МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Розробка програмного забезпечення пристрою

Виконав студент 4 курсу групи К-42
Спеціальність 6.050101 2
комп'ютерні науки,
Ліхачов Кирило Дмитрович
_____

Керівник асистент
Штефан Наталія Зінов'ївна

Консультант к.т.н., доцент
Великодний Станіслав Сергійович

Рецензент к.георг.н, доцент
Бояринцев Євген Львович

Одеса 2019

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Байт – одиниця виміру інформації, дорівнює 8 бітів

БК – блок керування

ГГц − одиниця виміру частоти періодичних процесів, дорівнює 10^9 Гц

Екран − шаблон відтворення надписів і елементів екрану, з якими користувач матиме змогу взаємодіяти для виконання певних операцій

Елемент екрану − елемент з яким, користувач може взаємодіяти

КБ − одиниця виміру інформації, дорівнює 2^{10} байтів

МГц − одиниця виміру частоти періодичних процесів, дорівнює 10^6 Гц

МК − мікроконтролер

ПЗ – програмне забезпечення

Піксель − найменший логічний елемент двовимірного цифрового з о-браження в растровій графіці, або елемент матриці дисплеїв, які формують зображення

Пін − роз’єм на платі Ардуіно

ТЗ – технічне завдання

Скетч − ім'я, яке Arduino використовує для програми

ТЕН − трубчастий електронагрівач

Шилд − підвид плат розширення

Adafruit – компанія, що займається розробкою, дрібносерійної складанням і реалізацією доповнень до популярних налагоджувальний платам: датчики, плати розширення, перехідники та спеціальні конектори

Adafruit_GFX − бібліотека для Arduino, що має в своєму наборі синтаксис і графічні функції для LCD і OLED дисплеїв

ALGOL − назва ряду мов програмування, які застосовуються при складанні програм для вирішення науково-технічних завдань на ЕОМ

Algorithm Builder − графічне середовище сторонньої розробки для створення програмного забезпечення для мікроконтролерів AVR

ANSI Terminal Emulator – емулятор терміналу

Arduino − торгова марка апаратно-програмних засобів для побудови простих систем автоматики і робототехніки, орієнтована на непрофесійних користувачів

Arduino Ethernet − пристрій на основі мікроконтролера ATmega328

Arduino IDE − програмне середовище розробки, призначена для програмування плат Arduino

Arduino Mega 2560 − пристрій на основі мікроконтролера ATmega2560

ARM − Advanced RISC Machine, сімейство ліцензованих 32-бітних і 64-бітових мікропроцесорних ядер розробки компанії ARM Limited

Assembler − мова програмування низького рівня, що представляє собою формат запису машинних команд, зручний для сприйняття людиною

ASCII − American standard code for information interchange, таблиця, в якій деяким поширеним друкованим і недрукованих символів зіставлені числові коди

ATmega2560 – модель мікроконтролера

AVR − Advanced Virtual RISC, сімейство восьми бітних мікроконтролерів, що раніше випускалися фірмою Atmel, потім Microchip

AVR-GCC – найменування набору розповсюджуваних за ліцензією GPL, LGPL пакетів програм, необхідних для компіляції з вихідних текстів виконуваного коду програм для процесорів AVR

BASIC – Beginner's All-purpose Symbolic Instruction Code, сімейство високорівневих мов програмування

BIN – бінарний файл

Bluetooth – виробнича специфікація бездротових персональних мереж.

BMT-1203UX – модель звукового п’єзоелементу

C/C++ – компільований, статично типізований мова програмування загального призначення

CodeVisionAVR – інтегроване середовище розробки програмного забезпечення для мікроконтролерів сімейства AVR фірми Atmel

COM-порт, Serial- порт – «послідовний» порт, сленгова назва інтерфейсу стандарту RS-232

DallasTemperature – бібліотека для праці з датчиками температури Dallas.

Dallas Semiconductor – назва компанії, що розробляє пристрої, які використовують протокол OneWire

Digital Research – компанія, створена американським вченим Гарі Кілдаллом для просування і розробки його операційної системи CP/M і пов'язаних з нею продуктів

DS1302, DS1307, DS3231 – моделі модулів годинників реального часу.

DS18B20 – модель датчика температури

EEPROM – Electrically Erasable Programmable Read-Only Memory

Energia IDE – середовище програмування для МК MSP430

FAT16, FAT32 – файлові системи

Flash – тип довготривалої комп'ютерної пам'яті, вміст якої можна видалити чи перепрограмувати електричним методом

FlowCode – графічне середовище розробки

FraunchPad – лінійка плат керування

Freeware – ПЗ, ліцензійна угода якого не вимагає виплат його правовласнику

HEX – формат файлу, призначеного для подання довільних двійкових даних в текстовому вигляді

HEX-код – шістнадцятковий код

HM-10 – модель модуля Bluetooth

HX8357B, HX8357C, ILI9481 і ILI9486 – контролери TFT дисплеїв.

IAR Systems – інтегроване середовище розробки випущене фірмою IAR Systems

IDE – Integrated Development Environment, інтегроване середовище розробки

Image Craft, ICC – середовище програмування

Intel – виробник електронних пристроїв і комп'ютерних компонентів, включаючи мікропроцесори, набори системної логіки

ISM (Industry, Science and Medicine) – радіо діапазон, виділений для промислових, наукових і медичних радіосистем за винятком телекомунікацій

I2C – послідовна асиметрична шина для зв'язку між інтегральними схемами всередині електронних приладів

Master – ведучий пристрій

MicroLan – інформаційна мережа, що використовує для здійснення цифрового зв'язку одну лінію даних і один поворотний (або земляний) провід.

