Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Програмно-апаратне забезпечення комп'ютерної системи контролю процесу ферментації винних продуктів

Виконав(ла): студент(ка) *VI* курсу, групи *СІм-61*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

(підпис) Ольховецька Х.А. (прізвище та ініціали)

Керівник (підпис) Осухівська Г.М. (прізвище та ініціали)

Нормоконтроль (підпис) Тиш Є.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Осухівська Г.М. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
« ____ » _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *магістр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту Ольховецькій Христині Андріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Програмно-апаратне забезпечення комп'ютерної системи контролю процесу ферментації винних продуктів

Керівник роботи Осухівська Галина Михайлівна, кандидат технічних наук, зав. кафедри КС
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*06*» *12* *2022* року № *4/7-986*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Процес ферментації винних продуктів та потреба його контролю

4. Зміст роботи (перелік питань, які потрібно розробити)
   1. Аналіз предметної області та огляд існуючих систем контролю процесу ферментації винних продуктів
   2. Математичне забезпечення КС та мат. модель процесу ферментації
   3. Розробка та тестування апаратно-програмної комплексу системи
   4. Охорона праці та безпека в надзвичайних ситуаціях;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
   1. Тема та мета роботи; 2. Актуальність роботи; 3. Новизна, об'єкт та мета дослідження;
   5. Структурна схема; 5. Пристрій; 6. Графіки; 7. Мат. модель; 8. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в	Осухівська Г.М., зав. каф. КС		
надзвичайних ситуація	Клепчик В.М., проректор з		
	адміністративно-господарської		
	роботи та будівництва		

7. Дата видачі завдання 15.11.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	14.11.2022	Виконано
2	Аналіз завдання	15.11.2022	Виконано
3	Виконання розділу 1	16.11.2022	Виконано
4	Виконання розділу 2	20.11.2022	Виконано
5	Виконання розділу 3	30.11.2022	Виконано
6	Виконання розділу 4	03.12.2022	Виконано
7	Оформлення пояснювальної записки	07.12.2022	Виконано
8	Оформлення графічного та презентаційного матеріалу	13.12.2022	Виконано
9	Перевірка роботи на антиплагіат	15.12.2022	Виконано
10	Попередній захист	16.12.2022	Виконано
11	Захист	21.12.2022	Виконано

Студент _____ Ольховецька Х. А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Осухівська Г.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Програмно-апаратне забезпечення комп’ютерної системи контролю процесу ферментації винних продуктів // Кваліфікаційна робота магістра // Ольховецька Христина Андріївна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних систем та мереж, група СІм-61 // Тернопіль, 2022 // с. – 52, рис. – 17, табл. – , аркушів А1 – 8, додат. – 2, бібліогр. – 35.

Ключові слова: система керування, ферментація, датчик, Internet of Things, LoRa, Mesh, nRF, математична модель.

Кваліфікаційна робота присвячена питанню розроблення програмно-апаратних засобів для побудови системи контролю за процесом ферментації винних продуктів на базі технології LoRa. Обґрунтовано вибір технології передачі даних для системи керування. Синтезовано структуру автоматизованої системи керування. Здійснено аналіз класичних математичних моделей процесу ферментації та на основі цього було її доповнено та розроблено математичне забезпечення КС. Здійснено вибір компонентів проектованої системи та розроблено апаратне забезпечення модуля для керування процесу ферментації. Розроблено алгоритмічне та програмне забезпечення для системи керування процесу ферментації. Впровадження запропонованої автоматизованої системи керування процесу ферментації винних продуктів дозволить зменшити відсоток браку на виробництві та підвищити ймовірність запобігання його утворення враховуючи смертність дріжджів та показників які на це впливають.

ANNOTATION

Software and hardware of the computer system for controlling the wine products fermentation process // Master's qualification work // Olkhovetska Khrystyna Andriivna // Ivan Pulyuy Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Systems and networks, SIM-61 group // Ternopil, 2022 // p. – 52, fig. - 17, tab. – , sheets A1 – 8, add. – 2, bibliography - 35.

Keywords: control system, fermentation, sensor, Internet of Things, LoRa, Mesh, nRF, mathematical model.

The qualification work is devoted to the development of software and hardware tools for building a control system for the fermentation process of wine products based on LoRa technology. The choice of data transmission technology for the control system is justified. The structure of the automated control system has been synthesized. An analysis of classical mathematical models of the fermentation process was carried out, and on the basis of this, it was supplemented and mathematical support of CS was developed. The components of the designed system were selected and the hardware of the module for controlling the fermentation process was developed. Algorithmic and software for the control system of the fermentation process have been developed. Implementation of the proposed automated control system for the fermentation process of wine products will allow to reduce the percentage of defects in production and increase the probability of preventing its formation, taking into account the mortality of yeast and indicators that affect it.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КС – комп’ютерна система

MAC – Medium access control

Phy – Physical layer

LoRa – Long Range Wide Area

SoC – System On a Chip

CMSIS – Cortex-M Microcontroller Software Interface Standard

CPU – central processing unit (центральний процесор)

SPI – Serial Peripheral Interface (послідовний периферійний інтерфейс)

PWM – pulse-width modulation (широтно-імпульсна модуляція) SCLK – Serial Clock

I2C – Inter-integrated Circuit

ФВН – Ферментація винних продуктів

КСК – Комп’ютерна система контролю

ОС – операційна система

ЗМІСТ

ВСТУП

Актуальність роботи. На сучасних виноробних великих підприємствах використовуються та запроваджуються сучасні автоматизовані лінії, лабораторії, системи моніторингу тих чи інших параметрів, а також використовуються різноманітні системи організації роботи підприємств. Важливим технологічним процесом виноробства, який впливає на якість кінцевого продукту і який доцільно було б постійно моніторити, є ферментація.

Дослідженнями в тематиці ферментації займались D. Dochain, R. David, A. Vande Wouwer, J.-R. Mouret, J.-M. Sablayrolles, S. A. J. Comfort проводив наукові дослідження щодо впливу окиснення на процес бродіння, науковці L. Michaelis and M. L. Menten займались дослідженнями кінетики ферментації та проблемою математичного моделювання для даного процесу. Але в їх дослідженнях не враховано вплив показників температури на утворення CO_2.

Крім того варто відмітити, що існуючі системи є досить дорогими, що унеможливлює їх використання для малого бізнесу. Тому розробка бюджетної комп'ютерної системи контролю процесу ферментації винних продуктів побудованої на базі вдосконаленої математичної моделі в якій враховано показники температури та CO_2 є актуальною задачею.

Мета роботи – розробка комп'ютерної системи для контролю процесу ферментації винних продуктів на базі мікроконтролера nRF.

В роботі необхідно розв'язати такі задачі:

— провести огляд існуючих сучасних систем, які використовуються у виноробстві, зокрема, для контролю процесу ферментації;

— сформувати вимоги до нової комп'ютерної системи з врахуванням переваг та недоліків попередньо досліджених систем;

— проаналізувати відомі моделі, які описують процес ферментації з метою використання чи вдосконалення їх для побудови комп'ютерної системи;

— обґрунтувати математичну модель процесу ферментації винних продуктів, на основі якої буде розроблена комп'ютерна система;

— розробити програмне та апаратне забезпечення комп’ютерної системи для контролю за технологічним процесом ферментації винних продуктів та провести тестування.

Новизна одержаних результатів.

— Запропоновано вдосконалену математичну модель контролю ферментації, яка враховує важливі параметри технологічного процесу виготовлення винних продуктів, що дозволило розробити комп’ютерну систему.

— На основі запропонованої моделі розроблено комп’ютерну систему контролю процесу ферментації на базі мікроконтролера nRF, яка є недорогою та може бути використана на вітчизняних малих виноробних підприємствах, зокрема, на старих виробничих потужностях.

Об’єкт досліджень – процес ферментації винних продуктів.

Предмет досліджень – комп’ютерна система контролю процесу ферментації винних продуктів.

Практичне значення дослідження полягає в тому, що представлена комп’ютерна система та її програмно-апаратна реалізація можуть використовуватись на виноробнях для контролю за процесом ферментації з метою покращення якості продукції.

Публікації. Дані досліджень були апробовані на трьох наукових конференціях.

.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ СИСТЕМ КОНТРОЛЮ ПРОЦЕСУ ФЕРМЕНТАЦІЇ ВИННИХ ПРОДУКТІВ

1.1. Огляд та аналіз процесу ферментації

Класичні методи виноробства почали розвиватись 7000 років тому, коли люди навчилися вирощувати виноград. З того часу процес постійно вдосконалювався, а рецепти та методи ускладнювалися, але основними етапами технічного процесу виробництва, як і раніше, залишаються збір урожаю, отримання сусла, ферментація та дозрівання.

На заводах шляхом бродіння виробляються такі види вина: класичне виноградне вино. вина сортові (купажовані та ізольовані), родзинки (з родзинок), фрукти (з яблучного та грушевого соку), ягоди. Залежно від способу приготування та обробки напоїв із будь-якої сировини може бути якісним, але справжнім вином вважається лише виноградне вино.

Класичне виноградне вино буває червоне, біле та рожеве - це залежить від сорту винограду та особливостей бродіння. Червоне вино отримують шляхом бродіння подрібненого чорного винограду. Всесвітньо відомі виноробні у Франції, Італії, Іспанії, Чилі, Новій Зеландії, Південній Африці, Австралії та США мають технології та перевірені методи витримки, що дозволяють виробляти кращі фірмові червоні вина. Біле та рожеве вино зазвичай роблять із соку білих сортів, хоча також використовують червоний виноград дотримуючись специфічного технологічного процесу.

Технологічний процес створення вина охоплює багато процесів, які можна поділити на розчавлення ягід винограду, а після цього відбувається ферментація мязги та сусла, що складає спиртове бродіння, розпад дріжджів та яблучно-молочне бродіння.

Чітко розмежувати дані стадії важко, оскільки вони взаємопов’язані, проте кожен з них має характерні особливості для всіх процесів, а саме технологічних, біохімічних та фізико-хімічних.

Кожна стадія процесу перетворення винограду в вино вимагає постійного контролю та керування, бо це напряму впливає на якість вина при завершенні дозрівання.

Великий вплив на якісне зберігання вин та правильні процеси окислення при дозріванні вина має такий показник як температура повітря. Оскільки при низьких температурах вино дозріває більш повільно, але при тому отримує більш тонкий та насичений смак та аромат. І навпаки, коли ферментація вина відбувається за високих температур, також ці умови сприяють розвитку хворобоутровюючих мікроорганізмів.

Різка зміна температури несприятливо впливають на процес дозрівання вина, результатом чого буде помутніння або ж нерівномірна зміна кольору [9].

Великий вплив на вино в процесі його витримки має вологість приміщення де відбувається цей процес, якщо повітря є сухим – відбувається випаровування, в випадку підвищеної вологості - розвивається пліснява, яка заражає вино. В приміщеннях, де бродить і зберігається вино, повинні регулярно провітрюватись, оскільки повітря повинно бути чистим, бо це може погано вплинути на якість продукту, що зумовлено високою поглинаючою властивістю сировини.

Вино є продуктом складних взаємодій між дріжджами, бактеріями та іншими грибами, які починаються на виноградниках і продовжуються процесом бродіння. Різні види дріжджів знаходяться на поверхні виноградних шкірок і в виноробному середовищі [2], і S.Cerevisiae визнаний основним мікроорганізмом, відповідальним за цей процес [4].

Виноробство – це сезонна сільськогосподарська практика, яка використовує традиційні методи виробництва, спеціальні приміщення та спеціалізоване обладнання. Процеси виробництва обертаються навколо збору врожаю винограду, що відбувається протягом короткого періоду восени, але виноробня активна цілий рік. Виноград транспортують на виноробню, подрібнюють, пресують, ферментують і витримують – традиційно в дубових бочках перед розливом. Кожен з цих етапів

передбачає набір спеціалізованого обладнання для обробки продукту при його поступовому переході від винограду до вина, і відбувається на виділених ділянках виноробні. Ці зони обробки часто підтримуються в умовах температури/навколишнього середовища з явною метою управління мікробною активністю під час ферментації та дозрівання.

Вплив вибраної кількості бактерій і грибів на якість вина добре встановлено, включаючи як корисні, так і згубні ролі, що описано в роботах [9], [10], [12]. Окрім інокуляції дріжджів Saccharomyces для ініціювання ферментації, кілька інших видів дріжджів визнані за їх роль у бродінні. Неінокульовані, «дикі» дріжджі вважаються такими, що підсилюють «складність» бродіння вина за рахунок отримання більш широкого спектру сенсорно-активних сполук, ніж той, що утворюється тільки з чистого інокулята [10]. Oenococcus oeni або інші види молочнокислих бактерій (LAB) також часто інокулюють для виконання вторинного, малолактичного бродіння, але в іншому роль бактерій у бродінні вина, як правило, вважається згубною [12]. Ряд диких дріжджів і бактерій вважаються псують організми при винних ферментаціях, знижуючи кінцеву якість за рахунок отримання офф-смаків, серпанків, карбонізації або інших дефектів [13], [11]. Походження більшості мікробів у винних ферментаціях мало вивчено і, як правило, передбачається з винограду [4]. Однак первинні мікроби, що беруть участь як позитивно, так і негативно у винних ферментаціях, включаючи Saccharomyces, Brettanomyces та Oenococcus oeni, виявляються лише як незначні популяції на поверхні здорового винограду[4], [5], [6]. Ще одним переважаючим джерелом для перенесення цих мікробів між бродіннями є сама виноробне середовище [3].

Обладнання для виноробних виробів, включаючи обладнання для дроблення/пресування та бочки, часто включає важко очищені, пористі поверхні. Специфічні штами Saccharomyces можуть знаходитись на поверхнях виноробні, що призводить до повторного виявлення протягом декількох років у неінокульованих винах [7], [8], [11], [10]. Поширення грибів, що не є Saccharomyces, на конкретних поверхнях виноробні в обмеженій мірі вивчено в ізольованих часових точках за допомогою методів на основі культури [2], [3], [13]. Повітряно-крапельні популяції

LAB [14], дріжджів, що не є цвіллю [11], [15], [17] також були досліджені на виноробнях за допомогою методів на основі культури. Однак джерело корінних мікробів у винних ферментаціях залишається дуже спірною темою, особливо щодо організмів, пов'язаних зі псуванням [14], [15], [17], і всі попередні дослідження поверхневої мікробіоти виноробні спиралися на методи на основі культури, які схильні до упереджень для вивчення мікробіоти бродіння харчових продуктів [16], [17] . Більш того, більшість досліджень виноробних середовищ були зосереджені на мікроорганізмах, раніше культивованих з винних бродінь. Повна часовопросторова екологія у виноробних підприємствах значною мірою невідома, і це вимагає дослідження, враховуючи роль, яку відіграють багато неінокульованих видів у формуванні якісних характеристик вина, а також псування вина.

1.2. Дослідження впливу температури на процес ферментації

Бродіння - це біохімічний процес, який перетворює цукор в етанол і вуглекислий газ. Під поняттям «бродіння» зазвичай розуміють процес дріжджового бродіння цукру до етанолу, але це поняття також охоплює інші види мікробного бродіння.

Дріжджі, що використовуються при бродінні вина, в основному належать до виду Saccharomyces cerevisiae, і цей процес відповідає за багато бажаних характеристик у вині, таких як його аромат, смак і відчуття рота.

Коли починається бродіння, дріжджі перетворюють цукор в спирт і вуглекислий газ. У деяких випадках він також може давати новий смак. Для того, щоб мати успішний процес бродіння, для нього повинна бути ідеальна температура. Ідеальна температура бродіння вина становить близько 18-26 °C (64-78 °F).

Час бродіння є одним з компонентів того, що визначає смак, аромат і колір вина. У випадку коли час бродіння занадто тривалий смак вина змінюється і нагадує оцет або соління. І навпаки смак вина буде прісним і без будь-яких сильних ароматів, якщо вино не ферментується досить довго. Час аерації залежить від бажаного кольору у

вині при бродінні. Темніші кольори досягаються при більшій аерації, тоді як менша аерація призведе до вин світлого кольору.

Температури бродіння є найважливішою частиною процесу виноробства. Вони надають вину його складність, збалансованість, аромат.

Бродіння відбувається, коли дріжджі мають доступ до достатньої кількості «їжі» – зазвичай у вигляді цукрів – для швидкого розмноження. Дріжджі виробляють молочну кислоту, оцтову кислоту або спирт, а вуглекислий газ - як продукти життєдіяльності.

