ДОДАТОК А

Апробація результатів роботи

Міністерство освіти і науки України

NURE

Харківський національний університет радіоелектроніки

ЗБІРНИК

студентських наукових статей

«Автоматизація та приладобудування»

«Automation and Development of Electronic Devices»

ADED-2025

(Випуск 1)

[електронне видання]

KITAP

http://nure.ua/department/kafedra-komp-yuterno-integrovanih-tehnologiy-avtomatizatsiyi-ta-mehatroniki-kitam

iTeD

http://itez.zntu.edu.ua/

http://kafea.kdu.edu.ua

Харків 2025

Рисунок А.1 – Титуальний аркуш електронного видання

УДК 630.18

АВТОМАТИЗОВАНІ ДИСПЕНСЕРИ ЛІКІВ: СУЧАСНИЙ СТАН ТА НАПРЯМКИ РОЗВИТКУ

Єрофєєв С.О
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: semen.ierofieiev@nure.ua

Анотація: У статті досліджується роль автоматизованих диспенсерів ліків як сучасного технологічного засобу для вирішення проблеми низької прихильності пацієнтів до призначеної терапії. Розглянуто поточний стан ринку цих пристроїв та їх ключові функціональні типи. Особливу увагу приділено аналізу технічних аспектів автоматизації процесу видачі ліків, включаючи використовувані мікроконтролерні системи, виконавчі механізми та елементи сенсорики. Висвітлено основні виклики та недоліки, властиві існуючим моделям диспенсерів. Окреслено перспективні напрямки розвитку технологій автоматизованої видачі ліків, що мають на меті подолання існуючих недоліків та подальше вдосконалення пристроїв задля підвищення ефективності лікування та покращення якості життя пацієнтів.

Ключові слова: автоматизований диспенсер ліків, прийом медикаментів, медичні технології, мікроконтролер, автоматизація дозування.

AUTOMATED MEDICAMENT DISPENSERS: CURRENT STATE AND DEVELOPMENT TRENDS

S. Yerofieiev
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky av.,14
E-mail: semen.ierofieiev@nure.ua

Annotation: This article explores the role of automated medication dispensers as a modern technological tool for addressing the problem of low patient adherence to prescribed therapy. It reviews the current market state of these devices and their key functional types. Particular attention is paid to analyzing the technical aspects of automating the medication dispensing process, including the microcontroller systems, executive mechanisms, and sensing elements used. The main challenges and disadvantages inherent in existing dispenser models are highlighted. Prospective development directions for automated medication dispensing technologies are outlined, aiming to overcome existing shortcomings and further improve devices to enhance treatment effectiveness and patient quality of life.

Key words: automated medication dispenser, medication adherence, medical technologies, microcontroller, dosage automation.

У сучасному світі, де тривалість життя зростає, а поширеність хронічних захворювань збільшується, питання ефективного управління процесом лікування стає дедалі гострішим. Однією з ключових проблем є забезпечення своєчасного та точного прийому лікарських засобів пацієнтами, що відоме як прихильність до терапії (medication adherence). Низький рівень прихильності може призвести до зниження ефективності лікування, розвитку ускладнень та збільшення навантаження на систему охорони здоров'я. У відповідь на цей виклик, технологічні рішення, зокрема автоматизовані диспенсери ліків, набувають особливої актуальності у 2025 році.

Рисунок А.2 – Зміст статті, 1 сторінка

Автоматизований диспенсер ліків — це пристрій, призначений для зберігання, планування, нагадування та автоматичної видачі ліків у чітко визначений час та дозуванні (рис 1). Ці системи виступають надійними помічниками як для самих пацієнтів, так і для їхніх опікунів чи медичного персоналу, мінімізуючи ризик помилок, пов'язаних із самостійним дозуванням та дотриманням складних графіків прийому.

а) Hero Health б) MedMinder

Рисунок 1 – Автоматизовані диспенсери

Якщо поглянути на сучасний ринок, то автоматизовані диспенсери ліків представлені досить широко. Існують різні моделі, що відрізняються за можливостями, складністю та, звісно, ціною. Можна виділити кілька основних типів. По-перше, це прості, базові автоматичні диспенсери. Зазвичай вони мають механізм на кшталт обертового диска або лотка (рис. 2) з певним числом комірок (наприклад, на тиждень).

Рисунок 2 – Диспенсер барабанного типу

Такі пристрої можуть подавати звукові або візуальні сигнали як нагадування і дають доступ до чергової дози за простим таймером. Їхній функціонал, відверто кажучи, обмежений, але для пацієнтів з нескладними схемами лікування вони є цілком доступним рішенням. По-друге, ми

Рисунок А.3 – Зміст статті, 2 сторінка

бачимо активне зростання сегменту "розумних" або підключених диспенсерів. Це вже пристрої зі складнішою "начинкою", які можуть підключатися до мережі (Wi-Fi, Bluetooth), працювати разом зі смартфонами через мобільні додатки та використовувати хмарні сервіси. А це вже зовсім інші можливості: віддалене налаштування графіка, надсилання сповіщень не лише пацієнту, а й опікунам, ведення обліку прийому ліків, а іноді навіть перевірка наявності препарату в контейнері. Такі системи, як правило, гнучкіші в налаштуванні і можуть працювати зі складнішими схемами прийому. Окремо можна згадати диспенсери для специфічних потреб, розроблені, наприклад, для рідких ліків, ін'єкцій або для використання в медичних закладах (аптеках, лікарнях). Ці моделі вирізняються підвищеною точністю та продуктивністю. Які ж головні тренди ми бачимо на цьому ринку? Це, безумовно, зменшення розмірів пристроїв (мініатюризація), підвищення їхньої надійності та безпеки, а також розширення можливостей для підключення та інтеграції з іншими системами.

Як же працюють сучасні автоматизовані диспенсери? В основі – продумана інтеграція цілого комплексу технологічних рішень. Фактично, "мозком" пристрою є мікроконтролерні системи, які керують усім: відраховують час, запускають механізми, обробляють дії користувача через інтерфейс і, якщо потрібно, комунікують із зовнішнім світом. Для цього використовуються різні платформи, вибір яких залежить від того, яка обчислювальна потужність потрібна та які додаткові функції (периферія) необхідні. А за фізичне переміщення та видачу ліків відповідають виконавчі механізми. Найчастіше тут зустрічаються крокові двигуни, які забезпечують точне позиціонування обертових частин або лінійне переміщення, а також сервомотори або соленоїди, що відкривають/закривають доступ до дози. Не менш важливі і датчики. Вони потрібні для контролю положення механізмів, перевірки наявності ліків у відсіках, фіксації моменту, коли пацієнт забрав дозу. Ці датчики можуть бути оптичними, механічними чи навіть ваговими. Звісно, є і інтерфейс користувача – це може бути екран (найчастіше LCD або OLED), кнопки, поворотні ручки (енкодери) або сенсорні панелі, а також звукові та світлові індикатори. Він забезпечує взаємодію з людиною та слугує для нагадувань. Окремо варто згадати про "розумні" моделі, де активно застосовують модулі бездротового зв'язку (Wi-Fi, Bluetooth) для підключення до смартфонів, домашньої мережі або хмарних платформ, суттєво розширюючи функціонал.