MISO (Master In Slave Out) − лінія для передачі даних від відомого пристрою (Slave) до ведучого (Master)

MOSI (Master Out Slave In) − лінія для передачі даних від провідного пристрою (Master) до веденим (Slave)

MSP430 – сімейство 16-розрядних мікроконтролерів фірми «Texas Instruments»

OneWire, 1-Wire – двунаправлена шина зв'язку для пристроїв з низько швидкісною передачею даних, в якій дані передаються по ланцюгу живлення

Open-source – програмне забезпечення з відкритим вихідним кодом

PIC – Peripheral Interface Controller, серія мікроконтролерів, що мають гарвардську архітектуру і вироблених американською компанією Microchip Technology Inc

PL/I – Programming Language I, мова програмування, створена для наукових, інженерних і бізнес-орієнтованих обчислень

PL/M – Programming Language for Microcomputers, процедурна мова програмування

Proteus – інтегроване середовище розробки

ROM – файл прошивки або оновлень BIOS або файл образу диска

RTC – Real Time Clock годинник реального часу

SCK – Serial Clock, послідовний тактовий сигнал

SD – формат карт пам’яті

sdfatlib – бібліотека для праці з SD картою

Slave – відомий пристрій

SPI – Serial Peripheral Interface, послідовний синхронний стандарт передачі даних в режимі повного дуплексу, призначений для забезпечення простого і недорогого високошвидкісного сполучення мікроконтролерів і периферії

SSR-25 DA – модель твердотільного реле

SRD-5VDC-SL-C – модель механічного реле

SRAM – статична пам'ять з довільним доступом, напівпровідникова оперативна пам'ять

SS (Slave Select) − вивід, присутній на кожному відомому пристрої

TFT – Thin-Film Transistor, різновид польового транзистора

TFT_HX8357 – бібліотека для праці з TFT дисплеями

UTFT – Бібліотека для праці з TFT дисплеями

Wiring – бібліотека програм, що поставляється разом з Arduino IDE

XPL – мова програмування

ВСТУП

Автоматизація виробництва − це вищий рівень розвитку машинної техніки, коли регулювання й керування виробничими процесами здійснюється без участі людини, а лише під її контролем. Сучасний стан розвитку автоматизації виробництва привів до появи якісно нової системи технологічних машин з керуючими засобами, що ґрунтуються на застосуванні електронних обчислюваних машин, програмованих логічних контролерів, інтелектуальних засобів вимірювання і контролю.

Для побудови пристроїв, які поліпшують керування тих чи інших невеликих технологічних процесів, використовують мікроконтролери, які інтегровані у плати керування. Одними з найпопулярнішими є плати Arduino.

Arduino – це open-source платформа, яка складається з двох основних частин: самої плати і програмного забезпечення або IDE (Integrated Development Environment). Програмне забезпечення запускається на персональному комп'ютері і дозволяє записувати розроблений код на плату[1][1)].

Метою проекту дипломної роботи є розробка ПЗ для версії домашньої пивоварні на базі Arduino, яка автоматизує деякі процеси пивоваріння.

Загальні характеристики кваліфікаційної роботи:

- повний обсяг сторінок пояснювальної записки – 57
- кількість рисунків – 6
- кількість таблиць – 0
- кількість посилань – 16

[1)] [1] Что такое Arduino? URL: http://arduino-diy.com/arduino-chto-eto-takoye. (дата звернення 20.02.2019).

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Характеристика об'єкта розробки

Кінцевий програмний продукт – це ПЗ для версії домашньої пивоварні, що реалізовано на платформі Arduino Mega 2560.

Програмний продукт буде застосовуватися в якості програмного забезпечення автоматизованої домашньої пивоварні, яке відповідатиме вимогам вказаним у ТЗ:

- ПЗ повинно автоматизувати процес виготовлення пива;
- ПЗ повинно мати графічний інтерфейс користувача;
- ПЗ повинно зберігати рецепти користувача на з’ємному носію;
- Закласти у ПЗ можливість керувати домашньою пивоварнею дистанційно.

В кінцевому вигляді програмний продукт матиме вигляд файлу з розширенням .ino, який користувач завантажить у свій пристрій керування домашньою пивоварнею.

1.2 Вхідні дані

Вхідними даними є ТЗ на розробку програмного забезпечення для БК автоматизованої домашньої пивоварні, у склад якої входять наступні елементи:

- плата Arduino Mega 2560 з мікроконтролером ATmega2560;
- датчик температури DS18B20 у волого захисному корпусі 3-Pin TO-92;
- годинник реального часу RTC DS3231;
- твердотільне напівпровідниковий реле SSR-25 DA;
- механічне реле SRD-5VDC-SL-C;
- 3.5" 320x480 TFT LCD кольоровий дисплей для Arduino Mega2560 з вбудованим слотом під SD карту;

- чотири кнопки, об’єднані у клавіатуру;
- звуковий п'єзоелемент BMT-1203UX;
- модуль Bluetooth HM-10.

За ТЗ повинно виконуватися наступні вимоги:

- ПЗ повинно автоматизувати процес виготовлення пива;
- ПЗ повинно мати графічний інтерфейс користувача;
- ПЗ повинно зберігати рецепти користувача на з’ємному носію;
- Закласти у ПЗ можливість керувати домашньою пивоварнею дистанційно.

1.3 Аналіз технологічного процесу виготовлення пива

Виготовлення пива в домашніх умовах складається з наступних кроків:

- підготовка;
- затирання сусла;
- кип’ятіння сусла;
- охолодження;
- бродіння;
- закупорювання і коксування;
- дозрівання.

