Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інформаційних технологій електронних засобів
(повне найменування інституту, факультету)
Інформаційних технологій та телекомунікацій
(повне найменування кафедри)

Пояснювальна записка

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

на тему Розробка програмно-апаратного комплексу обліку та контролю роботи зернових елеваторів за швидкісні видові показників

Виконав: студент(ка) 6 курсу, групи РТ-519м

Спеціальності 172 Телекомунікації та радіотехніка
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інтелектуальні технології аніко-системної радіоелектронної техніки
(прізвище та ініціали) Козлова Д.В.
Керівник Малий О.Ю.
(прізвище та ініціали)
Рецензент Вовкобійнік В.О.
(прізвище та ініціали)

20 20

Форма № 25

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

Інститут, факультет Факультет радіоелектроніки та телекомунікацій
Кафедра Інформаційних технологій електронних засобів
Ступінь вищої освіти Магістр
Спеціальність 172 Телекомунікації та радіотехніка
(код і найменування)
Освітня програма (спеціалізація) Інтелектуальні технології спеціалізованої радіоелектронної техніки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри Шило Г.М.
«____» _____ 2020 року

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

Козловій Дар'ї Василівні
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмно апаратного комплексу обліку та контролю роботи зернових елеваторів за непрямими ознаками показників
керівник проєкту (роботи) Малий Олександр Юрійович к.т.н. доцент каф. ІТЕЗ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «12» листопада 2020 року № 325
2. Строк подання студентом проєкту (роботи) 18 грудня
3. Вихідні дані до проєкту (роботи) перелік та показники якості зернових культур, що пуйлаються на зберігання елеваторами, алгоритм роботи зернових елеваторів
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області, огляд аналогів, розробка структурної схеми комплексу, розробка бази даних, розробка серверної частини комплексу, розробка програмного забезпечення для ПК, розробка методології роботи з комплексом, охорона праці, організаційно-економічні розрахунки, висновки, перелік посилань, додаток А, додаток Б.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
18 слайдів
6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв *виконане* завдання
1	Малий О.Ю.		
2	Малий О.Ю.		
3	Малий О.Ю.		
4	Малий О.Ю.		
5	Левченко Н.М.		
6	Якімцов Ю.В.		
7.	Поспєєва Т.Е		

7. Дата видачі завдання «01» вересня 20 20 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз аналогів	9.10.20	
2	Постановка технічного завдання	11.10.20	
3	Розробка структури комплексу	12.10.20	
4	Розробка модулів керування та вибір апаратних елементів	20.10.20	
5	Розробка бази даних	15.11.20	
6	Розробка серверної частини	25.11.20	
7	Розробка програмного забезпечення для ПК	02.12.20	
8	Оформлення ПЗ та захист дипломного проекту	18.12.20	

Студент(ка) _____ (підпис) Козлова Д.В. (прізвище та ініціали)

Керівник проєкту (роботи) _____ (підпис) Малий О.Ю. (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до магістерської роботи: 138 сторінок, 54 рисунків, 37 таблиць, 15 джерел, 2 додатки.

Мета роботи. Розробка методики автоматизованої оцінки якості зернових здаються на зберігання на елеватори, розробити алгоритми віддаленого контролю якості та кількості продукції на зернових елеваторах.

Об'єкт дослідження. Методика оцінки якості зернової продукції і контролю кількості надходить на зберігання сільськогосподарської продукції.

Предмет дослідження. Програмно-апаратного комплексу обліку та контролю роботи зернових елеваторів за широким рядом показників

У першому розділі розглядаються огляд області розробки та постановка завдань магістерської роботи, використання систем обліку кількості та якості зернових при прийомі та відправлені на зернових елеваторах, описуються параметри комплексу, що розробляється та розгляд аналогів.

У другому розділі проводиться розробка та опис структурної схеми системи, вибір елементів апаратної частини системи.

У третьому розділі розробляється структура бази даних, алгоритм та текст програми серверної частини комплексу, алгоритм та структура інтерфейсу та тексту програм вагової та термінальної частини комплексу.

Четвертий розділ включає в себе методичне забезпечення по роботі з розробленим комплексом.

У п’ятому розділі проводиться розрахунок економічної ефективності та теоретичної окупності науково-дослідного проекту.

У шостому розділі розглядаються питання з охорони праці та безпеки в надзвичайних ситуаціях.

ЗЕРНОВИЙ ЕЛЕВАТОР, ПОКАЗНИК ЯКОСТІ, АВТОМОБІЛЬНІ ВАГИ, ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС, СТРУКТУРНА СХЕМА, АЛГОРИТМ, ПРОГРАМА

ЗМІСТ

ВСТУП

Сьогодні запорукою продуктивної діяльності будь-якого підприємства є поняття його керованості. І підприємства зі зберігання і переробки сільськогосподарської продукції не виняток

Як відомо, Україна - один з лідерів експорту зерна. А будь-яка продукція вимагає контролю якості та обліку кількості.

Але якщо в частині керованості порівняти сільськогосподарське підприємство з будь-яким іншим, то розумієш, що в даній галузі втілити в життя це поняття набагато складніше, ніж на будь-якому іншому виробничому підприємстві. На руку цьому відіграють багато факторів, наприклад: низька кваліфікація персоналу в регіонах, часто низький культурний рівень населення, історично низький рівень зарплат (який провокує крадіжку), велика територіальна віддаленість виробничих ділянок (особливо якщо мова йде про рослинництво).

Останнім часом в сільськогосподарське виробництво вкладаються великі кошти, в агробізнес пішли великі компанії. Але як великої компанії з центром управління, наприклад, в Києві, контролювати роботу віддаленого афільованного підрозділу? Як забезпечити прозорість і достовірність обліку в компанії? Як ефективно управляти такими групами компаній? Всі ці питання дуже актуальні і для власників сільськогосподарських підприємств

Коли мова йде про елеваторі, то першочерговим завданням перед власником є контроль ввезення, вивезення та переробки зерна.

Існують різні види впливу на персонал з метою контролю, серед яких можна виділити основні:

- організаційні заходи (адміністративні рішення, організаційні заходи, посадові інструкції, внутрішні регламенти, розпорядки і графіки роботи);

- апаратні засоби (застосування механізмів, агрегатів, контролерів, додаткового обладнання або електронної техніки для контролю технологічних процесів, визначення ваги зерна, обліку кількості виконаних технологічних

операцій, визначення температури в сховищах, системи відеоспостереження, шлагбауми та ін.);

- програмні комплекси (застосування спеціальних програм з метою забезпечення збору даних, функції обліку діяльності підприємства, аналізу даних, обробки і передачі даних в керуючу компанію, формування звітності, а також управління бізнес-процесами).

Кожен з цих видів контролю має свої позитивні і негативні сторони при застосуванні. Так, цінність організаційних засобів управління полягає в адекватності, прозорості та оперативності прийняття необхідних для бізнесу керуючих рішень.

Наявність апаратних засобів контролю технологічних процесів дозволяє виключити людський фактор, забезпечити незалежну оцінку процесам, що відбуваються. Однак, апаратні засоби, як правило, є найменш гнучкими серед перерахованих видів інструментів, тому що обмежені попередньо закладеними в них функціональними можливостями.

Що стосується програмних комплексів, то вони, на наш погляд, є найбільш гнучкими засобами контролю, тому що дозволяють досить швидко реалізовувати нові функціональні можливості, адаптувати застосовується програмне забезпечення до реальних процесів, що змінюються стандартам і принципам управлінського обліку, що застосовуються в компанії.

Програмні комплекси дозволяють вести кількісно-якісний, бухгалтерський, податковий і управлінський обліки, контролювати основні бізнес-процеси на підприємстві, організувати імпорт даних з АСУ ТП і порівнювати фактичні процеси з обліковими даними, аналізувати дані, консолідувати звітність в керуючої компанії, обмежити права доступу до системам і багато іншого.

У дипломному проекті пропонується розробка системи яка дозволити автоматизувати облік і контроль діяльності зернових елеваторів.

Мета роботи. Розробка методики автоматизованої оцінки якості зернових здаються на зберігання на елеватори, розробити алгоритми віддаленого контролю якості та кількості продукції на зернових елеваторах.

Об'єкт дослідження. Методика оцінки якості зернової продукції і контролю кількості надходить на зберігання сільськогосподарської продукції.

Предмет дослідження. Програмно-апаратного комплексу обліку та контролю роботи зернових елеваторів за широким рядом показників

1 ОГЛЯД ОБЛАСТІ РОЗРОБКИ І ПОСТАНОВКА ЗАДАЧ

1.1 Системи обліку продукції на агропідприємствах

Елеватор - це зерновий склад - власник зерносховища. Він надає фізичним і юридичним особам послуги зі зберігання зерна, видає складські документи на зерно (п. 15 ст. 1, ст. 7 Закону України «Про зерно та ринок зерна в Україні» від 04.07.2002 № 37-IV). Такі ж послуги надають елеватори зернопереробних підприємств: борошномельні, круп'яні, комбікормові тощо.

При будівництві (переобладнанні) елеватора основна увага приділяється обладнанню, його вартості та ефективності. І на задній план переміщається питання обліку виконуваних на елеваторі (сховище) операцій. Хоча вже на етапі планування та реалізації будівництва нового елеватора можна передбачити деякі технічні моменти, які суттєво спростять подальшу автоматизацію обліку операцій з продукцією. Вирішення цих питань, на жаль, часто і фінансується, і виконується за залишковим принципом.

У той же час цілі, що досягаються автоматизованою системою обліку, необхідні для кожного хорошого керівника:

- прозорість;
- оперативність і достовірність;
- впорядкування відносин з клієнтами;
- ускладнення (аж до неможливості) зловживань персоналу;
- зниження трудомісткості рутинних операцій.

Перш за все давайте визначимося, що саме ми будемо розуміти під терміном «автоматизація обліку». Для підприємств зернопереробного профілю в першу чергу представляє інтерес автоматизація управлінського (позабалансового) обліку. Специфіка галузі вимагає автоматизації складського та виробничого обліку (на вагових, складах і в бухгалтерії), обліку якості сировини і продукції, кількісно-якісного обліку продукції та сировини (перерахунок фізичної ваги в заліковий). Також важливий фінансовий облік (взаєморозрахунки та інші фінансові потоки),

облік зобов'язань (накази, розпорядження, договори з постачальниками і покупцями). І, головне, облік операцій на підприємстві для недопущення зловживань на кожному з названих ділянок. Наведений перелік ділянок обліку показує: ведення обліку «по-старому» в сучасних умовах навряд чи можливо. Особливо, якщо врахувати, що джерелами даних для обліку зазвичай є нестабільність якісних показників сировини на вході і їх зміна в процесі зберігання і переробки, що відбивається на десятках показників і параметрів.

Звичайно, ручний облік і можливий, і застосовується. Ось тільки його точність і оперативність в нинішніх умовах, м'яко кажучи, недостатні. Не кажучи вже про широке поле для зловживань. Більш того, ручний облік зараз популярніше. Це ж нескладно: розробити на базі стандартних бухгалтерських операцій деяких ділянок (наприклад, форми 36), пару-трійку додаткових звітів. Такий підхід дозволяє швидко і недорого «залатати дірки» в обліку зерна. Однак, при змінах в законодавстві (зміни інструкцій) доводиться знову і знову правити раніше зроблені настройки. А оскільки найчастіше «неприємності вирішуються у міру їх надходження», тобто, облікові системи будуються без плану і структури, то на певному етапі такого програмування облікова система перестає задовольняти елементарним вимогам. І це якщо мова йде про одиничний підприємстві. Якщо ж розглядати керуючу компанію, яка об'єднує кілька підприємств, стає зрозуміло, що отримання загальної ясної та чіткої картини в принципі недосяжно

Автоматизація прийому зерна - прискорює операції прийому, зважування, спрощує роботу з документами, значно зменшує людський фактор і помилки.

При автоматизації прийому зерна - результати видно відразу: які автомобілі були зважені, скільки продукції отримано з полів (масивів), конкретних комбайнів або бригад. В оперативному режимі можна контролювати отримання зерна, його розміщення по смугах зберігання, очищення в зерноочисному агрегаті.

Зерновий ток - це територія з обладнанням, комплексом машин для післязбиральної обробки зерна і майданчиками тимчасового зберігання (смуги). На зерновому току зерно приймають з полів через автомобільні ваги, зважують,

очищають, сортують, тимчасово зберігають, протруюють, сушать і розміщують в складах тривалого зберігання.

Найважливішою точкою обліку є автомобільна вагова. Через автомобільні ваги приймається зернова продукція з полів, виконується облік внутрішніх переміщень, відпуск продукції споживачам і на внутрішнє споживання.

Автоматизовані пункти прийому зерна - це повний комплекс вдосконаленої техніки, автомобільні ваги, високотехнологічні бункери для завантаження і вивантаження зернових культур, будівлі та споруди, оснащені спеціальним обладнанням, що працюють в єдиній і злагодженої системі.

Принцип роботи такого пункту полягає в тому, що починаючи з контрольно-пропускного пункту (КПП), він налаштовується для співробітника, що займає конкретну посаду. За допомогою терміналу збору даних заноситься інформація про водія і автомобілі, який приїхав на пункт прийому зерна. Потім друкується пропуск на принтері етикеток - і тільки після цього машина заїжджає на територію.

Рисунок 1.1 – Автомобільні ваги для зважування машин з зерновими

Рисунок 1.2 – Основні цикли роботи зернового елеватора

Перед розвантаженням зерна береться проба і визначається його сортність. Для цих цілей на елеваторах лаборанти-технологи використовують ваги-вологоміри. Такі високоточні ваги є універсальним приладом для визначення вологості зерна з метою контролю якості продукції.

Після зважування автомобіля, його розвантажують, і він знову їде на зважування, після чого водієві видається акт другого зважування.

При автоматизації прийому зерна - можна оперативно отримувати звіти: які автомобілі були зважені, скільки продукції отримано з полів (масивів), конкретних комбайнів або бригад. В оперативному режимі можна контролювати отримання зерна, його розміщення по смугах зберігання, очищення в зерноочисному агрегаті.

Весь процес на елеваторі супроводжується відеоспостереженням.

Завдяки системі автоматизації пункту прийому зерна збільшується його пропускна здатність, здійснюється чіткий облік процесу - від початку і до кінця

прийому зернових, значно скорочується кількість розкрадань, аж до повної їх ліквідації. А прості і чіткі звіти дають можливість адміністрації підприємства «тримати руку на пульсі» і бути в курсі всього процесу.

Автоматизація системи абсолютно виключає людський фактор. Всі ланки роботи підприємства знаходяться під контролем. Незалежно від масштабів і обсягів, за гранично короткі терміни на вашому підприємстві будуть виконані максимально ефективно, швидко і якісно роботи по автоматизації пункту. Перевага системи полягає в тому, що облік зерна здійснюється в кількісному і якісному вираженні, а для зберігання зерна це дуже важливо.

Уявімо робочий день лабораторії. Надходження, переробка, відвантаження та переоформлення зерна і відходів, - немає жодної операції на зернопереробному підприємстві, яка проходила б без участі фахівців цього підрозділу. З одного боку, великий потік інформації, який вручну важко проконтролювати. З іншого боку, що надходять «нескромні пропозиції» за невелику зміну (частки відсотка) тих чи інших показників отримати певну винагороду. Спокуса велика. Без чіткої системи обліку та контролю експрес-аналізів і середньодобових зі спеціальними крос-перевірками виявити такі відхилення якщо і можливо, то лише до кінця заготовки, що вже досить неактуально. Коректно побудована система автоматизації дозволяє виявити ці відхилення на початок наступного дня.

Важливий момент - механізми по знеособлення прийнятих проб, що дозволяють повністю деперсоналізувати зразки, що надходять на дослідження. Лаборант просто виконує свою роботу, не знаючи власника надійшла продукції.

Другий момент - вагова. Якщо у зернопереробного підприємства немає електронних ваг, то у нього вже є великі складності. Але, тим не менше, в цьому випадку теж можна ускладнити неправомірні дії по фальсифікації ваги оброблюваного зерна. У разі ж використання електронних ваг некоректне вказівку ваги неможливо, тому що дані ваг прямо надходять в облікову систему. Автоматичне прийняття ваги брутто і тари дозволяє розраховувати фізичну вагу зерна для кожного завозу. Дані кожної товаротранспортної накладної об'єднуються в реєстри за якістю, місць вивантаження і іншим показникам.

Крім прямої економії коштів, систематизація економить час. Кожному автомобілю і причеп присвоюється штрих-код, після чого введення інформації про підприємство прискорюється в рази.

Слід також враховувати, що якісна облікова система повинна надавати можливість не тільки фіксування зазначених вище даних, але і забезпечувати автоматизацію всіх операцій зернопереробного підприємства: розраховувати послуги зі зберігання, переоформлення, переміщенню, супутніх послуг, приходу і відвантаження, розрахунку залікової ваги, операцій по зачистці , обліку природного убутку, продукції минулих років та ін.

Виходячи з усього вищесказаного, можна зробити висновок про те, що облікова система не тільки не є чимось, що можна відкласти «на потім», а, навпаки, рішення про встановлення подібних систем повинні прийматися саме в процесі підготовки виробничих потужностей. Необхідно заздалегідь готувати вагові з автоматичними вагами, пробовідбірники, канали зв'язку і кабельні системи, щогли кріплення відеокамер та інше, з тим, щоб не довелося напередодні сезону проводити роботи по автоматизації «ударними темпами».

1.2 Обзор аналогов

Розглянемо найближчі програмні комплекси для роботи на елеваторах.

Продукт "ІН-АГРО: Управління елеватором для України" призначений для автоматизації бухгалтерського, податкового, кількісно-якісного і управлінського обліку на хлібоприймальних, заготівельних і зернопереробних підприємствах України.

Рішення дозволяє створити єдиний інформаційний простір на підприємстві для контролю і управління всіма процесами.

Програмний продукт "ІН-АГРО: Управління елеватором для України" дозволяє:

- автоматизувати кількісно-якісний, бухгалтерський і податковий обліки, включаючи підготовку обов'язкової (регламентованої) звітності на хлібоприймальних заготівельних і переробних підприємствах України;

- вести облік сировини і готової продукції в повній відповідності з вимогою галузевих інструкцій на борошномельно-круп'яних підприємствах України;

- забезпечити вирішення всіх облікових завдань, що стоять перед бухгалтерською службою підприємства, працівниками цехів і елеваторів, виробничо-технологічною лабораторією підприємства для відображення фінансово-господарської діяльності;

- виконувати всі необхідні розрахунки, отримувати обов'язкові друковані форми документів, обов'язкову (регламентовану) звітність в електронному вигляді:

- товарно-транспортні накладні, журнали зважування залізничного і автотранспорту;

- журнали лабораторних аналізів як сировини під час приймання (форма 49) так сировини і продукції при переробці (форма 59). Електронні журнали дозволяють отримувати середньозважені значення характеристик за будь-який період, в довільних розрізах обліку;

- оперативні документи і журнали виробничих цехів: «Акти зачистки / сушки», «Форма 110» «Форма 112» «Звіт виробництва»;

- книги кількісно-якісного обліку (форма 36);

- отримувати аналітичну інформацію про складські залишки сировини і продукції в різних розрізах обліку;

- виконувати взаєморозрахунки з контрагентами, як по розрахунковому, так і по фактичному виходу готової продукції;

- розраховувати кількість виробленого комбікорму відповідно до його рецептурою.

При цьому бухгалтерський та податковий облік ведеться відповідно до чинного законодавства України. Кількісно-якісний облік ведеться у відповідності з галузевими інструкціями і стандартами.

Використання даного рішення дозволяє:

- системно управляти підприємством, оперуючи повною інформацією в режимі реального часу;

- у комплексі вирішувати завдання управління елеваторами і переробними підприємствами;

- автоматизувати основні бізнес-процеси і знизити зловживання з боку персоналу;

- вести кількісно-якісний і бухгалтерський обліки в єдиній інформаційній системі згідно з чинним законодавством, галузевим стандартам і інструкціям (оновлення продуктів виходять своєчасно);

- виключити дублювання інформації;

- забезпечити інформаційну безпеку підприємства;

- позбавити власника від несподіванок при перевірці, тому що елеватори, що працюють під управлінням даної системи неодноразово проходили перевірки ГІСХ;

- контролювати роботу підприємства, в т.ч. контроль дій користувачів; санкціонування внесення змін у введені раніше документи; виключення людського фактору при відображенні в обліку виробничих процесів і ін.

Наступний аналог «Керування елеватором 2.0».

Програма розроблена для належного ведення обліку й оформлення операцій із зерном і продуктами його переробки на хлібоприймальних та зернопереробних підприємствах.

Впровадження єдиної системи обліку - це дієвий механізм повного контролю діяльності елеватора, що дає можливість:

- кількісно-якісного обліку зерна;

- автоматизації роботи ваговій, лабораторії, бухгалтерів по зерну і взаєморозрахунках з власниками зерна, бухгалтера по переробці;

- автоматизації всіх операцій по руху зерна;

- розрахунку вартості послуг елеватора;

- управлінського та бухгалтерського обліку;

- контролю бізнес-процесів;

- аналізу даних;

- оперативного отримання необхідної звітності;

- відображення дій користувачів;

- консолідації даних і звітності;

- обмежених прав доступу користувачів

Функціональні можливості програми "Управління елеватором 2.0"

За допомогою програми "Управління елеватором" проводиться автоматизація повного спектру робіт, виконуваних на хлібоприймальних та зернопереробних підприємствах.

Автоматизація роботи вагової:

- контроль виконання операцій зважування;

- оформлення первинних документів, ведення журналу зважування;

- формування документів вагової (ТТН на ввезення / вивезення / внутрішнє переміщення, квитанція на зважування, накладна на ввезення / вивезення ж / д транспортом);

- формування звітів вагової (журнал реєстрації зважування, журнал вагаря).

Автоматизація основних робіт лабораторії:

- якісні характеристики зерна відображаються в документі «лабораторний аналіз»;

- ведення журналу лабораторних аналізів;

- формування звітів за якістю зерна;

- форми 34, 117;

- формування карток лабораторних аналізів;

- рецепт.

Автоматизація робіт бухгалтера по зерну:

- реєстрація ввезення зерна, оформлення вивезення зерна;

- виписка складських документів (складська квитанція - проста, подвійна, ТТН);

- переоформлення прав власності на зерно;

- переміщення зерна по території;

- зачистка складів;

- реєстрація рухів в обліковій системі.

Автоматизація робіт бухгалтера по розрахунках:

- встановлення вартості послуг елеватора;

- формування договорів з власниками зерна;

- формування первинних документів і звітності.

Автоматизація робіт бухгалтера по переробці:

- списання сировини і матеріалів у виробництво (формування акту на списання);

- оприбуткування готової продукції;

- фасовка готової продукції (формування актів фасування);

- формування виробничих звітів, форма 117.

Також програмою передбачено розрахунок зарплати, ведення кадрового обліку, облік виробництва, облік роботи сільгосптехніки і автотранспорту.

Таким чином, при розробці даної програми враховувалися:

- застосування прийнятих в галузі методів розрахунку;

- ведення кількісно-якісного обліку;

- автоматизація роботи ваговій, лабораторії, бухгалтерів по зерну і взаєморозрахунках з власниками зерна, бухгалтера по переробці;

- автоматизація всіх операцій по руху зерна;

- розрахунок вартості послуг елеватора;

- взаєморозрахунки з власниками зерна, переробка зерна.

Наступним розглянутим аналогом є Софтінформ: Управління елеватор для України - галузеве рішення створено на базі типової конфігурації «Бухгалтерія 8 для України».

Рішення призначене для автоматизації кількісно-якісного, регламентованого обліку та системи фінансового моніторингу на хлібоприймальних, заготівельних і переробних підприємствах України.

Бухгалтерський і податковий облік ведеться відповідно до чинного законодавства України. Кількісно-якісний облік ведеться у відповідності з галузевими інструкціями і стандартами.

Система дозволяє звести до мінімуму помилки людського фактора на елеваторі, а також прискорити роботу за рахунок зручного інтерфейсу і автоматичних обробок.

Головне призначення системи:

- організувати роботу підприємства, відповідно до чинного законодавства в питаннях обороту зерна;

- налагодити контроль за зловживання в частині обліку руху зерна та зерно продуктів;

- зробити прозорим облік зерна, що знаходиться на зберіганні і надавати всю необхідну звітність підприємству, що здає зернові на зберігання;

- надати керівництву найбільш повну та якісну інформацію про діяльність його підприємства;

- контроль дебіторської заборгованості, використання і товарно-матеріальних цінностей;

Функціональні можливості

На додаток до функціонала типового рішення «Бухгалтерія 8 для України», програмний продукт «Софтінформ: Управління елеватор для України містить функції, зумовлені особливостями ведення виробничої діяльності на хлібоприймальних підприємствах України, а також особливостями кількісно-якісного бухгалтерського обліку та системи контролю за використанням грошей.

Кількісно-якісний облік: конфігурація поставляється з підсистемою автоматизації виробничих процесів. В поставку включені налаштування основних процесів елеватора, пов'язані з ввезенням, вивезенням, переміщенням і підробкою зерна.

Інтеграція з іншими програмними комплексами та електронними пристроями: в рамках конфігурації реалізовані механізми зв'язку з програмою державного підприємства «Держреєстри України» - «ЕРЗС-реєстратор», використовуваними на підприємствах для реєстрації складських квитанцій. Реалізована можливість взаємодії з електронними вагами (автомобільними, залізничними і бункерних).

Ведення бухгалтерського і податкового обліку відповідно до національних стандартів України. Забезпечено виконання завдань, що стоять перед бухгалтерською службою підприємства - від обробки первинних документів до формування регламентованої звітності.

Бюджетування. Окремим розділом в конфігурації реалізований механізм контролю за грошовими коштами відповідно до затверджених статей бюджету, а також контролю за його виконанням з різною періодичністю в залежності від фінансового плану підприємства.

У таблиці 1.1 наведено зведений аналіз програмних комплексів по роботі з елеваторами, представлений на ринку.

Таблиця 1.1 – Порівняння аналогів

Параметр	«ІН-АГРО»	«Керування елеватором 2.0»	«Софтінформ»
Облік зважувань	+	+	+/-
Можливість формування звітів по різним параметрам за різні періоди часу	+	+	+
Журнали дій користувачів	-	+/-	-
Формування товарно-транспортних накладних	+	+	+
Можливість автоматичного зважування на основі датчиків	-	+/-	-
Можливість використання RFID міток для визначення та автоматичного обліку машин	-	-	-
Використання фотофіксації машин під час зважування	-	-	-

Продовження таблиці 1.1

Параметр	«ІН-АГРО»	«Керування елеватором 2.0»	«Софтінформ»
Використання датчиків положення машини для визначення правильного розташування машини на платформі	-	-	-
Керування світлофорами та шлагбаумами	-	-	-
Можливість введення параметрів якості зернових	+/-	+	-
Можливість автоматичного визначення класу зернових та автоматичного визначення місця зберігання	-	+/-	-
Віддалене автоматичне відправлення даних на сервер	-	-	-
Відправлення даних на меседжер або електронну пошту	-	-	-

Існуючі на ринку України аналоги систем, що працюють на зернових елеваторах направлені перш за все на роботу з бухгалтерією та документообігом та не дозволяють максимально автоматизувати процес. Крім того жоден з аналогів не має можливості керувати обладнанням допуску машин на зважування, контроль зважування та не має можливості віддаленого відправлення та контролю.

1.3 Постановка завдань

Основними задачами автоматизованої системи контролю кількості та якості зерна на елеваторах є:

- визначення якості зерна у лабораторії;
- внесення показників якості у систему обліку з автоматичним визначенням класу пшениці;
- визначення кількості отриманої з кожної окремої машини пшениці;
- направлення пшениці на визначену дільницю для зберігання в залежності від отриманого класу якості або відправлення пшениці для виготовлення кормових культур при її низькій якості;
- автоматизований процес пропуску автомобілів з зерном на ваги з фіксацією правильності розташування на ваговій платформі та оповіщенням водіїв о кількості отриманого зерна та подальшого місця зберігання отриманого зерна;
- можливість автоматизованого створення звітів як на місці зважування так й віддалено за допомогою віддаленого сервера;
- можливість автоматизованого контролю кількості пшениці різних сортів, що зберігається на окремих ділянках елеватору.

Отже в дипломному проекті пропонується розробка системи, що дозволить:

- проводити внесення лабораторних показників якості пшениці, що отримується на зберігання елеваторами з передачею цих показників в базу даних на віддаленому сервері;
- автоматизоване зважування машини з пшеницею з контролем правильності встановлення автомобіля на платформу вагів та фото фіксацією;
- сповіщення водія автомобіля про закінчення процесу зважування та подальше відправлення для зберігання вмісту;
- система повинна мати можливість зберігання даних, щодо кількості та якості зерна, часу отримання або вивезення з можливістю створювання звітів по усім показникам.

Етапами з розробки системи є:

- огляд показників якості пшениці, що має заноситися в систему лабораторією;

- огляд роботи функціональних елементів для забезпечення якісного контролю кількості отриманої або відвантаженої пшениці;

- розробка структурної схеми системи з вибором функціональних елементів;

- розробка структури бази даних для зберігання всіх показників якості та кількості отриманої та відвантаженої пшениці та вказанням місця зберігання;

- розробка комутаційної схеми з’єднання елементів системи;

- розробка програмного забезпечення для роботи на ваговій елеватора з фото фіксацією в процесі зважування та з автоматизованим контролем силових установок (датчиків положення машини, шлагбаума, світлофора)

- розробка віддаленого сервера для обробки та зберігання даних;

- розробка програмного забезпечення для віддаленого контролю та введення параметрів якості в лабораторії.

2 РОЗРОБКА СТРУКТУРИ СИСТЕМИ

2.1 Показники та методики оцінки кількості та якості зерна на елеваторах

Якісні показники пшениці, що впливають на її клас регламентуються стандартом ДСТУ 3768:2019 та визначаються узагальненими стандартизованими методами в лабораторіях при елеваторах. В таблиці 2.1 наведені показники якості пшениці та віднесення пшениці до класів в залежності від цих показників.

Таблиця 2.1 – Показники якості пшениці згідно з ДСТУ 3768:2019

Показники	Характеристика та норма для м’якої пшениці по класам			
	1	2	3	4
Натура, г/л, не менше	775	750	730	Не обмежено
Скловидність, %, не менше	50	40	Не обмежено	Не обмежено
Вологість, %, не більше ніж	14	14	14	14
Зернова домішка, %, не більше ніж	5	8	8	15
Біті зерна	5	5	5	В рамках зернової домішки
Зерна злакових культур	3	4,0	4,0	В рамках зернової домішки
Пророслі зерна	2	3	3	В рамках зернової домішки
Триходесма сива	Не допускається			

Продовження таблиці 2.1

Показники	Характеристика та норма для м’якої пшениці по класам			
	1	2	3	4
Куколь	В рамках шкідливої домішки			
Кожний з видів інших отруйних насінь	0,05	0,05	0,05	0,05
Головневі зерна,%, не більш ніж	8	8	8	10
Вагова доля білка в перерахунку на суху речовину,%, не менше	14	12,5	11,0	Не обмежено
Вагова доля сирої клейковини,%, не менше	28	23	18	Не обмежено
Якість клейковини, одиниці приладу ІДК	45-100	45-100	45-100	Не обмежено
Число падіння, с, не менш ніж	220	220	180	Не обмежено

Під натурою зерна мається на увазі маса певного об'єму насіння (тобто по суті щільність)

Склоподібність зерна визначають по розрізу зерна з зовнішнім оглядом зрізів і за допомогою діафаноскопії. Для визначення скловидності з чистого зерна, що залишився після визначення засміченості, виділяють без вибору 100 цілих зерен.

Склоподібність зерна характеризує консистенцію його ендосперму. Склоподібність вказує на білковий або крохмалисті характер зерна. Пшениця з переважанням склоподібних зерен зазвичай відрізняється порівняно високим вмістом білка, клейковини і хорошими хлібопекарськими якостями. Пшениця, що складається в основному з крохмалистих зерен, бідна білком, і її краще

використовувати для хлібопечення в підсортуванні до іншої багатшою білками пшениці.

При визначенні скловидності на діафаноскопію поміщають 50 зерен (з попередньо виділених 100 цілих зерен) борозенкою вниз на решітку з овальними отворами так, щоб між ними і отворами не було зазорів. Грати з зернами вставляють в проріз корпуса приладу. Джерелом світла служить електрична лампа 55 Вт, що встановлюється на дні приладу, під гратами. Між джерелом світла і гратами розташовано матове скло, що розсіює світло.

Для кращого розпізнавання зерен в приладі (над гратами) є лупа. Передбачено пристрій для фокусування зображення.

При перегляді зерен підраховують кількість склоподібних і борошняних. Склоподібні зерна прозорі, добре просвічуються; борошнисті - темні, що не просвічуються; частково склоподібні зерна напівпрозорі.

Сумнівні зерна слід розрізати лезом бритви поперек (посередині), після чого визначити, до якої групи вони належать.

У такому ж порядку ведуть визначення в інший порції зерна (теж 50 зерен). Результати підрахунку в кожній порції підсумовують. Результати визначення скловидності проставляють з точністю до 1,0%.

Загальну скловидність зерна (Ос) у відсотках обчислюють за формулою:

$$O_c = П_c + \frac{Ч_c}{2}, \qquad (2.1)$$

де $П_c$ – кількість повністю скловидних зерен, шт.;

$Ч_c$ – кількість частково скловидних зерен, шт.

Вологість зерна є однією з найбільш важливих характеристик його якості. Визначення вологості зернових (а також бобових і зернобобових) культур проводять відразу ж після приймання нової партії.

За параметром вологості можна встановити кількісну частку поживних речовин в зерні, а також визначити тривалість його зберігання. Якщо вміст води в

зерні перевищує встановлену норму, то зерно починає швидко псуватися, а кількість корисних речовин в ньому різко зменшується. Через надмірної вологості активізуються небажані фізичні і хімічні процеси, які призводять до таких негативних результатів, як:

- набухання і проростання зерна;
- розщеплення високомолекулярних біополімерів;
- зменшення натури (маси зерна в 1 літрі);
- активізація ферментів (процеси бродіння);
- зниження сипучості і підвищення уразливості від механічних пошкоджень;
- швидкий розвиток паразитів - мікробів, кліщів, шкідливих комах.

Якщо зерно залишається надмірно вологим довгий час, це призводить до неможливості його обробки і подальшого зберігання. Навіть якщо вологість не надто перевищує допустиму норму, істотно знижується вихід зерна, а також страждає якість продукції, виготовленої з неї. Саме тому так важливо завжди точно визначати вологості зерна в лабораторних умовах, що проводиться різними методами. Найбільш швидким і зручним з цих методів є вимір вологості спеціальним вологоміром для зерна.

Існує спосіб вимірювання вологості зерна шляхом його висушування в спеціальному сушильній шафі. Формула вологості зерна в цьому випадку виглядає так: W = 100 - (m3 - m4) * (m1 - m2). Вологість в цій формулі позначається буквою W і вимірюється у відсотках.

Величини m1 і m2 - це маси наважок (20-грамових порцій) зерна, розмеленого на лабораторному млині, до і після його висушування. Величини m3 і m4 - маси наважок цільного зерна, також до і після його висушування. Одиниці виміру всіх мас в цій формулі - грами.

Сучасна методика вимірювання вологості зерна вологоміром - це ефективний і швидкий спосіб визначення вологості в будь-яких умовах:

- польових (при зборі врожаю);
- складських (в зерносховищах, на зернових токах і т.д.);
- при транспортуванні;

- перед помелом.

Сучасні вологоміри для зерна відрізняються високою точністю, компактними розмірами і максимальною зручністю при експлуатації. Вищеописані «класичні» методи вимірювання вологості, засновані на висушуванні зерна, вельми затратні за часом і іншим ресурсам. Портативні ж вологоміри дозволяють проводити експрес-оцінку в будь-яких умовах і приймати оперативні рішення щодо подальшої роботи з зерном.

Вологоміри для зерна випускаються трьох типів - кондуктометрічного, діелькометричного і гравіметричного. Залежно від різновиду, розрізняється не тільки ціна вологомірів, а й інші їх характеристики.

Кондуктометрічні вологоміри вимірюють електричний опір ділянки середовища, розташованого між двома голкоподібними електродами. Отриманий показник змінюється в залежності від кількості вологи, що міститься в зерні. Такі вологоміри обладнані мікропроцесором, а показник вологи виводиться в процентах. Кондуктометрічні вологоміри відрізняються високою швидкістю вимірювань і більш низькою ціною в порівнянні з вологомірами інших типів. Вони придатні для роботи «в полях», а також на етапах зберігання та транспортування зерна.

Принцип роботи діелькометричних вологомірів заснований на аналізі змін діелектричної проникності середовища - цей показник залежить від вологості. Такі вологоміри оснащені радіочастотним генератором з сигналом 3-30 МГц - сигнал здатний проникати в глибину середовища на 20-30 мм, а вологомір оцінює загасання сигналу. Також діелькометричні вологоміри обладнані герметичною камерою, в яку поміщається проба зерна для аналізу. Використовувати ці прилади можна як в польових умовах, так і в лабораторіях.

Гравіметричні вологоміри - найдорожчі, але при цьому і найбільш високоточні прилади (точність до 0,005%). Їх мінімальна вартість - від десятків тисяч гривень. Однак при цьому швидкість аналізу вологості досить низька - для отримання результатів потрібно досить тривалий час. Гравіметричний метод визначення вологості - це вимір маси зразка до і після виділення з нього міститься вологи. Ця

методика застосовується не тільки в агропромисловій галузі, але також в хімічній і фармацевтичній промисловості.

Вміст білка - це кількість білка, виражене у відсотках. Воно повинно бути на рівні 11-17%. При підвищенні вмісту білка більше 17-19% і при зниженні менше 11% погіршується якість хліба. Вміст білка і клейковини знаходиться в тісному зв'язку - збільшення вмісту білка в 1,4 рази відповідає збільшенню клейковини в 2 рази (наприклад, при збільшенні вмісту білка з 11 до 17%, вміст клейковини збільшується з 16 до 32%). Збільшення гідротермічного коефіцієнта протягом вегетації на 1 одиницю призводить до зниження вмісту білка на 3,78%, збільшення дози добрива на 1 ц / га сприяє збільшенню вмісту білка на 0,63%. Внесок використання будь-якого традиційного засоби захисту оцінюється в 0,44%. Тобто, якщо застосовується тільки один засіб (або гербіцид, або інсектицид, або триазольного фунгіцид) - вміст білка збільшується на 0,44%, два засоби - 0,88%, все три засоби (гербіцид, інсектицид і фунгіцид) - 1, 32%.

Вміст клейковини - розраховують як відношення кількості сирої клейковини до сумарного білку. Наявність клейковини визначає хлібопекарське якість борошна, отриманого з зерна пшениці. Так, відповідно до стандарту зерно: вищого класу повинно містити 36% клейковини; 1-го - 32%; 2-го - 28%; 3-го - 23% і 4-го - 18%. Збільшення гідротермічного коефіцієнта на 1 одиницю призводить до зниження вмісту клейковини на 9,55%, збільшення дози добрива на 1 центнер сприяє збільшенню вмісту білка на 2%. Внесок використання будь-якого засобу захисту оцінюється в 1,04%. Тобто, якщо застосовується, наприклад, тільки гербіцид - вміст білка збільшується на 1,04%, гербіцид і інсектицид - на 2,08%, гербіцид, інсектицид і фунгіцид - на 3,12%.

Оцінки якості зернових оцінюють в лабораторії, що як правило знаходяться на території елеватора спеціалізованими сертифікованими працівниками. При прийманні зернових на зберігання оцінка якості робиться перш за все для визначення місця зберігання згідно встановленого класу, щоб не перемішувати більш якісні зернові з менш якісними та щоб той хто здає зернові на зберігання мав можливість отримати зерно такого ж класу якості як і здав на зберігання. При

відпуску при продажу зернових елеватором якість визначається з метою визначення класу, щоб оцінити вартість за тонну.

Доопрацювання зерна - це сукупність технологічних операцій, спрямованих на забезпечення або поліпшення (при необхідності) встановлених показників якості зерна відповідно до вимог договору складського зберігання зерна, контракту, державних стандартів.

Визначається залікова вага зерна (Зв) в кг за спеціальною формулою (2.2):

$$З_в = Ф_в - Ф_в \times [(С_с + Х_в) : 100], \quad (2.2)$$

де $Ф_в$ — фізична вага зерна, прийнятого елеватором з врахуванням втрат на доробку (кг);

$С_с$ — процент зниження сміттєвих домішок (%);

$Х_в$ — процент зниження вологості (%).

Процент зниження сміттєвих домішок визначається по формулі (2.3):

$$С_с = [(100 - Х_в) \times (С_н - С_д)] : (100 - С_д), \quad (2.3)$$

де $С_н$ — показник сміттєвих домішок по отриманню (%);

$С_д$ — показник сміттєвих домішок згідно угоди (%).

Що стосується відсотка зниження вологості зерна, то він визначається за формулою (2.4):

$$Х_в = 100 \times (a - b) : (100 - b), \quad (2.4)$$

де а — показник вологості при отриманні (%);

b — показник вологості згідно угоди (%).

Залікова вага зерна вказується в реєстрі накладних на прийняте зерно (ф. №№ ЗХС-3, ЗХС-4), а також в складських документах. Залікова маса зерна

використовується для проведення всіх видів грошових розрахунків, в тому числі для залогових закупівель.

Завдання цього показника - визначити в загальній масі зерна ту частку, яка відповідає базовим критеріям (сорт і т.п.). Тобто його розрахунок проводиться, якщо зерно дійсно потребує доопрацювання. При надходженні зерна з якісними показниками вищими, ніж вказані в договорі складського зберігання, залікова вага цього зерна відповідає фактичному.

Для визначення кількості зернових на елеваторах використовують автомобільні ваги. При цьому машина двічі вїзжає на ваги – при розвантаження (спочатку повна, потім порожня), при розвантаженні навпаки (спочатку порожня, потім повня). За рахунок різниці маси при цих зважуваннях отримується вага зернових. Крім того після аналізу в лабораторії встановляється клас та сорність вантажу, а отже проводиться автоматичний розрахунок не лише ваги зернових, а й ефективна вартість.

Оскільки, людський фактор при зважуваннях зернових та особливо олійних культур, що мають досить вагому вартість, може призводити до значних фінансових зловживань під час підрахунку кількості отриманої або відправленої продукції – процес зважування треба максимально відокремити від дій людини.

Для цього існує декілька напрямів автоматизації зважування:

- системи розпізнавання номерних знаків автомобілів;
- використання безконтактних міток автомобілів;
- використання ключів брелоків водіями автомобілів.

Кожен з напрямів є свої переваги та недоліки.

Системи розпізнавання номерних знаків будуть погано працювати в умовах поганої видимості та якщо номерний знак автомобіля пошкоджений або брудний.

Використання безконтактних міток та ключів брелоків можливо лише для ідентифікації власного транспорту підприємства або несе за собою необхідність видачі тимчасових міток чи брелоків, що додає роботи працівникам охорони та знов додає людський фактор.

Крім того всі ці напрями розпізнавання автомобілів паралельно пов’язані з датчиками, що визначають правильність встановлення машини на вагах та системами світлової та звукової сигналізації водіям о порядку процесу зважувань (датчики розташування, шлагбауми, світлофори, сирени, дублюючі табло).

Отже при розробці системи необхідно передбачити можливість підключення всіх перекислених можливостей.

2.2 Розробка структурної схеми системи

Розроблена структура комплексу наведена на рис.2.1.