Куди ж рухається розвиток автоматизованих диспенсерів далі? Можна виділити декілька основних напрямків. Наприклад, одним з головних пріоритетів є глибша інтеграція цих пристроїв з іншими елементами екосистеми здоров'я. Ідеться про системи, які зможуть обмінюватися даними з електронними медичними картками, носимими гаджетами для моніторингу стану здоров'я, телемедичними платформами. Це дозволить лікарям майже в реальному часі бачити, наскільки пацієнт дотримується схеми лікування, і оперативно вносити корективи, якщо це потрібно. Крім того, безумовно, буде розширюватися функціонал та зростатиме "інтелектуальність". Застосування елементів штучного інтелекту допоможе аналізувати, як саме пацієнт приймає ліки, прогнозувати можливі пропуски, робити нагадування більш персоналізованими і, можливо, навіть надавати базову інформацію про препарати. І, звісно, важливе завдання — це розробка більш універсальних пристроїв, які зможуть працювати не тільки зі звичними таблетками та капсулами, а й з рідкими формами, порошками, спреями для інгаляцій тощо. Не забуваймо і про такий важливий аспект, як безпека. Оскільки пристрої працюють з чутливою медичною інформацією, критично важливо забезпечити високий рівень захисту даних і відповідати всім регуляторним вимогам. І останнє, але не менш важливе — це питання доступності. Зниження вартості виробництва та спрощення використання є ключовими для того, щоб такі диспенсери стали доступнішими для ширшого кола людей, включаючи тих, хто має певні обмеження.

Рисунок А.4 – Зміст статті, 3 сторінка

Так, переваг автоматизованих диспенсерів чимало, і вони постійно вдосконалюються. Але варто говорити і про виклики та недоліки, які ще є у сучасних автоматизованих диспенсерів ліків. Саме вони часто стають на заваді їх повсюдному впровадженню та ефективному використанню різними категоріями користувачів. Серед найбільш суттєвих проблем можна виділити наступні:

- **Вартість.** Передові "розумні" моделі з широким набором функцій можуть коштувати досить дорого. Це робить їх недоступними для значної частини пацієнтів, особливо вразливих верств населення або в регіонах з обмеженим фінансуванням охорони здоров'я.
- **Складність використання.** Незважаючи на зусилля розробників, інтерфейс деяких пристроїв та процес їх початкового налаштування можуть бути занадто складними для літніх людей або тих, хто не має досвіду роботи з сучасною технікою. Потрібен простий та інтуїтивно зрозумілий дизайн.
- **Обмеженість типів ліків.** Як правило, більшість поширених моделей розраховані лише на таблетки та капсули. Це створює проблему для пацієнтів, яким потрібні ліки в інших формах (сиропи, порошки, інгалятори, спреї). Універсальних рішень поки що бракує.
- **Залежність від умов.** "Розумні" диспенсери потребують стабільного електроживлення та доступу до Інтернету (Wi-Fi/Bluetooth). Проблеми з електрикою чи зв'язком можуть обмежити їх функціональність, зокрема віддалений моніторинг та сповіщення.
- **Проблема завантаження ліків.** Хоча видача ліків автоматизована, сам процес заповнення диспенсера, особливо з багатьма відсіками та різними препаратами, залишається ручним. Це вимагає високої уважності і зберігає ризик помилки "людського фактору" на етапі підготовки пристрою до роботи.

ВИСНОВКИ. Підсумовуючи сказане, автоматизовані диспенсери ліків – це сьогодні дійсно важливий крок вперед. Вони стають ефективним інструментом у боротьбі з такою серйозною проблемою, як низька прихильність пацієнтів до лікування. Ми бачимо, як ринок цих пристроїв розвивається: від простих таймерів з лотками до складних "розумних" систем, що вміють нагадувати, планувати і навіть спілкуватися з іншими медичними сервісами. В основі всього цього – продумана робота електроніки, механіки та програмного забезпечення, яка дозволяє автоматизувати процес видачі ліків.

Прогрес очевидний, але чи є ці пристрої ідеальними? На жаль, ні. Як ми вже розглядали, існують цілком конкретні проблеми, які обмежують їх широке застосування. Це і досить висока ціна, яка робить "розумні" моделі недоступними для багатьох. Це і складність, з якою можуть зіткнутися літні люди при налаштуванні та використанні. Це і обмеженість у роботі – більшість поширених моделей призначені тільки для таблеток або капсул, не підтримуючи інші форми ліків. Плюс, не забуваймо про нюанси із завантаженням ліків, яке часто залишається ручним, та залежність "розумних" пристроїв від стабільного зв'язку та електрики.

Саме ці невирішені питання і спонукають до подальших розробок. Мета нашого проєкту – це спроба створити автоматизований диспенсер, який зможе подолати частину з цих недоліків. Що ми конкретно плануємо зробити? По-перше, попрацювати над тим, щоб механізм видачі міг справлятися не тільки з пігулками, а й з іншими формами ліків, підвищуючи універсальність пристрою. По-друге, зосередитись на створенні такого інтерфейсу та логіки роботи, щоб пристрій був простим і зрозумілим для будь-кого, незалежно від технічних навичок, зменшуючи таким чином поріг входу для літніх користувачів. І, звичайно, намагатимемося знайти оптимальні компоненти, щоб зробити пристрій більш доступним за ціною, підвищуючи його загальну доступність.

Рисунок А.5 – Зміст статті, 4 сторінка

ЛІТЕРАТУРА

1. Hughes, C. M., Barry, H. E., & Bradley, M. C. (2007). Effect of a personal automated dose-dispensing system on adherence: a case series. *Drugs & Aging*, 24(10), 911–918. https://pubmed.ncbi.nlm.nih.gov/17338479/

2. Tappen, R. M., Williams, C. L., & Fishman, M. (2017). Enhanced Adherence in Patients Using an Automated Home Medication Dispenser. *Journal of Gerontological Nursing*, 43(8), 30–35. https://pubmed.ncbi.nlm.nih.gov/28749791/

3. Jokanovic, N., Tan, E. C. K., van den Bosch, D., Kirkpatrick, C. M., Dooley, M. J., & Bell, J. S. (2021). Medication adherence support of an in-home electronic medication dispensing system for individuals living with chronic conditions: a pilot randomized controlled trial. *BMC Geriatrics*, 21, Article 10. https://pmc.ncbi.nlm.nih.gov/articles/PMC7807760/

4. Bourgeois, F. T., Shannon, M. W., Valim, C., & Mandl, K. D. (2018). The impact of automation on the safety of drug dispensing in nursing homes. *Journal of the American Medical Informatics Association*, 25(7), 865–872. https://pubmed.ncbi.nlm.nih.gov/29959837/

