ
void ReadSettings(){
    MTm=eeprom_read_byte(0x00);//время (в секундах) измерения концентрации алкоголя после
    прикладывания карты
    BuzzerTime=eeprom_read_byte(0x01);//время (в секундах) включения оповещателя при превышении уровня
    алкоголя
    AlcZero[0]=eeprom_read_word(0x02);//уровень АЦП для нормального дыхания вход
    AlcZero[1]=eeprom_read_word(0x04);//уровень АЦП для нормального дыхания выход
    AlcStep[0]=eeprom_read_float(0x06);//шаг показаний АЦП на 0,05 промиле
    AlcStep[1]=eeprom_read_float(0x16);//шаг показаний АЦП на 0,05 промиле
    if (eeprom_read_byte(0x20)==1) {NeedPCTest=true;} else {NeedPCTest=false;} //необходимость проверять
    ответ компа
    //ідентифікатор обладнання
    ReadID();
    //перевіряємо граничні значення
    if ((MTm<5)||((MTm>200))) {MTm=15;}
    if ((BuzzerTime<2)||((BuzzerTime>200))) {BuzzerTime=10;}
    if ((AlcZero[0]<80)||((AlcZero[0]>200))) {AlcZero[0]=95;}
    if ((AlcZero[1]<90)||((AlcZero[1]>200))) {AlcZero[1]=140;}
    if ((AlcStep[0]<0.1)||((AlcStep[0]>50))) {AlcStep[0]=10;}
    if ((AlcStep[1]<0.1)||((AlcStep[1]>50))) {AlcStep[1]=10;}
    return;
}
//-----
//пишем настройки в ППЗУ
void SaveSettings(){
    eeprom_update_byte(0x00, MTm);
    eeprom_update_byte(0x01, BuzzerTime);
    eeprom_update_word(0x02, AlcZero[0]);
    eeprom_update_word(0x04, AlcZero[1]);
    eeprom_update_float(0x06, AlcStep[0]);
    eeprom_update_float(0x16, AlcStep[1]);
    if (NeedPCTest) {eeprom_update_byte(0x20, 1);} else {eeprom_update_byte(0x20, 0);}
    return;
}
//-----
//очищаем буфер приема
void CleanRXBuffer()
{
    delay(1);
    while(Serial.available())
    {
        Serial.read();
    }
    return;
}
//-----
//Відчинити замок
void LockON(){
    digitalWrite(Lock,HIGH);
    Serial.println("Lock Opened");
    return;
}
//-----
//Зачинити замок
void LockOFF(){
    digitalWrite(Lock,LOW);
    Serial.println("Lock Closed");
}

```

```

    return;
}
//-----
//Вмикнути оповісчуватель
void BuzzerON() {
    digitalWrite(Buzzer,HIGH);
    Serial.println("Buzzer ON");
    return;
}
//-----
//Вимкнути оповісчуватель
void BuzzerOFF() {
    digitalWrite(Buzzer,LOW);
    Serial.println("Buzzer OFF");
    return;
}
//-----
//Відправляємо поточні налаштування на ПК
void SetsToPort() {
    //<XX#XX%XXX$XXX&XX.X>
    Serial.print("<");
    Serial.print(MTm);
    Serial.print("#");
    Serial.print(BuzzerTime);
    Serial.print("%");
    Serial.print(AlcZero[0]);
    Serial.print("$");
    Serial.print(AlcZero[1]);
    Serial.print("&");
    Serial.print(AlcStep[0],2);
    Serial.print("+");
    Serial.print(AlcStep[1],2);
    Serial.println(">");
    return;
}
//-----
//Відправляємо поточний ідентифікатор в порт
void IDToPort() {
    Serial.println(EqipID);
    return;
}
//-----
//печатаем на экран что ожидаем карту
void LCDWait(){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" Wait Card");
    return;
}
//-----
//Вивести на дисплей на 1шу або 2гу строку текст до 16 символів
void LCDPrint(byte LRow, String LCDText){
    if ((LRow<1)||(LRow>2)) {return;}
    if (LCDText.length()>16) {return;}
    lcd.setCursor(LRow, 0);// Устанавливаем курсор в начало строки
    lcd.print(LCDText);    // Выводим текст
    delay(2000);
    return;
}
//-----
//Режим калібрування
void Calibration(){
    //читаем текущие нули и сохраняем

```