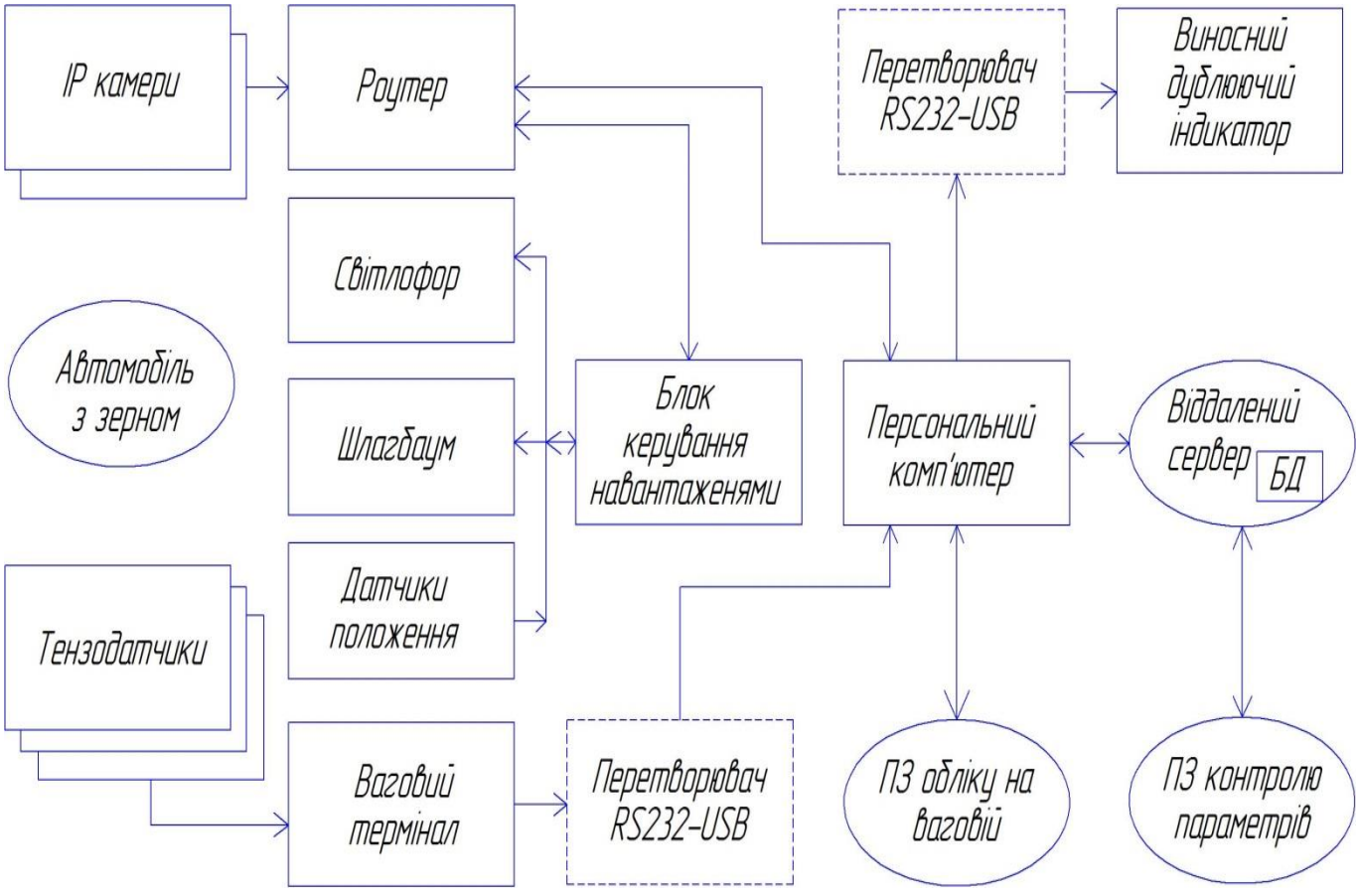

Рисунок 2.1 – Структурна схема програмно-апаратного комплексу

Автомобіль з зерном (або порожній для тарування) заїжджає на вагову платформу з тензометричними датчиками. Сигнал з тензодатчиків передається на ваговий термінал. Ваговий термінал фіксує значення ваги, відображає на

вбудованому дисплеї та передає дані за допомогою протоколу RS232 на комп'ютер. При цьому передача може відбуватись напряму у разі якщо комп'ютер має вбудований COM порт, або за допомогою перетворювача RS232-USB – у цьому випадку перетворювач приєднується до одного з портів USB, що наявні навіть у ноутбуків.

Отримані данні зберігаються на персональному комп'ютері за допомогою розробленої вагової програми. Для того, щоб водій автомобіля бачив зафіксовану вагу до комп'ютера також по протоколу RS232 через перетворювач або без нього підключається виносний дублюючий індикатор, що як правило розміщується на вулиці в місці зручному для того, щоб його показання були видні водієві навіть не виходячи з машини.

Крім того, для резервного зберігання та можливості віддаленого контролю з боку власника підприємства отримані дані з параметрами зважування передаються через мережу інтернет на віддалений сервер де додатково зберігаються в базі даних.

Керівник, власник або спеціально уповноважена ними особа має можливість керувати параметрами системи та продивлятися дані за допомогою програмного забезпечення контролю параметрів, що встановлюється на персональний комп'ютер. Це програмне забезпечення при виборі об'єкту контроль якого треба проводити – з'єднується з сервером та завантажує дані з бази даних серверу і відображає їх в зручній для людині формі.

Програмне забезпечення контролю параметрів також дозволяє працівникам лабораторії віддалено вказувати параметри якості для кожного автомобіля, що дозволяє зменшити паперову роботу та знімає необхідність супроводжувати автомобіль представником лабораторії.

Додатково комплекс дозволяє проводити фото та відео фіксацію процесу зважування за допомогою підключення до програмного забезпечення обліку на ваговій через Ethernet IP відеокамер.

Для керування процесом початку та завершення зважування в комплексі може використовуються датчики положення, світлофор та шлагбаум які підключаються

до блоку керування навантаженням який управляється сигналами через мережу Ethernet.

2.3 Розробка модуля керування навантаженнями та фотофіксація

Оскільки система складається не лише з програмної частини, а й з апаратної, під час проектування були обрані апаратні засоби.

Для проведення фото фіксації з подальшою можливістю автоматичного розпізнавання номерних знаків було обрано лінійку відеокамер. Відеокамери застосовуються цифрові з можливістю передачі даних цифровим відеопотоком по TCP/IP за протоколом rtsp та з можливістю передачі кадру за http запитом. Відеопотік може використовуватись для запису зображення охоронними системами. Можливість отримання кадру за http запитом використовується для реалізації можливості фото фіксації процесу зважування з можливим подальшим використанням алгоритму автоматичного розпізнавання номерного знаку для автоматизації заповнення даних щодо машини з продукцією.

В якості базових моделей камер було обрано камери лінійки Hikvision DS-2CD, що мають перелічені особливості (рис.2.2)

Рисунок 2.2 – IP камера Hikvision DS-2CD1023G0-I

Одним з важливих елементів комплексу є модуль керування навантаженням, що керується через мережу Ethernet. У якості блоку верування було обрано модуль

«Socket-2» української компанії «vkmodule», що має 2 дискретних входи (для забезпечення можливості підключення датчиків положення) та 2 релейні виходи (240В 10А) для керування шлагбаумом та світлофором.

Основні характеристики модулю:

- програмне управління по протоколу TCP/IP;

- управління Web-браузером по протоколу HTTP;

- можливість роботи 2-х пристроїв в парі (замикання входу на одному - включення реле на іншому);

- всі настройки і стани реле зберігаються в незалежній пам'яті;

- гальванічна розв'язка дискретних входів;

- напруга живлення 5В або 7 ... 30В постійного струму;

- корпус з кріпленням на DIN-рейку.

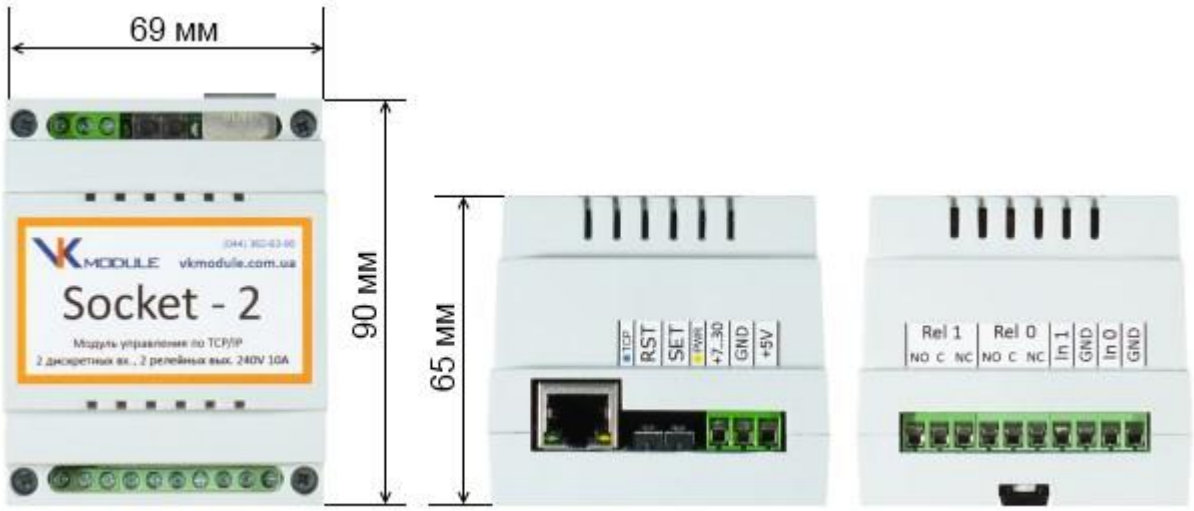

Рисунок 2.3 – Модуль «Socket-2»

Підключення по Ethernet виконується звичайним кабелем до комутатора або крос-кабелем безпосередньо до комп'ютера.

У контролері встановлений Web-сервер. Налаштування мережевих параметрів виконується через web-сторінку в браузері. Для входу на сторінку логін "admin", пароль "vkmodule". Налаштування зберігаються в незалежній пам'яті.

Адреса пристрою за замовчуванням 192.168.0.191.

Таблиця 2.2 – Характеристики модуля «Socket-2»

Параметр	Значення
Напруга живлення	5В або 7...30В постійного струму
Струм живлення	220 мА
Інтерфейс Ethernet	RJ-45
Світлодіодна індикація	"PWR"-наявність живлення "TCP" – встановлення зв’язку по TCP/IP "R0", "R1" - включення реле 0 и 1
Периферійні функції	- 2 дискретних входа з внутрішньою підтяжкою та гальванічною розв’язкою PC817; - 2 релейних виходи 240В/10А
Температурний діапазон роботи	-20..+85 С
Розміри ШхВхГ	70х90х65мм
Вага	180г

Для з’єднання вагового терміналу та/або виносного індикатору з персональним комп’ютером через порт USB (а таке з’єднання буде як мінімум одне оскільки навіть при наявності винесеного на системний блок COM порта він буде тільки один) використовується перетворювач USB - RS232.

Було обрано перетворювач фірми "vkmodule" в першу чергу із міркувань більшою у порівнянні з китайськими аналогами надійності.

Основні характеристики перетворювача:

- встановлюється як віртуальний COM-порт;
- індикація прийому / передачі;
- живлення від USB.
- роз'єм RS232 «тато» з терморегулятором аналогічний порту ПК;
- виконаний на американській мікросхемі MCP2200 компанії Microchip;

- фіксована настройка порту 8, N, 1.

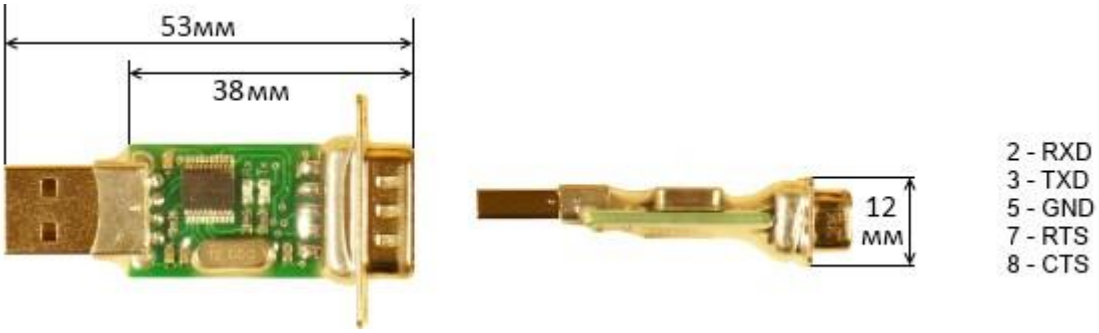

Рисунок 2.4 – Перетворювач USB - RS232 фірми "vkmodule"

Для визначення правильності встановлення машини, що проходить зважування особливо в автоматичному режимі використовуються датчики положення. В якості таких датчиків було обрано інфрачервоні датчики Tecsar Alert P-6110

Рисунок 2.5 – Інфрачервоний датчик Tecsar Alert P-6110

Принцип роботи.

Інфрачервоний бар'єр Tecsar Alert P-6110 складається з двох частин: приймача і випромінювача інфрачервоного променя. На ваговій платформі, де потрібно визначати положення автомобіля приймач і передавач розташовуються один напроти одного. Таким чином формується невидимий променевої ІК бар'єр. При

перетині машиною такого бар'єру, ІК промені перериваються, після чого бар'єр відправляє сигнал на блок керування навантаженнями.

Функції та особливості:

- компактні розміри приймача і випромінювача;
- легко встановлюється і налаштовується без спеціальних знань;
- виконано в пило-вологозахищеному корпусі;
- не реагує на рясні опади: дощ, сніг, град;
- переносить будь-які погодні умови, в тому числі низькі температури українських зим;
- ігнорує домашніх тварин;
- висота інфрачервоного бар'єру становить 7 см;
- максимальна відстань між приймачем і передавачем становить 15 метрів;
- завдяки світлодіодній індикації можна легко зрозуміти в якому статусі знаходиться ІК бар'єр;
- вертикальний і горизонтальний кут нахилу легко регулюється;

Для визначення значення ваги отриманої у вигляді сигналів з тензометричних датчиків в розробленому комплексі передбачена робота з максимально можливим переліком вагових терміналів. Це насамперед пов’язано з необхідністю забезпечення встановлення комплексу на об’єкти де вже використовуються електронні автомобільні ваги. Якщо на автомобільних вагах вже встановлено ваговий термінал, то необхідно забезпечити можливість роботи з ним. Різниця між різними типами вагових терміналів, що використовуються для автомобільних ваг з точки зору з’єднання з програмним забезпеченням полягає лише в форматі обміну даними. Відповідно для забезпечення роботи комплексу з різними ваговими терміналами достатньо лише розібратися з протоколом передавання даних цими ваговими терміналами. Ці протоколи, як правило, вказуються в документації, яку можна отримати у виробника або постачальника.

На даний момент в комплексі пропонується забезпечення можливості роботи з ваговими терміналами різних моделей наступних виробників:

- Keli;

- Esit;
- HBM;
- Rice Lake;
- Bilanciai;
- Мера;
- Елва.

Вибір вагових терміналів цих виробників пов'язаний з тим, що вони найширше представлені на ринку України. Але, треба зазначити, що як вже підкреслювалось в комплексі існує можливість з’єднання з будь яким документованим ваговим терміналом шляхом внесення доповнень про протокол його обміну даних в базу даних програмного забезпечення.

Для відображення інформації про зважування для водія в комплексі використовуються зовнішні виносні дублюючі індикатори. При розробці комплексу було запропоновано використовувати 2 типи таких індикатори:

- DPM;
- YHL.

Однак, як і у випадку з ваговими терміналами, діапазон моделей можна розширяти за допомогою доповнення програмного забезпечення у напрямку додавання протоколу обміну даними.

Рисунок 2.6 – Одна з моделей індикатора серії DPM

Рисунок 2.7 – Одна з моделей індикатора серії YHL

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка бази даних

Першим етапом при розробці програмних продуктів є створення їх структури та структури зберігання даних.

Оптимальним способом зберігання даних з точки зору швидкості пошуку і додавання даних є бази даних. Для виключення повторюють записів в полях таблиць баз даних використовують принцип побудови реляційних баз даних, які дозволяють максимально індексувати зберігання даних що дає можливість прискорити пошук.

Реляційна база даних - це набір даних з зумовленими зв'язками між ними. Ці дані організовані у вигляді набору таблиць, що складаються із стовпців і рядків. У таблицях зберігається інформація про об'єкти, представлених в базі даних. У кожному стовпчику таблиці зберігається певний тип даних, в кожному осередку - значення атрибута. Кожна стоку таблиці являє собою набір пов'язаних значень, що відносяться до одного об'єкту або сутності. Кожен рядок в таблиці може бути позначена унікальним ідентифікатором, званим первинним ключем, а рядки з декількох таблиць можуть бути пов'язані з допомогою зовнішніх ключів.

Для зберігання повної кількості необхідних параметрів була розроблена структура бази даних, що складається з 20 таблиць:

- таблиця додаткових описів Description;
- таблиця типів вагових терміналів Terminals;
- таблиця типів виносних індикаторів Tablo;
- таблиця користувачів Users;
- таблиця найменувань вантажів Cargos;
- таблиця водіїв Drivers;
- таблиця відправників Senders;
- таблиця отримувачів Recipients;
- таблиця платників Payers;
- таблиця перевізників Cariers;

- таблиця причепів Trailers;
- таблиця машин Cars;
- таблиця налаштувань камер Cameras;
- таблиця контролю версій Version
- таблиця дій користувачів UserLogs
- таблиця проведених зважувань Weighing
- таблиця відповідності фото - зважуванням PhotoWght
- таблиця RFID карт RFIDCards
- таблиця мов інтерфейсу Languages
- таблиця написів Inscriptions

Взаємозв’язок між таблицями наведено на рис.3.1. за допомогою ER діаграми.

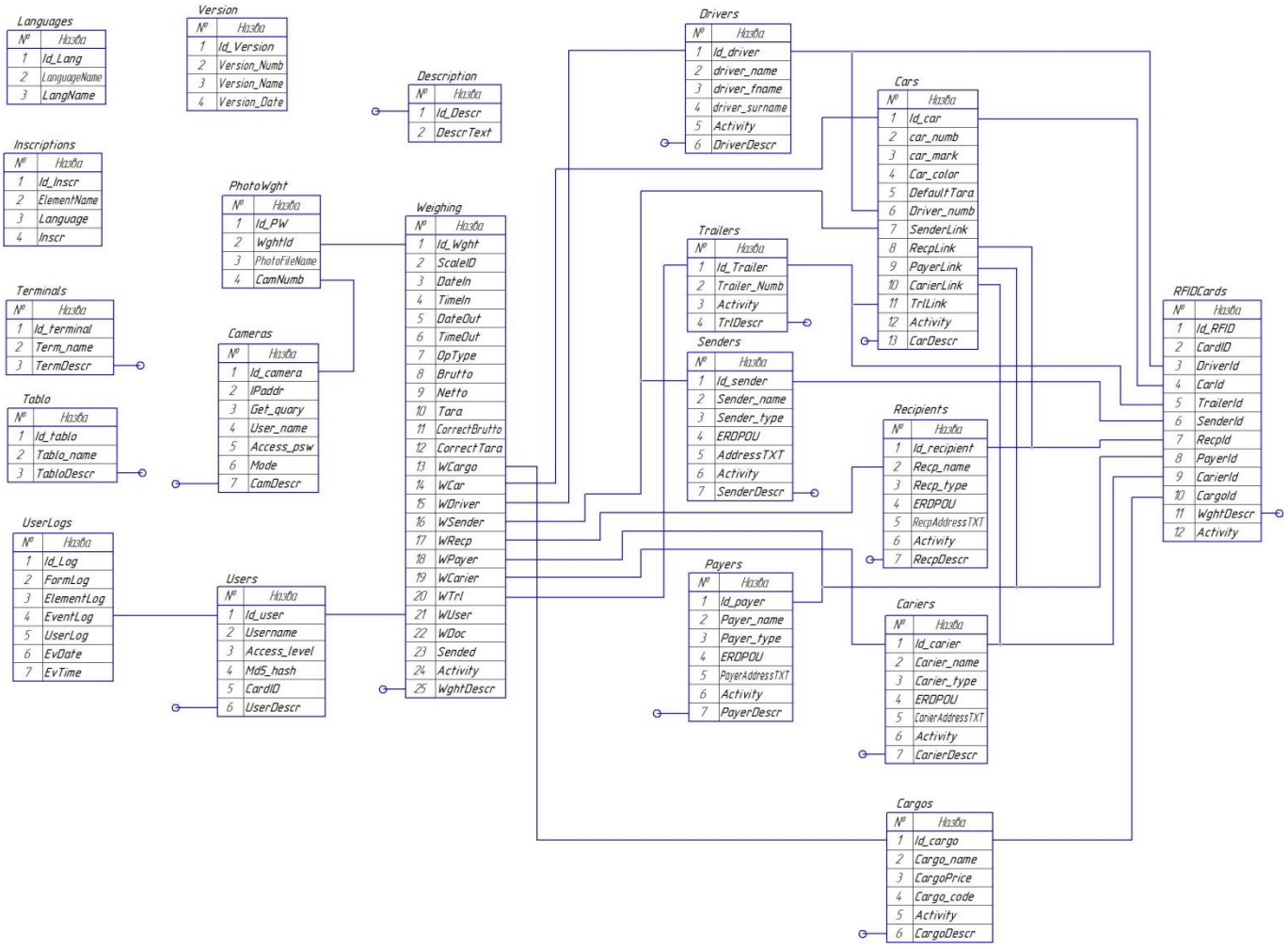

Рисунок 3.1 – ER-діаграма з’єднань таблиць бази даних

Таблиці бази даних створюються мовою SQL. SQL - це мова програмування декларативного типу. На відміну від звичних нам процедурних мов, в яких є умови, цикли і функції, в декларативних мовах подібних алгоритмічних конструкцій майже немає. Декларативні вислови є скоріше запити, опис того, що хоче отримати людина.

У разі SQL людина формулює запит на витяг чи внесення змін до даних, а алгоритм його виконання майже повністю лягає на плечі конкретної СУБД. Хоча якщо один і той же результат може бути отриманий за допомогою різних запитів, програмісту краще вибрати той, який створить менше навантаження на СУБД. Тобто програмісту бажано мати уявлення про те, як працює СУБД.

Запит оформлюється до таблиць бази даних, результатом обробки запиту також є таблиця, яку при бажанні можна зберегти.

Мова SQL призначений для створення і зміни реляційних баз даних, а також вилучення з них даних. Іншими словами, SQL - це інструмент, за допомогою якого людина управляє базою даних. При цьому ключовими операціями є створення таблиць, додавання записів в таблиці, зміна і видалення записів, вибірка записів з таблиць, зміна структури таблиць.

Однак в процесі розвитку мови SQL в ньому з'явилися нові засоби. Стало можливо описувати і зберігати такі об'єкти як індекси, уявлення, тригери і процедури. Тобто в сучасних діалектах SQL є елементи процедурних мов.

Мова SQL і СУБД зазвичай не використовуються самі по собі, а виконують функцію проміжного вбудованого компонента, що забезпечує зв'язок між прикладним ПО або програмою, яку пише програміст, і базою даних. У мовах програмування існують свої бібліотеки, що забезпечують API для різних СУБД.

Таблицею для описів всіх текстових пояснень є таблиця Description - вона створюється першою оскільки на неї найбільша кількість посилань у інших таблицях. На діаграмі (рис.3.1) з причини великої кількості послань зв’язки позначені колами щоб не було плутанини.

Таблиця 3.1 – Таблиця БД описів Description

№	Назва стовпця	Що зберігається	Тип даних
1	Id_Descr	Номер запису (ключ таблиці)	INT
2	DescrText	Текст опису/примітка	TEXT

Стовпець Id_Descr є унікальним ідентифікаційним номером таблиці, що дає можливість послання на неї.

Стовпець DescrText має на меті зберігати велике текстове повідомлення з описом якихось особливостей оскільки таблиця є резервною коли є необхідність додаткового опису параметрів або дописів до збережених даних.

SQL запит створення таблиці:

CREATE TABLE Description (Id_Descr INT, DescrText TEXT, PRIMARY KEY (Id_Descr));

Для переліку типів вагових терміналов, що використовуються в комплексі (з можливістю розширення переліку) в БД розроблена таблиця Terminals (структура наведена в табл.3.2)

Таблиця 3.2 – Таблиця БД вагових терміналів Terminals

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_terminal	Номер запису (ключ таблиці)	INT	
2	Term_name	Назва термінала	VARCHAR(30)	
3	TermDescr	Посилання на додатковий опис	INT	Description(Id_Descr)

Стовпець Id_terminal є унікальним ідентифікаційним номером таблиці, що дає можливість послання на неї.

Стовпець Term_name зберігає назву терміналу.

Стовпець TermDescr зберігає більш великий текстовий опис термінала шляхом посилання на таблицю описів Description, щоб зменшити розмір даних для зберігання та про індексувати поточну таблицю.

SQL запит створення таблиці терміналів:

CREATE TABLE Terminals (Id_terminal int, Term_name VARCHAR(30), TermDescr INT, PRIMARY KEY (Id_terminal), FOREIGN KEY (TermDescr) REFERENCES Description(Id_Descr));

Приклад SQL запиту для створення початкового запису з назвою терміналу або додавання нового терміналу в БД:

INSERT INTO Terminals VALUES (1,”XK3118T1”, NULL);

Для переліку типів дублюючих індикаторів, що використовуються в комплексі (з можливістю розширення переліку) в БД розроблена таблиця Tablo (структура наведена в табл.3.3)

Таблиця 3.3 – Таблиця БД дублюючих індикаторів Tablo

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_tablo	Номер запису (ключ таблиці)	INT	
2	Tablo_name	Назва табло	VARCHAR(30)	
3	TabloDescr	Посилання на додатковий опис	INT	Description(Id_Descr)

Стовпець Id_tablo є унікальним ідентифікаційним номером таблиці, що дає можливість послання на неї.

Стовпець Tablo_name зберігає назву індикатора.

Стовпець TabloDescr зберігає більш великий текстовий опис індикатора шляхом посилання на таблицю описів Description, щоб зменшити розмір даних для зберігання та про індексувати поточну таблицю.

SQL запит створення таблиці:

CREATE TABLE Tablo (Id_tablo INT, Tablo_name VARCHAR(30), TabloDescr INT, PRIMARY KEY (Id_tablo), FOREIGN KEY (TabloDescr) REFERENCES Description(Id_Descr));

Приклад SQL запиту для створення початкового запису з назвою табло або додавання нового табло в БД:

INSERT INTO Tablo VALUES (1,"YHF",NULL);

Для зберігання користувачів програми з усіма параметрами в БД розроблена таблиця Users (структура наведена в табл.3.4)

Таблиця 3.4 – Таблиця БД користувачів Users

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_user	Номер запису (ключ таблиці)	INT	
2	Username	Імя користувача	VARCHAR(50)	
3	Access_level	Рівень доступу	INT	
4	Md5_hash	Md5 Хеш пароля	CHAR(32)	
5	CardID	RFID ІД карти (4 байта в HEX)	CHAR(8)	
6	UserDescr	Посилання на додатковий опис	INT	Description(Id_Descr)

Стовпець Id_user є унікальним ідентифікаційним номером таблиці, що дає можливість послання на неї.

Стовпець Username зберігає ім’я користувача.

Стовпець Access_level зберігає рівень доступу.

Рівні доступу:

- значення 0 – супер адміністратор (може все, є меню розробника);

- значення 1 – адміністратор (доступні всі налаштування, редагування довідників);

- значення 2 – оператор (доступно зважування в ручному режимі);

- значення 3 – вагар (доступно тільки автоматичне зважування та робота з пультом).

Стовпець Md5_hash зберігає хеш паролю користувача.

Паролі користувачів саме хешируються, а не шифруються з наступних причин:

а) трудомісткість. Шифрування може тривати довше, а яке перетворення ми б не вибрали, його доведеться пробляти при кожній перевірці пароля. Однією з вимог до хеш-функцій ж є швидкість виконання;

б) довжина вихідних значень. Результат шифрування має змінну довжину, результат хешування – завжди однакову, а зберігати однорідні за розміром дані в базі даних дуже вже зручно. Не кажучи вже про те, що довжина пароля в зашифрованому вигляді буде давати деяку інформацію про довжину вихідного пароля. Однакова довжина, правда, призводить до можливості виникнення колізій, але про це нижче;

в) управління ключами. Для шифрування потрібно ключ, який теж десь доведеться зберігати і сподіватися, що його ніхто не знайде. У будь-якому випадку, генерація і управління ключами це окрема історія (вони не повинні бути слабкими, їх потрібно регулярно міняти і так далі).

Hash – хеш функція – функція однозначного відображення рядка (будь-якої довжини) на кінцеву множину (рядок заданої довжини).

Саме число (рядок) хеш – результат обчислення хеш-функції над даними.

Існують криптографічні та некриптографічні (класифікуються окремо, до них відносяться, наприклад, контрольні суми) хеш-функції.

Для криптографічних хеш є три додаткових умови, які відрізняють їх від всіх інших:

а) незворотність: для заданого значення хеш-функції m повинно бути обчислювально нездійсненно знайти блок даних X, для якого H (X) = m;

б) стійкість до колізій першого роду: для заданого повідомлення M має бути обчислювально нездійсненно підібрати інше повідомлення N, для якого H (N) = H (M);

в) стійкість до колізій другого роду: має бути обчислювально нездійсненно підібрати пару повідомлень ~ (M, M ‘), що мають однаковий хеш.

Стовпець CardID – унікальні 4 байти безконтактної карти для ідентифікації користувача в разі підключення

Стовпець UserDescr зберігає більш великий текстовий опис користувача шляхом посилання на таблицю описів Description, щоб зменшити розмір даних для зберігання та про індексувати поточну таблицю.

SQL запит створення таблиці:

CREATE TABLE Users (Id_user INT, Username VARCHAR(50), Access_level INT, Md5_hash CHAR(32), CardID CHAR(8), UserDescr INT, PRIMARY KEY (Id_user), FOREIGN KEY (UserDescr) REFERENCES Description(Id_Descr));

Для зберігання назв та параметрів вантажів було створено таблицю Cargos. (структура у табл.3.5).

Стовпець Id_cargo є унікальним ідентифікаційним номером таблиці, що дає можливість послання на неї.

Стовпець Cargo_name зберігає назву вантажу.

Стовпець CargoPrice зберігає ціну на вантаж.

Стовпець Cargo_code зберігає код вантажу на підприємстві.

Стовпець Activity використовується для вказання активності запису з точки зору відображення у списках. Використовується для того, щоб коли користувач видаляє запис із списку не було пошкодження зв’язків БД. Насамперед це пов’язано з тим, що на цю таблицю посилається основна таблиця (таблиця зважувань), а отже зважування, що проведені до моменту видалення вантажу мають зберегтись та відображатись навіть коли користувач вирішив зменшити перелік вантажів. Тобто записи з цієї таблиці програмно ніколи не видаляються, а лише в інтерфейсі прячуться від користувача.

Таблиця 3.5 – Таблиця БД найменувань вантажів Cargos

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_cargo	Номер запису (ключ таблиці)	INT	
2	Cargo_name	Назва вантажу	VARCHAR(60)	
3	CargoPrice	Ціна за одиницю товару	FLOAT	
4	Cargo_code	Внутрішній код товару на підприємстві	VARCHAR(10)	
5	Activity	Активність запису (1 – якщо видима користувачеві, 0 – якщо не видима) використовується щоб не видаляти записи та всі давні зважування були пов’язані	INT	
6	CargoDescr	Посилання на додатковий опис	INT	Description(Id_Descr)

SQL запит створення таблиці:

CREATE TABLE Cargos (Id_cargo INT, Cargo_name VARCHAR(60), CargoPrice FLOAT, Cargo_code VARCHAR(10), Activity INT, CargoDescr INT, PRIMARY KEY (Id_cargo), FOREIGN KEY (CargoDescr) REFERENCES Description(Id_Descr));

Для збереження водії та даних при них створено таблицю Drivers. В ній зберігаються прізвище, ім’я, по батькові водія, активність запису (подібно такого ж стовпця в таблиці вантажів) та посилання на більш детальний опис, що може включати і адресу проживання, і номер водійського посвідчення і ще багато чого.

Таблиця 3.6 – Таблиця БД водіїв Drivers

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_driver	Номер запису (ключ таблиці)	INT	
2	driver_name	Імя	VARCHAR(25)	
3	driver_fname	По батькові	VARCHAR(25)	
4	driver_surname	Прізвище	VARCHAR(25)	
5	Activity	Активність запису (1 – якщо видима користувачеві, 0 – якщо не видима) використовується щоб не видаляти записи та всі давні зважування були пов’язані	INT	
6	DriverDescr	Посилання на додатковий опис	INT	Description(Id_Descr)

SQL запит створення таблиці:

CREATE TABLE Drivers (Id_driver INT, driver_name VARCHAR(25), driver_fname VARCHAR(25), driver_surname VARCHAR(35), Activity INT, DriverDescr INT, PRIMARY KEY (Id_driver), FOREIGN KEY (DriverDescr) REFERENCES Description(Id_Descr));

Для зберігання даних щодо відправників продукції було створено таблицю Senders. В цій таблиці зберігаються дані про назву підприємства або ПІБ приватного підприємця, тип підприємства, ЄРДПОУ організації, адресу підприємства та додатковий опис (шляхом посилання на таблицю Description), а також присутній стовпець, що вказує активність запису.

Таблиця 3.7 – Таблиця БД відправників Senders

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_sender	Номер запису (ключ таблиці)	INT	
2	Sender_name	Назва відправника (ПІБ частної особи)	VARCHAR(200)	
3	Sender_type	Тип підприємства	INT	
4	ERDPOU	ЄРДПОУ організації (10 цифр для фіз особи та 8 цифр для юр особи)	VARCHAR(10)	
5	AddressTXT	Адреса в текстовому вигляді	TEXT	
6	Activity	Активність запису (1 – якщо видима користувачеві, 0 – якщо не видима) використовується щоб не видаляти записи та всі давні зважування були пов’язані	INT	
7	SenderDescr	Посилання на додатковий опис	INT	Description(Id_Descr)

SQL запит створення таблиці:

CREATE TABLE Senders (Id_sender INT, Sender_name VARCHAR(200), Sender_type INT, ERDPOU VARCHAR(10), AddressTXT TEXT, Activity INT, SenderDescr INT, PRIMARY KEY (Id_sender), FOREIGN KEY (SenderDescr) REFERENCES Description(Id_Descr));

Для зберігання даних про отримувачів продукції створено таблицю Recipients. Її структура та перелік даних, що зберігається ідентичні до таблиці відправників Senders.

Таблиця 3.8 – Таблиця БД отримувачів Recipients

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_recipient	Номер запису (ключ таблиці)	INT	
2	Recp_name	Назва відправника (ПІБ частої особи)	VARCHAR(200)	
3	Recp_type	Тип підприємства	INT	
4	ERDPOU	ЄРДПОУ організації (10 цифр для фіз осіб ті 8 цифр для юр. осіб)	VARCHAR(10)	
5	RecpAddressTXT	Адреса в текстовому вигляді	TEXT	
6	Activity	Активність запису (1 – якщо видима користувачеві, 0 – якщо не видима) використовується щоб не видаляти записи та всі давні зважування були пов’язані	INT	
7	RecpDescr	Посилання на додатковий опис	INT	Description(Id_Descr)

SQL запит створення таблиці:

CREATE TABLE Recipients (Id_recipient INT, Recp_name VARCHAR(200), Recp_type INT, ERDPOU VARCHAR(10), RecpAddressTXT TEXT, Activity INT, RecpDescr INT, PRIMARY KEY (Id_recipient), FOREIGN KEY (RecpDescr) REFERENCES Description(Id_Descr));

Для зберігання даних про платників за продукцію створено таблицю Payers. Її структура та перелік даних, що зберігається ідентичні до таблиці відправників Senders.

Таблиця 3.9 – Таблиця БД платників Payers

№	Назва стовпця	Що зберігається	Тип даних
1	Id_payer	Номер запису (ключ таблиці)	INT
2	Payer_name	Назва відправника (ПІБ частої особи)	VARCHAR(200)
3	Payer_type	Тип підприємства	INT
4	ERDPOU	ЄРДПОУ організації (10 цифр для фіз осіб ті 8 цифр для юр. осіб)	VARCHAR(10)
5	PayerAddressTXT	Адреса в текстовому вигляді	TEXT
6	Activity	Активність запису (1 – якщо видима користувачеві, 0 – якщо не видима) використовується щоб не видаляти записи та всі давні зважування були пов’язані	INT
7	PayerDescr	Посилання на додатковий опис	INT

SQL запит створення таблиці:

CREATE TABLE Payers (Id_payer INT, Payer_name VARCHAR(200), Payer_type INT, ERDPOU VARCHAR(10), PayerAddressTXT TEXT, Activity INT, PayerDescr INT, PRIMARY KEY (Id_payer), FOREIGN KEY (PayerDescr) REFERENCES Description(Id_Descr));

Для зберігання даних про перевізників продукції створено таблицю Cariers. Її структура та перелік даних, що зберігається ідентичні до таблиці відправників Senders.

Таблиця 3.10 – Таблиця БД перевізників Cariers

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_carier	Номер запису (ключ таблиці)	INT	
2	Carier_name	Назва відправника (ПІБ частої особи)	VARCHAR(200)	
3	Carier_type	Тип підприємства	INT	
4	ERDPOU	ЄРДПОУ організації (10 цифр для фіз осіб ті 8 цифр для юр. осіб)	VARCHAR(10)	
5	CarierAddressTXT	Адреса в текстовому вигляді	TEXT	
6	Activity	Номер запису (ключ таблиці)	INT	
7	CarierDescr	Посилання на додатковий опис	INT	Description(Id_Descr)

SQL запит створення таблиці:

CREATE TABLE Cariers (Id_carier INT, Carier_name VARCHAR(200), Carier_type INT, ERDPOU VARCHAR(10), CarierAddressTXT TEXT, Activity INT, CarierDescr INT, PRIMARY KEY (Id_carier), FOREIGN KEY (CarierDescr) REFERENCES Description(Id_Descr));

Для зберігання даних про причепи машин створено таблицю Trailers в якій зберігається номерний знак причепу, активність запису та посилання на додатковий опис.

Таблиця 3.11 – Таблиця БД причепів Trailers

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_Trailer	Номер запису (ключ таблиці)	INT	
2	Trailer_Numb	Номер причепу	VARCHAR(10)	
3	Activity	Активність (видалений запис чи використовується)	INT	
4	TrlDescr	Посилання на додатковий опис	INT	Description(Id_Descr)

SQL запит створення таблиці:

CREATE TABLE Trailers (Id_Trailer INT, Trailer_Numb VARCHAR(10), Activity INT, TrlDescr INT, PRIMARY KEY (Id_Trailer), FOREIGN KEY (TrlDescr) REFERENCES Description(Id_Descr));

Для зберігання даних щодо машин, які привозять та вивозять зернові з елеватору створено таблицю Cars. В цій таблиці зберігаються номерний знак автомобіля, модель автомобіля, колір автомобіля, значення маси тари за замовчуванням, посилання на водія, прицеп, відправника, отримувача, платника, одержувача продукції. Посилання дозволяють після ручного або автоматичного введення номерного знаку автомобіля автоматично підтягувати додаткові дані – такі як причеп, ПІБ водія, найменування всіх контрагентів – це значно пришвидшує процес внесення даних в програму.

Таблиця 3.12 – Таблиця БД машин Cars

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_car	Номер запису (ключ таблиці)	INT	
2	car_numb	Номерний знак машини	VARCHAR(8)	
3	car_mark	Модель машини	VARCHAR(30)	
4	Car_color	Колір машини	VARCHAR(15)	
5	DefaultTara	Тара за замовчуванням	FLOAT	
6	Driver_numb	Посилання на водія	INT	Drivers(Id_driver)
7	SenderLink	Посилання на відправника на отправителя	INT	Senders(Id_sender)
8	RecpLink	Посилання на отримувача	INT	Recipients(Id_recipient)
9	PayerLink	Посилання на платника	INT	Payers(Id_payer)
10	CarierLink	Посилання на перевізника	INT	Cariers(Id_carier)
11	TrlLink	Посилання на причеп	INT	Trailers(Id_Trailer)
12	Activity	Активність (видалений запис чи використовується)	INT	
13	CarDescr	Посилання на додатковий опис	INT	Description(Id_Descr)

SQL запит створення таблиці:

CREATE TABLE Cars (Id_car INT, car_numb VARCHAR(8), car_mark VARCHAR(30), Car_color VARCHAR(15), DefaultTara FLOAT, Driver_numb INT,

SenderLink INT, RecpLink INT, PayerLink INT, CarierLink INT, TrlLink INT, Activity INT, CarDescr INT, PRIMARY KEY (Id_car), FOREIGN KEY (Driver_numb) REFERENCES Drivers (Id_driver), FOREIGN KEY (SenderLink) REFERENCES Senders(Id_sender), FOREIGN KEY (RecpLink) REFERENCES Recipients(Id_recipient), FOREIGN KEY (PayerLink) REFERENCES Payers(Id_payer), FOREIGN KEY (CarierLink) REFERENCES Cariers(Id_carier), FOREIGN KEY (TrlLink) REFERENCES Trailers(Id_Trailer), FOREIGN KEY (CarDescr) REFERENCES Description(Id_Descr));

Для зберігання налаштувань камер, що підключаються до комплексу створено таблицю Cameras, що зберігає IP адресу камери в локальній мережі, HTTP запит для отримання кадра, імя та пароль адміністратора, режим авторизації камери.

Таблиця 3.13 – Таблиця БД налаштувань камер Cameras

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_camera	Номер запису (ключ таблиці)	INT	
2	IPaddr	IP адреса камери	VARCHAR(20)	
3	Get_quary	Запит для отримання кадра	VARCHAR(250)	
4	User_name	Імя адміністратора	VARCHAR(50)	
5	Access_psw	Пароль доступа	VARCHAR(50)	
6	Mode	Режим роботи та авторизації камер	INT	
7	CamDescr	Посилання на додатковий опис	INT	Description(Id_Descr)

Режими авторизації камери можуть мати наступні значення Mode

- значення 0 – базова авторизація без розпізнавання
- значення 1 – Digest авторизація без розпізнавання

SQL запит створення таблиці:

CREATE TABLE Cameras (Id_camera INT, IPaddr VARCHAR(20), Get_quary VARCHAR(250), User_name VARCHAR(50), Access_psw VARCHAR(50), Mode INT, CamDescr INT, PRIMARY KEY (Id_camera), FOREIGN KEY (CamDescr) REFERENCES Description(Id_Descr));

Додатково була створена таблиця контролю версій Version, що не пов’язана з жодною іншою таблицею БД та має завжди лише один запис. Використовується для того, щоб можна було у віддаленого клієнта автоматично оновити БД при появі нової версії програмного забезпечення.

Таблиця 3.14 – Таблиця БД контролю версій Version

№	Назва стовпця	Що зберігається	Тип даних	Приклад заповнення
1	Id_Version	Номер запису (ключ таблиці)	INT	1
2	Version_Numb	Номер версії	INT	2
3	Version_Name	Імя версії	VARCHAR(50)	Extended
4	Version_Date	Дата версії	DATETIME	2020-06-10 12:30:00

SQL запит створення таблиці:

CREATE TABLE Version (Id_Version INT, Version_Numb INT, Version_Name VARCHAR(50), Version_Date DATETIME, PRIMARY KEY (Id_Version));

Задаємо початкове значення SQL запитом:

INSERT INTO Version VALUES (1,1, NULL, “2020-02-04 12:30:00”);

Для додаткового контролю з боку власника підприємства над запобіганням несанкціонованих операцій з боку користувачів комплексу в системі передбачено контроль дій користувачів, що дозволяє у випадку виникнення сумнівів провести перегляд всіх дій, що проводил той чи інший користувач за певний період часу. Для цього в БД створено таблицю UserLogs. В таблиці зберігається посилання на користувача, який виконав дію, дата та час дії, а також вказання елементу над яким проводилася дія та опис самої дії.

Таблиця 3.15 – Таблиця БД журналу дій користувачів UserLogs

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилаєтьс я	Приклад заповнення
1	Id_Log	Номер запису (ключ таблиці)	INT		1
2	FormLog	Над якою формою була проведена дія	VARCHAR(50)		Form_Main
3	ElementLog	Елемент інтерфейсу	VARCHAR(50)		Button_Save
4	EventLog	Дія над елементом	VARCHAR(30)		Click
5	UserLog	Посилання на користувача	INT	Users(Id_us er)	2
6	EvDate	Дата дії	DATE		10.10.2020
7	EvTime	Час дії	TIME		12:23:24

SQL запит створення таблиці:

CREATE TABLE UserLogs (Id_Log INT, FormLog VARCHAR(50), ElementLog VARCHAR(50), EventLog VARCHAR(30), UserLog INT, EvDate DATE, EvTime TIME, PRIMARY KEY (Id_Log), FOREIGN KEY (UserLog) REFERENCES Users(Id_user));

Основною таблицею бази даних є таблиця зважувань Weighing оскільки вона зберіга в собі всі параметри прийому та відправлення вантажів та пов’язана з більшістю інших таблиць БД. Вона зберігає дані про найменування ваг, масу тари, брутто та нето автомобіля, посилання на автомобіль, водія, причеп, відправника, отримувача, платника, перевізника. В ній вказується користувач, який проводив операцію, дата та час першого та другого зважування для встановлення маси нетто, тип проведеної операції, номер або назва документа, чкий був сформований на основі проведення цієї операції.