5. Berdot, S., Korb, P., & Sabatier, B. (2014). Effect of automated drug distribution systems on medication error rates in a short-stay geriatric unit. *Drugs & Aging*, 31(10), 849–857. https://pubmed.ncbi.nlm.nih.gov/24917185/

6. Aldawish, A., & Alshammari, A. (2023). Reducing Medication Errors by Adopting Automatic Dispensing Cabinets in Critical Care Units. *Cureus*, 15(4), e37403. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10136387/

7. Aalto, U. L., & Saari, M. (2021). User Perception of Automated Dose Dispensed Medicine in Home Care: A Scoping Review. *Journal of Medical Internet Research*, 23(10), e25121. https://pmc.ncbi.nlm.nih.gov/articles/PMC8544441/

8. Hero Health. (2025). Smart Pill Dispenser. Retrieved from https://herohealth.com/

9. MedMinder. (2025). Automatic Pill Dispensers and Medication Management. Retrieved from https://medminder.com/

10. Chala, O., Yevsieiev, V., Maksymova, S., & Abu-Jassar, A. (2025). MATHEMATICAL MODEL BASED ON MULTI-AGENT REINFORCEMENT LEARNING (MARL) AND PARTIALLY OBSERVABLE MARKOV DECISION PROCESS (POMDP) FOR MODELING CARGO MOVEMENT FOR A MOBILE ROBOTS GROUP. *Multidisciplinary Journal of Science and Technology*, *5*(4), 480-489.

11. Attar, H., Abu-Jassar, A. T., Yevsieiev, V., Lyashenko, V., Nevliudov, I., & Luhach, A. K. (2022). Zoomorphic mobile robot development for vertical movement based on the geometrical family caterpillar. *Computational intelligence and neuroscience*, *2022*(1), 3046116.

12. Attar, H., Abu-Jassar, A. T., Amer, A., Lyashenko, V., Yevsieiev, V., & Khosravi, M. R. (2022). Control system development and implementation of a CNC laser engraver for environmental use with remote imaging. *Computational intelligence and neuroscience*, *2022*(1), 9140156.

13. Abu-Jassar, A. T., Attar, H., Yevsieiev, V., Amer, A., Demska, N., Luhach, A. K., & Lyashenko, V. (2022). Electronic user authentication key for access to HMI/SCADA via unsecured internet networks. *Computational intelligence and neuroscience*, *2022*(1), 5866922.

Науковий керівник: *Гурін Дмитро, доцент кафедри КІТАР Харківського національного університету радіоелектроніки.*