Проект БК для автоматизованої домашньої пивоварні розраховано на автоматизацію тільки двох процесів, а саме затирання і кип’ятіння сусла. Щоб розуміти які процеси відбуваються, розглянемо ці кроки більш детально.

Процес затирання – це змішування подрібненого солоду з гарячою водою для розщеплення крохмалю в зернах на цукор (мальтозу) і розчинні речовини (декстрини). В цьому процесі поступово виконуються температурні паузи, тобто нагрівання води до завданої рецептом температури і витримка цієї температури певну кількість часу, яку також вказано у рецепті. Перед

доданням солоду і виконанням основних температурних пауз треба нагріти воду ~30°C, цей етап назвемо нульовою паузою.

Процес кип’ятіння – це витримка сусла при температурі кипіння певну кількість часу, зазвичай півтори години, і поетапне додавання різновидів хмелю за рецептом в цей проміжок часу[2][1)].

1.4 Аналіз складових схеми БК

Для поліпшення аналізу складових схеми, старостою комплексної роботи були надані дані й визначення по всім компонентам схеми БК.

Плата Arduino Mega 2560 з мікроконтролером ATmega2560 має велику кількість пінів(54 цифрових і 16 аналогових), до яких можливо під'єднати всі необхідні датчики та модулі, а ATmega2560 має великий, як для мікроконтролера, об’єм Flash-пам'яті, SRAM і EEPROM: 256 КБ, 8 КБ і 4 КБ відповідно.

DS18B20 – цифровий датчик температури призначений для вимірювання температур від -55°C до +125°C. Показання передаються на керуючу плату за протоколом 1-Wire − для підключення знадобиться всього один вільний пін.

RTC DS3231 – це автономна дешева плата, в якій є вбудований кварц з термо-стабілізацією із винятковою точність ходу годинами в режимі реального часу. До складу модуля також входить літій-іонний акумулятор.

SSR-25DA − це однофазне твердотільне реле з низьким рівнем випромінювання електромагнітних завад. Буде використовуватися для керуванням живлення ТЕНу.

SRD-5VDC-SL-C − електромеханічне реле, яке буде використовуватися для керування живленням помпи.

[1)] [2] Как сварить пиво в домашних условиях классический рецепт. URL: https://alcofan.com/kak-svarit-pivo-v-domashnix-usloviyax-klassicheskij-recept.html. (дата звернення 02.03.2019).

Кольоровий дисплей 3,5" TFT − шилд для Arduino Mega 2560 з роздільною здатністю 480x320 пікселів, 262К кольорів. Побудований на контролері ILI9486, підтримує 16-бітний інтерфейс. На платі дисплея є роз'єм підключення SD-Card для зберігання даних.

За допомогою клавіатури, кінцевий користувач буде налаштовувати параметри роботи ПЗ.

BMT-1203UX – звуковий п’єзоелемент, для звукового оповіщення кінцевого користувача о різноманітних подіях.

Bluetooth HM-10 – це модуль бездротового зв'язку, що дозволяє передавати і приймати дані по радіоканалу на дозволеному ISM (Industry, Science and Medicine) діапазоні частот, від 2.4 ГГц до 2.5 ГГц. Працює за протоколом Bluetooth 4.0.

1.5 Аналіз інструментів розробки

Для успішного виконання проекту дипломної роботи необхідно оглянути та проаналізувати мови програмування мікроконтролерів та середовищ розробки.

1.5.1 Аналіз мов програмування мікроконтролерів

Assembler − мова програмування низького рівня, що представляє собою формат запису машинних команд, зручний для сприйняття людиною. Команди мови асемблера один в один відповідають командам процесора і, фактично, є зручну символьну форму запису (мнемо код) команд і їх аргументів. Також мова асемблера забезпечує базові програмні абстракції: зв'язування частин програми і даних через мітки з символьними іменами і директиви. Директиви асемблера дозволяють включати в програму блоки даних (описані явно або лічені з файлу); повторити певний фрагмент вказане число раз; компілю-

вати фрагмент за умовою; задавати адресу виконання фрагмента, змінювати значення міток в процесі компіляції; використовувати макрозначення з параметрами і ін. Кожна модель процесора, в принципі, має свій набір команд і відповідний йому мову (або діалект) асемблера[3][1].

C/C++ на сьогоднішній день є одними з найбільш популярних і поширених мов програмування, який дозволяють створювати додатки для будь-якого спектра завдань: розробка прикладних програм, мобільна розробка і особливо системне програмування. Відмінною особливістю програм на C/C++ є висока швидкість роботи, тому дані мови особливо часто використовуються в тих випадках, де необхідно забезпечити високу продуктивність і швидкодію[4][2].

Lua − швидка і компактна скриптова мова програмування, розроблена підрозділом Tecgraf Католицького університету Ріо-де-Жанейро (Computer Graphics Technology Group of Pontifical Catholic University of Rio de Janeiro in Brazil). Є вільно-поширюваною, з відкритим вихідним кодом на мові Ci[5][3].

BASIC (скорочення від англ. Beginner's All-purpose Symbolic Instruction Code) − універсальний код символічних інструкцій для початківців; (англ. basic − основний, базовий) − сімейство високорівневих мов програмування[6][4].

PL/M − процедурна мова програмування, розроблена в 1972 фірмою Digital Research для мікропроцесорів Intel. Мова запозичила ідеї з PL/I, ALGOL, XPL і мала інтегрований макропроцесор. Компілятори PL/M існува-

---

[1] [3] Assembler – Энциклопедия языков программирования. URL: http://progopedia.ru/language/assembler/. (дата звернення 02.03.2019).