На процес бродіння температура впливає по-різному. Якщо бродіння відбувається при високих температурах, дріжджі будуть активними, але вони не будуть виробляти багато алкоголю або аромату, тому що їх швидкість росту буде занадто швидкою, щоб дозволити цим речам розвиватися належним чином. У випадку, якщо дотримуватись більш холодних температур під час бродіння ріст бактерій буде пригнічуватись, що призведе до псування вина, що надасть сировині відповідний присмак та запах.

1.3. Аналіз існуючих рішень

Сучасні великі винні виробництва мають лабораторії, які здійснюють контроль за продукцією на різних етапах процесу виготовлення. Найчастіше на виноробнях використовують ферментатори або вініфікатори, проте вони, зазвичай, не передбачають керування з використанням давачів чи інтеграції програмного засобу для моніторингу та служать ємністю з ручним контролюванням та відбором проб.

1.3.1. Система моніторингу ферментації WINEGRID

Одним з найяскравіших прикладів систем моніторингу процесу бродіння є продукція компанії Winegrid, що розробляє та надає інтегроване віддалене рішення в режимі реального часу для інтелектуального моніторингу процесу виноробства. Компанія надає власну технологію, що складається з апаратного забезпечення

(датчики), обчислювальної платформи з механізмом штучного інтелекту та платформи візуалізації [12].

Система моніторингу ферментації (FMS), розроблена Португальською компанією WINEGRID і дозволяє здійснювати [12]:

— автоматичне визначення початку бродіння;

— автоматичне визначення кінетики бродіння;

— підвищення температури для підвищення кінетики бродіння;

— зменшення щільності.

На рис. 1.1 показано загальний вигляд давача, оптимізованого для щільності та температури бочок [12].

Рис. 1.1 FMS - давач, оптимізований для щільності та температури бочок

Перевагами такої системи є:

— точність;

— відмовостійкість;

— автоматизоване керування температурою;

— використання власної технології Oenosensing (запатентована і визнана технологія, яка дозволяє вимірювати температуру, щільність і рівень рідини в режимі реального часу під час процесу бродіння.);

— доступ до даних про ферментацію через власну платформу.

Незважаючи на переваги даного рішення система має і ряд недоліків:

— неможливість використання системи в старих бочках;

— вартість.

Саме ці недоліки є ключовими і не дозволяють FMS WINEGRID використовувати для потреб вітчизняних малих виноробних підприємств.

1.3.2. Винний монітор 5500/5600 компанії Anton Paar

Wine Monitor 5500 і Wine Monitor 5600 є приладами для постійного контролю вмісту алкоголю, екстракту, густини та концентрації CO_2 у всіх винах – від червоних, білих і рожевих вин до винних сумішей. Передбачено також можливість визначення кольору і кисню.

Загальний вигляд Wine Monitor 5500 і Wine Monitor 5600 компанії Anton Paar зображено на рис. 1.2 [3].

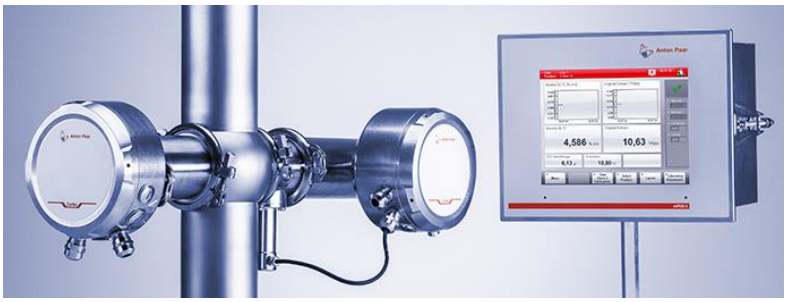

Рис. 1.2 Винний монітор 5500/5600 компанії Anton Paar

Недоліком таких систем є те, що вони призначені для використання на магістралях, а не для бочок, також монітор забезпечує лише зчитування показників давачів та їх передавання на платформу дистанційного доступу, при цьому система не здійснює контроль за процесом ферментації. Ще одним важливим недоліком є вартість такої системи. Саме ці недоліки Wine Monitor 5500/5600 компанії Anton Paar і обмежують їх використання.

1.3. Висновки до розділу 1

В цьому розділі було досліджено процес ФВН, в результаті аналізу літературних джерел встановлено, що ферментація є важливим технологічним процесом у виноробстві і потребує постійного моніторингу.

Проаналізовано сучасні системи контролю винного виробництва і встановлено їх недоліки, які обмежують їх використання.

Обґрунтовано актуальність та доцільність розробки комп’ютерної системи контролю процесу ферментації вина, яку можна було б використовувати на вітчизняних малих виноробних підприємствах.

Створення системи контролю за процесом ферментації винних продуктів є актуальним і сучасним завданням. Комп’ютерна система повинна забезпечувати контроль за етапами ферментації продукції при мінімальних втратах.

Дослідивши фактори впливу на процес утворення вина, від переробки ягід до ферментації, було виявлено що головним завданням підвищення якості вина є удосконалення технології виготовлення з врахуванням постійного моніторингу.

РОЗДІЛ 2
МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ КОНТРОЛЮ ПРОЦЕСУ ФЕРМЕНТАЦІЇ ВИННИХ ПРОДУКТІВ

2.1. Математична модель процесу ферментації

У харчовій промисловості завжди виникає потреба як в зниженні виробничих витрат, а також у використанні різних методів, які дозволяють здійснювати контроль з метою покращення та підтримки якості продукції. Для цього досліджуються методи математичного моделювання та оптимізації, які застосовуються до виробничого процесу. Тому важливим при розробці комп'ютерної системи контролю процесу ФВП є обґрунтування математичних моделей технологічних процесів, які використовуються.

В літературі описано різні математичні системи, що моделюють процес виробництва вина на основі різної кінетики. Зазвичай вони враховують активність дріжджів, цукру, спирту, концентрації азоту. Вплив кисню на ферментацію описано Shea A. J. Comfort [4].

Ферментація вина є дуже складним біохімічним процесом, на який може впливати як багато чинників навколишнього середовища, так і кількість біохімічних компонентів. Наприклад, кисень відіграє вирішальну роль для активності дріжджів, тому його додають до сусла у фазі, коли дріжджі не в змозі впоратися з високою концентрацією цукру та поживних речовин. Але дріжджі повинні поглинути доданий кисень, перш ніж вступити в реакцію з суслом, оскільки це призводить до окислення вина.

Загалом, для моделювання процесу ферментації вина існує безліч моделей, заснованих на звичайних диференціальних рівняннях. Є моделі, які описують еволюцію біомаси дріжджів та інших концентрацій субстратів, таких як азот, цукор та етанол. Цукор перетворюється на етанол, а дріжджі ростуть шляхом метаболізму цукру та інших поживних речовин.

Хоча варто відмітити, що більшість моделей не відображають поведінку дріжджових клітин, яка спостерігається в реальних експериментах. Фази росту дріжджів описані в [7]. Необхідно відмітити, що в моделях не достатньо враховано вплив кисню, що є важливим для активності дріжджів, особливо у випадку використання Saccharomyces cerevisiae.

При розробці математичного забезпечення запропонованої в роботі комп'ютерної системи досліджуються аспекти ефективної моделі, що представляє процес ферментації вина. Для цього розглянемо модель описану в [5], та в подальшому вдосконалимо її з метою для врахування динаміки дріжджів.

Модель описується наступними диференціальними рівняннями:

$$\begin{cases} \frac{\partial X}{\partial t} = \frac{\mu_{\max(T)N}}{K_N+N} X \\ \frac{\partial N}{\partial t} = \frac{k_1 \mu_{\max(T)N}}{K_N+N} X \\ \frac{\partial E}{\partial t} = \frac{\beta_{\max(T)S}}{K_S+S} \frac{K_E(T)}{K_E(T)+E} X \\ \frac{\partial S}{\partial t} = \frac{k_2 \beta_{\max(T)S}}{K_S+S} \frac{K_E(T)}{K_E(T)+E} X \end{cases} \quad (2.1)$$

де X_O = X(0), $E_O = 0$, $N_O = N(0)$, $S_O = S(0)$

Ця модель сформульована з кінетикою Міхаеліса-Ментена. Де *X* - концентрація дріжджів, *N* - концентрація азоту, *E* - концентрація етанолу, а *S* - концентрація цукру.

Концентрація цукру розбивається на кількість цукру, що розглядається для перетворення в етанол, і кількість цукру, необхідного як поживна речовина для активності дріжджів. Крім цього, присутність кисню, позначається як індекс *O,* додається до моделі, оскільки це має першочергове значення для активності дріжджів, особливо для деяких штамів - таких як Saccharomyces cerivisiae. Це відбувається на початку бродіння, коли дріжджі не в змозі впоратися з великою кількістю доступного цукру та поживних речовин. Однак дріжджі повинні поглинути доступний кисень, перш ніж вступити в реакцію з суслом. Інакше це призводить до окислення, що в основному означає, що вино перетворюється на оцет.

Вплив кисню на ферментацію описано J. Comfort [2].

$$\begin{cases} \frac{\partial X}{\partial t} = \mu_{\max}(T)\frac{N}{K_N+N}\frac{S}{K_{S1}+S}\frac{O}{K_O+O}X - \Phi(E)X \\ \frac{\partial N}{\partial t} = k_1\mu_{\max}(T)\frac{N}{K_N+N}\frac{S}{K_{S1}+S}\frac{O}{K_O+O}X \\ \frac{\partial E}{\partial t} = \beta_{max}(T)\frac{S}{K_{s2}+S}\frac{K_E(T)}{K_E(T)+E}X \\ \frac{\partial S}{\partial t} = -k_4\frac{\partial E}{\partial t} - k_3\mu_{\max}(T)\frac{N}{K_N+N}\frac{S}{K_{S1}+S}\frac{O}{K_O+O}X \\ \frac{\partial O}{\partial t} = -k_4\mu_{\max}(T)\frac{N}{K_N+N}\frac{S}{K_{S1}+S}\frac{O}{K_O+O}X \end{cases} \quad (2.2)$$

де X(0) = X_O, E(0) = 0, N(0) = N_O, S(0) = S_O, O(0) = O_O

Дріжджі ростуть шляхом метаболізму цукру та поживних речовин, тобто кисню та азоту. На додаток до параметрів, які вже введені для початкової моделі, необхідні наступні параметри. Замість K_s у цій моделі необхідні дві константи насичення, пов'язані з цукром, а саме $K_{s1} > 0$ і $K_{s2} > 0$. Таким чином, K_{s1} являє собою константу насичення, пов'язану з частиною цукру, що використовується як поживна речовина для дріжджів, а K_{s2} є константою насичення, пов'язаною з частиною цукру, необхідною для перетворення в спирт. Крім того, $k_3 > 0$ означає коефіцієнт виходу, пов'язаний з частиною цукру, яка використовується як живильна речовина для дріжджів. $K_O > 0$ – константа напівнасичення Міхаеліса–Ментена, пов'язана з киснем, а $K_4 > 0$ – вихід коефіцієнт, пов'язаний з киснем.

Для Ф(Е) розглядається такий доданок:

$$\Phi(E) = \left(0{,}5 + \frac{1}{\pi}\arctan\left(k_{d1}(E - tol)\right)\right) k_{d2}(E - tol)^2 \quad (2.3)$$

де *tol* > 0 – допуск концентрації етанолу, наприклад, tol = 79 г/л, який був визначений за набором даних, отриманих в Університеті Гайзенхайма. Крім того, $kd_1 > 0$ і $kd_2 > 0$ є параметрами, пов'язаними із старінням клітин дріжджів через перевищення концентрацію етанолу *tol*.

У разі багаторазового вирішення задачі оптимального керування та обчислення майбутнього керуючого входу система диференціальних рівнянь, що представляють процес, включає додаткове диференціальне рівняння для врахування температури. Це додаткове рівняння можна описати за формулою:

$$\frac{dT}{dt} = \alpha_1 \frac{dE}{dt} - \alpha_2 \frac{dO}{dt} - \alpha_3(T - u_c)\omega_1(t) - \alpha_4(T - T_{ext}) \qquad (2.4)$$

Цей опис базується на певних припущеннях, наприклад, припускаємо, що із накопиченням етанолу температура всередині ферментаційного резервуара підвищується, особливо на початку, коли ще присутній кисень [4].

У рівнянні (2.4) при $\alpha_1 > 0$ вказує, скільки тепла виділяється при перетворенні цукру в спирт. Крім того, $\alpha_2 > 0$ позначає міру того, як зникнення кисню зменшує теплоутворення. $\alpha_3 > 0$ і $\alpha_4 > 0$ можна інтерпретувати як коефіцієнти теплопередачі, перший описує передачу тепла від охолоджуючого елемента до внутрішньої частини резервуара, а другий - описує передачу тепла від зовнішньої сторони бака до внутрішньої частини резервуара. Значення α_3 є незначним, якщо резервуар для вина пропорційний великому охолоджувальному елементу і навпаки.

Таким чином, *T* позначає поточну температуру в ферментаційному резервуарі, а u_c – постійну температуру охолоджуючої рідини, що протікає через охолоджуючий елемент. Керуючий вхід ω_1 визначає профіль охолодження в часі. Існування та єдність розв'язку початкової задачі, заданої системою з початковими умовами, виходить з теореми Пікара-Лінделефа з неперервністю правої функції та локальною неперервністю правої функції за Ліпшицем.

У біохімічних реакціях зазвичай як каталізатори беруть участь ферменти. Ферменти відіграють вирішальну роль у контексті біологічних процесів, включаючи процеси ферментації.

У разі ферментації вина необхідно враховувати каталізованими ферментами реакції, так що швидкість метаболізму залучених біохімічних систем сильно залежить від температури.

Тому дуже важливо моделювати температурну залежність. Існують різні способи моделювання температурної залежності. Дуже популярним і ефективним способом є використання рівняння Ареніуса [7].

$$k = A_f \exp\left(\frac{-E_a}{RT}\right) \quad (2.5)$$

де A_f - коефіцієнт Ареніуса, R — газова стала [Дж/(моль×К)], Т — абсолютна температура [К], а E_a — енергія активації [Дж/моль].

У роботі [7] було обрано спрощений спосіб моделювання температурної залежності максимальних питомих швидкостей росту та інгібування етанолу для процесу бродіння вина, а саме тут передбачається, що ці швидкості реакції лінійно залежать від температури. Крім того, існують різні способи моделювання кінетики росту [8]. Одним із способів є використання кінетики Міхаеліса-Ментена. Міхаеліс і Ментен [9].

Це пояснює залежність активності ферменту від концентрації субстрату. При цьому питома швидкість росту нелінійно залежить від обмеженої концентрації субстрату [10]. Для ферментативної кінетики часто використовують кінетику Міхаеліса-Ментена, де *KSu* називають постійною Міхаеліса-Ментена.

Процеси ферментації виникають у багатьох контекстах, таких як виробництво харчових продуктів, промислових хімікатів та фармацевтичних хімікатів.

Під час ферментації дріжджі ростуть шляхом метаболізму цукру в присутності поживних речовин, таких як азот, що засвоюється. Спожитий цукор перетворюється на етанол. Однак етанол пригнічує ріст дріжджів, а його велика кількість викликає загибель клітин.

Загалом, для моделювання процесу бродіння вина існує безліч моделей, заснованих на звичайних диференціальних рівняннях. Модель представляє еволюцію біомаси дріжджів та інших концентрацій субстратів, таких як азот, цукор та етанол. Цукор перетворюється на етанол, а дріжджі ростуть шляхом метаболізму цукру та інших поживних речовин. З цієї літературної моделі було виведено модель, засновану

на звичайних диференціальних рівняннях (ОДР), яка також враховує старіння дріжджів, пов’язану з киснем та етанолом.

Загалом, задачі оптимального керування служать для пошуку оптимального закону керування, що мінімізує або максимізує функцію витрат, що підлягає системі, яка представляє процес.

Огляд продуктів ферментації описано в [11]. Модель також включає інші субстрати, такі як азот, кисень, цукор та етанол.

2.2. Математичне забезпечення комп’ютерної системи

Існує кілька різних способів розв’язання задач оптимального керування, описані в [10]. Доступні методи можна розділити на три основні підходи: непрямі методи, прямі методи та підхід Гамільтона-Якобі-Белмана. Прикладом непрямого методу є принцип максимуму Понтрягіна [11]. Більше інформації про процедуру використання диференціальних рівнянь Гамільтона-Якобі-Беллмана [12].

Для представлення процесу ферментації вина доцільно використовувати перемикаючу структуру, оскільки є охолоджуючий елемент, який можна ввімкнути або вимкнути. Це дає змішані цілі компоненти в задачі оптимального керування.

Загалом, задачі оптимального керування служать для пошуку оптимального закону керування, що мінімізує або максимізує функцію витрат, що підлягає системі, яка представляє процес. Таким чином цю змішану цілочисельну задачу оптимального керування можна переформулювати за допомогою підходу релаксації зовнішньої конвексифікації.

