МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
Факультет мехатроніки та комп'ютерних технологій
Кафедра комп'ютерно-інтегрованих технологій та вимірювальної техніки

Дипломна магістерська робота

на тему «**Автоматизована система оптимізації підбору раціону при вигодовуванні свійських тварин**»

Виконав: студент 2 курсу, групи
МгАт-19 спеціальності 151
«Автоматизація та комп'ютерно-інтегровані технології»

Ножка Сергій Ігорович

Науковий керівник к.ф.-м.н., доцент
Пилипенко Ю.М.
Рецензент д.т.н., проф. Чупринка В.І

Київ 2020

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерно-інтегрованих технологій та вимірювальної техніки

Спеціальність 151 - автоматизація та комп'ютерно-інтегровані технології

Освітня програма – автоматизоване управління технологічними процесами

ЗАТВЕРДЖУЮ

Завідувач кафедри КІТВТ

проф., д.т.н. Здоренко В.Г.

"___" _____ 2020 р.

З А В Д А Н Н Я

НА ДИПЛОМНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ
Ножці Сергію Ігоровичу

1. Тема роботи «Автоматизована система оптимізації підбору раціону при вигодовуванні свійських тварин», науковий керівник роботи Пилипенко Юрій Миколайович, к.ф.-м.н., доцент, затверджені наказом вищого навчального закладу від 29 вересня 2020 року, № 183-уч.
2. Строк подання студентом роботи - 07 грудня 2020 р.
3. Вихідні дані до роботи: числа для створення бази даних; номер місяця та дня, для проведення розрахунків; числовий параметр, який відповідає коду тварини у базі даних.
4. Зміст дипломної роботи (перелік питань, які потрібно розробити): Вступ. Розділ 1. Постанова задачі моделювання оптимального щоденного підбору раціону при вигодовування сільськогосподарських тварин та апаратна частина для створення моделі автоматизованої системи. Розділ 2. Інформаційне, математичне та алгоритмічне забезпечення поставленої задачі. Розділ 3. Реалізація моделі за

допомогою мікропроцесорної платформи Arduino. Розділ 4. Адаптація розробленого алгоритму для промислового використання. Загальні висновки.

5**. Консультанти розділів роботи (проекту)**

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Вступ			
Розділ 1			
Розділ 2			
Розділ 3			
Розділ 4			
Висновки			

6. Дата видачі завдання 03 жовтня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного роботи (проекту)	Строк виконання етапів роботи	Примітка
1	Вступ	05.10.20	
2	Розділ 1	18.10.20	
3	Розділ 2	02.11.20	
4	Розділ 3	15.11.20	
5	Розділ 4	22.11.20	
6	Висновки	24.11.20	
7	Оформлення магістерської роботи (чистовий варіант)	28.11.20	
8	Підготовування анотації іноземною мовою	29.11.20	
9	Здача магістерської роботи на кафедру для рецензування	30.11.20	
10	Перевірка магістерської роботи на наявність ознак плагіату	03.12.20	
11	Подання роботи у відділ магістратури для перевірки виконання додатку до індивідуального навчального плану	05.12.20	
11	Подання роботи на затвердження завідувачу кафедри	07.12.20	

Студент _____ **Ножка С.І.**
(підпис) **(прізвище та ініціали)**

Керівник проекту (роботи) _____ **доц. Пилипенко Ю.М.**
(підпис) **(прізвище та ініціали)**

Керівник відділу магістратури _____ **доц. Черниш О.В.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Ножка С. І. Автоматизована система оптимізації підбору раціону при вигодовуванні свійських тварин. Рукопис.

Дипломна магістерська робота за спеціальністю 151 – «Автоматизація та комп'ютерно-інтегровані технології», Київський національний університет технологій та дизайну, Київ, 2020 рік.

Дипломна магістерська робота присвячена дослідженню автоматизованої системи керування процесом розрахунку добової норми різних типів кормів для різних видів тварин на кожен день року. Розглядається автоматизована система комбінування потрібних кормів для кожної групи тварин.

Результатами проведеної роботи є створена макетна схема для автоматизованого керування процедурою комбінування кормів і формування кінцевої кормової суміші для кожного дня року, що відбувається завдяки програмного розрахунку оптимальної добової кількості корму.

Ключові слова: автоматизація, автоматизована система керування, мікроконтролер, масив даних, математична модель.

АННОТАЦИЯ

Ножка С. И. Автоматизированная система оптимизации подбора рациона при выкармливании домашних животных. – Рукопись.

Дипломная магистерская работа по специальности 151 - «Автоматизация и компьютерно-интегрированные технологии», Киевский национальный университет технологий и дизайна, Киев, в 2020 год.

Дипломная магистерская работа посвящена исследованию автоматизированной системы управления процессом расчета суточной нормы различных типов кормов для различных видов животных на каждый день года. Рассматривается автоматизированная система комбинирования необходимых кормов для каждой группы животных.

Результатами проведенной работы является созданная макетная схема для автоматизированного управления процедурой комбинирования кормов и формирования конечной кормовой смеси для каждого дня года, происходит благодаря программному расчету оптимального суточного количества корма.

Ключевые слова: автоматизация, автоматизированная система управления, микроконтроллер, массив данных, математическая модель.

ANNOTATION

Nozhcka SI Automated system for optimizing the selection of the diet when feeding domestic animals. – Manuscript.

Master's thesis on specialty 151 - "Automation and computer-integrated technologies", Kyiv National University of Technology and Design, Kyiv, 2020.

The master's thesis is devoted to the study of the Automated control system of the process of calculating the daily norm of different types of feed for different species of animals for each day of the year. An automated system of combining the necessary feeds for each group of animals is considered.

The results of this work are Created a layout scheme for automated control of the procedure of combining feed and the formation of the final feed mixture for each day of the year, which is due to the software calculation of the optimal daily amount of feed.

Keywords: automation, automated control system, microcontroller, Data array, mathematical model.

ЗМІСТ

ВСТУП

Актуальність теми магістерської дипломної роботи. Жива речовина земної кулі становить 3 х 1012 т (суха маса). Проте 40 % органічної речовини і 70 % мінеральних речовин кормів тваринами не використовуються, а у вигляді добрива повертаються у ґрунт, де в результаті життєдіяльності мікроорганізмів підтримується його родючість. Із сільським господарством пов’язано безліч інших галузей діяльності людини: рослинництво, землеробство, тваринництво, ветеринарна медицина.

Годування сільськогосподарських тварин є одним з головних факторів, які зумовлюють їх продуктивність, оплату корму продукцією та економічну ефективність тваринництва. Вона включає в себе організоване, контрольоване і регульоване людиною живлення тварин.

В практичному аспекті годування сільськогосподарських тварин варто охарактеризувати як виробничий процес у тваринництві, пов’язаний з приготуванням корму для тварин за певним режимом і рекомендованою технікою.

Підвищення ефективності використання кормових ресурсів, виробництво високоякісних високопоживних кормів і правильне їх згодовування є актуальним питанням.

На сьогоднішній день норми годування для сільськогосподарських тварин розробляються у вигляді щоденної норми на кожен місяць року. Використовуючи таку систему з’являються випадки поганої адаптації тварин до нових режимів харчування, тому, шо в залежності від місяця, кількість необхідного корму може бути різною, або певний вид корму зовсім відсутній в наступному місяці. І в таких випадках перехід від однієї добової кількості корму до іншої дуже різкий.

Актуальним є необхідність розробки раціону для сільськогосподарських тварин з поступовою зміною кормових норм, для легшої адаптації, від місяця до місяця, зі щоденним корегуванням. Для запобігання випадків коли в одному місяці тварина отримує певну кількість корму, а з першим днем наступного, на багато більше або менше.

Окрім того, що сам процес годування, інколи, автоматизований, процедура комбінування різних типів кормів часто проводиться вручну. Це, в свою чергу, може призвести до неточності вимірів кількості кожного типу корму в суміші. Тому є необхідність розробки автоматизованої системи розрахунку кількості та комбінування різних видів кормів для різних видів тварин та автоматичного дозування кожного виду корму, з подальшим формування повноцінної кормової суміші.

Харчування тварин в сільському господарстві відіграє важливу роль в якості продуктів, отриманих від цих тварин. Розробка максимально ефективних раціонів харчування дозволить отримувати більш якісну продукцію і зменшити її собівартість.

Мета досліджень. Метою досліджень є підвищення ефективності процесу вигодовування сільськогосподарських тварин шляхом удосконалення принципів і методів побудови систем автоматизованого керування розрахунком та комбінацією кормів.

Завдання дослідження. Для досягнення поставленої мети в магістерській дипломній роботі необхідно вирішити такі задачі:

- розглянути методи побудови макетів для автоматизованих систем для розрахунку та комбінування кормів для сільськогосподарських тварин;
- розглянути алгоритм побудови масиву даних з інформацією про добові норми кормів на кожен місяць для всіх видів тварин;
- розробити математичну модель, для розрахунку добової кількості корму на кожен окремий день кожного місяця для всіх видів тварин;
- скласти макетну модель системи автоматизації розрахунку та обробки даних згідно математичної моделі та автоматичної подачі кожного виду корму з подальшим формуванням кормової суміші.

Об’єкт дослідження - автоматизовані системи керування процесом розрахунку добової норми кормів та комбінування різних типів кормів.

Предмет дослідження - методи, моделі та апаратно-програмні засоби для побудови автоматизованих систем керування процесом розрахунку добової норми кормів та комбінування різних типів кормів для вигодовуванням сільськогосподарських тварин.

Методи дослідження. Для вирішення поставлених задач застосовувались методи побудови автоматизованих систем керування процесом вигодовування тварин, математичного моделювання раціонів тварин з поступовою зміною добової кількості корму, методи електротехнічного моделювання при побудови макетної схеми автоматизації процесу вигодовування тварин.

Наукова новизна одержаних результатів

Запропоновано новий підхід для знаходження щоденного раціону кормів при вигодовуванні тварин, який дозволяє здійснити, по суті, неперервну поступову адаптацію кормової суміші на протязі року, а також метод автоматизованого керування цим процесом.

Практичне значення одержаних результатів полягає в тому, що використання запропонованої системи автоматизації процесу розрахунку та комбінування добової норми кормів для сільськогосподарських тварин, дозволить полегшити їх адаптацію до сезонних змін норм харчування, що покращить якість продукції та рентабельність всього виробництва.

Апробація результатів магістерської дипломної роботи. Основні положення і результати магістерської дипломної роботи доповідалися і обговорювалися на IV Міжнародної науково-практичній конференції «Мехатронні системи: інновації та інжиніринг», (Київ, 22 жовтня 2020 р.), а також публікувалися в журналі «ТЕХНОЛОГІЇ ТА ДИЗАЙН». №4 за 2020 рік.

РОЗДІЛ 1. ПОСТАНОВА ЗАДАЧІ МОДЕЛЮВАННЯ ОПТИМАЛЬНОГО ЩОДЕННОГО ПІДБОРУ РАЦІОНУ ПРИ ВИГОДОВУВАННІ СІЛЬСЬКОГОСПОДАРСЬКИХ ТВАРИН ТА АПАРАТНА ЧАСТИНА ДЛЯ СТВОРЕННЯ МОДЕЛІ АВТОМАТИЗОВАНОЇ СИСТЕМИ

1.1.Сучасні норми годування сільськогосподарських тварин

Упродовж життя тварини постійно витрачають енергію і поживні речовини на підтримання життєдіяльності та утворення продукції. Тому їх організм потребує безперервного відновлення цих витрат, які прийнято називати кормовими, а їх кількісне вираження є нормою годування.

Норма годування – це кількість поживних речовин таенергії, яка необхідна тварині для підтримання життя і утворення продукції. Годівля, що відповідає нормам, називається нормованою.

Норми годування тварин, які рекомендуються для практичного застосування, опубліковані в спеціальних довідниках, вважаються орієнтовними, оскільки їх необхідно коригувати відповідно до умов конкретних господарств залежно від віку, вгодованості та способу утримання тварин. Нормоване годування тварин передбачає визначення їх потреб більше ніж за 30 елементами живлення, зокрема в енергії, сухій речовині, протеїні, клітковині, мінеральних елементах і вітамінах.

За структурою раціону, як правило, визначають тип годування, назва якого залежить від домінуючих кормів, що використовуються [1].

Щоб скласти раціон, необхідно:

- встановити норму годування тварин (групи тварин) на основі даних про живу масу і рівень продуктивності та відкоригувати її;
- знати оптимальні даванки окремих кормів, які відповідають рекомендованій структурі і типу годування тварин даного виду (групи виробничого призначення);
- знати поживність використовуваних кормів.

При розробці раціонів стадо розділяють на групи, до котрих відбирають тварин, подібних за рівнем продуктивності, фізіологічним станом, живою масою тощо. Раціон складають для тварин кожної групи з розрахунку на одну голову. Організація нормованого годування, окрім визначення норм і складання раціонів, передбачає підготовку та послідовність згодовування кормів, кратність та спосіб їх роздавання. Прикладом помісячного розподілу шаблонів годування є норми годування молодняку овець.

Норми вигодовування молодняку овець визначають залежно від його віку, статі, інтенсивності росту, вовнової продуктивності (для вовнових і комбінованих порід) та породних особливостей.

У перший місяць після народження належний рівень вигодовування ягнят забезпечується переважно молоком вівцематки, а з другого місяця – додатково підгодівлею високоякісним сіном (бобовим або злаково-бобовим), сумішшю подрібненого ячменю та плющеного вівса або спеціальними комбікормами [2]. Починати згодовувати зазначені корми ягнятам слід із 7–10-денного віку (табл. 1.1).

Таблиця 1.1

Вигодовування ягнят на одну голову, г

Корм	Вік, міс.			
	1	2	3	4
Концентровані	40	100	150	250
Сіно	-	150-200	200-250	300-400
Силос	-	200-250	250-300	500-800

Залежно від віку (2, 3, 4 міс.), у весняно-літній період замість сіна й силосу ягнята повинні поїдати на пасовищі відповідно 0,9; 1,6 і 2,5 кг трави.

На цьому прикладі можна яскраво побачити неоднорідність раціонів при розрахунку щоденної норми на кожен місяць. Це породжує проблему адаптації організму тварини до нового раціону.

1.2.Модель щоденного підбору добового раціону.

1На сьогоднішній день більшість добових раціонів для сільськогосподарських тварин розробляється на певний проміжок часу. Градація раціону в таких випадках розглядається в форматах: тижні, квартали, місяці [3].

За такої системи зміна розміру щоденного раціону відбувається не поступово. Навіть, у випадку зміни раціону тварин кожні декілька тижнів, це не виключає ситуацію, коли тварині в один день дають одну дозу кормів, а на наступний день зовсім іншу.

Особливо чітко таку ситуацію можна спостерігати на сільськогосподарських підприємствах, де харчування тварин розробляється на півроку – рік. В такій ситуації зазвичай розглядають добовий раціон на кожен місяць року.

В результаті ми маємо ситуацію, коли добові норми харчування тварин відрізняються в залежності від місяця. У такому випадку перед підприємством постає проблема, коли в перший день нового місяця тварині, згідно розробленого плану, потрібно дати суміш кормів, розмір і склад котрої може різко відрізнятися, від того, яку суміш тварина отримала в попередній день.

В таблиці 1.2 можна побачити різницю між етапами годування кролів [1].

Таблиця 1.2

Щоденний раціон кролів

Період	Тип корму			
	Сіно	Соковиті корми	Концентровані корми	Сіль, мінеральні добавки
Відкорм	150 г	500 г	80 г	1 г
Утримання	150 г	150 г	40 г	1 г
Підготовка до розмноження	150 г	200 г	55 г	1 г

Самки в період вагітності	175 г	200 г	60 г	1 г

Як можна побачити, різниця в подачі концентрованих кормів залежно від етапів вирощування, суттєво відрізняється. Для тварини різкий перехід від одного режиму харчування до іншого може бути шкідливим. Це може призвести до зниження продуктивності і, як наслідок до зниження ефективності господарства[4].

Окрім зміни кількості певних компонентів раціону, деякі його елементи можуть зникати зовсім (табл. 3).

Таблиця 1.3

Сезонні раціони курей

Літо		Зима	
Корм	Добова норма	Корм	Добова норма
Зерно	50 г	Зерно	100 г
Мішанка	45 г	Мішанка	50 г
Дріжджі	4 г	Варена картопля	100 г
Кісткове борошно	2 г	Коренеплоди	50 г
–	–	Кисломолочний сир	50 г
–	–	Висушена кропива	10 г
–	–	Висівки	10 г
–	–	Крейда	3 г

В наведеному прикладі загального сезонного раціону курей можна побачити, що з початком літа певні елементи зникають із їх раціону. Це породжує проблему з тим, коли саме вилучати з раціону той чи інший компонент. Так само варто пам’ятати, що різка зміна раціону може негативно позначитися на загальному сані та продуктивності тварин.

У випадку курей різка зміна раціону може призвести до того, що кури почнуть менше нестися. Це в свою чергу може призвести до фінансових втрат підприємства, особливо коли річ йде про значні підприємства, що орієнтовані на конкретний вид продукції птахівництва [5].

Для запобігання таких ситуацій слід розробити алгоритм розрахунку поступової зміни раціону із щоденним корегуванням. Окрім того необхідна система автоматичного комбінування різних груп кормів і формування єдиної кормової бази на кожен день.

1.3. Розробка автоматизованої моделі щоденного підбору добового раціону кормів за допомогою мікропроцесорної платформи Arduino.

При вирішенні будь-якої виробничої задачі, необхідно розробити тестову модель механізму, який покаже апаратне виконання того чи іншого варіанту рішення проблеми. Вона дозволить побачити чи буде працювати розроблений алгоритм і якого результату можна досягти.

Для підбору щоденного раціону, для декількох видів тварин, потрібно розробити макет системи автоматичного розрахунку добової кількості кормів та автоматичного комбінування цих кормів.

Для створення тестової моделі необхідний ряд елементів. Серед основних можна виділити: систему вводу/виводу інформації, групу датчиків, виконавчий механізм та керуючий пристрій [6]. Найбільш складним та важливим можна вважати саме керуючий пристрій, що і буде здійснювати управління всією схемою. Зазвичай, керуючий пристрій – це логічний контролер, який програмується за допомогою певних мов програмування. Для вирішення

поставленої задачі було обрано мікропроцесорну плату Arduino.(Рис. 1.1) Arduino – апаратно обчислювальна платформа для конструювання.

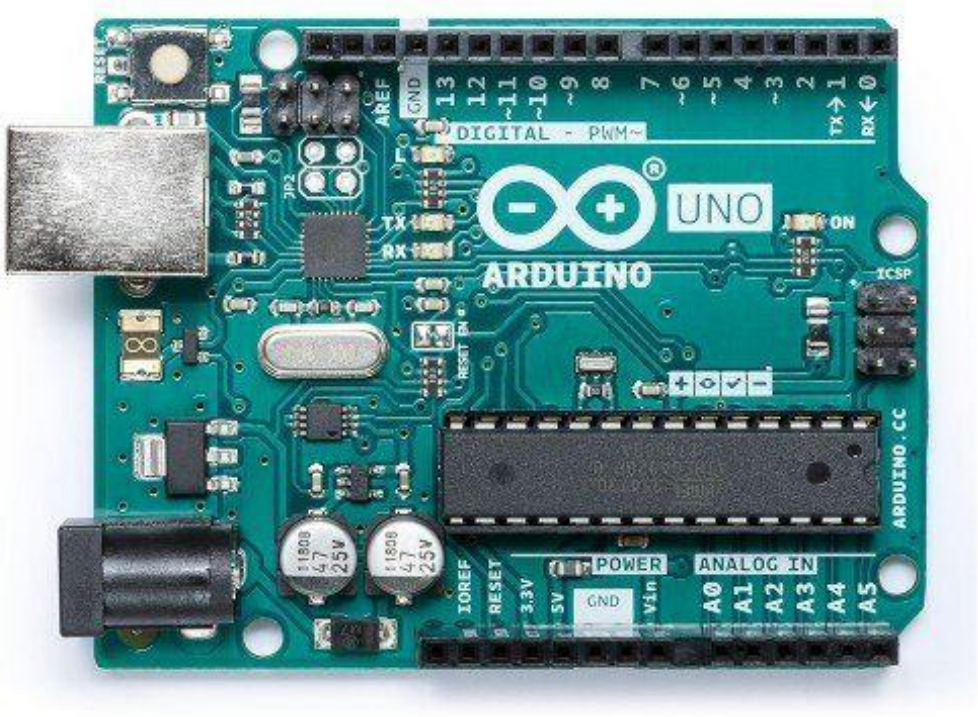

Рис. 1.1 Процесорна плата Arduino UNO

Ця платформа буде використана для побудови макетної, не промислової, схеми. Її завданням буде перевірити можливість реалізації розробленого підходу і показати як саме буде працювати запропонована система автоматизації [7].