Рисунок А. – Зміст статті, 5 сторінка

ДОДАТОК Б

Частковий лістинг програмного коду

```
// === БІБЛІОТЕКИ ТА ГЛОБАЛЬНІ ВИЗНАЧЕННЯ ===
#include <Arduino.h>
#include <Wire.h>
#include <U8g2lib.h>
#include "AiEsp32RotaryEncoder.h"
#include "AiEsp32RotaryEncoderNumberSelector.h"
#include "RTClib.h"
#include "esp_adc_cal.h"
#include "driver/adc.h"
#include "esp_err.h"
#include <Preferences.h>
#include "icons.h"
#include <ESP.h>
#include <ESP32Servo.h>
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <AsyncTCP.h>
#include <ArduinoJson.h>

// === ВИЗНАЧЕННЯ ПІНІВ ===
#define ROTARY_ENCODER_A_PIN 33
#define ROTARY_ENCODER_B_PIN 32
#define ROTARY_ENCODER_BUTTON_PIN 27
#define ROTARY_ENCODER_VCC_PIN -1
#define ROTARY_ENCODER_STEPS 4
#define BUZZER_PIN 16
#define ADC_READ_PIN 34
#define STEPPER_STEP_PIN 13
#define STEPPER_DIR_PIN 14
#define STEPPER_ENABLE_PIN 26
#define STEPPER_MS1_PIN 25
#define STEPPER_MS2_PIN 17
#define STEPPER_MS3_PIN 18
#define SERVO_PIN 15
#define OPTICAL_SENSOR_PIN 35

// === ПРОТОТИПИ ФУНКЦІЙ ===
void IRAM_ATTR readEncoderISR();
void setServoPosition(int angle);
void loadSettings();
void saveSettings();
void connectToWiFi();
void startAPMode();
void clearWifiCredentials();
void factoryResetSettings();
void checkAndResetDailyFlags();
```

```
void checkSchedules();
void displayCellCalibrationMenu();
void displaySettings();
void displayDispensingConfirmationScreen();
void displayWifiOptionsMenu();
void displayWifiAPSetupScreen();
void displayWifiConnectingScreen();
void displayWifiStatusScreen();
bool moveToNextCellBySensor();
bool executeThreePhaseSensorMovement(int offsetSteps, bool
updateCellNumbersOnSuccess, bool isHomingSequence);
void performSingleStep(bool direction);
bool moveToDispenseCell(int targetCell);
void playDispenseAlertBeep();
void startMainWebServer();
void notFound(AsyncWebServerRequest *request);
void configureNumberSelectorForState(AppState state);
void displayMain();
void displayMenu();
void displayMode();
void displaySchedule();
void displayBuzzer();
void displayTime();
void displayDate();
void encoderLoop();
void beep(int durationMs);
const unsigned char* getBatteryIcon();
void logo();
void handleNavigation(EventType event);

// === ОСНОВНІ ФУНКЦІЇ: SETUP ТА LOOP ===
// --- Функція setup ---
void setup() {
    Wire.begin();
    u8g2.begin();
    u8g2.enableUTF8Print();

    pinMode(BUZZER_PIN, OUTPUT);
    digitalWrite(BUZZER_PIN, HIGH);

    pinMode(STEPPER_STEP_PIN, OUTPUT);
    pinMode(STEPPER_DIR_PIN, OUTPUT);
    pinMode(STEPPER_ENABLE_PIN, OUTPUT);
    digitalWrite(STEPPER_ENABLE_PIN, LOW);
    pinMode(STEPPER_MS1_PIN, OUTPUT);
    pinMode(STEPPER_MS2_PIN, OUTPUT);
    pinMode(STEPPER_MS3_PIN, OUTPUT);
    digitalWrite(STEPPER_MS1_PIN, LOW);
    digitalWrite(STEPPER_MS2_PIN, HIGH);
    digitalWrite(STEPPER_MS3_PIN, HIGH);
```

```
    pinMode(OPTICAL_SENSOR_PIN, INPUT_PULLUP);

    medicationServo.attach(SERVO_PIN);
    setServoPosition(SERVO_CLOSED_POSITION);

    logo();

    if (! rtc.begin()) {
        u8g2.clearBuffer();
        u8g2.setFont(u8g2_font_ncenB08_tr);
        u8g2.drawStr(0,15,"RTC not found!");
        u8g2.sendBuffer();
        while (1) delay(10);
    }
    lastCheckedRtcDay = rtc.now().day();
    for (int i = 0; i < MAX_SCHEDULES; i++) {
        scheduleActionedToday[i] = false;
    }

    adc1_config_width(ADC_WIDTH_BIT_12);
    adc1_config_channel_atten(adc_channel_battery, atten);
    esp_adc_cal_characterize(unit, atten, ADC_WIDTH_BIT_12, 1100,
&adc1_chars);
    getBatteryIcon();

    encoder->begin();
    encoder->setup(readEncoderISR);
    numberSelector.attachEncoder(encoder);

    loadSettings();

    preferences.begin("wifi_creds", true);
    currentSsid = preferences.getString(KEY_WIFI_SSID, "");
    currentPassword = preferences.getString(KEY_WIFI_PASS, "");
    preferences.end();

    bool cellSettingsWereJustInitialized = false;
    preferences.begin("app_settings", true);
    if (!preferences.isKey(KEY_CURRENT_CELL)) {
        preferences.end();
        currentCellNumber = 0;
        nextDispenseCellNumber = 1;
        cellSettingsWereJustInitialized = true;
    } else {
        preferences.end();
    }

    if (cellSettingsWereJustInitialized) {
        preferences.begin("app_settings", false);
        preferences.putInt(KEY_CURRENT_CELL, currentCellNumber);
```

```
        preferences.putInt(KEY_NEXT_DISP_CELL,
nextDispenseCellNumber);
        preferences.end();
    }

    if (currentSsid.length() > 0) {
        connectToWiFi();
    } else {
        startAPMode();
    }

    appState = STATE_MAIN_SCREEN;
    configureNumberSelectorForState(appState);
    lastInteractionTime = millis();
}

// --- Функція loop ---
void loop() {
    encoderLoop();
    checkAndResetDailyFlags();

    if (appState == STATE_MAIN_SCREEN) {
        checkSchedules();
    }

    if (appState != STATE_MAIN_SCREEN &&
        appState != STATE_DISPENSING_CONFIRMATION &&
        appState != STATE_DISPENSING_ACTIVE &&
        appState != STATE_WIFI_AP_SETUP &&
        appState != STATE_WIFI_CONNECTING &&
        (millis() - lastInteractionTime > inactivityTimeoutMs)) {

        if (appState == STATE_MODE_MENU || appState ==
STATE_BUZZER_MENU) saveSettings();
        if (appState == STATE_SCHEDULE_MENU) { scheduleEditField =
-1; editingScheduleIndex = 0; saveSettings(); }
        if (appState == STATE_TIME_MENU) timeEditField = -1;
        if (appState == STATE_DATE_MENU) dateEditField = -1;

        appState = STATE_MAIN_SCREEN;
        configureNumberSelectorForState(STATE_MAIN_SCREEN);
        lastInteractionTime = millis();
    }

    if (appState == STATE_DISPENSING_CONFIRMATION) {
        if (millis() - dispensingStartTime >
dispensingConfirmationTimeoutMs) {
            appState = STATE_MAIN_SCREEN;
            configureNumberSelectorForState(appState);
            beep(150);
        }
```

```
    }
    else if (appState == STATE_DISPENSING_ACTIVE) {
        if (millis() - dispensingStartTime >
SERVO_DISPENSE_DURATION_MS) {
            setServoPosition(SERVO_CLOSED_POSITION);
            appState = STATE_MAIN_SCREEN;
            configureNumberSelectorForState(appState);
        }
    }

    u8g2.firstPage();
    do {
        switch (appState) {
            case STATE_MAIN_SCREEN: displayMain(); break;
            case STATE_MAIN_MENU: displayMenu(); break;
            case STATE_MODE_MENU: displayMode(); break;
            case STATE_SCHEDULE_MENU: displaySchedule(); break;
            case STATE_BUZZER_MENU: displayBuzzer(); break;
            case STATE_TIME_MENU: displayTime(); break;
            case STATE_DATE_MENU: displayDate(); break;
            case STATE_CELL_CALIBRATION_MENU:
displayCellCalibrationMenu(); break;
            case STATE_SETTINGS_MENU: displaySettings(); break;
            case STATE_DISPENSING_CONFIRMATION:
displayDispensingConfirmationScreen(); break;
            case STATE_DISPENSING_ACTIVE:
                u8g2.clearBuffer();
                u8g2.setFont(u8g2_font_ncenB08_tr);
                u8g2.drawStr((128 -