Таблиця 3.16 – Таблиця БД зважуваннь Weighing

№	Назва стовпця	Що зберігається	Тип даних
1	Id_Wght	Номер запису (ключ таблиці)	INT
2	ScaleID	ІД ваг (при відправленні зважувань на сервер та для комплексного перегляду)	CHAR(5)
3	DateIn	Дата заїзду	DATE
4	TimeIn	Час заїзду	TIME
5	DateOut	Дата виїзду	DATE
6	TimeOut	Час виїзду	TIME
7	OpType	Тип зважування (0– завантаження / тарування перше зважування 2 - друге, 1 – розвантаження /брутування, (перше, 3 – друге)	INT
8	Brutto	Брутто	FLOAT
9	Netto	Нетто	FLOAT
10	Tara	Тара	FLOAT
11	CorrectBrutto	Корректування Брутто	FLOAT
12	CorrectTara	Корректування Тари	FLOAT
13	WCargo	Посилання на вантаж	INT
14	WCar	Посилання на машину	INT
15	WDriver	Посилання на водія	INT
16	WSender	Посилання на відправника	INT
17	WRecp	Посилання на отримувача	INT
18	WPayer	Посилання на платника	INT
19	WCarier	Посилання на перевізника	INT
20	WTrl	Посилання на причеп	INT
21	WUser	Посилання на користувача	INT
22	WDoc	Супроводжуючий документ	VARCHAR(40)

Продовження таблиці 3.16

№	Назва стовпця	Що зберігається	Тип даних
23	Sended	Ознака відправлення (0 – не відправлено, 1- відправлено тільки на сервер, 2 – відправлено тільки у телеграмм, 3 – сервер+телеграмм)	INT
24	Activity	0 видалено, 1 активно	INT
25	WghtDescr	Посилання на додатковий опис	INT

SQL запит створення таблиці:

CREATE TABLE Weighing (Id_Wght INT, ScaleID CHAR(5), DateIn DATE, TimeIn TIME, DateOut DATE, TimeOut TIME, OpType INT, Brutto FLOAT, Netto FLOAT, Tara FLOAT, CorrectBrutto FLOAT, CorrectTara FLOAT, WCargo INT, WCar INT, WDriver INT, WSender INT, WRecp INT, WPayer INT, WCarier INT, WTrl INT, WUser INT, WDoc VARCHAR(40), Sended INT, Activity INT, WghtDescr INT, PRIMARY KEY (Id_Wght), FOREIGN KEY (WCargo) REFERENCES Cargos(Id_cargo), FOREIGN KEY (WCar) REFERENCES Cars(Id_car), FOREIGN KEY (WDriver) REFERENCES Drivers(Id_driver), FOREIGN KEY (WSender) REFERENCES Senders(Id_sender), FOREIGN KEY (WRecp) REFERENCES Recipients(Id_recipient), FOREIGN KEY (WPayer) REFERENCES Payers(Id_payer), FOREIGN KEY (WCarier) REFERENCES Cariers(Id_carier), FOREIGN KEY (WTrl) REFERENCES Trailers(Id_Trailer), FOREIGN KEY (WTrl) REFERENCES Trailers(Id_Trailer), FOREIGN KEY (WUser) REFERENCES Users(Id_user), FOREIGN KEY (WghtDescr) REFERENCES Description(Id_Descr));

Якщо під час роботи комплексу для фото фіксації використовувались камери, для зберігання відповідності зроблений фото операціям зважування створена таблиця відповідності фото зважуванням PhotoWght. Призначення цієї таблиці полягає в тому, що при проведені зважування може проводитися фото фіксація з невідомої заздалегідь кількості камер або може не проводитись взагалі. А отже заздалегідь не відомо скільки файлів з фото будуть відповідати кожному

зважуванню та не має сенсу зберігати посилання на фото в основній таблиці зважувань. Ця таблиця зберігає посилання на проведене зважування, посилання на камеру як зробила фото фіксацію та назву файла з фотографією. Використовуючи пошук по параметрах БД можна швидко отримати перелік фотографій для зважування, якщо такі роболися та не займати додаткове місце в БД, якщо фото фіксації не було.

Таблиця 3.17 – Таблиця БД відповідність фото проведеним зважуванням PhotoWght

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_PW	Номер запису (ключ таблиці)	INT	
2	WghtId	Посилання на зважування	INT	Weighing(Id_Wght)
3	PhotoFileName	Розташування фото на диску	VARCHAR(250)	
4	CamNumb	Посилання на камеру яка зробила фото	INT	Cameras(Id_camera)

SQL запит створення таблиці:

CREATE TABLE PhotoWght (Id_PW INT, WghtId INT, PhotoFileName VARCHAR(250), CamNumb INT, PRIMARY KEY (Id_PW), FOREIGN KEY (WghtId) REFERENCES Weighing(Id_Wght), FOREIGN KEY (CamNumb) REFERENCES Cameras(Id_camera));

Якщо користувач використовує в комплексі авторизацію і автофіксацію зважувань за допомогою безконтактних міток важливо зберігати інформацію про ці картки та їх відповідність машині, причепу, вантажу, відправнику, отримувачу, платнику та або перевізнику. Для цього створена таблиця RFID карт RFIDCards

Таблиця 3.18 – Таблиця БД RFID карт RFIDCards

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_RFID	Номер запису (ключ таблиці)	INT	
2	CardID	ІД карти	INT	VARCHAR(20)
3	DriverId	Посилання на водія	INT	Drivers(Id_driver)
4	CarId	Посилання на машину	INT	Cars(Id_car)
5	TrailerId	Посилання на причеп	INT	Trailers(Id_Trailer)
6	SenderId	Посилання на відправника	INT	Senders(Id_sender)
7	RecpId	Посилання на отримувача	INT	Recipients(Id_recipient)
8	PayerId	Посилання на платника	INT	Payers(Id_payer)
9	CarierId	Посилання на перевізника	INT	Cariers(Id_carier)
10	CargoId	Посилання на вантаж	INT	Cargos(Id_cargo)
11	WghtDescr	Посилання на додатковий опис	INT	Description(Id_Descr)
12	Activity	Стан (0 – видалена, 1-активна)	INT	

SQL запит створення таблиці:

CREATE TABLE RFIDCards (Id_RFID INT, CardID VARCHAR(20), DriverId INT, CarId INT, TrailerId INT, SenderId INT, RecpId INT, PayerId INT, CarierId INT, CargoId INT, WghtDescr INT, Activity INT, PRIMARY KEY (Id_RFID), FOREIGN KEY (DriverId) REFERENCES Drivers(Id_driver), FOREIGN KEY (CarId) REFERENCES Cars(Id_car), FOREIGN KEY (TrailerId) REFERENCES Trailers(Id_Trailer), FOREIGN KEY (SenderId) REFERENCES Senders(Id_sender), FOREIGN KEY (RecpId) REFERENCES Recipients(Id_recipient), FOREIGN KEY

(PayerId) REFERENCES Payers(Id_payer), FOREIGN KEY (CarierId) REFERENCES Cariers(Id_carier), FOREIGN KEY (CargoId) REFERENCES Cargos(Id_cargo), FOREIGN KEY (WghtDescr) REFERENCES Description(Id_Descr));

Оскільки комплекс пропонується продавати та використовувати не лише в Україні, а й в інших країнах – дуже важливо мати можливість перекладу інтерфейсу на різні мови світу. Для зберігання переліку мов інтерфейсу створена таблиця Languages.

Таблиця 3.19 – Таблиця БД мови інтерфейсу Languages

№	Назва стовпця	Що зберігається	Тип даних	Приклад заповнення
1	Id_Lang	Номер запису (ключ таблиці)	INT	1
2	LanguageName	Нава мови в міжнародній класифікації ISO 639-3	VARCHAR(5)	“rus”
3	LangName	Назва мови на ній самій	VARCHAR(50)	“русский”

SQL запит створення таблиці:

CREATE TABLE Languages (Id_Lang INT, LanguageName VARCHAR(5), LangName VARCHAR(50), PRIMARY KEY (Id_Lang));

Для безпосереднього перекладу інтерфейсу на різні мови необхідно зберігати написи та їх відповідність елементам інтерфейсу та мові для яких ці написи призначені. Для цього в БД створена таблиця Inscriptions

Таблиця 3.20 – Таблиця БД для написів інтерфейсу Inscriptions

№	Назва стовпця	Що зберігається	Тип даних	На які таблиці посилається
1	Id_Inscr	Номер записи (ключ таблицы)	INT	
2	ElementName	Имя элемента	VARCHAR(150)	
3	Language	Язык надписи	INT	Languages(Id_Lang)
4	Inscr	Надпись ссылка на таблицу описаний	INT	Description(Id_Descr)

SQL запит створення таблиці:

CREATE TABLE Inscriptions (Id_Inscr INT, ElementName VARCHAR(150), Language INT, Inscr INT, PRIMARY KEY (Id_Inscr), FOREIGN KEY (Language) REFERENCES Languages(Id_Lang), FOREIGN KEY (Inscr) REFERENCES Description(Id_Descr));

3.2 Розробка серверної частини системи

Після розробки структури бази даних наступним етапом при створенні комплексу була розробка серверної частини, що дозволяє віддалено відправляти для перегляду та створення звітів даних отриманих на ваговій та у лабораторії елеваторів. Первим етапом для створення серверної частини є розробка протоколу обміна між програмними елементами комплексу. В якості базової технології формування повідомлень між програмними елементами було обрано JSON.

JSON - простий, заснований на використанні тексту, спосіб зберігати і передавати структуровані дані. За допомогою простого синтаксису ви можете легко зберігати все, що завгодно, починаючи від одного числа до рядків, масивів і об'єктів, в простому тексті. Також можна пов'язувати між собою масиви і об'єкти, створюючи складні структури даних.

Після створення рядка JSON, її легко відправити іншому додатку або в інше місце мережі, так як вона являє собою простий текст.

JSON має такі переваги:

- він компактний.

- його пропозиції легко читаються і складаються як людиною, так і комп'ютером.

- його легко перетворити в структуру даних для більшості мов програмування (числа, рядки, логічні змінні, масиви і так далі)

- багато мов програмування мають функції і бібліотеки для читання і створення структур JSON.

Назва JSON означає JavaScript Object Notation (уявлення об'єктів JavaScript). Як і представляє ім'я, він заснований на способі визначення об'єктів (дуже схоже на створення асоціативних масивів в інших мовах) і масивів.

Серверна частина комплексу складається з 9 скриптів, кожен з яких виконує свою функцію. Опис відповідності назв скриптів їх функціоналу наведено у таблиці 3.21.

Скрипти були написані на мові PHP.

Структура запиту:

http://avrobit.org/WghtOut.php?ScID=A2004&Cmd=101&From=0

ScID – ІД об’єкта на сервері;

Cmd – код команди;

From – починаючи з якого внутрішнього ІД події (щоб не отримувати кожен раз весь перелік подій після попередньої синхронізації)

Ответ: JSON

Приклад відповіді скрипта на запит:

{"Id_cargo":"38", "IdScales":"11", "IdOnObject":"1", "Cargo_name":"\u042f\u0427\u041c\u0415\u041d\u042c", "CargoPrice":"0", "Cargo_code":null, "Activity":"1", "CargoDescr":null}

Треба зазначити, що оскільки комплекс може використовувати будь які мови для зберігання інформації при відправленні та отриманні даних всі текстові

значення передаються в кодировці UTF-8, що має можливість зберігати символи усіх існуючих мов.

Таблиця 3.21 – Скрипти сервера з описом функціоналу.

№	Скрипт	Опис
1	NewWghtAuto.php	Отримання поточної ваги з ваг або програми та збереження в файлі с ІД
2	TelSend.php	Відправлення текстового повідомлення в телеграм
3	TelSendDoc.php	Відправлення документу в телеграм
4	TelSendPhoto.php	Відправлення фото в телеграм
5	WebTerminal.php	Отримання поточних показань термінала по ІД та ключу
6	NewWght.php	Отримання параметрів зважувань та інших супутніх вимірювань в локальних БД
7	GetWghts.php	Онлайн сторінка перегляду показань на об’єктах
8	WghtOut.php	Отримання даних по об’єктам термінальною програмою
9	Update.php	Доступні версії автоматичного оновлення

Скрипт WghtOut.php – отримання даних щодо проведених операцій на об’єктах.

Таблиця 3.22 – Команди скрипта WghtOut.php

№	Код	Опис
1	101	Синхронізація таблиці вантажів
2	102	Синхронізація таблиці водіїв
3	103	Синхронізація таблиці відправників
4	104	Синхронізація таблиці отримувачів
5	105	Синхронізація таблиці платників

Продовження таблиці 3.22

№	Код	Опис
6	106	Синхронізація таблиці перевізників
7	107	Синхронізація таблиці причепів
8	108	Синхронізація таблиці машин
9	109	Синхронізація таблиці камер
10	110	Синхронізація таблиці зважувань
11	111	Синхронізація таблиці відповідності зважувань фотографіям
12	112	Отримання з сервера файлц Crt.dat

Скрипт NewWght.php – отримання сервером даних від елеваторів для збереження у глобальній базі даних. Текст програми скрипта NewWght.php нведено у додатку А.

Таблиця 3.23 – Команди скрипта NewWght.php

№	Код	Опис
Команди синхронізації таблиць БД		
1	101	Синхронізація таблиці вантажів
2	102	Синхронізація таблиці водіїв
3	103	Синхронізація таблиці відправників
4	104	Синхронізація таблиці отримувачів
5	105	Синхронізація таблиці платників
6	106	Синхронізація таблиці перевізників
7	107	Синхронізація таблиці причепів
8	108	Синхронізація таблиці машин
9	109	Синхронізація таблиці камер
10	110	Синхронізація таблиці зважувань
11	111	Синхронізація таблиці відповідності зважувань фотографіям

Продовження таблиці 3.23

Команди створення нових записів та файлів налаштувань		
№	Код	Опис
12	201	Створення нового вантажу
13	202	Створення нового водія
14	203	Створення нового відправника
15	204	Створення нового отримувача
16	205	Створення нового платника
17	206	Створення нового перевізника
18	207	Створення нового причепа
19	208	Створення нової машини
20	209	Створення нової камери
21	210	Нове зважування
22	211	Створення нової фотографії для зважування
23	212	Створити або замінити файл Crt.dat
24	213	Створити або замінити файл DataSend.dat
25	214	Створити або замінити файл equip.dat
26	215	Створити або замінити файл settings.dat
27	216	Створити або замінити файл COMSet.dat
28	217	Створити або замінити файл Sync.dat
Команди оновлення записів в БД на сервері		
№	Код	Опис
29	301	Оновити вантаж
30	302	Оновити водія
31	303	Оновити відправника
32	304	Оновити отримувача
33	305	Оновити платника
34	306	Оновити перевізника
35	307	Оновити причеп

Продовження таблиці 3.23

№	Код	Опис
36	308	Оновити машину
37	309	Оновити камеру
38	310	Оновити зважування
39	311	Оновити фотографії для зважування
Команди перевірки змін з боку сервера		
40	400	Отримати код змін проведених сервером для поточного об’єкта
41	401	Отримати зміни таблиці грузов
42	402	Отримати зміни таблиці водителей
43	403	Отримати зміни таблиці отправителей
44	404	Отримати зміни таблиці получателей
45	405	Отримати зміни таблиці плательщиков
46	406	Отримати зміни таблиці перевозчиков
47	407	Отримати зміни таблиці прицепов
48	408	Отримати зміни таблиці машин
49	409	Отримати зміни таблиці камер
50	410	Отримати зміни таблиці взвешиваний
51	411	Отримати зміни таблиці таблицы
52	412	Отримати змінений файл Crt.dat
53	413	Отримати змінений файл DataSend.dat
54	414	Отримати змінений файл equip.dat
55	415	Отримати змінений файл settings.dat
56	416	Отримати змінений файл COMSet.dat
57	417	Отримати змінений файл Sync.dat
58	499	Виправити код змін після виконання дій команди 400

Більш детально опис команди необхідності проведення змін (команда 400) описано у таблиці 3.24

Таблиця 3.24 – Команда 400 коди змін – відпровляється одним числом

цифра	Опис змін
1я цифра	Внесені зміни в таблицю вантажів 0 – не внесені; 1 –змінена один запис; 2 –змінено декілька записів; 3 – добавлена один запис; 4 – добавлено декілька записів; 5 – видалена один запис; 6 – видалено декілька записів
2я цифра	Внесені зміни в таблицю водителей
3я цифра	Внесені зміни в таблицю отправителей
4я цифра	Внесені зміни в таблицю получателей
5я цифра	Внесені зміни в таблицю плательщиков
6я цифра	Внесені зміни в таблицю перевозчиков
7я цифра	Внесені зміни в таблицю прицепов
8я цифра	Внесені зміни в таблицю машин
9я цифра	Внесені зміни в таблицю камер
10я цифра	Внесені зміни в таблицю взвешиваний
11я цифра	Внесені зміни в таблицю соответствия фотографий
12я цифра	Була зміна налаштувань граничної засмітненості (файл Crt.dat) 0 – не було; 1 – було; 2 – запит поточних налаштувань у ваговій частині
13я цифра	Внесені зміни в налаштування відправлення даних (DataSend.dat) 0 – не внесені; 1 – внесені; 2 – запит поточних налаштувань у ваговій частині; 3 – запит на одноразову синхронізацію БД
14я цифра	Внесені зміни в налаштування силового обладнання та камер (equip.dat)
15я цифра	Внесені зміни в налаштування режимів зєднання з вагами та табло (settings.dat)
16я цифра	Внесені зміни в налаштування COM портов (COMSet.dat)
17я цифра	Внесені зміни в налаштування синхронізації (Sync.dat)

Приклад синхронізації для скрипту NewWght.php (синхронізація зважувань):

{“Command”:110, “IdScale”:”XXXXX”, “Wghts”:[{“IdOnObject”:XXX, “DateIn”:”xxxxx”, “TimeIn”:”XXXX”, “DateOut”:”XXXXXX”, “TimeOut”:”XXXXX”,

“OpType”:X, “Brutto”:X, “Netto”:X, “Tara”:X, “CorrectBrutto”:X, “CorrectTara”:X, “WCargo”:X, “WCar”:X, “WDriver”:X, “WSender”:X, “WRecp”:X, “WPayer”:X, “WCarier”:X, “WTrl”:X, “WDoc”:”XXX”, “Activity”:X},{}…]}

Основною функцією для формування відповідей на запит скрипта WghtOut.php є функція CreateJSONAnsw($dbcnx,'Cars',$idindb,$From,$To);

Приклад відповіді на запит синхронізації (команда 110) по зважуванню з ІД=1651:

{"Id_Wght":"1651", "IdScales":"11", "DateIn":"2020-07-04", "TimeIn":"14:09:30", "DateOut":"2020-07-04", "TimeOut":"14:51:18", "OpType":"3", "Brutto":"12560", "Netto":"7260", "Tara":"5300", "CorrectBrutto":"380", "CorrectTara":"0", "WCargo":"44", "WCar":"545", "WDriver":null, "WSender":"227", "WRecp":"112", "WPayer":"263", "WCarier":"343", "WTrl":null, "WUser":null, "WDoc":"474741", "IdOnObject":"119", "Activity":"1", "WghtDescr":null}

3.3 Розробка програм для вагового обліку та віддалено контролю

Розробка програм для вагового обліку та віддалено контролю проводилася з використання мови Object Pascal у середовищі розробки Delphi (рис.3.2).

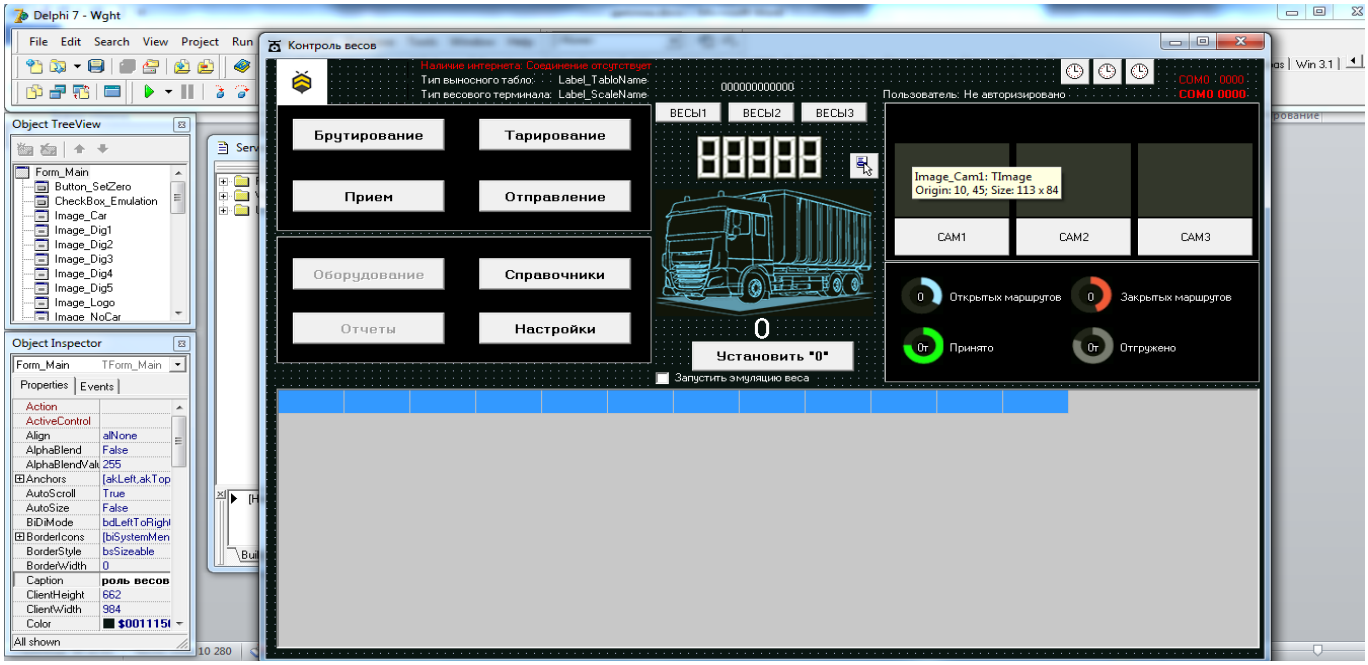

Рисунок 3.2 – Середовище розробки Delphi

Під час написання програмного забезпечення вагового обліку було створено 27 модулів:

- appearance;
- auth;
- bindex;
- cardOfCargo;
- catalogs;
- dBWork;
- emulation;
- euipKey;
- equipment;
- HBMDigital;
- IPCams;
- Main;
- md5hash;
- ParametersSetings;
- RecieveTransmit;
- Reports;
- RFID;
- Scales;
- SendToServer;
- ServerSync;
- Settings;
- Size;
- Sync;
- Tablo;
- TTNCreate;
- UsersSetup;
- VKSocket.

Основним модулем програми вагової частини є модуль Main, що викликається при запуску програми. Він має прив’язано до нього форму (рис.3.3).

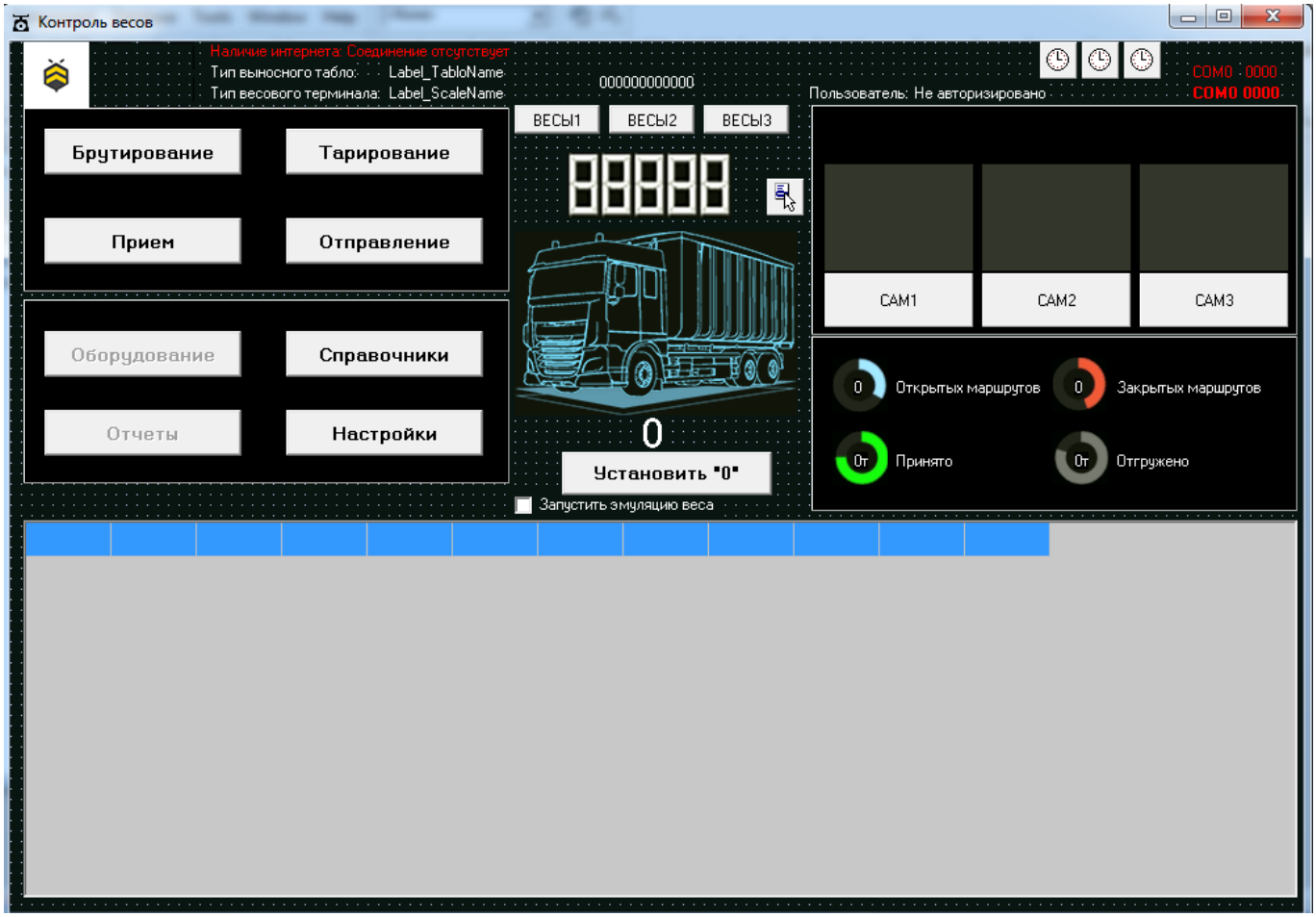

Рисунок 3.3 – Форма модуля Main

У програм передбачено захист від несанкціонованого копіювання щляхом програмної привязки до серійного номера жорсткого диска. Для цього було створено модулі bindex та euipKey, що не мають окремих форм однак проводять перевірку наявності ключового файлу та за допомогою алгоритму з відкритим ключом (яким у даному випадку є серійний номер жорсткого диску) відповідності змісту цього файлу.

Для налаштувань зовнішнього вигляду (мова, колір форм, написів, ярлика програми) використовується модуль appearance з власною формою (рис.3.4)

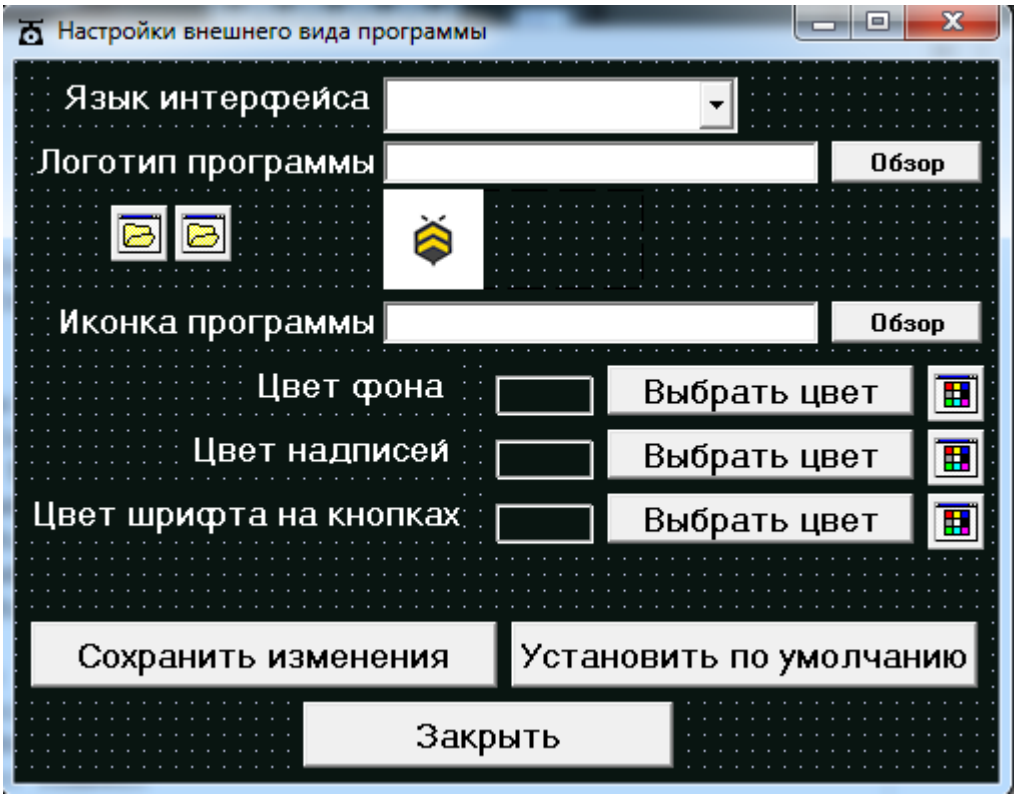

Рисунок 3.4 – Форма модуля appearance

Модуль Auth використовується для авторизації користувачів при вході у прогами. Модуль має власну форму (рис.3.5) та використовує функції модуля md5hash для перевірки введено пароля.

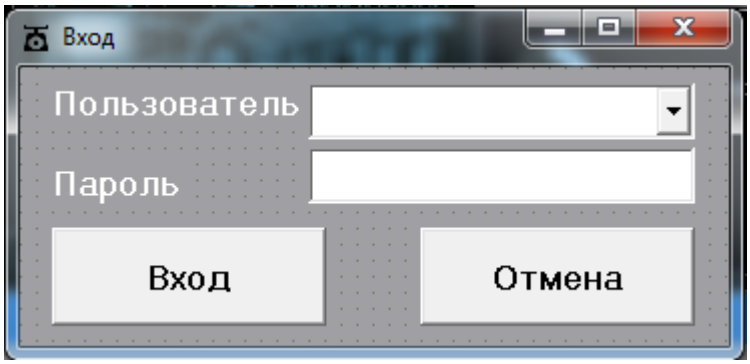

Рисунок 3.5 – Форма модуля Auth

Для можливостей внесення змін та доповнень у запис про зважування використовується модуль CardOfCargo з власною формою (рис.3.6).

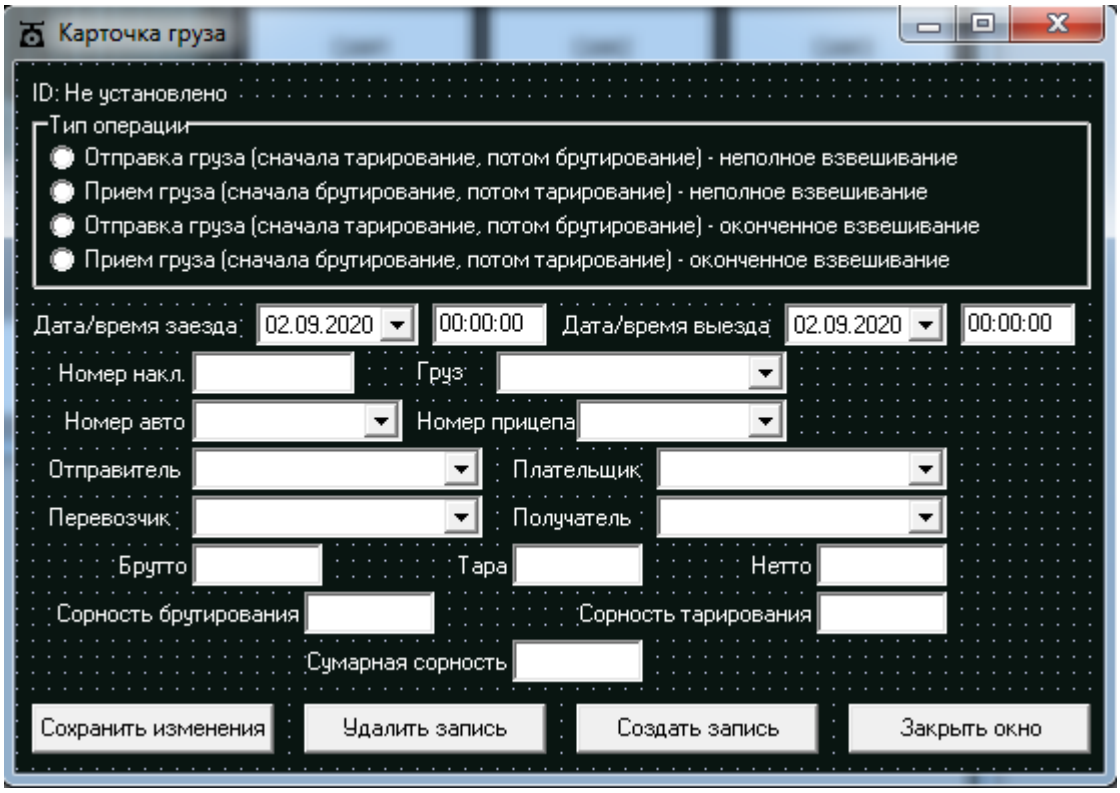

Рисунок 3.6 – Форма модуля CardOfCargo

Для додавання вручну даних у довідники відправників. Отримувачів, платників, перевізників та вантажів (і відповідні таблиці БД) використовується модуль Catalogs з власно. Формою (рис.3.7).

Рисунок 3.7 – Форма модуля Catalogs

Оскільки в процесі написання та від лагодження комплексу у розробника не завжди є апаратний доступ до реальних автомобільних ваг в програмі створено модуль Emulation з власною формою (рис.3.8). Цей модуль дозволяє видати бажане значення ваги для тестування роботи системи у період від лагодження.

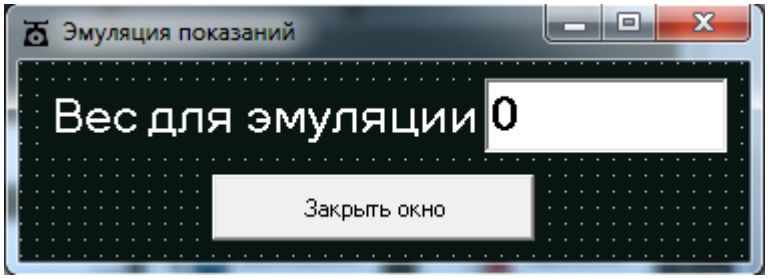

Рисунок 3.8 – Форма модуля Emulation

Для налаштувань режимів роботи обладнання (камер, світлофорів, шлагбаумів, датчиків положення, безконтактних карт ідентифікації транспорта) створено модуль Equipment з власною формою, що має декілька заклади нок (рис.3.9 – рис.3.10).

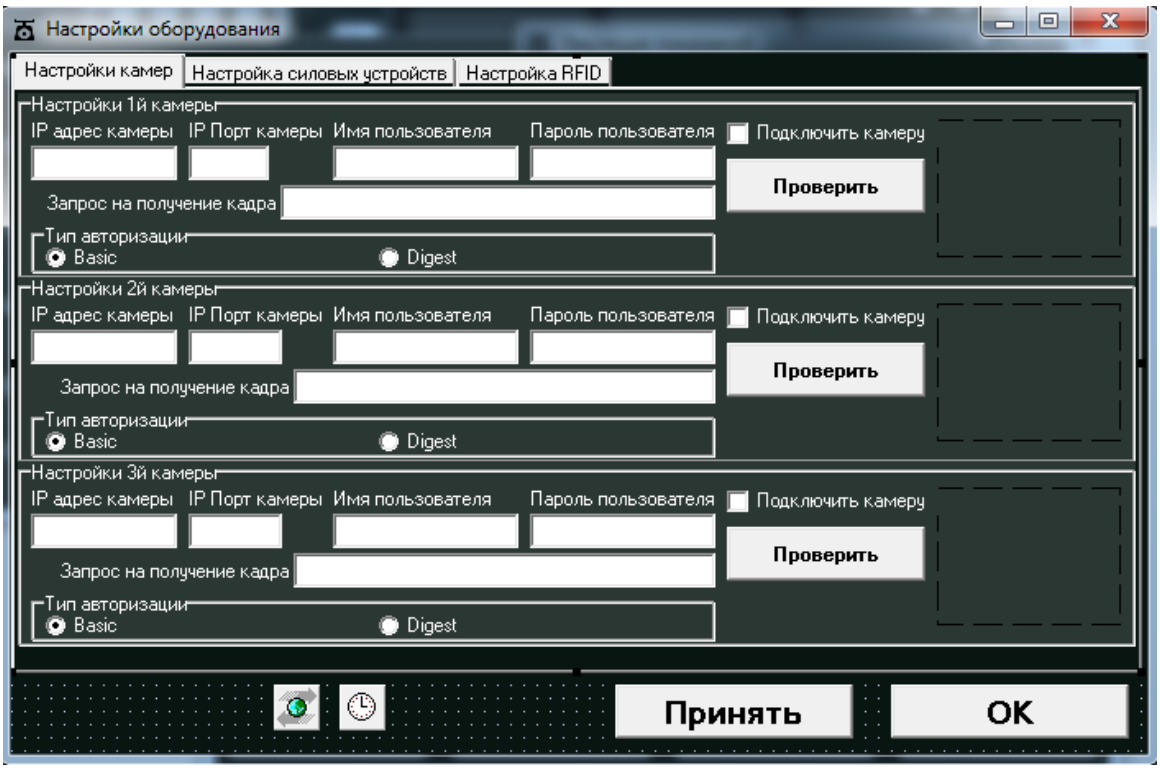

Рисунок 3.9 – Форма модуля Equipment (налаштування камер)

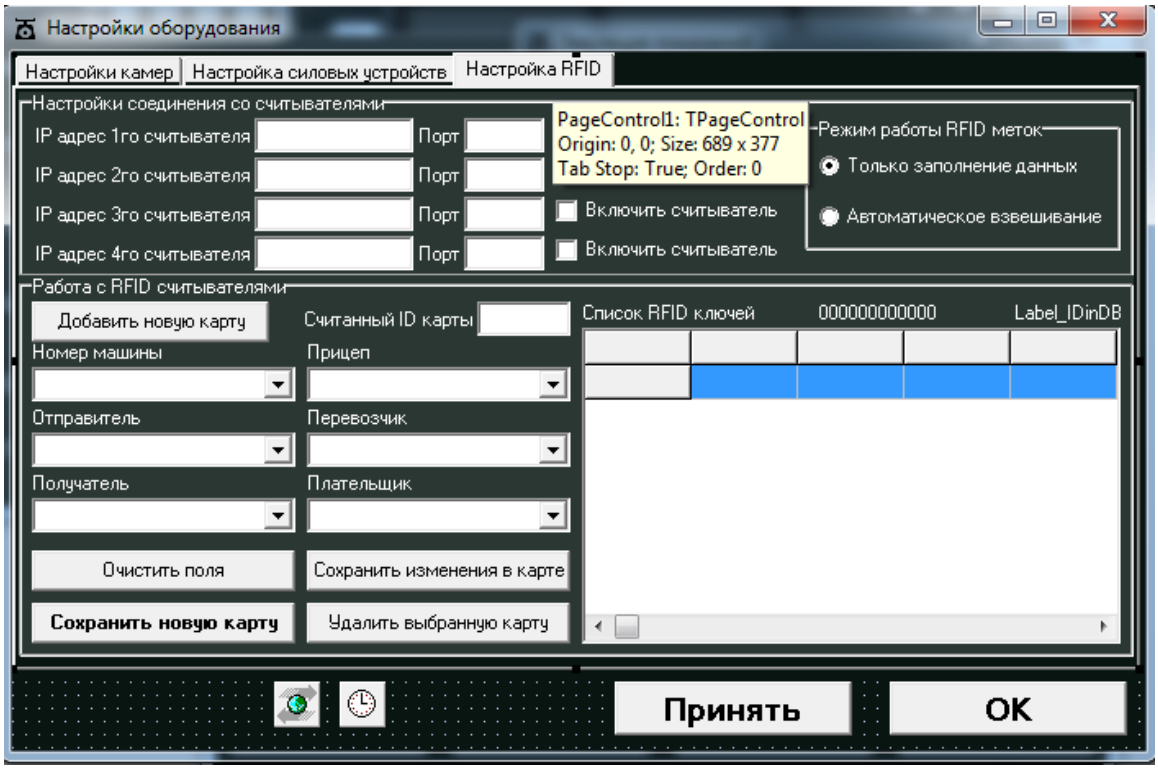

Рисунок 3.10 – Форма модуля Equipment (налаштування безконтактних міток)

Для створення звітів по різним показникам за різні періоди, а також для екпорту даних у інші прогами, друку результатів звітів, створення та друку товарно-транспортних накладних було створено модуль Reports з формою (рис.3.11)

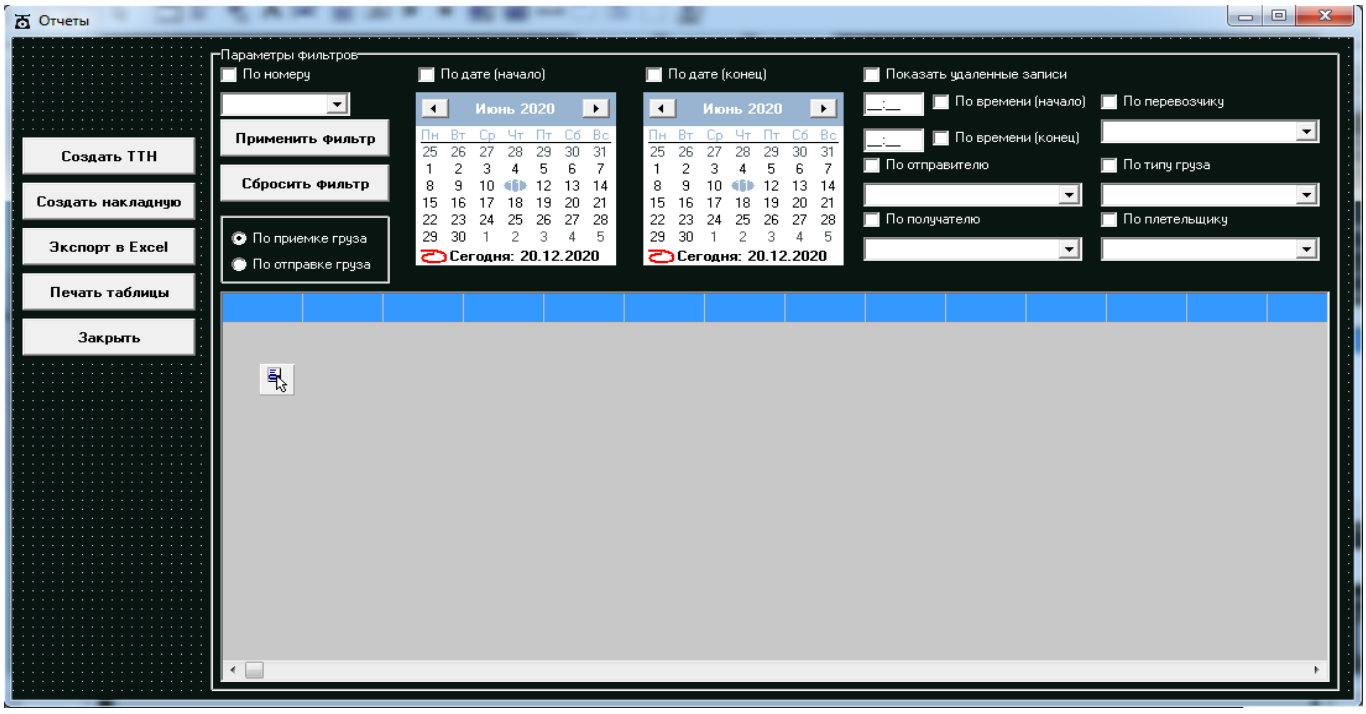

Рисунок 3.11 – Форма модуля Reports

Для введення даних по автомобілю, що проходить зважування в програмі було створено модуль RecieveTransmit з власною формою (рис.3.12).

Рисунок 3.12 – Форма модуля RecieveTransmit

Для роботи з налаштуваннями основних зєднань (ваговий термінал та виносний індикатор), налаштуваннями відправлення даних (відправлення на сервер та у телеграм), а також налаштувань цифровіх датчиків та відображення їх стану було створено модуль Settings, що має власну форму (рис.3.13-рис.3.15), що має декілька заклади нок, а також може викликати налаштування зовнішнього вигляду програми та налаштування користувачів.