[2] [4] Языки программирования C и C++. URL: https://metanit.com/cpp/. (дата звернення 02.03.2019).

[3] [5] Lua – Энциклопедия языков программирования. URL: http://progopedia.ru/language/lua/. (дата звернення 02.03.2019).

[4] [6] Basic – Энциклопедия языков программирования. URL: http://progopedia.ru/language/basic. (дата звернення 02.03.2019).

ли для ранніх моделей процесорів Intel: 8008, 8080, 8051, 8086, 286, 386 і Intel 80486[7][1)].

1.5.2 Аналіз середовищ розробки ПЗ

Arduino IDE − це безкоштовна середовище розробки для платформи Arduino, яка містить редактор коду, компілятор і модуль передачі прошивки в плату. Це середовище прекрасно підійде для програмістів, які вважають за краще мови програмування C/C++. Програми (скетчі), написані за допомогою Arduino IDE, обробляються препроцесором, а потім компілюються в AVR-GCC. Середовище розробки Arduino поставляється разом з бібліотекою програм, яка називається «Wiring», що бере початок від проекту Wiring, який дозволяє робити багато стандартних операцій введення/виводу набагато простіше[8][2)].

Energia − середовище програмування для МК MSP430, яка найбільш популярною середовищем програмування серед початківців. Має Сі-подібна мова програмування, але він відрізняється від мов, які використовуються в перерахованих вище середовищах. Мова Energia (і Arduino IDE) більш зрозумілий, подібний англійським словам. Energia підтримує додаткові бібліотеки, до складу яких входять драйвера для підключення платформи LaunchPad MSP430 на базі ARM Cortex, FraunchPad і lm4f120 StallerPad. Дане середовище програмування є модифікованою версією середовища Arduino IDE. Працює с 1, 16МГц МК MSP430 і 80 МГц lm4f120. Впроваджено функція перегляду COM-порту.

FlowCode − графічна універсальне середовище програмування МК. Програмування здійснюється завдяки побудові логічної структури, тобто

---

1) [7] PL/M – Энциклопедия языков программирования. URL: http://progopedia.ru/language/plm/. (дата звернення 05.03.2019).

2) [8] Arduino IDE – програми для Windows. URL: https://programy.com.ua/ua/arduino_ide/. (дата звернення 05.03.2019).

блок-схем, аналогічно середовищі HiAsm. Функція експорту дозволяє експортувати написаний код PIC МК в програму AVR МК і навпаки. Доповненням даного середовища програмування є створення HEX-коду, який може бути використовувати при прошивки МК, або при проектуванні схеми з підтримкою МК, наприклад, в середовищі Proteus[9][1)].

Algorithm Builder − безкоштовне середовище (умови розповсюдження Freeware) для МК AVR, яка забезпечує повний цикл розробки вбудованого ПЗ, в т.ч. такі етапи як введення алгоритму, налагодження та внутрішньо схемне програмування. Розробку програми можна вести як на рівні асемблера, так і на макрорівні, при якому можлива робота зі знакозмінними величинами довільної довжини. Це наближає можливості програмування до мови високого рівня[10][2)].

CodeVisionAVR − популярна умовно-безкоштовне середовище програмування AVR МК. Об'єднує в собі Сі-подібна мова програмування і асемблер. Функції програми дозволяють самостійно прошивати МК і встановлювати fuse-бити і ПЗУ. Кінцевим результатом розробки програми під. МК є створення HEX, BIN або ROM-файлу, для прошивки МК за допомогою програматора.

Середовище IAR Systems підтримує програмування МК AVR і MSP430, але функції програмування двох МК не є об'єднані в одному середовищі. Для кожного МК були розроблені окремі середовища програмування. Аналогічним чином була розроблена середовище програмування Image Craft (ICC). ICC підтримує Сі-подібний синтаксис і асемблер. IAR Systems і Image Craft в їх склад входить цілеспрямовані бібліотеки по роботі з окремими частинами МК. До складу ICC додана утиліта для генерації коду і ініціалізації периферії

[1)] [9] Анализ сред программирования для МК. URL: http://entropiya-blog.ru/analiz-sred-programmirovaniya-dlya-mk.html. (дата звернення 05.03.2019).

[2)] [10] Algorithm Builder. Графическая среда разработки программного обеспечения для микроконтроллеров AVR. URL: http://www.gaw.ru/html.cgi/txt/soft/avr/Algorithm_Builder.htm. (дата звернення 05.03.2019).

МК, впроваджений ANSI Terminal Emulator, який надає можливість працювати з COM-портом[9][1)].

1.6 Результати аналізу

За час проведення аналізу було проаналізовано:

- технологічний процес виготовлення пива;
- складові схеми БК;
- мови програмування мікроконтролерів;
- середовища розробки ПЗ.

За результатами проведення аналізу було прийнято рішення. Розробляти програмного забезпечення, для плати Arduino Mega2560 з мікроконтролером ATmega2560, використовувати середовище розробки Arduino IDE та мову C/C++ для написання основного скетчу програми.

[1)] [9] Анализ сред программирования для МК. URL: http://entropiya-blog.ru/analiz-sred-programmirovaniya-dlya-mk.html. (дата звернення 05.03.2019).