Платформа складається з апаратних і програмних частин. Обидві вони надзвичайно гнучкі і прості у використанні. У програмуванні використовується спрощена версія C++, також відома як Wiring. Розробка може бути виконана як за допомогою вільного середовища Arduino IDE (Рис.2), так і за допомогою мови графічного програмування XOD IDE або довільного інструментарію C/C++.

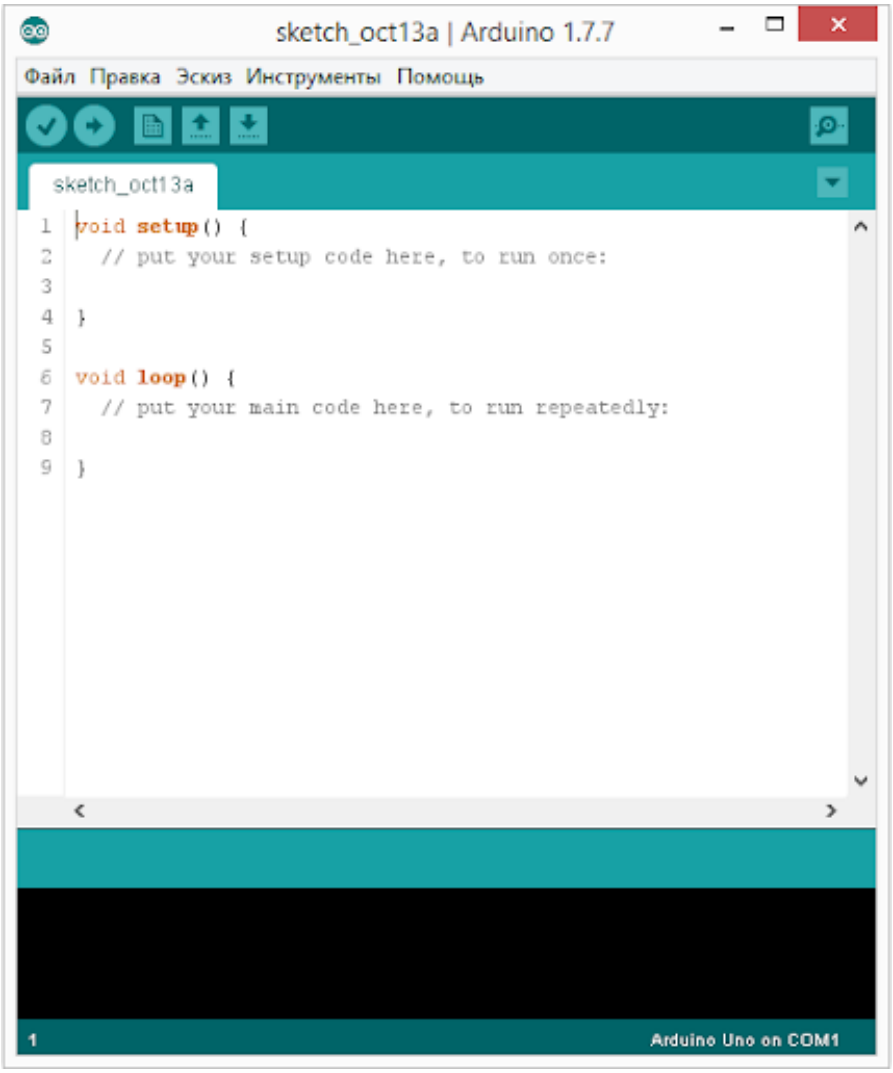

Рис. 1.2 Вікно інтегрованого середовища розробки Arduino IDE

Arduino Uno може живити як USB-з'єднання, так і зовнішнє джерело: акумулятор або мережевий адаптер. Джерело визначається автоматично.

Платформа може працювати з джерелом напруги від 5 до 20 В. Однак при напрузі менше 7В робота може бути нестійкою і напруга більше 12 В може призвести до перегріву і пошкоджень. Таким чином рекомендований діапазон від 7 до 12 В (оптимально 9В)[8].

У Arduino доступні наступні контакти для доступу до живлення:

- Vin забезпечує ту ж напругу, яка використовується для живлення платформи. При підключенні через USB напруга становить 5В.
- 5V забезпечує 5В незалежно від вхідної напруги. За такої напруги функціонує процесор. Максимально допустимий струм від цього контакту становить 800 мА.

- 3.3V забезпечує 3.3 В. Максимально допустимий струм від цього контакту становить 50 мА.
- GND - це земля.

Особливості живлення плат Arduino дозволяють використовувати їх як автономно, так і з підключенням по USB-кабелю.

Введення-виведення інформації.

На платформі є 14 контактів, які можна використовувати для цифрового введення та виведення даних. Роль, що відіграє кожен контакт залежить від програми. Всі вони працюють при напрузі 5 В і розраховані на струм до 40 мА. Крім того кожен контакт має вбудований, але вимкнений за замовченням, резистор від 20 до 50 кОм. Деякі контакти мають додаткові ролі:

- *Serial*: 0-й і 1-й. Використовується для даних USB.
- Зовнішнє переривання: 2-е і 3-е. Ці контакти можна налаштувати, щоб ініціалізувати виклик даної функції, коли вхід змінюється вхідний сигнал.
- PWM: 3-й, 5-й, 6-й, 9-й, 10-й і 11-й.
- Світлодіод: 13-й. Якщо 5В підведено до контакту, світлодіод світиться і при нулі гасне.

На додаток до цифрових контактів введення-виведення, Arduino має 6 аналогових контактів. Крім того, на платі є вхідний контакт *Reset*. Його установка в логічному нулі призводить до скидання процесора. Це аналогічно звичайній кнопці скидання комп'ютера.

1.4.Пам'ять в Arduino.

Платформа оснащена флеш-пам'яттю 32 КБ, 2 КБ котрої зарезервовано для так званого bootloader. Він дозволяє прошивати Arduino зі звичайного комп'ютера через USB. Ця пам'ять є постійною і не призначена для зміни доки пристрій працює. Її метою є зберігання програми та пов'язаних статичних ресурсів.

Існує також 2KB SRAM-пам’яті, що використовуються для зберігання тимчасових даних, наприклад змінних програм. По суті це оперативна пам'ять платформи. Пам'ять SRAM очищається при знеструмленні.

Існує також 1КБ пам'яті EEPROM для тривалого зберігання. Це еквівалент жорсткого диска для Arduino.

EEPROM – це пам'ять, до якої ми маємо повний доступ з поточної програми. Тобто ми можемо читати і записувати дані там під час запуску, саме ці дані не скидаються при перезапуску контролеру [9].

Для реалізації системи підпору раціону використання енергонезалежної пам’яті є ключовим. Це зумовлено тим, що для вдалого розрахунку нам необхідно мати масив даних з інформацією про тварин та добові норми раціонів на кожен місяць.

Також варто звернути увагу на те, що в процесі може з’явитися необхідність редагування даних масиву. Зазвичай для цього необхідно заново перепрошивати контролер. Оскільки в умовах підприємства зробити це буває складно, то слід використати енергонезалежну пам’ять. Вона дозволить зберегти змінені дані, не змінюючи основний скєтч програми, і в подальшому їх використовувати.

Варіанті застосування енергонезалежної пам’яті

1. Зберігання налаштувань, які змінюються «з меню» пристрою, без перепрошивки;
2. Калібрування, зберігання відкаліброваних даних;
3. Використовувати як додаткову пам'ять SRAM при нестачі пам'яті;

EEPROM – це область пам'яті, що складається з елементарних чарунок розміром в один байт (на кшталт SRAM). Обсяг EEPROM відрізняється в різних моделях МК (Табл. 1.4).

Табл. 1.4

Обсяг пам’яті для різних чіпів.

	ATmega168	ATmega328	ATmega1280	ATmega2560
Flash (1 кБ flash-Пам'яті зайнятий завантажувачем)	16 Кбайт	32 Кбайт	128 Кбайт	256 Кбайт
SRAM	1 Кбайт	2 Кбайт	8 Кбайт	8 Кбайт
EEPROM	512 байт	1024 байта	4 Кбайт	4 Кбайт

Маючи таку задачу слід звернути увагу на можливості керуючої плати Arduino та підключення до неї додаткових пристроїв для виконання необхідних маніпуляцій.

Важливим питанням при розробці раціону з поступовою зміною щоденної норми кормів є зберігання масивів з інформацію про щоденні норми раціонів на кожен місяць для всіх груп тварин. Доцільно для цього буде використати енергонезалежну пам’ять (EEPROM).

Не зважаючи на те, що це одна з найпростіших варіацій модульних плат Arduino, вона є досить функціональною для даної задачі[10].

Arduino UNO обладнана процесором ATmega328, що означає наявність 1024 байта енергонезалежної пам’яті. Це дає нам можливість записувати достатні масиви даних для розрахунку приблизно 3-4 видів тварин, записавши данні добових норм кормів по всім місяцям.

Швидкість роботи EEPROM:

- Запис одного байта займає ≈ 3,3 мс (мілісекунд)
- Читання одного байта займає ≈ 0,4 мкс (мікросекунди)

За рахунок високої швидкості запису і считування, навіть при повному заповненні енергонезалежної пам’яті, система буде працювати досить комфортно для користувача, що спростить керування всім процесом комбінування кормів.

1.5.Середа програмування Arduino IDE

IDE (інтегроване середовище розробки) – це програма або група програм (середовище), призначених для створення, настроювання, тестування та підтримки програмного забезпечення. Інтегроване середовище розробки характеризується складною функціональністю, включаючи редагування та компіляцію вихідного коду, створення програмного забезпечення, створення бази даних тощо.

В рамках проекту Arduino було створено програмне забезпечення, яке відповідає основним вимогам типового середовища IDE [10]. Це не потужне програмне забезпечення, а проста та функціональна програма, яка дозволяє нам писати, компілювати і завантажувати програму на мікроконтролер.

Проста структура Arduino IDE є перевагою, оскільки забезпечує швидке освоєння програми і перехід до розробки додатків для Arduino. Незважаючи на свою простоту та інтуїтивно зрозуміле управління, варто звернути увагу на найважливіші елементи програми. За допомогою меню програми ви можете керувати проектом, наприклад, створювати новий проект, зберегти поточний і друкувати вихідний код на принтері.

Цікавою особливістю програми є вбудований набір прикладів програм. Це дуже зручно, так як приклади програм можна відразу перевірити, завантаживши їх на мікроконтролер. При необхідності ви можете зберегти приклад і змінити його відповідно до ваших потреб.

Корисним елементом IDE є меню «Інструменти», котре включає в себе функції автоматичного форматування коду, архівування проекту, включення послідовного монітора порту (USB в Arduino розглядається як звичайний послідовний порт).

Найважливішим елементом меню Інструменти є можливість вибору відповідної плати, тобто вашої системи Arduino, підключеної до вашого комп'ютера. Всі офіційні версії Arduino є в списку. Якщо типу плати немає у списку, його можна додати, змінивши один із файлів програми.

У меню Інструменти ви також можете встановити порт, до якого підключено плату Arduino. Пакет Arduino IDE визначає сам порт, але іноді вам потрібно вручну встановити номер порту в налаштуваннях[12].

За допомогою Arduino IDE також можна завантажити, тобто програмувати, Bootloader для нового, чистого мікроконтролера Atmega, що дозволяє клонувати чіпи або просто замінити несправний мікроконтролер в Arduino.

Для нормальної роботи з Arduino IDE ви використовуєте панель швидкого доступу, яка має найважливіші кнопки. Це рішення, що полегшує роботу з пакетом IDE, дає нам прямий доступ практично до всіх необхідних параметрів при написанні та тестуванні програми. (Рис. 3)

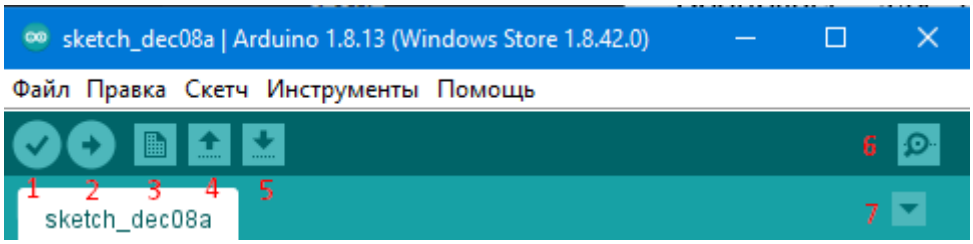

Рис. 3 Панель швидкого доступу програми Arduino IDE

Вони дозволяють:

1. Компіляцію програми;
2. Завантажити програму на мікроконтролер (перед прошивкою код компілюється);
3. Почати роботу над новим проектом;
4. Відкрити існуючий проект;
5. Зберегти проект на диску;
6. Увімкнути монітор послідовного порту;
7. Меню керування вкладками.

Усі параметри на панелі швидкого доступу дублюються в меню програми.

Додатковим корисним елементом під кнопкою послідовного монітора портів є меню керування вкладками. Вкладки в Arduino IDE полегшують написання складних проектів, а також дозволяють працювати з декількома проектами одночасно.

Найбільша частина вікна програми призначена для написання самого програмного коду. Редактор Arduino IDE не дуже просунутий, але має найважливіші елементи, щоб полегшити написання простих програм. До таких елементів відноситься освітлення синтаксису і блоків. Це не так багато, але цього досить для простих проектів.

Останнім елементом програми є вікно повідомлення та стану. Інформація, що відображається там, дозволяє користувачеві знайти помилки в коді і отримати підтвердження завершення компіляції і завантаження програми на мікроконтролер.

Загалом Arduino IDE - це простий програмний пакет, який дозволяє програмувати будь-яку відому плату Arduino, спілкуватися з послідовним портом і легко керувати проектами [13].

1.6.Підключення до Arduino UNO датчиків та індикаторів.

Для контролю процесу автоматизації, в якому присутні плати Arduino UNO, використовують датчики та індикатори. Плати Arduino UNO мають 14 (0-13) входів та виходів, щоб їх підключати.

Прикладом пристроїв для виведення даних можуть бути:

1. Світлодіод
2. Lcd-екран
3. Різноманітні пульти
4. Різні види двигунів

Всі ці пристрої можна під’єднати до нашої плати та запрограмувати на виконання певних функцій. Для прикладу, світлодіод зазвичай використовують для світлової індикації і контролю певних процесів.

Підключення додаткової індикації дозволяє краще контролювати ті процеси, що були ініціалізовані користувачем. За допомогою підключення екрану можна виводити текст для полегшення спілкування користувача та написаної програми.

Серед пристроїв виведення варто звернути увагу на двигуни. Вони надають можливість, за допомогою програми на керуючій платі, маніпулювати зовнішніми процесами.

Як приклад можна привести приєднання серводвигуна до механізму відкриття заслінок в певному резервуарі. Це дозволить відкривати резервуар не наближаючись до нього.

Також варто пам’ятати, що пристрої для введення та виведення інформації, виконавчо, тісно пов’язані між собою. Так на екран під’єднаний до плати можна вивести інформацію, що була отримана з певного датчика. Водночас двигун може починати роботу при спрацюванні, скажімо, фоторезистора. Така взаємодія і лежить в основі проектування схем автоматизації.

Найсучаснішим варіантом розробки прототипів є безпайкова монтажна плата (Рис. 1.4,1.5), яка має безсумнівні переваги:

- Можливість проводити налагодження велику кількості разів, зміна модифікації схем і способів підключення пристроїв;
- Можливість з'єднати кілька плат в одну велику, що дозволяє працювати з більш складними і більшими проектами;
- Простота і швидкість проектування;
- Довговічність і надійність.

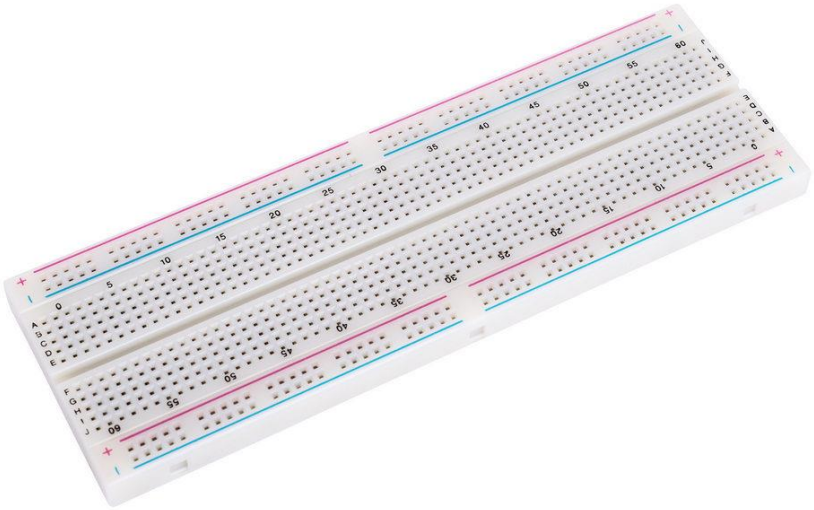

Рис. 1.4 Макетна плата для монтаже без пайки

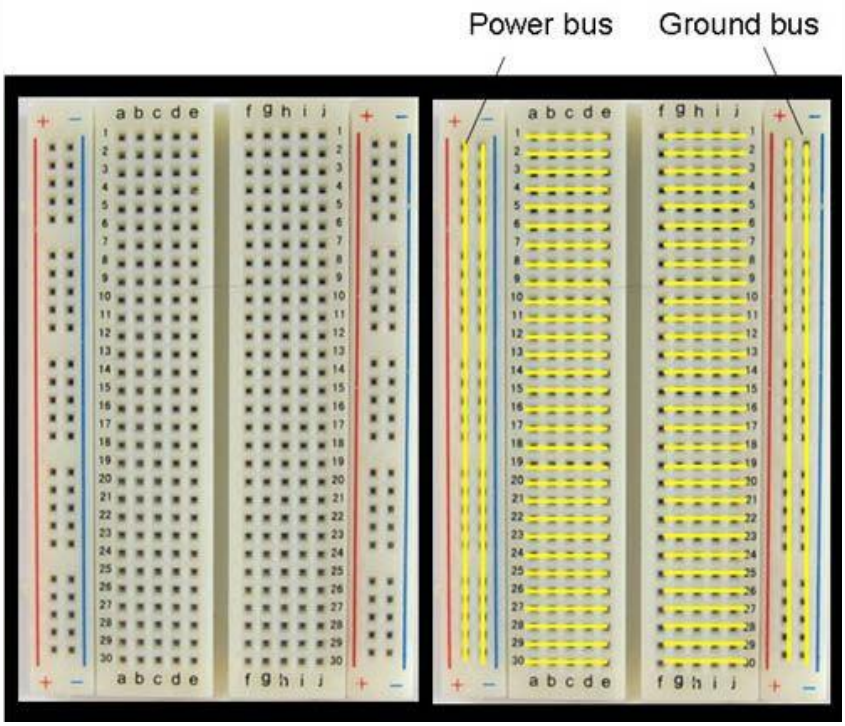

Рис. 1.5 Схема макетної плати

Макетна плата має пластикову основу з безліччю отворів (стандартна відстань між ними становить 2,54 мм). Всередині конструкції знаходяться ряди металевих пластин. Кожна пластина має кліпси, які заховані в пластиковій частині установки.

Саме в ці кліпси вмикають дроти. Коли провідник підключається до одного з окремих отворів, контакт одночасно підключається до всіх інших контактів окремого ряду. Тому, підключивши контакти інших пристроїв до інших затискачів, пов'язуємо їх з провідником – рейкою з кліпсами.

Звичайно, є деякі недоліки цього варіанту монтажу:

- У реальних проектах зв'язки на дошці будуть не такими надійними, як при пайці. Будь-яка вібрація повільно послабить контакти і це обов'язково в кінцевому підсумку призведе до несподіваних проблем. Тому в реальних проектах використовуються інші види установки елементів.
- Зовнішній вигляд проектів з локшиною у вигляді проводів над безмежними білими просторами дошки не можна назвати професійним і естетичним. Хоча цей вид завжди зачаровує глядачів і формує образ проекту чогось "жахливо складного, колись так багато проводів".
- Плата з таким видом монтажу завжди займе більше місця за рахунок нависання проводів. Отже, йому потрібен великий корпус з фіксацією і захистом від вібрації.
- Вартість макетної плати. Хоча плати не є дорогими пристроями, вам все одно потрібно буде придбати їх на додаток до мікроконтролера та інших елементів. На щастя, сьогодні на ринку є велика кількість недорогих варіантів і готових наборів з монтажною платою в комплекті.