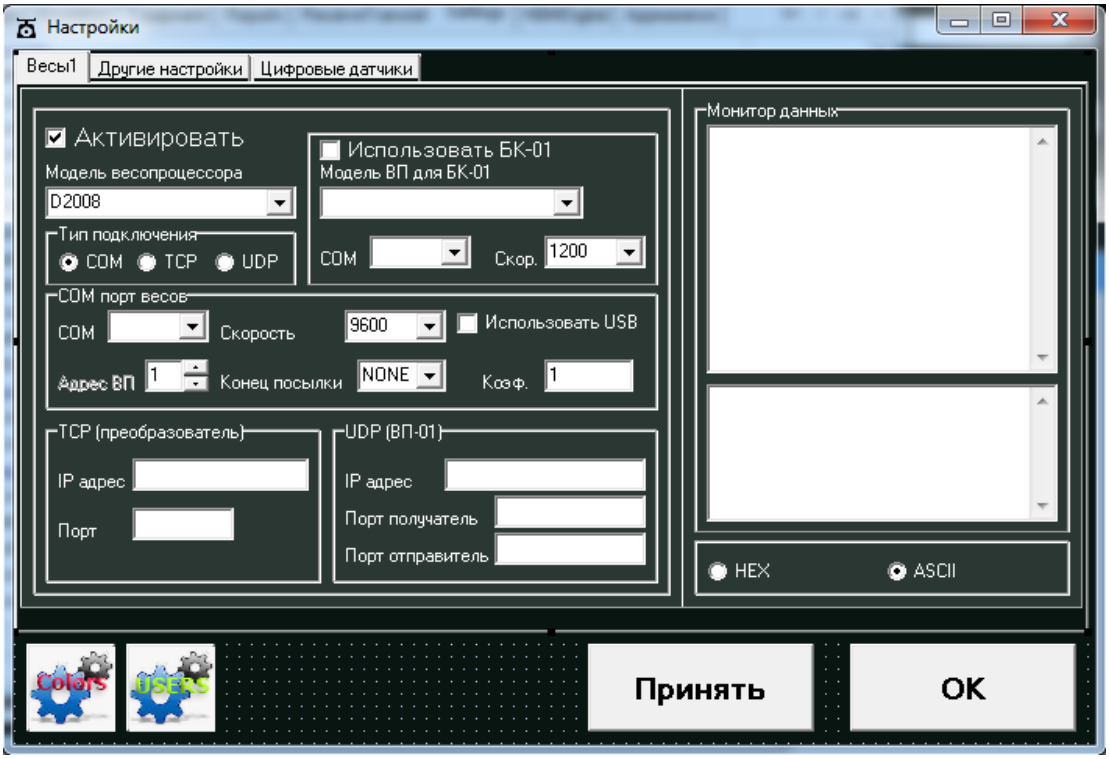

Рисунок 3.13 – Форма модуля Settings (налаштування зєднання з вагами та табло)

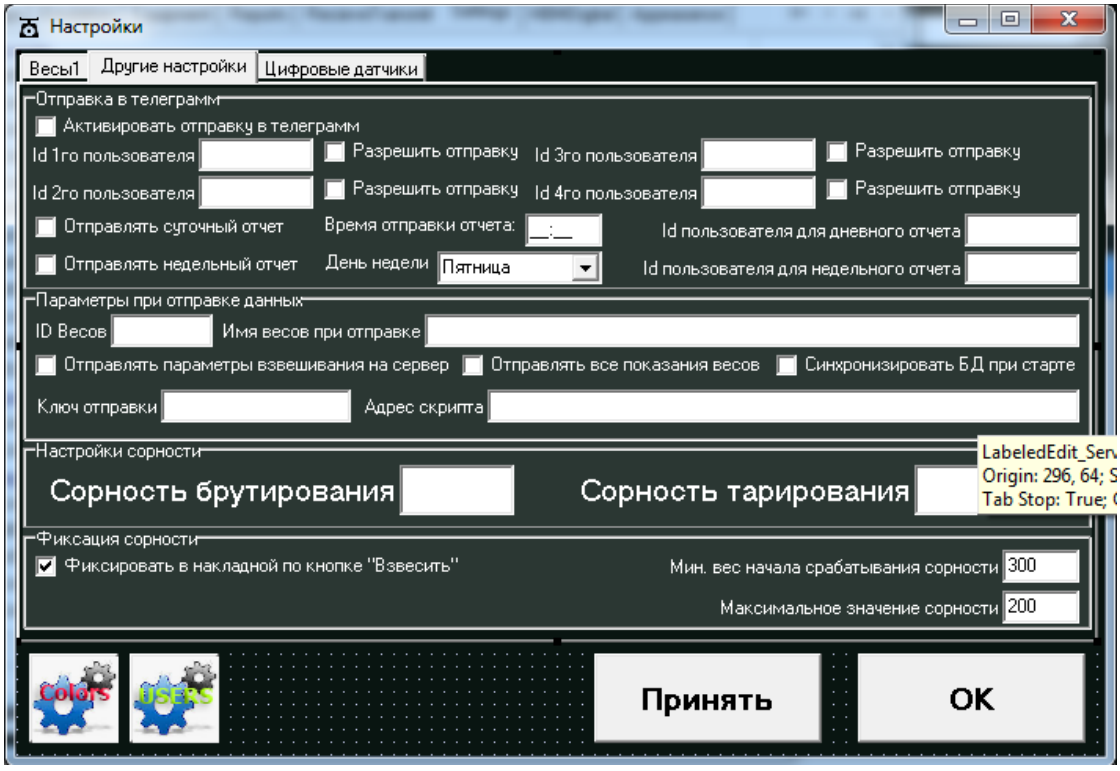

Рисунок 3.14 – Форма модуля Settings (налаштування відправлення даних)

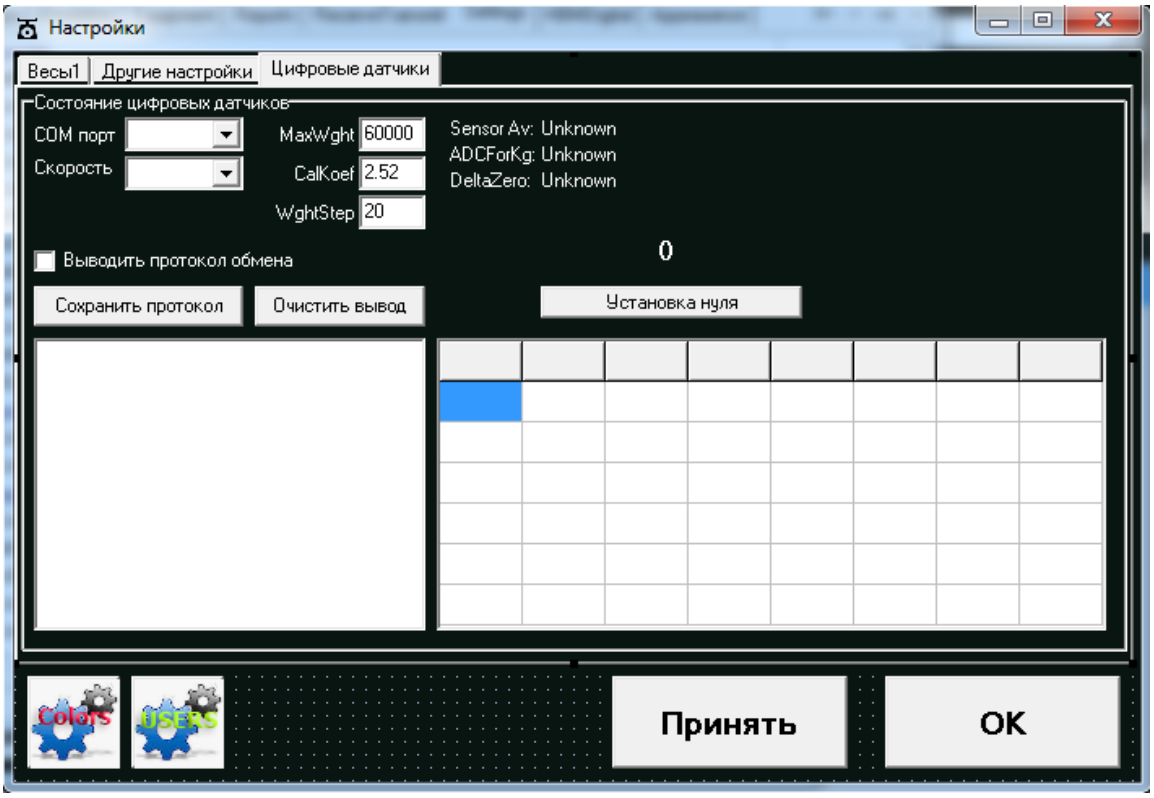

Рисунок 3.15 – Форма модуля Settings (налаштування цифрових датчиків)

Оскільки при великій кількості даних та/або не дуже швидкому інтернет зєднанні синхронізація з сервером може займати декілька секунд або навіть десятків секунд – щоб користувач не вважав, що програма зависла біло розроблено модуль Sync з власною формою (рис.3.16), що відображає процес та стан синхронізації.

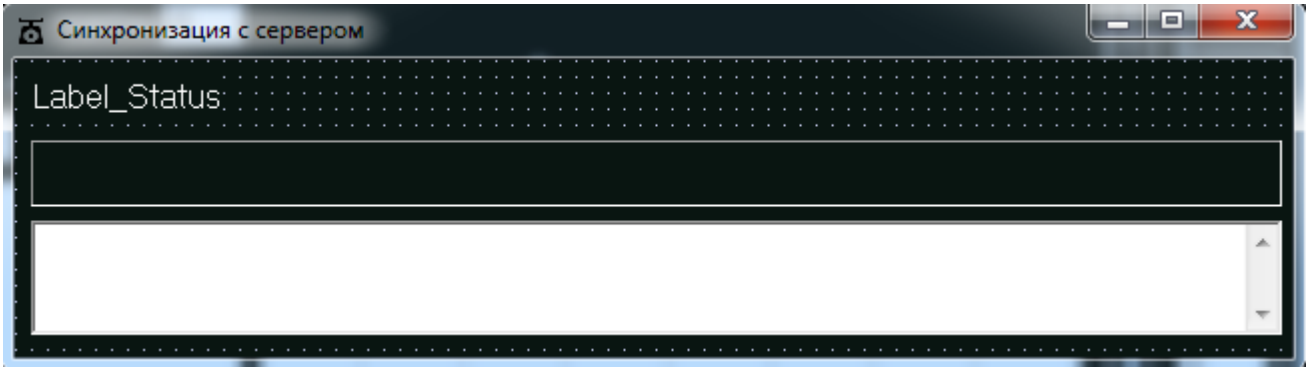

Рисунок 3.16 – Форма модуля Sync

Інші створені модулі мають наступні призначення:

- dBWork – робота з таблицями БД;
- HBMDigital – апартна робота з цифровими тензометричними датчиками;
- IPCams – робота за камерами;

- ParametersSetings – робота з файлами налаштувань;

- RFID – робота з безконтактними картами ідентифікації автомобілів;

- Scales – робота з ваговими терміналами;

- SendToServer – відправлення даних на сервер та телеграм;

- ServerSync – отримання налаштувань з сервера та синхронізація даних;

- Settings – завантаженя налаштувань при запуску програми;

- Size – зміна розмірів елементів форм в залежності від розміру екрану користувача;

- Tablo – робота з виносними індикаторами;

- TTNCreate – створення товарно транспортних накладних;

- UsersSetup – налаштування користувачів;

- VKSocket – робота з модулем керування навантаженнями.

4 МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ ПО РОБОТІ З СИСТЕМОЮ

4.1 Алгоритм роботи з системою

Оскільки комплекс має досить велику кількість функцій, а працівники елеваторів не завжди мають достатній досвід роботи з комп'ютером було розроблено основну інструкцію дій при роботі на ваговій яка наведена нижче.

Для запуску програми натисніть на ярлик з назвою Wght на робочому столі який виглядає як на рис.4.1.

Рисунок 4.1 – Зовнішній вигляд ярлика запуску ваговій програми

Після запуску програми ви побачите вікно як показано на рис.4.2

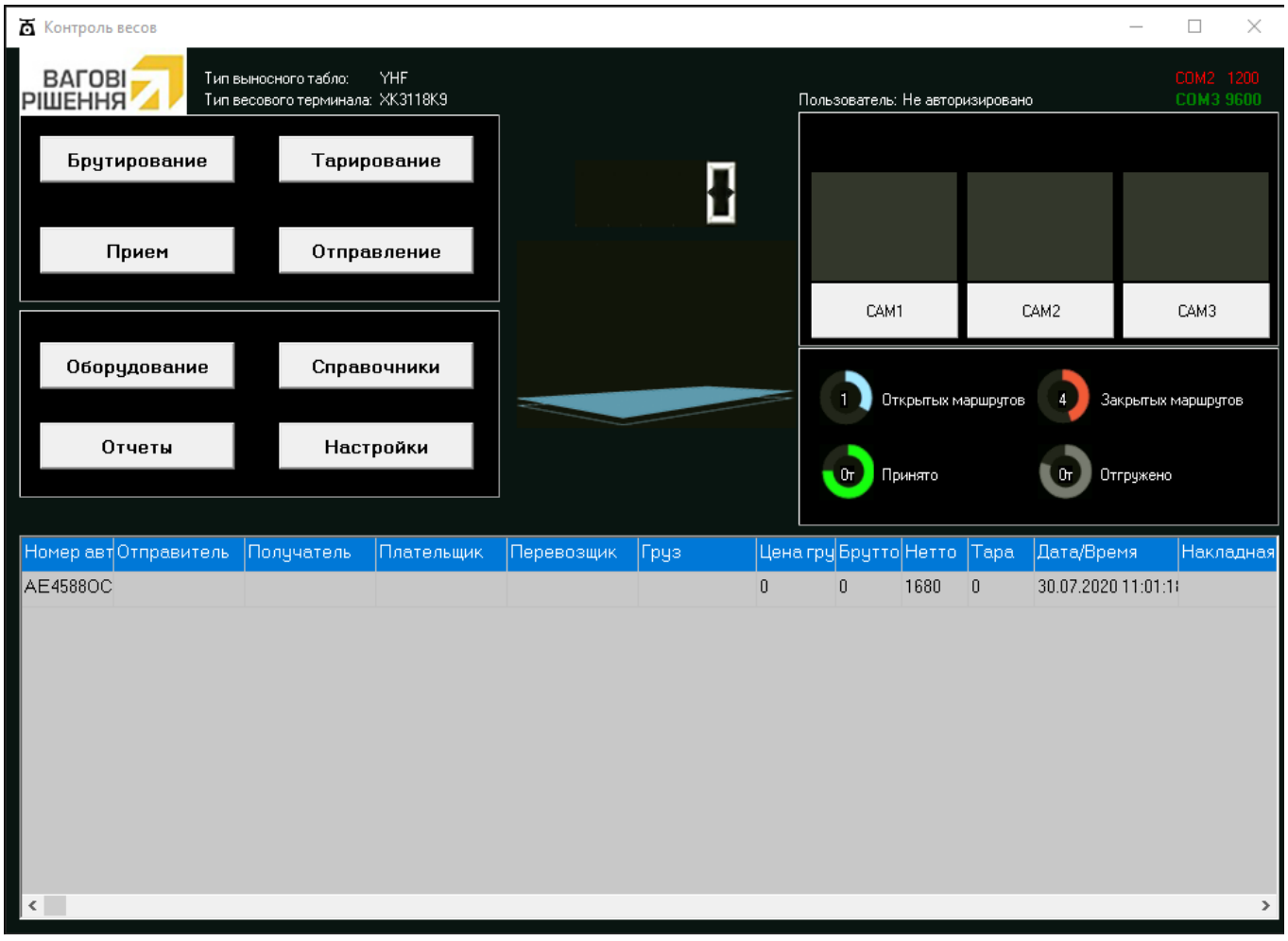

Рисунок 4.2 – Вікно при запуску

Зверніть увагу на колір підключень в правому верхньому куті (рис. 4.3)

Верхній рядок говорить про параметри з'єднання з виносним табло (в разі якщо табло підключено до комп'ютера, а не безпосередньо до вагового терміналу)

Нижній рядок показує параметри з'єднання з ваговим терміналом. Колір рядків говорить про успішність з'єднання. Якщо колір зелений - значить відповідне з'єднання встановлено. Написи говорять про номер і швидкості роботи COM порту.

У разі якщо якесь із з'єднань червоного кольору (крім випадку коли виносне табло підключається безпосередньо до вагового терміналу) перевірте настройки, переконайтеся чи включений ваговий термінал і якщо це не допомогло або самостійно перевірте кабельне з'єднання або зверніться до технічного фахівця.

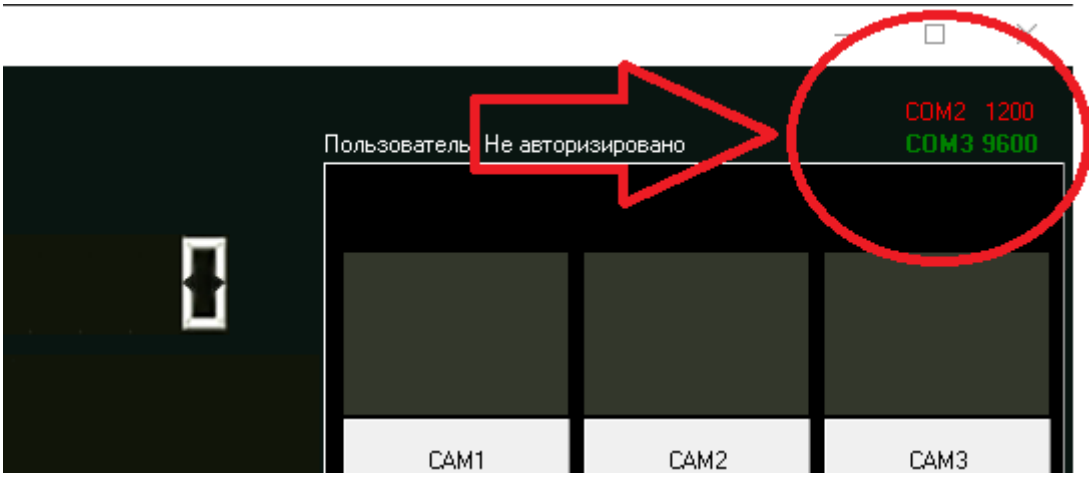

Рисунок 4.3 – Покажчики стану з'єднань з виносним табло і ваговим терміналом

У лівій верхній частині програми розташовується інформація про тип зазначеного в налаштуваннях виносного табло і вагового терміналу, а також наявності інтернет з'єднання для видаленої відправки даних про зважування (рис.4).

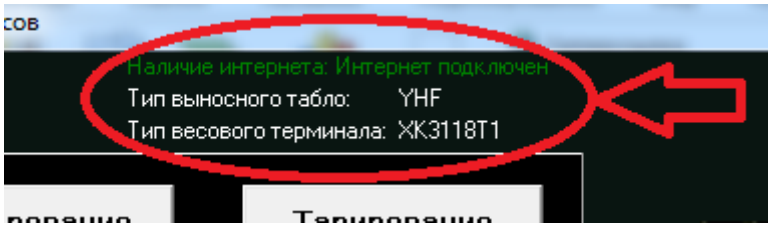

Рисунок 4.4 – Покажчики типу виносного табло і вагового терміналу, а також наявності інтернету

По центру програми розташовуються поточні показання вагового терміналу. Вікно в залежності від з'єднання з ваговим терміналом і наявності машини на вагах може приймати вид як показано на рис 4.5-4.7.

На рис. 4.5 область ваги програми показує користувачеві, що ваговий термінал не передати показання ваги. Це може бути з наступних причин:

- вимкнений ваговий термінал (для усунення включіть ваговий термінал);

- неправильно обраний COM порт або швидкість з'єднання з ваговим терміналом (переконайтеся що в налаштуваннях обрано правильний COM порт);

- неправильно обраний тип вагового терміналу (переконайтеся що в настройках обов'язково встановіть тип вагового терміналу);

- пошкоджений або відключений кабель (перевірте з'єднання або зверніться до технічного фахівця).

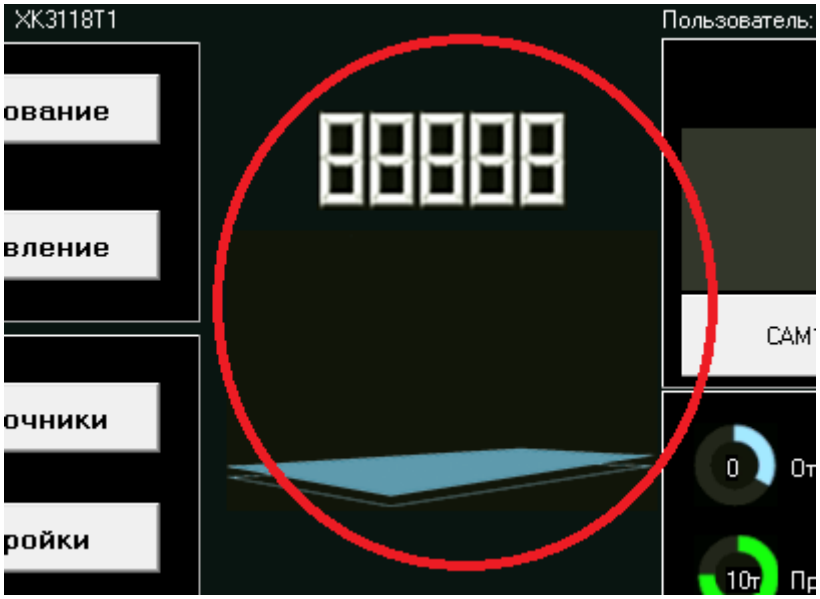

Рисунок 4.5 – Область показаний веса в случае отсутствия связи с весовым терминалом

На рис.4.6 в області показань ваги програми висвітлено 0, що говорить про розвантажену вагову платформу.

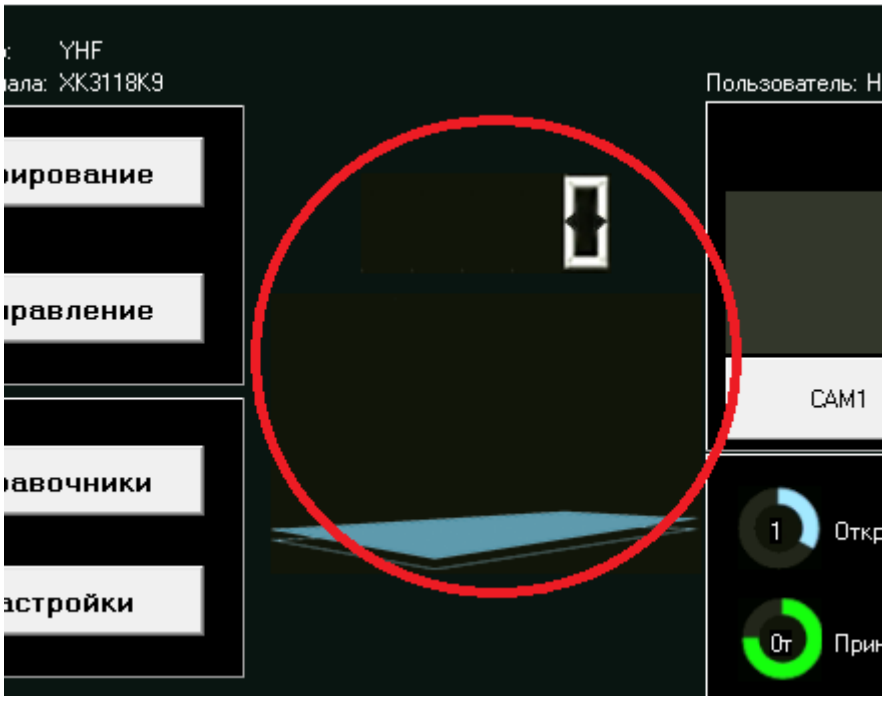

Рисунок 6 – Нульови покази ваги

На рис.4.7 в області показань ваги присутній ненульова маса (в прикладі 860). Зверніть увагу, що коли показання ваг ненульові під показаннями ваги з'являється зображення автомобіля на відміну від випадків відсутності з'єднання і нульових показань.

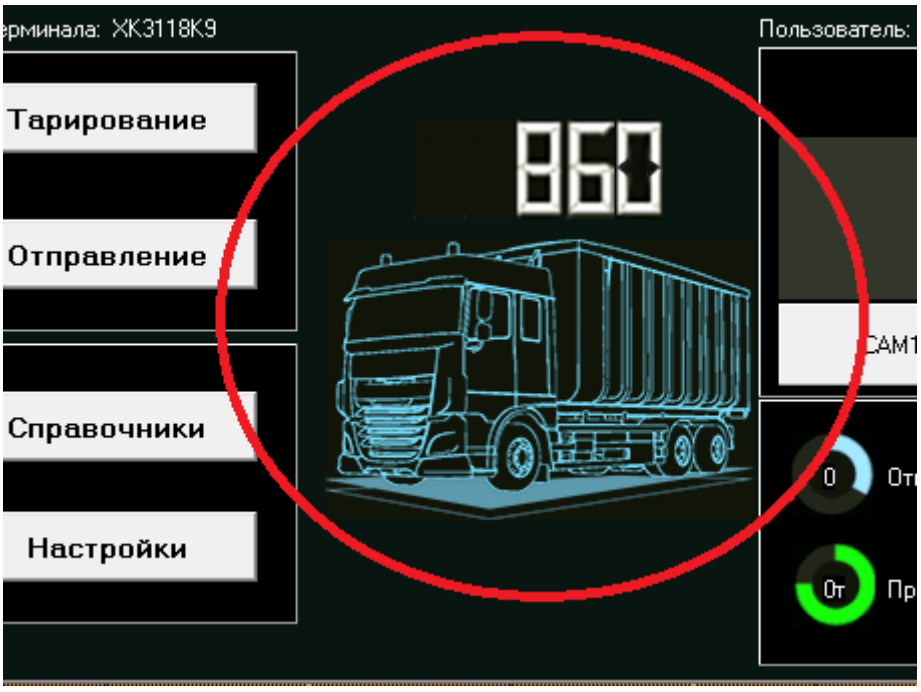

Рисунок 4.7 – Ненульові показання ваги

Порядок зважування машини.

Програма розрахована на зважування машини два рази і автоматичне обчислення маси нетто на підставі двох зважувань.

Якщо ви приймаєте товар (перший раз машина на ваги заїжджає повна, другий раз порожня - після вивантаження вантажу), якщо відправляєте товар (перший раз машина на ваги заїжджає порожній, а другий раз повної після завантаження вантажу).

Для проведення зважування при першому заїзді машини на ваги виберіть режим «Прийом» або «Відправлення» (см.рис.4.8).

Рисунок 4.8 – Вибір режиму зважування

При виборі режиму «Прийом» активна буде тільки кнопка «Брутірованіе» над кнопкою «Прийом» тому, що при прийомі товарів першої зважування це отримання ваги брутто.

При виборі режиму «Відправлення» активна буде тільки кнопка «Тарування» над кнопкою «Відправлення» тому, що при прийомі товарів першої зважування це отримання ваги Тара.

При виборі будь-якого з режимів «Прийом» або «Відправлення» знизу будуть показані проведені перші зважування в цьому режимі.

Для фіксації ваги натисніть кнопку «Брутірованіе» або «Тарування» в залежності від обраного режиму. Після цього з'явиться вікно як показано на рис.4.9

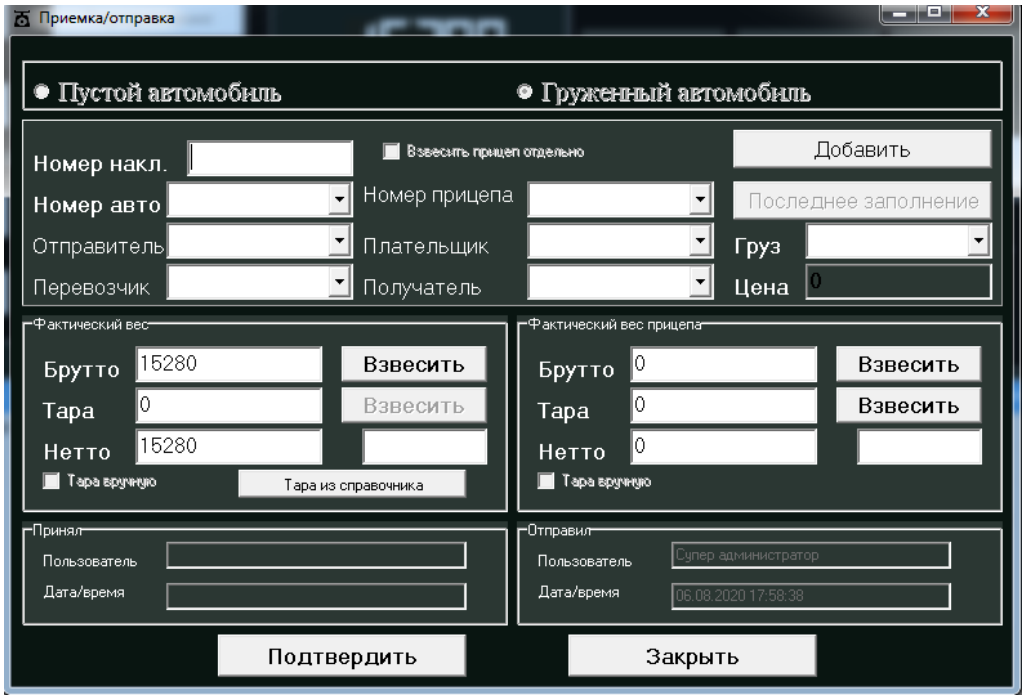

Рисунок 4.9 – Вікно підтвердження зважування

Зверніть увагу що при брутуванні маса на вагах буде відображатися в полі «Брутто», а при таруванні в поле «Тара» і відповідно поруч з цими полями буде активна кнопка «Зважити». Для фіксації ваги натисніть кнопку «Зважити» при цьому вага зафіксується в поле, але на терміналі може змінюватися якщо наприклад водій сяде в машину або платформу буде розгойдуватися (рис.4.10).

Для внесення параметрів зважування заповніть поля у верхній частині вікна (заповнити можна як простим набором тексту так і вибором зі списків якщо такий параметр уже вводився). Крім того якщо параметр вже вводився програма за першими введеним буквах буде підказувати продовження тексту (рис.4.11). Якщо потрібно ввести установки уникайте введення лапок в іменах і прогалин в номерах машин для виключення помилок при збереженні в базі даних.

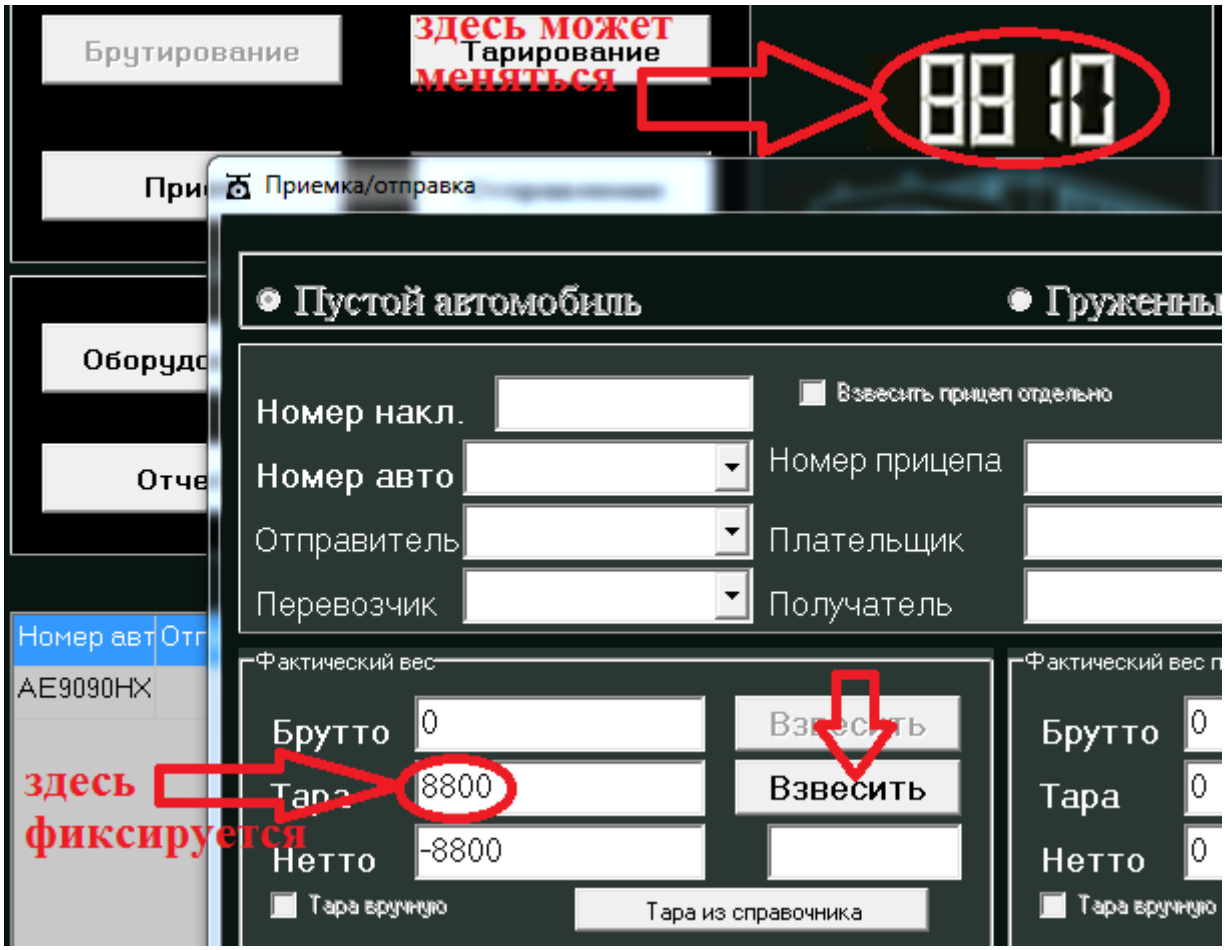

Рисунок 4.10 – Фіксація ваги в поле після натискання кнопки «Зважити»

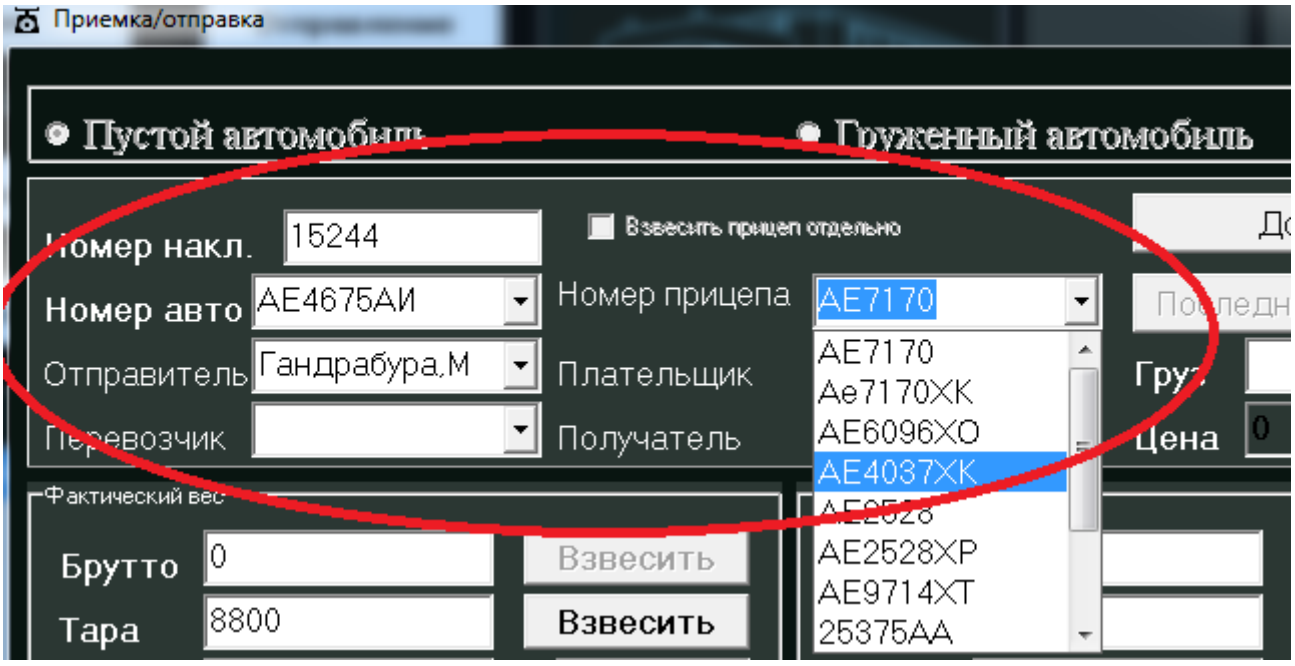

Рисунок 4.11 – Введення установок зважування

Для запису введених даних з масою машини в базу даних натисніть кнопку «Підтвердити» в лівому нижньому кутку вікна - при цьому в таблиці в основному вікні з'явиться новий рядок з введеними вами параметрами, у відкритому вікні очистяться параметри (крім ваги), а кнопка «Підтвердити» стане неактивною (рис.4.12).

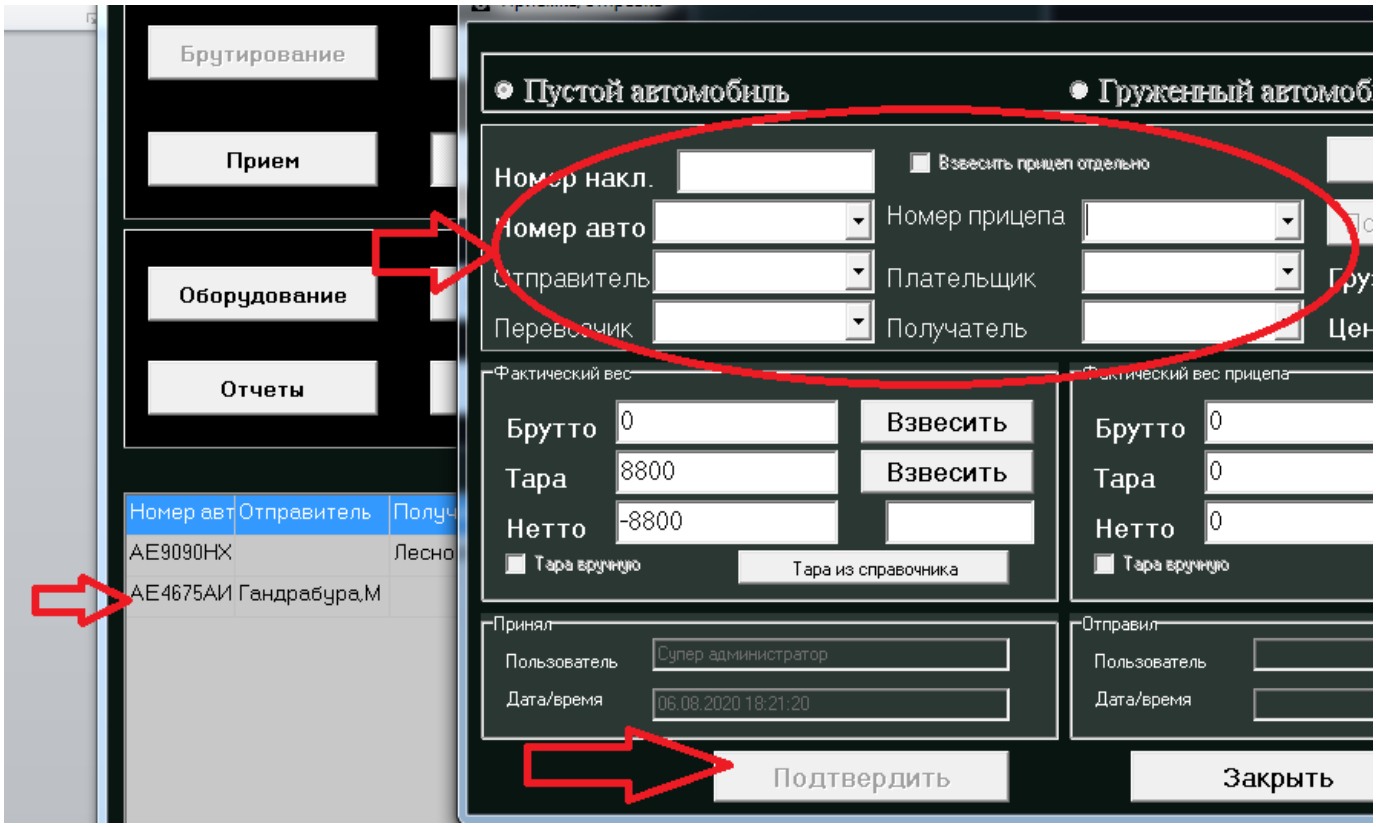

Рисунок 4.12 – Запис зважування в базу даних

Після того як машина з'їдять з терезів - натисніть кнопку «Закрити» в правому нижньому кутку вікна.

При другому заїзді машини на ваги виберіть в таблиці в нижній частині екрана рядок з параметрами машини і зробіть подвійне клацання по ній лівою кнопкою миші. Після цього відкриється вікно з введенням при першому зважуванні параметрами і в відповідному полі перший вагу і поточний вага машини (рис.4.13). У прикладі на ваги для другого зважування заїхала машини з номером «AE4675AI» з масою 15230, при цьому при першому зважуванні у цієї машини маса була 8800 - програма автоматично заповнила поля з даними і порахувала масу «Нетто» вантажу.

В даному випадку це була відправка товару (при першому зважуванні машина була порожня і перша операція була тарування, а при другому зважуванні машина повна і операція - брутірованіе). Зверніть увагу, що якщо ви при першому зважуванні забули ввести якісь дані - ви можете їх заповнити при другому зважуванні.

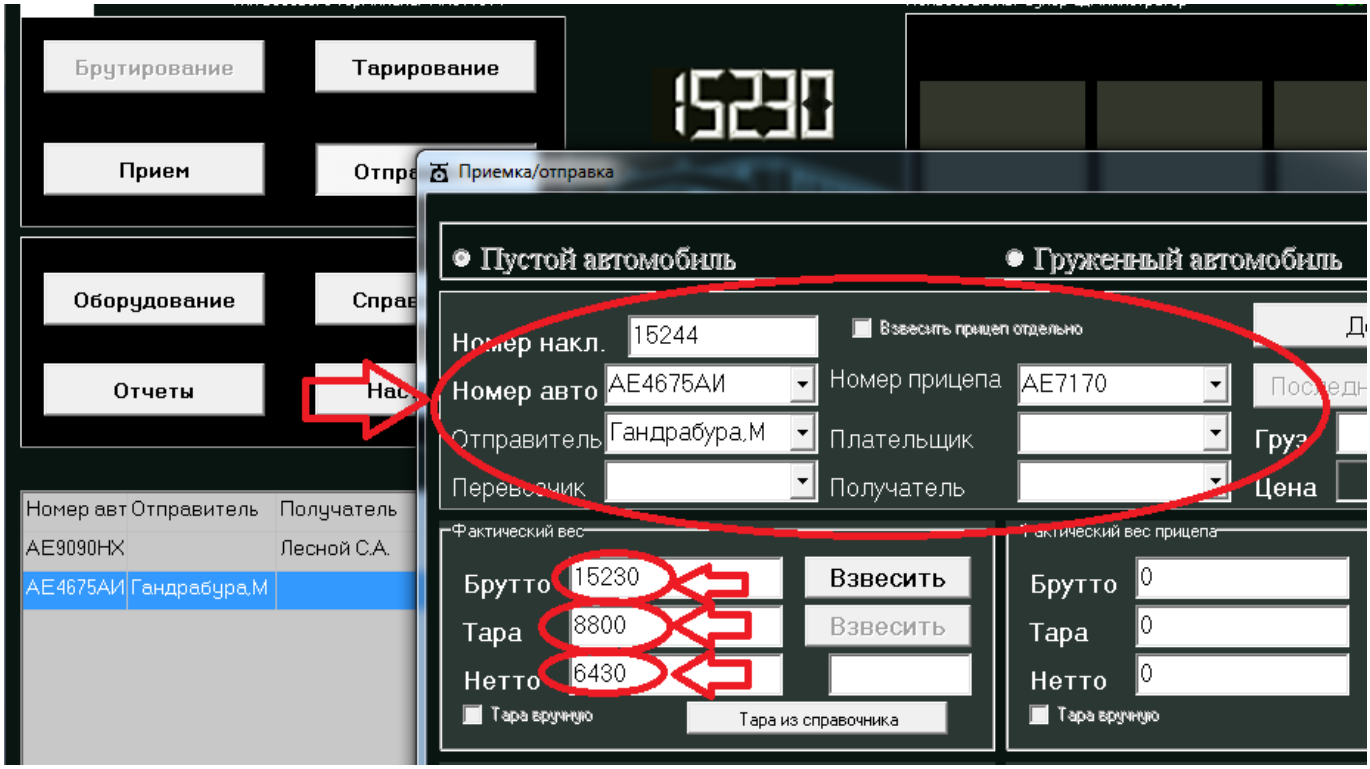

Рисунок 4.13 – Приклад другого зважування машини

Також як і при першому зважуванні для фіксації ваги у вікні натисніть активну кнопку «Зважити», а для запису всіх даних в базу даних натисніть кнопку «Підтвердити», а після того як машина з'їдять з ваг натисніть кнопку «Закрити».