```

AlcZero[0]=ReadAlcoADC(1)+15;
AlcZero[1]=ReadAlcoADC(2)+15;
SaveSetings();
//а также выводим компу
SetsToPort();
return;
}
//-----
//проверяем команды с ПК
void TestPC() {
    String s_temp="";
    char rc_ch;
    static int PCDeadTime=200;//максимальное время которое могут ждать команды от ПК
    unsigned long last=millis();
    while(millis()-last<=PCDeadTime) //пока есть время делаем попытки считывая
    {
        while(Serial.available())
        {
            rc_ch=Serial.read();
            s_temp+=rc_ch;
            if(rc_ch==0x03){ //0x03 байт завершения команды
                CleanRXBuffer();//очищаем буфер приема
                AnalyzePCComs(s_temp);//отвечаем и анализируем полученные с ПК данные
            }
        }
    }
    return;
}
//-----
//отвечаем и анализируем полученные с ПК данные
void AnalyzePCComs(String PCCom){
    int ComLen=PCCom.length()-1;
    String str=PCCom.substring(0,ComLen);
    //Спочатку тестуємо команди без параметрів
    //Ініціювати калібрування StartCal
    if (str=="StartCal") {Calibration(); return;}
    //Отримати поточні значення калібрування GetCal
    if (str=="GetCal") {SetsToPort(); return;}
    //Відкрити замок LockON
    if (str=="LockON") {LockON(); return;}
    //Закрити замок LockOFF
    if (str=="LockOFF") {LockOFF(); return;}
    //Вмикнути оповісுவатель на встановлений в налаштуваннях час BuzzerON
    if (str=="BuzzerON") {BuzzerON(); return;}
    //Вимкнути оповісுவатель BuzzerOFF
    if (str=="BuzzerOFF") {BuzzerOFF(); return;}
    //Зчитати ідентифікатор обладнання GetID
    if (str=="GetID") {IDToPort(); return;}
    //Перевірити зв'язок з платою TestConnect
    if (str=="TestConnect") {Serial.println("Connect OK"); return;}
    //Встановити необхідність перевірки відповіді ПК SetPCTest
    if (str=="SetPCTest") {
        NeedPCTest=true;
        SaveSetings();
        Serial.println("TestSet");
        return;}
    //Скинути необхідність перевірки відповіді ПК ClearPCTest
    if (str=="ClearPCTest") {
        NeedPCTest=false;
        SaveSetings();
        Serial.println("TestClear");
        return;}
    //Тестуємо команди з параметрами

```

```

//Встановити рівень нормального дихання (вхід) SetZIn=XXX
String stemp=str.substring(0,7);
if (stemp=="SetZIn=") {
    stemp=PCCom.substring(7,ComLen);
    AlcZero[0]=stemp.toInt();
    SaveSetings();
    Serial.print("AlcZeroIn=");
    Serial.println(AlcZero[0]);
    return;}
//Встановити рівень нормального дихання (вихід) SetZOut=XXX
stemp=str.substring(0,8);
if (stemp=="SetZOut=") {
    stemp=PCCom.substring(8,ComLen);
    AlcZero[1]=stemp.toInt();
    SaveSetings();
    Serial.print("AlcZeroOut=");
    Serial.println(AlcZero[1]);
    return;}
//Встановити крок на 0,05 проміле SetAStepIn=XXX
stemp=str.substring(0,11);
if (stemp=="SetAStepIn=") {
    stemp=PCCom.substring(11,ComLen);
    AlcStep[0] = atof(stemp.c_str()); //преобразовываем строку в число с запятой
    SaveSetings();
    Serial.print("AlcStepIn=");
    Serial.println(AlcStep[0]);
    return;}
//Встановити крок на 0,05 проміле SetAStepOut=XXX
stemp=str.substring(0,12);
if (stemp=="SetAStepOut=") {
    stemp=PCCom.substring(12,ComLen);
    AlcStep[1] = atof(stemp.c_str()); //преобразовываем строку в число с запятой
    SaveSetings();
    Serial.print("AlcStepOut=");
    Serial.println(AlcStep[1]);
    return;}
//Встановити час звукового оповіщення про перевищення рівня алкоголю (в секундах) SetBTm=XX
stemp=str.substring(0,7);
if (stemp=="SetBTm=") {
    stemp=PCCom.substring(7,ComLen);
    BuzzerTime=stemp.toInt();
    SaveSetings();
    Serial.print("BuzzerTime=");
    Serial.println(BuzzerTime);
    return;}
//Встановити час очікування дихання після прикладування карти (в секундах) SetWTm=XX
stemp=str.substring(0,7);
if (stemp=="SetWTm=") {
    stemp=PCCom.substring(7,ComLen);
    BuzzerTime=stemp.toInt();
    SaveSetings();
    Serial.print("MTm=");
    Serial.println(MTm);
    return;}
//Встановити напис на дисплеї (1ша строка) Display1=XXXX
stemp=str.substring(0,9);
if (stemp=="Display1=") {
    stemp=PCCom.substring(9,ComLen);
    LCDPrint(1, stemp);
    Serial.print("Was Printed1 ");
    Serial.println(stemp);
    return;}
//Встановити напис на дисплеї (2га строка) Display2=XXXX

```