Існує кілька різних способів розв’язання задач оптимального керування для процесу ФВП, описані в [10]. Доступні методи можна розділити на три основних: непрямі методи, прямі методи та підхід Гамільтона-Якобі-Белмана. Прикладом непрямого методу є принцип максимуму Понтрягіна [12]. Використання диференціальних рівнянь Гамільтона-Якобі-Беллмана описано в джерелі[13].

Спочатку потрібно сформувати модель, яка представляє пакетний процес. З цього процесу буде отримано вихід системи, а саме дані, виміряні давачами, і з цих

даних буде знову оцінено процес і обчислено новий контрольний вхід, який потім знову повторюється. Все це виконується в циклі. Далі необхідно здійснити поєднання нелінійної моделі прогнозного керування та оцінки параметрів і стану.

Загалом, модель поєднує нелінійну модель передбачуваного керування та управління з прогнозуванням. У випадку нелінійної моделі - цільовий функціонал зазвичай має тип відстеження, а у випадку лінійної моделі - мета полягає в максимізації прибутковості системи.

Модель предиктивного контролю являє собою поєднання оцінки поточного стану та параметрів з контролем межі і це зазвичай для коротких майбутніх періодів часу. Тому потрібно сформулювати та розв'язати задачу оцінки параметрів і стану. При вирішенні цієї задачі, параметри та стани ідентифікуються за допомогою доступних попередніх вимірювань. Це можна реалізувати за допомогою різних стратегій, таких як підхід до повної оцінки інформації, коли межа оцінки зростає з часом. Іншими стратегіями оцінки є, наприклад, оцінка рухомої межі, де точка оцінки постійно зміщується вперед, або розширений фільтр Калмана.

Таким чином, далі буде використано оцінку станів та параметрів для поточного моменту часу для ініціалізації моделі.

$$\sum_{i=0}^{N_c} || \, n_i - o(t_i, y(t_i), d, p) \, \Big\|_{S_i}^2 \tag{2.6}$$

З урахуванням того що $\mathrm{y(t)} = \mathrm{f(t, y, d, p)}, \mathrm{t} \in [t_0, t_c]$, $c(y(t_0), \dots, y(t_c), p) = 0$ або ≥ 0 з цільовим функціоналом методу найменших квадратів, зваженим за допомогою додатних напіввизначених вагових матриць $S_{0 \dots} S_c$. Зазвичай вони задаються оберненими коваріаційними матрицями дисперсії, пов'язаними з похибками вимірювань, але в нашому випадку вони виражаються середнім значенням вимірювань. η_i позначає виміряні дані.

У випадку нашого конкретного застосування – контролю процесу ферментації вина здійснюється вимірювання концентрації цукру та етанолу в моменти проби t_0, t_c як дані. Крім того, $o(t_i, \mathrm{y}(t_i), d, p)$, представляє відповідний вихід моделі.

Система диференціальних рівнянь $y(t) = f(t, y, d, p)$ з диференціальними станами $y = [X, N, E, S, O]$, змодельованими за допомогою дискретного стрибка d та параметра p для функції c.

Після створення основи для ініціалізації моделі, тобто постановки задачі оцінки параметрів і стану з використанням повного інформаційного підходу, можна сформулювати задачу математичного моделювання.

На рис. 2.2 показана дискретна схема роботи і прогнозування. Таким чином, стани та параметри ініціалізуються в поточний момент часу за допомогою оцінки параметрів і стану на основі попередніх вимірювань. На наступному кроці здійснюється обчислення наступного контрольного входу із відкритим циклом і проводиться прогноз для наступних станів [10].

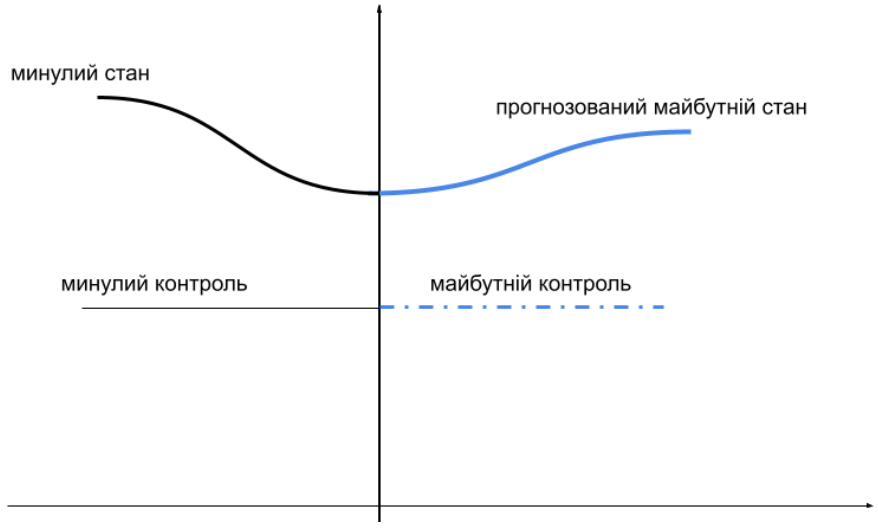

Рис. 2.2 Дискретна схема роботи і прогнозування

Роль моделі на основі поточних станів $y(t_c)$ додавання речовин d та оцінок параметрів p полягає в прогнозуванні динамічної поведінки системи та визначенні майбутніх керуючих даних. Це призводить до розв'язання задачі керування з відкритим циклом за фіксований часовою межею T. З наявністю нових вимірювань буде вирішено задачі оцінки параметрів та стану знову і будуть використовуватись нові оцінки, отримані в результаті рішення повної інформаційної оцінки. Потім

зміщується межа і знову створюється новий прогноз динамічної поведінки системи зого моменту часу.

Для опису процесу контролю за ферментацією винних продуктів доцільно в моделях враховувати показники температури та CO_2 з метою прогнозування проведення технологічних операцій (наприклад, мінімізація старіння дріжджів). Модель, яка пропонується, описується:

$$\int_{t_c}^{t_c+\tau} F(t, y(t), y(t_c))dt \tag{2.7}$$

З урахуванням:

$$y(t) = f\left(t, y(t), u(t), \hat{d}, \hat{p}\right), \forall t \in [t_c, t_c + \tau\,]$$
$$y(t_c) = \hat{y}(t_c)$$
$$c\left(t, y(t), u(t), \hat{d}, \hat{p}\right) \geq 0, \forall t \in [t_c, t_c + \tau\,],$$

де t_c означає поточний момент часу, а точне формулювання $f\left(t, y(t), u(t)\right)$ залежить від конкретного розглянутого випадку. Для моделі формулювання $f\left(t, y(t), u(t)\right)$ залежить від цільового стану та контрольних даних, що надходять від попереднього обчислення.

На рис. 2.3 зображено графіки вироблення етанолу для двадцяти днів із постійним початковим розподілом та неявним правилом трапеції: дискретизація маси з 30 проти 50 клітин

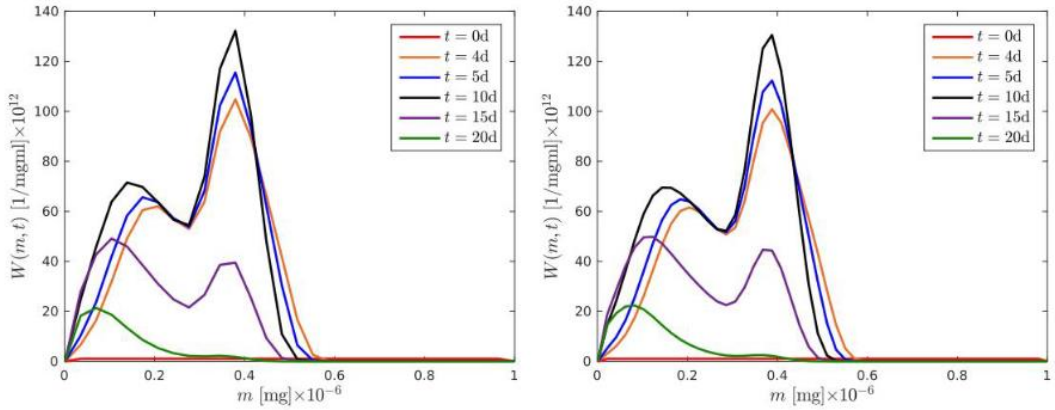

Рис. 2.3 Графіки вироблення етанолу для двадцяти днів із постійним початковим розподілом

2.3. Висновки до розділу 2

В даному розділі було досліджено класичну математичну модель процесу ферментації, виділено основні параметри, які варто враховувати при розробці моделі.

Запропоновано математичну модель з врахуванням параметрів температури та CO2. Таким чином, на основі запропонованої вдосконаленої моделі розроблено математичне забезпечення комп’ютерної системи контролю процесу ФПВ.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ СИСТЕМИ КОНТРОЛЮ ПРОЦЕСУ ФВП

3.1. Апаратне забезпечення

Комп'ютерна система контролю (КСК) процесу ФВП розроблена на базі мікроконтролера nRF. Основний елементом її є nRF52840 – як блок опрацювання даних, як модуль передачі даних виступає E19-433M30S SX1278 LoRa.

Схема з'єднань КСК процесу ферментації зображена на рис. 3.1.

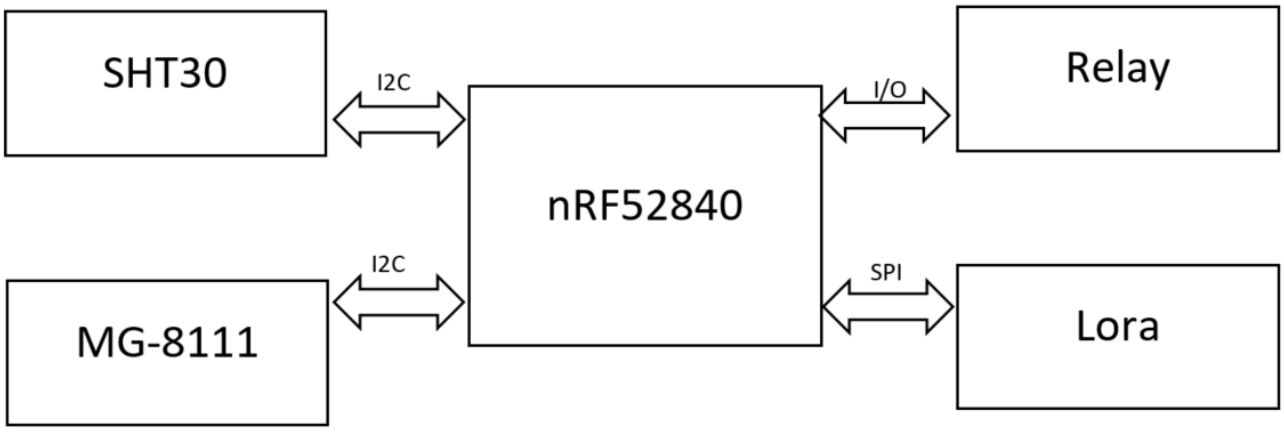

Рис. 3.1 Схема з'єднань комп'ютерної системи

nRF52840 — це передове, дуже гнучке однокристальне рішення для LKZ, енергоефективне та з достатніми обчислюваними потужностями для програми ULP з використанням підключених пристроїв, з підтримкою основних функцій Bluetooth LE, що реалізовано з збільшеними можливостями продуктивності, які включають довший режими дальності та високої пропускної здатності. Внутрішня безпека промислового рівня необхідні для сучасних програм. nRF52840 забезпечує найкращий у своєму класі захист для використання в пристроях для серії CortexTM-M із вбудованим ARM криптографічним прискорювачем CryptoCell.

nRF52840 використовує ту саму апаратну та програмну архітектуру, що й існуючі SoC серії nRF52. Його ядром є процесор Arm Cortex-M4 дозволяючи швидше та ефективніше обчислювати складні функції для DSP і ті, що потребують обчислень з плаваючою комою. Повна швидкість (12Мбіт/с) Контролер USB 2.0 включено на мікросхемі. Широкий асортимент периферійні пристрої доступні з низкою високопродуктивних цифрових інтерфейсів, такі як високошвидкісний SPI (32 МГц) і QUAD SPI (32 МГц), щоб дозволити пряме підключення до дисплеїв і зовнішніх джерел пам'яті. nRF52840 може працювати від напруги живлення від +5,5 В до 1,7 В, що дозволяє пряме живлення від акумуляторних батарей і джерел USB [22].

Сертифікована підтримка протоколу Thread і підтримка 802.15.4. Це означає, що він підходить для розробки продуктів, пов’язаних з домашньою екосистемою. Радіо підтримує рівні 802.15.4 PHY і MAC і робить його придатним для додаткових стеків за допомогою 802.15.4, наприклад Zigbee [22].

nRF52840 побудований навколо 32-розрядного процесора ARM® Cortex-M4 з блоком з плаваючою комою, що працює на частоті 64 МГц. Він має тег NFC-A для використання в спрощених рішеннях для підключення та оплати. Криптографічний блок ARM TrustZone CryptoCell включений на кристалі і пропонує широкий спектр криптографічних опцій, які виконують дуже ефективно незалежно від процесора. Він має численні цифрові периферійні пристрої та інтерфейси, такі як PDM та I^2S для цифрових мікрофонів і аудіо, а також повношвидкісний USB-пристрій для передачі даних і блоку живлення для підзарядки акумулятора.

Виключно низьке енергоспоживання досягається за допомогою складної адаптивної системи управління живленням на кристалі.

Процесор Cortex-M4 із модулем обчислення з плаваючою комою (FPU) має 32-розрядний набір інструкцій (Thumb-2 технологія), яка реалізує надмножину 16- та 32-розрядних інструкцій для максимізації щільності коду та продуктивність.

Цей процесор реалізує такі функції, які забезпечують енергоефективну арифметику та високу продуктивність обробки сигналів:

- інструкції з обробки цифрового сигналу (DSP);
- одноциклові інструкції множення та накопичення (MAC);

- апаратне переривання;
- 8- та 16-розрядні команди SIMD (single instruction multiple data).

Рівень апаратної абстракції стандарту програмного інтерфейсу мікроконтролера (CMSIS) для ARM Cortex.

Виконання в режимі реального часу є високодетермінованим у режимі потоку, переході з режимів сну та під час обробки події з конфігурованими рівнями пріоритету через вкладений векторизований контролер переривань (NVIC). Виконання коду з флеш-пам’яті матиме заборону для стану очікування для серії nRF52. Кеш інструкцій можна ввімкнути, щоб мінімізувати стани очікування переривання під час отримання інструкцій.

Для контролю за процесом ферментації використано давачі SHT30 та MG-811 (див. рис. 3.1.)

Давачі температури та вологості SHT3x відкривають новий рівень у технології вимірювань. Лінійка SHT3x складається з трьох моделей: базової моделі SHT30, стандартної SHT31 та високоефективної SHT35 [21].

У функціонал давачів входять: схема опрацювання та підсилення сигналу, схема лінеаризації сигналу, блок пам'яті калібрування, АЦП та схема скидання живлення. Цифровий інтерфейс I2C передачі даних має швидкість до 1МГц та два адресні осередки з можливістю вибору. Традиційно, компанія Sensirion пропонує давачі повністю калібровані, з широким діапазоном напруги живлення, від 2.4В до 5.5В; діапазоном вимірювання температури: -40…+125°C; діапазоном вимірювання відносної вологості: 0–100%; точність вимірювання: 3%; час відгуку: 8с; потужність споживання: 4,8мкВт.

Газовий давач MG-811, який використано в системі, дозволяє легко контролювати рівні концентрації вуглекислого газу за допомогою міні-модуля I2C. MG-811 підключений до вбудованого підсилювача стану сигналу, а підсилювач сигналу - підключений до 12-розрядного аналого-цифрового перетворювача ADC121C, який здатний розширити до 9 давачів газу на порт I2C за допомогою лише двох адресних перемичок (з повним використанням плаваючої адресної системи). Цей давач CO_2 MG-811 менш чутливий до оксиду вуглецю (CO) та алкоголю [22].

MG-811 здатний визначати концентрацію вуглекислого газу в повітрі від 350 до 10 000ppm. Ідеальними умовами для давача газу MG-811 є температура 25°C ±2°C при вологості 65% ±5%. Внутрішній попередній підігрівач всередині датчика допомагає досягнути ідеальних умов вимірювання, але для досягнення оптимальної точності для попереднього підігріву рекомендовано час понад 48 годин.

Усі міні-модулі I2C розроблені для роботи від 5В постійного струму. Використовуючи зручний 4-контактний штекер, пристрої можна послідовно під’єднати до шини I2C, усуваючи необхідність пайки [22].