Незважаючи на деякі недоліки, альтернативи по простоті і доступності для монтажу перших схем практично немає. Сьогодні можна знайти величезну кількість проектів, в яких всі елементи розміщені на макетній платі. Майже всі приклади з підручників з основ робототехніки і Arduino використовують цей варіант установки.

Залежно від особливостей, найпоширенішими видами є:

- Для складання великих мікросхем в основному використовуються безпайкові плати на 830 або 400 отворів.
- З наявністю канавок для зчеплення плат, які дозволяють реалізовувати досить великі проекти;
- При наявності самоклеючої стрічки на корпусі для надійної фіксації на приладі;
- З наявністю маркування на платі для підключення пристроїв.

Залежно від вартості і виробника, в упаковку можуть входити додаткові аксесуари – дротяні перемички, різноманітні роз'єми. Але головним критерієм якості завжди є кількість контактних роз'ємів і їх технічні характеристики.

Висновки до розділу 1

1. В першому розділі була проаналізована проблема нерівномірної системи годування сільськогосподарських тварин. Зокрема проблема розподілу добових раціонів тварин по місяцям. Внаслідок чого в певний момент на, етапі вигодовування тварини, може відбутися різка зміна добового раціон. Особливо, ця проблема спостерігається у випадках, коли на певному етапі вигодовування той чи інший компонент раціону вилучається. В таких випадках це може негативно позначитися на здоров'ї та розвитку тварини. Також це може призвести до погіршення продуктивності тварин.
2. Були проаналізовані можливості плати Arduino UNO, варіанти підключень додаткових пристроїв введення та виведення інформації та характеристики макетної плати для розміщення на ній необхідних додаткових пристроїв.
3. Розглянуті основні функції інтегрованого середовища розробки Arduino IDE. Можливість зручного написання програмного коду для проведення необхідних розрахунків, обумовлених поставленим завданням.
4. Проаналізовані доступні типи пам'яті та їх призначення та розмір. Для зберігання масиву з інформацією про нормі кормових раціонів на кожен місяць для всіх видів тварин, доцільно буде використовувати енергонезалежну пам'ять (EEPROM).

Розділ 2. Інформаційне, математичне та алгоритмічне забезпечення поставленої задачі.

2.1.Масив даних та внесення змін до нього

Розробка системи підбору оптимального раціону, потребує інформації, на основі якої буде створюватися наша автоматизована система. А саме інформації про щоденні норми раціонів на кожен місяць для всіх груп тварин. Для збереження цієї інформації використаємо масив даних.

Масив – це впорядкований набір однотипних змінних, що мають спільне ім’я, доступ до яких здійснюється через їх індекс. У мові програмування С, на якому заснований Arduino, масиви можуть бути досить складними[14]. Але використання простих масивів не таїть в собі особливих труднощів.

Використання масивів спрощує зберігання та обробку даних. Для розробки алгоритму обробки даних масиву, потрібно формалізувати постановку завдання.

Вважаємо, що раціон підбираємо для видів тварин, які пронумеровані числами 1,2,3…k.

Для формування нашої інформації використаємо тривимірний масив даних. Перший індекс масиву має відповідати за вид тварини, другий за номер місяця, третій за тип корму. Індексація в масивах починається з нуля. Тобто, перший елемент масиву матиме порядковий номер 0. Відповідно, значення 1-го корму для першого виду тварин на перший місяць, буде записано з індексом 0.0.0. Так само кількість другого типу корму для того ж виду тварин на перший місяць матиме індекс 0.0.1. Для прикладу кількість 3го типу корму для 2-го виду тварин на 7-й місяць, записується під індексом 1.6.2.

До такого масиву будуть входити дані про раціони для кожного виду тварин. Як приклад, для розрахунку раціону для трьох видів тварин ми матимемо три пов’язані між собою таблиці з даними про раціони. В кожну таку таблицю будуть внесенні добові норми на кожен місяць року і для різних груп тварин вони

будуть окремі. Корми для кожного виду тварин представлені також у формалізованому вигляді.

Для зберігання, обробки та внесення змін до даного масиву розроблено алгоритм з використанням масиву і зберігання змін до нього в енергонезалежній пам’яті пристрою[15].

Згідно цього алгоритму, в першу чергу на етапі написання коду програми, розробляється шаблонний масив з необхідною інформацією. До такого шаблону вносяться дані про щоденний раціон для всіх видів тварин на кожен місяць і готовий шаблон записується в основний код програми. Всі подальші розрахунки будуть проводитись з даним шаблоном.

Однак у випадку, коли користувач забажає змінити заданий шаблон, потрібно передбачити таку можливість, не прибігаючи до перепрошивки всієї системи. Для цього розроблено додаткове меню, яке дозволяє ввести зміни до активного шаблону. Блок-схема процесу внесення змін до активного шаблону раціонів представлена на рис. 2.1.

Згідно цієї схеми для зміни значень в масиві раціонів користувач повинен обрати конкретне значення, яке бажає змінити. Для цього йому пропонується ввести конкретну цифру, що відповідає номеру тварини, потім, відповідно, цифри місяця та певного корму. Після цього на екрані відображається поточне значення обраного елементу масиву. Далі користувач може ввести нове значення, для цього компоненту масиву.

Загалом користувач може змінити значення кожного елементу масиву в залежності від його потреб. Таким чином можна корегувати весь масив з інформацією про раціони. Внесення змін відбувається в поточний масив, на основі якого в подальшому будуть проводитись розрахунки оптимальної порцій на для певного дня.

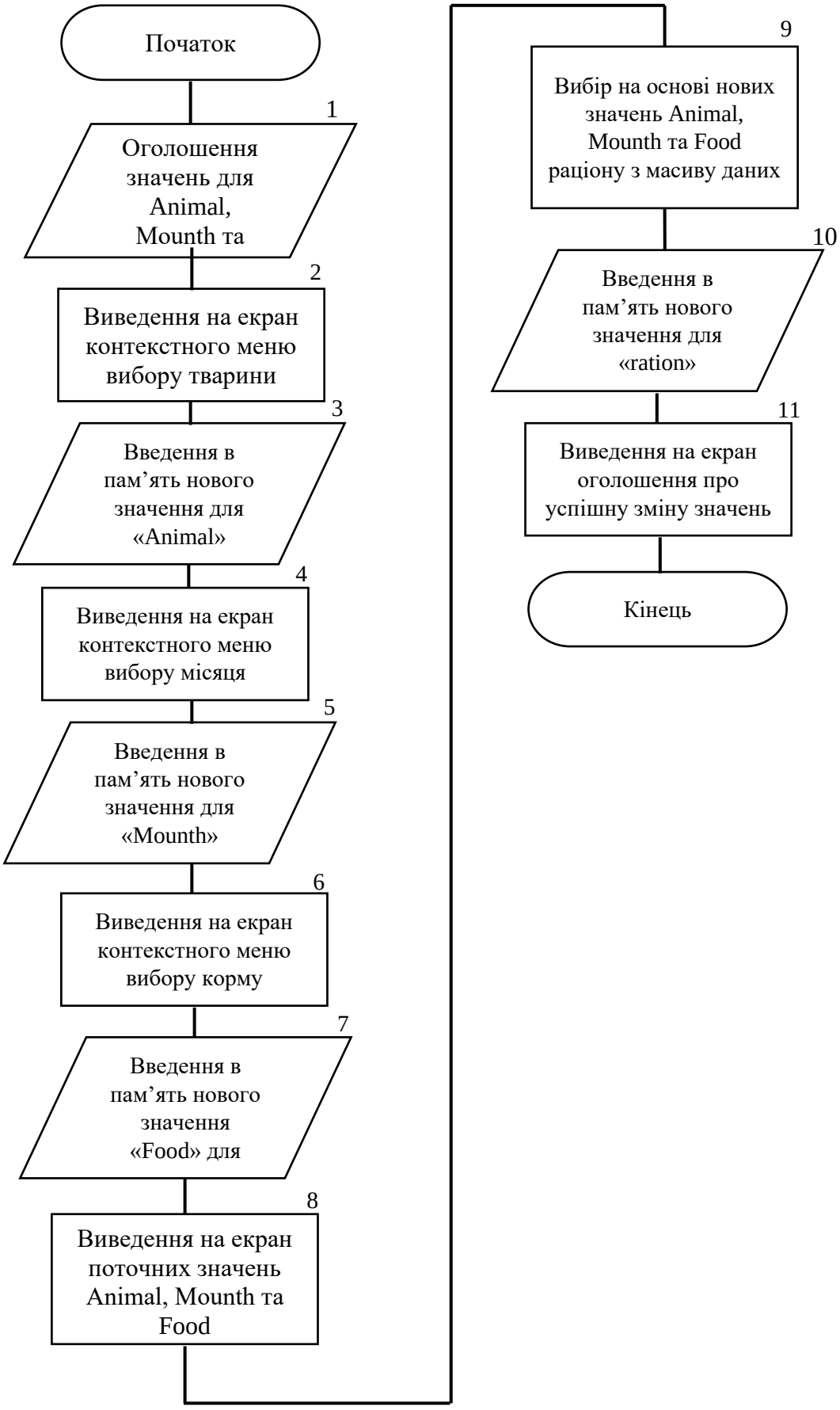

Рис. 2.1 Блок-схема процесу внесення змін до активного шаблону раціонів.

4.1.Збереження масиву даних в енергонезалежній пам’яті

Можливість зміни шаблону з інформацією про добові раціони без переписування початкового коду досить важлива. Але це породжує проблему збереження змінених даних. Справа в тому, що при вимкненні струму та повторному старті системи, внесені зміни не збережуться.

Після повторного старту системи вона почне роботу з шаблоном, який було запрограмовано при написанні коду. Щоб не вносити зміни повторно, змінену версію шаблону потрібно завантажити на енергонезалежну пам’ять. Можливості вбудованої бібліотеки «EEPROM.h», яка підключається простою командою можна зберігати не лише конкретні елементи а й масиви з даними, що значно полегшує подальшу обробку [16].

Для запису інформації на енергонезалежну пам’ять використовують команди:

- eeprom_write_byte**(**адреса, значення**)**
- eeprom_write_word**(**адреса, значення**)**
- eeprom_write_dword**(**адреса, значення**)**
- eeprom_write_float**(**адреса, значення**)**
- eeprom_write_block**(**адреса в SRAM, адреса в EEPROM, розмір**)** – записує інформацію за адресою в SRAM на адресу в EEPROM.

Для повернення інформації з енергонезалежної пам’яті використовують команди:

- eeprom_read_byte**(**адреса**)** - поверне значення
- eeprom_read_word**(**адреса**)** - поверне значення
- eeprom_read_dword**(**адреса**)** - поверне значення
- eeprom_read_float**(**адреса**)** - поверне значення
- eeprom_read_block**(**адреса в SRAM, адреса в EEPROM, розмір**)** - прочитає вміст за адресою в EEPROM в адресу в SRAM

Команда «eeprom_write_block» дозволяє нам записати наш шаблонний масив даних до енергонезалежної пам’яті після внесення змін до нього. Це дозволить використовувати вже змінений шаблон при нових розрахунках навіть після повторного вмикання системи.

У бібліотеці «avr/eeprom.h» є ще один дуже корисний інструмент – «EEMEM», він дозволяє автоматично адресувати дані, створюючи покажчики, значення яких призначать компілятор [17].

Для зберігання масивів особливо зручно використовувати цей інструмент адже він дозволяє автоматично визначити адресу і дає можливість повернути значення всього масиву одразу.

Розроблена програма дозволяє зберегти поточний шаблон (той з яким користувач працює в даний момент), не зважаючи на те, чи були внесені в нього зміни чи ні. Вибравши відповідний пункт в основному меню, поточний шаблон записується на енергонезалежну пам’ять.

2.2.Считування масиву з енергонезалежної пам’яті

Для того, щоб почати роботу зі збереженим масивом даних, потрібно його вивантажити в поточну програму. Процес має декілька етапів:

1. Считування збереженого масиву даних, що знаходиться на енергонезалежній пам’яті.
2. Запис прочитаного масиву до тимчасового.
3. Заміна значень шаблонного масиву, значеннями тимчасового.

Програма розроблена таким чином, що вона проводить розрахунки, беручи данні з шаблонного масиву. Це означає, що для роботи зі зміненим і збереженим масивом нам необхідно замінити існуючий шаблонний масив, тобто той який зберігається на енергонезалежній пам’яті.

Після заміни масиву, програма буде оперувати даними, які були внесені до масиву останнього разу. Використання шаблону, також дозволить не втрачати

базову інформацію, яку було введено на етапі програмування і корегувати зміни, що були внесені.

За замовченням, програма починає свою роботу із шаблонним масивом, який записано в програмі. Але в програмі передбачено процес, який дозволяє одразу почати роботу зі збереженим масивом (Рис. 2.2,2.3,2.4).

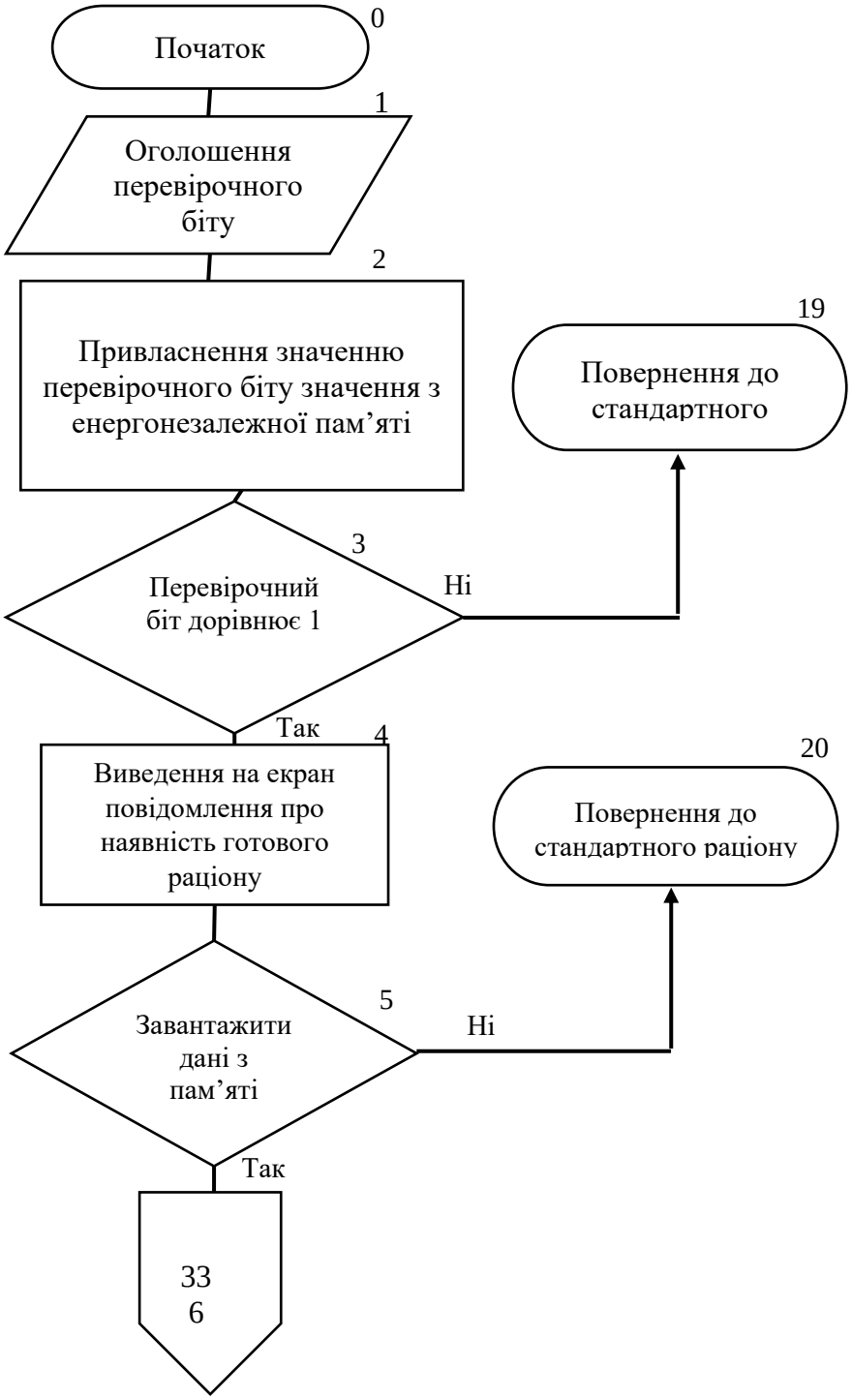

Рис. 2.2 Блок схема перевірки наявності збереженого масиву

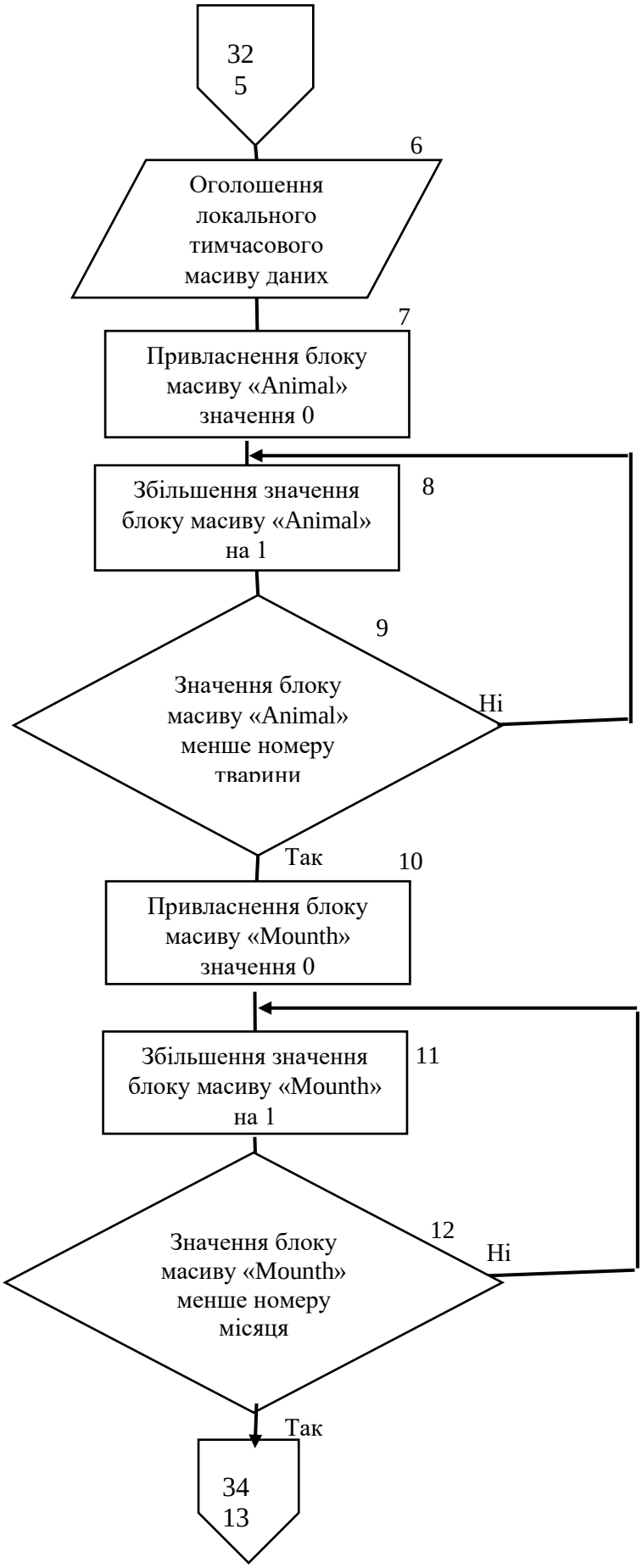

Рис. 2.3 Продовження блок-схеми 2.3

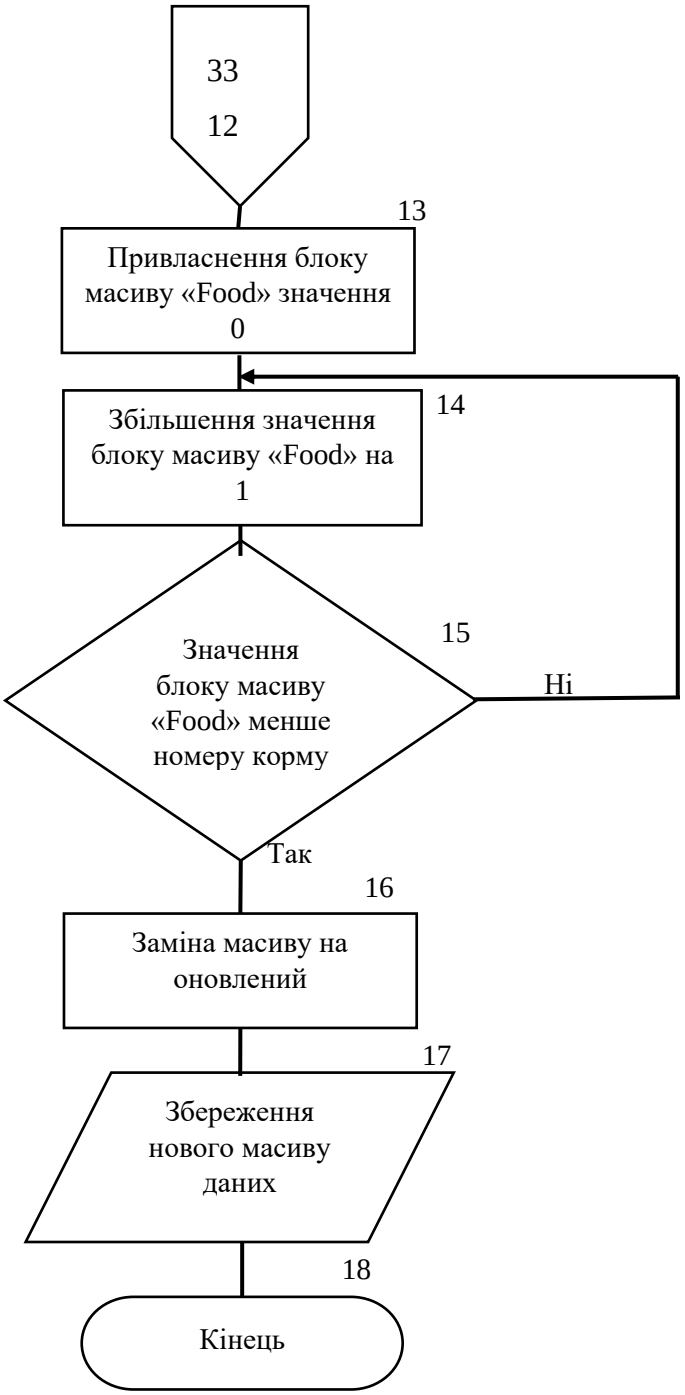

Рис. 2.4 Продовження блок-схеми 2.4

На блок-схемі видно, що система перевіряє наявність збереженого масиву в енергонезалежній пам’яті. Перевірка відбувається шляхом запиту до «перевірочного» байту. Цей байт заповнюється нашою системою на етапі збереження масиву в енергонезалежній пам’яті. До алгоритму збереження масиву було додано цей процес за для зручності перевірки та продовження роботи після перезавантаження.