2 РОЗРОБКА АЛГОРИТМІВ РОБОТИ ПЗ

2.1 Постановка завдань

За результатами огляду і аналізу технологічних процесів виготовлення пива та вимог ТЗ до ПЗ, що розробляється, можна виділити ряд задач, які треба виконати для успішної розробки ПЗ, а саме:

- спроектувати і розробити алгоритм виконання процесу затирання;
- спроектувати і розробити алгоритм виконання процесу кип’ятіння;
- спроектувати і розробити алгоритму налаштування рецепту пива, тобто меню налаштувань;
- спроектувати і розробити алгоритм збереження рецептів пива;
- спроектувати і алгоритм дистанційного керування.

Після виконання вище перелічених задача, отримуємо окремі компоненти програми, які необхідно буде об’єднати у єдиний алгоритм виконання ПЗ. Також до задач, які були поставлені треба додати задачу вирішення проблеми появи випадків, коли датчик температури або модуль годинник реального часу вийдуть з ладу, або не будуть під’єднані до плати БК.

2.2 Розробка алгоритму виконання процесу затирання

У пункті 1.2 було детально розглянуто процес затирання солоду. Розділимо виконання роботи алгоритму на дві частини. Перша частина – це частина до додавання солоду, друга – виконання температурних пауз.

В першій частині перед набором температури для додавання солоду буде прокачуватися помпа, після чого ввімкнемо помпу, щоб користувач власноруч не перемішував сусло, і ТЕН, який нагріє воду до завданої температури.

Як тільки процес нагрівання дійде до завданої точки, то алгоритм призупинить своє виконання, що в свою чергу дасть змогу користувачу додати солод у ємність, і буде очікувати натиснення кнопки, щоб перейти до вико-

нання температурних пауз. Треба звернути увагу на те, що під час очікування натиску кнопки, температуру води у ємність треба підтримувати на вказаній відмітці.

Для зміни поточної температури, треба на пін, до якого під'єднане реле, керуюче роботою ТЕНу, подати живлення, для ввімкнення ТЕНу, і зняти живлення, щоб вимкнути ТЕН. Таким чином, якщо поточна температура менше за вказану, то ТЕН треба ввімкнути, та навпаки, якщо поточна температура більше, тоді вмикаємо ТЕН.

Послідовність кроків у першій частині набуде наступного вигляду:

а) прокачати помпу;

б) ввімкнути помпу у робочий режим;

в) дізнатися поточну температуру;

г) якщо поточна температура менше за вказану, то ввімкнути ТЕН, інакше вимкнути;

д) якщо поточна температура дорівнює вказаній, то призупиняємо виконання алгоритму і очікуємо підтвердження користувача про перехід до другої частини. Як підтвердження надійде, то переходимо до другої частини, інакше повторюємо кроки в-д.

Розглянемо, що відбувається у другій частин. В цій частині, як було зазначено, виконуються температурні паузи у кількості п'яти штук. Температурні паузи діють за одним й тим самим алгоритмом: нагрів води до вказаної температури, потім витримка цієї температури вказану кількість часу.

Тому доцільно виділити цей алгоритм у іншу сутність і викликати її виконання певну кількість разів. Цій сутності надамо назву – пауза затирання. Протягом перебігу виконання процесу затирання будемо надавати користувачу інформацію про поточний етап у графічному і звуковому вигляді. Тоді алгоритм виконання процесу затирання набуде наступного вигляду(див. рис. 1).