Для передачі даних з модуля було обрано технологію LoRa. LoRa - це бездротова технологія, розроблена для малопотужних, широкоформатних мереж (LPWAN), необхідних для додатків Інтернету речей (IoT) та машин (M2M) [23].

Технологія пропонує високе проникнення, низьку пропускну здатність, низьку енергію, широку площу, безпечні дані та набирає значної популярності в мережах IoT, які розгортаються операторами бездротових мереж та урядом. Він має власний сегмент, що працює далеко від стільникової мережі та Wi-Fi.

Ключові особливості технології:

- велика дальність: Одна базова станція LoRa забезпечує можливість глибокого проникнення для щільного міського середовища та внутрішнього покриття, а також забезпечує можливість підключення до датчиків на відстані понад 24140 – 50000м у сільській місцевості;
- низька вартість: LoRa зменшує як попередні інвестиції в інфраструктуру, так і експлуатаційні витрати, а також витрати на датчики кінцевих вузлів;
- стандартизований: LoRaWAN забезпечує сумісність між додатками, постачальниками рішень IoT та операторами зв'язку для прискорення прийняття та розгортання;
- низька потужність: Протокол LoRaWAN був розроблений спеціально для низької потужності та забезпечує безпрецедентний багаторічний термін служби акумулятора.

LoRa використовує технологію CSS (стрекотіння розширеного спектру), розроблену компанією Semtech. Він зосереджений на безпечному двонаправленому

зв'язку, асинхронному протоколі, який є оптимальним для терміну служби та вартості акумулятора.

CSS використовує всю виділену пропускну здатність для трансляції сигналу, що робить його надійним для канального шуму, відмінним при обробці перешкод і перекриття мереж.

Бездротова система LoRa використовує неліцензійні частоти нижче 1ГГц, які доступні у всьому світі. Він поставляється без додаткових витрат з найбільш широко використовуваними частотами:

- 868МГц для Європи;
- 915МГц для Північної Америки;
- 433МГц діапазон для Азії.

Використання більш низьких частот, ніж у діапазонах ISM 2,4 або 5,8ГГц, дозволяє досягти набагато кращого покриття, особливо коли вузли знаходяться в будівлях, що забезпечує відмінне проникнення високих будівель і стін.

3.2. Програмне забезпечення комп’ютерної системи

Для програмного забезпечення комп’ютерної системи контролю процесу ФВП було використано Zephyr OS, дані передаються на платформу ThinkSpeak.

Zephyr — це ОС реального часу для підключених пристроїв із обмеженими ресурсами та ВП, що підтримують декілька архітектур і включає ядро. Що містить для розробки ППЗ всі компоненти, бібліотеки, драйвери пристроїв, файлові системи, стеки протоколів та оновлення мікропрограм [18]. Zephyr використовує систему конфігурацій Kconfig і devicetree, успадковані від ядра Linux, але реалізовані на мові Python для використання в ОС, що не належать до Unix [18]. Система збірки RTOS заснована на CMake, це дозволяє збирати програми Zephyr на Linux, macOS та Microsoft Windows [18].

ThingSpeak – це програмне забезпечення з відкритим кодом, написане на Ruby, яке дозволяє користувачам спілкуватися з пристроями з підтримкою Інтернету. Він полегшує доступ до даних, пошук та реєстрацію даних, надаючи API як для пристроїв,

так і для веб -сайтів соціальних мереж. ThingSpeak — це служба аналітичної платформи IoT, яка дозволяє об’єднувати, візуалізувати та аналізувати потоки даних у реальному часі в хмарі. Платформа дає можливість надсилати дані до ThingSpeak зі своїх пристроїв, створювати миттєві візуалізації даних в реальному часі. За допомогою аналітики MATLAB всередині ThingSpeak можна писати та виконувати код для виконання попереднього опрацювання, візуалізації та аналізу. ThingSpeak дає змогу створювати прототипи та системи Інтернету речей без налаштування серверів чи розробки веб-програм [19].

В даній системі верхнім мережевим рівнем для передачі даних по LoRa є Mesh мережа. Мережа mesh — це мережа, у якій пристрої — або вузли — з’єднані разом, відгалужуючись від інших пристроїв або вузлів. Ці мережі налаштовані на ефективну маршрутизацію даних між пристроями та клієнтами. Вони допомагають організаціям забезпечити послідовне з’єднання у всьому фізичному просторі. Топології сітчастої мережі створюють кілька маршрутів для переміщення інформації між підключеними вузлами. Такий підхід підвищує стійкість мережі у разі збою вузла або з’єднання. Більші сітчасті мережі можуть включати кілька маршрутизаторів, комутаторів та інших пристроїв, які працюють як вузли. Мережа mesh може включати сотні бездротових вузлів mesh, що дозволяє їй охоплювати велику територію [24].

У топології повної сітчастої мережі кожен вузол підключений безпосередньо до всіх інших вузлів. У частковій сітчастій топології лише деякі вузли з’єднуються один з одним. У деяких випадках вузол повинен пройти через інший вузол, щоб досягти третього вузла.

З’єднання в повній або частковій мережі можуть бути дротовими або бездротовими сітчастими мережами. Рішення використовувати повну або часткову сітку залежить від таких факторів, як загальна структура трафіку мережі та ступінь ризику збою вузлів або з’єднань.

Здається, що майже всі мережі є повністю сітчастими мережами, оскільки кожен у мережі може з’єднуватися з усіма іншими. Цей повний зв’язок є властивістю мережевих протоколів, а не топології; будь-яка мережа може виглядати повністю сітчастою на логічному рівні, якщо дані можуть бути маршрутизовані між кожним з

її користувачів. У сітчастих мережах різниця між логічною та фізичною топологіями є найбільш важливою.

Вузли в мережі запрограмовані за допомогою програмного забезпечення, яке повідомляє вузлу, як обробляти інформацію та взаємодіяти з мережею.

Mesh-мережі використовують методи маршрутизації або лавинної передачі для надсилання повідомлень. Під час маршрутизації повідомлення переходить від вузла до вузла, щоб дістатися до місця призначення. Мережа mesh повинна мати безперервні з’єднання та переконфігурувати себе, якщо шлях порушено, використовуючи алгоритми самовідновлення. Між джерелом і пунктом призначення часто існує більше ніж один шлях.

Для практичного застосування було обрано мережу RadioHead. RadioHead складається з 2 основних комплектів класів: шлюзи та менеджери.

Шлюзи забезпечують низькорівневий доступ до ряду різних пакетних радіоприймачів та інших пакетних транспортів повідомлень. Менеджери забезпечують засоби відправки та отримання повідомлень високого рівня для різних вимог.

Кожна програма RadioHead має екземпляр шлюза для надання доступу до радіо або маршрутизації передачі даних, і, як правило, менеджера, який використовує цей драйвер для надсилання та отримання повідомлень для програми.

В додатку Б наведено лістинги програми для комп’ютерної системи.

Функція надсилання повідомлення до модема з конкретним ID зображено на рис. 3.2.

```
uint8_t RHMesh::sendtoWait(uint8_t* buf, uint8_t len, uint8_t address, uint8_t flags)
{
    if (len > RH_MESH_MAX_MESSAGE_LEN)
        return RH_ROUTER_ERROR_INVALID_LENGTH;

    if (address != RH_BROADCAST_ADDRESS)
    {
        RoutingTableEntry* route = getRouteTo(address);
        if (!route && !doArp(address))
            return RH_ROUTER_ERROR_NO_ROUTE;
    }

    // Now have a route. Contruct an application layer message and send it via that route
    MeshApplicationMessage* a = (MeshApplicationMessage*)&_tmpMessage;
    a->header.msgType = RH_MESH_MESSAGE_TYPE_APPLICATION;
    memcpy(a->data, buf, len);
    return RHRouter::sendtoWait(_tmpMessage, sizeof(RHMesh::MeshMessageHeader) + len, address, flags);
}
```

Рис. 3.2 Лістинг коду для надсилання повідомлення в меш мережі

Частина лістингу, яка реалізовує маршрутизацію в мережі для того, щоб доставити повідомлення до свого отримувача, зображена на рис. 3.3.

```
uint8_t RHMesh::route(RoutedMessage* message, uint8_t messageLen)
{
    uint8_t from = headerFrom(); // Might get clobbered during call to superclass route()
    uint8_t ret = RHRouter::route(message, messageLen);
    if (   ret == RH_ROUTER_ERROR_NO_ROUTE
        || ret == RH_ROUTER_ERROR_UNABLE_TO_DELIVER)
    {
        // Cant deliver to the next hop. Delete the route
        deleteRouteTo(message->header.dest);
        if (message->header.source != _thisAddress)
        {
            // This is being proxied, so tell the originator about it
            MeshRouteFailureMessage* p = (MeshRouteFailureMessage*)&_tmpMessage;
            p->header.msgType = RH_MESH_MESSAGE_TYPE_ROUTE_FAILURE;
            p->dest = message->header.dest; // Who you were trying to deliver to
            // Make sure there is a route back towards whoever sent the original message
            addRouteTo(message->header.source, from);
            ret = RHRouter::sendtoWait((uint8_t*)p, sizeof(RHMesh::MeshMessageHeader) + 1, message->header.source);
        }
    }
    return ret;
}
```

Рис. 3.3 Лістинг коду для маршрутизації повідомлення в мережі

Частина лістингу, яка реалізовує отримання повідомлення в мережі з використанням поля ACK зображена на рис. 3.4.

```
bool RHMesh::recvfromAckTimeout(uint8_t* buf, uint8_t* len, uint16_t timeout, uint8_t* from, uint8_t* to, uint8_t* id, uint8_t* flags, uint8_t* hops)
{
    unsigned long starttime = millis();
    int32_t timeLeft;
    while ((timeLeft = timeout - (millis() - starttime)) > 0)
    {
        if (waitAvailableTimeout(timeLeft))
        {
            if (recvfromAck(buf, len, from, to, id, flags, hops))
                return true;
            YIELD;
        }
    }
    return false;
}
```

Рис. 3.4 Лістинг коду для отримання повідомлення в мережі

Частина лістингу коду з конфігурацією LoRa в ОС Zephyr зображено на рис. 3.5, де вказується частота, пропускна здатність, пропускна здатність швидкість передачі даних, довжина заголовка, швидкість кодування, тип мережі та інше.

```
const struct device *const lora_dev = DEVICE_DT_GET(DEFAULT_RADIO_NODE);
struct lora_modem_config config;
int ret;

if (!device_is_ready(lora_dev)) {
        LOG_ERR("%s Device not ready", lora_dev->name);
        return;
}

config.frequency = 865100000;
config.bandwidth = BW_125_KHZ;
config.datarate = SF_10;
config.preamble_len = 8;
config.coding_rate = CR_4_5;
config.iq_inverted = false;
config.public_network = false;
config.tx_power = 4;
config.tx = true;
```

Рис. 3.5 Лістинг коду з конфігурацією LoRa трансівера

3.3. Тестування комп’ютерної системи

Розроблена КСК процесу ферментації зображена на рис. 3.6. Давачі комп’ютерної системи, які необхідні для контролю за процесом ферментації, мають буди поміщені в пробку для бочки, незалежно від її типу, та під’єднані до контролера, який, в свою чергу, за допомогоюLoRa буде передавати дані до пристрою який є central. Контролер кількісно визначає швидкість виділення CO_2 і, на основі цього, розраховує норму споживання цукру. Потім здійснюється регулювання температури на основі підтримки постійної норми цукру. Контролер досягає компромісу щодо мінімізації споживання енергії (R - процес відновлення цукру S(t)). Додавання інших речовини здійснюється відповідно до технологічного процесу виготовлення того чи іншого виду вина та необхідності підтримки процесу ферментації. Залежно від

використовуваного штаму дріжджів необхідно також враховувати значення температури ферментації та подальшого додавання поживних речовин. Варто відмітити, що процес ферментації триває близько 20 днів.

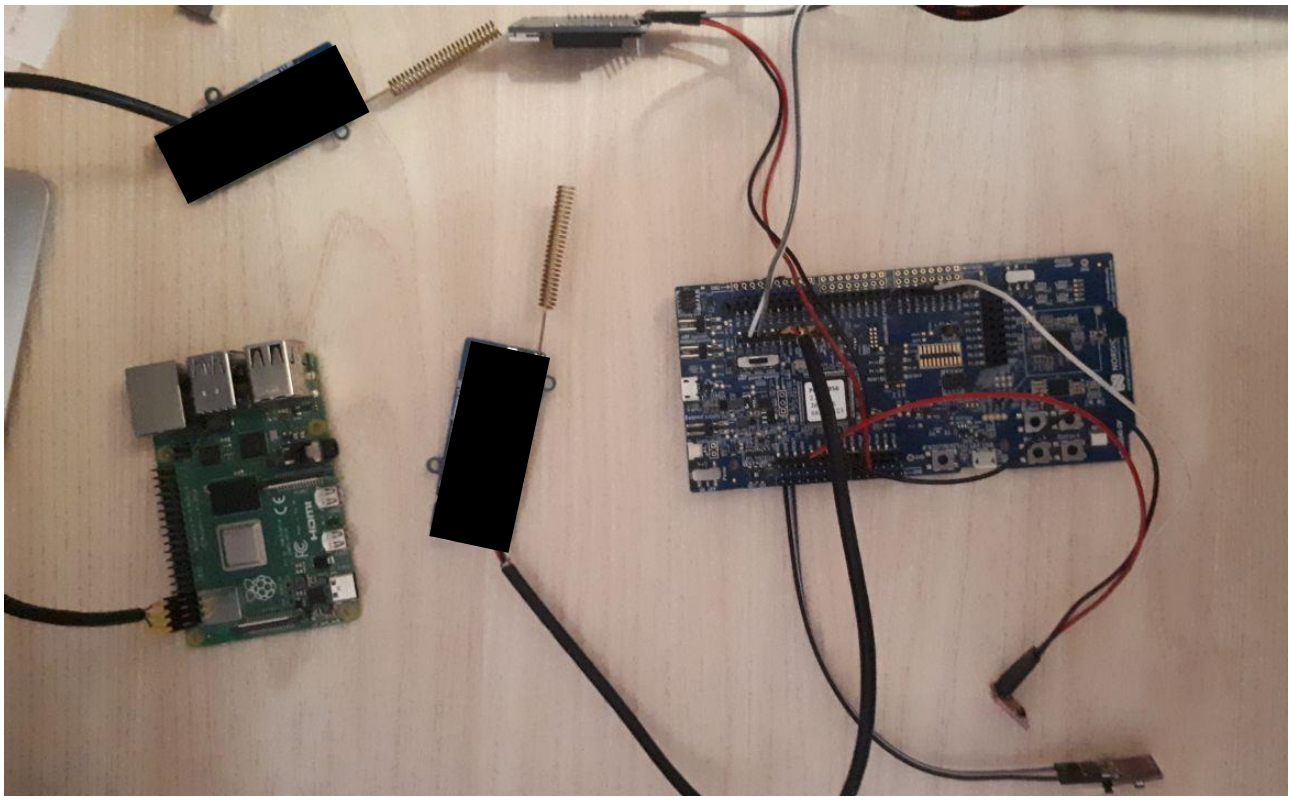

Рис. 3.6 Комп’ютерна система контролю процесу ФВП

При тестуванні системи для початку було взято значення параметрів, визначених в ідентифікації параметрів і стану, виконаних для подібного типу дріжджів, для розрахунку майбутнього контролю.

Результати оцінки цукру зображено на рис. 3.7, де показано відповідність даних вимірювання цукру для аналогічного типу дріжджів і майбутнього стану. Дані вимірювань позначені червоним кольором, майбутній стан - представлено жовтим кольором.