Процес перевірки розпочинається одразу під час старту системи. При наявності перевірочного байта програма запропонує завантажити збережений масив та продовжити роботу з ним. Якщо користувач відмовиться від завантаження збереженого масиву або він не буде виявлений, система автоматично перейде в основне меню та продовжить роботу з існуючим шаблоном.

Також користувач може спочатку відмовитись від завантаження збереженого масиву і провести розрахунки на основі базового шаблону. Потім можна завантажити збережений масив за допомогою основного меню і провести повторний розрахунок. Це дозволить порівняти данні і зручніше їх корегувати.

4.3.Внесення даних до масиву

На платах Arduino розташований конвертер USB-TTL, що дозволяє мікроконтролеру в текстовому режимі «консоль» спілкуватися з комп’ютером на послідовному інтерфейсі Serial.

Комп’ютер створює віртуальний COM-порт, до якого можна підключитись і отримувати-надсилати текстові дані. Прошивка завантажується через той самий порт [19].

Сам Arduino IDE також має вбудовану «консоль» – портовий монітор, представлений як «кнопка з лупою» у верхньому правому куті програми. Натиснувши на цю кнопку ми відкриємо сам монітор порту (Рис. 2.5).

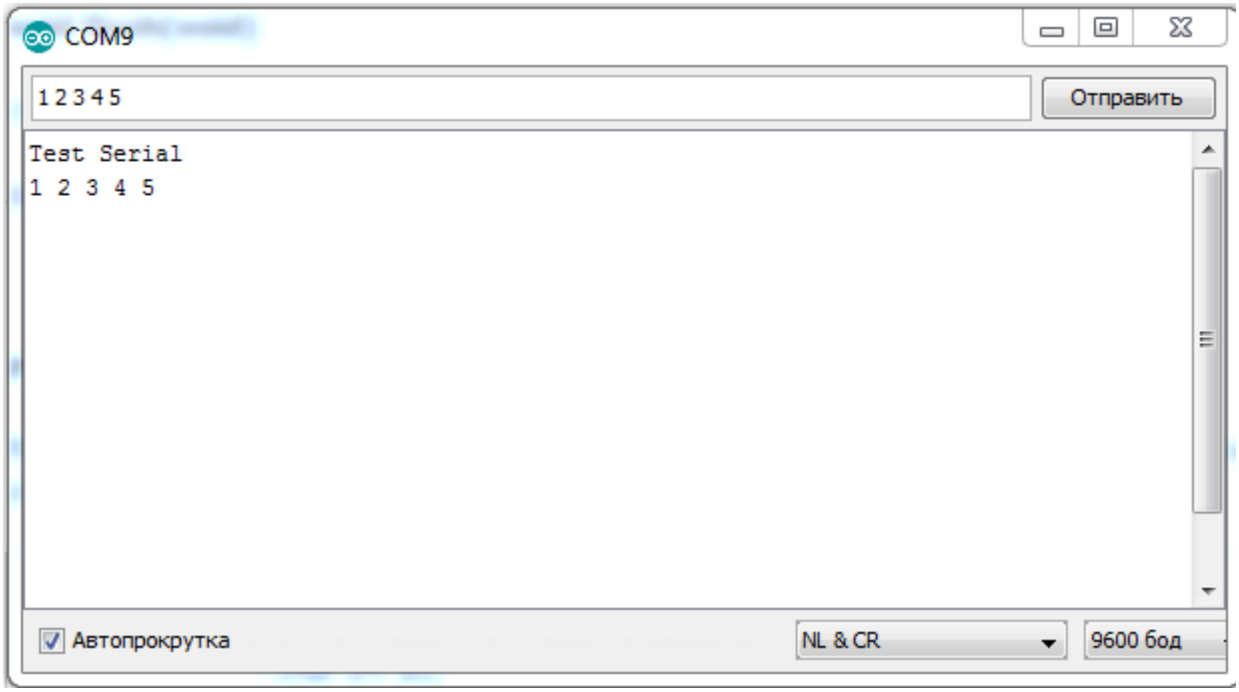

Рис. 2.5 Виведення інформації з віртуального COM-порту

Для початку роботи з монітором COM-порту необхідно вказати швидкість з якою відбуватиметься обмін даними. Спілкування по Serial можна робити з різною швидкістю, виміряною в бод (baud), а якщо швидкість прийому і відправки не збігаються – дані будуть отримані неправильно. Для Arduino Uno оптимальна швидкість це 9600.

Для запуску зв'язку по Serial, використовують команду «Serial.begin(**speed**)», де **speed – швидкість в** baud [20]. В програмному коді це буде записано так (Рис. 2.6):

```
void setup()
{
  Serial.begin(9600);
}
```

Рис. 2.6 Запуск зв'язку по Serial на швидкості 9600

Для виводу інформації на екран використовується команда «Serial.print()», або «Serial.println()». За допомогою цих команд можна виводити текстові повідомлення, числа або значення змінних. Це дозволить відображати для

користувача поточні значення, з якими працює програма, та спілкуватись з користувачем.

Для керування процесом розрахунку користувач повинен мати можливість вводити певні команди. Цю можливість забезпечують такі функції, як: «Serial.parseInt()», «Serial.read()», і «Serial.available()» [21].

Ці команди дозволяють отримувати інформацію, яку вводить користувач на клавіатурі. Програма розроблена таким чином, щоб користувач вводив виключно числові команди. Наприклад, для при зміні значень в масиві, користувача запитують данні якої тварини він бажає змінити. Користувача просять ввести цифру, яка відповідає номеру тварини в програмі (Рис. 2.7).

```
void rationChange()
{
  int changedAnimal = 0;
  int changedMonth = 0;
  int changedFood = 0;
  Serial.println(" Choose type of animal to change ");
  Serial.println(" 1- Cow");
  Serial.println(" 2- Chicken");
  Serial.println(" 3- Pigs");
  while (Serial.available() == 0);
  changedAnimal = Serial.parseInt();
```

Рис 2.7 Процес вибору виду тварин, для зміни.

Як можна побачити, кожній тварині присвоєний певний номер, інформація про це виводиться на екран за допомогою команди «Serial.println()». Далі користувач повинен обрати номер тварини і ввести його. Саме на такому принципі і будується спілкування між програмою та користувачем.

2.4.Заповнення масиву добових раціонів.

На сьогоднішній день існує багато моделей добових раціонів для різноманітних сільськогосподарських тварин. Необхідно чітко розуміти, як на їх основі заповнювати наш масив.

Ми маємо тривимірний масив з інформацією про добові норми для певних тварин на кожен місяць. Кожен елемент такого масиву відповідає за добову норму певного корму, певного місяця для ного виду тварин. Для прикладу візьмемо

добові норми для трьох видів тварин: корів, курей та свиней(Табл. 2.1, Табл. 2.2, Табл. 2.3) [22].

Табл. 2.1

Добовий помісячний раціон годування корів на 1 особину

Корми / Місяць	Борошно, г	Коренеплоди, г	Солома, г	Сіно, г
Січень	-	2450	3200	4500
Лютий	-	1000	3020	4500
Березень	-	1950	2700	3800
Квітень	1500	1900	3730	4450
Травень	1100	2450	3200	4500
Червень	1200	2200	3520	4950
Липень	1500	2450	3840	4210
Серпень	-	2380	3340	4340
Вересень	-	2920	3980	4640
Жовтень	1900	2340	3340	4090
Листопад	1100	2260	3650	4720
Грудень	1000	2000	3000	4000

Табл. 2.2

Добовий помісячний раціон годування курей на 1 особину

Корми / Місяць	Рибне борошно, г	Ячмінь, г	Пшениця, г	Кукурудза, г
Січень	14	25	36	48
Лютий	16	24	32	45
Березень	-	29	36	44
Квітень	-	24	34	46
Травень	15	21	33	48
Червень	14	22	34	42

Липень	16	-	-	44
Серпень	13	-	-	43
Вересень	15	-	-	45
Жовтень	18	29	35	41
Листопад	19	22	39	42
Грудень	11	28	35	49

Табл. 2.3

Добовий помісячний раціон годування свиней на 1 особину

Корми / Місяць	Ячмінь, г	Висівки пшеничні, г	Пшениця, г	Шрот соєвий, г
Січень	121	256	353	442
Лютий	198	234	391	-
Березень	135	227	353	-
Квітень	194	241	374	-
Травень	165	293	315	496
Червень	174	246	353	456
Липень	185	234	394	474
Серпень	162	268	371	453
Вересень	-	-	323	484
Жовтень	-	-	394	419
Листопад	194	226	323	427
Грудень	185	223	394	465

Маючи такі початкові дані, необхідно перенести ці показники до шаблонного масиву. Слід зазначити, що програма не розглядатиме конкретний корм, на кшталт ячмінь, пшениця, кукурудза. Програма представить їх в якості номеру: корм 1, корм 2, корм 3.

Кожна група тварин матиме в раціоні корми типів 1, 2, 3 і 4. Однак, для кожної групи тварин вони будуть різними. Для прикладу, у корів під номером 1 буде стояти борошно, а для курей – рибне борошно.

Для перенесення даних раціонів до масиву, кожен показник повинен відповідати певному сектору (Рис. 2.8) [23].

```
int ration[numberOfAnimal][12][numberOfFood]=
{
  { {0, 2450, 3200, 4500},
    {0, 1000, 3020, 4500},
    {0, 1950, 2700, 3800},
    {1500, 1900, 3730, 4450},
    {1100, 2450, 3200, 4500},
    {1200, 2200, 3520, 4950},
    {1500, 2450, 3840, 4210},
    {0, 2380, 3340, 4340},
    {0, 2920, 3980, 4640},
    {1900, 2340, 3340, 4090},
    {1100, 2260, 3650, 4720},
  {1000, 2000, 3000, 4000} },

  { {14, 25, 36, 48},
    {16, 24, 32, 45},
    {0, 29, 33, 44},
    {0, 24, 34, 46},
    {15, 21, 39, 48},
    {14, 22, 34, 42},
    {16, 0, 0, 44},
    {13, 0, 0, 43},
    {15, 0, 0, 45},
    {18, 29, 35, 41},
    {19, 22, 39, 42},
  {11, 28, 35, 49} },

  { {121, 256, 353, 442},
    {198, 234, 391, 0},
    {135, 227, 353, 0},
    {194, 241, 374, 0},
    {165, 293, 315, 496},
    {174, 246, 353, 456},
    {185, 234, 394, 474},
    {162, 268, 371, 453},
    {0, 0, 323, 484},
    {0, 0, 394, 419},
    {194, 226, 323, 427},
  {185, 223, 394, 465} }


};
```

Рис. 2.8 Шаблонний масив для трьох видів тварин.

Згідно цього масиву, можна бачити, що у випадку коли певного корму нема в конкретному місяці, його значення записуємо, як 0. Загалом кожне значення в масиві відповідає добовій нормі певного корму певного місяця для певного виду тварин.

Такий масив формується на етапі написання коду. Після цього, за допомогою команд «`Serial.parseInt()`», «Serial.read()», і «Serial.available()» у користувача буде можливість вносити в нього зміни [24]. Для цього користувач

повинен обрати який елемент масиву він хоче змінити. Якщо, наприклад користувач вибере 2-й вид тварин, 4-й місяць та 3-й корм, це означає, що він бажає змінити добову норму для 2-го виду тварин(кури), на 4-й місяць(квітень), 3-го типу корму(пшениця), поточне значення якого – 34г. Після цього за допомогою функції «Serial.parseInt()», яка дозволяє вводити цифри з більш, як одним знаком(Рис. 2.13).

```
while (Serial.available() == 0);
int changedValue = Serial.parseInt();
ration[changedAnimal-1][changedMonth-1][changedFood-1] = changedValue;
Serial.println("Value succesffully changed");
```

Рис 2.13 Заміна значень певного елемента масиву.

2.5.Математична модель та алгоритм розрахунку добової норми.

Математична модель являє собою формулу для розрахунку щоденної норми по кожному дню місяця [25].

$$x_{F_j}(i) = a_{kj} + \frac{a_{(k+1)j} - a_{kj}}{l}\ i,\ \ i = 1,2,\dots,l,\ \ j = 1,2,\dots,n. \qquad (1)$$

де: $x_{F_j}(i)$ – це кількість корму F в i-день місяця, n – загальна кількість кормів для певного виду тварин, j – номер корму, a_{kj} – це добова норма корму j в місяці k, l – кількість днів в місяці.

Табл. 3

Таблиця бази даних оптимальної кормової суміші по місяцям.

корм / місяць	F_1	F_2	…	F_n
Січень	$a_{1,1}$	$a_{1,2}$	…	$a_{1,n}$
Лютий	$a_{2,1}$	$a_{2,2}$	…	$a_{2,n}$
…	…	…	…	…
Грудень	$a_{12,1}$	$a_{12,2}$	…	$a_{12,n}$

де aij – кількість грамів корму F_j в і-тому місяці року, i= 1,2,..,12, j = 1,2….,n, де n – кількість різних кормів.

Завдяки використанню методу апроксимації отримана математична модель дозволяє змінювати добовий раціон не раз на місяць, а поступово. Для більш комфортного та стабільного розвитку тварин. На діаграмі добового раціону для корів, видно наскільки сильно можуть відрізнятися добові норми в залежності від місяця (Рис. 2.9).

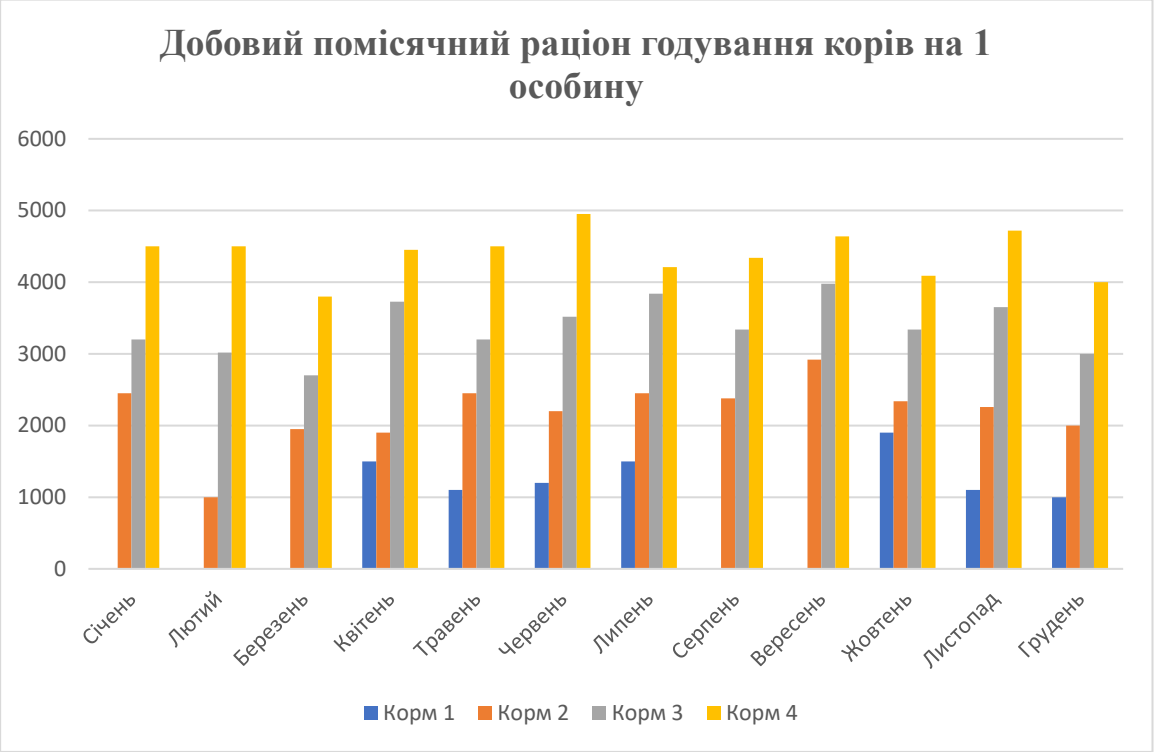

Рис. 2.9 Добовий помісячний раціон годування корів

Використовуючи рівняння (1), можна розрахувати кількість певного корму для кожного окремого дня. Дані необхідні для розрахунку – це поточний день місяця, кількість днів в цьому місяці, добова норма на цей місяць(з таблиці), та добова норма на наступний місяць[26].

На діаграмі 2, можна побачити, як змінюється щоденний раціон за 1 місяць. Розрахунок дозволяє отримати стабільне щоденне зростання раціону з кожним днем. В результаті ми отримаємо поступову адаптацію до наступної норми корму (Рис. 2.10).

Рис. 2.10 Щоденний раціон за 1 місяць

Для програмного розрахунку та автоматичного комбінування кормів, користувач повинен вказати місяць та день згідно якого будуть проводитись розрахунки.

Для введення цих даних користувач повинен обрати перейти в меню введення дати(Рис. 2.11).

```
void dateSelection()
{
  Serial.println(" Choose the required month ");
  Serial.println(" 1- January");
  Serial.println(" 2- February");
  Serial.println(" 3- March");
  Serial.println(" 4- April");
  Serial.println(" 5- May");
  Serial.println(" 6- June");
  Serial.println(" 7- Jule");
  Serial.println(" 8- August");
  Serial.println(" 9- September");
  Serial.println(" 10- October");
  Serial.println(" 11- November");
  Serial.println(" 12- December");
  while (Serial.available() == 0);
  selectedMonth = Serial.parseInt();
  delay(100);

  Serial.print("Choose the required day between 1 and ");
  Serial.println(daysInMonth[selectedMonth - 1]);
  while (Serial.available() == 0);
  selectedDay = Serial.parseInt();

 info = 0;
}
```

Рис. 2.11 Код процесу вибору дати.