Після другого зважування дані про машину зникнуть з основного вікна програми і будуть доступні в розділі «Звіти»

Якщо коротко порядок дій зі зважування машини виглядає наступним чином:

При першому заїзді машини:

1. Вибрати режим «Прийом» або «Відправлення»

2. Натиснути кнопку «Брутірованіе» або кнопку «Тарування» (в завсімості від обраного режиму)

3. Заповнити поля з даними у верхній частині вікна, що з'явилося

4. Натиснути кнопку «Зважити» для фіксації ваги

5. Натиснути кнопку «Підтвердити» для запису даних в базу даних

6. Після того як машина з'їдять натиснути кнопку «Закрити»

При другому заїзді машини:

1. Зробити подвійне клацання лівою кнопкою миші по рядку з машиною в таблиці основного вікна програми

2. Зафіксувати вага кнопкою «Зважити»

3. Зберегти параметри в базі даних кнопкою «Підтвердити»

4. Після того як машина з'їдять натиснути кнопку «Закрити»

Для формування звітів по різним параметрам за різний період часу та для можливості внесення змін та доповнень використовується режим «Звіти». Для переходу у відповідну форму натисніть кнопку «Звіти» у головному вікні програми.

Повійним натисканням на обране зважування буде викликана картка вантажу де можна буде провести редагування або додавання даних – ця функція доступна лише адміністраторам системи (рис.4.14).

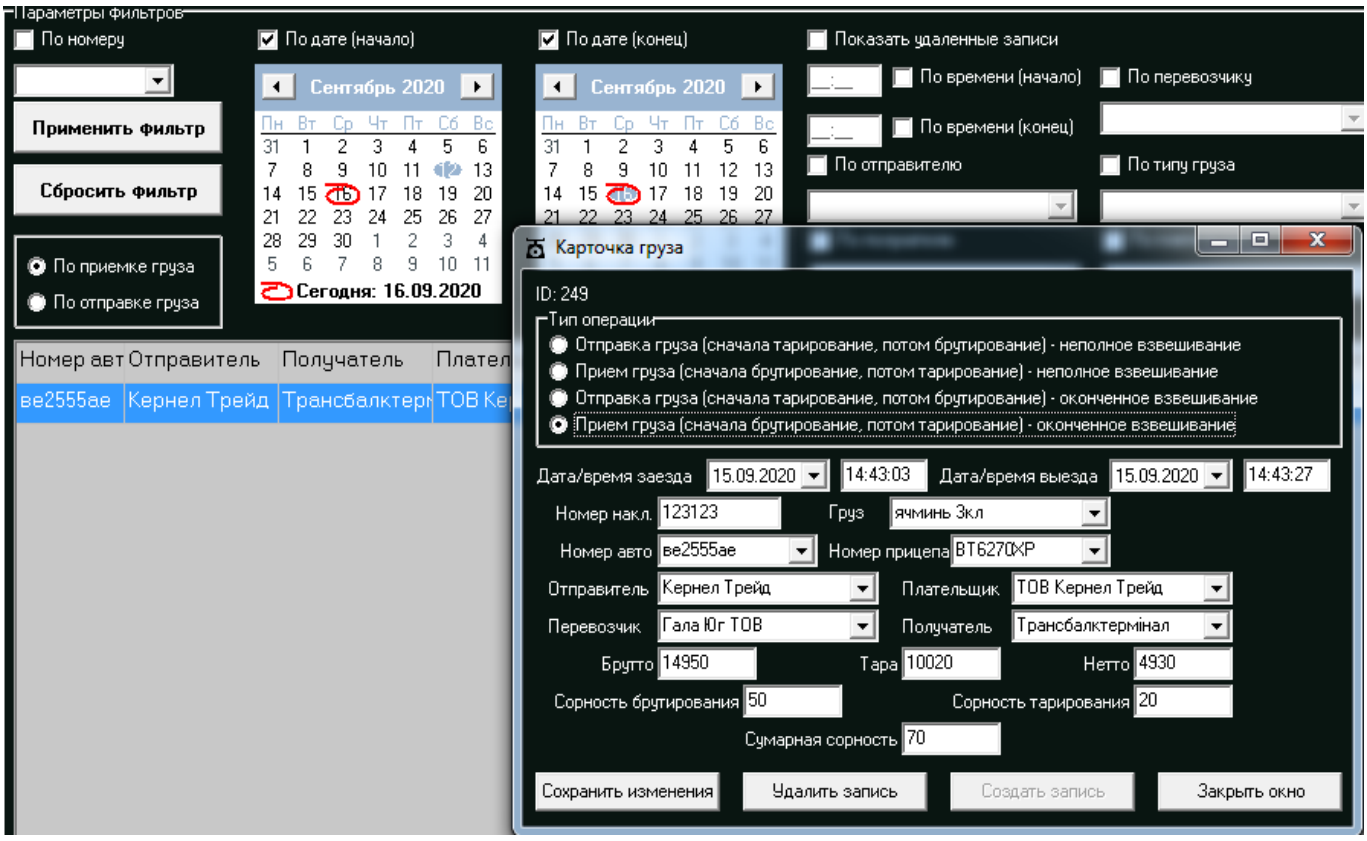

Рисунок 4.14 –Внесення змін у записи

Якщо є необхідність додати вручну запис (наприклад у випадках коли з технічних причин зважування проводилось без використання комплексу) в свобідному полі треба натиснути праву кнопку миші та обрати «Додати запис» (рис.4.15). Ці дії також доступні лише для адміністраторів системи з метою запобігання зловживань.

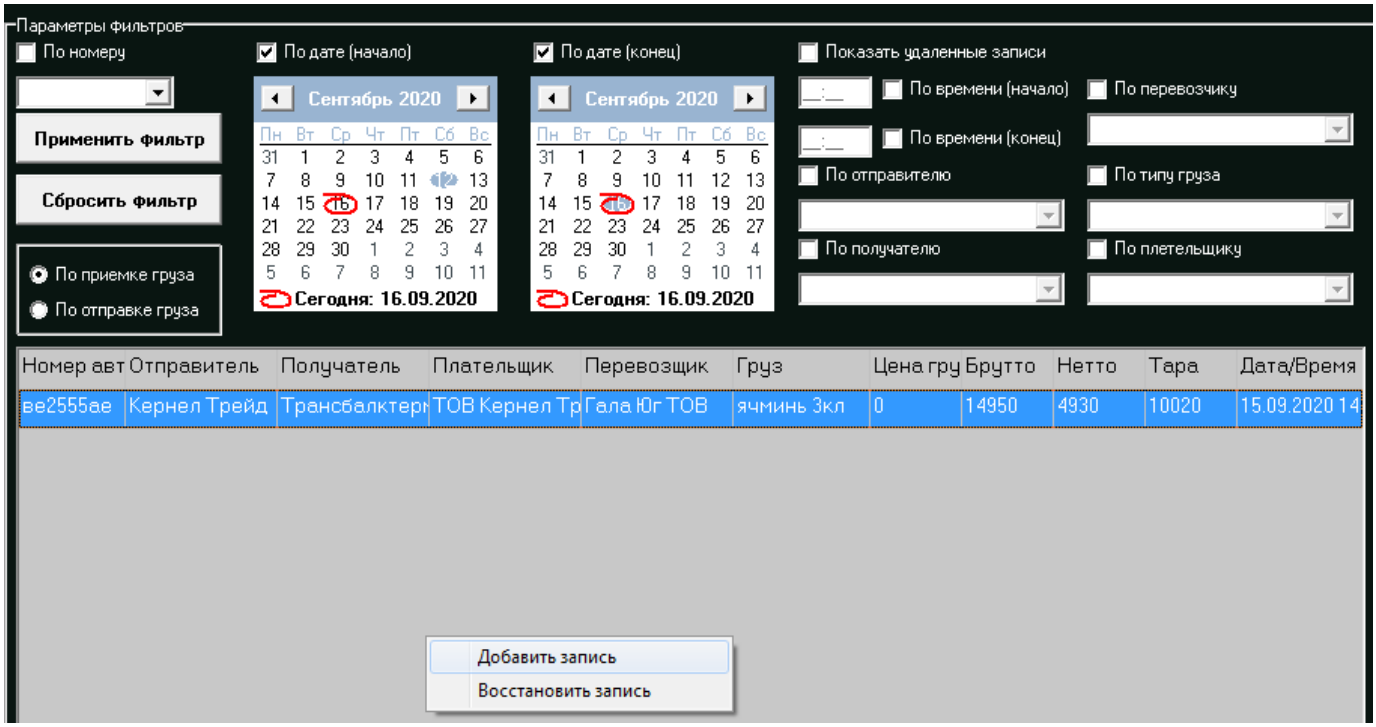

Рисунок 4.15 – Додавання нового запису

4.2 Керівництво технічного співробітника

Після встановлення комплексу на новий об’єкт необхідно встановити програму обліку, створити запис об’єкту на сервері, налаштувати відправлення даних на сервер та телеграм та провести налаштування приєднаного обладнання. Всі ці дії проводить віддалено або знаходячись на об’єкті представник розробника.

Створення ключового файлу.

Генератор ключових файлів виконано окремими програмами, та складається з програми отримання серійного номеру, що поставляється разом з основним пакетом програми для вагового контролю (рис.4.16) та програми генерації ключового файлу на основі отриманого серійного номеру, що зберігається лише у розробника (рис.4.17).

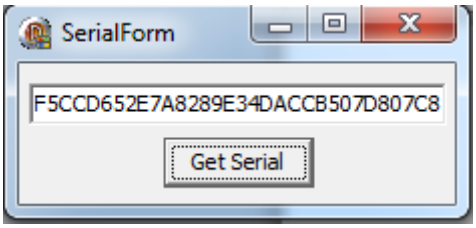

Рисунок 4.16 – Програма отримання серійного номеру

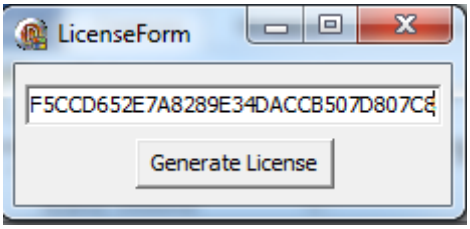

Рисунок 4.17 – Програма генерації ключового файлу

Створення об’єкту прив’язаного до серверу.

Для створення на сервері загального запису про об’єкт з власним ідентифікаційним номером та ключом авторизації необхідно зайти в хостинг де розташовано серверні скрипти та провести внесення запису з використання phpMyAdmin (вбудованої утіліти для роботи з БД) за допомогою SQL запиту. На рис.4.18 наведено приклад додавання об’єкту з назвою «Козлова_Даша», ідентифікаційним номером «T1005», ключом авторизації «D9335F85F470E11F» для вагового терміналу «XK3118K9». Ідентифікаційний номер та ключ авторизації використовується для запобігання несанкціонованого запису даних на сервер та отримання даних з серверу.

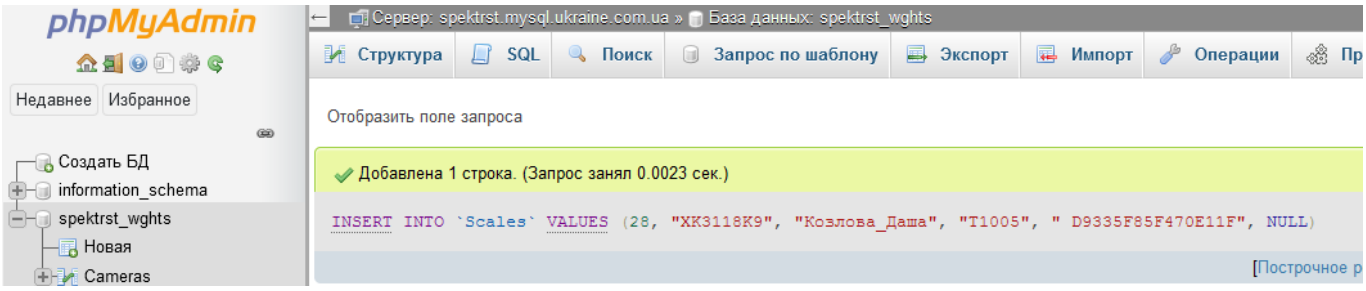

Рисунок 4.18 – Додавання об’єкту в панелі хостингу

Налаштування відправлення на сервер на боці програми вагового обліку

Після додавання об’єкту на сервері адміністратору системи або представнику розробника необхідно внести параметри об’єкта у програмі вагового контролю. Для цього натиснути кнопку «Налаштування» в головній формі програми, обрати закладинку «Інші налаштування» та в області з назвою «ID ваг» ввести ідентифікаційний номер, в поле «Імя при відправці» ввести назву об’єкту, в поле «Ключ відправлення» ввести ключ авторизації - ці параметри до цього було введено при створені запису на сервері (вони мають співпадати). Крім того в полі «Адреса скрипта» треба вказати доменну адресу сайту (віддаленого сервера) де знаходяться скрипти серверу (рис.4.19).

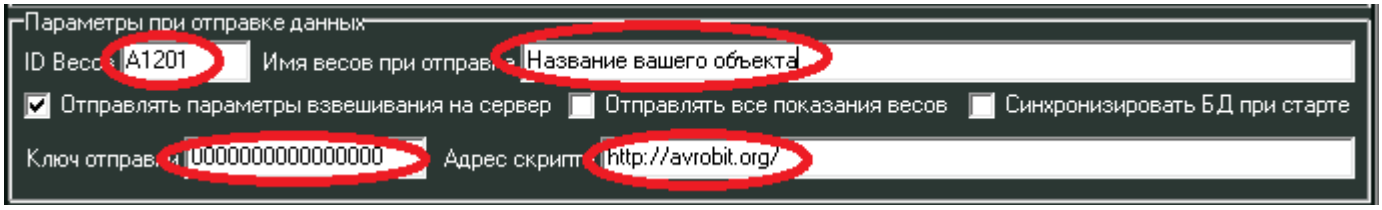

Рисунок 4.19 – Внесення параметрів відправки на сервер

Після введення параметрів відправлення треба дозволити відправлення виділивши галочку «Відправляти параметри зважування на сервер» (рис.4.20).

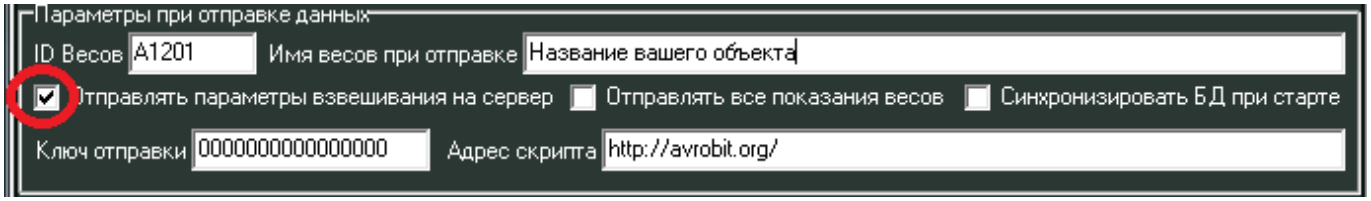

Рисунок 4.20 – Дозвіл на відправлення зважувань з параметрами

Якщо необхідно відправляти усі показання ваг в незалежності від того чи були введені оператором параметри (найчастіше це при автоматичному зважуванні) треба відмітити галочку «Відправляти всі показання ваг» (рис.4.21).

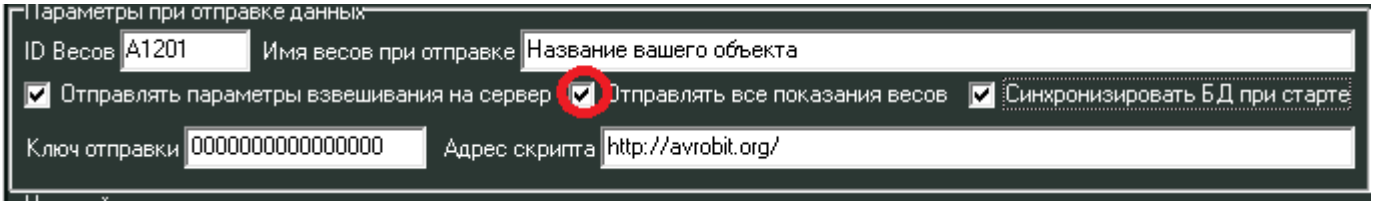

Рисунок 4.21 – Дозвіл на відправлення усіх зважувань

Якщо є необхідність синхронізувати всі проведені зважування при старті програми (найчастіше це потрібно при слабкому інтернеті коли не всі зважування можуть відправитися в момент вимірювання) треба обрати галочку «Синхронізувати БД при старті» (рис.4.22). При цьому при кожному запуску програми буде відкриватися вікно синхронізації та вся локальна БД та файли налаштувань будуть синхронізуватися з даними на сервері.

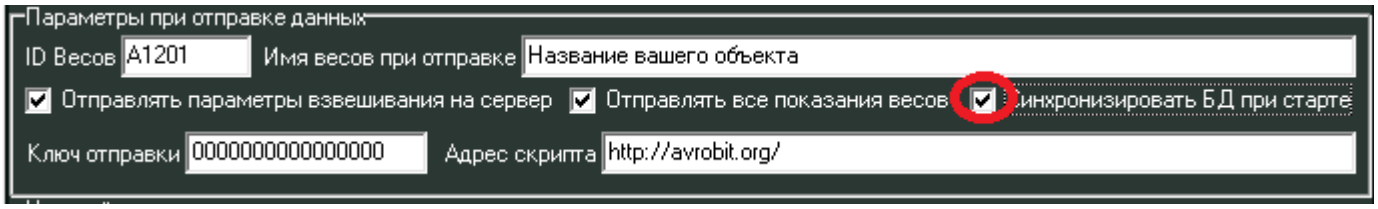

Рисунок 4.21 – Дозвіл синхронізації при старті

Налаштування роботи з сервером на стороні термінальної програми перегляду та редагування даних у лабораторії та у власника, директора чи уповноваженої на це особи.

Дл роботи лабораторії та контролю з боку власника, директора чи уповноваженої особи треба додатково додати створений на сервері об’єкт у термінальну програму перегляду даних. Для цього при запуску програми треба пройти авторизацію як «Адміністратор» (рис.4.22).

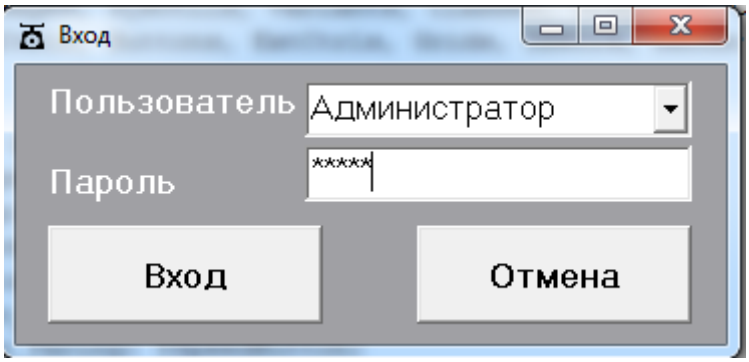

Рисунок 4.23 – Авторизація адміністратора

Після вдалої авторизації в списку посередені головної форми програми буде доступний перелік вже підключених об’єктів (рис.4.24).

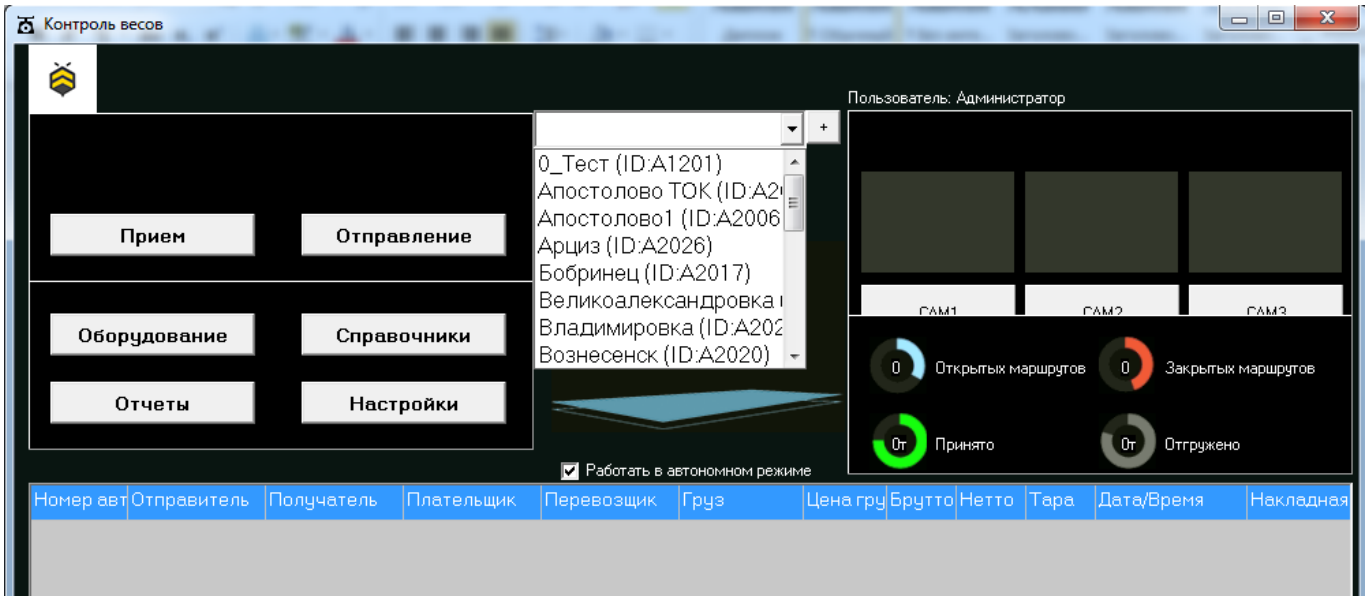

Рисунок 4.24 – Перелік підключених об’єктів

Для додавання нового об’єкту треба натиснути кнопку «+» справа від переліку. Після цього зявиться форма як показано на рис.2.15. У цю форму траба внести дані з назвою об’єкту, ідентифікаційним номером на сервері та ключом шифрування, що також вносився на сервер та натиснути кнопку «Додати»

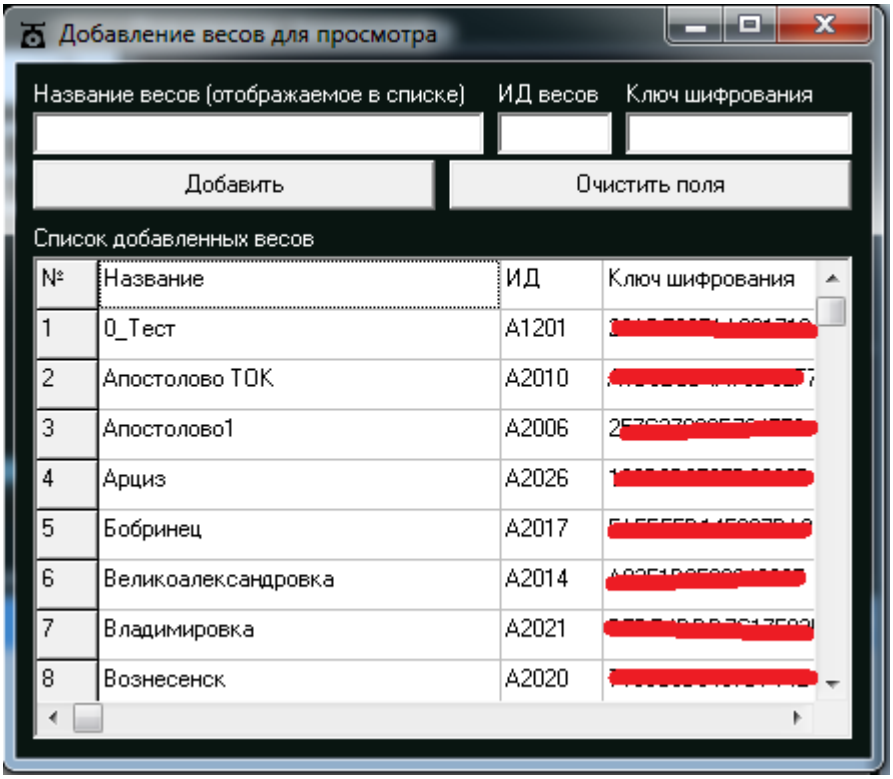

Рисунок 4.25 – Форма додавання нового об’єкту у перелік

Налаштування відправлення у Telegram

Програма має можливість відправки даних в вигляді повідомлень в телеграм. Перед тим як прописувати налаштування для відправки дізнайтеся свій ІД в телеграмі і додайте в контакти бот, який буде робити розсилку вироблених зважувань.

Для того щоб дізнатися свій ІД в телеграмі

Додати @userinfobot, попросити того кому треба висилати написати будь-яке повідомлення боту @userinfobot - він у відповідь дасть ВД який прописується в настройках. Другого бота можна додати тільки собі і попросити адресата звітів вислати вам повідомлення, а потім це повідомлення переслати боту @userinfobot (див. рис.4.26 – 4.27).

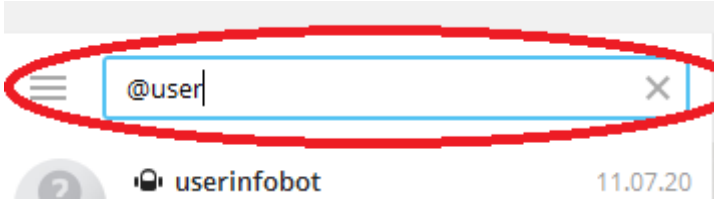

Рисунок 4.26 – Пошук бота @userinfobot

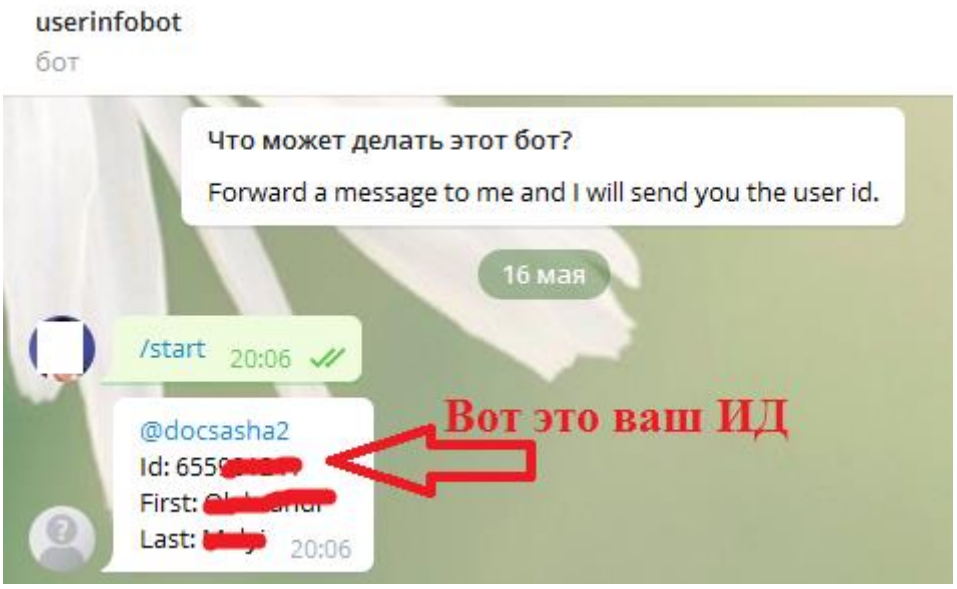

Рисунок 4.27 –Отримання ІД користувача телеграма

Для отримання повідомлень потрібно додати (тому хто буде отримувати дані) в телеграмі в контакти бота який буде ці повідомлення висилати @Avrobit_bot

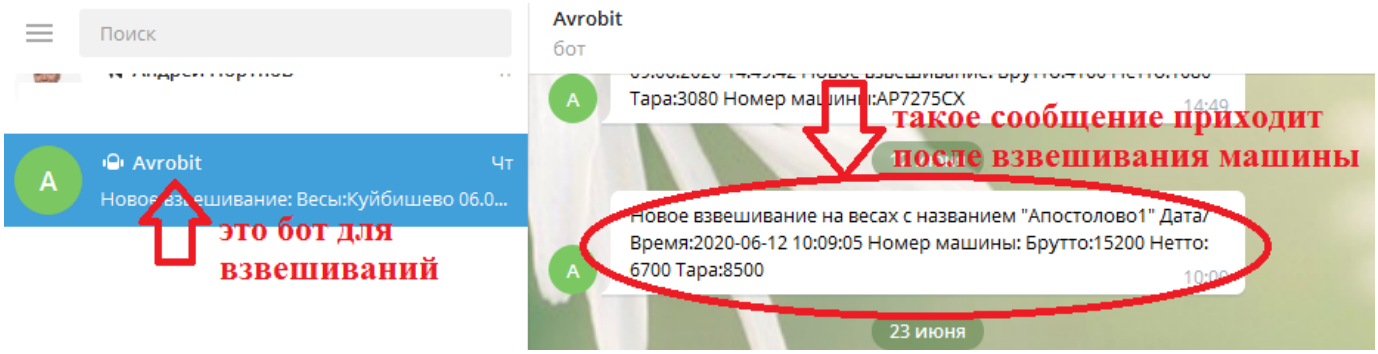

Рисунок 4.28 – Приклад отримання даних про зважування

Після того як ви дізналися ВД всіх одержувачів в налаштуваннях програми треба увімкнути надсилання в телеграм. Для цього натисніть кнопку «Налаштування» в головному вікні програми, у вікні виберіть вкладку «Інші настройки».

Відзначте галочкою пункт «Активувати відправку в телеграм», в полях з назвою «Id користувача» введіть ІД одержувача і справа також поставте позначку галочкою «Дозволити відправку» (рис.4.29).

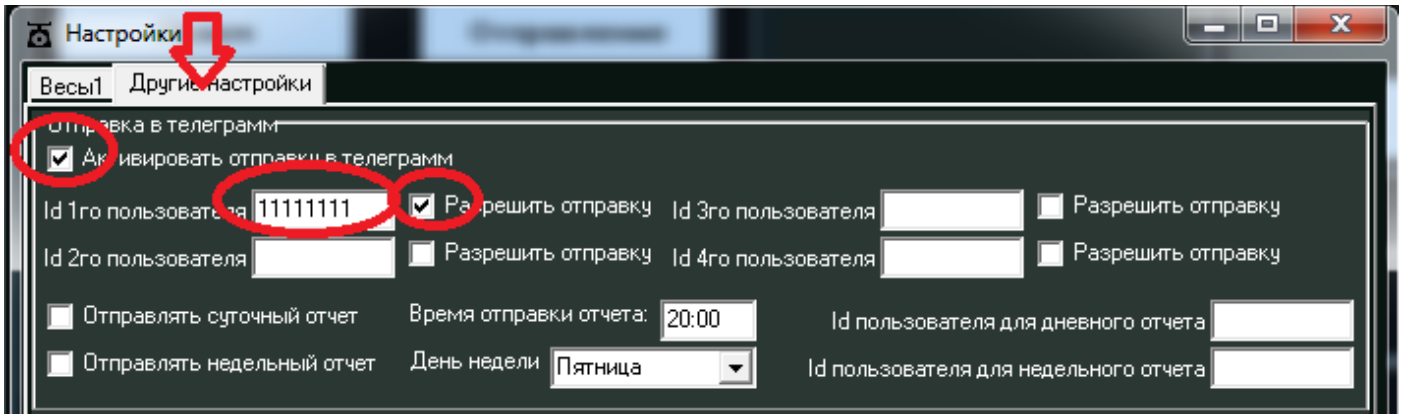

Рисунок 4.29 – Внесення налаштувань відправки в телеграм в програму

Розсилку може отримувати до 4х користувачів

5 ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

Розроблена магістерська робота «Розробка програмно-апаратного комплексу обліку та контролю роботи зернових елеваторів за широким рядом показників» призначена для контролю кількості та якості зерна яке отримують для зберігання зернові елеватори.

5.1 Аналіз економічної привабливості проекту

В магістерській роботи було проведено розробку програмно-апаратного комплексу, що дозволить максимально автоматизувати облік кількості та якості зернових культур, що надходять на елеватори та відпускаються покупцям.

Основні параметри реалізованої ідеї наведено у таблиці 5.1

Розроблений програмно-апаратний комплекс може бути широко застосований на підприємствах, що мають елеватори для зберігання зернових культур (жито, кукурудза, насіння соняшника, рапс, соя та інших).

На таких підприємствах як правило при прийомі продукції від фермерів присутня лабораторія, що територіально не завжди знаходиться поруч з автомобільними вагами. Це призводить до того, що після проведення аналізу показників якості зернових культур перед зважування (для визначення кількості зернової культури на вагах) необхідно створити та роздрукувати велику кількість документів які потім надаються експедитору машини для проведення зважування та визначення місця подальшого зберігання. Цей процес потребує додаткового часу роботи працівників та витрат на канцелярські матеріали. Використання розробленого комплексу надасть можливість відійти від необхідності створення додаткових супровідних документів оскільки дані відразу будуть відправлятися в загальну мережу та при зважуванні автоматично підтягуватись в базу даних з вказанням цільового місця зберігання.

Таблиця 5.1 - Опис ідеї

Зміст ідеї	Напрямки застосування	Вигоди для споживачів (користувачів)
Програмно апаратний комплекс надає можливість автоматизувати введення параметрів якості зернових культур (отримані в лабораторії при елеваторі), проводити автоматизований облік отриманої та відпущеної продукції, автоматизує пропускний режим автомобілів через ваги. Комплекс складається з програми для ПК, віддаленого серверу та обладнання для зважування та реалізації про пускового режиму.	1. Внесення даних про якість зернових культур зі зберіганням на віддаленому сервері	1.Можливість внесення параметрів якості зернових культур, що поступають на зберігання через комп'ютерну мережу. Відсутність необхідності застосування зайвих паперових носіїв.
	2. Автоматизоване зважування автомобілів з зерновими культурами.	2. Можливість виключення людського фактору з процесу зважування. Автоматичне встановлення находження машини на вагах та визначення ваги товару в ній зі збереженням на захищеному сервері.
	3. Формування звітів про рух продукції	3. Прискорення створення звітної документації
	4. Облік автомобілів, що проходять зважування	4. Реалізація автоматизованого про пускового режиму автомобілів

Використання в розробленому комплексі можливості розпізнавання номеру, автоматичного шлагбауму та світлофору, що керуються програмо. Дозволить автоматично після проходження аналізу якості запустити машину на зважування. Використання датчиків положення на платформі та алгоритму визначення машини на вагах надасть можливість проведення автоматичного зважування, що дає виключити людський фактор при обліку продукції і навіть обійтися без робітника вагової.

При створенні звітів на підприємстві необхідно враховувати отриману та відпущену продукцію за певний період та її якість, орієнтуватися з місцями зберігання продукції. Використання розробленого комплексу дозволить роботи звіти автоматично по вказаним параметрам (період часу, тип продукції, показники її якості, місце зберігання, а при додаванні цін на продукцію надасть можливості отримати суму отриманої та відпущеної продукції).

Оскільки комплекс включає в себе використання камер з автоматичним розпізнаванням номерних знаків автомобілів – це надає можливості керування рухом та проводити аналізу руху транспортних засобів через ваги, що спрощує пропускний режим підприємства.

Загалом користувач системи отримує вигоду у вигляді прискорення процесу отримання та відпущення продукції та можливість скоротити фонд робочого часу за рахунок автоматизації великої кількості процесів обліку та контролю.

Для реалізації розроблено комплексу необхідно два розробника: програміст для написання програмного коду програми керування та збереження даних та інженер радіоелектронщик для проектування електронних компонентів системи (модулів керування шлагбаумом, світлофором, датчиками положення, тензометричними датчиками та електронними вагами).

Для реалізації процесу розробки комплексу необхідно проаналізувати основні параметри якості зернових культур, процес зважування автомобілів з отриманими зерновими та зерновими, що підлягають відпуску продукції.

Реалізовувати розроблений комплекс можна усім підприємствам, що займаються зберіганням та реалізацією зернових культур. Насамперед це зернові

елеватори – згідно статистичної інформації в Україні нараховується 1308 зернових елеватора.

У таблиці 5.2 коротко охарактеризовано потенційний ринок.

Таблиця 5.2 – Попередня характеристика потенційного ринку

№	Показники стану ринку (найменування)	Характеристика
1	Головні конкуренти це підприємства, що займаються обслугованням автомобільних ваг	ТОВ «ВІС» м.Дніпро. Займається розробкою та постачанням програмного та апаратного забезпечення для автомобільних ваг на основі цифрових датчиків.
		ТОВ «Техноваги» м.Львів. Займається виробництвом електронних ваг, співпрацює з розробниками програмного забезпечення для автомобільних ваг.
2	Динаміка ринку (якісна оцінка)	Оскільки намітилася тенденція максимально автоматизовувати всі виробничі процеси в Україні все більше підприємств переходять на використання програмного забезпечення та систем автоматичного обліку та контролю, а отже ринок зростає.
3	Наявність обмежень для входу (вказати характер обмежень)	Необхідність охоплення віддалених районів України – це насамперед необхідність виїздів на віддалені підприємства Західної України для встановлення обладнання
4	Специфічні вимоги до стандартизації та сертифікації	Необхідність отримання авторського свідоцтва на розроблений програмний продукт

Загалом за рахунок специфічності роботи елеваторів з необхідністю врахування якісних характеристик зернових культур на ринку України не дуже велика кількість розробників програмного забезпечення займаються процесом розробки подібних систем. Загалом не враховуючи програм написаних, як-то кажуть на «скору руку» адміністраторами комп'ютерних мереж самих елеваторів лише 2 підприємства співпрацюють з зерновими елеваторами в рамках надання програмного забезпечення та систем автоматизації. Це ТОВ «ВІС» та ТОВ «Техноваги». Самописне програмне забезпечення не стандартизоване та, як правило, налаштоване лише на один окремий елеватор та не має можливості взаємодії через мережу інтернет з віддаленим сервером та дані зберігаються локально на окремому комп'ютері. Стандартні рішення, що надаються ТОВ «ВІС» та ТОВ «Техноваги» не дозволяють автоматичне розпізнавання номерних знаків та не мають можливостей віддаленого керування параметрами якості зернових культур.

Загалом єдиною проблемою при входженні на ринок є необхідність наявності доброї маркетингової стратегії та виїздами на потенційні об'єкти. Це пов'язано насамперед з тим, що зернові елеватори розташовуються біля невеликих населених пунктів біля яких багато полів де вирощуються зернові культури та відповідно для пропонування комплексу та подальшого встановлення обладнання необхідно реалізовувати виїзди у віддалені райони.

5.2 Визначення трудомісткості та тривалості

Весь комплекс моделювання можна розділити на етапи. Для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт. У розробці пристрою приймають участь радіотехнік протягом одного місяця і програміст протягом 0,5 місяця. Дослідження починається першого вересня і повинна бути виконана до п'ятнадцятого жовтня 2020 року. Тривалість робіт визначають за формулою 5.1:

$$T_{\text{ц}} = \frac{Q}{R} = \frac{45}{2} = 22{,}5 \quad (5.1)$$

де $T_ц$ - тривалість циклу, днів;

Q - трудомісткість, людино-днів;

R - кількість виконавців, чол.

Отримана інформація зведена в табл. 5.3

Таблиця 5.3 - Завдання та обов'язки по розробці пристрою

Найменування роботи	Трудомісткість		Виконавці	Тривалість, днів
	люд.-дні	%к підсумку		
1. Отримання технічного завдання	6	13,3	Програміст Радіотехнік	3 3
2. Огляд літературних джерел	5	11,1	Радіотехнік	5
3. Аналіз матеріалів та технологій	5	11,1	Радіотехнік	5
4. Розробка конструкції системи	10	22,2	Радіотехнік	10
5. Розробка програмного забезпечення	5	11,1	Програміст	5
6. Тестування системи та програмного забезпечення	10	22,2	Програміст Радіотехнік	5 5
7. Аналіз отриманих результатів	4	8,8	Програміст Радіотехнік	2 2
Разом	45	100		45

За даними табл. 5.3 складається зведений стрічковий графік планування розробки моделі, який представляє собою таблицю, в першому стовпці якої

розміщені в порядку збільшення термінів початку виконання всі види роботи, а навпаки - календарний період їх виконання. Даний графік наведено в табл. 5.4.

Таблиця 5.4 - Зведений стрічковий графік планування розробки

Найменування робіт	Календарний період, дні						
	01.09-06.09.20	07.09-12.09.20	12.09-17.09.20	17.09-27.09.20	27.09-02.10.20	02.10-12.10.20	12.10-16.10.20
1. Отримання технічного завдання							
2. Огляд літературних джерел							
3. Аналіз матеріалів та технологій							
4. Розробка конструкції системи							
5. Розробка програмного забезпечення							
6. Тестування системи та програмного забезпечення							
7. Аналіз отриманих результатів							

- програміст - радіотехнік

5.3 Визначення витрат на розробку пристрою

Для визначення витрат на розробку системи складається калькуляція вартісної вартості робіт, яка включає наступні статті:

- основна заробітна плата;
- додаткова заробітна плата;
- єдиний соціальний внесок (ЄСВ);
- витрати на спеціальне обладнання;
- матеріали і комплектуючі вироби;
- накладні витрати;
- податки.

5.3.1 Розрахунок основної заробітної плати

Витрати за цією статтею складаються з планового фонду зарплати всіх категорій працівників, зайнятих в розробці пристрою. Розрахунок зарплати ведеться на підставі даних про трудомісткості, представлених в табл. 5.3.

Таблиця 5.5 - Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чол.	Місячний оклад, грн.	Кількість місяців роботи	Сума ЗП, грн.
Радіотехнік	1	10000	1	10000
Програміст	1	12000	0,5	6000
Разом	2			16000

5.3.2 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати працівників і розраховують за формулою 5.2:

$$ЗП_{доп} = ЗП_{осн} \cdot 0{,}1 \quad (5.2)$$

Підставивши величину основної заробітної плати в формулу 5.2, отримуємо:

$$ЗП_{доп} = 16000 \cdot 0{,}1 = 1600 \text{ грн}$$

5.3.3 Відрахування на єдиний соціальний внесок

Вони становлять 22% і беруться від основної та додаткової заробітної плати.

$$ОТ = (ЗП_{осн} + ЗП_{доп}) \cdot 0{,}22 \quad (5.3)$$

$$ОТ = (16000 + 1600) \cdot 0{,}22 = 3872 \text{ грн.}$$

5.3.4 Визначення затрат на матеріали

У цю статтю включають вартість основних і допоміжних матеріалів, напівфабрикатів, що купуються, і комплектуючих виробів. Транспортно-заготовчі витрати приймають рівними 3-10% від вартості матеріалів.

Таблиця 5.6 - Витрати на матеріали

Матеріали	Одиниця виміру	Кількість	Ціна за одиницю, грн.	Загальна вартість, грн.
Vkmodule Socket 2	шт.	1	650	650
Бездротовий WiFi маршрутизатор D-Link DIR-615/T4	шт.	1	299	299
IP Камера Hikvision DS-2CD1021-I	шт.	1	1800	1800
Ваговий термінал Keli XK3118T1	шт.	1	1830	1830

Продовження таблиці 5.6

Матеріали	Одиниця виміру	Кіль кість	Ціна за одиницю, грн.	Загальна вартість, грн.
Перетворювач інтерфейсів RS232-USB CH340G	шт.	2	110	220
Виносне дублююче табло YHF-3	шт.	1	5300	5300
Блок живлення 12В 1А для YHF-3	шт.	1	120	120
Блок живлення 12В 3А для відеокамери	шт.	1	250	250
Блок живлення 12В 1А для Socket2	шт.	1	120	120
Разом				10589
Транспортно-підготовчі роботи 5%				529,45
Разом матеріальних затрат				11118,45

Витрати на комплектуючі розраховують за формулою 5.4:

$$M = \sum_{i=1}^{n} (Ц_i \cdot N_i \cdot (1 + K_{т.з.}) - Ц_{io} \cdot N_{io}), \quad (5.4)$$

де M - витрати на покупні напівфабрикати і комплектуючі вироби, грн.;

$K_{т.з.}$ - коефіцієнт, що враховує транспортно-заготівельні витрати;

$Ц_i$ - ціна і-го найменування напівфабрикату і комплектуючого, грн.;

N_i - потреба в і-му напівфабрикаті і комплектуючому;

$Ц_{io}$ - вартість зворотних відходів і-го найменування комплектуючого, грн.;

N_{io} - кількість зворотних відходів і-го найменування;

n - кількість найменувань напівфабрикатів і комплектуючих.

$$Ц_{io} = 0;\ N_{io} = 0;\ К_{т.з.} = 0{,}05;$$

$$М = (1 + 0{,}05)\cdot(10589) = 11118{,}45 \text{ грн.}$$

Разом, витрати на матеріали становлять 11118,45 грн.

5.3.5 Витрати на спеціальне обладнання

У цю статтю входять витрати на придбання, транспортування, монтаж і налагодження нестандартного обладнання.

Практично, в даному випадку, в цій статті враховуються витрати на оплату машинного часу ЕОМ для розробки програмної частини. Для чого необхідно скласти кошторис «витрат на утримання і експлуатацію устаткування» виходячи з якої визначиться вартість одного машино-години роботи ПК, після множення якої на машинний час пішло на розробку програмної частини отримаємо витрати на оплату машинного часу. У табл. 5.5 наведене обладнання яке використовується при роботі.