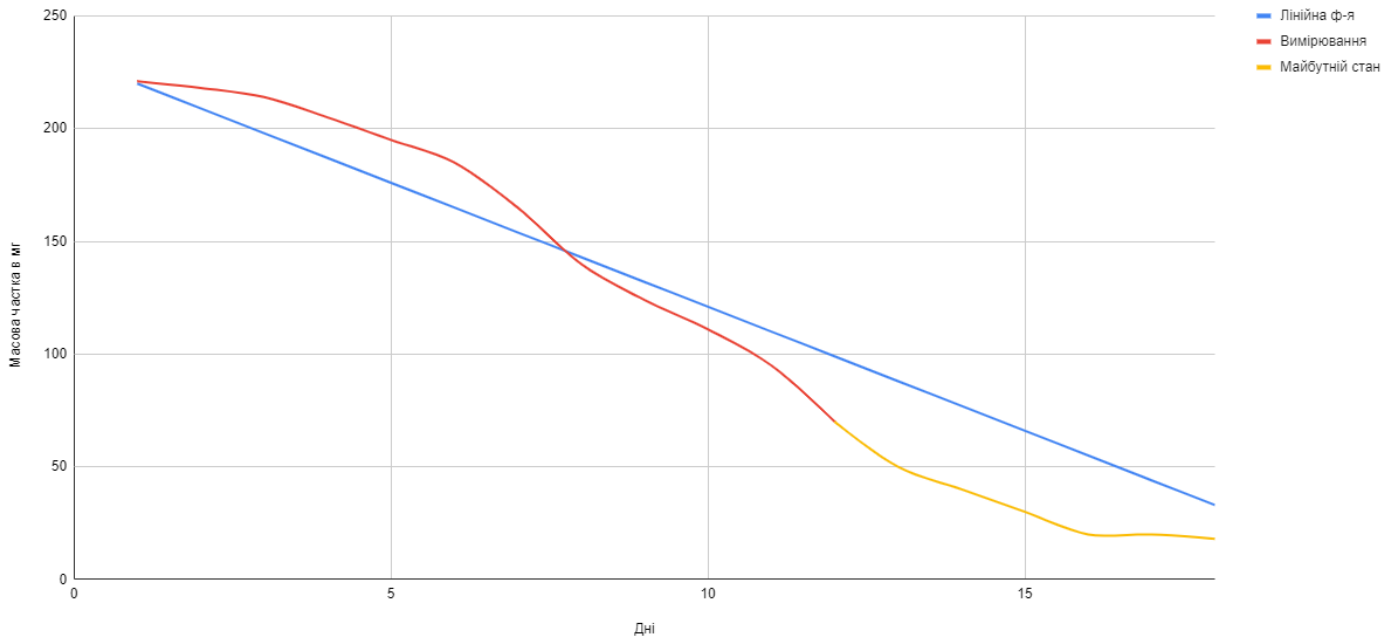

Рис. 3.7 Результати оцінки цукру

Результати оцінки етанолу зображено на рис. 3.8, де показано відповідність даних вимірювання і майбутнього стану: показники вимірювань позначені червоним кольором, майбутній стан - представлено синім кольором.

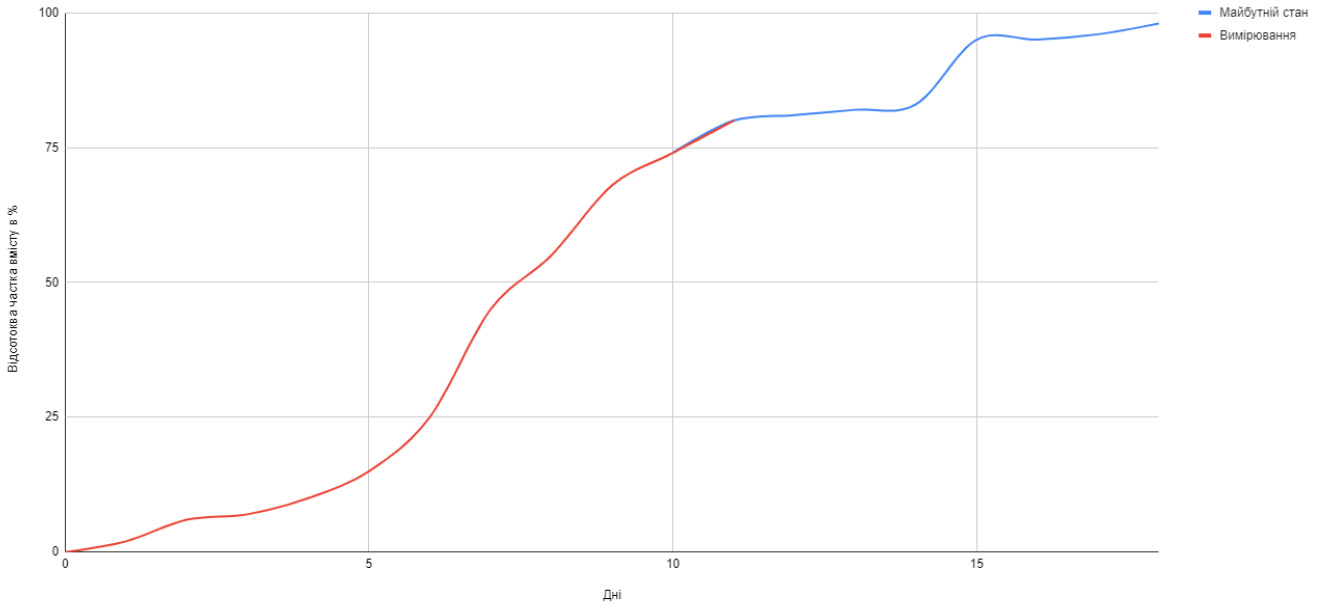

Рис. 3.8 Результат оцінки етанолу

Графік коефіцієнта керування температурою зображено на рис. 3.9.

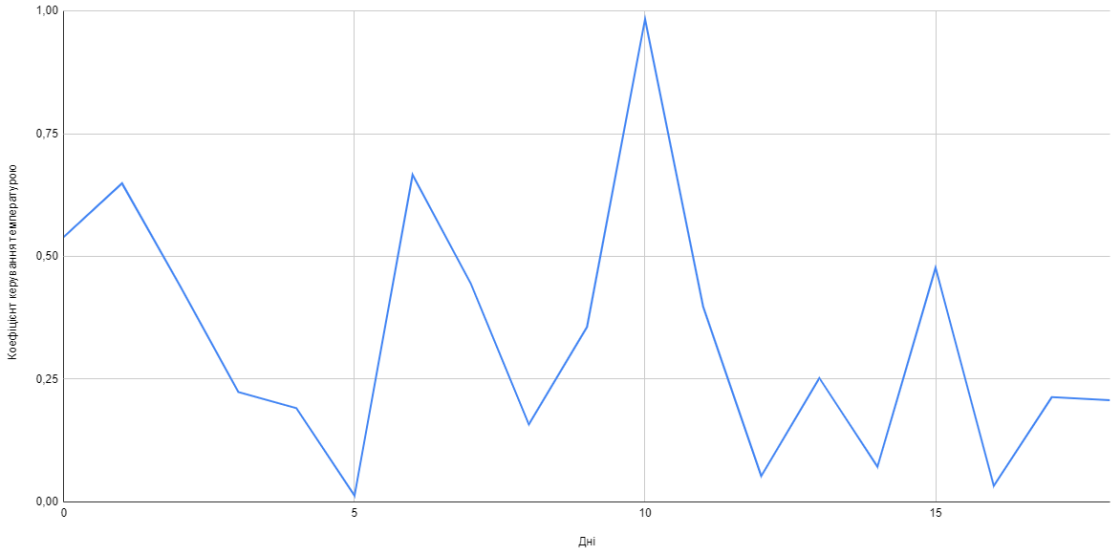

Рис. 3.9 Керування температурою

В розробленій КСК процесу ФВП отримані дані надсилаються на хмарний сервіс ThingSpeak, а результат представляється у вигляді графіків.

На рис. 3.10 показано отримані температурні показники для контролю процесу ФВП, візуалізовані за допомогою ThingSpeak.com.

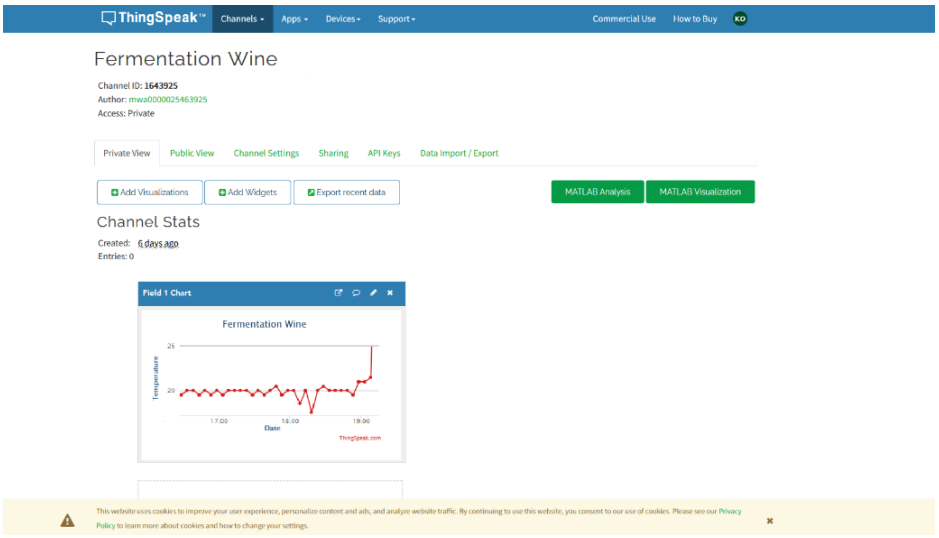

Рис. 3.10 Графік показників температури процесу ферментації винних продуктів

Процес передачі/отримання даних по LoRa в мережі RadioHead відбувається з логуванням, передача повідомлення зображена на рис. 3.11.

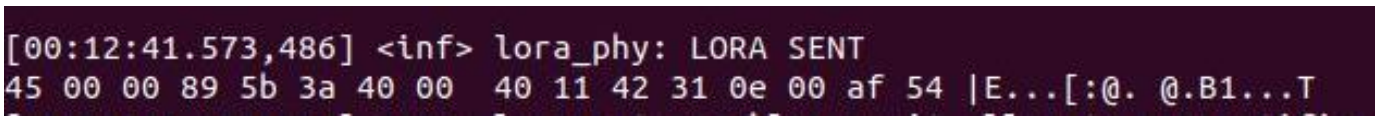

Рис. 3.11 Лог передачі даних по LoRa

В майбутньому планується додати передачу інформації про прогнозований стан ферментації, а також кількісні значення кисню та етанолу.

3.4. Висновки до розділу 3

В роботі розроблено КС для контролю процесу ферментації винних продуктів на базі мікроконтролера nRF52. Запропоновано використання хмарного сервісу ThingSpeak для передачі та візуалізації отриманих показників, необхідних для контролю за процесом ФВП.

Розроблена комп’ютерна система та її програмно-апаратна реалізація дозволяє здійснити контроль за якістю важливого у виноробстві технологічного процесу ферментації вина.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

В кваліфікаційній роботі магістра розроблено систему для керування процесом ферментації винних продуктів. Враховуючи потенційні небезпечні ситуації, які може спричинити дана комп'ютерну система, для їх уникнення, необхідно дотримуватись всіх правил охорони праці та техніки безпеки. Згідно ДСТ 12.0.003-74*, при роботі працівник може піддаватися таким небезпечним та шкідливим впливам: – небезпека ураження електричним струмом; – підвищений рівень утворення пожежі в приміщенні; – незадовільні параметри мікроклімату; – електромагнітне випромінювання монітору; – несприятлива освітленість; – підвищений рівень шуму [25]. Електротравма – травма, яка спричинена дією на організм людини електричного струму і (або) електричної дуги. Працюючи з електричними компонентами КС слід дотримуватись комплексу заходів щодо забезпечення електробезпеки. Основними заходами для захисту від ураження електричним струмом є: – забезпечення недоступності провідників, що знаходяться під напругою, від випадкового дотику; – усунення небезпеки ураження з появою напруги на корпусах, в кожухах та ін. частинах електроустаткування, що досягається застосуванням малих напруг, використанням подвійної ізоляції, захисним зануленням, захисним відключенням [25]. Доцільним є застосування занулення мережі. Занулення – це навмисне з'єднання з нульовим захисним проводом металевих струмоведучих частин, що можуть виявитися під напругою. Відповідно до ДБН В.1.1.7-2016 усі виробництва поділяють на пожежо-, і вибухонебезпечної категорії. Дане приміщення відноситься до категорії Д. Мікроклімат у приміщенні, де проводяться роботи, нормується відповідно до ДСН 3.3.6.042-99 [26]. Умови, що визначають стан повітря робочої зони, характеризуються: – температурою навколишнього повітря; – відносною вологістю; – швидкістю руху повітря. Оптимальні показники мікроклімату, які необхідно

забезпечити у приміщеннях, де експлуатуються ПК у теплу пору року повинні становити: температура – +22 - +24°С, відносна вологість – 40-60 %, швидкість руху повітря 0,1 м/с. На робочому місці розробника системи для керування зовнішнім освітленням необхідно забезпечити дотримання вимог НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями». Основними вимогами, визначеними у цьому нормативному документі є: – площу та об'єм для одного робочого місця оператора визначають згідно з вимогами ДСанПіН 3.3.2-007-98. Площа має бути не менше 6,0 кв.м, об'єм – не менше 20,0 куб.м; – заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця операторів (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), мають бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння працівника під напругу; – приміщення, де розміщені робочі місця операторів, крім приміщень, у яких розміщені робочі місця операторів великих ЕОМ загального призначення (сервер), повинні бути оснащені системою автоматичної пожежної сигналізації. 61 Заходи для захисту від випромінювань: – застосування захисних екранів; – застосування спеціальних екранів зі слабким випромінюванням; – застосування монохромних або рідкокристалічних екранів. Природне освітлення приміщення здійснюється бічним світлом через світлові пройми в зовнішніх стінках (вікна), а штучне - утворюване електричними лампами. Використовується також суміщене освітлення - при якому у світлий час доби, коли недостатньо за нормами природного освітлення, додається штучне. Для забезпечення високої освітленості на робочих поверхнях застосовують комбіноване освітлення, якщо застосування загального освітлення неекономічне. Виробниче освітлення нормується ДБН В.2.5-28-2018. Використовуване приміщення належить до приміщень І групи за зоровою роботою. При розробці системи для керування процесом ферментації винних продуктів враховані всі вимоги охорони праці, техніки безпеки та техніки безпеки [26].

4.2. Безпека в надзвичайних ситуаціях

Аварійно-рятувальні роботи (АРР) на промисловому підприємстві – це першочергові заходи на території, де сталася надзвичайна ситуація (НС), з пошуку і рятування персоналу, матеріалів та устаткування, що має суттєву матеріальну цінність, сукупність робіт по обмеженню та гасінню пожеж, аварійного відключення джерел рідкого палива, газу, електроенергії та води, та, в тому числі, надання потерпілим працівникам невідкладної допомоги медичного характеру і в разі потреби їх евакуації в спеціалізовані медичні установи поза зоною проведення АРР. Невідкладні роботи – це заходи першочергового характеру на території, де сталася надзвичайна ситуація, із всебічного забезпечення АРР, усунення окремих вогнищ (причин) підвищеної небезпеки, локалізації аварій і ушкоджень на енергетичних мережах, надання першочергової допомоги медичного характеру, забезпечення мінімальних умов для персоналу, а також роботи по санітарній очистці та знезараженню територій [27]. Надзвичайна ситуація (НС) на промисловому підприємстві - це подія на виробничому об’єкті, яка сталася внаслідок техногенної трощі, метеоявища небезпечної характеру, катастрофи, катаклізму, викликаному природною стихією, що може викликати або вже призвела до смерть людей, погіршення здоров’я персоналу або довкіллю, суттєві грошові втрати і негативний вплив на життєдіяльність працівників. Область НС – це територія, де сталася така ситуація. Аварійно-рятувальні та інші невідкладні заходи (АРІНР) на підприємстві включають в себе три етапи:

1. Вжиття екстрених заходів:

1.1. Екстрений захист працівників: – своєчасне інформування посадових осіб і уповноважених служб про загрозу настання НС і її розвитку, а також інструктаж працівників про порядок дій у екстреній ситуації;

– використання засобів захисту, впорядковане вилучення працівників із території, де трапилась НС в безпечні місця, введення встановлених режимів поведінки, проведення заходів медичного захисту;

– розшук та вилучення постраждалих та надання їм медичної допомоги.

1.2. Запобігання розвитку і зменшення небезпечних впливів НС:

– локалізація аварії;

– перекриття і глушіння (припинення дії) джерела небезпечних речовин;

– припинення (екстрене відключення) технологічних процесів.

1.3. Підготовчий етап виконання робіт:

– мобілізація служб міської ланки територіальної організації попередження і дій при виникненні НС;

– попередня оцінювання ситуації і координування комплексного обстеження в зоні НС;

– виїзд оперативних груп сил міського та окружних ланок територіальної підсистеми до місця НС;

– вирішення питання початку АРІНР.

2. Виконання АРІНР:

– переміщення в область, де сталася НС, засобів проведення АРІНР відповідно до вирішеного питання;

– безпосереднє виконання робіт аварійно-рятувального характеру і інших невідкладних робіт;

– виведення спецзасобів із зони НС, по завершенні АРІНР і переміщення їх до вихідної точки.