u8g2.getStrWidth("Dispensing..."))/2, 35, "Dispensing...");
                break;
            case STATE_WIFI_OPTIONS_MENU: displayWifiOptionsMenu();
break;
            case STATE_WIFI_AP_SETUP: displayWifiAPSetupScreen();
break;
            case STATE_WIFI_CONNECTING:
displayWifiConnectingScreen(); break;
            case STATE_WIFI_STATUS: displayWifiStatusScreen();
break;
        }
    } while (u8g2.nextPage());
}

// === ДОПОМІЖНІ ФУНКЦІЇ ===
// --- Функція для звукового сигналу ---
void beep(int durationMs) {
    if (buzzerSetting == 1) {
        digitalWrite(BUZZER_PIN, LOW);
        delay(durationMs);
        digitalWrite(BUZZER_PIN, HIGH);
    }
```

```
}

// --- Функція для відображення логотипу ---
void logo() {
    int counter = 0;
    u8g2.clearBuffer();
    u8g2.drawXBM(0, 0, 128, 64, logoBackround);
    while (counter < 14){
        u8g2.drawXBM(44, 16, 41, 33, logoFrogArr[counter]);
        counter++;
        delay(110);
        u8g2.sendBuffer();
    }
    beep(300);
    delay(500);
}

// === ОБРОБКА ЕНКОДЕРА ТА КНОПКИ ===
// --- ISR для обробки обертання енкодера ---
void IRAM_ATTR readEncoderISR() {
    encoder->readEncoder_ISR();
}

// --- Обробник короткого натискання кнопки ---
void onButtonShortClick() {
    handleNavigation(EVENT_SHORT_PRESS);
}

// --- Обробник довгого натискання кнопки ---
void onButtonLongClick() {
    handleNavigation(EVENT_LONG_PRESS);
}

// --- Функція для обробки стану кнопки енкодера ---
void handleEncoderButton() {
    static unsigned long lastTimeButtonDown = 0;
    static bool wasButtonDown = false;
    static bool longPressHandled = false;

    bool isEncoderButtonDown = encoder->isEncoderButtonDown();

    if (isEncoderButtonDown) {
        if (!wasButtonDown) {
            lastTimeButtonDown = millis();
            longPressHandled = false;
        }
        wasButtonDown = true;
        if (millis() - lastTimeButtonDown >=
longPressAfterMiliseconds && !longPressHandled) {
            onButtonLongClick();
            longPressHandled = true;
```

```
        }
        if(longPressHandled) return;

    } else {
        if (wasButtonDown) {
            if (!longPressHandled && (millis() - lastTimeButtonDown
>= shortPressAfterMiliseconds)) {
                onButtonShortClick();
            }
        }
        wasButtonDown = false;
        longPressHandled = false;
    }
}

// --- Функція для обробки енкодера ---
void encoderLoop() {
    if (encoder->encoderChanged()) {
        handleNavigation(EVENT_ENCODER_ROTATE);
    }
    handleEncoderButton();
}

// === ЛОГІКА СТАНІВ ТА НАВІГАЦІЯ ===
// --- Головна функція обробки навігації ---
void handleNavigation(EventType event) {
    if (event != EVENT_NONE) {
        lastInteractionTime = millis();
    }

    if (event == EVENT_ENCODER_ROTATE) {
        long value = numberSelector.getValue();
        bool valueChanged = false;
        switch (appState) {
            case STATE_MAIN_MENU: if (selectedMenuItem != value) {
selectedMenuItem = value; valueChanged = true; } break;
            case STATE_MODE_MENU: if (modeSetting != value) {
modeSetting = value; valueChanged = true; } break;
            case STATE_SCHEDULE_MENU:
                if (scheduleEditField == -1) { if
(editingScheduleIndex != value) { editingScheduleIndex = value;
valueChanged = true; } }
                else if (scheduleEditField == 0) { if
(scheduleHours[editingScheduleIndex] != value) {
scheduleHours[editingScheduleIndex] = value; valueChanged = true; }
}
                else if (scheduleEditField == 1) { if
(scheduleMinutes[editingScheduleIndex] != value) {
scheduleMinutes[editingScheduleIndex] = value; valueChanged = true;
} }
                break;
```

```
            case STATE_BUZZER_MENU: if (buzzerSetting != value) {
buzzerSetting = value; valueChanged = true; } break;
            case STATE_TIME_MENU:
                if (timeEditField == 0) { if (editHour != value) {
editHour = value; valueChanged = true; } }
                else if (timeEditField == 1) { if (editMinute !=
value) { editMinute = value; valueChanged = true; } }
                else if (timeEditField == 2) { if (editSecond !=
value) { editSecond = value; valueChanged = true; } }
                break;
            case STATE_DATE_MENU:
                if (dateEditField == 0) { if (editDay != value) {
editDay = value; valueChanged = true; } }
                else if (dateEditField == 1) { if (editMonth !=
value) { editMonth = value; valueChanged = true; } }
                else if (dateEditField == 2) { if (editYear !=
value) { editYear = value; valueChanged = true; } }
                break;
            case STATE_SETTINGS_MENU: if (selectedSettingsMenuItem
!= value) { selectedSettingsMenuItem = value; valueChanged = true;
} break;
            case STATE_CELL_CALIBRATION_MENU:
                if (selectedCellMenuItem != value) {
selectedCellMenuItem = value; valueChanged = true; }
                break;
            case STATE_WIFI_OPTIONS_MENU:
                if (selectedWifiOptionsMenuItem != value) {
selectedWifiOptionsMenuItem = value; valueChanged = true; }
                break;
            default: break;
        }
        if (valueChanged) { beep(20); }
    }

    switch (appState) {
        case STATE_MAIN_SCREEN:
            if (event == EVENT_SHORT_PRESS) {
                appState = STATE_MAIN_MENU; selectedMenuItem = 0;
                configureNumberSelectorForState(STATE_MAIN_MENU);
beep(50);
            }
            break;
        case STATE_MAIN_MENU:
            if (event == EVENT_SHORT_PRESS) {
                beep(50);
                switch (selectedMenuItem) {
                    case 0: appState = STATE_MODE_MENU; break;
                    case 1: appState = STATE_SCHEDULE_MENU;
editingScheduleIndex = 0; scheduleEditField = -1; break;
                    case 2: appState = STATE_BUZZER_MENU; break;
```

```
                    case 3: appState = STATE_TIME_MENU;
timeEditField = -1; { DateTime now = rtc.now(); editHour =
now.hour(); editMinute = now.minute(); editSecond = now.second(); }
break;
                    case 4: appState = STATE_DATE_MENU;
dateEditField = -1; { DateTime now = rtc.now(); editDay =
now.day(); editMonth = now.month(); editYear = now.year(); } break;
                    case 5: appState = STATE_CELL_CALIBRATION_MENU;
selectedCellMenuItem = 0; break;
                    case 6: appState = STATE_SETTINGS_MENU;
selectedSettingsMenuItem = 0; break;
                }
                configureNumberSelectorForState(appState);
            } else if (event == EVENT_LONG_PRESS || event ==
EVENT_TIMEOUT) {
                appState = STATE_MAIN_SCREEN;
configureNumberSelectorForState(STATE_MAIN_SCREEN); if(event ==
EVENT_LONG_PRESS) beep(100);
            }
            break;
        case STATE_MODE_MENU:
        case STATE_BUZZER_MENU:
            if (event == EVENT_SHORT_PRESS) {
                if (appState == STATE_MODE_MENU || appState ==
STATE_BUZZER_MENU) {
                    saveSettings(); appState = STATE_MAIN_MENU;