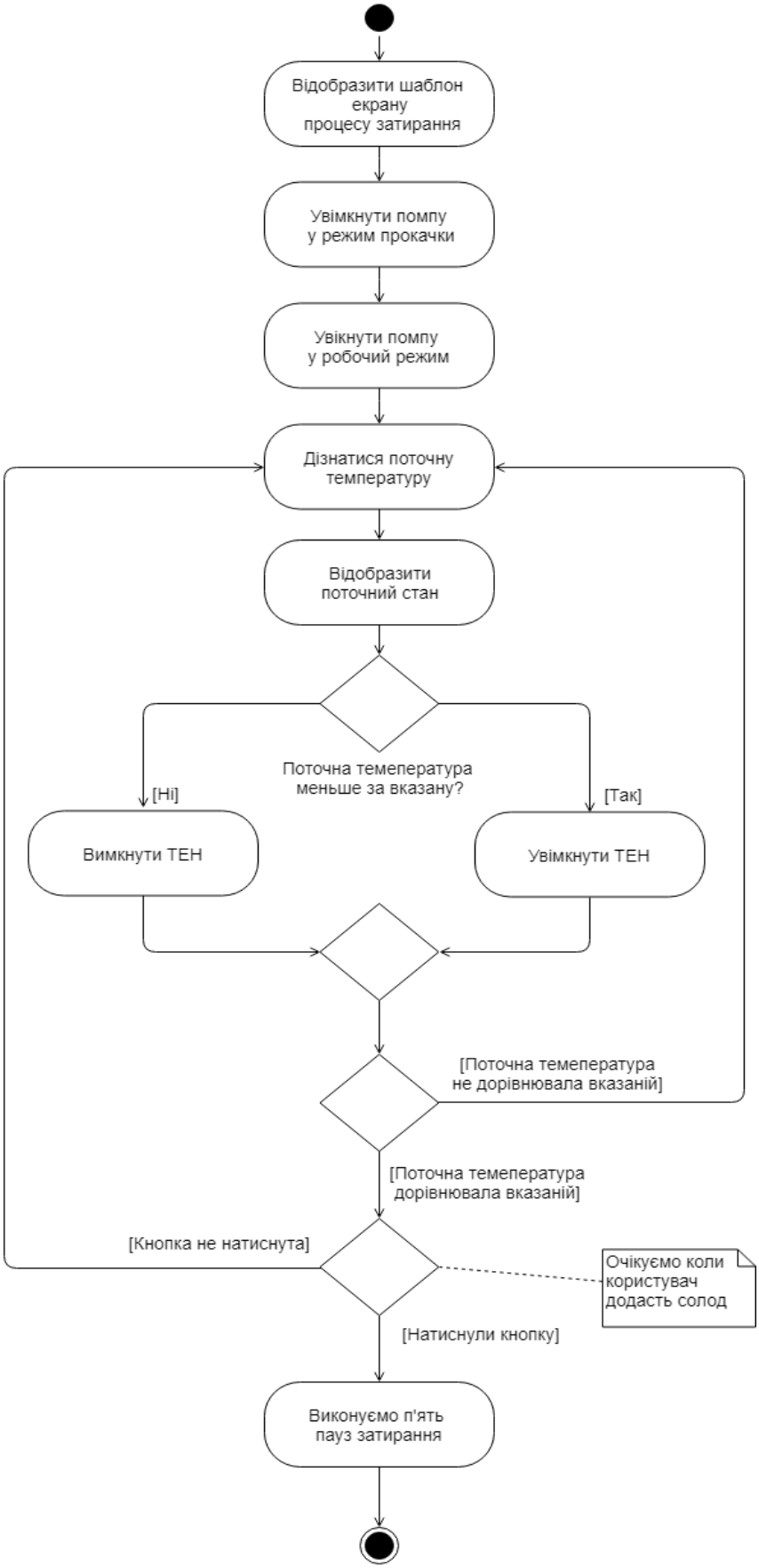

Рисунок 1 – Діаграма діяльності виконання процесу затирання

2.3 Розробка алгоритму виконання паузи затирання

Вхідними параметрами у паузі затирання є температура і час температурної паузи. Виконання алгоритму паузи затирання стане можливим, якщо жоден з вхідних параметрів не дорівнює нулю. Проте якщо все ж таки один з параметрів дорівнює нулю, то пауза затирання завершить своє виконання.

Розглянемо кроки виконання алгоритму, коли жоден з параметрів не дорівнює нулю. Спочатку отримуємо поточну температуру і перевіряємо, якщо поточна температура менше за передану, то вмикаємо ТЕН, інакше ТЕН вимкнемо. У випадку коли поточна температура(в перший раз) дорівнює переданій температурі, то запускається таймер, щоб розпочати відлік часу температурної паузи. Протягом відліку часу поточна температура буде підтримуватися на вказаній відмітці. Коли час сплине, то виконання алгоритму завершиться.

Алгоритм паузи затирання має наступний вигляд(див. рис. 2).

2.4 Розробка алгоритму виконання процесу кип’ятіння

Для того щоб запустити відлік часу, спочатку треба довести поточну температуру до температури кип’ятіння. І коли поточна температура в перший раз дорівнюватиме вказаній, то запускаємо таймер.

Після того як таймер буде запущено, потрібно буде перевіряти наявність збігу значення поточного часу і часу додавання хмелю, і коли матиметься збіг, то ввімкнемо звукове оповіщення користувачу про необхідність додати у ємність хміль. По закінченню відліку часу алгоритм завершить своє виконання.

Тоді алгоритм процесу кип’ятіння набуде вигляду(див. рис. 3).