В першу чергу програма запропонує вибрати номер місяця зі списку. За допомогою функції «Serial.parseInt()», користувач може обрати потрібний місяць ввівши потрібну цифру [27]. Потім програма пропонує обрати день, максимальна кількість, яких визначається з масиву, що відповідає за зберігання максимальної кількості днів в кожному місяці. Коли користувач обрав місяць, програма вибирає відповідне число з цього масиву і пропонує користувачу обрати день між 1-м та останнім для обраного місяця. Після вводу програма запам’ятовує введені значення дня та місяця.

Якщо користувач обрав день та місяць для розрахунку, він може дати команду програмі провести розрахунки. Як і всі інші, ця команда відповідає певній цифрі, що задає користувач.

Коли користувач обирає команду для розрахунку добового раціону для обраного дня, розпочинається цикл з розрахунку за формулою (1) добової норми всіх виді кормів для кожного виду тварин (Рис. 2.12).

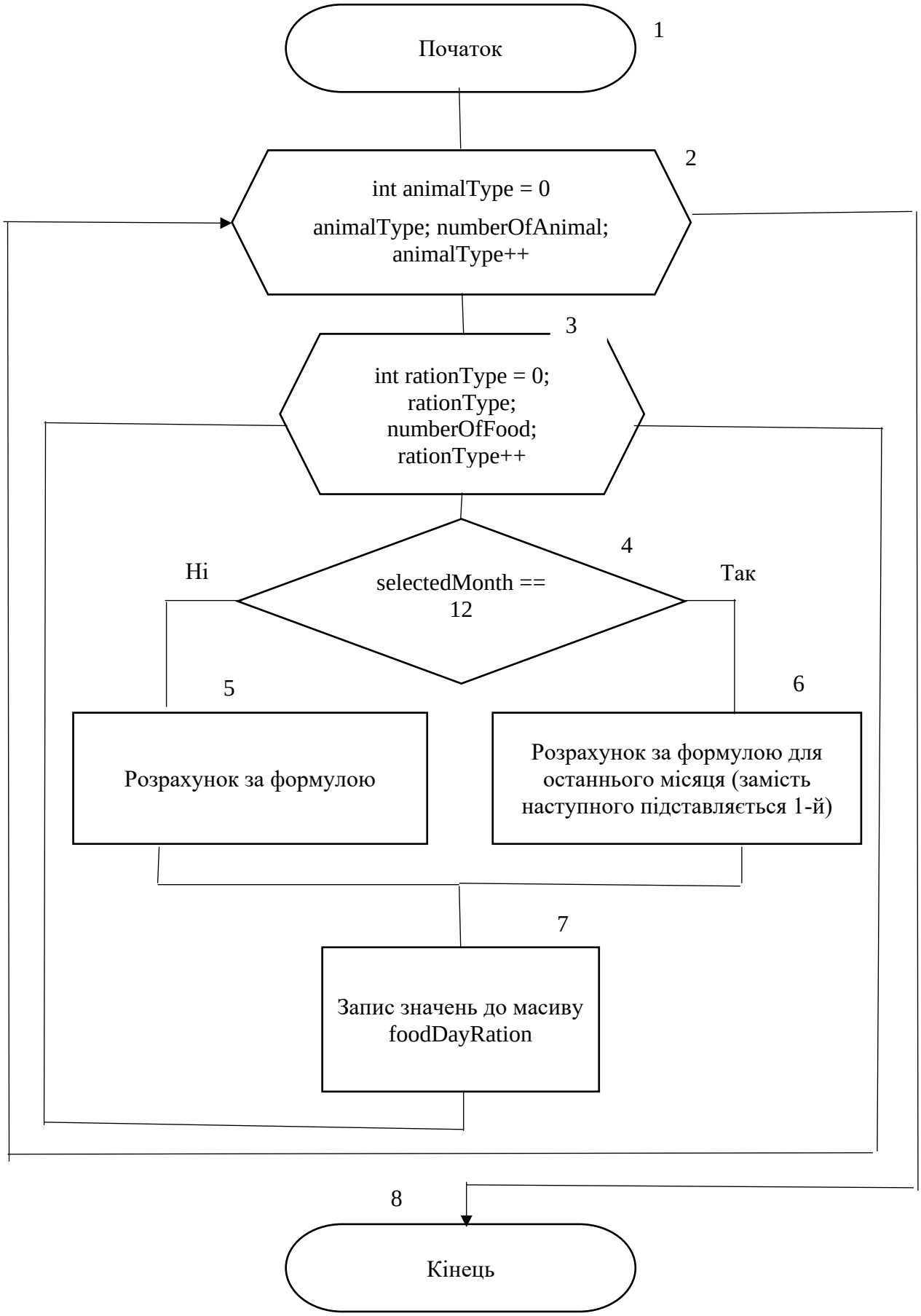

Рис. 2.12 Блок-схема алгоритму розрахунку добового раціону.

Процес розрахунку складається з двох циклів в процесі яких, проводяться розрахунки для кожного значення раціону в обраному місяці для кожного виду тварин. Згідно формули, підставляються значення з шаблонного масиву, згідно з обраним місяцем та днем. Отриманий результат записується до масиву «foodDayRation».

Після цього за командою користувача масив добових раціонів зі значеннями виводиться на екран про команді «Serial.println(foodDayRation[animalType][rationType])».

Масив «foodDayRation» – це двовимірний масив, з інформацією про розрахований добовий раціон для кожного виду тварин по кожному типу кормів, на обраний день обраного місяця.

Висновки до розділу 2

1) Розроблена структура масиву даних, який буде містити вихідні дані про розв’язувану задачу.
2) Розглянуто алгоритм побудови шаблонного масиву даних. Також програмні способи введення та корегування введеної інформації. Проаналізована механізм запису та завантаження шаблонного масиву даних до енергонезалежної пам’яті. Приведені блок-схеми цих процесів.
3) Показана система перевірки наявності збереженого шаблонного масиву в енергонезалежній пам’яті, і процес завантаження цього масиву з енергонезалежної пам’яті. Представлена блок-схема для цієї перевірки.
4) Розроблена математична модель розрахунку добової норми корму на кожен день для декількох видів тварин. Представлена порівняльна характеристика класичного підходу до розробки раціону тварин та результати розрахунків по запропонованій моделі.
5) Представлений процес вибору користувачем дня та місяця, для подальших розрахунків за формулою. Наведені приклади програмних команд, що використовуються для введення даних в систему. Продемонстрована блок-схема процесу і розрахунку добового раціону.

Розділ 3. Реалізація моделі за допомогою мікропроцесорної платформи Arduino

3.1 Побудова макетної моделі комбінування кормів

Для побудови моделі автоматичного комбінування кормів, для сільськогосподарських тварин, була використана мікропроцесорна платформа Arduino. Вона дозволяє скласти модель, що буде відображати виробничі процеси, які потрібно автоматизувати. Також вона дозволить побачити, як працюють обрані компоненти та зробити необхідні корективи.

Основою для розрахунків та контролю виконавчих пристроїв є мікропроцесорна платформа Arduino UNO. З її допомогою побудований макет системи для комбінації кормів. Основним завданням для такої системи, є система отримання потрібної кількості денного раціону по кожному компоненту кормів для змішування, для будь-якого виду тварин.

За дозування кормів, згідно розробленої схеми, відповідають двигуни постійного струму та серводвигун [28]. Ці двигуни розташовані на спеціальній платформі, котра буде рухатись на спеціальних рейках. Двигуни постійного струму відповідають за переміщення платформи по рейці.

Вважаємо, що кожний із кормів знаходиться в спеціальному контейнері, з якого потім буде відбуватися дозування, відповідно розрахункам. Всі контейнери розташовуються неподалік одне від одного, на певній, чітко визначеній відстані. Заповнення контейнерів відбувається послідовно для кожного виду тварин.

За допомогою двигуна постійного струму 1 платформа пересувається від одного контейнера з кормом до іншого. Час роботи двигуна відповідає відстані між контейнерами. Варто зазначити, що відстань між групами контейнерів з кормами для різних видів тварин може відрізняється від відстані між контейнерам з кормами для певного виду тварин.

Для побудови тестової моделі, будемо вважати, що відстань від одного контейнера до іншого, платформа, за допомогою серводвигуна, проходить за час в 2 секунди, тоді як відстань між групами контейнерів за 4. Це означає, що для

проходу платформи між контейнерами однієї групи, час роботи двигуна має складати також 2 секунди. Між групами контейнерів 4 секунди відповідно. За умовами тестової моделі час підходу платформи до першого контейнера також 4 секунди.

Коли платформа переміщається до певного контейнера, вона зупиняється. Тоді вмикається серводвигун, який перехиляє контейнер на 100 градусів, для того, щоб корм почав висипатись. Далі відбувається затримка на певний проміжок часу, в залежності від того, скільки грамів корму необхідно зсипати. Після того, як необхідна кількість корму висипалась, серводвигун повертає контейнер у вихідне положення і платформа продовжує рух до наступного контейнера.

Платформа підходить до кожного контейнера і насипає з нього необхідну кількість корму відповідно до проведених розрахунків. Якщо за результатами розрахунків кількість певного корму має бути 0, тобто у вказаний день тварина не повинна отримувати цей корм, тоді платформа одразу продовжує рух до наступного контейнера [29].

Коли платформа пройде всі контейнери, вмикається двигун постійного струму 2, який повертає платформу у вихідне положення. Запуск процесу комбінування кормів відбувається за відповідною командою користувача. За умовами тестової моделі час роботи двигуна постійного струму, необхідний для повернення платформи у вихідне положення вважається рівним 32-м секундам. Варто зазначити, що час роботи двигуна постійного струму необхідний для кожного з процесів, в роботі обрано довільно. В масштабах реального виробництва цей час буде залежати від обраного двигуна і відстані між контейнерами. Запуск процесу комбінування кормів відбувається за відповідною командою користувача. На блок-схемі можна побачити процес роботи двигунів (Рис. 3.1).

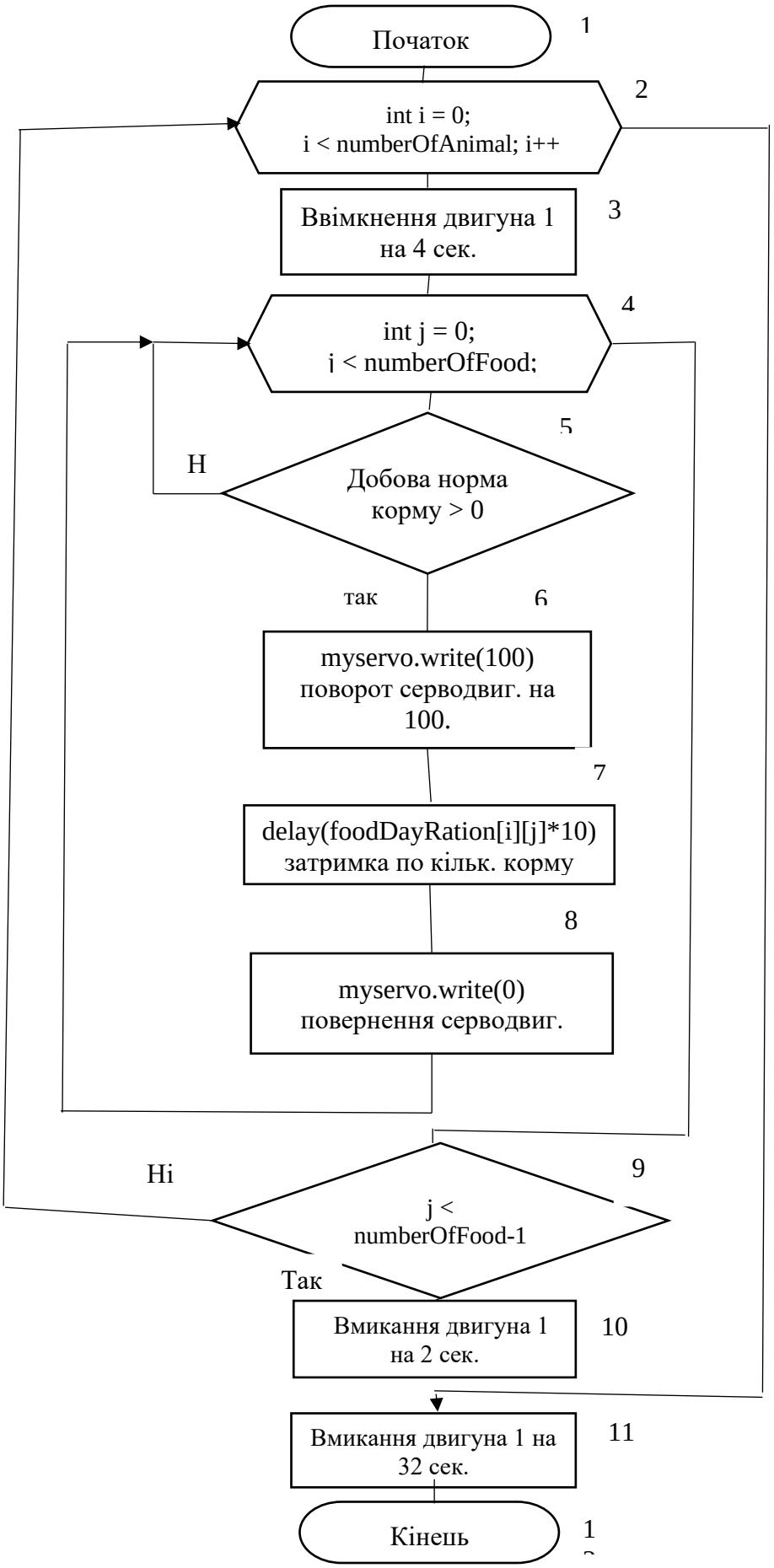

Рис. 3.1 Блок-схема роботи двигунів.

3.2 Керування двигунами за допомогою Arduino.

Керування обома типами двигунів відбувається за рахунок програмних команд записаних в системі. Підключення двигунів виконано до наступних виходів:

1. Двигун постійного струму №1 – пін № 13;
2. Двигун постійного струму №2 – пін № 12;
3. Серводвигун – пін № 9.

Схему підключень двигунів до плати Arduino UNO зображено на рис. 3.2.

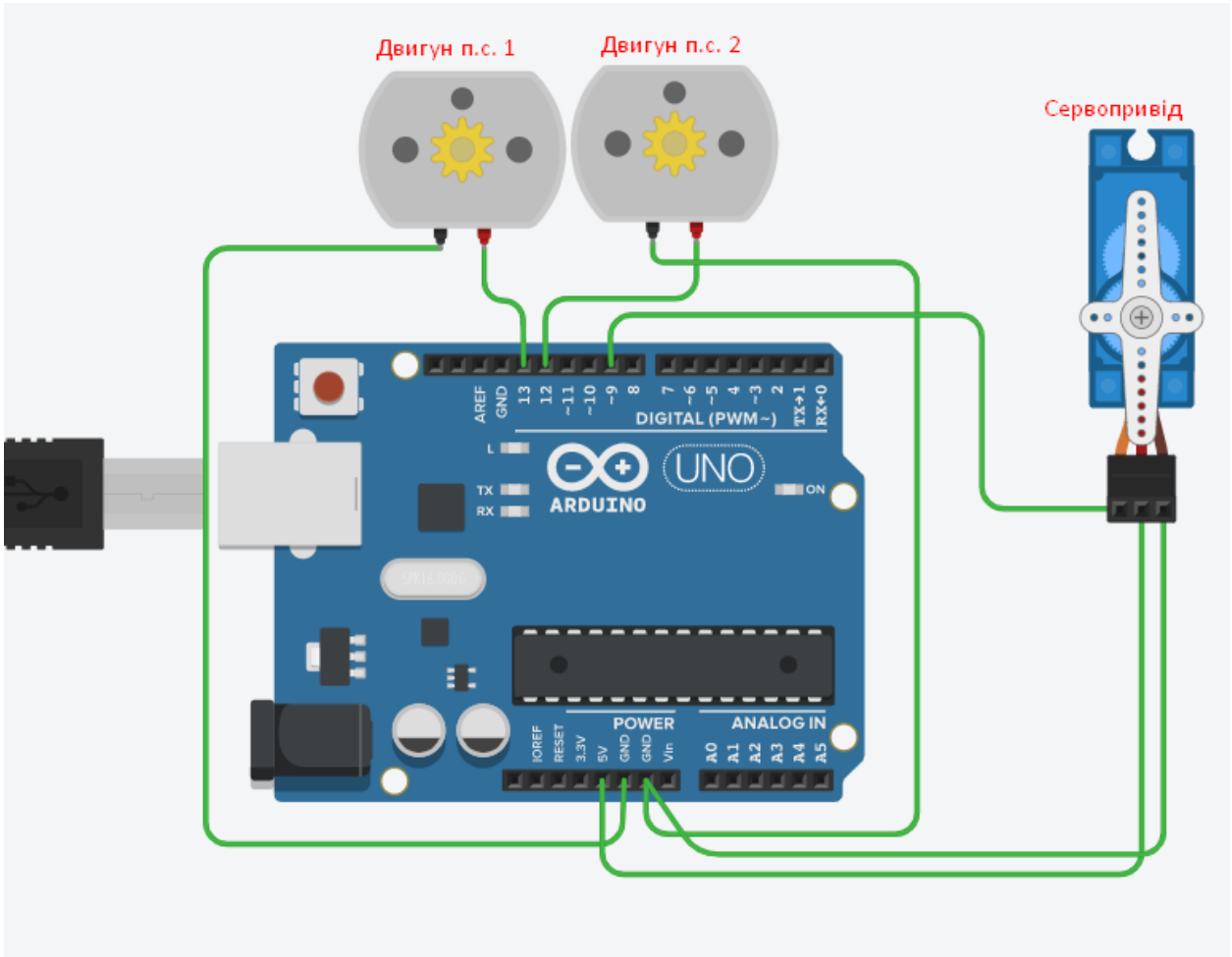

Рис. 3.2 Схема підключень двигунів до плати Arduino UNO.

Відповідні піни, до яких підключені двигуни, позначаються в програмі командою «pinMode(13, OUTPUT)», «pinMode(12, OUTPUT)» для двигунів

постійного струму та «myservo.attach(9);» для серводвигуна. На ці піни буде подаватись керуючий сигнал. Для керування серводвигунами за допомогою Arduino, при написанні коду, необхідно підключити стандартну бібліотеку «Servo.h», що дозволяє легко керувати сервоприводами. Також при оголошенні змінних необхідно створити об'єкт Servo («серво») з ім'ям «myservo», яке буде відповідати нашому серводвигуну, а також оголосити змінну, яка відповідатиме за позиціонування серводвигуна «int pos = 0;» [30].

Керування двигунами постійного струму відбувається за рахунок подачі на відповідний вихід керуючої плати, сигналу «HIGH», для запуску та «LOW», для зупинки. Час роботи двигуна визначається за допомогою затримки між цими командами. Затримка встановлюється командою «delay(2000);», де число 2000, вказує на час затримки в 2 секунди. Для того, щоб подати на відповідні виходи потрібний сигнал для відповідного двигуна, використовується команда «digitalWrite(13, HIGH);». Вона означає, що ми подаємо на вихід №13 сигнал зі значенням «HIGH». Це дозволить нам увімкнути двигун, що підключений до виходу №13.

Обидва двигуни постійного струму використовуються для переміщання платформи. Вони починають працювати лише тоді, коли серводвигун не працює. В інакшому випадку, одночасна робота може призвести до пошкодження обладнання.

Для керування серводвигуном ми використовуємо «myservo.write()». Це дозволяє нам здійснити поворот на необхідний кут (Рис 3.3).

```
if(foodDayRation[i][j] > 0)
{
  delay(1000);
 myservo.write(100);
delay(foodDayRation[i][j]*10);
 myservo.write(0);
  delay(1000);
}
```

Рис. 3.3 Команда поступового повороту серводвигуна.

На рисунку 3.3, показано процес повороту серводвигуна на кут від 0 до 100. Команда «myservo.write(100);» – команда сервоприводу перейти у позицію 100. Таким чином ми можемо задавати необхідний нам кут повороту.

Після повороту на визначений кут, необхідна певна затримка, для того, щоб висипались необхідна кількість корму. Ця затримка визначається в залежності від розрахованої кількості корму. Для кожного контейнера вона буде розраховуватись окремо. Команда для розрахунку затримки – «delay(foodDayRation[i][j]*10);». Завдяки цій команді, програма обирає певне значення з масиву, що був утворений під час розрахунку оптимальної кількості корму. Після цього обране значення множиться на 10 для отримання необхідної кількості мс.