configureNumberSelectorForState(STATE_MAIN_MENU); beep(50);
                }
            } else if (event == EVENT_LONG_PRESS) {
                if (appState == STATE_MODE_MENU || appState ==
STATE_BUZZER_MENU) saveSettings();
                appState = STATE_MAIN_MENU;
                configureNumberSelectorForState(STATE_MAIN_MENU);
                beep(100);
            }
            break;
        case STATE_SCHEDULE_MENU:
            if (event == EVENT_SHORT_PRESS) {
                beep(50);
                if (scheduleEditField == -1) {
                    scheduleEditField = 0;
                } else if (scheduleEditField == 0) {
                    scheduleEditField = 1;
                } else if (scheduleEditField == 1) {
                    scheduleEditField = -1;
                    saveSettings();
                }

configureNumberSelectorForState(STATE_SCHEDULE_MENU);
            } else if (event == EVENT_LONG_PRESS) {
                scheduleEditField = -1; editingScheduleIndex = 0;
```

```
                saveSettings();
                appState = STATE_MAIN_MENU;
                configureNumberSelectorForState(STATE_MAIN_MENU);
                beep(100);
            }
            break;
        case STATE_TIME_MENU:
            if (event == EVENT_SHORT_PRESS) {
                beep(50);
                if (timeEditField == -1) { timeEditField = 0; }
                else {
                    timeEditField++;
                    if (timeEditField > 2) {
                        timeEditField = -1;
                        rtc.adjust(DateTime(rtc.now().year(),
rtc.now().month(), rtc.now().day(), editHour, editMinute,
editSecond));
                        appState = STATE_MAIN_MENU;
                    }
                }
                configureNumberSelectorForState(appState ==
STATE_MAIN_MENU ? appState : STATE_TIME_MENU);
            } else if (event == EVENT_LONG_PRESS) {
                timeEditField = -1;
                appState = STATE_MAIN_MENU;
                configureNumberSelectorForState(STATE_MAIN_MENU);
                beep(100);
            }
            break;
        case STATE_DATE_MENU:
            if (event == EVENT_SHORT_PRESS) {
                beep(50);
                if (dateEditField == -1) { dateEditField = 0; }
                else {
                    dateEditField++;
                    if (dateEditField > 2) {
                        dateEditField = -1;
                        bool dateValid = true;
                        if (editMonth < 1 || editMonth > 12 ||
editDay < 1) dateValid = false;
                        else if (editMonth == 2) {
                            if ((editYear % 4 == 0 && editYear %
100 != 0) || (editYear % 400 == 0)) {
                                if (editDay > 29) dateValid =
false;
                            } else {
                                if (editDay > 28) dateValid =
false;
                            }
                        } else if (editMonth == 4 || editMonth == 6
|| editMonth == 9 || editMonth == 11) {
```

```
                            if (editDay > 30) dateValid = false;
                        } else {
                            if (editDay > 31) dateValid = false;
                        }

                        if(dateValid)
rtc.adjust(DateTime(editYear,editMonth,editDay,rtc.now().hour(),rtc
.now().minute(),rtc.now().second()));
                        else {beep(500);}
                        appState = STATE_MAIN_MENU;
                    }
                }
                configureNumberSelectorForState(appState ==
STATE_MAIN_MENU ? appState : STATE_DATE_MENU);
            } else if (event == EVENT_LONG_PRESS) {
                dateEditField = -1;
                appState = STATE_MAIN_MENU;
                configureNumberSelectorForState(STATE_MAIN_MENU);
                beep(100);
            }
            break;
        case STATE_CELL_CALIBRATION_MENU:
            if (event == EVENT_SHORT_PRESS) {
                beep(50);
                switch (selectedCellMenuItem) {
                    case 0:
                        moveToNextCellBySensor();
                        break;
                    case 1:
                        currentCellNumber = HOME_CELL_NUMBER;
                        nextDispenseCellNumber = 1;
                        saveSettings();

                        appState = STATE_MAIN_MENU;

configureNumberSelectorForState(STATE_MAIN_MENU);
                        break;
                }
            } else if (event == EVENT_LONG_PRESS) {
                appState = STATE_MAIN_MENU;
                configureNumberSelectorForState(STATE_MAIN_MENU);
                beep(100);
            }
            break;
        case STATE_SETTINGS_MENU:
            if (event == EVENT_SHORT_PRESS) {
                beep(50);
                switch (selectedSettingsMenuItem) {
                    case 0:
                        appState = STATE_WIFI_OPTIONS_MENU;
                        selectedWifiOptionsMenuItem = 0;
```

```
                        break;
                    case 1:
                        factoryResetSettings();
                        break;
                }
                configureNumberSelectorForState(appState);
            } else if (event == EVENT_LONG_PRESS) {
                appState = STATE_MAIN_MENU;
                configureNumberSelectorForState(STATE_MAIN_MENU);
                beep(100);
            }
            break;
        case STATE_WIFI_OPTIONS_MENU:
            if (event == EVENT_SHORT_PRESS) {
                beep(50);
                switch (selectedWifiOptionsMenuItem) {
                    case 0:
                        appState = STATE_WIFI_STATUS;
                        break;
                    case 1:
                        clearWifiCredentials();
                        break;
                }
                if (appState != STATE_MAIN_SCREEN) {
                    configureNumberSelectorForState(appState);
                }
            } else if (event == EVENT_LONG_PRESS) {
                appState = STATE_SETTINGS_MENU;

configureNumberSelectorForState(STATE_SETTINGS_MENU);
                 if(event == EVENT_LONG_PRESS) beep(100);
            }
            break;
        case STATE_WIFI_AP_SETUP:
            if (event == EVENT_LONG_PRESS) {
                appState = STATE_WIFI_OPTIONS_MENU;
                configureNumberSelectorForState(appState);
                beep(100);
            }
            break;
        case STATE_WIFI_CONNECTING:
            if (event == EVENT_LONG_PRESS) {
                WiFi.disconnect(true);
                appState = STATE_WIFI_OPTIONS_MENU;
                configureNumberSelectorForState(appState);
                beep(100);
            }
            break;
        case STATE_WIFI_STATUS:
             if (event == EVENT_LONG_PRESS) {
                appState = STATE_WIFI_OPTIONS_MENU;
```

```
                configureNumberSelectorForState(appState);
                beep(100);
            }
            break;