```

stemp=str.substring(0,9);
if (stemp=="Display2=") {
    stemp=PCCom.substring(9,ComLen);
    LCDPrint(2, stemp);
    Serial.print("Was Printed2 ");
    Serial.println(stemp);
    return;}
//Встановити ідентифікатор обладнання SetID=XXXXX
stemp=str.substring(0,6);
if (stemp=="SetID=") {
    stemp=PCCom.substring(6,ComLen);
    stemp.toCharArray(EquipID, 5);
    SaveID();
    Serial.print("EquipID=");
    Serial.println(EquipID);
    return;}
//Отримати поточні покази датчика алкоголю (1 – вхід, 2 – вихід) GetAlcoX
//AlcoX:YYY/ZZZ
stemp=str.substring(0,7);
if (stemp=="GetAlco") {
    stemp=PCCom.substring(7,ComLen);
    int ANumb=stemp.toInt();
    Serial.print("Alco");
    Serial.print(ANumb);
    Serial.print(":");
    int AlADC=ReadAlcoADC(ANumb);
    float AlPrm=CalcPromille(ANumb, AlADC);
    Serial.print(AlADC);
    Serial.print("/");
    Serial.println(AlPrm);
    return;}
//Отримати останній номер зчитаної карти GetLastCardX
stemp=str.substring(0,11);
if (stemp=="GetLastCard") {
    stemp=PCCom.substring(11,ComLen);
    int ANumb=stemp.toInt();
    if (ANumb==1) {Serial.println(LastuidDec[0]); return;}
    if (ANumb==2) {Serial.println(LastuidDec[1]); return;}
    Serial.println("Command Error");
    return;}
return;
}
//-----
// сопротивление датчика
float calculateResistance(int sensorADC) {
    float sensorVoltage = sensorADC * (5.0 / 1024);
    float sensorResistance = (5.0 - sensorVoltage) / sensorVoltage * 200;
    return sensorResistance;
}
//-----
float readScaled(int SNumb, int Alvl, float a, float b) {
    float ratio = calculateResistance(analogRead(Alvl)) / ro[SNumb-1];
    return exp((log(ratio) - b) / a);
}
//-----
float readAlcoholPpm(int SNumb, int Alvl) {
    if ((SNumb<1)|| (SNumb>2)) {return 0;}
    return readScaled(SNumb, Alvl, -0.66, -0.62) * 2.2;
}

```

ДОДАТОК Б – ОСНОВНИЙ МОДУЛЬ ПРОГРАМИ ДЛЯ ПК