Рятування персоналу при виникненні НС на підприємстві являється одним із найбільш важливим при проведенні АРІНР і включає в себе сукупність мір по виведенню працівників із області, де виявлені шкідливі фактори впливу НС та їх похідні або захищення працівників від дії таких факторів, у т. ч. застосовуючи засоби індивідуального захисту та укриття. Способами, що використовуються в основному для порятунку працівників, матеріалів і обладнання є: – переміщення їх у безпечне місце, у тому числі з використанням спеціальних технічних засобів; – захист від впливу небезпечних факторів надзвичайної ситуації. Для порятунку працівників потрібно обрати найбезпечніші напрямки і методи. Вивезення потерпілих у безпечну локацію проводиться із розрахунку умов, в яких відбувається ліквідація НС і важкості

їх ураження. Засобами, що використовуються в основному для порятунку працівників, матеріалів і обладнання є: – аварійно-рятувальне обладнання і механізми: гідравлічне аварійнорятувальне обладнання, ремені, обладнані карабінами; різальний інструмент у газовому полум'ї оснащений різаком, напірним рукавом, редуктором і газовим балоном (бензорізи, газозварювальні апарати тощо), ломи, кувалди, лопати, кіркимотики важкі, сокири, пилки, підйомні засоби (включно з лебідками, домкратами тощо), мотузки, окуляри захисної дії, освітлювальні пристрої, бензо- і електропилки та ін. – рятувальне обладнання (рятувальні рукави, канати, плетені драбини та індивідуальні засоби порятунку), засоби захисту, дрони та квадрокоптери, плавзасоби; – стаціонарні та ручні драбини, що використовуються в якості пожежного інвентарю, тощо; автопідйомники та драбини на базі автомобілів та інші доступні рятувальні засоби.

4.3. Висновок до розділу 4

В даному розділі описані актуальні питання щодо охорони праці та забезпечення безпеки в надзвичайних ситуаціях. Була опрацьована інформація стосовно комплексу заходів щодо забезпечення електробезпеки під час розробки проектованої системи. Також, розглянуто питання щодо застосування основних способів та засобів в ході проведення невідкладних аварійно-рятувальних робіт на промисловому підприємстві.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було отримано такі основні результати:

- в результаті аналізу літературних джерел встановлено, що ферментація є важливим технологічним процесом у виноробстві і потребує постійного моніторингу;
- проаналізовано сучасні системи контролю винного виробництва і встановлено їх недоліки, які обмежують їх використання;
- обґрунтовано актуальність та доцільність розробки комп'ютерної системи контролю процесу ферментації вина, яку можна використовувати на вітчизняних малих виноробних підприємствах;
- проведено аналіз класичних математичних моделей для процесу ферментації та виділено основні параметрі, які варто враховувати при розробці моделі;
- на основі запропонованої моделі розроблено математичне забезпечення комп'ютерної системи контролю процесу ферментації винних продуктів;
- спроєктовано апаратно-програмне забезпечення комп'ютерної системи контролю процесу ферментації винних продуктів з використанням mesh мережі та LoRa;
- проведено тестування на оцінку майбутнього стану вироблення етанолу та переробки цукру дріжджами на основі попередніх показників з-за допомогою розробленої системи;
- розроблена комп'ютерна система та її програмно-апаратна реалізація дозволяє здійснити контроль за якістю важливого у виноробстві технологічного процесу ферментації вина. Запропоновані в роботі рішення дозволяють реалізувати постійний моніторинг за процесом ферментації, що в свою чергу позитивно впливає на якість вина.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ольховецька Х. Комп’ютеризована система контролю якості процесу ферментації винних продуктів. *IX науково-технічна конференція «Інформаційні моделі, системи та технології»*: матеріали науково-техн. конф., м. Тернопіль, 8–9 груд. 2021 р. ст. 120

2. FERMENTATION MONITORING SYSTEM. *Winegrid*. URL: https://www.winegrid.com/en/?fbclid=IwAR3aXSXvK6IqZC4E8U3wAmluyr9OefSi-LK5TupCKIX34XDFxeJKeaqs9qU (дата звернення: 10.12.2021).

3. Винный монитор 5500/5600. *Donaulab*. URL: https://donaulab.md/Винный-монитор-5500-5600-1 (дата звернення: 04.11.2021).

4. Oxygen & Fermentation. *MoreWine*. URL: https://morewinemaking.com/web_files/intranet.morebeer.com/files/oxyfer09.pdf (дата звернення: 16.12.2021).

5. R. David, D. Dochain, J.-R. Mouret, A. Vande Wouwer, and J.-M. Sablayrolles. Dynamical Modeling of Alcoholic Fermentation and Its Link with Nitrogen Consumption. In Proceedings of the 11th International Symposium on Computer Applications in Biotechnology (CAB 2010), Leuven, Belgium, ст. 496–501, 2010.

6. H. H. Dittrich and M. Großmann. Mikrobiologie des Weines. Ulmer, 4th edition, 2011. 288 с.

7. R. E. Zeebe and D. Wolf-Gladrow. CO2 in Seawater: Equilibrium, Kinetics, Isotopes. Elsevier Oceanography Series. Elsevier Science, 2001. 346 с.

8. An analysis of available mathematical models for anaerobic digestion of organic substances for production of biogas. *Proceedings of the International Gas Research Conference* : Internet. NY, 2008. С. 294–323. URL: https://www.researchgate.net/publication/283518957_An_analysis_of_available_mathematical_models_for_anaerobic_digestion_of_organic_substances_for_production_of_biogas (дата звернення: 18.11.2021).

9. L. Michaelis and M. L. Menten, “Die Kinetik der Invertinwirkung,” Biochemische Zeitschrift, Випуск 49, 1913, ст. 333-369.

10. Monod J. The Growth of Bacterial Cultures. *Annual Review of Microbiology*. 1949. Т. 3, № 1. С. 371–394.

URL: https://doi.org/10.1146/annurev.mi.03.100149.002103 (дата звернення: 14.11.2022).

11. Optimal Control of Beer Fermentation Process Using Differential Transform Method / M. Shehu та ін. *Journal of Applied Sciences and Environmental Management*. 2017. Т. 21, № 4. С. 751. URL: https://doi.org/10.4314/jasem.v21i4.16 (дата звернення: 24.12.2022).

12. Less cooling energy in wine fermentation – A case study in mathematical modeling, simulation and optimization / C. Schenk та ін. *Food and Bioproducts Processing*. 2017. Т. 103. С. 131–138. URL: https://doi.org/10.1016/j.fbp.2017.04.001 (дата звернення: 14.10.2022).

13. Pinch E. R. Optimal control and the calculus of variations. Oxford : Oxford University Press, 1995. 242 с.

14. ROSS, R. Paul; MORGAN, Sheila; HILL, Collin. Preservation and fermentation: past, present and future. *International journal of food microbiology*, 2002, 79.1-2: 3-16.

15. STANBURY, Peter F.; WHITAKER, Allan; HALL, Stephen J. *Principles of fermentation technology*. Elsevier, 2013.

16. BUSWELL, A. M.; MUELLER, H. F. Mechanism of methane fermentation. Industrial & Engineering Chemistry, 1952, 44.3: 550-552.

17. KENNEDY, M.; KROUSE, Donal. Strategies for improving fermentation medium performance: a review. *Journal of Industrial Microbiology and Biotechnology*, 1999, 23.6: 456-475.

18. An introduction to Zephyr. URL: Zephyr Project - Zephyr Project

19. STM32F767ZI. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f767zi.html.

20. Thingspeak. *Thingspeak*. URL: https://thingspeak.com/.

21. Datasheet. URL: shorturl.at/jnvD4

22. Datasheet. URL: https://sandboxelectronics.com/files/SEN-000007/MG811.pdf

23. Lora. URL: https://en.wikipedia.org/wiki/LoRa

24. Mesh. URL: https://uk.wikipedia.org/wiki/Mesh-%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D1%96

25. НПАОП 0.00-1.28-18. Про затвердження правил охорони праці під час експлуатації електронно-обчислювальних машин. URL: https://dnaop.com/html/31562/docНПАОП_0.00-1.28-18 (дата звернення: 01.12.2022).

26. Наказ Міністерства внутрішніх справ України № 340 від 26.04.2018 року “Про затвердження Статуту дій у надзвичайних ситуаціях органів управління та підрозділів Оперативно-рятувальної служби цивільного захисту та Статуту дій органів управління та підрозділів Оперативно-рятувальної служби цивільного захисту під час гасіння пожеж”.

27. Васійчук В.О., Гончарук В.Є., Качан С.І., Мохняк С.М. Основи цивільного захисту: Навчальний посібник. Львів: Видавництво Національного університету "Львівська політехніка". 2010. 417с.

28. Аветисян В.Г., Сенчихін Ю.М., Кулаков С.В., Куліш Ю.О., Александров В.Л., Адаменко М.І., Ткачук Р.С., Тригуб В.В. Рятувальні роботи під час ліквідації надзвичайних ситуацій. Частина 1: Посібник. За загальною редакцією Пшеничного В.Н. К.: Основа, 2006. 240 с. 44.

29. Желібо Є. П., Сагайдак І. С. Безпека життєдіяльності. Навчальний посібник для аудиторної та практичної роботи. К.:ЕКОМЕН. 2011. 200 с. 45. Депутат О. П., Коваленко І. В., Мужик І. С. Цивільна оборона. Навчальний посібник. За редакцією полковника В.С. Франчука. Львів: Афіша. 2000. 336 с

30. Романов Д.В., Осухівська Г.М., Паламар А.М. Функціональна схема системи керування зовнішнім освітленням на основі технології LoRa. *Інформаційні моделі, системи та технології:* матеріали науково-техн. конф., м. Тернопіль, 08-09 грудня 2021 р. ст. 124.

31. Ольховецька Х., Осухівська Г. М. Комп’ютерна система контролю ферментації винних продуктів. *Природничі та гуманітарні науки. Актуальні питання*: матеріали науково-техн. конф., м. Тернопіль, 28-29 кв. 2022 р. ст. 142-143.

32. Ольховецька Х., Осухівська Г.М. Математичне забезпечння системи контролю процесу ферментації. *IX науково-технічна конференція «Інформаційні моделі, системи та технології»*: матеріали науково-техн. конф., м. Тернопіль, 7–8 груд. 2022 р. ст. 140

33. З. Заверуха, Г. Осухівська Дослідження динамічної маршрутизації. *Матеріали XVIII наукової конференції ТНТУ ім. І. Пулюя*: *»*: матеріали науково-техн. конф., м. Тернопіль, 29–30 жовтня. 2014 р. ст. 59

34. О. В. Шевченко, Г. М. Осухівська, М. Я. Горінин Побудова моделі поточного стану елементів комп’ютерної мережі. *Актуальні задачі сучасних технологій*: матеріали науково-техн. конф., м.Тернопіль, 25-26 листопада. 2015 р. ст.70-71.

35. Н. Горячий, А. Луцків, Г. Осухівська, В. Яцишин Основні метрики якості мереж передавання даних. *Інформаційні моделі, системи та технології* : матеріали науково-техн. конф., м.Тернопіль, 12-13 грудня 2018 року. С. 65.

ДОДАТКИ

ДОДАТОК А

Тези конференцій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

ІХ НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

8–9 грудня 2021 року

ТЕРНОПІЛЬ
2021

УДК 004.3:663.252
Х. Ольховецька
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

КОМП'ЮТЕРИЗОВАНА СИСТЕМА КОНТРОЛЮ ЯКОСТІ ПРОЦЕСУ ФЕРМЕНТАЦІЇ ВИННИХ ПРОДУКТІВ

UDC 004.3:663.252
Kh. Olkhovetska

COMPUTERIZED QUALITY CONTROL SYSTEM OF WINE PRODUCTION FERMENTATION PROCESS

Ферментація – це складна хімічна реакція, і цілком природний процес. Цей процес для виноградного соку починається, як тільки порушується цілісність виноградної шкірки: цукор, що міститься в м'якоті стиглої ягоди, вступає в контакт із присутніми на ній дріжджами, а саме Saccharomyces cerevisiae. Під дією мікроскопічних грибків, що викликають бродіння, спочатку виділяється вуглекислий газ і етанол, тобто етиловий спирт, а також утворюються і деякі інші речовини: гліцерин, складні ефіри або ароматичні сполуки, вищі спирти, альдегіди та кислоти [1].

Кожен з етапів виготовлення винних продуктів вимагає ретельного контролю, оскільки саме від цього і залежить їх якість. Проаналізувавши різнобічні фактори впливу на процес переробки винограду – актуальності набуває створення комп'ютеризованих систем для контролю за параметрами та етапами ферментації з метою покращення якості продукції.

Така система дозволяє проводити відбір проб для контролю ферментації без втручання людини. Оскільки відбір проб вручну, в більшості випадків, призводить не тільки до втрати деякої кількості продукції на день, але і до збільшення додаткового втручання в технологічний процес [2–4].

Інформація, що надається системою, про рівень або об'єм дріжджів також дозволяє визначати момент для додавання вхідних речовин у відповідних кількостях, оскільки здійснюється автоматизований контроль моніторингу кінетики бродіння. Що дозволить виявляти збої в технологічному процесі у випадку уповільнення бродіння і, таким чином, запобігати виникненню дефектів у виготовленому вині [3].

Реалізація вищезгаданої комп'ютеризованої системи дає можливість отримати в результаті апаратно-програмний комплекс, який контролює процес ферментації винного продукту та зробить можливим зменшення кількості браку або ж продукції невідповідної якості.

Література.

1. V. Lavric, I. D. Ofiţeru, and A. Woinaroschy, "Continuous hybridoma bioreactor: sensitivity analysis and optimal control," *Biotechnology and Applied Biochemistry*, vol. 44, no. 2, pp. 81–92, 2006.
2. J. M. van Zyl, E. van Rensburg, W. H. van Zyl, T. M. Harms, and L. R. Lynd, "A kinetic model for simultaneous saccharification and fermentation of avicel with Saccharomyces cerevisiae," Biotechnology and Bioengineering, vol. 108, no. 4, pp. 924–933, 2011.
3. Ranasinghe, D.C.; Falkner, N.J.G.; Pan, C.; Wu, H. Wireless sensing platform for remote monitoring and control of wine fermentation. In Proceedings of the IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne, Australia, 2–5 April 2013; pp. 503–508.
4. Di Gennaro, S.F.; Matese, A.; Primicerio, J.; Genesio, L.; Sabatini, F.; di Blasi, S.; Vaccari,F.P. Wireless real-time monitoring of malolactic fermentation in wine barrels: the Wireless Sensor Bung system. Austr. J. Grape Wine Res. 2013, 19, 20–24.

Міністерство освіти і науки України,
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

V МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

28-29 квітня 2022 р.

(збірник тез конференції)

Тернопіль 2022

УДК 004.3:663.252
Ольховецька Х. – ст. гр. СІм-51
Тернопільський національний університет імені Івана Пулюя

КОМП'ЮТЕРНА СИСТЕМА КОНТРОЛЮ ФЕРМЕНТАЦІЇ ВИННИХ ПРОДУКТІВ

Науковий керівник: к.т.н., доцент Осухівська Г.М.

Olkhovetska Kh.
Ternopil Ivan Puluj National Technical University

COMPUTER SYSTEM FOR CONTROLLING WINE PRODUCTS FERMENTATION

Supervisor: Ph.D., Assoc. Prof. Osukhivska H.M.

Ключові слова: комп'ютерна система, контроль, ферментація
Keywords: computer system, control, fermentation

Всі стадії переробки виноградної сировини вимагають неабиякого контролю, оскільки від цього залежить якість вихідного винного продукту, але особливої уваги все ж таки потребує контроль за процесом ферментації. Тому актуального значення набуває розробка комп'ютерних систем, що дозволять за рахунок здійснення моніторингу завчасно попередити відхилення в технологічному процесі, що дає можливість забезпечити виробництво якісної продукції виноперероблки [1].

Інформація, що надається такою системою, дозволяє визначати момент для додавання вхідних речовин та необхідну їх кількість, а також проводити відбір проб для контролю ферментації без втручання людини, що в свою чергу призводить не тільки до зменшення втрат продукції, води і т. ін, але і до зменшення зовнішніх впливів у технологічний процес.

Розроблена комп'ютерна система контролю процесу ферментації зображена на рисунку 1, а її схема з'єднань – на рисунку 2.