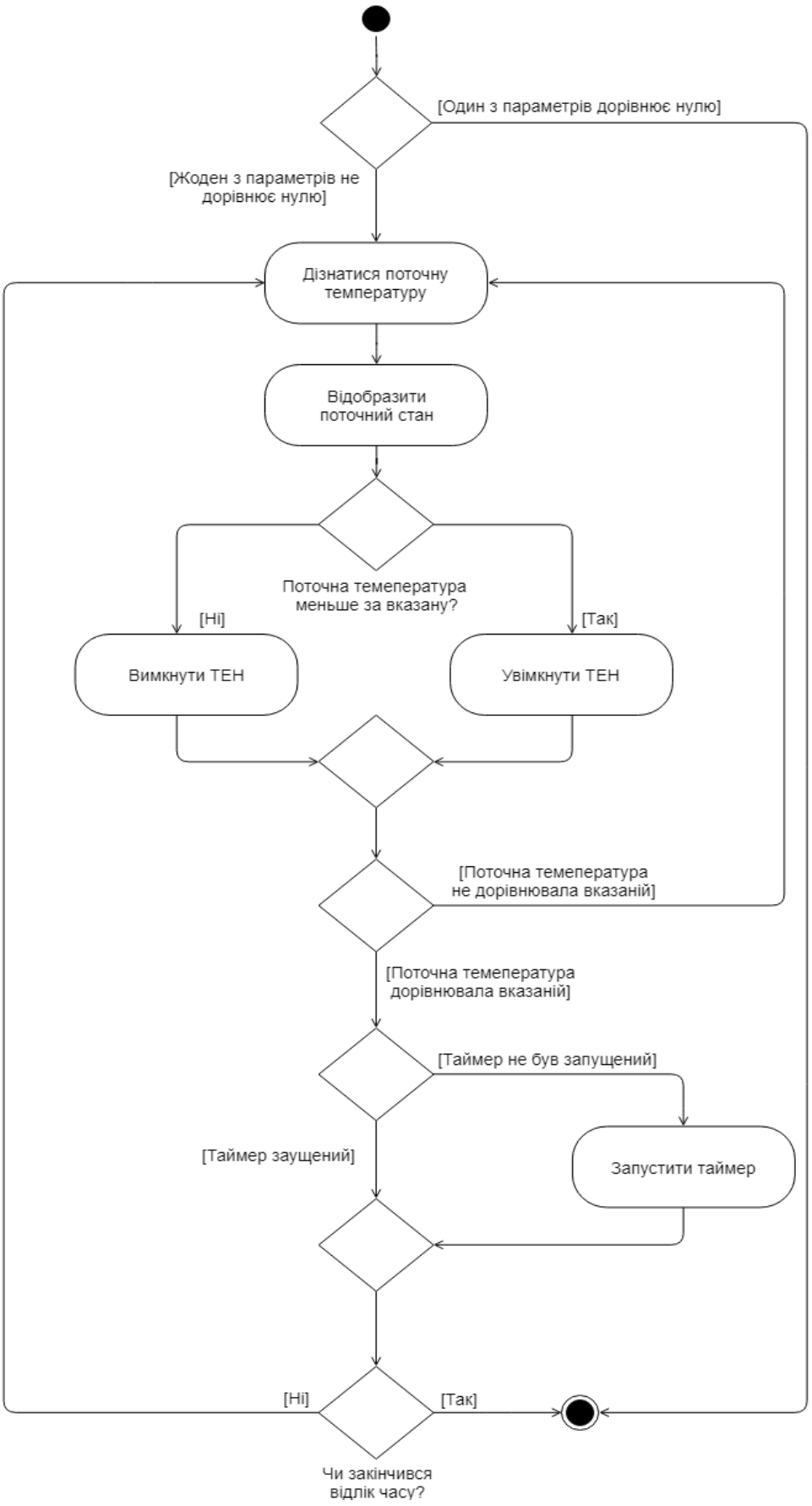

Рисунок 2 – Діаграма діяльності виконання паузи затирання

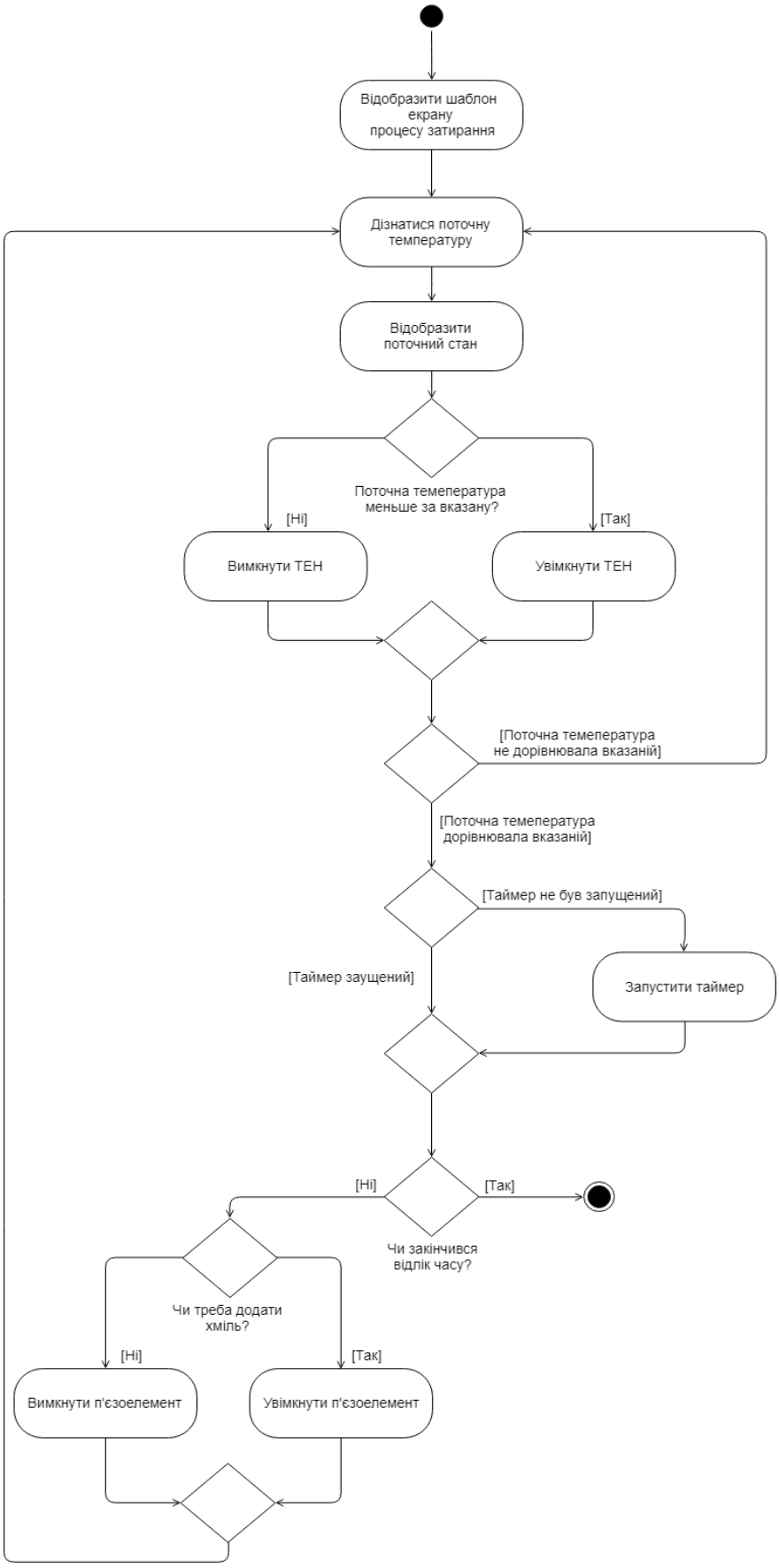

Рисунок 3 – Діаграма діяльності виконання процесу кип’ятіння

2.5 Розробка меню налаштувань

Для успішного виконання алгоритмів, наведених у попередніх пунктах, треба надати користувачу змогу налаштовувати рецепт виготовлення пива. Для налаштування рецепта у користувача є клавіатура з чотирма кнопками, як пристрій вводу інформації, і кольоровий дисплей, як пристрій виводу введеної інформації.