```

unit Main;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Menus, Grids, ComCtrls, ExtCtrls, Size;
type
  TForm_Main = class(TForm)
    StringGrid_Measures: TStringGrid;
    MainMenu1: TMainMenu;
    N_Menu: TMenuItem;
    N_Settings: TMenuItem;
    N_About: TMenuItem;
    N_Exit: TMenuItem;
    N_ConnectSetup: TMenuItem;
    N_CreateReport: TMenuItem;
    N_Excell: TMenuItem;
    N_PrintTbl: TMenuItem;
    N_MenuExit: TMenuItem;
    N_PrintSetup: TMenuItem;
    N_ParamsSetup: TMenuItem;
    Label_Measuring_Inscr: TLabel;
    N_CameraSetup: TMenuItem;
    Button_ExportToExcell: TButton;
    Button_Print: TButton;
    DateTimePickerStart: TDateTimePicker;
    DateTimePickerEnd: TDateTimePicker;
    Button_CreateReport: TButton;
    Button_Measure: TButton;
    Label_DateTime_Inscr: TLabel;
    Label_DateTime: TLabel;
    Label_COMNumb_Inscr: TLabel;
    Label_COMBr: TLabel;
    Label_COMNumb: TLabel;
    Label_COMBr_Inscr: TLabel;
    Label_AverageAlco_Inscr: TLabel;
    Label_Storage_Inscr: TLabel;
    Label_DBFolder: TLabel;
    Timer_Clock: TTimer;
    Label_Promilles: TLabel;
    Label_DegAv_Inscr: TLabel;
    Label_StatusMessage: TLabel;
    PrinterSetupDialog1: TPrinterSetupDialog;
    Label_ProductKey: TLabel;
    Label_EquipID: TLabel;
    Timer_Port: TTimer;
    Label_WName_Inscr: TLabel;
    Label_WName: TLabel;
    Label_CardID_Inscr: TLabel;
    Label_CardID: TLabel;
    CheckBox_OnlyBig: TCheckBox;
    CheckBox_ShowAll: TCheckBox;
    Label_MeasureTime_Inscr: TLabel;
    Label_MeasureTime: TLabel;
    N_ChangeUser: TMenuItem;
    N_RFID: TMenuItem;
    N_Catalogs: TMenuItem;
    N_Workers: TMenuItem;
    Label_UserInscr: TLabel;
    Label_User: TLabel;
  end;

```

```

N_PortStatus: TMenuItem;
CheckBox_ShowProtokol: TCheckBox;
Label_EnteranceInscr: TLabel;
Label_Enterance: TLabel;
Timer_IscrWait: TTimer;
procedure N_ExitClick(Sender: TObject);
procedure N_MenuExitClick(Sender: TObject);
procedure N_AboutClick(Sender: TObject);
procedure N_ConnectSetupClick(Sender: TObject);
procedure N_PrintSetupClick(Sender: TObject);
procedure Timer_ClockTimer(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
procedure Timer_PortTimer(Sender: TObject);
procedure N_ParamsSetupClick(Sender: TObject);
procedure CheckBox_ShowAllClick(Sender: TObject);
procedure CheckBox_OnlyBigClick(Sender: TObject);
procedure FormResize(Sender: TObject);
procedure N_ChangeUserClick(Sender: TObject);
procedure N_WorkersClick(Sender: TObject);
procedure N_PortStatusClick(Sender: TObject);
procedure CheckBox_ShowProtokolClick(Sender: TObject);
procedure Timer_IscrWaitTimer(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
procedure ShowSetting; //выводим текущие настройки на форму
procedure TryStartPort; //попробуем запустить порт с текущими настройками
procedure ShowCOMStatus;
procedure ProgrammExit;
procedure LoadMeasForDay;
end;
var
Form_Main: TForm_Main;
MyDir: String; //папка с программой
DBFolder: String; //путь к базе данных
defaultLang: integer; //язык интерфейса
NeedProtokol: boolean; //необходимость показывать протокол в форме статуса
//параметры оборудования
MTm: byte; //время (в секундах) измерения концентрации алкоголя после прикладывания карты
BuzzerTime: byte; //время (в секундах) включения оповещателя при превышении уровня алкоголя
AlcZero: array [1..2] of integer; //максимальный уровень показаний АЦП для нормального дыхания (вход [1],
выход [2])
AlcStep: array [1..2] of real; //шаг показаний АЦП на 0,05 промиле (вход [1], выход [2])
NeedPCTest: boolean; //необходимость проверять ответ ПК
EquipID: string; //идентификационный номер оборудования