        case STATE_DISPENSING_CONFIRMATION:
            if (event == EVENT_SHORT_PRESS) {
                beep(100);
                appState = STATE_DISPENSING_ACTIVE;
                dispensingStartTime = millis();
                setServoPosition(SERVO_OPEN_POSITION);
                configureNumberSelectorForState(appState);
            } else if (event == EVENT_TIMEOUT || event ==
EVENT_LONG_PRESS) {
                appState = STATE_MAIN_SCREEN;
                configureNumberSelectorForState(appState);
                if(event == EVENT_LONG_PRESS) beep(100);
            }
            break;
    }
}

// === УПРАВЛІННЯ НАЛАШТУВАННЯМИ (PREFERENCES) ===
// --- Збереження налаштувань ---
void saveSettings() {
    preferences.begin("app_settings", false);
    preferences.putInt("mode", modeSetting);
    preferences.putInt("buzzer", buzzerSetting);
    preferences.putBytes("sched_h", scheduleHours,
sizeof(scheduleHours));
    preferences.putBytes("sched_m", scheduleMinutes,
sizeof(scheduleMinutes));
    preferences.putInt(KEY_CURRENT_CELL, currentCellNumber);
    preferences.putInt(KEY_NEXT_DISP_CELL, nextDispenseCellNumber);
    preferences.end();
}

// --- Завантаження налаштувань ---
void loadSettings() {
    preferences.begin("app_settings", true);
    modeSetting = preferences.getInt("mode", 0);
    buzzerSetting = preferences.getInt("buzzer", 1);
    size_t read_h = preferences.getBytes("sched_h", scheduleHours,
sizeof(scheduleHours));
    size_t read_m = preferences.getBytes("sched_m",
scheduleMinutes, sizeof(scheduleMinutes));
    currentCellNumber = preferences.getInt(KEY_CURRENT_CELL, 0);
    nextDispenseCellNumber = preferences.getInt(KEY_NEXT_DISP_CELL,
1);

    if (nextDispenseCellNumber < 1 || nextDispenseCellNumber >
MED_CELLS_COUNT) {
```

```
        nextDispenseCellNumber = 1;
    }

    preferences.end();

    if (read_h != sizeof(scheduleHours) || read_m !=
sizeof(scheduleMinutes)) {
        for (int i = 0; i < MAX_SCHEDULES; ++i) { scheduleHours[i]
= 0; scheduleMinutes[i] = 0; }
    }
}

// --- Скидання до заводських налаштувань ---
void factoryResetSettings() {
    preferences.begin("app_settings", false);
    preferences.clear();
    preferences.end();

    preferences.begin("wifi_creds", false);
    preferences.remove(KEY_WIFI_SSID);
    preferences.remove(KEY_WIFI_PASS);
    preferences.end();
    currentSsid = "";
    currentPassword = "";

    if (WiFi.isConnected()) {
        WiFi.disconnect(true);
    }
    if (WiFi.getMode() == WIFI_AP || WiFi.getMode() == WIFI_AP_STA)
{
        WiFi.softAPdisconnect(true);
    }
    server.end();

    modeSetting = 0; buzzerSetting = 1;
    for (int i = 0; i < MAX_SCHEDULES; ++i) { scheduleHours[i] = 0;
scheduleMinutes[i] = 0; }
    currentCellNumber = 0;
    nextDispenseCellNumber = 1;

    u8g2.clearBuffer();
    u8g2.setFont(u8g2_font_ncenB08_tr);
    u8g2.drawStr(10, 25, "Factory Reset...");
    u8g2.drawStr(10, 40, "Restarting...");
    u8g2.sendBuffer();

    delay(2500);
    ESP.restart();
}

// === УПРАВЛІННЯ КРОКОВИМ ДВИГУНОМ ТА ДАТЧИКОМ ===
```

```
// --- Виконання одного кроку двигуна ---
void performSingleStep(bool direction) {
    digitalWrite(STEPPER_DIR_PIN, direction);
    digitalWrite(STEPPER_STEP_PIN, HIGH);
    delayMicroseconds(STEP_DELAY_MICROSECONDS / 2);
    digitalWrite(STEPPER_STEP_PIN, LOW);
    delayMicroseconds(STEP_DELAY_MICROSECONDS / 2);
}

// --- Виконання трифазного руху по датчику ---
bool executeThreePhaseSensorMovement(int offsetSteps, bool
updateCellNumbersOnSuccess, bool isHomingSequence) {
    digitalWrite(STEPPER_ENABLE_PIN, LOW);

    bool sensorDeAsserted = false;
    if (digitalRead(OPTICAL_SENSOR_PIN) == LOW) {
        for (long k = 0; k < TOTAL_STEPS_PER_REVOLUTION * 2; k++) {
            if (digitalRead(OPTICAL_SENSOR_PIN) == HIGH) {
                sensorDeAsserted = true;
                break;
            }
            performSingleStep(STEPPER_DIR_FORWARD);
            if (k % 100 == 0) delay(1);
        }
    } else {
        sensorDeAsserted = true;
    }

    if (!sensorDeAsserted) {
        digitalWrite(STEPPER_ENABLE_PIN, HIGH);
        return false;
    }

    bool sensorReAsserted = false;
    for (long k = 0; k < TOTAL_STEPS_PER_REVOLUTION * 2; k++) {
        if (digitalRead(OPTICAL_SENSOR_PIN) == LOW) {
            sensorReAsserted = true;
            break;
        }
        performSingleStep(STEPPER_DIR_FORWARD);
        if (k % 100 == 0) delay(1);
    }

    if (!sensorReAsserted) {
        digitalWrite(STEPPER_ENABLE_PIN, HIGH);
        return false;
    }

    for (int i = 0; i < offsetSteps; i++) {
        performSingleStep(STEPPER_DIR_FORWARD);
    }
```

```
    if (updateCellNumbersOnSuccess) {
        if (isHomingSequence) {
            currentCellNumber = HOME_CELL_NUMBER;
            nextDispenseCellNumber = 1;
        } else {
            if (currentCellNumber == 0 || currentCellNumber ==
HOME_CELL_NUMBER) {
                currentCellNumber = 1;
            }
            else if (currentCellNumber >= 1 && currentCellNumber <
MED_CELLS_COUNT) {
                currentCellNumber++;
            }
            else if (currentCellNumber == MED_CELLS_COUNT) {
                currentCellNumber = 1;
            } else {
                 currentCellNumber = 1;
            }

            if (currentCellNumber == MED_CELLS_COUNT) {
                nextDispenseCellNumber = 1;
            } else if (currentCellNumber >= 1 && currentCellNumber
< MED_CELLS_COUNT) {
                nextDispenseCellNumber = currentCellNumber + 1;
            } else {
                nextDispenseCellNumber = 1;
            }
        }
        saveSettings();
    }
    return true;
}

// --- Переміщення до наступної комірки по датчику ---
bool moveToNextCellBySensor() {
    return
executeThreePhaseSensorMovement(ADDITIONAL_STEPS_AFTER_SENSOR,
true, false);
}

// --- Переміщення до комірки для видачі ---
bool moveToDispenseCell(int targetCell) {
    if (targetCell < 1 || targetCell > MED_CELLS_COUNT) {
        return false;
    }

    if (currentCellNumber == 0 || currentCellNumber ==
HOME_CELL_NUMBER) {
```

```
        if
(!executeThreePhaseSensorMovement(ADDITIONAL_STEPS_AFTER_SENSOR,
true, true)) {
             u8g2.clearBuffer();
u8g2.setFont(u8g2_font_ncenB08_tr);
             u8g2.drawStr(0, 15, "Home Calib Fail");
u8g2.sendBuffer(); delay(2000);
            return false;
        }
    }

    if (currentCellNumber == HOME_CELL_NUMBER && targetCell != 1) {
        if (!moveToNextCellBySensor()) {
             return false;
        }
    }

    int max_moves = MED_CELLS_COUNT + 2;
    int moves_done = 0;

    while (currentCellNumber != targetCell && moves_done <
max_moves) {
        if (!moveToNextCellBySensor()) {
            return false;
        }
        if (currentCellNumber == 0 || currentCellNumber ==
HOME_CELL_NUMBER ) {
            return false;
        }
        moves_done++;
    }

    if (currentCellNumber == targetCell) {
        return true;
    } else {
        return false;
    }
}
```