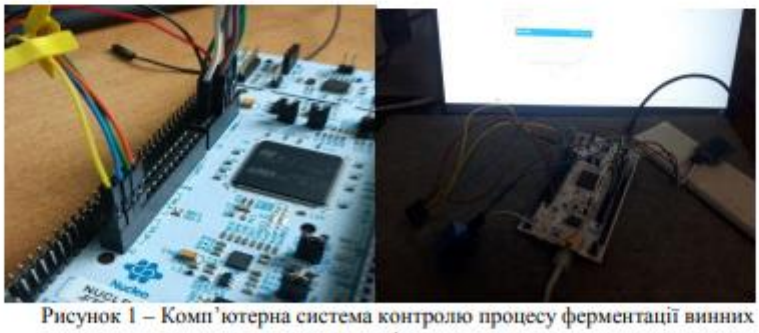

Рисунок 1 – Комп'ютерна система контролю процесу ферментації винних продуктів.

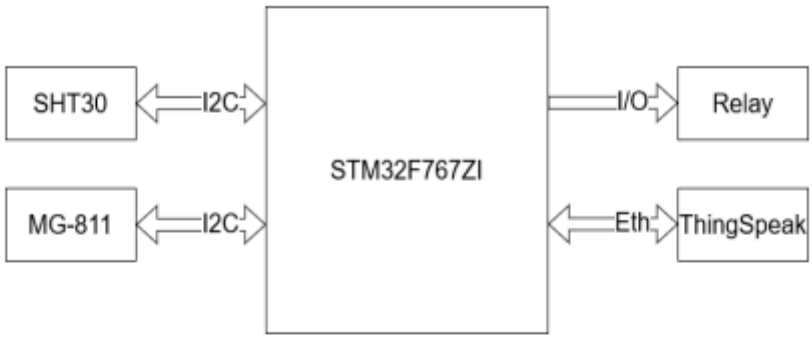

Рисунок 2 – Схема з'єднань комп'ютерної системи.

Запропонована комп'ютерна система контролю процесу ферментації винних продуктів розроблена на базі мікроконтролера STM32. Основний елементом її є STM32F767ZI NUCLEO – як блок опрацювання даних [2], а для контролю за процесом ферментації - використано давачі температури та вологості SHT30 [3] та газові давачі MG-811 [4].

Варто відмітити, що давачі комп'ютерної системи, які необхідні для контролю за процесом ферментації, мають буди поміщені в пробку бочки з вином та під'єднані до контролера, який, в свою чергу, підключений до мережі Ethernet. Контролер кількісно визначає швидкість виділення CO_2 і, на основі цього, розраховує норму споживання цукру. Потім здійснюється регулювання температури на основі підтримки постійної норми цукру. Додавання інших речовини здійснюється відповідно до технологічного процесу виготовлення того чи іншого виду вина та необхідності підтримки процесу ферментації.

В розробленій комп'ютерній системі контролю процесу ферментації винних продуктів отримані дані надсилаються на хмарний сервіс ThingSpeak [4], а результат представляється у вигляді графіків.

Використання розробленої комп'ютерної системи є економічно доцільним, оскільки вона є недорогою, а також може бути використана на старих виробничих потужностях, що актуальне для малих підприємств.

Література

1. Ольховецька Х. Комп'ютеризована система контролю якості процесу ферментації винних продуктів. Інформаційні моделі, системи та технології: матеріали IX науково-технічної конференції (8–9 грудня 2021 р). Тернопіль, 2021. С. 123.

2. STM32F767ZI. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f767zi.html (дата звернення: 10.10.2021).

Datasheet. URL: https://datasheetspdf.com/pdf/1116082/Sensirion/SHT30-DIS/1 (дата звернення: 14.03.2022)

3. Datasheet. URL: https://sandboxelectronics.com/files/SEN-000007/MG811.pdf (дата звернення: 14.03.2022).

4. Thingspeak. Thingspeak. URL: https://thingspeak.com/ (дата звернення: 02.04.2022).

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

X НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

7–8 грудня 2022 року

ТЕРНОПІЛЬ
2022

УДК 004.031.6
Х. Ольховецька, Г. Осухівська
(Тернопільський національний університет імені Івана Пулюя, Україна)

МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ КОНТРОЛЮ ПРОЦЕСУ ФЕРМЕНТАЦІЇ

UDC 004.031.6
K. Olkhovetska, H. Osukhivska

MATHEMATICAL SUPPORT OF THE COMPUTER SYSTEM FOR CONTROL OF THE FERMENTATION PROCESS

При розробці комп'ютерної системи контролю процесу ферментації винних продуктів важливим є вибір та обгрунтування відповідного математичного забезпечення. У літературі можна знайти багато різних математичних систем, що моделюють процес виробництва вина на основі різної кінетики. Зазвичай враховують зміни стану таких показників, як дріжджі, цукор, спирт та концентрація азоту, що засвоївся. Класична модель яка використовується для математичного представлення запропонована David et al. [1], що описується диференціальними рівняннями. Ця модель сформульована з кінетикою Міхаеліса-Ментена [2].

Загалом, для моделювання процесу бродіння вина існує безліч моделей, заснованих на звичайних диференціальних рівняннях. За основу було взято модель, описану в [3], яка представляє еволюцію біомаси дріжджів та інших концентрацій субстратів, таких як азот, цукор та етанол. Цукор перетворюється на етанол, а дріжджі ростуть шляхом метаболізму цукру та інших поживних речовин. З цієї відомої моделі було виведено модель, засновану на звичайних диференціальних рівняннях, яка також враховує "смерть" дріжджів, пов'язану з киснем та етанолом.

Для опису процесу контролю за ферментацією винних продуктів доцільно в моделях враховувати показники температури та CO2 з метою прогнозування проведення технологічних операцій (наприклад, мінімізація старіння дріжджів). Модель, яка пропонується, описується:

$$\int F(t,y(t),u(t))dttc+\tau tc$$

З урахуванням:

$$y(t)=f(t,y(t),u(t),\vec{d},\vec{p}),\forall t\in[tc,tc+\tau\ ]\ y(tc)=\bar{y}(tc)$$
$$\vec{c}(t,\dot{y}(t),\dot{u}(t),\vec{d},\vec{p})\geq 0,\forall t\in[t\vec{c},t\vec{c}+\vec{\tau}\ ],$$

де tc означає поточний момент часу, а точне формулювання $f(t,y(t),u(t))$ залежить від конкретного розглянутого випадку. Для моделі формулювання $f(t,y(t),u(t))$ залежить від цільового стану та контрольних даних, що надходять від попереднього обчислення.

Таким чином, на основі запропонованої моделі розроблено математичне забезпечення комп'ютерної системи контролю процесу ферментації винних продуктів.

Література

1. Shea A. J. Comfort. Oxygen and fermentation. More Wine – Absolutely Everything! for Wine-Making, 2009.
2. R. David, D. Dochain, J.-R. Mouret, A. Vande Wouwer, and J.-M. Sablayrolles. Dynamical modeling of alcoholic fermentation and its linkwith nitrogen consumption. In Proceedings of the 11th International Symposium on Computer Applications in Biotechnology (CAB 2010), 2010, P. 496–501.
3. H. H. Dittrich and M. Großmann. Mikrobiologie des Weines. Ulmer, fourth edition, 2011, 288 p.

ДОДАТОК Б

Лістинги програми

RHEncryptedDriver.cpp

```
// RHEncryptedDriver.cpp
//
// Author: Philippe.Rochat'at'gmail.com
// Contributed to the RadioHead project by the author
// $Id: RHEncryptedDriver.cpp,v 1.7 2020/08/04 09:02:14 mikem Exp $

#include <RadioHead.h>
#ifdef RH_ENABLE_ENCRYPTION_MODULE
#include <RHEncryptedDriver.h>

RHEncryptedDriver::RHEncryptedDriver(RHGenericDriver& driver, BlockCipher&
blockcipher)
  : _driver(driver),
   _blockcipher(blockcipher)
{
  _buffer = (uint8_t *)calloc(_driver.maxMessageLength(), sizeof(uint8_t));
}

bool RHEncryptedDriver::recv(uint8_t* buf, uint8_t* len)
{
  int h = 0; // Index of output _buffer

  bool status = _driver.recv(_buffer, len);
  if (status && buf && len)
  {
        int blockSize = _blockcipher.blockSize(); // Size of blocks used by
encryption
        int nbBlocks = *len / blockSize;       // Number of blocks in that message
        if (nbBlocks * blockSize == *len)
        {
          // Or we have a missmatch ... this is probably not symetrically encrypted
          for (int k = 0; k < nbBlocks; k++)
          {
                // Decrypt each block
                _blockcipher.decryptBlock(&buf[h], &_buffer[k*blockSize]); // Decrypt
that block into buf
                h += blockSize;
#ifdef STRICT_CONTENT_LEN
                if (k == 0)
                {
//                if (buf[0] > *len - 1)
//                      return false; // Bogus payload length
                  *len = buf[0]; // First byte contains length
                  h--;    // First block is of length--
                  memmove(buf, buf+1, blockSize - 1);
                }
#endif
          }
        }
  }

  return status;
}
```

```
bool RHEncryptedDriver::send(const uint8_t* data, uint8_t len)
{
  if (len > maxMessageLength())
        return false;

  bool status = true;
  int blockSize = _blockcipher.blockSize(); // Size of blocks used by encryption

  if (len == 0) // PassThru
        return _driver.send(data, len);

  if (_cipheringBlocks.blockSize != blockSize)
  {
        // Cipher has changed it's block size
        _cipheringBlocks.inputBlock = (uint8_t *)realloc(_cipheringBlocks.inputBlock,
blockSize);
        _cipheringBlocks.blockSize = blockSize;
  }

  int max_message_length = maxMessageLength();
#ifdef STRICT_CONTENT_LEN
  uint8_t nbBlocks = len / blockSize + 1; // How many blocks do we need for that
message
  uint8_t nbBpM = (max_message_length + 1) / blockSize; // Max number of blocks per
message
#else
  uint8_t nbBlocks = (len - 1) / blockSize + 1; // How many blocks do we need for
that message
  uint8_t nbBpM = max_message_length / blockSize; // Max number of blocks per message
#endif
  int k = 0, j = 0; // k is block index, j is original message index
#ifndef ALLOW_MULTIPLE_MSG
#ifdef STRICT_CONTENT_LEN
  for (k = 0; k < nbBpM && k * blockSize < len + 1; k++)
#else
  for (k = 0; k < nbBpM && k * blockSize < len; k++)
#endif
  {
        // k blocks in that message
        int h = 0; // h is block content index
#ifdef STRICT_CONTENT_LEN
        if (k == 0)
          _cipheringBlocks.inputBlock[h++] = len; // put in first byte of first block
the message length
#endif
        while (h < blockSize)
        {
          // Copy each msg byte into inputBlock, and trail with 0 if necessary
          if (j < len)
                _cipheringBlocks.inputBlock[h++] = data[j++];
          else
                _cipheringBlocks.inputBlock[h++] = 0; // Completing with trailing 0
        }
        _blockcipher.encryptBlock(&_buffer[k * blockSize],
_cipheringBlocks.inputBlock); // Cipher that message into _buffer
  }
//  Serial.println(max_message_length);
//  Serial.println(nbBlocks);
//  Serial.println(nbBpM);
//  Serial.println(k);
//  Serial.println(blockSize);
//  printBuffer("single send", _buffer, k * blockSize);
  if (!_driver.send(_buffer, k*blockSize)) // We now send that message with it's new
length
```

```
        status = false;
#else
  uint8_t nbMsg = (nbBlocks * blockSize) / max_message_length + 1; // How many
message do we need

  for (int i = 0; i < nbMsg; i++)
  {
        // i is message index
        for (k = 0; k < nbBpM && k * blockSize < len ; k++)
        {
          // k blocks in that message
          int h = 0;
#ifdef STRICT_CONTENT_LEN
          if (k == 0 && i == 0)
                _cipheringBlocks.inputBlock[h++] = len; // put in first byte of first
block of first message the message length
#endif
          while (h < blockSize)
          {
                // Copy each msg byte into inputBlock, and trail with 0 if necessary
                if (j < len)
                  _cipheringBlocks.inputBlock[h++] = data[j++];
                else
                  _cipheringBlocks.inputBlock[h++] = 0;
          }
          _blockcipher.encryptBlock(&_buffer[k * blockSize],
_cipheringBlocks.inputBlock); // Cipher that message into buffer
        }
//      printBuffer("multiple send", _buffer, k * blockSize);
        if (!_driver.send(_buffer, k * blockSize)) // We now send that message with
it's new length
          status = false;
  }
#endif
  return status;
}

uint8_t RHEncryptedDriver::maxMessageLength()
{
  int driver_len = _driver.maxMessageLength();

#ifndef ALLOW_MULTIPLE_MSG
  driver_len = ((int)(driver_len/_blockcipher.blockSize()) ) *
_blockcipher.blockSize();
#endif

#ifdef STRICT_CONTENT_LEN
  driver_len--;
#endif
  return driver_len;
}

#endif
```

RHHardwareSP1I.cpp

```
// RHHardwareSPI1.h
// Author: Mike McCauley (mikem@airspayce.com)
// Copyright (C) 2011 Mike McCauley
// Contributed by Joanna Rutkowska
// $Id: RHHardwareSPI1.cpp,v 1.16 2016/07/07 00:02:53 mikem Exp mikem $
// This is a copy of the standard SPI node, that is hopefully setup to work on those
processors
```

```
// who have SPI1. Currently I only have it setup for Teensy 3.5/3.6 and LC
#if defined(__arm__) && defined(TEENSYDUINO) && (defined(KINETISL) ||
defined(__MK64FX512__) || defined(__MK66FX1M0__) || defined(__IMXRT1052__)||
defined(__IMXRT1062__))

#include <RHHardwareSPI1.h>

// Declare a single default instance of the hardware SPI interface class
RHHardwareSPI1 hardware_spi1;

#ifdef RH_HAVE_HARDWARE_SPI

RHHardwareSPI1::RHHardwareSPI1(Frequency frequency, BitOrder bitOrder, DataMode
dataMode)
  :
  RHGenericSPI(frequency, bitOrder, dataMode)
{
}

uint8_t RHHardwareSPI1::transfer(uint8_t data)
{
  return SPI1.transfer(data);
}

void RHHardwareSPI1::attachInterrupt()
{
#if (RH_PLATFORM == RH_PLATFORM_ARDUINO)
  SPI1.attachInterrupt();
#endif
}

void RHHardwareSPI1::detachInterrupt()
{
#if (RH_PLATFORM == RH_PLATFORM_ARDUINO)
  SPI1.detachInterrupt();
#endif
}

void RHHardwareSPI1::begin()
{
  // Sigh: there are no common symbols for some of these SPI options across all
platforms
#if (RH_PLATFORM == RH_PLATFORM_ARDUINO) || (RH_PLATFORM == RH_PLATFORM_UNO32) ||
(RH_PLATFORM == RH_PLATFORM_CHIPKIT_CORE)
  uint8_t dataMode;
  if (_dataMode == DataMode0)
        dataMode = SPI_MODE0;
  else if (_dataMode == DataMode1)
        dataMode = SPI_MODE1;
  else if (_dataMode == DataMode2)
        dataMode = SPI_MODE2;
  else if (_dataMode == DataMode3)
        dataMode = SPI_MODE3;
  else
        dataMode = SPI_MODE0;
#if (RH_PLATFORM == RH_PLATFORM_ARDUINO) && defined(__arm__) && defined(CORE_TEENSY)
  // Temporary work-around due to problem where avr_emulation.h does not work
properly for the setDataMode() cal
  SPCR &= ~SPI_MODE_MASK;
#else
 #if (RH_PLATFORM == RH_PLATFORM_ARDUINO) && defined (__arm__) &&
defined(ARDUINO_ARCH_SAMD)
  // Zero requires begin() before anything else :-)
  SPI1.begin();
```

```
 #endif

  SPI1.setDataMode(dataMode);
#endif
#if (RH_PLATFORM == RH_PLATFORM_ARDUINO) && defined(SPI_HAS_TRANSACTION)
  uint32_t frequency32;
  if (_frequency == Frequency16MHz) {
    frequency32 = 16000000;
  } else if (_frequency == Frequency8MHz) {
    frequency32 = 8000000;
  } else if (_frequency == Frequency4MHz) {
    frequency32 = 4000000;
  } else if (_frequency == Frequency2MHz) {
    frequency32 = 2000000;
  } else {
    frequency32 = 1000000;
  }
  _settings = SPISettings(frequency32,
    (_bitOrder == BitOrderLSBFirst) ? LSBFIRST : MSBFIRST,
    dataMode);
#endif


#if (RH_PLATFORM == RH_PLATFORM_ARDUINO) && defined (__arm__) &&
(defined(ARDUINO_SAM_DUE) || defined(ARDUINO_ARCH_SAMD))
  // Arduino Due in 1.5.5 has its own BitOrder :-(
  // So too does Arduino Zero
  ::BitOrder bitOrder;
#else
  uint8_t bitOrder;
#endif
  if (_bitOrder == BitOrderLSBFirst)
        bitOrder = LSBFIRST;
  else
        bitOrder = MSBFIRST;
  SPI1.setBitOrder(bitOrder);
  uint8_t divider;
  switch (_frequency)
  {
        case Frequency1MHz:
        default:
#if F_CPU == 8000000
          divider = SPI_CLOCK_DIV8;
#else
          divider = SPI_CLOCK_DIV16;
#endif
          break;

        case Frequency2MHz:
#if F_CPU == 8000000
          divider = SPI_CLOCK_DIV4;
#else
          divider = SPI_CLOCK_DIV8;
#endif
          break;

        case Frequency4MHz:
#if F_CPU == 8000000
          divider = SPI_CLOCK_DIV2;
#else
          divider = SPI_CLOCK_DIV4;
#endif
          break;
```

```
        case Frequency8MHz:
          divider = SPI_CLOCK_DIV2; // 4MHz on an 8MHz Arduino
          break;

        case Frequency16MHz:
          divider = SPI_CLOCK_DIV2; // Not really 16MHz, only 8MHz. 4MHz on an 8MHz
Arduino
          break;

  }

  SPI1.setClockDivider(divider);
  SPI1.begin();
  // Teensy requires it to be set _after_ begin()
  SPI1.setClockDivider(divider);

#else
 #warning RHHardwareSPI does not support this platform yet. Consider adding it and
contributing a patch.
#endif
}

void RHHardwareSPI1::end()
{
  return SPI1.end();
}

// If our platform is arduino and we support transactions then lets use the begin/end
transaction
#if (RH_PLATFORM == RH_PLATFORM_ARDUINO) && defined(SPI_HAS_TRANSACTION)
void RHHardwareSPI1::beginTransaction()
{
 SPI1.beginTransaction(_settings);
}

void RHHardwareSPI1::endTransaction()
{
 SPI1.endTransaction();
}
 #endif

#endif

#endif
```