implementation
uses About, ComSetup, Scales, DBWork, EquipKey, GrdPrnt, ParamSetup, Auth,
Workers, ParametersSetings, PortStatus;
{$R *.dfm}
//проверяем наличие установленного в настройках COM порта в системе
//и в зависимости от наличия меняем цвет
procedure TForm_Main.ShowCOMStatus;
begin
if Scales.IsCOMInSystem(Scales.COMPortNumb) then Form_Main.Label_COMNumb.Font.Color:=clGreen
else Form_Main.Label_COMNumb.Font.Color:=clRed;
end;
//выводим текущие настройки на форму
procedure TForm_Main.ShowSetting;
begin
Form_Main.Label_COMNumb.Caption:='COM'+IntToStr(Scales.COMPortNumb);

```

```

Form_Main.Label_COMBr.Caption:=IntToStr(Scales.COMPortBr);
Form_Main.Label_DBFolder.Caption:=DBFolder;
Form_Main.Label_DBFolder.Hint:=DBFolder;
ShowCOMStatus;//от наличия меняем цвет номера порта
end;
//попробуем запустить порт с текущими настройками
procedure TForm_Main.TryStartPort;
begin
  Scales.StartPort(Scales.COMPortNumb, Scales.COMPortBr);
end;
//настройка внешнего вида таблицы со взвешиваниями
procedure TableSetup;
begin
  Form_Main.StringGrid_Measures.Cells[0,0]:='№';
  Form_Main.StringGrid_Measures.Cells[1,0]:='Дата/время';
  Form_Main.StringGrid_Measures.Cells[2,0]:='ФИО работника';
  Form_Main.StringGrid_Measures.Cells[3,0]:='Алкоголь';
  Form_Main.StringGrid_Measures.ColWidths[0]:=40;
  Form_Main.StringGrid_Measures.ColWidths[1]:=100;
  Form_Main.StringGrid_Measures.ColWidths[2]:=150;
  Form_Main.StringGrid_Measures.ColWidths[3]:=50;
end;

//загрузка измерений в таблицу
procedure ShowMeas(MTemp:Meass);
var
  i,n,k:integer;
  showall:boolean;
  onlybig:boolean;
  rtemp,dtemp:real;
begin
  showall:=Form_Main.CheckBox_ShowAll.Checked;
  onlybig:=Form_Main.CheckBox_OnlyBig.Checked;
  n:=length(MTemp);
  if n=0 then Exit;
  Form_Main.StringGrid_Measures.RowCount:=2;
  Form_Main.StringGrid_Measures.Rows[1].Clear;
  k:=0;
  for i:=n downto 1 do begin
    if (not showall) then begin
      rtemp:=MTemp[i-1].MValue;
      if rtemp=0 then Continue;
    end;
    Form_Main.StringGrid_Measures.RowCount:=Form_Main.StringGrid_Measures.RowCount+1;
    k:=k+1;
    Form_Main.StringGrid_Measures.Cells[0,k]:=IntToStr(k);
    Form_Main.StringGrid_Measures.Cells[1,k]:=MTemp[i-1].MDate+' '+MTemp[i-1].MTime;
    Form_Main.StringGrid_Measures.Cells[2,k]:=MTemp[i-1].WorkerS;
    Form_Main.StringGrid_Measures.Cells[3,k]:=MTemp[i-1].MValueS;
  end;
  Form_Main.StringGrid_Measures.RowCount:=k+1;
  Form_Main.StringGrid_Measures.Row:=1;
end;

//загружаем таблицу за текущие сутки
procedure TForm_Main.LoadMeasForDay;
var
  ThisDayMeas:Meass;
  s_temp:string;
begin
  s_temp:=DateToStr(Now);
  DBWork.MeasuresLoadByCondition(s_temp,s_temp, 0, 0, ThisDayMeas);
  ShowMeas(ThisDayMeas);

```