Після затримки, сервопривід повертає контейнер у вихідне положення. Це відбувається за командою «myservo.write(0);»

Завдяки такій системі можна перехилити поступово контейнер для насипання необхідної кількості корм кожного типу. В результаті ми отримаємо дозування кожного корму відповідно до проведених розрахунків.

3.3 Керування функціями розробленої програми. Основне меню.

Загальна структура програми, являє собою набір функцій, які виконують певну процедуру. Вводячи певну команду, користувач викликає відповідну функцію, яка в свою чергу виконує записані в неї процеси. Загальний список команд основного меню виглядає наступним чином:

1) Введення дати(місяць та день)
2) Розрахунок добового раціону
3) Відобразити добовий раціон
4) Змінити масив раціонів
5) Відобразити поточний(активний) масив раціонів
6) Збереження поточного раціону
7) Завантаження поточного раціону

8) Старт двигунів.

Кожен пункт цього меню викликає відповідну функцію. Після завершення виконання певної функції, програма повертається в основне меню(Рис. 3.4).

```
void menu()
{
  if(info == 0)
  {
    Serial.println("Enter 1 to enter the date selection menu");
    Serial.println("Enter 2 to calculate day ration");
    Serial.println("Enter 3 to print calculated ration for current day");
    Serial.println("Enter 4 to enter the ration change menu");
    Serial.println("Enter 5 to show all rations");
    Serial.println("Enter 6 to save ration to EEPROM");
    Serial.println("Enter 7 to load ration from EEPROM");
    Serial.println("Enter 8 to start rotor");
    info = 1;
  }
```

Рис. 3.4 Основне меню програми.

На початку для користувача, у вікні віртуального COM-порту, виводиться повідомлення за допомогою команд «Serial.println». В цих повідомленнях програма пропонує ввести цифру, що відповідає обраному пункту меню. Програма записує команду користувача за допомогою «Serial.available» та «Serial.read». Далі в залежності від обраного пункту меню, програма викликає відповідну функцію. Оскільки можливих команд досить багать, замість «if, else» використовується оператор «switch.. case» (Рис. 3.5).

```
int command = Serial.read();
switch (command)
 {
   case '1':
     Serial.println("Date selection");
     dateSelection();
     break;
   case '2':
     Serial.println("Calculating");
     calcCurrDayRation();
     break;
   case '3':
     Serial.println("RationForDay");
     printRationForDay();
     break;
   case '4':
     Serial.println("rationChangeMenu");
     rationChange();
     break;
   case '5':
     Serial.println("ration");
     rationShow();
     break;
   case '6':
     Serial.println("saveRation");
     saveRation();
     break;
  case '7':
     Serial.println("loadRation");
     loadRation();
     break;
  case '8':
     Serial.println("rotor");
     rotor();
     break;
```

Рис. 3.5 Ініціалізація виклику функцій

Після того, як користувач обрав пункт меню, ввівши відповідну цифру, програма виведе повідомлення з інформацією про те, який саме варіант був обраний.

При виборі першого пункту, розпочнеться процес введення користувачем номера місяця та дня. Другий пункт викликає функцію, що проводить розрахунок добової кількості корму на основі введених користувачем даних. Третій пункт виводить у вікні віртуального COM-порту значення масиву з розрахованими добовими значеннями кормів для кожної тварини. Четвертий пункт переводить користувача у меню зміни даних в шаблонному масиві раціонів, яке було описано в 2-му розділі. П’ятий – відповідає за відображення поточного(активного)шаблонного масиву добових раціонів, з яким на даний момент працює користувач. 6-й і 7-й пункти відповідають за збереження та завантаження збереженого шаблонного масиву з енергонезалежної пам’яті. Останній, восьмий, пункт викликає функцію, яка відповідає за роботу двигунів.

Після старту системи програма не одразу переходить в основне меню. В першу чергу вона перевіряє наявність збереженого масиву в енергонезалежній пам’яті. Цей процес також був описаний в 2-му розділі. Після його завершення програма переходить до основного меню.

Варто зазначити, що за рахунок використання функцій для кожного процесу і виклику їх з меню, користувач не зобов’язаний вводити всі команди. Наприклад, для розрахунку добової кількості корму та процесу комбінування, для користувача достатньо лише пунктів 1, 2, і 8. Тобто, користувач повинен лише ввести необхідний місяць та день, вказати програмі провести розрахунки, та запустити двигуни. Йому зовсім не обов’язково виводити на екран результати розрахунків, якщо він впевнений в правильності роботи систем. Користувач може викликати лише ті функції, що необхідні йому на даний момент.

3.4 Система взаємного розташування контейнерів.

Для того, щоб сформувати добову кормову суміш недостатньо лише насипати корм в одну ємність. Важливо також розподілити корма відповідно до того, якій тварині потрібно його дати.

Наповнення контейнерів кормами відіграє досить важливе значення. Якщо наповнити контейнери кормами в довільному порядку, то система буде працювати не правильно. Наповнення контейнерів з кормами має відбуватися відповідно до масиву добової кількості кормів для певного дня.

Згідно шаблону спочатку потрібно розташувати контейнери для першого виду тварин, потім для другого і для третього. На кожну групу тварин відводиться відповідно по чотири контейнера. Заповнення контейнерів однієї крупи також має проводитись послідовно. Це зумовлено тим, що функція, котра керує двигунами обирає значення затримки, яка відповідає за необхідну кількість певного корму, послідовно бере значення з масиву, що був заповнений на основі розрахунків (Табл. 3.1).

Табл. 3.1

Кількість корму на певний день певного місяця для всіх видів тварин

Корми / Тварини	Корм 1	Корм 2	Корм 3	Корм 4
Тварин 1	X_{ij}, г	X_{ij}, г	X_{ij}, г	X_{ij}, г
Тварина 2	X_{ij}, г	X_{ij}, г	X_{ij}, г	X_{ij}, г
Тварина 3	X_{ij}, г	X_{ij}, г	X_{ij}, г	X_{ij}, г

де X_{ij} – це денні норма корму на і-й день, j-го місяця

Відповідно до такої таблиці і першим потрібно заповнювати контейнер першим типом корму, для першого виду тварин. Другий, відповідно, другим типом корму для першого виду тварин, і т.д. Згідно цього алгоритму схема заповнення контейнерів матиме такий вигляд (Табл. 3.2):

Табл. 3.2

Схема заповнення контейнерів кормами.

Корми / Тварини	Корм 1	Корм 2	Корм 3	Корм 4
Тварин 1	Контейнер №1	Контейнер №2	Контейнер №3	Контейнер №4
Тварина 2	Контейнер №5	Контейнер №6	Контейнер №7	Контейнер №8
Тварина 3	Контейнер №9	Контейнер №10	Контейнер №11	Контейнер №12

Як було сказано раніше, кормі для різних видів тварин можуть відрізнятись, або співпадати, але це не впливає на процес заповнення контейнерів. Він повинен відбуватись за чітко визначеною схемою.

Крім наповнення контейнерів, не менш важливим є і їх розташування відносно одне-одного. Так, як за насипання всіх кормів відповідає один механізм, то для прогнозування його роботи, відстань між контейнерами має бути чітко вивіреною.

Важливо не лише розташування контейнерів однієї групи відносно одне-одного, а й відстань між різними групами контейнерів. Для правильного спрацювання механізму відстань між сусідніми контейнерами кожної групи має бути однаковою.

Те ж стосується і відстані між різними групами контейнерів. Відстань між контейнерами в одній групі та між групами може співпадати. Приклад розташування контейнерів показано на рисунку 3.6. Приклад конструкції, що реалізує розроблену модель можна побачити на рисунку 3.7.

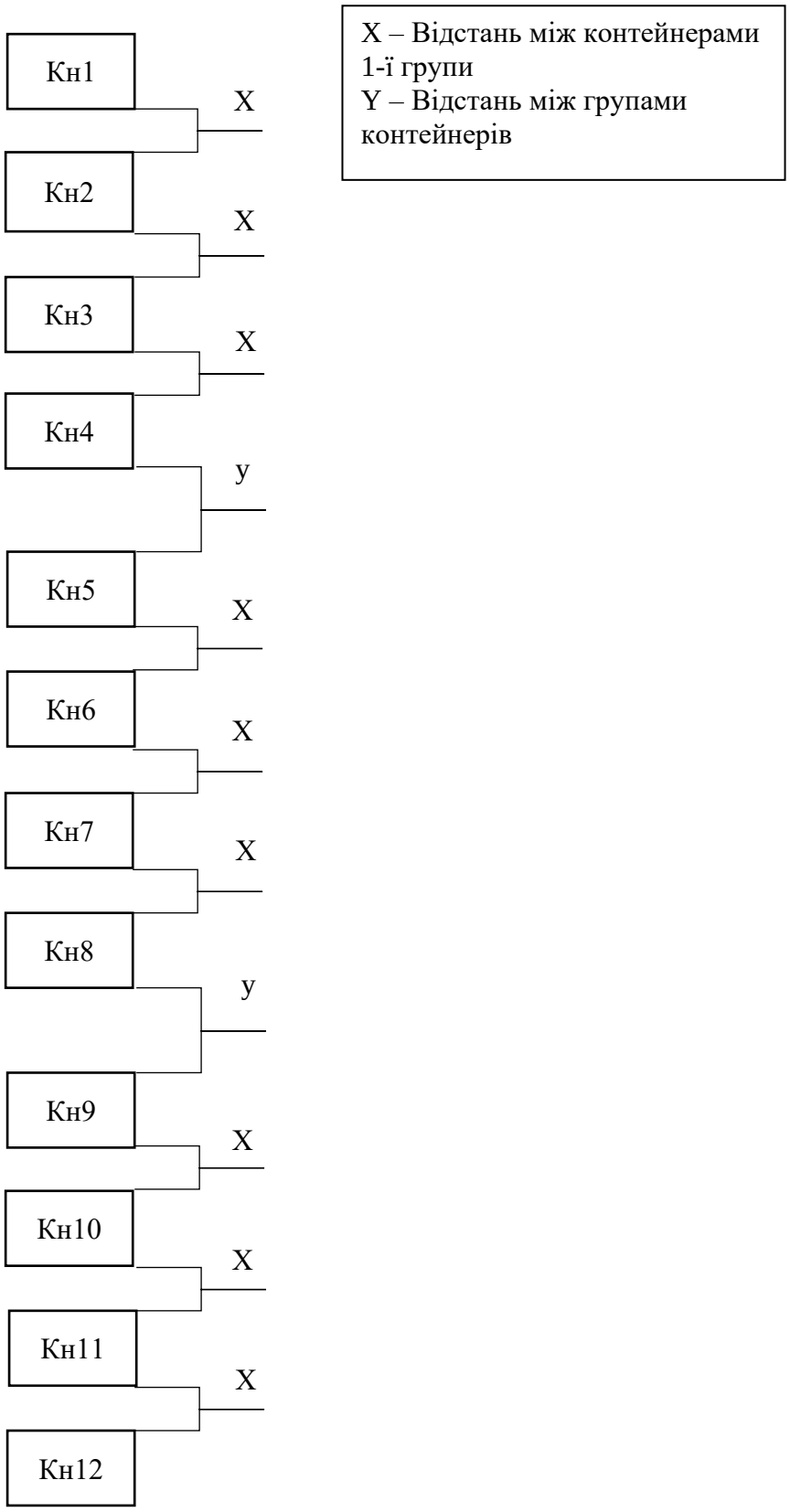

Рис. 3.6 Приклад розташування контейнерів для кормів.

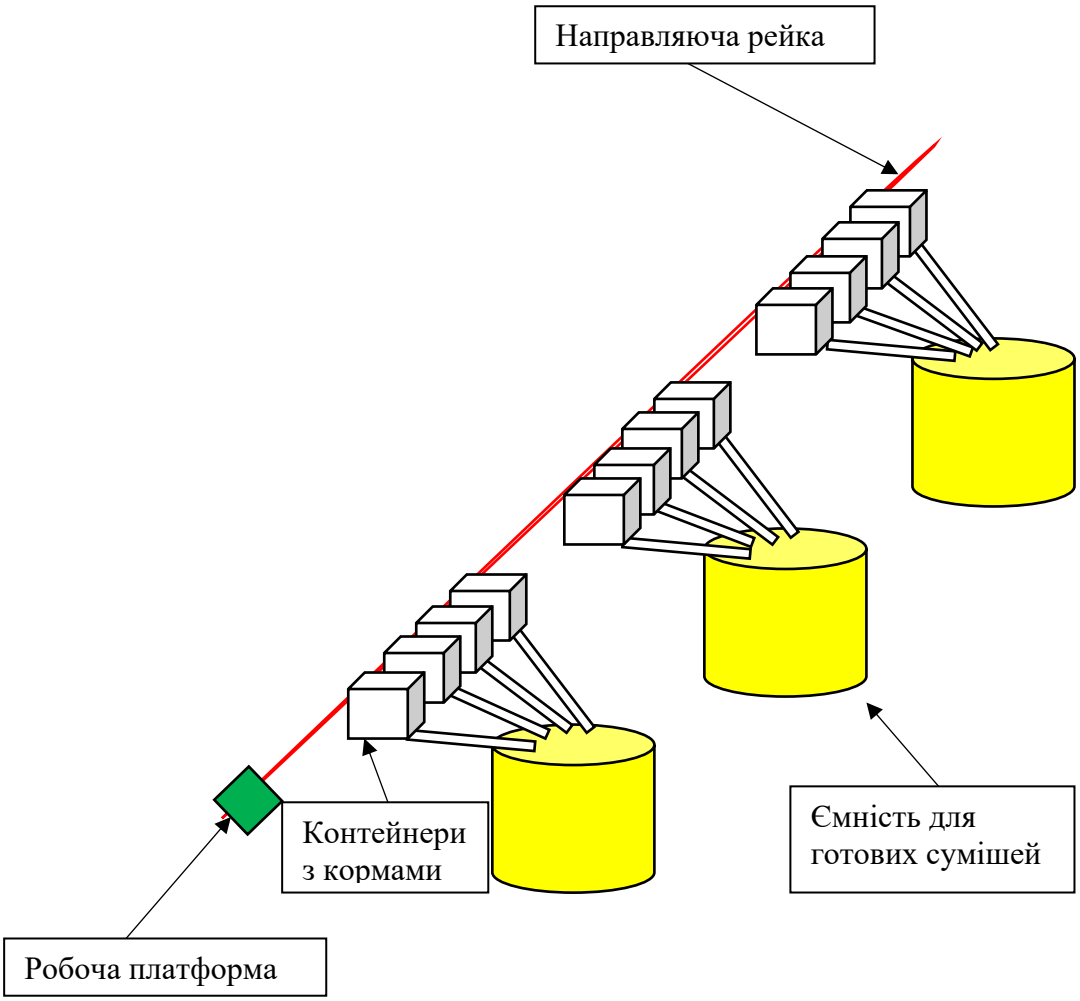

Рис. 3.7 Конструкція, реалізує розроблену модель.

Як видно на схемі, корма в залежності від відповідності певній групі тварин, після дозування, зсипаються у певні ємності. Отримана суміш може бути направлена на подальшу обробку, якщо вона необхідна, або одразу роздана тваринам.

Суміш кормів, що поступає у кінцеві ємності являє собою комбінацію всіх необхідних кормів для кожного виду тварин, в необхідній кількості, на поточний день поточного місяця.

Висновки до розділу 3

1) Побудована тестова макетна модель комбінування кормів, на основі запропонованих параметрів. Завдяки використанню серводвигуна та двигунів постійного струму. Макетна модель імітує процес засипання та комбінування різних типів корму, для різних видів тварин. Всі маніпуляції проводяться на основі програмних розрахунків за математичною моделлю.
2) Визначена схема взаємного розташування контейнерів з кормами для правильного спрацювання системи, а також відповідна система правильного алгоритму заповнення контейнерів.
3) Запропонована схематична модель виробничого комплексу для побудови системи автоматизованого розрахунку та зсипання кормів на кожен день.

Розділ 4 Адаптація розробленого алгоритму для промислового використання.

4.1.Промислові варіанти зберігання масивів даних. Бази даних.

В залежності від масштабів підприємства різниться як тип так і складність розробки баз даних для тваринних раціонів. Для невеликих ферм де необхідно лише розрахувати щоденній раціон тварин виходячи з місячного об’єму, та його вартість, можна використати прості бази даних.

При проектуванні необхідної бази даних важливо звертати увагу на спосіб її зберігання і на те, як керуючий модуль буде отримувати інформацію від нашої бази даних.

Першим варіантом буде зберігання, безпосередньо в пам’яті керуючого модулю. В нашому випадку в пам’яті плати Arduino. Тоді блок-схема матиме такий вигляд (Рис. 4.1).

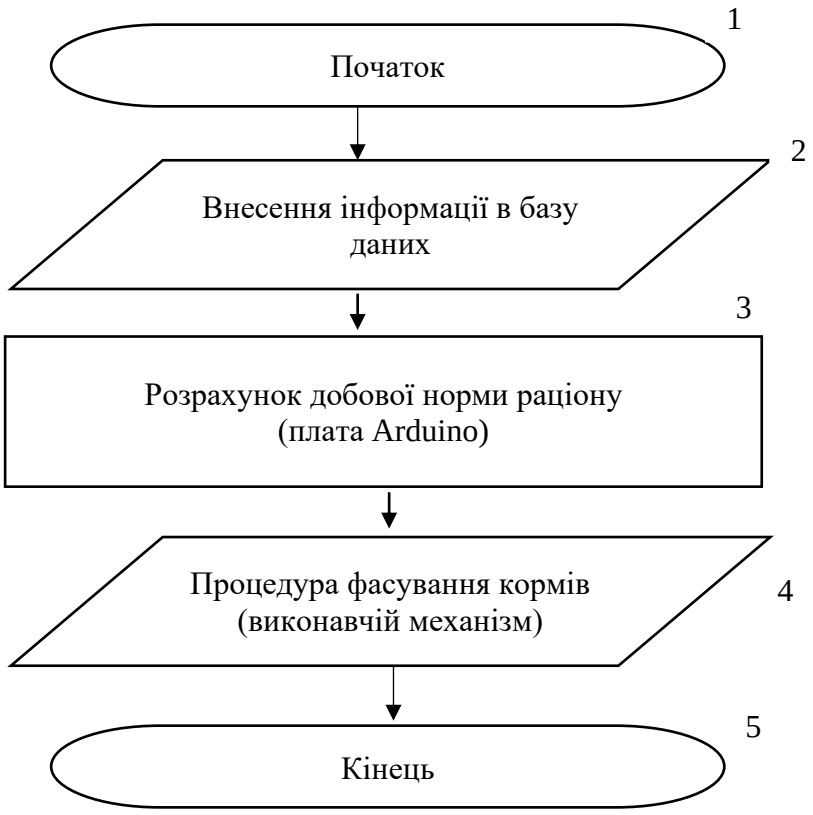

Рис. 4.1 Блок схема механізму комбінування кормів із внутрішньою базою даних

Такий варіант прийнятний для підприємств малого масштабу. Це зумовлено тим, що пам’ять керуючого пристрою, зазвичай досить обмежена. В результаті, що ми не зможемо записати в таку базу даних багато інформації.

Однак у випадку коли на підприємстві не так багато різних видів тварин, цього може вистачити. Для невеликої системи нам потрібно лише записати інформацію про місячну норму кожного виду кормів для кожного місяця.

Використання вбудованої пам’яті керуючого пристрою різко обмежує доступні ресурси. Це означає, що навіть на фермерських підприємствах середнього розміру, буде складно реалізувати запропоновану схему автоматизації.

Для автоматизації процесу підбору кормів на підприємствах середнього розміру доцільно зберігати бази даних на додаткових електронних носіях. Прикладом може стати жорсткий диск комп’ютера (Рис. 4.2).