Повний лістинг коду представлено в репозиторії GitHub: https://github.com/Lyaguhh/Dispenser

ДОДАТОК В

Демонстраційний матеріал

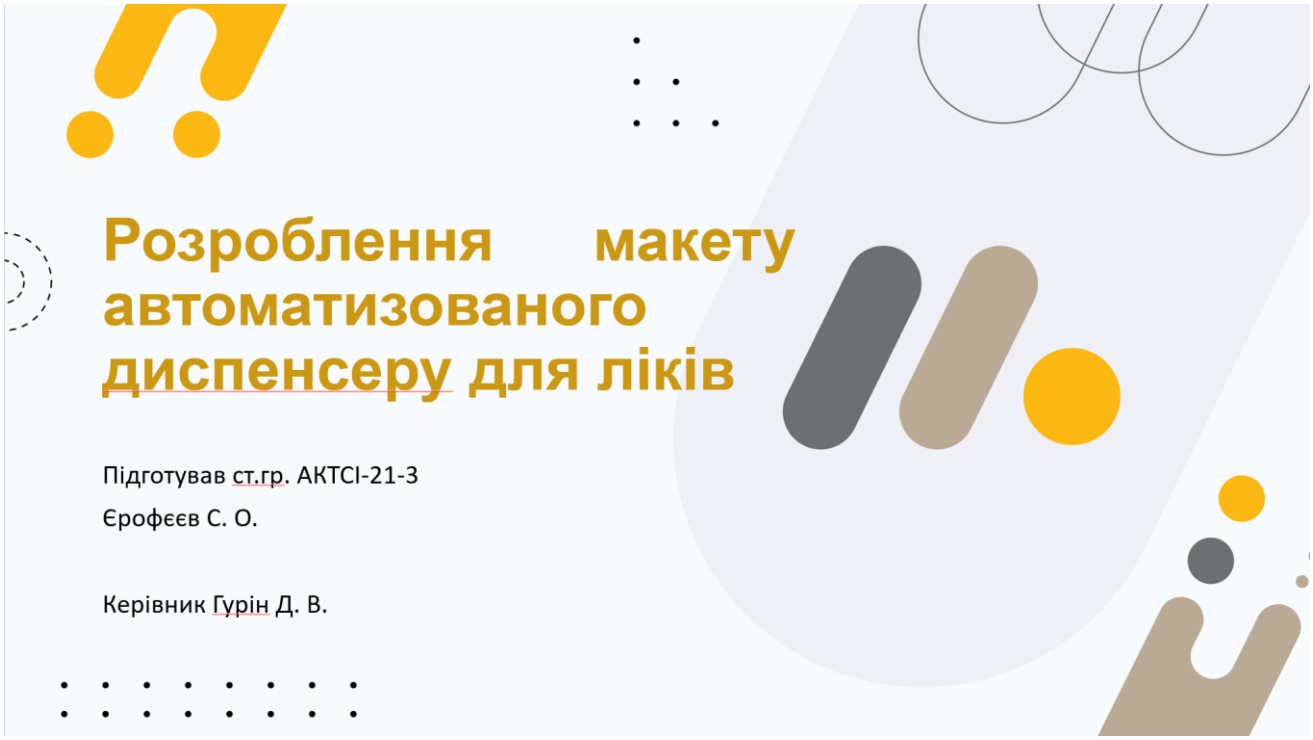

Рисунок В.1 – Титульна сторінка прещентації

Рисунок В.2 –Зміст презентаії, 1 сторінка

Рисунок В.3 –Зміст презентаії, 2 сторінка

Рисунок В.4 –Зміст презентаії, 3 сторінка

Рисунок В.5 –Зміст презентаії, 4 сторінка

Рисунок В.6 –Зміст презентаії, 5 сторінка

Рисунок В.7 –Зміст презентаії, 6 сторінка

Рисунок В.8 –Зміст презентаії, 7 сторінка

Рисунок В.9 –Зміст презентаії, 8 сторінка

Рисунок В.10 –Зміст презентаії, 9 сторінка

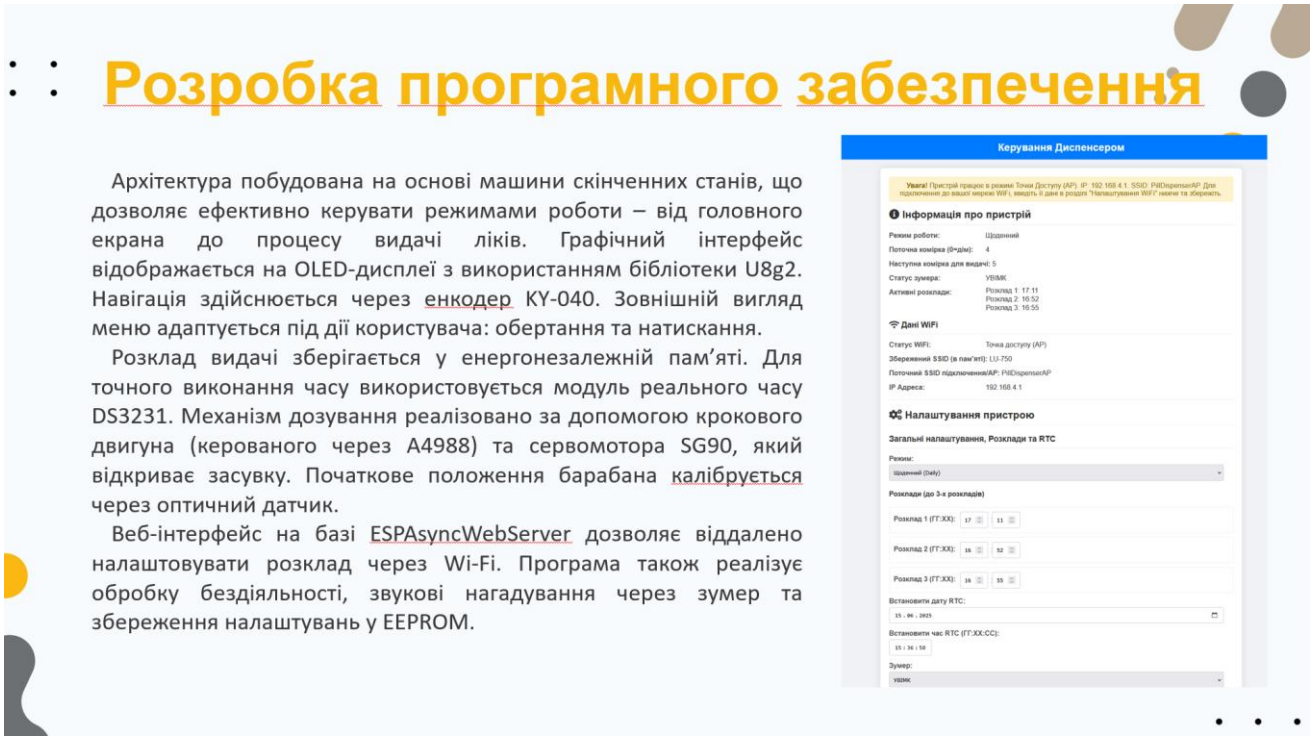

Рисунок В.11 –Зміст презентаії, 10 сторінка

Рисунок В.12 –Зміст презентаії, 11 сторінка

Рисунок В.13 –Зміст презентаії, 12 сторінка

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 418455.003	Пояснювальна записка	А4, 82 с.
		Додаткові матеріали	
2		Апробація результатів	А4, 6 с.
		кваліфікаційної роботи	
3		Частковий лістинг програмного коду	А4, 17 с.
4		Демонстраційний матеріал	А4, 12 с.

					ГЮИК. 418455.003	
Змін.	Арк.	Номер докум.	Підпис	Дата		
Розроб.		Єрофєєв С. О.			Розроблення макету автоматизованого диспенсеру для ліків	Літера: Н; Аркуш: 1; Аркушів: 1
Перевір.		Гурін Д. В.				
						Кафедра КІТАМ ХНУРЕ
Н.контр.		Демська Н. П.				
Затв.		Невлюдов І.Ш.				