```

end;

procedure TForm_Main.FormCreate(Sender: TObject);
begin
  Auth.WasAuthorised:=false; //авторизации пока не было
  defaultLang:=2; //язык интерфейса по умолчанию - русский
  MyDir:=ExtractFileDir(Application.ExeName); //сохраняем имя папки с программой
  NeedProtokol:=false;
  //проверяем ключ
  EquipKey.KeySalt:='SecretKey';
  EquipKey.EquipID:=EquipKey.GenerateEquipID;
  Form_Main.Label_EquipID.Caption:=EquipKey.EquipID;
  ParametersSetings.LoadParams;

  // if not FileExists(MyDir+'key.lic') then Application.Terminate;
  // CheckKeyFile; //если ключевой файл не соответствует - программа завершит работу
  TableSetup; //Настройка внешнего вида таблицы
  LoadSettings; //загружаем текущие настройки соединения
  LoadParams; //загружаем текущие настройки параметров измерений
  ShowSetting; //выводим текущие настройки на форму
  //соединяемся с БД
  DBWork.LinkToDB(DBFolder+'avrobit.mdb');
  // DBWork.SaveEvents(0); //Сохраняем время входа
  //выставляем текущую дату в качестве конечной для отчета
  Form_main.DateTimePickerEnd.DateTime:=Now;
  //если порт есть в системе запускаем его
  if Scales.IsCOMInSystem(Scales.COMPortNumb) then TryStartPort;
  //загружаем в таблицу взвешивания за текущие сутки
  LoadMeasForDay;
end;

//При выходе из программе
procedure TForm_Main.ProgrammExit;
begin
  DBWork.SaveEvents(1); //Сохраняем время выхода
  sleep(500);
  Application.Terminate;
end;

procedure TForm_Main.N_ExitClick(Sender: TObject);
begin
  ProgrammExit;
end;

procedure TForm_Main.N_MenuExitClick(Sender: TObject);
begin
  ProgrammExit;
end;

procedure TForm_Main.N_AboutClick(Sender: TObject);
begin
  Form_About.Show;
end;

procedure TForm_Main.N_ConnectSetupClick(Sender: TObject);
begin
  Form_ComSetup.Show;
end;

procedure TForm_Main.N_PrintSetupClick(Sender: TObject);
begin
  Form_Main.PrinterSetupDialog1.Execute;
end;

```

```

procedure TForm_Main.Timer_ClockTimer(Sender: TObject);
begin
  Form_Main.Label_DateTime.Caption:=DateToStr(Now)+' '+TimeToStr(Now);
  if not WasAuthorised then begin
    Form_Main.Hide;
    Form_Auth.Show;
  end;
end;

procedure TForm_Main.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
begin
  ProgrammExit;
end;

procedure TForm_Main.N_ParamsSetupClick(Sender: TObject);
begin
  Form_ParamsSetup.Show;
end;

procedure TForm_Main.CheckBox_ShowAllClick(Sender: TObject);
begin
  if Form_Main.CheckBox_ShowAll.Checked then Form_Main.CheckBox_OnlyBig.Checked:=false;
  LoadMeasForDay;
end;

procedure TForm_Main.CheckBox_OnlyBigClick(Sender: TObject);
begin
  if Form_Main.CheckBox_OnlyBig.Checked then Form_Main.CheckBox_ShowAll.Checked:=false;
  LoadMeasForDay;
end;

procedure TForm_Main.FormResize(Sender: TObject);
begin
  Size.SetMainFormElemSize;
end;

procedure TForm_Main.N_ChangeUserClick(Sender: TObject);
begin
  Form_Auth.Show;
end;

procedure TForm_Main.N_WorkersClick(Sender: TObject);
begin
  Form_Workers.Show;
end;

procedure TForm_Main.N_PortStatusClick(Sender: TObject);
begin
  Form_PortStatus.Show;
end;

procedure TForm_Main.CheckBox_ShowProtokolClick(Sender: TObject);
begin
  NeedProtokol:=Form_Main.CheckBox_ShowProtokol.Checked;
end;

//таймер который заведает время проведения измерения
procedure TForm_Main.Timer_IscrWaitTimer(Sender: TObject);
begin
  Scales.DefaultInscr;
  Form_Main.Timer_IscrWait.Enabled:=false;

```

end;

end.

ДОДАТОК В – ПРЕЗЕНТАЦІЯ