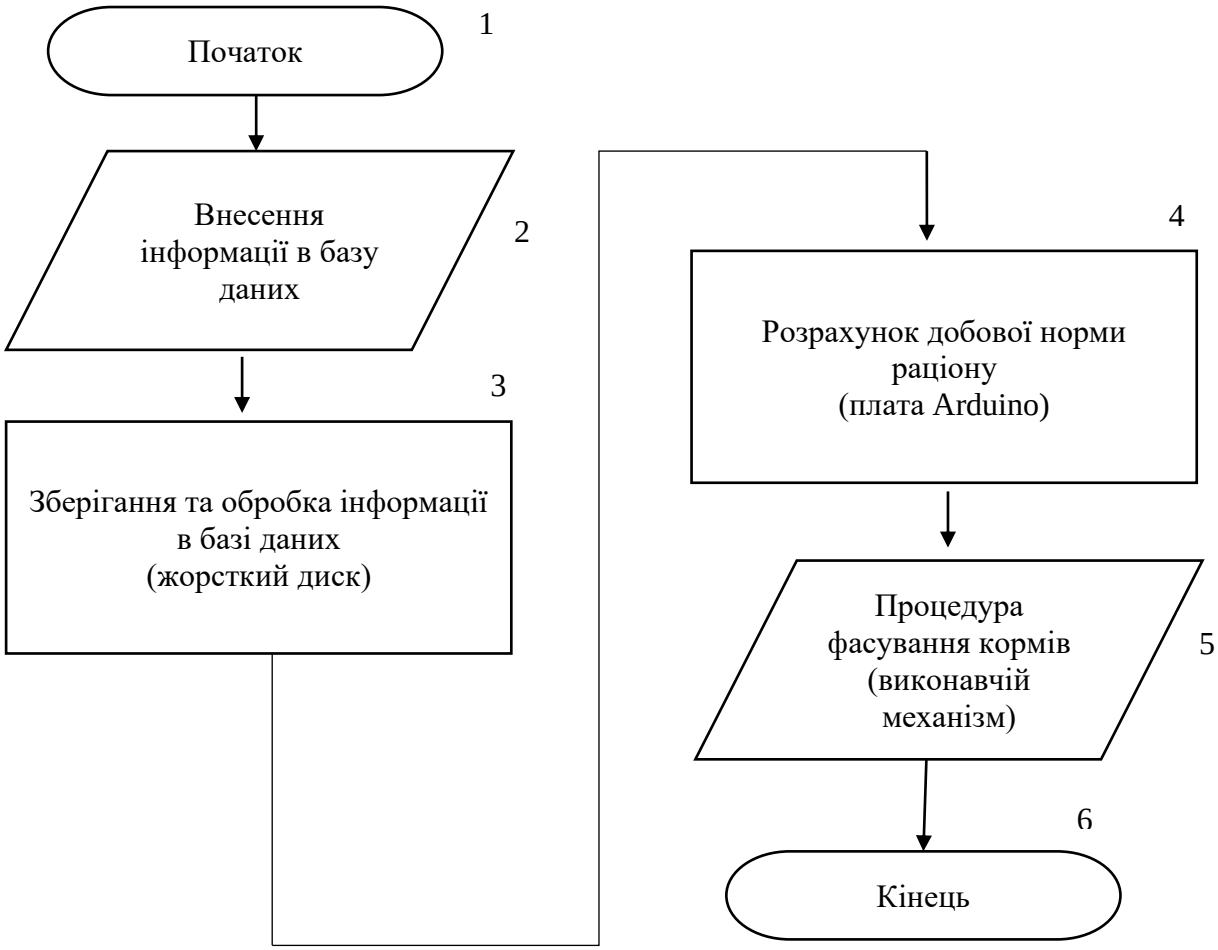

Рис. 4.2 Блок схема комбінування кормів із зовнішньою базою даних

Розробка алгоритму в такому форматі дозволить створити бази даних більшого розміру. На підприємствах де вирощують декілька різних видів сільськогосподарських тварин, необхідно створювати бази даних для кожної групи тварин.

В свою чергу, керуючий елемент повинен взаємодіяти з усіма базами даних. Використання внутрішньої пам’яті керуючого пристрою це не дозволить, в наслідок обмеженого розміру вбудованої пам’яті.

Для проектування бази даних для нашої задачі, а також для полегшення взаємодії користувача з нею, необхідно використовувати системи управління базами даних (СУБД).

Система управління базами даних - це сукупність мовних і програмних засобів, яка здійснює доступ до даних, дозволяє їх створювати, змінювати і видаляти, забезпечує безпеку даних і т.д. Загалом СУБД - це система, що дозволяє створювати бази даних і маніпулювати даними з них. А здійснює цей доступ до даних СУБД за допомогою спеціальної мови - SQL.

SQL - мова структурованих запитів, основним завданням якого є надання простого способу зчитування і запису інформації в базу даних.

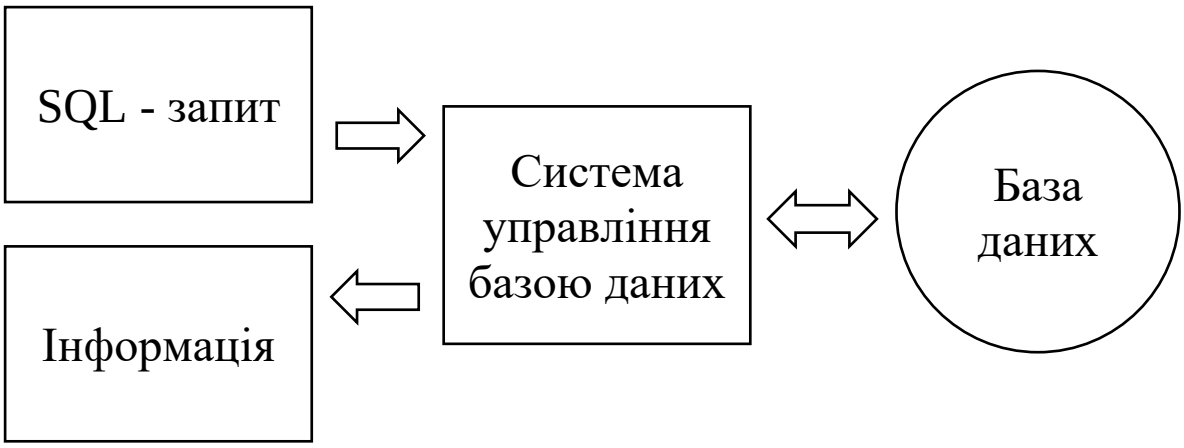

Рис. 4.3 Найпростіша схема роботи з базою даних.

За характером використання СУБД ділять на ті, що призначені для одного користувача (призначені для створення і використання БД на персональному комп'ютері) і розраховані на багато користувачів (призначені для роботи з єдиною БД декількох комп'ютерів, об'єднаних в локальні мережі).

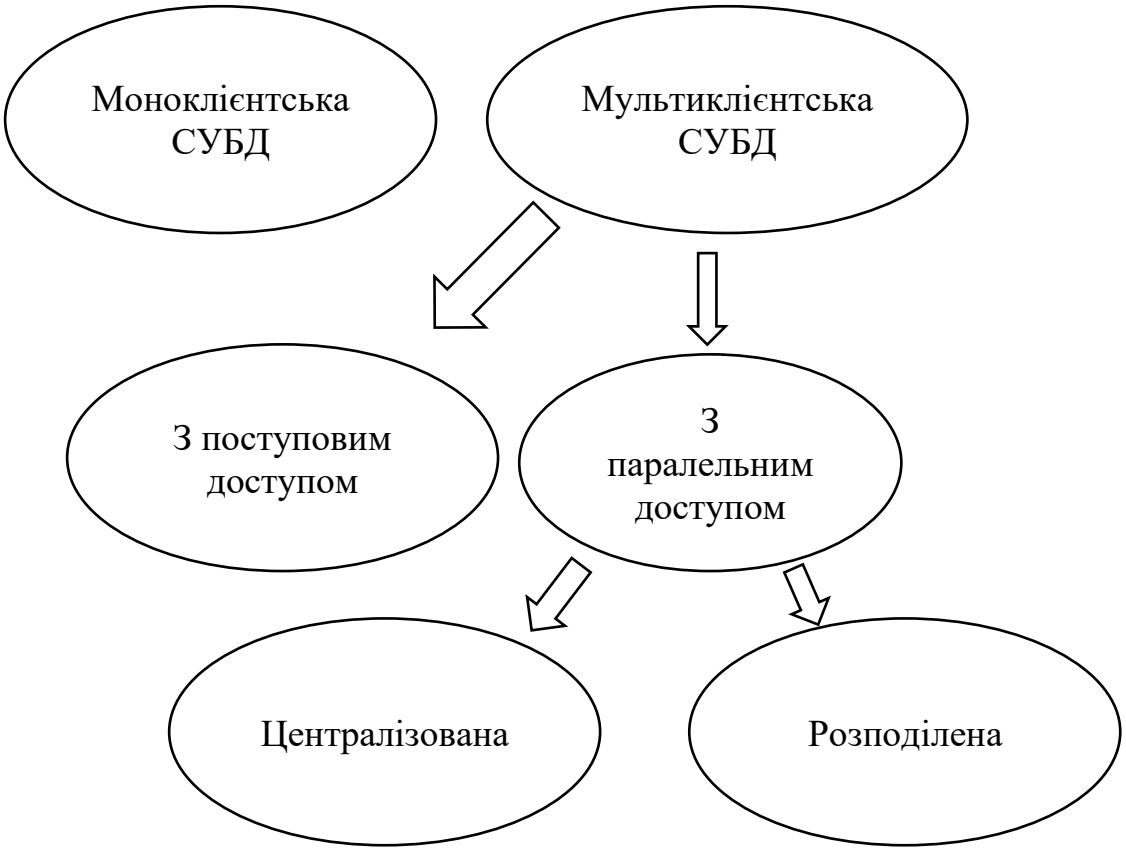

Рис. 4.4 Загальна схема розподілення СУБД за характером використання.

4.2.Аналіз систем управління базами даних (СУБД)

На сьогоднішній день число використовуваних СУБД обчислюється десятками. Найбільш відомі СУБД для одного користувача - Microsoft Visual FoxPro і Access, розраховані на багато користувачів - MS SQL Server, Oracle та MySQL.

Однією з найпростіших версій СУБД є SQLite3. (Рис. 4.5) Ця версія найкраще підійде для невеликих баз даних з обмеженими ресурсами пам’яті.

SQLite - це бібліотека на мові C, яка реалізує невеликий, швидкий, автономний, високонадійний, повнофункціональний механізм бази даних SQL. SQLite - це найбільш часто використовуваний механізм баз даних в світі. SQLite вмонтований в усі мобільні телефони і більшість комп'ютерів і входить до складу безлічі інших додатків, які люди використовують кожен день.

Формат файлу SQLite є стабільним, кросплатформним і назад сумісним, і розробники обіцяють зберегти його в такому вигляді до 2050 року. Файли бази даних SQLite зазвичай використовуються в якості контейнерів для передачі

багатого контенту між системами і як формат довгострокового архівування даних [31]. Активно використовується понад 1 трильйон (1e12) баз даних SQLite [32]. Вихідний код SQLite знаходиться у відкритому доступі і може бути використаний будь-яким бажаючим для будь-яких цілей.

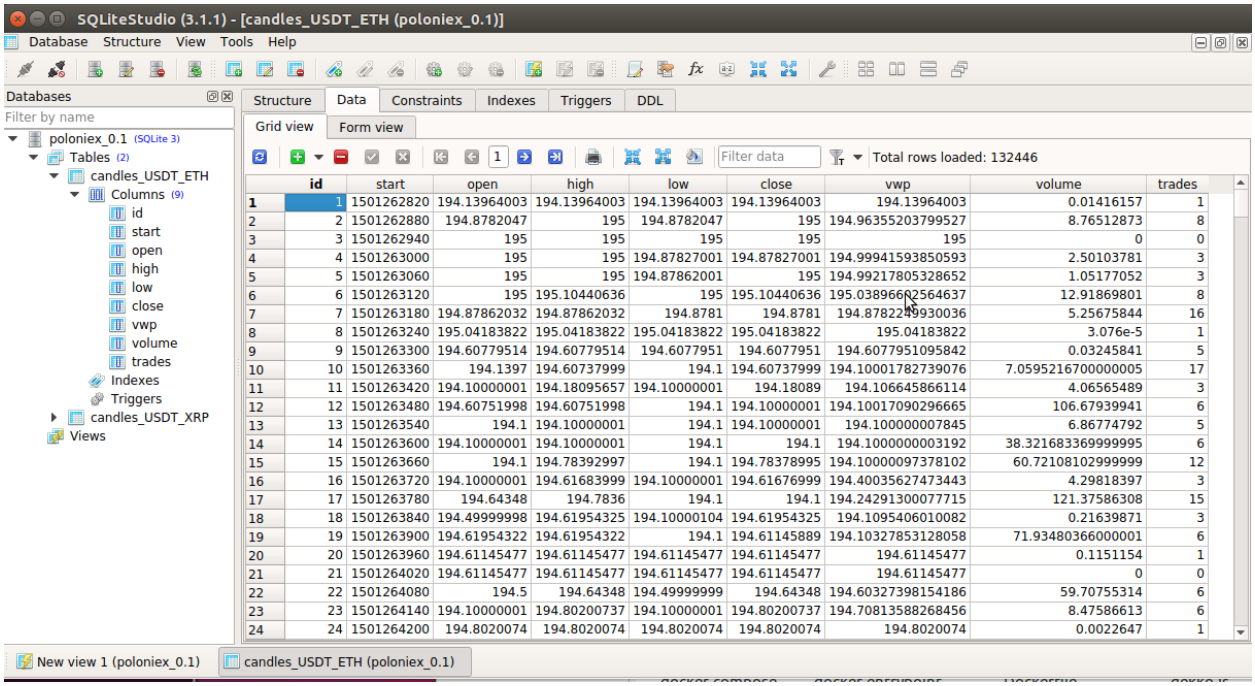

Рис. 4.5 Вікно програми SQLite

Ця версія СУБД за рахунок простоти і відносно невеликого розміру БД, дозволяє вирішити проблему недостатньої пам’яті на керуючому модулі. Окрім цього розробка таких БД займає набагато менше часу.

SQLite3 також вдало адаптована для використання з платами Arduino, що значно спрощує процес розробки бази даних і програмного коду в цілому. Схожість мов програмування для керуючого модулю і СУБД також значною мірою спростить процес розробки.

SQLite читає і записує маленькі об’єкти (наприклад, мініатюрні зображення) на 35% швидше, ніж ті ж самі об’єкти можна читати або записувати в окремі файли на диску за допомогою fread () або fwrite ().

Різниця в продуктивності виникає, тому що при роботі з базою даних SQLite системні виклики open () і close () викликаються тільки один раз, тоді як

open () і close () викликаються один раз для кожного великого двійкового об'єкта при використанні великих двійкових об'єктів, що зберігаються в окремі файли.

Цифра 35% є приблизною. Фактичний час залежить від обладнання, операційної системи і деталей експерименту, а також з-за випадкових коливань продуктивності на реальному обладнанні.

Джим Грей та інші досліджували продуктивність читання великих двійкових об'єктів в порівнянні з файловим введенням-висновком для Microsoft SQL Server і виявили, що читання великих двійкових об'єктів з бази даних відбувається швидше для великих двійкових об'єктів розміром менше 250-1 МБ.[33]

BLOB — спеціальний тип даних, призначений, в першу чергу, для зберігання зображень, аудіо і відео, а також компільованого програмного коду.

У цьому дослідженні база даних як і раніше зберігає ім'я файлу контенту, навіть якщо контент зберігається в окремому файлі. Таким чином, база даних запитується для кожного BLOB, навіть якщо це тільки для вилучення імені файлу. Ключем для великого двійкового об'єкта є ім'я файлу, тому попередній доступ до бази даних не потрібно. Оскільки база даних ніколи не використовується при читанні вмісту з окремих файлів, поріг, при якому пряме введення-виведення файлів стає швидше.

У наведеній нижче таблиці показано час, необхідне для читання великих двійкових об'єктів, що зберігаються в окремих файлах, поділена на час, необхідний для читання великих двійкових об'єктів, повністю зберігаються в базі даних. Отже, для чисел, що перевищують 1.0, великі двійкові об'єкти швидше зберігати безпосередньо в базі даних. Для чисел менше 1,0 великі двійкові об'єкти швидше зберігати в окремих файлах. (Табл. 4.1)

У кожному разі розмір кеш-пам'яті пейджера був скоректований таким чином, щоб об'єм кеш-пам'яті становив приблизно 2 МБ. Наприклад, кеш на 2000 сторінок використовувався для сторінок розміром 1024 байт, а кеш на 31 сторінку - для сторінок розміром 65536 байтів. Значення BLOB зчитувалися у випадковому порядку.

Табл. 4.1

Відношення швидкості зберігання даних

Розмір сторінки бази даних	Розмір BLOB						
	10 тис.	**20 тис.**	**50 тис.**	**100 тис.**	**200 тис.**	**500 тис.**	**1 міс.**
1024	1,535	1.020	0,608	0,456	0,330	0,247	0,233
2048	2,004	1,437	0,870	0,636	0,483	0,372	0,340
4096	2,261	1,886	1,173	0,890	0,701	0,526	0,487
8192	2,240	1,866	1,334	1.035	0,830	0,625	0,720
16384	2,439	1,757	1,292	1.023	0,829	0,820	0,598
32768	1,878	1,843	1,296	0,981	0,976	0,675	0,613
65536	1,256	1,255	1,339	0,983	0,769	0,687	0,609

Ми виводимо такі практичні правила з наведеної таблиці:

- Розмір сторінки бази даних 8192 або 16384 дає найкращу продуктивність для операцій введення-виведення великих двійкових об'єктів.
- Для великих двійкових об'єктів розміром менше 100 КБ читання виконується швидше, коли великі двійкові об'єкти зберігаються безпосередньо у файлі бази даних. Для великих двійкових об'єктів розміром понад 100 КБ читання з окремого файлу відбувається швидше.

Загалом, СУБД SQLite, найкращим чином підходидь для розробки та обслуговування баз даних на невеликих підприємствах і дозволить створити

прийнятні бази даних для зберігання даних про норми раціонів тварин. Також вона дозволить легко вносити корективи і змінювати вміст баз даних.

Більш функціональним аналогом SQLite, може бути СУБД MySQL.

MySQL — вільна система управління реляційними базами даних. MySQL був розроблений компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL — компактний багато-потоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багато-поточності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

Створення більш складних баз даних дозволить розширити можливості розробленої системи з підбору раціонів.

4.3.Огляд можливості використання промислових логічних контролерів

На відміну від невеликих підприємств, масштабні фермерські господарства утримують набагато більше сільськогосподарських тварин. Тому потреби в кормах, на таких підприємствах значно більші.

В таких умовах з’являється необхідність розробки автоматизованої системи керування з більш габаритними виконавчими механізмами. Для такої задачі, використання мікроконтролерів, буде недоцільним. В такій системі вдалим рішенням буде використання повноцінного програмованого логічного контролера (ПЛК).

Програмований логістичний контролер (PLC, Programmable Logic Controller, PLC) або програмований контролер - це електронний компонент промислового контролера, спеціалізованого (комп'ютеризованого) пристрою, який використовується для автоматизації процесів. Основним режимом тривалої роботи ПЛК, часто в несприятливих умовах навколишнього середовища, є його автономне використання, без серйозного технічного обслуговування і при незначному або без втручання людини.

На відміну від:

- мікроконтролера, сфера застосування ПЛК зазвичай є автоматизованими процесами промислового виробництва, в контексті виробничого підприємства;
- комп'ютерів, ПЛК орієнтовані на роботу з машинами і мають розвинений "машинний" режим введення вхідних та вихідних сигналів, з датчиків, і виконавчі механізми, на відміну від можливостей комп'ютера, орієнтованого на людину (клавіатура, миша, монітор і т.д.);
- інтегрованих систем - ПЛК виготовляється як самостійний продукт, окремо від контрольованого ним обладнання.

В системах управління технологічними об'єктами переважають логічні команди над числовими операціями, що дозволяє, на відміну від мікроконтролера, отримати потужні системи, що працюють в режимі реального часу. У сучасних ПЛК числові операції проводяться на рівні з логічними. У той же час, на відміну

від більшості комп'ютерних процесорів, ПЛК забезпечує доступ до окремих бітів пам'яті.

ПЛК не має класичного інтерфейсу, наприклад клавіатури та дисплея. Їх програмування, діагностика і технічне обслуговування виконується підключеними для цього програматорами - спеціальним пристроєм або пристроями на основі більш сучасних технологій - персонального комп'ютера або ноутбука, зі спеціальними інтерфейсами і спеціальним програмним забезпеченням. У системах управління процесами, ПЛК взаємодіє з різними компонентами людино-машинних інтерфейсів або робочих місць операторів на базі ПК, в промисловому варіанті, як правило, через промислову мережу.

Основною відмінністю ПЛК від ПК є значна кількість пристроїв введення-виведення для датчиків і виконавчих пристроїв. Сфера ж використання нерозривно пов’язана з повсюдним розповсюдженням автоматизації систем виробництва. Нижче, на рисунку 1.1, представлений ПЛК100 від фірми ОВЕН, як типовий представник промислових ПЛК для систем середньої складності.