RHRouter.cpp

```
// RHRouter.cpp
//
// Define addressed datagram
//
// Part of the Arduino RH library for operating with HopeRF RH compatible
transceivers
// (see http://www.hoperf.com)
// RHDatagram will be received only by the addressed node or all nodes within range
if the
// to address is RH_BROADCAST_ADDRESS
//
// Author: Mike McCauley (mikem@airspayce.com)
// Copyright (C) 2011 Mike McCauley
// $Id: RHRouter.cpp,v 1.10 2020/08/04 09:02:14 mikem Exp $
```

```
#include <RHRouter.h>

RHRouter::RoutedMessage RHRouter::_tmpMessage;

//////////////////////////////////////////////////////////////////////
// Constructors
RHRouter::RHRouter(RHGenericDriver& driver, uint8_t thisAddress)
  : RHReliableDatagram(driver, thisAddress)
{
  _max_hops = RH_DEFAULT_MAX_HOPS;
  _isa_router = true;
  clearRoutingTable();
}

//////////////////////////////////////////////////////////////////////
// Public methods
bool RHRouter::init()
{
  bool ret = RHReliableDatagram::init();
  if (ret)
        _max_hops = RH_DEFAULT_MAX_HOPS;
  return ret;
}

//////////////////////////////////////////////////////////////////////
void RHRouter::setMaxHops(uint8_t max_hops)
{
  _max_hops = max_hops;
}

//////////////////////////////////////////////////////////////////////
void RHRouter::setIsaRouter(bool isa_router)
{
  _isa_router = isa_router;
}
//////////////////////////////////////////////////////////////////////
void RHRouter::addRouteTo(uint8_t dest, uint8_t next_hop, uint8_t state)
{
  uint8_t i;

  // First look for an existing entry we can update
  for (i = 0; i < RH_ROUTING_TABLE_SIZE; i++)
  {
        if (_routes[i].dest == dest)
        {
          _routes[i].dest = dest;
          _routes[i].next_hop = next_hop;
          _routes[i].state = state;
          return;
        }
  }

  // Look for an invalid entry we can use
  for (i = 0; i < RH_ROUTING_TABLE_SIZE; i++)
  {
        if (_routes[i].state == Invalid)
        {
          _routes[i].dest = dest;
          _routes[i].next_hop = next_hop;
          _routes[i].state = state;
          return;
        }
  }
```

```
  // Need to make room for a new one
  retireOldestRoute();
  // Should be an invalid slot now
  for (i = 0; i < RH_ROUTING_TABLE_SIZE; i++)
  {
        if (_routes[i].state == Invalid)
        {
          _routes[i].dest = dest;
          _routes[i].next_hop = next_hop;
          _routes[i].state = state;
        }
  }
}

//////////////////////////////////////////////////////////////////////
RHRouter::RoutingTableEntry* RHRouter::getRouteTo(uint8_t dest)
{
  uint8_t i;
  for (i = 0; i < RH_ROUTING_TABLE_SIZE; i++)
        if (_routes[i].dest == dest && _routes[i].state != Invalid)
          return &_routes[i];
  return NULL;
}

//////////////////////////////////////////////////////////////////////
void RHRouter::deleteRoute(uint8_t index)
{
  // Delete a route by copying following routes on top of it
  memmove(&_routes[index], &_routes[index+1],
          sizeof(RoutingTableEntry) * (RH_ROUTING_TABLE_SIZE - index - 1));
  _routes[RH_ROUTING_TABLE_SIZE - 1].state = Invalid;
}

//////////////////////////////////////////////////////////////////////
void RHRouter::printRoutingTable()
{
#ifdef RH_HAVE_SERIAL
  uint8_t i;
  for (i = 0; i < RH_ROUTING_TABLE_SIZE; i++)
  {
        Serial.print(i, DEC);
        Serial.print(" Dest: ");
        Serial.print(_routes[i].dest, DEC);
        Serial.print(" Next Hop: ");
        Serial.print(_routes[i].next_hop, DEC);
        Serial.print(" State: ");
        Serial.println(_routes[i].state, DEC);
  }
#endif
}

//////////////////////////////////////////////////////////////////////
bool RHRouter::deleteRouteTo(uint8_t dest)
{
  uint8_t i;
  for (i = 0; i < RH_ROUTING_TABLE_SIZE; i++)
  {
        if (_routes[i].dest == dest)
        {
          deleteRoute(i);
          return true;
        }
  }
```

```
  return false;
}

//////////////////////////////////////////////////////////////////
void RHRouter::retireOldestRoute()
{
  // We just obliterate the first in the table and clear the last
  deleteRoute(0);
}

//////////////////////////////////////////////////////////////////
void RHRouter::clearRoutingTable()
{
  uint8_t i;
  for (i = 0; i < RH_ROUTING_TABLE_SIZE; i++)
        _routes[i].state = Invalid;
}


uint8_t RHRouter::sendtoWait(uint8_t* buf, uint8_t len, uint8_t dest, uint8_t flags)
{
  return sendtoFromSourceWait(buf, len, dest, _thisAddress, flags);
}

//////////////////////////////////////////////////////////////////
// Waits for delivery to the next hop (but not for delivery to the final destination)
uint8_t RHRouter::sendtoFromSourceWait(uint8_t* buf, uint8_t len, uint8_t dest,
uint8_t source, uint8_t flags)
{
  if (((uint16_t)len + sizeof(RoutedMessageHeader)) > _driver.maxMessageLength())
        return RH_ROUTER_ERROR_INVALID_LENGTH;

  // Construct a RH RouterMessage message
  _tmpMessage.header.source = source;
  _tmpMessage.header.dest = dest;
  _tmpMessage.header.hops = 0;
  _tmpMessage.header.id = _lastE2ESequenceNumber++;
  _tmpMessage.header.flags = flags;
  memcpy(_tmpMessage.data, buf, len);

  return route(&_tmpMessage, sizeof(RoutedMessageHeader)+len);
}

//////////////////////////////////////////////////////////////////
uint8_t RHRouter::route(RoutedMessage* message, uint8_t messageLen)
{
  // Reliably deliver it if possible. See if we have a route:
  uint8_t next_hop = RH_BROADCAST_ADDRESS;
  if (message->header.dest != RH_BROADCAST_ADDRESS)
  {
        RoutingTableEntry* route = getRouteTo(message->header.dest);
        if (!route)
          return RH_ROUTER_ERROR_NO_ROUTE;
        next_hop = route->next_hop;
  }

  if (!RHReliableDatagram::sendtoWait((uint8_t*)message, messageLen, next_hop))
        return RH_ROUTER_ERROR_UNABLE_TO_DELIVER;

  return RH_ROUTER_ERROR_NONE;
}

//////////////////////////////////////////////////////////////////
// Subclasses may want to override this to peek at messages going past
```

```
void RHRouter::peekAtMessage(RoutedMessage* message, uint8_t messageLen)
{
 // Default does nothing
 (void)message; // Not used
 (void)messageLen; // Not used
}

////////////////////////////////////////////////////////////////////
bool RHRouter::recvfromAck(uint8_t* buf, uint8_t* len, uint8_t* source, uint8_t*
dest, uint8_t* id, uint8_t* flags, uint8_t* hops)
{
  uint8_t tmpMessageLen = sizeof(_tmpMessage);
  uint8_t _from;
  uint8_t _to;
  uint8_t _id;
  uint8_t _flags;
  if (RHReliableDatagram::recvfromAck((uint8_t*)&_tmpMessage, &tmpMessageLen, &_from,
&_to, &_id, &_flags))
  {
        // Here we simulate networks with limited visibility between nodes
        // so we can test routing
#ifdef RH_TEST_NETWORK
        if (
#if RH_TEST_NETWORK==1
          // This network looks like 1-2-3-4
            (_thisAddress == 1 && _from == 2)
          || (_thisAddress == 2 && (_from == 1 || _from == 3))
          || (_thisAddress == 3 && (_from == 2 || _from == 4))
          || (_thisAddress == 4 && _from == 3)

#elif RH_TEST_NETWORK==2
            // This network looks like 1-2-4
            //             | | |
            //             --3--
            (_thisAddress == 1 && (_from == 2 || _from == 3))
          || _thisAddress == 2
          || _thisAddress == 3
          || (_thisAddress == 4 && (_from == 2 || _from == 3))

#elif RH_TEST_NETWORK==3
            // This network looks like 1-2-4
            //             |  |
            //             --3--
            (_thisAddress == 1 && (_from == 2 || _from == 3))
          || (_thisAddress == 2 && (_from == 1 || _from == 4))
          || (_thisAddress == 3 && (_from == 1 || _from == 4))
          || (_thisAddress == 4 && (_from == 2 || _from == 3))

#elif RH_TEST_NETWORK==4
            // This network looks like 1-2-3
            //              |
            //              4
            (_thisAddress == 1 && _from == 2)
          || _thisAddress == 2
          || (_thisAddress == 3 && _from == 2)
          || (_thisAddress == 4 && _from == 2)

#endif
          )
        {
          // OK
        }
        else
        {
```

```
          return false; // Pretend we got nothing
        }
#endif

        peekAtMessage(&_tmpMessage, tmpMessageLen);
        // See if its for us or has to be routed
        if (_tmpMessage.header.dest == _thisAddress || _tmpMessage.header.dest ==
RH_BROADCAST_ADDRESS)
        {
          // Deliver it here
          if (source) *source = _tmpMessage.header.source;
          if (dest)  *dest  = _tmpMessage.header.dest;
          if (id)   *id   = _tmpMessage.header.id;
          if (flags) *flags  = _tmpMessage.header.flags;
          if (hops)  *hops  = _tmpMessage.header.hops;
          uint8_t msgLen = tmpMessageLen - sizeof(RoutedMessageHeader);
          if (*len > msgLen)
                *len = msgLen;
          memcpy(buf, _tmpMessage.data, *len);
          return true; // Its for you!
        }
        else if (  _tmpMessage.header.dest != RH_BROADCAST_ADDRESS
                 && _tmpMessage.header.hops++ < _max_hops)
        {
          // Maybe it has to be routed to the next hop
          // REVISIT: if it fails due to no route or unable to deliver to the next
hop,
          // tell the originator. BUT HOW?

          // If we are forwarding packets, do so. Otherwise, drop.
          if (_isa_router)
            route(&_tmpMessage, tmpMessageLen);
        }
        // Discard it and maybe wait for another
  }
  return false;
}

////////////////////////////////////////////////////////////////////
bool RHRouter::recvfromAckTimeout(uint8_t* buf, uint8_t* len, uint16_t timeout,
uint8_t* source, uint8_t* dest, uint8_t* id, uint8_t* flags, uint8_t* hops)
{
  unsigned long starttime = millis();
  int32_t timeLeft;
  while ((timeLeft = timeout - (millis() - starttime)) > 0)
  {
        if (waitAvailableTimeout(timeLeft))
        {
          if (recvfromAck(buf, len, source, dest, id, flags, hops))
                return true;
        }
        YIELD;
  }
  return false;
}
```

SHT30.cpp

```
#include "mbed.h"
#include "SHT30DISB.h"
```

```
SHT30DISB::SHT30DISB (PinName sda, PinName scl) : _i2c(sda, scl) {
  init();
}
SHT30DISB::SHT30DISB (I2C& p_i2c) : _i2c(p_i2c) {
  init();
}

void SHT30DISB::init(void)
{
  _i2c.frequency(400000);
  buf[0]=0x2c;
  buf[1]=0x06;
  data_buf[6]= _i2c.write(SHT30DISB_ADDR,buf,2);
  buf[0]=0x30;
  buf[1]=0x66;
  data_buf[7]= _i2c.write(SHT30DISB_ADDR,buf,2);

//  put(0x2C,0x06); //Config measurement
//  put(0x30, 0x66); //Heat off

}
void SHT30DISB::put(unsigned char a, unsigned char b)
{
  buf[0]=a;
  buf[1]=b;
  _i2c.write(SHT30DISB_ADDR, buf, 2);
}


void SHT30DISB::get(unsigned char a)
{
  buf[0] = a;
  _i2c.write(SHT30DISB_ADDR, buf, 1, true); // no stop, repeated
  _i2c.read(SHT30DISB_ADDR, buf, 1);

}


void SHT30DISB::readSensor()
{
//
data_buf[0]=_i2c.read(SHT30DISB_ADDR, SHT30DISB_DATA, 6, true); // no
stop, repeated
  buf[0]=0x2c;
  buf[1]=0x06;
  data_buf[0]=_i2c.write(0x8A, buf,2);
  wait(.1);
  data_buf[1]=_i2c.read(0x8A, SHT30DISB_DATA, 6, false);

}
```

```
float SHT30DISB::cTemp(){
  readSensor();
  c_Temp = -
45 + 175 * ((((SHT30DISB_DATA[0] * 256.0) + SHT30DISB_DATA[1])) / 6553
5.0);
  return c_Temp;
}

float SHT30DISB::fTemp(){
  f_Temp = 32 + 1.8 * cTemp();
  return f_Temp;
}

float SHT30DISB::humidity(){

humi = 100 * ((((SHT30DISB_DATA[3] * 256.0) + SHT30DISB_DATA[4])) / 65
535.0);
  return humi;
}
```

SHT30.h

```
#ifndef SHT30DISB_H_
#define SHT30DISB_H_

#define SHT30DISB_ADDR 0x8A
#define SHT30DISB_CLKENHI 0x2C06
#define SHT30DISB_HEATON 0x306D
#define SHT30DISB_HEATOFF 0x3066


#include "mbed.h"
#include "typedef.h"

class SHT30DISB{
public:
  SHT30DISB (PinName sda, PinName scl);
  SHT30DISB (I2C& p_i2c);

  void put(unsigned char a, unsigned char b);
  void get(unsigned char a);
  void readSensor();
  void init();
  float cTemp();
  float fTemp();
  float humidity();
```

ThingSpeak.cpp

```
#include "mbed.h"
#include "Thingspeak.h"

HTTPResult Thingspeak::PostDataToChannel(char* channelWriteAPIKey, int
 fieldNumber, int fieldValue){
  char url[100];

sprintf( url, "https://api.thingspeak.com/update?api_key=%s&field%d=%d
", channelWriteAPIKey, fieldNumber, fieldValue);
  HTTPMap map;
  HTTPText text("", 1);
  map.put("", "");
  return _http.post(url, map, &text,1);
}
```

ThingSpeak.h

```
#ifndef MBED_THINGSPEAK_H
#define MBED_THINGSPEAK_H

#include "mbed.h"
#include "HTTPClient.h"

class Thingspeak {

public:

//@return 0 on success, HTTP error (<0) on failure

HTTPResult PostDataToChannel(char* channelWriteAPIKey, int fieldNumber
, int fieldValue);


private:
  HTTPClient _http;

};

#endif
```