Таблиця 5.7 – Спеціальне обладнання

Матеріали	Одиниця виміру	Кількість	Ціна за одиницю, грн.	Загальна вартість, грн.
Ноутбук Lenovo IdeaPad 330-15IKB	шт.	1	11000	11000
Artline Business B27	шт.	1	7800	7800
Стіл комп’ютерний	шт.	1	950	950
Крісло комп’ютерне	шт.	1	1100	1100
Разом				20850
Транспортно-підготовчі роботи 5%				1043
Разом				21893

Амортизаційні відрахування визначають за формулою 5.5:

$$A = Ф_б \cdot \frac{H_а}{100}, \quad (5.5)$$

де $Ф_б$ - балансова вартість обчислювальної техніки, грн .;

$H_а$ - норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, для ПК 25%.

Балансова вартість обчислювальної техніки становить 21893 грн.

Отримуємо:

$$А = 21893 \cdot 0,25 = 5473,25 \text{ грн.}$$

Статтю «Експлуатація обладнання» розраховують підсумовуванням витрат на електроенергію і допоміжні комплектуючі.

$$C_е = N_н \cdot Ф_{еф} \cdot К_{зв} \cdot К_{зм} \cdot Ц_е, \quad (5.6)$$

де $N_н$ - номінальна потужність ЕОМ, кВт;

$Ф_{еф}$ - річний ефективний фонд часу роботи ЕОМ, машино-год;

$К_{зв}$ - середній коефіцієнт завантаження за часом;

$К_ш$ - коефіцієнт завантаження по потужності;

$Ц_е$ - ціна одного кВт-год електроенергії, грн./(кВт-ч).

Номінальна потужність ЕОМ - 0,5 кВт. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження за часом і за потужністю рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії становить 2,68 грн.

Отримуємо:

$$C_е = 0,5 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,68 = 1302,5 \text{ грн.}$$

Зарплата обслуговуючого персоналу розраховується за формулою 5.7:

$$ЗП_{обсл} = ФЗП_г \cdot (1 + К_{отч}) \cdot \frac{t_{обсл}}{Ф_{еф.обсл}} \quad (5.7)$$

де $ФЗП_г$ - річний фонд заробітної плати (основної і додаткової) обслуговуючих робітників, грн.;

$К_{отч}$ - коефіцієнт, що враховує відрахування на соціальне страхування і в інші фонди;

$t_{обсл}$ - час протягом року, необхідне на технічне обслуговування ЕОМ, ч/рік;

$Ф_{еф.обсл}$ - річний ефективний фонд часу обслуговуючого персоналу, ч/рік.

Приймаємо умовно, що місячна заробітна плата обслуговуючого персоналу становить 16000 грн., а річний фонд заробітної плати відповідно дорівнює 192000 грн. Річний ефективний фонд робочого часу обслуговуючого ПК працівника дорівнює 1750 год / рік.

$$ЗП_{обсл} = 192000 \cdot (1+0,22) \cdot 12/1750 = 1606,22 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і складає 600 грн.

Стаття «Інші витрати» приймається рівною п'яти відсоткам від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Сума всіх попередніх статей дорівнює 8381,97 грн., 5% від суми складають 419,1 грн.

Розраховані статті витрат на утримання і експлуатацію устаткування внесені в табл. 5.6.

Витрати на оплату машинного часу ЕОМ для розробки програмної частини і налагодження програмних засобів визначаються за формулою 5.8:

$$C_{мо} = P_{екс} \cdot t_{мо}, \tag{5.8}$$

де $C_{мо}$ - витрати на оплату машинного часу, грн.;

$P_{екс}$ - експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино-год.;

$t_{мо}$ - машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино-год.

Таблиця 5.8 - Кошторис витрат на утримання і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація обладнання	5473,25
Експлуатація обладнання (крім витрат на поточний ремонт)	1302,5
Заробітна плата основна і додаткова обслуговуючих робітників з ЄСВ	1606,22
Поточний ремонт обладнання	600
Інші витрати	419,1
Разом	9401,07

Експлуатаційні витрати на одну годину машинного часу використовуваної ЕОМ розраховують діленням суми витрат за кошторисом «Витрати на утримання та експлуатацію обладнання (ЕОМ)» (табл. 5.4) на річний ефективний фонд часу роботи ЕОМ. Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин. В результаті експлуатаційні витрати на одну годину машинного часу рівні:

$$P_{екс} = 9401,07/1800 = 5,22 \text{ грн./машино-год}$$

ЕОМ експлуатується 45 днів в одну зміну, що становить в сумі 360 годин. Таким чином, витрати на оплату машинного часу складуть:

$$С_{мо} = 5,22 \cdot 360 = 1879,2 \text{ грн.}$$

5.3.6 Інші прямі витрати

В інші прямі витрати включаються витрати на яке використовується при розробці системи комерційне програмне забезпечення:

- дольове ПЗ, що використовується постійно при роботі ПК (Windows 10) - 5000 грн. без НДС;

- цільове ПЗ, що купується для даного конкретного завдання (Компас-Электрик) - 5000 грн. без НДС.

$$S_{дол.ПЗ} = \frac{Ц_{ПЗWindows} \cdot T_{КТС}}{Ф_{еф.КТС} \cdot T_{с.ПЗ}} \quad (5.9)$$

$$S_{діл.ПЗ} = Ц_{ПЗ\ А}$$

де $S_{дол.ПЗ}$ - витрати на дольове ПЗ при розробці програмної частини розробляється в розрахунку ПЗ, грн .;

$S_{діл.ПЗ}$ - витрати на цільове ПЗ, що купується виключно для розробки програмної частини в розрахунку ПЗ, грн .;

$Ц_{ПЗWindows}$ - ціна ПЗ Windows (без ПДВ), грн;

$Ц_{ПЗ\ А}$ - ціна ПЗ Компас-Электрик (без ПДВ), грн;

$T_{КТС}$ - машинний час КТС, необхідне користувачеві для розробки програмної частини, машино-год / рік;

$Ф_{еф.ктс}$ - річний ефективний фонд часу роботи КТС, машино-год / рік;

$T_{с.ПЗ}$ - термін служби дольової ПЗ, років.

$$S_{доп.ПЗ} = \frac{10000 \cdot 360}{1800 \cdot 5} = 400 грн.$$

$$S_{діл.ПЗ} = 10000 \text{ грн.}$$

$$S_{\Sigma} = 400 + 10000 = 10400 \text{ грн.}$$

5.3.7 Розрахунок загальновиробничих витрат

До загальновиробничих витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т.д.), утримання та експлуатацію будівель. Загальновиробничі витрати включаються до вартості розробки пристрою непрямим шляхом - у відсотках до основної заробітної плати розробників. В даному випадку накладні витрати становлять 40% до основної заробітної плати розробників, що складає (16000 грн. х 0,4 = 6400 грн).

Результати визначення витрат на розробку пристрою у вигляді калькуляції кошторисної вартості робіт наведені в табл. 5.7.

Таблиця 5.9 - Калькуляція кошторисної вартості робіт з розробки пристрою

№	Найменування статей витрат	Сума, грн.	Питома вага до підсумку,%
1	Основна заробітна плата	16000	22,61
2	Додаткова заробітна плата	1600	2,26
3	ЄСВ	3872	5,47
4	Матеріали	10589	14,96
5	Витрати на спец. обладнання	21893	30,94
6	Інші прямі витрати	10400	14,69
7	Накладні витрати	6400	9,05
8	Разом ($S_{пр}$)	70754	100

5.4 Розрахунок техніко-економічної ефективності моделі

Для теоретичних досліджень у більшості випадків важко чи навіть неможливо розрахувати економічний ефект, тому доцільно визначити їхню техніко-економічну ефективність з урахуванням наступних показників:

- важливості дослідження для народного господарства;
- складності розробки;
- результативності й можливості використання.

Важливість теоретичного дослідження оцінюємо як пошук принципово нових конструктивних і технологічних рішень і ін.

Результативність НДР визначається по повноті рішень поставленого завдання: отриманий результат відповідає планованому, задовільний (часткове рішення) чи негативний.

Аналіз залежності між цими показниками й витратами на їхнє досягнення дає можливість кількісної оцінки техніко-економічної ефективності теоретичних НДР і визначається за формулою (5.10):

$$K_{НДР} = \frac{J^n \cdot R \cdot T}{B_{НДР} \cdot t_{НДР}}, \quad (5.10)$$

де $K_{НДР}$ - рівень ефективності дослідження (коефіцієнт техніко-економічної ефективності НДР):

J^n - важливість роботи;

R - результативність роботи;

T - технічна складність виконання НДР;

$B_{НДР}$ - витрати на проведення НДР, років:

n - показник використання результатів НДР:

n = 0 - результати НДР не використовуються;

n = 1 - результати НДР використовуються частково;

n = 2 - результати НДР використовуються в дослідно-конструкторських роботах (ДКР);

n = 3 - результати НДР можуть бути використані без проведення ДКР.

Для НДР, у яких $B_{НДР}$ >30 тис. грн. і $t_{НДР} \leq 2$ років, можна застосовувати такі значення оцінних факторів наведених в табл. 5.8

Таблиця 5.10 – Значення оцінних факторів

Оцінні фактори	J	R	T	C	$t_ф$	n
Припустимі значення	2…5	1…4	1…3	-	-	1…8
Прийняті значення	4	3	2	-	-	3

Згідно значень з таблиці оцінних факторів, отримуємо такий вираз:

$$К_{ндр} = \frac{4^3 \cdot 3 \cdot 2}{50398{,}6 \cdot 0{,}03} = 1{,}12$$

Таким чином, так як коефіцієнт техніко-економічної ефективності НДР $K_{НДР} \geq 1$, в нашому випадку рівний $K_{НДР}$ = 1,12, то дослідницька робота вважається ефективною.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

6.1 Аналіз потенційних небезпек

Тема дипломного проекту «Розробка програмно-апаратного комплексу обліку та контролю роботи зернових елеваторів за широким рядом показників», тому розглянемо робоче місце інженера радіотехніка.

Основними потенційними небезпеками при проведенні робіт у лабораторії є такі:

- небезпека ураження електричним струмом, внаслідок недотримання правил електробезпеки або виходу з ладу електроприладів;
- порушення роботи кістково-м’язового апарату внаслідок тривалих статичних навантажень при роботі з ПК.
- нервово-психічні перевантаження внаслідок постійного контакту з клієнтами, колегами по роботі, керівництвом при вирішенні робочих питань, які можуть носити конфліктний характер і призвести до емоційного дискомфорту, внутрішнього роздратування, емоційної нестабільності та захворювань нервової системи;
- незадовільні ергономічні характеристики робочого місця внаслідок нераціонального планування робочого місця, що може призвести до механічних травм, уражень електричним струмом та порушень кістковом’язового апарату;
- негативний вплив недостатнього освітлення робочої зони на зір та продуктивність роботи працюючого, внаслідок несправності освітлювальних приладів або неправильного проектування освітлювальної системи;
- негативний вплив незадовільних параметрів повітряного середовища робочої зони на здоров’я працюючого, внаслідок неправильного проектування системи вентиляції або несправності її несправності;
- негативний вплив підвищеного рівня шуму на психоемоційний стан працюючого, який пов’язаний з використанням застарілої периферійної техніки, кондиціонерів, копіювальної техніки, освітлювальних приладів;

- небезпека загоряння у зв'язку із несправністю електричного обладнання, недотримання, або порушення правил протипожежної безпеки обслуговуючим персоналом, що може призвести до пожежі.

Приведення цих чинників у відповідність до сучасних норм, нормативів і стандартів є передумовою нормальної працездатності людини.

Небезпечні і шкідливі наслідки для людини від дії електричного струму, електричної дуги, електричного і магнітного полів, електростатичного поля і ЕМВ проявляються у вигляді електротравм, механічних пошкоджень і професійних захворювань. Ступінь впливу залежить від експозиції фактора, в тому числі: роду і величини напруги і струму, частоти електричного струму, шляху струму через тіло людини, тривалості впливу електричного струму або електричного і магнітного полів на організм людини, умов зовнішнього середовища.

6.2 Заходи по забезпеченню техніки безпеки

Приміщення лабораторії, в яких перебувають співробітники галузі радіотехніка, відносяться до приміщень без підвищеної небезпеки ураження електричним струмом.

Організаційні та технічні заходи щодо забезпечення електробезпеки (ДСТУ 7237:2011) полягають у відповідному навчанні, інструктажі і допуску до роботи осіб, дотриманні особливих вимог при роботах на струмопровідних частинах, що знаходяться під напругою, або поблизу них.

Важливим заходом, що забезпечує електробезпеку обслуговуючого персоналу, є захисне заземлення або занулення металевих неструмоведучих частин електрообладнання. Відповідно до «Правил улаштування електроустановок» застосовується захисне заземлення, виконане для забезпечення електробезпеки, їм називається навмисне металеве з'єднання із заземлюючим пристроєм елементів електроустановок, які нормально не знаходяться під напругою. Занулення в електроустановках і мережах напругою до 1000 В - це навмисне електричне з'єднання металевих елементів установки, нормально ізольованих від частин, що

знаходяться під напругою з глухозаземленою нейтраллю генератора або трансформатора в мережах змінного струму, а також з глухозаземленою середньою точкою в трьохпровідних мережах постійного струму з нульовим проводом.

Основними причинами поразки персоналу електричним струмом є:

- дотик до струмоведучих частин, що знаходяться під напругою (оголені кабелі живлення, розетки - 220В);

- дотик до металевих неструмоведучих частин електроустановок або пов'язаного з електроустановками виробничого обладнання (корпусу, кожуха, огорожі і т. Д.) Після переходу на них напруги з струмоведучих частин.

Захисні заходи від ураження електричним струмом:

1) організаційні заходи:

- спеціальне навчання персоналу;

- інструктажі персоналу;

- призначені особи відповідальної за безпеку;

- проводяться періодичні огляди, вимірювання та випробування електрообладнання.

2) технічні заходи:

- застосовані пристрої (запобіжники, що вимикають реле і т. д.) Захист електрообладнання і мереж від перевантажень, а також струмів коротких замикань;

- захист людей від дотику до струмоведучих частин обладнання забезпечено за допомогою ізоляції струмоведучих частин електрообладнання;

- захист від ураження електричним струмом при переході напруги на металеві корпуси електроустановок здійснюється пристроєм захисного заземлення, занулення електрообладнання в мережах з глухо-заземленою нейтраллю, застосуванням захисного відключення.

6.3 Заходи по забезпеченню виробничої санітарії та гігієни праці

Площа приміщень, в яких розташовують персональні комп'ютери, визначають згідно з чинними нормативними документами. Згідно ДСанПіН 3.3.2.007-98 з розрахунку на одне робоче місце, обладнане ПК, встановлені такі норми:

- площа - не менше ніж 6,0 $м^2$;

- обсяг - не менше ніж 20,0 $м^3$.

Для внутрішнього оздоблення інтер'єру приміщень з комп'ютерами використовуються дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі - 0,7-0,8; для стін - 0,5-0,6; для підлоги - 0,3-0,5.

Приміщення для роботи з персональними комп'ютерами обладнані системами опалення, кондиціонування повітря і припливно-витяжною вентиляцією. У приміщеннях на робочих місцях забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості і рухливості повітря відповідно до норм і правил, а також ДБН В.2.5-67 діє до: 2013 «Опалення, вентиляція и кондіцюювання», затверджених наказом Мінрегіону від 25.01.2013 р № 24.

Для підтримки допустимих значень мікроклімату та концентрації позитивних і негативних іонів передбачені установки або прилади зволоження та / або штучної іонізації, кондиціювання повітря.

Освітлення має важливе санітарно-гігієнічне значення. Зі збільшенням ступеня освітленості підвищується продуктивність праці (іноді на 15% і більше) і якість робіт, знижується виробничий травматизм і аварійність.

Робочі місця, згідно п. 4.3 ДСанПіН 3.3.2.007-98, розташовуються щодо світлових прорізів так, щоб природне світло падало переважно з лівого боку.

Робоче приміщення відповідає вимогам роботи високої точності з розрядом зорової роботи III, подразряд "В" за умовами ДБН В.2.5 - 28 - 2006 "Природне и штучне освітлення" освітлення повинно бути не менше 300 люкс.

Приміщення, їх розміри (площа, обсяг) повинні в першу чергу відповідати кількості працюючих і розташованому в них комплекту технічних засобів. У них

передбачаються відповідні параметри температури, освітлення, чистоти повітря, забезпечують ізоляцію, від виробничих шумів і т.п.

При невеликій забрудненості зовнішнього повітря кондиціонування приміщень здійснюється зі змінними витратами зовнішнього повітря і рециркуляційного.

Висота залу над технологічним підлогою до підвісної стелі повинна бути 3 - 3,5 м. Відстань між підвісною і основною стелями при цьому має бути 0,5 - 0,8 м. Висоту підпільного простору приймають рівною 0,2 - 0,6 м.

Рівні шуму на робочих місцях користувачів персональних комп'ютерів не перевищують значень, встановлених СанПіН 2.2.4 / 2.1.8.562-96 і становлять не більше 50 дБА.

Зниження рівня шуму в приміщеннях можливе використанням звукопоглинальних матеріалів з максимальними коефіцієнтами звукопоглинання в області частот 63-8000 Гц для обробки стін і стелі приміщень. Додатковий звуковбирний ефект створюють однотонні фіранки з щільною тканини, повішені в складку на відстані 15-20 см від огорожі. Ширина фіранки в 2 рази більше ширини вікна.

Робочий стіл відповідає сучасним вимогам ергономіки і дозволяє зручно розмістити на робочій поверхні обладнання з урахуванням його кількості, розмірів і характеру виконуваної роботи. Також застосовуються столи, які мають окрему від основної стільниці спеціальну робочу поверхню для розміщення клавіатури. Висота столів знаходиться в межах від 680 до 800 мм.

Глибина робочої поверхні столу становить 600-800 мм, ширина - відповідно 1200-1600 мм. Робоча поверхня стола не має гострих кутів і країв, а також має матову або напівматову фактору.

Робочий стіл має простір для ніг висотою не менше 600 мм, шириною - не менше 500 мм, глибиною на рівні колін - не менше 450 мм і на рівні витягнутих ніг - не менше 650 мм.

Режим праці і відпочинку передбачає дотримання певної тривалості безперервної роботи на ПК і перерв, регламентованих з урахуванням тривалості робочої зміни, видів і категорії трудової діяльності.

Види трудової діяльності на ПК поділяються на 3 групи: група А - робота з зчитування інформації з екрану з попереднім запитом; група Б - робота з введення інформації; група В - творча робота в режимі діалогу з ПК.

При 8-годинній робочій зміні і роботі на ПК регламентовані перерви слід встановлювати:

- для першої категорії робіт через 2 години від початку зміни і через 2 години після обідньої перерви тривалістю 15 хвилин кожен;

- для другої категорії робіт - через 2 години від початку робочої зміни і через 1,5-2,0 години після обідньої перерви тривалістю 15 хвилин кожен або тривалістю 10 хвилин через кожну годину роботи;

- для третьої категорії робіт - через 1,5- 2,0 години від початку робочої зміни і через 1,5-2,0 години після обідньої перерви тривалістю 20 хвилин кожен або тривалістю 15 хвилин через кожну годину роботи.

При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перерв при 8-годинній робочій зміні, а протягом останніх 4 годин роботи, незалежно від категорії і виду робіт, щогодини тривалістю 15 хвилин.

Тривалість безперервної роботи на ПК без регламентованого перерви не повинна перевищувати 2 години.

При роботі на ПК в нічну зміну тривалість регламентованих перерв збільшується на 60 хвилин незалежно від категорії і виду трудової діяльності.

Ефективними є нерегламентовані перерви (мікропаузи) тривалістю 1-3 хвилини.

Регламентовані перерви і мікропаузи доцільно використовувати для виконання комплексу вправ і гімнастики для очей, пальців рук, а також масажу. Комплекси вправ доцільно змінювати через 2-3 тижні.

Користувачам ПК, який виконує роботу з високим рівнем напруженості, показана психологічне розвантаження під час регламентованих перерв і в кінці робочого дня в спеціально обладнаних приміщеннях (кімнатах психологічного розвантаження).

Необхідно розрахувати систему загального рівномірного освітлення з люмінесцентними лампами для приміщення.

Розміри приміщення: довжина a = 9м, ширина b = 6м, висота H =3,2м. Приміщення має світлу побілку: коефіцієнт відбиття $\rho_{стелі}$ = 70%, $\rho_{стін}$ = 50%. Висота робочих поверхонь (столів) h_p = 0,8м, $E_н$ = 200лк. Як освітлювальні прилади приймаємо світильники типу ЛПО01 (з двома лампами), які доцільно використовувати в нашому випадку. Оскільки світильники кріпляться до стелі, то їх висота над підлогою практично дорівнює висоті приміщення h_O = 3,2 м, що не суперечить вимогам ДБН В.2.5-28-2006, відповідно до яких h_O = 2,6…4м, коли в світильнику менше 4 ламп.

Показник приміщення (чисельне значення індексу приміщення) i становить:

$$i = \frac{ab}{h(a+b)}, \qquad (6.1)$$

де a та b – довжина і ширина приміщення, м;

h – висота світильника над робочою поверхнею, м.

$$i = \frac{9 \cdot 6}{2{,}4(9+6)} = 1{,}5.$$

Визначаємо кількість рядів світильників у приміщенні:

$$N_p = \frac{B}{(H-h_p)\cdot[L/h]} \qquad (6.2)$$

де [L/h] – числове значення коефіцієнта світильника [12, додаток В]

Для світильника ЛПО L/h=1,4

$$N_p = \frac{6}{(3,2-0,8)\cdot 1,4} = 1,78 \approx 2 \qquad (6.3)$$

Визначаємо максимально допустиму відстань між рядами:

$$L_{max} = \frac{B}{N_p} = \frac{6}{2} = 3\text{м} \qquad (6.4)$$

Визначаємо висоту підвісу світильника над робочою поверхнею:

$$h = \frac{L_{max}}{[L/h]} = \frac{3}{1,4} = 2,14\text{м} \qquad (6.5)$$

Знаходимо висоту звисання світильника від стелі:

$$h_3=H-h_p-h=3,2-0,8-2,14=0,26\text{м} \qquad (6.6)$$

При i = 1,5, $\rho_{стелі}$ = 70%, $\rho_{стін}$ = 50%, $\rho_п$ = 30% для світильника ЛПО01 коефіцієнт використання дорівнює η = 0,5.

Визначаємо сумарний світловий потік освітлювальної установки у даному виробничому приміщенні:

$$\Phi_\Sigma = \frac{E_н S k_з Z}{\eta}, \qquad (6.6)$$

де Φ_Σ – розрахункове значення сумарного світлового потоку у приміщенні, лм;

$E_н$ – нормоване значення освітленості, лк;

S – площа освітлювального приміщення, м^2;

$k_З$ – коефіцієнт запасу, який враховує зниження освітленості в результаті забруднення і старіння ламп;

Z – коефіцієнт нерівномірності освітлення (Z = 1,1 для люмінісцентних ламп);

η – коефіцієнт використання світлового потоку.

$$\Phi_{\sum} = \frac{200 \cdot 54 \cdot 1{,}5 \cdot 1{,}1}{0{,}5} = 35640\ лм$$

Визначаємо розрахункову загальну кількість світильників N* у приміщені, виходячи з позиції розташування їх у вершинах квадрата:

$$N^* = AB/L^2_{max} = 9 \cdot 6/3^2 = 6 \qquad (6.8)$$

Визначаємо розрахунковий світловий потік лампи $\Phi_л^*$:

$$\Phi_л^* = \frac{\Phi_{\sum}}{N_л} \qquad (6.9)$$

де $N_л$ – загальна кількість ламп у приміщенні;

$$N_л = N^* \cdot n = 6 * 2 = 12 \qquad (6.10)$$

де n – кількість ламп у світильнику;

$$\Phi_л^* = \frac{35640}{12} = 2970\ лм$$

Обираємо стандартний світильник в якому встановлено дві лампи ЛБ-40, світловий потік однієї такої лампи складає $\Phi_л$ = 3200лм, довжина лампи 1,2м.

Коефіцієнт пропорційності для обраної лампи:

$$m = \Phi_л^* / \Phi_л = 2970/3200 = 0{,}928 \qquad (6.11)$$

Потужність освітлювальної установки:

$$P_{\sum}=N_{фл}\cdot P_{л} = 12*40=480 \text{ Вт} \quad (6.12)$$

6.4 Заходи безпеки у надзвичайних ситуаціях

6.4.1 Заходи з пожежної безпеки

Пожежна безпека забезпечується системою запобігання пожежі і системою пожежного захисту. У всіх службових приміщеннях обов'язково повинен бути «План евакуації людей при пожежі», що регламентує дії персоналу в разі виникнення вогнища і вказує місця розташування пожежної техніки.

Продумуючи евакуацію співробітників при можливій пожежі, враховують особливості лабораторного приміщення. На плані позначені шляхи евакуації до сходових кліток або виходу на вулицю. Також зображений запасний шлях, по якому можна швидко покинути будівлю. На плані також вказують, де розташовані ручні пожежні сповіщувачі через часті переноси і перепланування, вони домальовуються вручну.

Приміщення з ЕОМ оснащені системою автоматичної пожежної сигналізації відповідно до вимог Переліку однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації, затвердженого наказом МВС України від 20.11.97 №779 (z0567-97) і зареєстрованого в Міністерстві юстиції України 28.11.97 за №567 / 2371, і СНіП 2.04.09-84 «Пожежна автоматика будинків і споруд» з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками ВВК – 1.4 з розрахунку 2 шт. на кожні 20 $м^2$ площі приміщення з урахуванням граничнодопустимих концентрацій вогнегасної рідини відповідно до вимог Правил пожежної безпеки в Україні і вимог нормативно-технічної та експлуатаційної документації заводу-виробника.

Вогнегасники розташовуються відповідно до вимог ГОСТ 12.4.009-83 таким чином, щоб вони були захищені від впливу прямих сонячних променів, будь-яких механічних впливів і інших несприятливих чинників, таких як вібрація, підвищена

вологість та інших. Вогнегасники розміщуються в легкодоступних і помітних місцях. Не допускається зберігання і експлуатація вогнегасників в місцях, де температура може перевищувати 50 °С і під прямими променями сонця. Не рідше одного разу на рік перевіряється маса заряду вогнегасників, згідно з рекомендаціями. Експлуатація вогнегасників без чеки і пломби заводу-виготовлювача або організації, що проводила перезарядку, не дозволяється.

Під час монтажу та експлуатації ліній електромережі повністю виключені можливості виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежено застосування проводів з легкозаймистою ізоляцією, по можливості застосовується негорюча ізоляція. У приміщенні, де одночасно експлуатуються понад п'яти комп'ютерів, на видному і доступному місці встановлений аварійний резервний вимикач, який може повністю відключити електричне живлення приміщення, крім освітлення.

Клас пожежі - це умовно прийнята характеристика об'єкта пожежі в залежності від виду горючих речовин і матеріалів для зручності позначення вогнегасних речовин і (або) засобів гасіння (вогнегасників, установок пожежогасіння) пожежі.

У нашому випадку клас А, Е.

Класифікація виробництв за пожежною небезпекою (точніше, вибухопожежної) досить складна.

У нашому випадку приміщення відноситься до категорії Д (це виробництва, пов'язані з обробкою негорючих речовин і матеріалів в холодному стані)

При будівництві або експлуатації будівлі і приміщення враховується ступінь їх вогнестійкості. Тобто здатності протистояти поширенню вогню без втрати експлуатаційних властивостей. Ступінь вогнестійкості для виробничих будівель встановлюється будівельними нормами і правилами (СНіП 31-03-2001) і залежить від категорії з вибухопожежної та пожежної небезпеки. Всі види споруд ділять на п'ять ступенів (позначаються римськими цифрами I-V).

При визначенні ступеня вогнестійкості проводиться аналіз параметрів приміщення (призначення, поверховість, площа), а також якісних показників і рівня

займистості матеріалів, які застосовувались для спорудження об'єкта – III (у нашому випадку).

Практично на всі вогнегасники маркування наноситься у вигляді етикетки, в якій вказана вся необхідна споживачеві інформація. На прикладах інформації, нанесеної на етикетки порошкового і вуглекислотного вогнегасників, можна розглянути загальні правила, яким повинні керуватися виробники при маркуванні вогнегасників.

Позначення типів вогнегасників:

ВВ - вогнегасник водяний;

ВВП - вогнегасник водопінний;

ВВПА - пристрій вогнегасний водопінний аерозольний;

ВГ - вогнегасник газовий, у тому числі вуглекислотний (ВВК);

ВП - вогнегасник порошковий.

Цифра після позначення типу вогнегасника означає масу вогнегасної речовини в кілограмах (для ВВПА - в грамах), що міститься в його корпусі. Наприклад, ВП-5 - вогнегасник порошковий з масою вогнегасної речовини 5 кг.

Під час вибору типу і необхідної кількості вогнегасників як елементів протипожежного захисту об'єкта слід також керуватися галузевими правилами пожежної безпеки, нормами технологічного проектування та іншими нормативно-правовими актами, що регламентують вимоги до оснащення об'єктів вогнегасниками.

Критеріями вибору типу і необхідної кількості вогнегасників для захисту об'єкта є:

1) категорія виробничого та складського приміщення за вибухопожежною та пожежною небезпекою;

2) клас можливої пожежі;

3) придатність вогнегасника для гасіння пожежі певного класу та відповідність умовам його експлуатації;

4) вогнегасна здатність вогнегасника конкретного типу за ДСТУ 3675-98 "Пожежна техніка. Вогнегасники переносні. Загальні технічні вимоги та методи

випробувань", ДСТУ 3734 (ГОСТ 30612-99) "Пожежна техніка. Вогнегасники пересувні. Загальні технічні вимоги";

5) гранична захищувана площа.

Класи пожеж визначено в ДСТУ EN 2:2014 "Класифікація пожеж" (EN 2:1992, EN 2:1992/A1:2004, IDT).

Категорія будинків, приміщень та зовнішніх установок виробничого і складського призначення за вибухопожежною або пожежною небезпекою визначається відповідно до вимог ДСТУ Б В.1.1-36:2016 "Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою".

Якщо на об'єкті можливі осередки пожеж різних класів, слід обирати вогнегасники окремо для кожного класу пожежі або віддавати перевагу більш універсальному вогнегаснику. При виборі таких вогнегасників їх кількість має дорівнювати більшому значенню, що отримане для кожного класу пожежі окремо.

За потреби використання різних типів вогнегасників допускається здійснювати заміну одного типу на інший із забезпеченням рівності сумарної вогнегасної здатності за класом пожежі, характерної для цього об'єкта. Коефіцієнти ефективності вогнегасників за їх вогнегасною здатністю щодо гасіння модельних вогнищ пожеж класів А та В наведено в додатку 9 до цих Правил. Наприклад, порошковий вогнегасник ВП-9 для пожежі класу В, що має коефіцієнт ефективності 13, можна замінити на два вогнегасники - порошковий ВП-6 (має коефіцієнт ефективності 8) та водопінний ВВП-6 (має коефіцієнт ефективності 5), які мають сумарний коефіцієнт ефективності 13.

Об'єкти різного призначення оснащуються переносними вогнегасниками, перелік яких наведено в додатку 10 до цих Правил, та пересувними вогнегасниками, перелік яких наведено в додатку 11 до цих Правил.

Будинки адміністративного та побутового призначення і громадські будинки на кожному поверсі повинні мати не менше двох переносних (порошкових, водопінних або водяних) вогнегасників з масою заряду вогнегасної речовини 5 кг і більше.

Крім того, слід передбачати по одному газовому вогнегаснику з величиною заряду вогнегасної речовини 3 кг і більше:

на 20 м^2 площі підлоги в офісних приміщеннях з оргтехнікою, коморах, електрощитових, вентиляційних камерах та інших технічних приміщеннях;

на 50 м^2 площі підлоги в приміщеннях архівів, машзалів, бібліотек, музеїв.

Приміщення, у яких розміщено оргтехніку, слід оснащувати переносними газовими вогнегасниками з розрахунку один вогнегасник ВВК-1,4 чи ВВК-2, але не менше ніж один вогнегасник зазначених типів на приміщення.

Для захисту квартир багатоквартирних житлових будинків і будинків індивідуальної забудови слід використовувати переносні вогнегасники з розрахунку один водяний (ВВ-5, ВВ-6), або водопінний (ВВП-6), або один порошковий (ВП-2, ВП-3) вогнегасник на одну квартиру або на один будинок індивідуальної забудови.

Додатково будинки та приміщення, зазначені в пунктах 5 - 7 цього розділу, можуть оснащуватися ВВПА з масою заряду вогнегасної речовини 400 г і більше.

Для захисту приміщень, призначених для виготовлення кулінарної продукції та (або) приготування їжі, слід використовувати переносні вогнегасники з можливістю гасіння пожежі класу F з розрахунку один вогнегасник на одне окреме робоче місце для виготовлення кулінарної продукції та (або) приготування їжі.

6.4.2 Заходи по забезпеченню безпеки у надзвичайних ситуація

Згідно з нормами Технічного регламенту засобів індивідуального захисту, затв. постановою КМУ від 27.08.2008 № 761, засоби індивідуального захисту (далі - ЗІЗ) - це спорядження, призначене для носіння користувачем та (або) забезпечення його захисту від одного або декількох видів небезпеки для життя або здоров'я.

Засоби індивідуального захисту на виробництві за призначенням класифікують на такі види:

- спеціальні ізолюючі костюми;
- спорядження для захисту органів дихання;
- спецодяг;

- спецвзуття;

- засоби захисту голови, рук, обличчя, органів слуху, очей;

- захисні дерматологічні засоби;

- комплексні засоби захисту.

Індивідуальні засоби захисту органів дихання Даний вид ЗІЗ класифікують за принципом дії і поділяють на фільтрувальні (Ф) і ізолюючі (І). Якщо перші подають в зону дихання очищене повітря з робочої зони, то другі забезпечують подачу повітря із спеціальних резервуарів або чистого середовища, яка знаходиться поза робочою зоною. За призначенням індивідуальні засоби захисту органів дихання поділяють на: протиаерозольні (пилозахисні); протигазові (газозахисні); універсальні (пилегазозащітние). Ізолюючі ЗІЗ органів дихання в свою чергу діляться на шлангові і автономні. Автономні ЗІЗ органів дихання мають власне джерело дихальної суміші, який знаходиться в корпусі. Ці джерела бувають двох типів - резервуарні і генеративні.

Індивідуальні засоби захисту людини. Спецодяг та спецвзуття Спецодяг - це різноманітні костюми, куртки, комбінезони, халати, плащі, фартухи тощо. Спецодяг ділиться на спеціальні групи: М, 3, Т, Р, Е, П, Я, В, К, Щ, В, Н, Б, що відповідають тій середовищі, для якої спецодяг призначений. Основні вимоги до ізолюючих костюмів, призначеним для захисту від небезпечних і шкідливих факторів хімічно агресивних середовищ і т.д .: повинен бути герметичним і виключати попадання води, поверхнево-активних речовин, газів, парів, рідин; не повинен втрачати захисних властивостей при низькій або високій температурі; повинен забезпечувати захисні властивості при наявності в навколишньому середовищі одночасно декількох шкідливих факторів.

Спецвзуття також класифікується відповідно до критеріїв їх подальшої експлуатації. До спецвзуття належать різноманітні чоботи, діелектричні боти, черевики, напівчеревики, валянки, бахіли, калоші тощо. Окремі види спецвзуття мають спеціальну посилену підошву від запобігання поранень ступні від гострих предметів на будівельному майданчику. Також існує взуття для захисту від вібрації тощо.

Засоби індивідуального захисту рук ЗІЗ для захисту рук від забруднень або для недопущення удару електрострумом, недопущення опіків від агресивних середовищ, від обморожень і т.п., налічує величезну кількість рукавичок і рукавиць, яких сьогодні маса на ринку. Виготовляють їх з бавовни, льону, шкіри, шкірозамінника, гуми, азбесту, полімерів.

Дерматологічні засоби індивідуального захисту Порівняно новим видом індивідуальних захисних засобів для рук є дерматологічні засоби, багато з яких використовуються і для захисту особи. Вони не тільки захищають шкіру під час роботи, але і ефективно очищають її від стійких забруднень. Виділяють 3 групи дерматологічних ЗІЗ: захисні, очищувальні, що регенерують. Такі ЗІЗ можуть бути у вигляді кремів, гелів, спреїв. Для захисту працівників, що контактують з олійними (водонерозчинними) робочими матеріалами (нафтопродуктами, маслами, фарбами, клеями, смолами, сажею, кіптявою і т.д.), застосовуються захисні креми гідрофільної дії. Для захисту працівників, що мають справу з розчинами солей, кислот, лугів, мастильно-охолоджуючими рідинами, миючими та дезінфікуючими засобами, вапном, цементом застосовують креми гидрофобної дії. Існують креми універсальні, що захищають від всіх перерахованих вище факторів.

Засоби індивідуального захисту голови Застосування цих ЗІЗ направлено на зниження травматизму співробітників на виробництві. Згідно з нормами і правилами безпеки, вони повинні використовуватися всюди, де є ймовірність падіння на голову будь-яких предметів. Класифікація ЗІЗ цієї категорії передбачає: захисні каски, каскетці, шапки, шоломи, ковпаки і так далі. Найбільш поширеними в цій категорії ЗІЗ є каски, які можуть бить6 загального призначення (газовики, будівельники і т.д.), спеціального (шахтарі, лісоруби). Основним матеріалом для їх виготовлення є пластмаса різного ступеня міцності з додаванням інших речовин (визначається конкретними умовами експлуатації).

Засоби індивідуального захисту очей Засоби індивідуального захисту працівників також повинні забезпечувати і захист очей. У більшості випадків з цією метою використовують захисні окуляри, а для деяких видів робіт - захисні маски

Індивідуальні засоби захисту органів слуху До ЗІЗ органів слуху відносяться: навушники, протишумні шоломи, протишумові вкладиші.

Відповідно до Положення про забезпечення працівників ЗІЗ № 53 на підприємстві повинен бути документ, який встановлює нормативи видачі засобів індивідуального захисту (положення, наказ і графіки видачі).

До цього документа завжди додаються правила з використання працівником ЗІЗ, порядок зберігання, а також відповідальність за псування або втрату виданого майна. Кожен співробітник, якого беруть на роботу, повинен бути ознайомлений з даними правилами, підписати їх в установленому порядку і неухильно дотримуватися. При отриманні нових комплектів засобів індивідуального захисту працівник ставить підпис в журналі обліку. На кожного працівника заводиться спеціальна облікова картка, в якій вказуються зростання співробітника, його розміри тіла, голови, ноги, а також набір засобів індивідуального захисту відповідно до обсягів робіт, які він виконує.

ВИСНОВКИ

Підвищення керованості елеватора для власника, в першу чергу, залежить від впровадження ефективної системи контролю його діяльності. При цьому для організації максимального контролю на підприємствах по зберіганню і переробці зерна необхідно застосовувати в комплексі і в оптимальній пропорції організаційні заходи, апаратні засоби та програмні комплекси.

Крім цього, просто створити систему контролю на елеваторі, вибравши описані вище засоби, недостатньо. Необхідно постійно її вдосконалювати для виключення появи нових способів обходу контролю.

В результаті виконання магістерської роботи, організація системи контролю на елеваторі дозволить: виключити зловживання з боку персоналу, знизити втрати зерна в сховищах, скоротити час перебування машин на території підприємства; зменшити число зауважень і претензій від клієнтів, знизити витрати підприємства і, внаслідок, відчутно підвищити його прибутковість.

Як показує практика, впровадження системи контролю на елеваторі дозволяє підвищити ефективність управління підприємством, зменшити втрати зерна в сховищах на 10-15%, скоротити час перебування машин на території підприємства на 20-30%; знизити витрати підприємства на 5-10%; зменшити число зауважень і претензій від клієнтів на 30-50%.

На додаток до мінімального варіанта система дозволяє контролювати:

- час перебування транспорту на об'єкті,
- дані по проходженню «контрольних точок»,
- можливість побудови знеособленої роботи лабораторії,
- більш глибокі контрольні функції, що дозволяють не тільки постфактум виявити порушення, але і роблять їх вельми і вельми важкореалізованими в реальному масштабі часу.

На додаток до базового: система відеоспостереження, електронний автоматичний пробовідбірник, шлагбауми, світлофори, датчики положення, зчитувачі електронних міток та карток і т.д.

Крім чисто числових даних, в системі зберігається інформація про візуальні параметрах транспорту, стани платформ ваг в момент зняття ваги, вигляд привезеної продукції, обстановці на ключових ділянках підприємства в процесі виробництва. Також стає можливим обмежити доступ процес проведення лабораторних аналізів на зовсім іншому якісному рівні. Можливість побудови «силосної дошки», де в реальному масштабі часу видно, що і де зберігається і в якому обсязі і куди переміщається.

ПЕРЕЛІК ПОСИЛАНЬ

1. ДСТУ 4138:2002. Семена сельскохозяйственных культур. Методы определения качества.

2. ДСТУ 7270:2012. Метрология. Приборы весовые эталонные. Общие технические требования, порядок и методы аттестации.

3. ДСТУ 7453:2013. Зерно, зернобобовые и продукты их переработки. Определение содержания кадмия, свинца и мышьяка методом атомно-абсорбционной спектрофотометрии с электротермической атомизацией.

4. ДСТУ 7670:2014. Сырье и продукты пищевые. Приготовление проб. Минерализация для определения содержания токсичных элементов.

5. ДСТУ ISO 712:2015. Зерновые и продукты их них. Определение содержания влаги. Контрольный метод (ISO 712:2009, IDT).

6. ДСТУ ISO 3093:2009. Пшеница, рожь и мука из них. Пшеница твердая и манные крупы из твердой пшеницы. Определение числа падения методом Хагберга-Пертена (ISO 3093:2004, IDT).

7. ДСТУ ISO 6639-4. Зерновые и бобовые. Определение скрытого заселения насекомыми. Часть 4. Ускоренные методы (ISO6639-4:1987, IDT).

8. ДСТУ ISO 6644:2008. Зерновые и продукты их помола. Автоматический отбор проб механическими средствами (ISO 6644:2002, IDT).

9. ДСТУ ISO 20483:2016. Злаковые и бобовые культуры. Определение содержания азота и сырого протеина методом Къельдаля (ISO 20483:2013, IDT).

10. ДСТУ ISO 6322-1:2004. Хранение зерновых и бобовых. Часть 1. Основные положения (ISO 6322-1:1996, IDT).

11. ДСТУ ISO 6322-2:2004. Хранение зерновых и бобовых. Часть 2. Практические рекомендации (ISO 6322-2:2000, IDT).

12. ДСТУ ISO 6322-3:2004. Хранение зерновых и бобовых. Часть 3. Защита от вредителей (ISO 6322-3:1989, IDT).

13. Закон України «Про зерно та ринок зерна в Україні» від 04.07.2002 р. N 37-IV

14. Наказ Міністерства аграрної політики України «Про затвердження Технічного регламенту зернового складу» від 15.06.2004 р. N 228.

15. Наказ Міністерства аграрної політики України «Про затвердження Інструкції про ведення обліку й оформлення операцій із зерном і продуктами його переробки на хлібоприймальних та зернопереробних підприємствах» від 13.10.2008 р. N 661.