Рис. 4.6 ПЛК100 ОВЕН

На прикладі ПЛК100 фірми ОВЕН можна відобразити основні характеристики ПЛК. Варто окремо оцінювати характеристики ПЛК, як комп’ютера і як системи керування. До характеристик ПЛК як комп’ютера варто віднести наступні: характеристики центрального процесору, об’єм оперативної пам’яті, об’єм Retain-пам'яті, об’єм енергонезалежної пам'яті зберігання програм та архівів, інтерфейси, протоколи. Саме ці характеристики визначають сферу використання ПЛК, його швидкодію і складність доступних до реалізації за його допомогою програм. Щодо характеристик ПЛК, як системи керування, то до них відносяться такі: кількість дискретних входів, кількість дискретних виходів, кількість аналогових входів, конструктивне виконання, напруга живлення, споживна потужність, ступінь захисту корпусу.

Таким чином правильний підбір контролера може стати проблемою. Також проблема може виникнути при модифікації системи автоматизованого керування. Найпростіша з таких перешкод – банальний брак вільних входів-виходів. Інша проблема – робочий діапазон. Для ПЛК100 ОВЕН робочі умови експлуатації це: відносна вологість 0-80%, температура 0-60 °С; атмосферний тиск 84-106,7 кПа; вибухобезпечне приміщення без агресивних випарів та газів. Нормальні умови експлуатації здебільшого дуже схожі. Окремо варто додати, що правильність роботи ПЛК не гарантується в умовах високого і агресивного магнітного поля. Якщо об’єднати ці умови, то контролер просто замінює людину на монотонній роботі, гарантуючи більшу точність в рамках програми.

Окремо варто згадати про деякі особливості експлуатації, а саме пункт «ремонт». Ремонту після фізичних пошкоджень ПЛК підлягають в виключно рідкісних випадках. Але це дійсно лише у випадку ремонту контролеру компанією-виробником в рамках гарантійного терміну або послуг технічної підтримки.

4.4. **Обслуговування і програмування ПЛК**

Як і будь-яка комп’ютеризована техніка контролер для своєї роботи потребує правильного програмного забезпечення. Якщо файли для оновлення вбудованого програмного забезпечення є індивідуальним для кожного виробника та кожної окремої лінійки моделей ПЛК, то середовище для запису програм керування виробництвом є більш уніфікованими. Прикладом такого програмного середовища є easySoftCoDeSys. Дане середовище дозволяє досить широко створювати програми для реального або віртуального виробництва. Крім того загальний об’єм файлів з програмою є досить незначним. Зазвичай об’єм файлу з програмою не перевищує 1 мБ.

Саме середовище CoDeSys дає змогу використовувати в програмах цілий ряд інструментів розробника. Здебільшого ці інструменти дозволяють запам’ятовувати певні дії, що надалі дає змогу задати певну кількість спрацювань до зміни команди, або відраховувати певний час, опираючись на вбудований в ПЛК годинник реального часу.

Висновки до розділу 4

1) Проведений аналіз баз даних, та можливість їх використання для зберігання інформації про добові раціони сільськогосподарських тварин.

   Показана модель взаємодії та обміну інформацією між базою даних та керуючим модулем.
2) Проведений аналіз систем управління базами даних SQLite та MySQL.
3) Оцінено різницю продуктивності при читанні та записування малих та великих об’єктів. Визначено розмір сторінки бази даних, що дає найкращу продуктивність для операцій введення-виведення великих двійкових об'єктів.
4) Проведений аналіз сучасних програмованих логічних контролерів та визначена доцільність їх використання, для реалізації розробленої системи автоматизації.

ЗАГАЛЬНІ ВИСНОВКИ

1. В результаті проведеного аналізу сучасних методів вигодовування тварин, розробки раціонів та процесу формування кормових сумішей було визначено, що сучасна система вигодовування тварин орієнтована на добову норму, що змінюється один раз на місяць.
2. Розроблено алгоритм внесення інформації про щомісячні добові норми кормів до масиву даних в енергонезалежну пам’ять для подальшої їх обробки. Описано методику збереження та завантаження масиву до енергонезалежної пам’яті.
3. Представлена математична модель для розрахунку добової норми кожного виду корму для тварин, що внесені у базу даних. Модель дозволяє розраховувати щоденну кількість раціону кормів на кожен день року, що дозволить зробити перехід від однієї місячної норми до іншої поступовим.
4. Побудована макетна модель для автоматизованого розрахунку та змішування кормів для різних тварин на кожен окремий день року. Представлена схема розташування контейнерів з кормами та алгоритм, необхідний для правильного спрацювання розробленої системи автоматизації. Відпрацьовані програми, що діють по представленим алгоритмам.
5. Розглянуті програмні та апаратні компоненти, що можна використати для промислової реалізації розробленої автоматизованої системи. Показаний метод зберігання інформації про кормові норми за допомогою баз даних. Представлена промислова альтернатива керуючого модуля у вигляді програмованого логічного контролера.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бергнер Х. Научные основы питания сельскохозяйственных животных / Х. Бергнер, Х.А. Кетц – М.: Колос, 1973. – 597 с.
2. Мосолов Н.Д. Кормление сельскохозяйственных животных: [Учеб.пособие] / Н.Д. Мосолов, Л.А. Билый – К.: Вища шк., 1990. – 359 с.
3. Бомко В.С., Годівля сільськогосподарських тварин: Підручник / В.С. Бомко, С.П. Бабенко О.Ю. Москалик - К., 2010. 278с.
4. Єгоров Б.В., Шаповаленко О. І., Макаринська А.В. Технологія виробництва преміксів: Підручник [Текст] / Б.В. Єгоров, О. І. Шаповаленко, А.В. Макаринська - К.: Центр учбової літератури, 2007. 288 с.
5. НУБІП України. Серія: Технологія виробництва і переробки продукції тваринництва. 2017. Т. 9 (№5-6). С. 68-75.
6. Васильев, А.Е. Микроконтроллеры. Разработка встраиваимых приложений / А.Е. Васильев. - СПб.: BHV, 2012. - 304 с.
7. Хофманн, М. Микроконтроллеры для начинающих / М. Хофманн. - СПб.: BHV, 2013. - 304 с.
8. Документация для микроконтроллера ArduinoUno. [Электронный ресурс]. – Режим доступа: http://arduino.ru/Hardware/ArduinoBoardUno
9. Довідова стаття EEPROM Library [Электронный ресурс]. – Режим доступа: https://www.arduino.cc/en/Reference/EEPROM - EEPROM Library
10. Петин В. А. Проекты с использованием контроллера Arduino [Текст] / В. А. Петин. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2016. — 464 с
11. Белов А.В. Программирование микроконтроллеров для начинающих и не только. Книга + виртуальный диск / А.В. Белов. - СПб.: Наука и техника, 2016. - 352 с.
12. Иванов В.Б. Программирование микроконтроллеров для начинающих: Визуальное проектирование / В.Б. Иванов. - СПб.: Корона-Век, 2010. - 176 с.
13. Программирование микроконтроллера ArduinoUno [Электронный ресурс]. – Режим доступа: http://arduino.ru/reference

14. Соммер У. Программирование микроконтроллерных плат Arduino / Freeduino [Текст] // У. Соммер. — СПб.: БХВ — Петербург, 2012. — 256 с.
15. Магда, Ю.С. Программирование и отладка C/C++ приложений для микроконтроллеров ARM / Ю.С. Магда. - М.: ДМК, 2014. - 168 с.
16. Иванов, В.Б. Программирование микроконтроллеров для начинающих. Визуальное проектирование, язык С, ассемблер / В.Б. Иванов. - СПб.: КОРОНА-Век, 2015. - 176 с.
17. Архангельский А.Я. Приемы программирования в C++ Builder / А.Я. Архангельский — М.: Бином-Пресс, 2010. — 896
18. Саттер Герб Новые сложные задачи на плате Ардуино / Саттер Герб - Вильямс, 2017г – 272 с.
19. Овсяницкая, Л.Ю. Курс программирования робота EV3 в среде Lego Mindstorms EV3/ Л.Ю. Овсяницкая, Д.Н. Овсяницкий, А.Д. Овсяницкий. 2-е изд., перераб. и доп – М.: Издательство «Перо», 2016. – 300 с.
20. Петин В. А., Биняковский А. А., Практическая энциклопедия Arduino / В. А. Петин, А. А Биняковский - изд. ДМК-Пресс, 2017 г.стр.152-163.
21. Блум Д. Изучаем Arduino: инструменты и методы технического волшебства / Д. Блум: Пер. с англ. — СПб.: БХВ-Петербург, 2012. — 336 с.:
22. Тунеев М.М., Сухоруков В.Ф. Экономико-математические методы в организации и планировании сельскохозяйственного производства / М. М. Тунеев, В.Ф. Сухоруков– М.: Финансы и статистика», 1986 – стр. 50.
23. Монк С. Программируем Arduino. Профессиональная работа со скетчами / С. Монк — СПб.: Питер, 2017. — 272 с.:
24. Бачинин А. Основы программирования микроконтроллеров: Учебно – методическое пособие к образовательному набору по микроэлектронике «Амперка» / А. Бачинин // образовательный робототехнический модуль, под редакцией Сергея Косаченко — М.: Издательство «Экзамен», 2017. — 184 с.
25. Ножка С. І., Пилипенко Ю. М. Автоматизовані системи подачі кормів при вигодовуванні свійських тварин. Технології та дизайн. 2020. №4

26. Красс, М. С. Моделирование эколого-экономических систем / М.С. Красс. - М.: ИНФРА-М, 2013. - 272 с.
27. Коробов, П. Н. Математическое программирование и моделирование экономических процессов / П.Н. Коробов. - М.: ДНК, 2015. - 376 с.
28. Юревич Е. И. Основы робототехники [Текст] / Е. И. Юревич. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2005. — 416 с.
29. Бербюк, В. Е. Динамика и оптимизация робототехнических систем / В.Е. Бербюк. - М.: Наукова думка, 2014. - 192 с
30. Бейктал, Дж. Конструируем роботов на Arduino. Первые шаги / Дж. Бейктал. - М.: Лаборатория знаний, 2016. - 320 с.
31. LoC Recommended Storage Format. [Электронный ресурс]. – Режим доступа: https://www.sqlite.org/locrsf.html
32. SQLite As An Application File Format [Электронный ресурс]. – Режим доступа: https://www.sqlite.org/appfileformat.html
33. Internal Versus External BLOBs in SQLite [Электронный ресурс]. – Режим доступа: https://sqlite.org/intern-v-extern-blob.html

Додаток А

```
#include <avr/eeprom.h>
#include <Servo.h>
Servo myservo;
int pos = 0;
int listener = 0;
bool info = 0;
bool isInit = 0;
int selectedMonth = 0;
int selectedDay = 0;
int selectedAnimal = 0;
const int numberOfAnimal = 3;
const int numberOfFood = 4;
float foodDayRation[numberOfAnimal][numberOfFood] = {0};
int EEMEM intArrayAddr[numberOfAnimal][12][numberOfFood];
byte EEMEM check_addr;
int ration[numberOfAnimal][12][numberOfFood]=
{
  { {0, 2450, 3200, 4500},
    {0, 1000, 3020, 4500},
    {0, 1950, 2700, 3800},
    {1500, 1900, 3730, 4450},
    {1100, 2450, 3200, 4500},
    {1200, 2200, 3520, 4950},
    {1500, 2450, 3840, 4210},
    {0, 2380, 3340, 4340},
    {0, 2920, 3980, 4640},
    {1900, 2340, 3340, 4090},
    {1100, 2260, 3650, 4720},
  {1000, 2000, 3000, 4000} },
```

```
 { {14, 25, 36, 48},
   {16, 24, 32, 45},
   {0, 29, 33, 44},
   {0, 24, 34, 46},
   {15, 21, 39, 48},
   {14, 22, 34, 42},
   {16, 0, 0, 44},
   {13, 0, 0, 43},
   {15, 0, 0, 45},
   {18, 29, 35, 41},
   {19, 22, 39, 42},
  {11, 28, 35, 49} },

 { {121, 256, 353, 442},
   {198, 234, 391, 0},
   {135, 227, 353, 0},
   {194, 241, 374, 0},
   {165, 293, 315, 496},
   {174, 246, 353, 456},
   {185, 234, 394, 474},
   {162, 268, 371, 453},
   {0, 0, 323, 484},
   {0, 0, 394, 419},
   {194, 226, 323, 427},
  {185, 223, 394, 465} }


};
```

```
int daysInMonth[12] = {31,28,31,30,31,30,31,31,30,31,30,31};

void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  myservo.attach(9);
}

void loop()
{
  myservo.write(pos);
  if (isInit == 0)
  {
    initialization();
  }
  menu();
}

void initialization()
{
  byte checkByte = 0;
  checkByte = eeprom_read_byte(&check_addr);
  if (checkByte == 1)
  {
    Serial.println("Saved ration detected in EEPROM");
    Serial.println("Do you want to load it?");
    Serial.println("1- Yes");
    Serial.println("0- No(Use default recion template)");
```

```
    while (Serial.available() == 0);
    int toCheck = Serial.parseInt();
    if (toCheck == 1)
    {
      int temp[numberOfAnimal][12][numberOfFood];
      eeprom_read_block((void*)&temp, (const void*)&intArrayAddr, sizeof(temp));
      for(int x = 0; x < numberOfAnimal; x++)
      {
        for(int y = 0; y < 12; y++)
        {
           for(int z = 0; z < numberOfFood; z++)
          {
              ration[x][y][z] = temp[x][y][z];
          }
        }
      }
    }
  }
  isInit = 1;
}

void menu()
{
  if(info == 0)
  {
    Serial.println("Enter 1 to enter the date selection menu"); // Меню выбора даты(Месяц, день)
    Serial.println("Enter 2 to calculate day ration"); // Расчет дневного рациона для каждого животного на текущий день
```

```
  Serial.println("Enter 3 to print calculated ration for current day"); // Показать расчитанный рацион для каждого животного на текущий день
  Serial.println("Enter 4 to enter the ration change menu"); // Меню изменения массива рационов
  Serial.println("Enter 5 to show all rations"); // Отобразитть массив рационов
  Serial.println("Enter 6 to save ration to EEPROM"); // Сохранение массива рационов в енергонезависимую память
  Serial.println("Enter 7 to load ration from EEPROM"); // Загрузка массива рационов из енергонезависимой памяти
  Serial.println("Enter 8 to start rotor"); // Запуск двигателя
  info = 1;
 }
 while (Serial.available() == 0);
 int command = Serial.read();
 switch (command)
  {
   case '1':
    Serial.println("Date selection");
    dateSelection();
    break;
   case '2':
    Serial.println("Calculating");
    calcCurrDayRation();
    break;
   case '3':
    Serial.println("RationForDay");
    printRationForDay();
    break;
   case '4':
    Serial.println("rationChangeMenu");
```

```
      rationChange();
      break;
    case '5':
      Serial.println("ration");
      rationShow();
      break;
    case '6':
      Serial.println("saveRation");
      saveRation();
      break;
   case '7':
      Serial.println("loadRation");
      loadRation();
      break;
   case '8':
      Serial.println("rotor");
      rotor();
      break;
   }
}

void rotor()
{
  for(int i = 0; i < numberOfAnimal; i++)
  {
   digitalWrite(13, HIGH);
   delay(4000);
   digitalWrite(13, LOW);
   for(int j = 0; j < numberOfFood; j++)
   {
```

```
      if(foodDayRation[i][j] > 0)
      {
        delay(1000);
       myservo.write(100);
  delay(foodDayRation[i][j]*10);
       myservo.write(0);
        delay(1000);
      }
      if(j < numberOfFood-1)
      {
        digitalWrite(13, HIGH);
        delay(2000);
        digitalWrite(13, LOW);
      }
      myservo.write(pos);
    }
  }
         delay(1500);
        digitalWrite(12, HIGH);
        delay(32000);
        digitalWrite(12, LOW);

  info = 0;
}

void loadRation()
{
  int temp[numberOfAnimal][12][numberOfFood];
  eeprom_read_block((void*)&temp, (const void*)&intArrayAddr, sizeof(temp));
  for(int x = 0; x < numberOfAnimal; x++)
```

```
    {
     for(int y = 0; y < 12; y++)
     {
       for(int z = 0; z < numberOfFood; z++)
     {
         ration[x][y][z] = temp[x][y][z];
     }
      }
    }
  info = 0;
}

void saveRation()
{
  eeprom_write_byte(&check_addr, 1);
  eeprom_write_block((void*)&ration, (void*)&intArrayAddr, sizeof(ration));
  info = 0;
}

void printRationForDay()
{
  for(int animalType = 0; animalType < numberOfAnimal; animalType++)
    {
     Serial.print("Animal type:");
     Serial.println(animalType + 1);
     for(int rationType = 0; rationType < numberOfFood; rationType++)
     {
         Serial.print("Ration type:");
         Serial.println(rationType + 1);
         Serial.print("Value:");
```

```
        Serial.println(foodDayRation[animalType][rationType]);
       }
    }
  info = 0;
}

void calcCurrDayRation()
{
  float dayRation = 0;
  for(int animalType = 0; animalType < numberOfAnimal; animalType++)
  {
    for(int rationType = 0; rationType < numberOfFood; rationType++)
    {
      if(selectedMonth == 12)
      {
        dayRation = float(ration[animalType][selectedMonth -1][rationType])
          +(float(ration[animalType][0][rationType]
            -                    ration[animalType][selectedMonth                    -
1][rationType])/float(daysInMonth[selectedMonth - 1]))
          * float(selectedDay);
      }
      else
      {
        dayRation = float(ration[animalType][selectedMonth -1][rationType])
          +(float(ration[animalType][selectedMonth ][rationType]
            -                    ration[animalType][selectedMonth                    -
1][rationType])/float(daysInMonth[selectedMonth - 1]))
          * float(selectedDay);
      }
      foodDayRation[animalType][rationType] = dayRation;
```

```
    }
  }
  info = 0;
}

void rationShow()
{
  for(int x = 0; x < numberOfAnimal; x++)
  {
    Serial.print("Animal: ");
    Serial.print(x);
    Serial.println();
    for(int y = 0; y < 12; y++)
    {
       for(int z = 0; z < numberOfFood; z++)
      {
         Serial.print(ration[x][y][z]);
         Serial.print("   ");
      }
      Serial.println();
    }
  }
  info = 0;
}

void rationChange()
{
  int changedAnimal = 0;
  int changedMonth = 0;
  int changedFood = 0;
```

```
Serial.println(" Choose type of animal to change ");
Serial.println(" 1- Cow");
Serial.println(" 2- Chicken");
Serial.println(" 3- Pigs");
while (Serial.available() == 0);
changedAnimal = Serial.parseInt();

Serial.println(" Choose the month to change  ");
Serial.println(" 1- January");
Serial.println(" 2- February");
Serial.println(" 3- March");
Serial.println(" 4- April");
Serial.println(" 5- May");
Serial.println(" 6- June");
Serial.println(" 7- Jule");
Serial.println(" 8- August");
Serial.println(" 9- September");
Serial.println(" 10- October");
Serial.println(" 11- November");
Serial.println(" 12- December");
while (Serial.available() == 0);
changedMonth = Serial.parseInt();
delay(100);

Serial.print("Choose type of food to change between 1 and ");
Serial.println(numberOfFood);
while (Serial.available() == 0);
changedFood = Serial.parseInt();

Serial.print("Current value of[Animal-");
```

```
  Serial.print(changedAnimal);
  Serial.print(", Month-");
  Serial.print(changedMonth);
  Serial.print(", Food-");
  Serial.print(changedFood);
  Serial.print("] is:");
  Serial.print(ration[changedAnimal-1][changedMonth-1][changedFood-1]);
  Serial.println(".  Type new value");
  while (Serial.available() == 0);
  int changedValue = Serial.parseInt();
  ration[changedAnimal-1][changedMonth-1][changedFood-1] = changedValue;
  Serial.println("Value succesffully changed");

  info = 0;
}

void dateSelection()
{
  Serial.println(" Choose the required month ");
  Serial.println(" 1- January");
  Serial.println(" 2- February");
  Serial.println(" 3- March");
  Serial.println(" 4- April");
  Serial.println(" 5- May");
  Serial.println(" 6- June");
  Serial.println(" 7- Jule");
  Serial.println(" 8- August");
  Serial.println(" 9- September");
  Serial.println(" 10- October");
  Serial.println(" 11- November");
```

```
 Serial.println(" 12- December");
 while (Serial.available() == 0);
 selectedMonth = Serial.parseInt();
 delay(100);

 Serial.print("Choose the required day between 1 and ");
 Serial.println(daysInMonth[selectedMonth - 1]);
 while (Serial.available() == 0);
 selectedDay = Serial.parseInt();

info = 0;
}
```