ДОДАТОК А – ОСНОВНА ЧАСТИНА СКРИПТУ ОТРИМАННЯ ДАНИХ НА СЕРВЕРІ

```
<?php
//Скрипт для приема запросов с программы для сохранения изменений и БД
   header("Content-Type: text/html; charset=UTF-8");
        include 'Settings.php';

//преобразовываем полученный запрос в массив
// Получить JSON как строку
        $json_str = file_get_contents('php://input');
//!!!временно сохраним в журнал
        saveLog($json_str);
//Получить объект
        $json_obj = json_decode($json_str);
        $ThisCommand = $json_obj->Command;
        $IdScale = $json_obj->IdScale;
        $ScaleKey = $json_obj->ScaleKey;

//соединяемся с БД
        $dbcnx = @mysqli_connect($dblocation,$dbuser,$dbpasswd,$dbname);
        if (!$dbcnx) // Если дескриптор равен 0 соединение не установлено
   {
     echo("ERROR_DB_CONNECT_0<br>");
     exit();
   }
        if ($dbcnx>0)
        {
                //echo("DB_CONNECT_OK<br>");
        }
// Код соединения с базой данных
        if (!@mysqli_select_db($dbcnx,$dbname))
        {
        echo( "ERROR_DB_CONNECT_1<br>" );
        exit();
        }
//Указываем кодировку для записи в БД
        mysqli_query($dbcnx,"SET NAMES UTF8");
        mysqli_query($dbcnx,"SET CHARACTER SET UTF8");

//проверяем наличие таких весов с ИД и соответствия им ключа
//проверяем чтобы параметрами не были пустыми
        if (($IdScale == "")||($ScaleKey == "")) {
                echo("not enough parameters");
                exit();
        }
// проверяем, чтоб был такой ИД в БД
        $idindb = ScalesIDInDB($dbcnx,$IdScale);
        //echo("IDinDB: ".$idindb);//!!!
        if ($idindb=="0") {
                echo( "INCORRECT ID" );
                exit();
        }
// проверяем, чтоб б данному ИД соответствовал высланный ключ
        $dbkey = GetSKeyByID($dbcnx,$idindb);
        if ($dbkey!=$ScaleKey) {
                echo( "INCORRECT KEY" );
                exit();
        }
```

```
//если ключ соответствует обрабатываем соответствующую команду

//Синхронизируем таблицу грузов
/**{“Command”:101,  “IdScale”:”XXXXX”,  "ScaleKey":"XXXXXXXXXXXXXXXX","Cargos”:[{“IdOnObject”:XXX,
“CargoName”:”xxxxx”, “CargoPrice”:XX.XX,
        “CargoCode”:”XXXXXX”, “Activity”:X},{}…]} **/
        if ($ThisCommand==101) {
                $Cargos = $json_obj->Cargos;//массив с грузами
                $CargosCnt=count($Cargos); //кол-во элементов массива
                if ($CargosCnt<1) {
                        echo ("No Cargos founded");
                        exit();
                        }
                for ($i=0; $i<$CargosCnt; $i++){
                        $IdOnObject=$Cargos[$i]->IdOnObject;
                        $CargoName=$Cargos[$i]->CargoName;
                        $CargoPrice=$Cargos[$i]->CargoPrice;
                        $CargoCode=$Cargos[$i]->CargoCode;
                        $Activity=$Cargos[$i]->Activity;
                        //проверяем наличие локального ИД
                        $GlobalCargoId=CargosFindIdByLocalId($dbcnx, $idindb, $IdOnObject);
                        //если нет добавляем запись
                        if ($GlobalCargoId==0) {
                                echo("AddNewCargo:".$CargoName." ");//!!
                                CargosAddNew($dbcnx, $IdScale, $IdOnObject, $CargoName, $CargoPrice, $CargoCode,
$Activity);
                                }
                        //если есть обновляем остальные параметры
                        else {
                                echo("UpdateCargo:".$CargoName." ");//!!
                                CargosUpdateById($dbcnx,  $GlobalCargoId,  $CargoName,  $CargoPrice,  $CargoCode,
$Activity);
                                }
                        }
                echo("Cargo sync OK");
        }
//Синхронизируем таблицу водителей
//{“Command”:102,  “IdScale”:”XXXXX”,  “Drivers”:[{“IdOnObject”:XXX,  "DriverName":”xxxxx”,
“DriverFName”:”XXXX”, “DriverSName”:”XXXXXX”,
//“DriverPhone”:”XXXXXX”, “DriverDoc”:”XXXXXX”, “Activity”:X},{}…]}
        if ($ThisCommand==102) {
                $Drivers = $json_obj->Drivers;//массив с водителями
                $DriversCnt=count($Drivers); //кол-во элементов массива
                if ($DriversCnt<1) {
                        echo ("No Drivers founded");
                        exit();
                        }
                for ($i=0; $i<$DriversCnt; $i++){
                        $IdOnObject=$Drivers[$i]->IdOnObject;
                        $DriverName=$Drivers[$i]->DriverName;
                        $DriverFName=$Drivers[$i]->DriverFName;
                        $DriverSName=$Drivers[$i]->DriverSName;
                        $DriverPhone=$Drivers[$i]->DriverPhone;
                        $DriverDoc=$Drivers[$i]->DriverDoc;
                        $Activity=$Drivers[$i]->Activity;
                        //проверяем наличие локального ИД
                        $GlobalDriverId=DriversFindIdByLocalId($dbcnx, $idindb, $IdOnObject);
                        //если нет добавляем запись
                        if ($GlobalDriverId==0) {
                                DriversAddNew($dbcnx,  $IdScale,  $IdOnObject,  $DriverName,  $DriverFName,
$DriverSName, $DriverPhone, $DriverDoc, $Activity);
                                }
                        //если есть обновляем остальные параметры
```

```
                        else {
                                DriversUpdateById($dbcnx, $GlobalDriverId, $DriverName, $DriverFName,
$DriverSName, $DriverPhone, $DriverDoc, $Activity);
                                }
                        }
                echo("Drivers sync OK");
                exit();
        }
//Синхронизируем таблицу отправителей
//{“Command”:103, “IdScale”:”XXXXX”,  “Senders”:[{“IdOnObject”:XXX, "SenderName":”xxxxx”, “ERDPOU”:”XXXX”,
“AddressTXT”:”XXXXXX”, “Activity”:X},{}…]}
        if ($ThisCommand==103) {
                $Senders = $json_obj->Senders;//массив с отправителями
                $SendersCnt=count($Senders); //кол-во элементов массива
                if ($SendersCnt<1) {
                        echo ("No Senders founded");
                        exit();
                        }
                for ($i=0; $i<$SendersCnt; $i++){
                        $IdOnObject=$Senders[$i]->IdOnObject;
                        $SenderName=$Senders[$i]->SenderName;
                        $ERDPOU=$Senders[$i]->ERDPOU;
                        $AddressTXT=$Senders[$i]->AddressTXT;
                        $Activity=$Senders[$i]->Activity;
                        //проверяем наличие локального ИД
                        $GlobalSenderId=SendersFindIdByLocalId($dbcnx, $idindb, $IdOnObject);
                        //если нет добавляем запись
                        if ($GlobalSenderId==0) {
                                SendersAddNew($dbcnx, $IdScale, $IdOnObject, $SenderName, $ERDPOU,
$AddressTXT, $Activity);
                                }
                        //если есть обновляем остальные параметры
                        else {
                                SendersUpdateById($dbcnx, $GlobalSenderId, $SenderName, $ERDPOU, $AddressTXT,
$Activity);
                                }
                        }
                echo("Senders sync OK");
                exit();
        }
//Синхронизируем таблицу получателей
//{“Command”:104, “IdScale”:”XXXXX”,  “Recipients”:[{“IdOnObject”:XXX, "RcpName":”xxxxx”, “ERDPOU”:”XXXX”,
“AddressTXT”:”XXXXXX”, “Activity”:X},{}…]}
        if ($ThisCommand==104) {
                $Recipients = $json_obj->Recipients;//массив с получателями
                $RecipientsCnt=count($Recipients); //кол-во элементов массива
                if ($RecipientsCnt<1) {
                        echo ("No Recipients founded");
                        exit();
                        }
                for ($i=0; $i<$RecipientsCnt; $i++){
                        $IdOnObject=$Recipients[$i]->IdOnObject;
                        $RcpName=$Recipients[$i]->RcpName;
                        $ERDPOU=$Recipients[$i]->ERDPOU;
                        $AddressTXT=$Recipients[$i]->AddressTXT;
                        $Activity=$Recipients[$i]->Activity;
                        //проверяем наличие локального ИД
                        $GlobalRecipientId=RecipientsFindIdByLocalId($dbcnx, $idindb, $IdOnObject);
                        //если нет добавляем запись
                        if ($GlobalRecipientId==0) {
                                RecipientsAddNew($dbcnx, $IdScale, $IdOnObject, $RcpName, $ERDPOU,
$AddressTXT, $Activity);
                                }
```

```
                        //если есть обновляем остальные параметры
                        else {
                                RecipientsUpdateById($dbcnx, $GlobalRecipientId, $RcpName, $ERDPOU,
$AddressTXT, $Activity);
                                }
                        }
                echo("Recipients sync OK");
                exit();
        }
//Синхронизируем таблицу плательщиков
//{“Command”:105, “IdScale”:”XXXXX”, “Payers”:[{“IdOnObject”:XXX, "PayerName":”xxxxx”, “ERDPOU”:”XXXX”,
“AddressTXT”:”XXXXXX”, “Activity”:X},{}…]}
        if ($ThisCommand==105) {
                $Payers = $json_obj->Payers;//массив с плательщиками
                $PayersCnt=count($Payers); //кол-во элементов массива
                if ($PayersCnt<1) {
                        echo ("No Payers founded");
                        exit();
                        }
                for ($i=0; $i<$PayersCnt; $i++){
                        $IdOnObject=$Payers[$i]->IdOnObject;
                        $PayerName=$Payers[$i]->PayerName;
                        $ERDPOU=$Payers[$i]->ERDPOU;
                        $AddressTXT=$Payers[$i]->AddressTXT;
                        $Activity=$Payers[$i]->Activity;
                        //проверяем наличие локального ИД
                        $GlobalPayerId=PayersFindIdByLocalId($dbcnx, $idindb, $IdOnObject);
                        //если нет добавляем запись
                        if ($GlobalPayerId==0) {
                                PayersAddNew($dbcnx, $IdScale, $IdOnObject, $PayerName, $ERDPOU, $AddressTXT,
$Activity);
                                }
                        //если есть обновляем остальные параметры
                        else {
                                PayersUpdateById($dbcnx, $GlobalPayerId, $PayerName, $ERDPOU, $AddressTXT,
$Activity);
                                }
                        }
                echo("Payers sync OK");
                exit();
        }
//Синхронизируем таблицу перевозчиков
//{“Command”:106, “IdScale”:”XXXXX”, “Cariers”:[{“IdOnObject”:XXX, "CarierName”:”xxxxx”, “ERDPOU”:”XXXX”,
“AddressTXT”:”XXXXXX”, “Activity”:X},{}…]}
        if ($ThisCommand==106) {
                $Cariers = $json_obj->Cariers;//массив с перевозчиками
                $CariersCnt=count($Cariers); //кол-во элементов массива
                if ($CariersCnt<1) {
                        echo ("No Cariers founded");
                        exit();
                        }
                for ($i=0; $i<$CariersCnt; $i++){
                        $IdOnObject=$Cariers[$i]->IdOnObject;
                        $CarierName=$Cariers[$i]->CarierName;
                        $ERDPOU=$Cariers[$i]->ERDPOU;
                        $AddressTXT=$Cariers[$i]->AddressTXT;
                        $Activity=$Cariers[$i]->Activity;
                        //проверяем наличие локального ИД
                        $GlobalCarierId=CariersFindIdByLocalId($dbcnx, $idindb, $IdOnObject);
                        //если нет добавляем запись
                        if ($GlobalCarierId==0) {
                                CariersAddNew($dbcnx, $IdScale, $IdOnObject, $CarierName, $ERDPOU, $AddressTXT,
$Activity);
```

```
                                }
                        //если есть обновляем остальные параметры
                        else {
                                CariersUpdateById($dbcnx, $GlobalCarierId, $CarierName, $ERDPOU, $AddressTXT,
$Activity);
                                }
                        }
                echo("Cariers sync OK");
                exit();
        }
//Синхронизируем таблицу прицепов
//{“Command”:107, “IdScale”:”XXXXX”, “Trailers”:[{“IdOnObject”:XXX, "TrailerNumb":"xxxxx”, “Activity”:X},{}…]}
        if ($ThisCommand==107) {
                $Trailers = $json_obj->Trailers;//массив с прицепами
                $TrailersCnt=count($Trailers); //кол-во элементов массива
                if ($TrailersCnt<1) {
                        echo ("No Trailers founded");
                        exit();
                        }
                for ($i=0; $i<$TrailersCnt; $i++){
                        $IdOnObject=$Trailers[$i]->IdOnObject;
                        $TrailerNumb=$Trailers[$i]->TrailerNumb;
                        $Activity=$Trailers[$i]->Activity;
                        //проверяем наличие локального ИД
                        $GlobalTrailerId=TrailersFindIdByLocalId($dbcnx, $idindb, $IdOnObject);
                        //если нет добавляем запись
                        if ($GlobalTrailerId==0) {
                                TrailersAddNew($dbcnx, $IdScale, $IdOnObject, $TrailerNumb, $Activity);
                                }
                        //если есть обновляем остальные параметры
                        else {
                                TrailersUpdateById($dbcnx, $GlobalTrailerId, $TrailerNumb, $Activity);
                                }
                        }
                echo("Trailers sync OK");
                exit();
        }
//Синхронизируем таблицу машин
//{“Command”:108, “IdScale”:”XXXXX”, “Cars”:[{“IdOnObject”:XXX, "CarNumb":”xxxxx”, “CarMark”:”XXXX”,
“CarColor”:”XXXXXX”,
//“DriverId”:X, “TrailerId”:X, “Activity”:X},{}…]}
        if ($ThisCommand==108) {
                $Cars = $json_obj->Cars;//массив с машинами
                $CarsCnt=count($Cars); //кол-во элементов массива
                if ($CarsCnt<1) {
                        echo ("No Cars founded");
                        exit();
                        }
                for ($i=0; $i<$CarsCnt; $i++){
                        $IdOnObject=$Cars[$i]->IdOnObject;
                        $CarNumb=$Cars[$i]->CarNumb;
                        $CarMark=$Cars[$i]->CarMark;
                        $CarColor=$Cars[$i]->CarColor;
                        $DriverId=$Cars[$i]->DriverId;
                        $TrailerId=$Cars[$i]->TrailerId;
                        $Activity=$Cars[$i]->Activity;
                        //проверяем наличие локального ИД
                        $GlobalCarId=CarsFindIdByLocalId($dbcnx, $idindb, $IdOnObject);
                        //если нет добавляем запись
                        if ($GlobalCarId==0) {
                                CarsAddNew($dbcnx, $IdScale, $IdOnObject, $CarNumb, $CarMark, $CarColor,
$DriverId, $TrailerId, $Activity);
                                }
```

```
                        //если есть обновляем остальные параметры
                        else {
                                CarsUpdateById($dbcnx, $idindb, $GlobalCarId, $CarNumb, $CarMark, $CarColor,
$DriverId, $TrailerId, $Activity);
                                }
                        }
                echo("Cars sync OK");
                exit();
        }
//Синхронизируем таблицу камер
//{“Command”:109, “IdScale”:”XXXXX”, “Cameras”:[{“IdOnObject”:XXX, “IPaddr”:”xxxxx”, “GetQuary”:”XXXX”,
“UserName”:”XXXXXX”, “AccessPsw”:”XXXXX”, “Mode”:X },{}…]}
        if ($ThisCommand==109) {
                $Cameras = $json_obj->Cameras;//массив с камерами
                $CamerasCnt=count($Cameras); //кол-во элементов массива
                if ($CamerasCnt<1) {
                        echo ("No Cameras founded");
                        exit();
                        }
                for ($i=0; $i<$CarsCnt; $i++){
                        $IdOnObject=$Cameras[$i]->IdOnObject;
                        $IPaddr=$Cameras[$i]->IPaddr;
                        $GetQuary=$Cameras[$i]->GetQuary;
                        $UserName=$Cameras[$i]->UserName;
                        $AccessPsw=$Cameras[$i]->AccessPsw;
                        $Mode=$Cameras[$i]->Mode;
                        //проверяем наличие локального ИД
                        $GlobalCameraId=CamerasFindIdByLocalId($dbcnx, $idindb, $IdOnObject);
                        //если нет добавляем запись
                        if ($GlobalCarId==0) {
                                CamerasAddNew($dbcnx, $IdScale, $IdOnObject, $IPaddr, $GetQuary, $UserName,
$AccessPsw, $Mode);
                                }
                        //если есть обновляем остальные параметры
                        else {
                                CamerasUpdateById($dbcnx, $idindb, $GlobalCameraId, $IPaddr, $GetQuary,
$UserName, $AccessPsw, $Mode);
                                }
                        }
                echo("Cameras sync OK");
                exit();
        }
//Синхронизируем таблицу взвешиваний
//{“Command”:110, “IdScale”:”XXXXX”, “Wghts”:[{“IdOnObject”:XXX, “DateIn”:”xxxxx”, “TimeIn”:”XXXX”,
“DateOut”:”XXXXXX”, “TimeOut”:”XXXXX”,
//“OpType”:X, “Brutto”:X, “Netto”:X, “Tara”:X, “CorrectBrutto”:X, “CorrectTara”:X, “WCargo”:X, “WCar”:X,
“WDriver”:X, “WSender”:X, “WRecp”:X,
//“WPayer”:X, “WCarier”:X, “WTrl”:X, “WDoc”:”XXX”, “Activity”:X},{}…]}
        if ($ThisCommand==110) {
                $Wghts = $json_obj->Wghts;//массив с взвешиваниями
                $WghtsCnt=count($Wghts); //кол-во элементов массива
                if ($WghtsCnt<1) {
                        echo ("No Wghts founded");
                        exit();
                        }
                for ($i=0; $i<$WghtsCnt; $i++){
                        $IdOnObject=$Wghts[$i]->IdOnObject;
                        $DateIn=$Wghts[$i]->DateIn;
                        $TimeIn=$Wghts[$i]->TimeIn;
                        $DateOut=$Wghts[$i]->DateOut;
                        $TimeOut=$Wghts[$i]->TimeOut;
                        $OpType=$Wghts[$i]->OpType;
                        $Brutto=$Wghts[$i]->Brutto;
```

```
                        $Netto=$Wghts[$i]->Netto;
                        $Tara=$Wghts[$i]->Tara;
                        $CorrectBrutto=$Wghts[$i]->CorrectBrutto;
                        $CorrectTara=$Wghts[$i]->CorrectTara;
                        $WCargo=$Wghts[$i]->WCargo;
                        $WCar=$Wghts[$i]->WCar;
                        $WDriver=$Wghts[$i]->WDriver;
                        $WSender=$Wghts[$i]->WSender;
                        $WRecp=$Wghts[$i]->WRecp;
                        $WPayer=$Wghts[$i]->WPayer;
                        $WCarier=$Wghts[$i]->WCarier;
                        $WTrl=$Wghts[$i]->WTrl;
                        $WDoc=$Wghts[$i]->WDoc;
                        $Activity=$Wghts[$i]->Activity;
                        //проверяем наличие локального ИД
                        $GlobalWghtId=WghtsFindIdByLocalId($dbcnx, $idindb, $IdOnObject);
                        //если нет добавляем запись
                        if ($GlobalWghtId==0) {
                                WghtsAddNew($dbcnx, $IdScale, $IdOnObject,
$DateIn,$TimeIn,$DateOut,$TimeOut,$OpType,$Brutto,$Netto,$Tara,$CorrectBrutto,

        $CorrectTara,$WCargo,$WCar,$WDriver,$WSender,$WRecp,$WPayer,$WCarier,$WTrl,$WDoc,$Activity);
                                }
                        //если есть обновляем остальные параметры
                        else {
                                WghtsUpdateById($dbcnx, $idindb, $GlobalWghtId,
$DateIn,$TimeIn,$DateOut,$TimeOut,$OpType,$Brutto,$Netto,$Tara,$CorrectBrutto,

        $CorrectTara,$WCargo,$WCar,$WDriver,$WSender,$WRecp,$WPayer,$WCarier,$WTrl,$WDoc,$Activity);
                                }
                        }
                echo("Wghts sync OK");
                exit();
        }
//Синхронизируем таблицу соответствия взвешиваний фотографиям
//{“Command”:111, “IdScale”:”XXXXX”, “PhotoWght”:[{“IdOnObject”:XXX, “WghtId”: x, “PhotoFileName”:”XXXX”,
“CamNumb”:X },{}…]}

//Создание нового(ых) груза(ов)
//{“Command”:201, “IdScale”:”XXXXX”, “Cargos”:[{“IdOnObject”:XXX, “CargoName”:”xxxxx”, “CargoPrice”:XX.XX,
“CargoCode”:”XXXXXX”, “Activity”:X},{}…]}
        if ($ThisCommand==201) {
                $Cargos = $json_obj->Cargos;//массив с грузами
                $CargosCnt=count($Cargos); //кол-во элементов массива
                if ($CargosCnt<1) {
                        echo ("No Cargos founded");
                        exit();
                        }
                for ($i=0; $i<$CargosCnt; $i++){
                        $IdOnObject=$Cargos[$i]->IdOnObject;
                        $CargoName=$Cargos[$i]->CargoName;
                        $CargoPrice=$Cargos[$i]->CargoPrice;
                        $CargoCode=$Cargos[$i]->CargoCode;
                        $Activity=$Cargos[$i]->Activity;
                        CargosAddNew($dbcnx, $IdScale, $IdOnObject, $CargoName, $CargoPrice, $CargoCode,
$Activity);
                        }
                echo("Cargo create OK");
                exit();
        }
//Создание нового(ых) водителя(ей)
//{“Command”:202, “IdScale”:”XXXXX”, “Drivers”:[{“IdOnObject”:XXX, "DriverName":”xxxxx”,
“DriverFName”:”XXXX”, “DriverSName”:”XXXXXX”,
```

```
//“DriverPhone”:”XXXXXX”, “DriverDoc”:”XXXXXX”, “Activity”:X},{}…]}
	if ($ThisCommand==202) {
		$Drivers = $json_obj->Drivers;//массив с водителями
		$DriversCnt=count($Drivers); //кол-во элементов массива
		if ($DriversCnt<1) {
			echo ("No Drivers founded");
			exit();
			}
		for ($i=0; $i<$DriversCnt; $i++){
			$IdOnObject=$Drivers[$i]->IdOnObject;
			$DriverName=$Drivers[$i]->DriverName;
			$DriverFName=$Drivers[$i]->DriverFName;
			$DriverSName=$Drivers[$i]->DriverSName;
			$DriverPhone=$Drivers[$i]->DriverPhone;
			$DriverDoc=$Drivers[$i]->DriverDoc;
			$Activity=$Drivers[$i]->Activity;
			DriversAddNew($dbcnx, $IdScale, $IdOnObject, $DriverName, $DriverFName, $DriverSName,
$DriverPhone, $DriverDoc, $Activity);
			}
		echo("Driver create OK");
		exit();
	}
//Создание нового(ых) отправителя(ей)
//{“Command”:203, “IdScale”:”XXXXX”, “Senders”:[{“IdOnObject”:XXX, "SenderName":”xxxxx”, “ERDPOU”:”XXXX”,
“AddressTXT”:”XXXXXX”, “Activity”:X},{}…]}
	if ($ThisCommand==203) {
		$Senders = $json_obj->Senders;//массив с отправителями
		$SendersCnt=count($Senders); //кол-во элементов массива
		if ($SendersCnt<1) {
			echo ("No Senders founded");
			exit();
			}
		for ($i=0; $i<$SendersCnt; $i++){
			$IdOnObject=$Senders[$i]->IdOnObject;
			$SenderName=$Senders[$i]->SenderName;
			$ERDPOU=$Senders[$i]->ERDPOU;
			$AddressTXT=$Senders[$i]->AddressTXT;
			$Activity=$Senders[$i]->Activity;
			SendersAddNew($dbcnx, $IdScale, $IdOnObject, $SenderName, $ERDPOU, $AddressTXT,
$Activity);
			}
		echo("Senders create OK");
		exit();
	}
//Создание нового(ых) получателя(ей)
//{“Command”:204, “IdScale”:”XXXXX”, “Recipients”:[{“IdOnObject”:XXX, "RcpName":”xxxxx”, “ERDPOU”:”XXXX”,
“AddressTXT”:”XXXXXX”, “Activity”:X},{}…]}
	if ($ThisCommand==204) {
		$Recipients = $json_obj->Recipients;//массив с получателями
		$RecipientsCnt=count($Recipients); //кол-во элементов массива
		if ($RecipientsCnt<1) {
			echo ("No Recipients founded");
			exit();
			}
		for ($i=0; $i<$RecipientsCnt; $i++){
			$IdOnObject=$Recipients[$i]->IdOnObject;
			$RcpName=$Recipients[$i]->RcpName;
			$ERDPOU=$Recipients[$i]->ERDPOU;
			$AddressTXT=$Recipients[$i]->AddressTXT;
			$Activity=$Recipients[$i]->Activity;
			RecipientsAddNew($dbcnx, $IdScale, $IdOnObject, $RcpName, $ERDPOU, $AddressTXT,
$Activity);
			}
```

```
                echo("Recipients create OK");
                exit();
        }
//Создание нового(ых) плательщика(ов)
//{“Command”:205, “IdScale”:”XXXXX”, “Payers”:[{“IdOnObject”:XXX, "PayerName":”xxxxx”, “ERDPOU”:”XXXX”,
“AddressTXT”:”XXXXXX”, “Activity”:X},{}…]}
        if ($ThisCommand==205) {
                $Payers = $json_obj->Payers;//массив с плательщиками
                $PayersCnt=count($Payers); //кол-во элементов массива
                if ($PayersCnt<1) {
                        echo ("No Payers founded");
                        exit();
                        }
                for ($i=0; $i<$PayersCnt; $i++){
                        $IdOnObject=$Payers[$i]->IdOnObject;
                        $PayerName=$Payers[$i]->PayerName;
                        $ERDPOU=$Payers[$i]->ERDPOU;
                        $AddressTXT=$Payers[$i]->AddressTXT;
                        $Activity=$Payers[$i]->Activity;
                        PayersAddNew($dbcnx, $IdScale, $IdOnObject, $PayerName, $ERDPOU, $AddressTXT,
$Activity);
                        }
                echo("Payers create OK");
                exit();
        }
//Создание нового(ых) перевозчика(ов)
//{“Command”:206, “IdScale”:”XXXXX”, “Cariers”:[{“IdOnObject”:XXX, "CarierName”:”xxxxx”, “ERDPOU”:”XXXX”,
“AddressTXT”:”XXXXXX”, “Activity”:X},{}…]}
        if ($ThisCommand==206) {
                $Cariers = $json_obj->Cariers;//массив с перевозчиками
                $CariersCnt=count($Cariers); //кол-во элементов массива
                if ($CariersCnt<1) {
                        echo ("No Cariers founded");
                        exit();
                        }
                for ($i=0; $i<$CariersCnt; $i++){
                        $IdOnObject=$Cariers[$i]->IdOnObject;
                        $CarierName=$Cariers[$i]->CarierName;
                        $ERDPOU=$Cariers[$i]->ERDPOU;
                        $AddressTXT=$Cariers[$i]->AddressTXT;
                        $Activity=$Cariers[$i]->Activity;
                        CariersAddNew($dbcnx, $IdScale, $IdOnObject, $CarierName, $ERDPOU, $AddressTXT,
$Activity);
                        }
                echo("Cariers create OK");
                exit();
        }
//Создание нового(ых) прицепа(ов)
//{“Command”:207, “IdScale”:”XXXXX”, “Trailers”:[{“IdOnObject”:XXX, "TrailerNumb":"xxxxx”, “Activity”:X},{}…]}
        if ($ThisCommand==207) {
                $Trailers = $json_obj->Trailers;//массив с прицепами
                $TrailersCnt=count($Trailers); //кол-во элементов массива
                if ($TrailersCnt<1) {
                        echo ("No Trailers founded");
                        exit();
                        }
                for ($i=0; $i<$TrailersCnt; $i++){
                        $IdOnObject=$Trailers[$i]->IdOnObject;
                        $TrailerNumb=$Trailers[$i]->TrailerNumb;
                        $Activity=$Trailers[$i]->Activity;
                        TrailersAddNew($dbcnx, $IdScale, $IdOnObject, $TrailerNumb, $Activity);
                        }
                echo("Trailers create OK");
```

```
                exit();
        }
//Создание новой(ых) машин(ы)
//{“Command”:208, “IdScale”:”XXXXX”, “Cars”:[{“IdOnObject”:XXX, "CarNumb":”xxxxx”, “CarMark”:”XXXX”, “CarColor”:”XXXXXX”,
//“DriverId”:X, “TrailerId”:X, “Activity”:X},{}…]}
        if ($ThisCommand==208) {
                $Cars = $json_obj->Cars;//массив с машинами
                $CarsCnt=count($Cars); //кол-во элементов массива
                if ($CarsCnt<1) {
                        echo ("No Cars founded");
                        exit();
                        }
                for ($i=0; $i<$CarsCnt; $i++){
                        $IdOnObject=$Cars[$i]->IdOnObject;
                        $CarNumb=$Cars[$i]->CarNumb;
                        $CarMark=$Cars[$i]->CarMark;
                        $CarColor=$Cars[$i]->CarColor;
                        $DriverId=$Cars[$i]->DriverId;
                        $TrailerId=$Cars[$i]->TrailerId;
                        $Activity=$Cars[$i]->Activity;
                        CarsAddNew($dbcnx, $IdScale, $IdOnObject, $CarNumb, $CarMark, $CarColor, $DriverId, $TrailerId, $Activity);
                        }
                echo("Cars create OK");
                exit();
        }
//Создание новой(ых) камер(ы)
//{“Command”:209, “IdScale”:”XXXXX”, “Cameras”:[{“IdOnObject”:XXX, “IPaddr”:”xxxxx”, “GetQuary”:”XXXX”, “UserName”:”XXXXXX”, “AccessPsw”:”XXXXX”, “Mode”:X },{}…]}
        if ($ThisCommand==209) {
                $Cameras = $json_obj->Cameras;//массив с камерами
                $CamerasCnt=count($Cameras); //кол-во элементов массива
                if ($CamerasCnt<1) {
                        echo ("No Cameras founded");
                        exit();
                        }
                for ($i=0; $i<$CarsCnt; $i++){
                        $IdOnObject=$Cameras[$i]->IdOnObject;
                        $IPaddr=$Cameras[$i]->IPaddr;
                        $GetQuary=$Cameras[$i]->GetQuary;
                        $UserName=$Cameras[$i]->UserName;
                        $AccessPsw=$Cameras[$i]->AccessPsw;
                        $Mode=$Cameras[$i]->Mode;
                        CamerasAddNew($dbcnx, $IdScale, $IdOnObject, $IPaddr, $GetQuary, $UserName, $AccessPsw, $Mode);
                        }
                echo("Cameras create OK");
                exit();
        }
//Создание нового(ых) взвешивания(й)
//{“Command”:210, “IdScale”:”XXXXX”, “Wghts”:[{“IdOnObject”:XXX, “DateIn”:”xxxxx”, “TimeIn”:”XXXX”, “DateOut”:”XXXXXX”, “TimeOut”:”XXXXX”,
//“OpType”:X, “Brutto”:X, “Netto”:X, “Tara”:X, “CorrectBrutto”:X, “CorrectTara”:X, “WCargo”:X, “WCar”:X, “WDriver”:X, “WSender”:X, “WRecp”:X,
//“WPayer”:X, “WCarier”:X, “WTrl”:X, “WDoc”:”XXX”, “Activity”:X},{}…]}
        if ($ThisCommand==210) {
                $Wghts = $json_obj->Wghts;//массив с взвешиваниями
                $WghtsCnt=count($Wghts); //кол-во элементов массива
                if ($WghtsCnt<1) {
                        echo ("No Wghts founded");
                        exit();
                        }
```

```
		for ($i=0; $i<$WghtsCnt; $i++){
			$IdOnObject=$Wghts[$i]->IdOnObject;
			$DateIn=$Wghts[$i]->DateIn;
			$TimeIn=$Wghts[$i]->TimeIn;
			$DateOut=$Wghts[$i]->DateOut;
			$TimeOut=$Wghts[$i]->TimeOut;
			$OpType=$Wghts[$i]->OpType;
			$Brutto=$Wghts[$i]->Brutto;
			$Netto=$Wghts[$i]->Netto;
			$Tara=$Wghts[$i]->Tara;
			$CorrectBrutto=$Wghts[$i]->CorrectBrutto;
			$CorrectTara=$Wghts[$i]->CorrectTara;
			$WCargo=$Wghts[$i]->WCargo;
			$WCar=$Wghts[$i]->WCar;
			$WDriver=$Wghts[$i]->WDriver;
			$WSender=$Wghts[$i]->WSender;
			$WRecp=$Wghts[$i]->WRecp;
			$WPayer=$Wghts[$i]->WPayer;
			$WCarier=$Wghts[$i]->WCarier;
			$WTrl=$Wghts[$i]->WTrl;
			$WDoc=$Wghts[$i]->WDoc;
			$Activity=$Wghts[$i]->Activity;
			WghtsAddNew($dbcnx, $IdScale, $IdOnObject,
$DateIn,$TimeIn,$DateOut,$TimeOut,$OpType,$Brutto,$Netto,$Tara,$CorrectBrutto,

	$CorrectTara,$WCargo,$WCar,$WDriver,$WSender,$WRecp,$WPayer,$WCarier,$WTrl,$WDoc,$Activity);

			}
		echo("Wghts create OK");
		exit();
	}

//Создание или изменение файла Crt.dat для данных весов
//{“Command”:212, “IdScale”:”XXXXX”, "ScaleKey":"XXXXXXXXXXXXXXXX", "MinCorrectWght":300,
"NeedFixAfterButton":1, "MaxCorrect":200}
	if ($ThisCommand==212) {
		//проверяем есть ли папка с указанным ИД в папке Sets
		if (!file_exists('./Sets/'.$IdScale)) { //если нет создаем
			if (mkdir('./Sets/'.$IdScale, 0777, true)) {
				saveLog('Folder '.$IdScale.' Created');
			} else {saveLog('Error of Folder '.$IdScale.' Creating');
				}
		}
		//получаем параметры из запроса
		$MinCorrectWght = $json_obj->MinCorrectWght;
		$NeedFixAfterButton = $json_obj->NeedFixAfterButton;
		$MaxCorrect = $json_obj->MaxCorrect;
		//Создаем идентичную локальной структуру
		$MinCorrectWght = "MinCorrectWght=".$MinCorrectWght."\r\n";
		$NeedFixAfterButton = "NeedFixAfterButton=".$NeedFixAfterButton."\r\n";
		$MaxCorrect = "MaxCorrect=".$MaxCorrect."\r\n";
		//сохраняем параметры в файл
		$file = fopen('./Sets/'.$IdScale.'/Crt.dat', 'w');
		fwrite($file, $MinCorrectWght.$NeedFixAfterButton.$MaxCorrect);
		fclose($file);
		echo("Crt.dat sync OK");
		exit();
	}

//Создание или изменение файла DataSend.dat для данных весов
//{"Command":213, "IdScale":"A1201", "ScaleKey":"20ADF02FAA061712", "SendBrutto":0, "SendNetto":0, "SendTara":0,
"SendCarNumb":0,
```

```
//"SendCargo":0, "SendDriver":0, "SendContragent":0, "SendStock":0, "SendTrailer":0, "SendTrash":0, "SendClearWght":0, "SendPhotos":0,
//"NeedTelegram":0, "TelegramToken":"1178159188:AAHojsviOPqPcYk-ChQ5837OkcbJB0GnzYs", "TelUserId1":nill, "TelUserId2":nill, "TelUserId3":nill,
//"TelUserId4":nill, "NeedTelUserSend1":0, "NeedTelUserSend2":0, "NeedTelUserSend3":0, "NeedTelUserSend4":0, "NeedTelReport":0,
//"TelReportTime":"20:00", "UserIdToSend":nill, "NeedWeekTelReport":0, "TelWeekReportDay":5, "UserIdWeekReport":nill,
//"ServerURL":"http://avrobit.org/", "NeedWghtSend":1, "NeedAllWghtSend":0, "ScalesServerKey":"20ADF02FAA061712"}

	if ($ThisCommand==213) {
		//проверяем есть ли папка с указанным ИД в папке Sets
		if (!file_exists('./Sets/'.$IdScale)) { //если нет создаем
			if (mkdir('./Sets/'.$IdScale, 0777, true)) {
				saveLog('Folder '.$IdScale.' Created');
			} else {saveLog('Error of Folder '.$IdScale.' Creating');
				}
		}
		//получаем параметры из запроса
		//$MinCorrectWght = $json_obj->MinCorrectWght;
		//$WasFixed = $json_obj->WasFixed;
		//$MaxCorrect = $json_obj->MaxCorrect;
		//Создаем идентичную локальной структуру
		//$MinCorrectWght = "MinCorrectWght=".$MinCorrectWght."\r\n";
		//$WasFixed = "WasFixed=".$WasFixed."\r\n";
		//$MaxCorrect = "MaxCorrect=".$MaxCorrect."\r\n";
		//сохраняем параметры в файл
		//$file = fopen('./Sets/'.$IdScale./Crt.dat', 'w');
		//fwrite($file, $MinCorrectWght.$WasFixed.$MaxCorrect);
		//fclose($file);
		echo("DataSend.dat sync OK");
		exit();
	}


//обновление взвешиваний
//{“Command”:310, “IdScale”:”XXXXX”, “Wghts”:[{“IdOnObject”:XXX, “DateIn”:”xxxxx”, “TimeIn”:”XXXX”, “DateOut”:”XXXXXX”, “TimeOut”:”XXXXX”,
//“OpType”:X, “Brutto”:X, “Netto”:X, “Tara”:X, “CorrectBrutto”:X, “CorrectTara”:X, “WCargo”:X, “WCar”:X, “WDriver”:X, “WSender”:X, “WRecp”:X,
//“WPayer”:X, “WCarier”:X, “WTrl”:X, “WDoc”:”XXX”, “Activity”:X},{}…]}
	if ($ThisCommand==310) {
		$Wghts = $json_obj->Wghts;//массив с взвешиваниями
		$WghtsCnt=count($Wghts); //кол-во элементов массива
		if ($WghtsCnt<1) {
			echo ("No Wghts founded");
			exit();
			}
		for ($i=0; $i<$WghtsCnt; $i++){
			$IdOnObject=$Wghts[$i]->IdOnObject;
			$DateIn=$Wghts[$i]->DateIn;
			$TimeIn=$Wghts[$i]->TimeIn;
			$DateOut=$Wghts[$i]->DateOut;
			$TimeOut=$Wghts[$i]->TimeOut;
			$OpType=$Wghts[$i]->OpType;
			$Brutto=$Wghts[$i]->Brutto;
			$Netto=$Wghts[$i]->Netto;
			$Tara=$Wghts[$i]->Tara;
			$CorrectBrutto=$Wghts[$i]->CorrectBrutto;
			$CorrectTara=$Wghts[$i]->CorrectTara;
			$WCargo=$Wghts[$i]->WCargo;
			$WCar=$Wghts[$i]->WCar;
			$WDriver=$Wghts[$i]->WDriver;
```

```
                $WSender=$Wghts[$i]->WSender;
                $WRecp=$Wghts[$i]->WRecp;
                $WPayer=$Wghts[$i]->WPayer;
                $WCarier=$Wghts[$i]->WCarier;
                $WTrl=$Wghts[$i]->WTrl;
                $WDoc=$Wghts[$i]->WDoc;
                $Activity=$Wghts[$i]->Activity;
                $GlobalWghtId=WghtsFindIdByLocalId($dbcnx, $idindb, $IdOnObject);
                WghtsUpdateById($dbcnx, $idindb, $GlobalWghtId,
$DateIn,$TimeIn,$DateOut,$TimeOut,$OpType,$Brutto,$Netto,$Tara,$CorrectBrutto,

        $CorrectTara,$WCargo,$WCar,$WDriver,$WSender,$WRecp,$WPayer,$WCarier,$WTrl,$WDoc,$Activity);
                }
            echo("Wghts update OK");
            exit();
        }


//Получить код изменений произведенных сервером для текущего объекта
//{“Command”:400, “IdScale”:”XXXXX”, "ScaleKey":"XXXXXXXXXXXXXXXX"}
        if ($ThisCommand==400) {
            //проверяем есть ли папка с указанным ИД в папке Sets
            if (!file_exists('./Sets/'.$IdScale)) { //если нет создаем
                if (mkdir('./Sets/'.$IdScale, 0777, true)) {
                    saveLog('Folder '.$IdScale.' Created');
                } else {saveLog('Error of Folder '.$IdScale.' Creating');
                    }
            }
            //проверяем существует ли файл с необходимыми изменениями в этой папке
            if (!file_exists('./Sets/'.$IdScale.'/Changes.dat')) {
                //если нет - создаем с нулями
                $chngs='00000000000000000';
                $file = fopen('./Sets/'.$IdScale.'/Changes.dat', 'w');
                fwrite($file, $chngs);
                fclose($file);
                echo($chngs);
                exit();
            }
            $file = fopen('./Sets/'.$IdScale.'/Changes.dat', 'r');
            $chngs = fgets($file);
            fclose($file);
            echo($chngs);
            exit();
        }

//Исправить код изменений после выполнения действий команды 400
//{“Command”:499, “IdScale”:”XXXXX”, "ScaleKey":"XXXXXXXXXXXXXXXX",
"ChangeCode":"00000000000000000"}
        if ($ThisCommand==499) {
            //проверяем есть ли папка с указанным ИД в папке Sets
            if (!file_exists('./Sets/'.$IdScale)) { //если нет создаем
                if (mkdir('./Sets/'.$IdScale, 0777, true)) {
                    saveLog('Folder '.$IdScale.' Created');
                } else {saveLog('Error of Folder '.$IdScale.' Creating');
                    }
            }
            $chngs=$json_obj->ChangeCode;
            $file = fopen('./Sets/'.$IdScale.'/Changes.dat', 'w');
            fwrite($file, $chngs);
            fclose($file);
            echo('Changes Saved');
            exit();
        }
```

```
	echo ("Uncorrect Command"); //если не одна команда не совпала
	exit();
//*****************************************************************************************************************
****************************************

//-----------------------------------------------------------------------------------------------------------------------------------------
//обновляем машину по ИД
function WghtsUpdateById($mysqli, $idindb, $GlobalWghtId,
$DateIn,$TimeIn,$DateOut,$TimeOut,$OpType,$Brutto,$Netto,$Tara,$CorrectBrutto,

	$CorrectTara,$WCargo,$WCar,$WDriver,$WSender,$WRecp,$WPayer,$WCarier,$WTrl,$WDoc,$Activity) {

	$DateIn=ConvertDateForDB($DateIn);
	if ($DateOut=="") {$DateOut="NULL"; } else {$DateOut="'".ConvertDateForDB($DateOut)."'";}
	if ($TimeOut=="") {$TimeOut="NULL"; } else {$TimeOut="'".$TimeOut."'";}


	$query="UPDATE Weighing SET DateIn='".$DateIn."', ";
	$query=$query."TimeIn='".$TimeIn."', ";
	if ($DateOut!="") {$query=$query."DateOut='".$DateOut."', ";} else {$query=$query."DateOut=NULL, ";}
	if ($TimeOut!="") {$query=$query."TimeOut='".$TimeOut."', ";} else {$query=$query."TimeOut=NULL, ";}
	$query=$query."OpType=".$OpType.", ";
	$query=$query."Brutto=".$Brutto.", ";
	$query=$query."Netto=".$Netto.", ";
	$query=$query."Tara=".$Tara.", ";
	$query=$query."CorrectBrutto=".$CorrectBrutto.", ";
	$query=$query."CorrectTara=".$CorrectTara.", ";
	if ($WCargo>0) {
		$GCrgId=CargosFindIdByLocalId($mysqli, $idindb, $WCargo);
		$query=$query."WCargo=".$GCrgId.", ";
		}
		else {$query=$query."WCargo=NULL, ";}
	if ($WCar>0) {
		$GCarId=CarsFindIdByLocalId($mysqli, $idindb, $WCar);
		$query=$query."WCar=".$GCarId.", ";
		}
		else {$query=$query."WCar=NULL, ";}
	if ($WDriver>0) {
		$GDrvId=DriversFindIdByLocalId($mysqli, $idindb, $WDriver);
		$query=$query."WDriver=".$GDrvId.", ";
		}
		else {$query=$query."WDriver=NULL, ";}
	if ($WSender>0) {
		$GSndId=SendersFindIdByLocalId($mysqli, $idindb, $WSender);
		$query=$query."WSender=".$GSndId.", ";
		}
		else {$query=$query."WSender=NULL, ";}
	if ($WRecp>0) {
		$GRcpId=RecipientsFindIdByLocalId($mysqli, $idindb, $WRecp);
		$query=$query."WRecp=".$GRcpId.", ";
		}
		else {$query=$query."WRecp=NULL, ";}
	if ($WPayer>0) {
		$GPayId=PayersFindIdByLocalId($mysqli, $idindb, $WPayer);
		$query=$query."WPayer=".$GPayId.", ";
		}
		else {$query=$query."WPayer=NULL, ";}
	if ($WCarier>0) {
		$GCrrId=CariersFindIdByLocalId($mysqli, $idindb, $WCarier);
		$query=$query."WCarier=".$GCrrId.", ";
		}
		else {$query=$query."WCarier=NULL, ";}
```

```
	if ($WTrl>0) {
		$GTrlId=TrailersFindIdByLocalId($mysqli, $idindb, $WTrl);
		$query=$query."WTrl=".$GTrlId.", ";
		}
		else {$query=$query."WTrl=NULL, ";}
	if ($WDoc!="") {$query=$query."WDoc='".$WDoc."', ";} else {$query=$query."WDoc=NULL, ";}
	$query=$query."Activity=".$Activity;
	$query=$query." WHERE Id_Wght=".$GlobalWghtId.";";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//находим максимальный ИД машины в БД
function WghtsGetMaxId($mysqli){
	$result=0;
	$query="SELECT MAX(Id_Wght) as _maxID FROM Weighing;";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['_maxID'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//конвертируем дату для сохранения в БД
function ConvertDateForDB($DateIn){
	$result=substr($DateIn, -4)."-".substr($DateIn, 3,-5)."-".substr($DateIn,0,-8);
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//добавляем новую машину в глобальную БД
function WghtsAddNew($mysqli, $IdScale, $IdOnObject,
$DateIn,$TimeIn,$DateOut,$TimeOut,$OpType,$Brutto,$Netto,$Tara,$CorrectBrutto,

	$CorrectTara,$WCargo,$WCar,$WDriver,$WSender,$WRecp,$WPayer,$WCarier,$WTrl,$WDoc,$Activity) {
	$newid=WghtsGetMaxId($mysqli)+1;
	$scid=ScalesIDInDB($mysqli, $IdScale);
	$DateIn=ConvertDateForDB($DateIn);
	if ($DateOut=="") {$DateOut="NULL"; } else {$DateOut="'".ConvertDateForDB($DateOut)."'";}
	if ($TimeOut=="") {$TimeOut="NULL"; } else {$TimeOut="'".$TimeOut."'";}
	if ($DriverId>0) {$GDId=DriversFindIdByLocalId($mysqli, $scid, $DriverId);}
		else {$GDId="NULL";}
	if ($WCargo>0) {$GCrgId=CargosFindIdByLocalId($mysqli, $scid, $WCargo);}
		else {$GCrgId="NULL";}
	if ($WCar>0) {$GCarId=CarsFindIdByLocalId($mysqli, $scid, $WCar);}
		else {$GCarId="NULL";}
	if ($WDriver>0) {$GDrvId=DriversFindIdByLocalId($mysqli, $scid, $WDriver);}
		else {$GDrvId="NULL";}
	if ($WSender>0) {$GSndId=SendersFindIdByLocalId($mysqli, $scid, $WSender);}
		else {$GSndId="NULL";}
	if ($WRecp>0) {$GRcpId=RecipientsFindIdByLocalId($mysqli, $scid, $WRecp);}
		else {$GRcpId="NULL";}
	if ($WPayer>0) {$GPayId=PayersFindIdByLocalId($mysqli, $scid, $WPayer);}
		else {$GPayId="NULL";}
	if ($WCarier>0) {$GCrrId=CariersFindIdByLocalId($mysqli, $scid, $WCarier);}
		else {$GCrrId="NULL";}
	if ($WTrl>0) {$GTrlId=TrailersFindIdByLocalId($mysqli, $scid, $WTrl);}
		else {$GTrlId="NULL";}
	if ($WDoc=="") {$WDoc="NULL"; } else {$WDoc="'".$WDoc."'";}
	$query="INSERT INTO Weighing VALUES(".$newid.", ".$scid.", '".$DateIn."', '".$TimeIn."', ".$DateOut.",
".$TimeOut.", ".$OpType.", ".$Brutto.", ";
	$query=$query.$Netto.", ".$Tara.", ".$CorrectBrutto.", ".$CorrectTara.", ".$GCrgId.", ".$GCarId.", ".$GDrvId.",
".$GSndId.", ".$GRcpId.", ".$GPayId.", ";
```

```
	$query=$query.$GCrrId.", ".$GTrlId.", NULL, ".$WDoc.", ".$IdOnObject.", ".$Activity.", NULL);";

	echo(" ".$query." ");//!!!!
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------
//Находим глобальный индекс машины по локальному ИД и ИД весов - или 0 если такого нет
function WghtsFindIdByLocalId($mysqli, $scid, $localid) {
	$result=0;
	$query="SELECT Id_Wght FROM Weighing WHERE IdScales=".$scid." AND IdOnObject=".$localid.";";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['Id_Wght'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------
//обновляем камеру по ИД
function CamerasUpdateById($mysqli, $idindb, $GlobalCameraId, $IPaddr, $GetQuary, $UserName, $AccessPsw, $Mode) {
	$query="UPDATE Cameras SET IPaddr='".$IPaddr."', ";
	if ($GetQuary!="") {$query=$query."Get_quary='".$GetQuary."', ";} else {$query=$query."Get_quary=NULL, ";}
	if ($UserName!="") {$query=$query."User_name='".$UserName."', ";} else {$query=$query."User_name=NULL,
";}
	if ($AccessPsw!="") {$query=$query."Access_psw='".$AccessPsw."', ";} else {$query=$query."Access_psw=NULL,
";}
	$query=$query."Mode=".$Mode;
	$query=$query." WHERE Id_camera=".$GlobalCameraId.";";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------
//находим максимальный ИД камеры в БД
function CamerasGetMaxId($mysqli){
	$result=0;
	$query="SELECT MAX(Id_camera) as _maxID FROM Cameras;";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['_maxID'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------
//добавляем новую камеру в глобальную БД
function CamerasAddNew($mysqli, $IdScale, $IdOnObject, $IPaddr, $GetQuary, $UserName, $AccessPsw, $Mode) {
	$newid=CamerasGetMaxId($mysqli)+1;
	$scid=ScalesIDInDB($mysqli, $IdScale);
	if ($GetQuary=="") {$GetQuary="NULL"; } else {$GetQuary="'".$GetQuary."';}
	if ($UserName=="") {$UserName="NULL"; } else {$UserName="'".$UserName."';}
	if ($AccessPsw=="") {$AccessPsw="NULL"; } else {$AccessPsw="'".$AccessPsw."';}
	$query="INSERT INTO Cameras VALUES(".$newid.", ".$scid.", ".$IdOnObject.", '".$IPaddr."', ".$GetQuary.",
".$UserName.", ".$AccessPsw.", ";
	$query=$query.$Mode.", NULL);";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------
//Находим глобальный индекс камеры по локальному ИД и ИД весов - или 0 если такого нет
function CamerasFindIdByLocalId($mysqli, $scid, $localid) {
	$result=0;
	$query="SELECT Id_camera FROM Cameras WHERE IdScales=".$scid." AND IdOnObject=".$localid.";";
	$res = mysqli_query($mysqli,$query);
```

```
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['Id_camera'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------
//обновляем машину по ИД
function CarsUpdateById($mysqli, $idindb, $GlobalCarId, $CarNumb, $CarMark, $CarColor, $DriverId, $TrailerId, $Activity) {
	$query="UPDATE Cars SET car_numb='".$CarNumb."', ";
	if ($CarMark!="") {$query=$query."car_mark='".$CarMark."', ";} else {$query=$query."car_mark=NULL, ";}
	if ($CarColor!="") {$query=$query."Car_color='".$CarColor."', ";} else {$query=$query."Car_color=NULL, ";}
	if ($DriverId>0) {
		$GDId=DriversFindIdByLocalId($mysqli, $idindb, $DriverId);
		$query=$query."Driver_numb=".$GDId.", ";
		}
		else {$query=$query."Driver_numb=NULL, ";}
	if ($TrailerId>0) {
		$GTId=TrailersFindIdByLocalId($mysqli, $idindb, $TrailerId);
		$query=$query."TrlLink=".$TrailerId.", ";
		}
		else {$query=$query."TrlLink=NULL, ";}
	$query=$query."Activity=".$Activity;
	$query=$query." WHERE Id_car=".$GlobalCarId.";";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------
//находим максимальный ИД машины в БД
function CarsGetMaxId($mysqli){
	$result=0;
	$query="SELECT MAX(Id_car) as _maxID FROM Cars;";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['_maxID'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------
//добавляем новую машину в глобальную БД
function CarsAddNew($mysqli, $IdScale, $IdOnObject, $CarNumb, $CarMark, $CarColor, $DriverId, $TrailerId, $Activity) {
	$newid=CarsGetMaxId($mysqli)+1;
	$scid=ScalesIDInDB($mysqli, $IdScale);
	if ($CarMark=="") {$CarMark="NULL"; } else {$CarMark="'".$CarMark."';}
	if ($CarColor=="") {$CarColor="NULL"; } else {$CarColor="'".$CarColor."';}
	if ($DriverId>0) {$GDId=DriversFindIdByLocalId($mysqli, $scid, $DriverId);}
		else {$GDId="NULL";}
	if ($TrailerId>0) {$GTId=TrailersFindIdByLocalId($mysqli, $scid, $TrailerId);}
		else {$GTId="NULL";}
	$query="INSERT INTO Cars VALUES(".$newid.", ".$scid.", ".$IdOnObject.", '".$CarNumb."', ".$CarMark.", ".$CarColor.", 0, ".$GDId.", ";
	$query=$query."NULL, NULL, NULL, NULL, ".$GTId.", ".$Activity.", NULL);";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------
//Находим глобальный индекс машины по локальному ИД и ИД весов - или 0 если такого нет
function CarsFindIdByLocalId($mysqli, $scid, $localid) {
	$result=0;
	$query="SELECT Id_car FROM Cars WHERE IdScales=".$scid." AND IdOnObject=".$localid.";";
	$res = mysqli_query($mysqli,$query);
```

```
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['Id_car'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//обновляем прицеп по ИД
function TrailersUpdateById($mysqli, $GlobalSenderId, $TrNumb, $Activity) {
	$query="UPDATE Trailers SET Trailer_Numb='".$TrNumb."', ";
	$query=$query."Activity=".$Activity;
	$query=$query." WHERE Id_Trailer=".$GlobalDriverId.";";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//находим максимальный ИД прицепа в БД
function TrailersGetMaxId($mysqli){
	$result=0;
	$query="SELECT MAX(Id_Trailer) as _maxID FROM Trailers;";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['_maxID'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//добавляем новый прицеп в глобальную БД
function TrailersAddNew($mysqli, $IdScale, $IdOnObject, $TrNumb, $Activity) {
	$newid=TrailersGetMaxId($mysqli)+1;
	$scid=ScalesIDInDB($mysqli, $IdScale);
	$query="INSERT INTO Trailers VALUES(".$newid.", ".$scid.", ".$IdOnObject.", '".$TrNumb."', ".$Activity.",
NULL);";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//Находим глобальный индекс прицепа по локальному ИД и ИД весов - или 0 если такого нет
function TrailersFindIdByLocalId($mysqli, $scid, $localid) {
	$result=0;
	$query="SELECT Id_Trailer FROM Trailers WHERE IdScales=".$scid." AND IdOnObject=".$localid.";";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['Id_Trailer'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//обновляем перевозчика по ИД
function CariersUpdateById($mysqli, $GlobalCarierId, $CarierName, $ERDPOU, $AddressTXT, $Activity) {
	$query="UPDATE Cariers SET Carier_name='".$CarierName."', ";
	if ($ERDPOU!="") {$query=$query."ERDPOU='".$ERDPOU."', ";} else {$query=$query."ERDPOU=NULL, ";}
	if ($AddressTXT!="") {$query=$query."CarierAddressTXT='".$AddressTXT."', ";} else
{$query=$query."CarierAddressTXT=NULL, ";}
	$query=$query."Activity=".$Activity;
	$query=$query." WHERE Id_carier=".$GlobalCarierId.";";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//находим максимальный ИД перевозчика в БД
```

```
function CariersGetMaxId($mysqli){
	$result=0;
	$query="SELECT MAX(Id_carier) as _maxID FROM Cariers;";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['_maxID'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//добавляем нового перевозчика в глобальную БД
function CariersAddNew($mysqli, $IdScale, $IdOnObject, $CarierName, $ERDPOU, $AddressTXT, $Activity) {
	$newid=CariersGetMaxId($mysqli)+1;
	$scid=ScalesIDInDB($mysqli, $IdScale);
	if ($ERDPOU=="") {$ERDPOU="NULL"; } else {$ERDPOU="'".$ERDPOU."'";}
	if ($AddressTXT=="") {$AddressTXT="NULL"; } else {$AddressTXT="'".$AddressTXT."'";}
	$query="INSERT INTO Cariers VALUES(".$newid.", ".$scid.", ".$IdOnObject.", '".$CarierName."', 1,
".$ERDPOU.", ".$AddressTXT.", ".$Activity.", NULL);";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//Находим глобальный индекс перевозчика по локальному ИД и ИД весов - или 0 если такого нет
function CariersFindIdByLocalId($mysqli, $scid, $localid) {
	$result=0;
	$query="SELECT Id_carier FROM Cariers WHERE IdScales=".$scid." AND IdOnObject=".$localid.";";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['Id_carier'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//обновляем плательщика по ИД
function PayersUpdateById($mysqli, $GlobalPayerId, $PayerName, $ERDPOU, $AddressTXT, $Activity) {
	$query="UPDATE Payers SET Payer_name='".$PayerName."', ";
	if ($ERDPOU!="") {$query=$query."ERDPOU='".$ERDPOU."', ";} else {$query=$query."ERDPOU=NULL, ";}
	if ($AddressTXT!="") {$query=$query."PayerAddressTXT='".$AddressTXT."', ";} else
{$query=$query."PayerAddressTXT=NULL, ";}
	$query=$query."Activity=".$Activity;
	$query=$query." WHERE Id_payer=".$GlobalPayerId.";";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//находим максимальный ИД плательщика в БД
function PayersGetMaxId($mysqli){
	$result=0;
	$query="SELECT MAX(Id_payer) as _maxID FROM Payers;";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['_maxID'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//добавляем нового плательщика в глобальную БД
function PayersAddNew($mysqli, $IdScale, $IdOnObject, $PayerName, $ERDPOU, $AddressTXT, $Activity) {
	$newid=PayersGetMaxId($mysqli)+1;
```

```
	$scid=ScalesIDInDB($mysqli, $IdScale);
	if ($ERDPOU=="") {$ERDPOU="NULL"; } else {$ERDPOU="'".$ERDPOU."'";}
	if ($AddressTXT=="") {$AddressTXT="NULL"; } else {$AddressTXT="'".$AddressTXT."'";}
	$query="INSERT INTO Payers VALUES(".$newid.", ".$scid.", ".$IdOnObject.", '".$PayerName."', 1, ".$ERDPOU.",
".$AddressTXT.", ".$Activity.", NULL);";
	$res = mysqli_query($mysqli,$query);
}
//-------------------------------------------------------------------------------------------------------------------------------
//Находим глобальный индекс плательщика по локальному ИД и ИД весов - или 0 если такого нет
function PayersFindIdByLocalId($mysqli, $scid, $localid) {
	$result=0;
	$query="SELECT Id_payer FROM Payers WHERE IdScales=".$scid." AND IdOnObject=".$localid.";";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['Id_payer'];
		}
	}
	return $result;
}
//-------------------------------------------------------------------------------------------------------------------------------
//обновляем получателя по ИД
function RecipientsUpdateById($mysqli, $GlobalRcpId, $RcpName, $ERDPOU, $AddressTXT, $Activity) {
	$query="UPDATE Recipients SET Recp_name='".$RcpName."', ";
	if ($ERDPOU!="") {$query=$query."ERDPOU='".$ERDPOU."', ";} else {$query=$query."ERDPOU=NULL, ";}
	if ($AddressTXT!="") {$query=$query."RecpAddressTXT='".$AddressTXT."', ";} else
{$query=$query."RecpAddressTXT=NULL, ";}
	$query=$query."Activity=".$Activity;
	$query=$query." WHERE Id_recipient=".$GlobalRcpId.";";
	$res = mysqli_query($mysqli,$query);
}
//-------------------------------------------------------------------------------------------------------------------------------
//находим максимальный ИД получателя в БД
function RecipientsGetMaxId($mysqli){
	$result=0;
	$query="SELECT MAX(Id_recipient) as _maxID FROM Recipients;";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['_maxID'];
		}
	}
	return $result;
}
//-------------------------------------------------------------------------------------------------------------------------------
//добавляем нового получателя в глобальную БД
function RecipientsAddNew($mysqli, $IdScale, $IdOnObject, $RcpName, $ERDPOU, $AddressTXT, $Activity) {
	$newid=RecipientsGetMaxId($mysqli)+1;
	$scid=ScalesIDInDB($mysqli, $IdScale);
	if ($ERDPOU=="") {$ERDPOU="NULL"; } else {$ERDPOU="'".$ERDPOU."'";}
	if ($AddressTXT=="") {$AddressTXT="NULL"; } else {$AddressTXT="'".$AddressTXT."'";}
	$query="INSERT INTO Recipients VALUES(".$newid.", ".$scid.", ".$IdOnObject.", '".$RcpName."', 1,
".$ERDPOU.", ".$AddressTXT.", ".$Activity.", NULL);";
	$res = mysqli_query($mysqli,$query);
}
//-------------------------------------------------------------------------------------------------------------------------------
//Находим глобальный индекс получателя по локальному ИД и ИД весов - или 0 если такого нет
function RecipientsFindIdByLocalId($mysqli, $scid, $localid) {
	$result=0;
	$query="SELECT Id_recipient FROM Recipients WHERE IdScales=".$scid." AND IdOnObject=".$localid.";";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
```

```
                        $result=$row['Id_recipient'];
                }
        }
        return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//обновляем отправителя по ИД
function SendersUpdateById($mysqli, $GlobalSenderId, $SenderName, $ERDPOU, $AddressTXT, $Activity) {
        $query="UPDATE Senders SET Sender_name='".$SenderName."', ";
        if ($ERDPOU!="") {$query=$query."ERDPOU='".$ERDPOU."', ";} else {$query=$query."ERDPOU=NULL, ";}
        if ($AddressTXT!="") {$query=$query."AddressTXT='".$AddressTXT."', ";} else {$query=$query."AddressTXT=NULL, ";}
        $query=$query."Activity=".$Activity;
        $query=$query." WHERE Id_sender=".$GlobalSenderId.";";
        $res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//находим максимальный ИД отправителя в БД
function SendersGetMaxId($mysqli){
        $result=0;
        $query="SELECT MAX(Id_sender) as _maxID FROM Senders;";
        $res = mysqli_query($mysqli,$query);
        if ($res){
                while($row = mysqli_fetch_array($res)) {
                        $result=$row['_maxID'];
                }
        }
        return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//добавляем нового отправителя в глобальную БД
function SendersAddNew($mysqli, $IdScale, $IdOnObject, $SenderName, $ERDPOU, $AddressTXT, $Activity) {
        $newid=SendersGetMaxId($mysqli)+1;
        $scid=ScalesIDInDB($mysqli, $IdScale);
        if ($ERDPOU=="") {$ERDPOU="NULL"; } else {$ERDPOU="'".$ERDPOU."';}
        if ($AddressTXT=="") {$AddressTXT="NULL"; } else {$AddressTXT="'".$AddressTXT."';}
        $query="INSERT INTO Senders VALUES(".$newid.", ".$scid.", ".$IdOnObject.", '".$SenderName."', 1, ".$ERDPOU.", ".$AddressTXT.", ".$Activity.", NULL);";
        $res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//Находим глобальный индекс отправителя по локальному ИД и ИД весов - или 0 если такого нет
function SendersFindIdByLocalId($mysqli, $scid, $localid) {
        $result=0;
        $query="SELECT Id_sender FROM Senders WHERE IdScales=".$scid." AND IdOnObject=".$localid.";";
        $res = mysqli_query($mysqli,$query);
        if ($res){
                while($row = mysqli_fetch_array($res)) {
                        $result=$row['Id_sender'];
                }
        }
        return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//обновляем водителя по ИД
function DriversUpdateById($mysqli, $GlobalDriverId, $DriverName, $DriverFName, $DriverSName, $DriverPhone, $DriverDoc, $Activity) {
        $query="UPDATE Drivers SET driver_surname='".$DriverSName."', ";
        if ($DriverName!="") {$query=$query."driver_name='".$DriverName."', ";}
        if ($DriverFName!="") {$query=$query."driver_fname='".$DriverFName."', ";}
        if ($DriverPhone!="") {$query=$query."DriverPhone='".$DriverPhone."', ";}
        if ($DriverDoc!="") {$query=$query."DriverDoc='".$DriverDoc."', ";}
        $query=$query."Activity=".$Activity;
```

```
	$query=$query." WHERE Id_driver=".$GlobalDriverId.";";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//находим максимальный ИД водителя в БД
function DriversGetMaxId($mysqli){
	$result=0;
	$query="SELECT MAX(Id_driver) as _maxID FROM Drivers;";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['_maxID'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//добавляем новый груз в глобальную БД
function DriversAddNew($mysqli, $IdScale, $IdOnObject, $DriverName, $DriverFName, $DriverSName, $DriverPhone, $DriverDoc, $Activity) {
	$newid=DriversGetMaxId($mysqli)+1;
	$scid=ScalesIDInDB($mysqli, $IdScale);
	if ($DriverName=="") {$DriverName="NULL"; } else {$DriverName="'".$DriverName."';}
	if ($DriverFName=="") {$DriverFName="NULL"; } else {$DriverFName="'".$DriverFName."';}
	if ($DriverPhone=="") {$DriverPhone="NULL"; } else {$DriverPhone="'".$DriverPhone."';}
	if ($DriverDoc=="") {$DriverDoc="NULL"; } else {$DriverDoc="'".$DriverDoc."';}
	$query="INSERT INTO Drivers VALUES(".$newid.", ".$scid.", ".$IdOnObject.", ".$DriverName.", ".$DriverFName.", '".$DriverSName."', ".$Activity.", NULL, ".$DriverPhone.", ".$DriverDoc.");";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//Находим глобальный индекс груза по локальному ИД и ИД весов - или 0 если такого нет
function DriversFindIdByLocalId($mysqli, $scid, $localid) {
	$result=0;
	$query="SELECT Id_driver FROM Drivers WHERE IdScales=".$scid." AND IdOnObject=".$localid.";";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['Id_driver'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//обновляем груз по ИД
function CargosUpdateById($mysqli, $GlobalCargiId, $CargoName, $CargoPrice, $CargoCode, $Activity) {
	$query="UPDATE Cargos SET Cargo_name='".$CargoName."', ";
	$query=$query."CargoPrice=".$CargoPrice.", ";
	if ($CargoCode!="") {$query=$query."Cargo_code='".$CargoCode."', ";}
	$query=$query."Activity=".$Activity;
	$query=$query." WHERE Id_cargo=".$GlobalCargiId.";";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------
//находим максимальный ИД грузов
function CargosGetMaxId($mysqli){
	$result=0;
	$query="SELECT MAX(Id_cargo) as _maxID FROM Cargos;";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['_maxID'];
		}
```

```
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------------
//добавляем новый груз в глобальную БД
function CargosAddNew($mysqli, $IdScale, $IdOnObject, $CargoName, $CargoPrice, $CargoCode, $Activity) {
	$newid=CargosGetMaxId($mysqli)+1;
	$scid=ScalesIDInDB($mysqli, $IdScale);
	if ($CargoCode=="") {$CargoCode="NULL";} else {$CargoCode="'".$CargoCode."'";}
	$query="INSERT INTO Cargos VALUES(".$newid.", ".$scid.", ".$IdOnObject.", '".$CargoName."', ".$CargoPrice.",
".$CargoCode.", ".$Activity.", NULL );";
	$res = mysqli_query($mysqli,$query);
}
//-----------------------------------------------------------------------------------------------------------------------------------------------
//Находим глобальный индекс груза по локальному ИД и ИД весов - или 0 если такого груза нет
function CargosFindIdByLocalId($mysqli, $scid, $localid) {
	$result=0;
	$query="SELECT Id_cargo FROM Cargos WHERE IdScales=".$scid." AND IdOnObject=".$localid.";";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['Id_cargo'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------------
//запись запросов в журнал
function saveLog($string){
	$file = fopen('logs4.txt', 'a');
	$serverdate = date('d/m/Y h:i:s a', time());
	$clientIP=$_SERVER['REMOTE_ADDR'];
	fwrite($file, $serverdate." ".$clientIP.":".$string."\r\n");
	fclose($file);
}
//-----------------------------------------------------------------------------------------------------------------------------------------------
//функция которая вернет ИД в таблице терминалов/программ по ИД высланному пользователем или "0" если такого
нет
function ScalesIDInDB($mysqli,$sid){
	$result="0";
	$query="SELECT Id_scale FROM Scales WHERE Id_terminal='".$sid."';";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['Id_scale'];
		}
	}
	return $result;
}
//-----------------------------------------------------------------------------------------------------------------------------------------------
//функция которая вернет ключ проверки или "0" если нет указанного ИД=$sid весов
function GetSKeyByID($mysqli,$sid){
	$result="0";
	$query="SELECT Secret_key FROM Scales WHERE Id_scale=".$sid.";";
	$res = mysqli_query($mysqli,$query);
	if ($res){
		while($row = mysqli_fetch_array($res)) {
			$result=$row['Secret_key'];
		}
	}
	return $result;
}
?>
```

ДОДАТОК Б – ТЕКСТ ПРОГРАМНОГО МОДУЛЯ ПО РОБОТІ З ЦИФРОВИМИ ДАТЧИКАМИ

```
unit HBMDigital;

interface

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

//определим тип HBMThread - поток для цифровых датчиков
type THBMThread = class(TThread)
  private
    procedure Measuring; //процедура, занимающаяся опросом порта (датчиков)
    function HBMPortInit(numb:byte; br:longint):boolean;//инициализация
    //procedure COM_Close(port_numb:byte);//закрытие порта
    function ReadHBM(var Buf; Size: Word): Integer; //чтение порта
    function WriteHBMPort(var Buf; Size: Word): Integer;
    function SendStrToHBM(s_temp:string):boolean; //отправка строки в порт
    function SendAndWaitHBM(str:string):string; //отправка и ожидание ответа
    procedure StartReadingHBM;//стартовая посылка для чтения данных
    procedure HBMReadS0x(x:integer); //чтение показаний очередного датчика
    procedure HBMReadZeroLvl(x:integer);//чтение уровня нуля для отдельного датчика
    procedure HBMReadParamsSx(x:integer); //чтение параметров для отдельного датчика
    procedure HBMReadSensors; //читаем текущие показания датчиков
    procedure ADCForKgCalc; //расчет уровня АЦП для одного кг веса
    procedure HBMTakeParameters; //получение параметров датчиков
  protected
    procedure Execute; override;//переопределим метод запуска потока
  end;

var
  HBMThread:THBMThread; //поток чтени данных с цифровых датчиков
  FCOMHandle2:THandle; //заголовок работы с портом
  OldDCB2:TDCB; //структура содержащая настройки порта
  FCOM2:string; //строка с именем порта "COM1" например
  PortNumb:byte; //номер COM порта
  PortBR:longint; //скорость COM порта
  SZA:array[1..6] of integer;//массив начальных точек характеристики датчиков
  SFA:array[1..6] of integer;//массив конечных точек характеристики датчиков
  LIC1:array[1..6] of integer;//массив 1х точек линеризации датчика
  LIC2:array[1..6] of integer;//массив 2х точек линеризации датчика
  LIC3:array[1..6] of integer;//массив 3х точек линеризации датчика
  LIC4:array[1..6] of integer;//массив 4х точек линеризации датчика
  ZeroLvl:array[1..6] of integer;//массив нулей датчиков
  SensorLvl:array[1..6] of integer;//массив текущих показаний АЦП датчиков
  ADCForKg:real;//рассчетный коэффициент
  DeltaZero:real; //значение средней разницы показаний с нулем
  SensorAverage:real;//усредненное показание датчиков
  PortOpened:boolean;//переменная признак соединения с портом
  MeasStarted:boolean; //запущено измерение
  WghtNowInt:integer;//значение веса
  WghtPrev:integer;//предыдущее значение веса
  NeedHBMProtokol:boolean;//необходимость вывода протокола обмена в форму
  HBMMaxWght:integer;//максимальный предел взвешивания весов
  HBMCalKoef:real;//калибровочный коэффициент
  HBMWghtStep:integer;//шаг показаний весов

  procedure SaveHBMWghtToFile; //запись текущих показаний в файл
```

```
  procedure DeltaZeroCalc; //рассчет разницы показаний с нулем
  procedure Taring; //обтаривание весов
  procedure HBMWghtCalc;
  procedure StartHBMThread;
  procedure HBMZeroLvlShow;//запись уровней нуля на форму

implementation

uses
  Main,Scales,Settings;

//инициальзация порта в потоке чтения цифровых датчиков
function THBMThread.HBMPortInit(numb:byte; br:longint):boolean;
var
  Timeouts:TCommTimeouts;
  s_temp:string;
begin
  FCOM2:='COM'+inttostr(numb);
  Result:=False;
  //Создание и иницализация порта
  FComHandle2:=CreateFile(PChar('\\.\'+FCOM2), GENERIC_READ or GENERIC_WRITE,
        0, nil, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
  if (FComHandle2=INVALID_HANDLE_VALUE) or (FComHandle2=0) then Exit;
  if not(Windows.GetCommState(FComHandle2,OldDCB2)) 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.
  OldDCB2.BaudRate:=br;
  OldDCB2.Parity:=EVENPARITY; //!!!!добавить заполнение из настроек
  // бывает EVENPARITY, MARKPARITY, NOPARITY, ODDPARITY, SPACEPARITY
  OldDCB2.ByteSize:=8; //!!!!добавить заполнение из настроек
  OldDCB2.StopBits:=ONESTOPBIT;  //!!!!добавить заполнение из настроек
  //бывает ONESTOPBIT, ONE5STOPBIT, TWOSTOPBIT
  if not(Windows.SetCommState(FComHandle2,OldDCB2)) then Exit;//если нет портов - выход
  if Not SetupComm(FComHandle2,256,256) then Exit; //размеры буфера приема и передачи
  if GetCommTimeouts(FComHandle2,Timeouts) then
    begin
      Timeouts.ReadIntervalTimeout:=50;//50
      Timeouts.ReadTotalTimeoutMultiplier:=0;
      Timeouts.ReadTotalTimeoutConstant:=50; //50
      Timeouts.WriteTotalTimeoutMultiplier:=1;
      Timeouts.WriteTotalTimeoutConstant:=10;//10
      if not SetCommTimeouts(FComHandle2,Timeouts) then Exit;
    end
  else
    Exit;
  // Сброс порта
  if not(PurgeComm(FComHandle2, PURGE_TXABORT or PURGE_RXABORT or PURGE_TXCLEAR or
PURGE_RXCLEAR)) then Exit;
  EscapeCommFunction(FComHandle2, CLRRTS);
  if not SetCommMask(FComHandle2, EV_RXCHAR) then Exit;
  Result:=True;
  PortOpened:=true;
end;

//запись пакета в COM порт
function THBMThread.WriteHBMPort(var Buf; Size: Word): Integer;
var
  p: pointer;
  i: cardinal;
```

```
begin
  p := @Buf;
  result := 0;
  while size > 0 do begin
    if not WriteFile(FComHandle2, p^, 1, i, nil) then exit;
    inc(result, i);
    inc(integer(p));
    dec(size);
    Application.ProcessMessages;
    end;
 // PostComm := result;
end;

//отправка строки s_temp в порт
function THBMThread.SendStrToHBM(s_temp:string):boolean;
var
  OutputBuffer: array[0..63] of char;  // выходной буфер размером 64 символов (или сколько надо)
  ErrorCode: integer;  // переменная для приёма кода ошибки
  SendLen:integer;
begin
// буфер  заполняется
  Result:=false; //'Ошибка передачи пакета';
  StrPCopy(OutputBuffer,s_temp);
//запись
  ErrorCode:= WriteHBMPort(OutputBuffer, Length(s_temp));  // здесь собственно передача пакета
  SendLen:=length(s_temp);
  if ErrorCode = SendLen then Result:=true//'Пакет передан'
  else begin
    Result:=false;//'Ошибка передачи пакета';
    end;
end;

//чтение с COM порта
function THBMThread.ReadHBM(var Buf; Size: Word): Integer;
var
  NumberOfBytesReaded: Cardinal;
begin
  Result:=0;
  if not ReadFile(FCOMHandle2, Buf, Size, NumberOfBytesReaded, nil) then exit;
  Result:=NumberOfBytesReaded;
end;

//отправка и проверка ответа если отправка успешная
//возврат - считанное после отправки значение
function THBMThread.SendAndWaitHBM(str:string):string;
var
  InputBuffer: array[0..255] of char;  // приёмный буфер размером 256 символов (или сколько надо)
  ErrorCode:integer;
  s_temp:string;
begin
  ErrorCode:=0;
  Result:='';
  if SendStrToHBM(str) then begin
    FillChar(InputBuffer, Length(InputBuffer),0);
    ErrorCode:=ReadHBM(InputBuffer, Length(InputBuffer));
    end;
  if ErrorCode > 0 then begin
    s_temp:=string(InputBuffer);
    Result:=s_temp;
    end;
end;

procedure THBMThread.StartReadingHBM;
```

```
var
  s_temp:string;
begin
  s_temp:=#$0A+'S98;msv?;';
  if NeedHBMProtokol then Form_Settings.Memo_Debug.Lines.Add('Sended:'+s_temp);
  SendAndWaitHBM(s_temp);
end;

procedure THBMThread.HBMReadS0x(x:integer);
var
  s_temp,s_temp1:string;
  c,n:integer;
begin
  s_temp:=#$0A+'S0'+IntToStr(x)+';';
  if NeedHBMProtokol then Form_Settings.Memo_Debug.Lines.Add('Sended:'+s_temp);
  s_temp:=SendAndWaitHBM(s_temp);
  if NeedHBMProtokol then Form_Settings.Memo_Debug.Lines.Add('Readed:'+s_temp);
  val(s_temp,SensorLvl[x],c);
  Form_Settings.StringGrid_Prms.Cells[7,x]:=IntToStr(SensorLvl[x]);
end;

//сохранение веса в файл
procedure SaveHBMWghtToFile;
var
  t:textfile;
  SLTmp:TStringList;
begin
  SLTmp:=TStringList.Create;
  SLTmp.Add(IntToStr(WghtNowInt));
  SLTmp.SaveToFile(MyDir+'\Wght.txt');
  FreeAndNil(SLTmp);
end;

//выводим на форму уровни нуля датчиков
procedure HBMZeroLvlShow;
var
  i:integer;
  s_temp,s_temp1:string;
begin
  for i:=1 to 6 do begin
    s_temp:=IntToStr(ZeroLvl[i]);
    Form_Settings.StringGrid_Prms.Cells[6,i]:=s_temp;
    end;
end;

//чтение уровня нуля для датчика номер x
procedure THBMThread.HBMReadZeroLvl(x:integer);
var
  s_temp:string;
  c:integer;
begin
  s_temp:=#$0A+'S0'+IntToStr(x)+';';
  if NeedHBMProtokol then Form_Settings.Memo_Debug.Lines.Add('Sended:'+s_temp);
  s_temp:=SendAndWaitHBM(s_temp);
  if NeedHBMProtokol then Form_Settings.Memo_Debug.Lines.Add('Readed:'+s_temp);
  val(s_temp,ZeroLvl[x],c);
  Form_Settings.StringGrid_Prms.Cells[6,x]:=IntToStr(ZeroLvl[x]);
end;

//чтение параметров для датчика номер x
procedure THBMThread.HBMReadParamsSx(x:integer);
var
  s_temp,s_temp1:string;
```

```
  n,c:integer;
begin
  HBMReadZeroLvl(x);
  s_temp:='SZA?;';
  if NeedHBMProtokol then Form_Settings.Memo_Debug.Lines.Add('Sended:'+s_temp);
  s_temp:=SendAndWaitHBM(s_temp);
  if NeedHBMProtokol then Form_Settings.Memo_Debug.Lines.Add('Readed:'+s_temp);
  val(s_temp,SZA[x],c);
  if (copy(s_temp,1,1)='-') and (SZA[x]>0) then SZA[x]:=-1*SZA[x];
  Form_Settings.StringGrid_Prms.Cells[0,x]:=IntToStr(SZA[x]);
  s_temp:='SFA?;';
  if NeedHBMProtokol then Form_Settings.Memo_Debug.Lines.Add('Sended:'+s_temp);
  s_temp:=SendAndWaitHBM(s_temp);
  if NeedHBMProtokol then Form_Settings.Memo_Debug.Lines.Add('Readed:'+s_temp);
  val(s_temp,SFA[x],c);
  Form_Settings.StringGrid_Prms.Cells[1,x]:=IntToStr(SFA[x]);
  s_temp:='LIC?;';
  if NeedHBMProtokol then Form_Settings.Memo_Debug.Lines.Add('Sended:'+s_temp);
  s_temp:=SendAndWaitHBM(s_temp);
  if NeedHBMProtokol then Form_Settings.Memo_Debug.Lines.Add('Readed:'+s_temp);
  n:=pos(',',s_temp);
  s_temp1:=copy(s_temp,1,n-1);
  val(s_temp1,LIC1[x],c);
  Form_Settings.StringGrid_Prms.Cells[2,x]:=IntToStr(LIC1[x]);
  s_temp:=copy(s_temp,n+1,length(s_temp)-n);
  n:=pos(',',s_temp);
  s_temp1:=copy(s_temp,1,n-1);
  val(s_temp1,LIC2[x],c);
  Form_Settings.StringGrid_Prms.Cells[3,x]:=IntToStr(LIC2[x]);
  s_temp:=copy(s_temp,n+1,length(s_temp)-n);
  n:=pos(',',s_temp);
  s_temp1:=copy(s_temp,1,n-1);
  val(s_temp1,LIC3[x],c);
  Form_Settings.StringGrid_Prms.Cells[4,x]:=IntToStr(LIC3[x]);
  s_temp:=copy(s_temp,n+1,length(s_temp)-n);
  val(s_temp,LIC4[x],c);
  Form_Settings.StringGrid_Prms.Cells[5,x]:=IntToStr(LIC4[x]);
end;

//читаем текущие показания датчиков
procedure THBMThread.HBMReadSensors;
var
  i:integer;
begin
  StartReadingHBM;
  for i:=1 to 6 do
    HBMReadS0x(i);
end;

procedure DeltaZeroCalc;
var
  i:integer;
  itemp:integer;
begin
  itemp:=0;
  for i:=1 to 6 do
    itemp:=itemp+ZeroLvl[i];
  DeltaZero:=itemp/6;
end;

procedure THBMThread.ADCForKgCalc;
var
  i:integer;
```

```
  itemp:integer;
  rtemp:real;
begin
  itemp:=0;
  for i:=1 to 6 do
    itemp:=itemp+SFA[i]-SZA[i];
  rtemp:=itemp/6;
  ADCForKg:=(rtemp-DeltaZero)/HBMMaxWght;
end;

//проверяем не поврежден ли файл с параметрами - если нормальный читаем уровни нуля
function IsPrmsFileGood:boolean;
var
  s_temp:string;
  SLTemp:TStringList;
  i,n,c:integer;
begin
  Result:=false;
  if not FileExists(MyDir+'\Params.txt') then Exit;
  SLTemp:=TStringList.Create;
  SLTemp.LoadFromFile(MyDir+'\Params.txt');
  if SLtemp.Count<42 then begin //если меньше строчек в настройках - файл поврежден
    FreeAndNil(SLTemp);
    Exit;
    end;
  for i:=0 to 5 do begin
    s_temp:=copy(SLTemp[i],10,length(SLTemp[i])-9);
    val(s_temp,n,c);
    if c<>0 then begin
      FreeAndNil(SLTemp);
      Exit;
      end;
    if n<10000 then begin
      FreeAndNil(SLTemp);
      Exit;
      end;
    ZeroLvl[i+1]:=n;
    end;
  FreeAndNil(SLTemp);
  Result:=true;
end;

procedure THBMThread.HBMTakeParameters;
var
  i:integer;
begin
  //читаем параметры датчиков
  for i:=1 to 6 do begin
    HBMReadParamsSx(i);
    end;
  //достаем из файла или читаем параметры уровня нуля
  //если файл с параметрами существует и не поврежден - читаем из него уровни нулей
  if not IsPrmsFileGood then
    for i:=1 to 6 do
      while ZeroLvl[i]<10000 do begin
        StartReadingHBM;
        HBMReadZeroLvl(i);
        end;
  HBMZeroLvlShow;
  DeltaZeroCalc;
  ADCForKgCalc;
end;
```

```
//сохраняем дополнительные параметры в файл
procedure SaveParamsToFile;
var
  t:textfile;
  i:integer;
  s_temp:string;
begin
  AssignFile(t,'Params.txt');
  Rewrite(t);
  for i:=1 to 6 do begin
    s_temp:='ZeroLvl'+IntToStr(i)+'='+IntToStr(ZeroLvl[i]);
    writeln(t,s_temp);
    end;
  for i:=1 to 6 do begin
    s_temp:='SZA'+IntToStr(i)+'='+IntToStr(SZA[i]);
    writeln(t,s_temp);
    end;
  for i:=1 to 6 do begin
    s_temp:='SFA'+IntToStr(i)+'='+IntToStr(SFA[i]);
    writeln(t,s_temp);
    end;
  for i:=1 to 6 do begin
    s_temp:='LIC'+IntToStr(i)+'1='+IntToStr(LIC1[i]);
    writeln(t,s_temp);
    s_temp:='LIC'+IntToStr(i)+'2='+IntToStr(LIC2[i]);
    writeln(t,s_temp);
    s_temp:='LIC'+IntToStr(i)+'3='+IntToStr(LIC3[i]);
    writeln(t,s_temp);
    s_temp:='LIC'+IntToStr(i)+'4='+IntToStr(LIC4[i]);
    writeln(t,s_temp);
    end;
  closefile(t);
end;

//вычисление среднего показания датчиков
procedure SensorAvCalc;
var
  i:integer;
begin
  SensorAverage:=0;
  for i:=1 to 6 do
     SensorAverage:=SensorAverage+SensorLvl[i];
  SensorAverage:=SensorAverage/6;
end;

//вычисление текущего веса
procedure HBMWghtCalc;
begin
  SensorAvCalc;
  WghtNowInt:=round((SensorAverage-DeltaZero)/ADCForKg*HBMCalKoef);
end;

//округление веса до указанного шага весов
procedure WghtRounding;
begin
  WghtNowInt:=round(WghtNowInt/HBMWghtStep)*HBMWghtStep;
end;

//расчет веса по параметрам
procedure CalcWght;
var
  i:integer;
  s_temp:string;
```

```
begin
  for i:=1 to 6 do
    if SensorLvl[i]<10000 then Exit;
  HBMWghtCalc; //вычисление текущего веса
  Form_Settings.Label_SensAv.Caption:=FormatFloat('0.##',SensorAverage);
  Form_Settings.Label_ADCForKg.Caption:=FormatFloat('0.##',ADCForKg);
  Form_Settings.Label_DeltaZero.Caption:=FormatFloat('0.##',DeltaZero);
  WghtRounding;//округление веса до указанного шага весов
  if WghtNowInt<>WghtPrev then SaveHBMWghtToFile;
  WghtPrev:=WghtNowInt;
  if WghtNowInt<0 then begin
    HBMDigital.Taring;
    HBMDigital.HBMZeroLvlShow;
    WghtNowInt:=0;
    end;
  s_temp:=IntToStr(WghtNowInt);
  Form_Settings.Label_WghtVal.Caption:=s_temp;
  Form_Main.Label_Wght.Caption:=s_temp;
end;

procedure THBMThread.Measuring;
begin
  if not PortOpened then HBMPortInit(COMPortNumb, COMPortBr);
  if not PortOpened then begin
    Form_Settings.Memo_Debug.Lines.Add('No Connection');
    Form_Main.Label_COMNumb.Font.Color:=clRed;
    Exit;
    end;
  if not MeasStarted then begin
    MeasStarted:=true;
    HBMTakeParameters;
    SaveParamsToFile;
    Form_Main.Label_COMNumb.Font.Color:=clGreen;
    end;
  if MeasStarted then begin
    HBMReadSensors;
    CalcWght;
    end;
  if (NeedHBMProtokol) and (Form_Settings.Memo_Debug.Lines.Count>50) then
    Form_Settings.Memo_Debug.Lines.Clear;
end;

//обтаривание весов - прописываем текущие показания как точку нуля
procedure Taring;
var
  j:integer;
begin
//  ReadSensors;
  for j:=1 to 6 do
    if SensorLvl[j]>0 then ZeroLvl[j]:=SensorLvl[j];
  DeltaZeroCalc;
  SaveParamsToFile;
end;

//устанавливаем начальные нулевые параметры
procedure ResetHBMPrms;
var
  i:integer;
begin
  //массив начальных точек характеристики датчиков
  for i:=1 to 6 do
    SZA[i]:=0;
  //массив конечных точек характеристики датчиков
```

```
  for i:=1 to 6 do
    SFA[i]:=0;
  //массив 1х точек линеризации датчика
  for i:=1 to 6 do
    LIC1[i]:=0;
  //массив 2х точек линеризации датчика
  for i:=1 to 6 do
    LIC2[i]:=0;
  //массив 3х точек линеризации датчика
  for i:=1 to 6 do
    LIC3[i]:=0;
  //массив 4х точек линеризации датчика
  for i:=1 to 6 do
    LIC4[i]:=0;
  //массив нулей датчиков
  for i:=1 to 6 do
    ZeroLvl[i]:=0;
  //массив текущих показаний АЦП датчиков
  for i:=1 to 6 do
    SensorLvl[i]:=0;
  ADCForKg:=0;//рассчетный коэффициент
  DeltaZero:=0; //значение средней разницы показаний с нулем
  SensorAverage:=0;//усредненное показание датчиков
  PortOpened:=false;//переменная признак соединения с портом
  MeasStarted:=false; //запущено измерение
  WghtNowInt:=0;//значение веса
  WghtPrev:=0;//предыдущее значение веса
end;

//инициализация потока для цифровых датчиков
procedure StartHBMThread;
begin
  ResetHBMPrms; //устанавливаем начальные нулевые параметры
  //пытаемся инициализировать поток
  HBMThread := THBMThread.Create(False);
  //проверяем получилось или нет
  if HBMThread = nil then
  begin
    //ошибка, все выключаем
    SysErrorMessage(GetLastError);
    Exit;
  end;
end;

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

end.
```