Перед описанням меню налаштувань дамо декілька визначень тим термінам, які будемо користуватися:

Екран – шаблон відтворення надписів і елементів екрану, з якими користувач матиме змогу взаємодіяти для виконання певних операцій, наприклад налаштування поточної дати. Не слід плутати з дисплеєм.

Елемент екрану – елемент з яким, користувач може взаємодіяти.

Меню налаштувань складатиметься з трьох гілок:

- швидкий рецепт(Quick recipe) – в цій гілці налаштовуються параметри рецепта, який можна потім запустити на виконання;
- мої рецепти(My recipes) – в цій гілці можна переглянути список рецептів, які збережено на SD картці, і сам рецепт. Збережений рецепт користувач може модифікувати, запустити на виконання або видалити;
- налаштування(Settings) – в цій гілці користувач зможе налаштувати ті параметри які не підпадають у групу параметрів рецепта, у яку входитимуть: температура додання солоду, п’ять температур і п’ять часів температурних пауз, температура і час кип’ятіння, п’ять часів додання хмелю.

Для відображення етапів, які передують рецепт на виконання, будемо використовувати шість екранів:

- головний екран, на якому міститься головне меню з трьома елементами екрану для переходу на відповідну гілку;

- екран відображення налаштування швидкого рецепта, на якому міститься елементи екрану переходу і елементи екрану налаштування параметрів рецепта;
- екран зі списком збережених рецептів, на якому міститься список з елементів екрану, які є назвами збережених рецептів;
- екран відображення збереженого рецепта, на якому міститься елементи екрану переходу, елементи екрану модифікації рецепта і елементи екрану налаштування параметрів рецепта;
- екран перевірки рецепта перед виконанням, містить таблицю з параметрами рецепта, для перевірки, і елементи екрану підтвердження параметрів рецепта та повернення до попереднього екрану;
- екран налаштувань, містить елементи екрану для налаштування кількості разів прокачки помпи та поточної дати і часу.

В гілці Швидкий рецепт(Quick recipe) будуть викликатися екран відображення налаштування швидкого рецепта і екран перевірки рецепта перед виконанням. В гілці Мої рецепти(My recipes), викликатимуться наступні екрани: екран зі списком збережених рецептів, екран відображення збереженого рецепта і екран перевірки рецепта перед виконанням. Гілка Налаштування(Settings) викликатиме лише екран налаштувань(див. рис. 4).

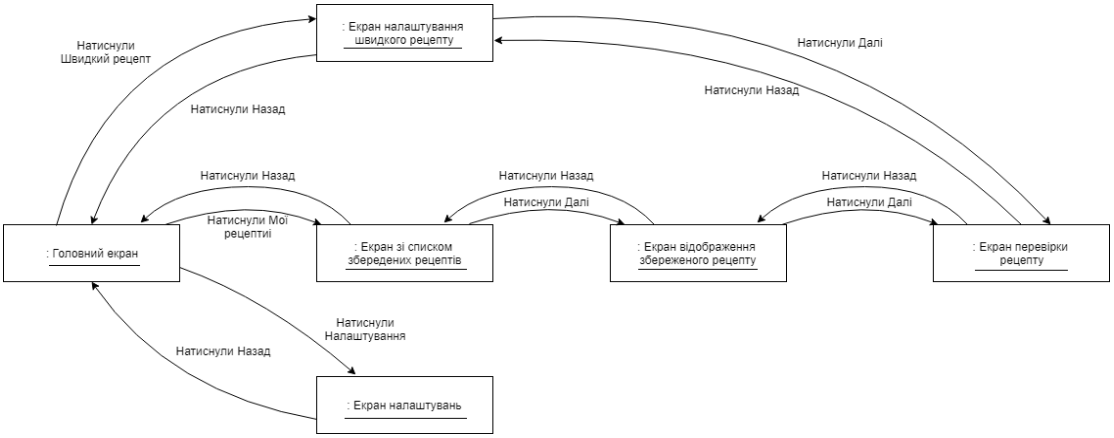

Рисунок 4 – Діаграма потоків екранів меню налаштувань

2.6 Розробка алгоритму збереження рецептів пива

Для збереження рецептів у схемі проекту передбачено слот під SD картку, на якій і будуть зберігатися рецепти. Рецепт складатиметься із чіткої послідовності всіх параметрів рецепта і роздільників. Послідовність параметрів наступна: температура додання солоду, температура першої паузи, температура другої паузи, температура третьої паузи, температура четвертої паузи, температура п’ятої паузи, час першої паузи, час другої паузи, час третьої паузи, час четвертої паузи, час п’ятої паузи, температура кип’ятіння, час кип’ятіння. В якості роздільника у рядку буде використовуватися кома.

Алгоритм збереження рецепта буде складатися з наступних кроків:

- формування рядку рецепта;
- створення або відкриття файлу з іменем, який вказав користувач;
- видалення попереднього рядку рецепта;
- запис рядку рецепта у файл;
- закриття з’єднання файл.

2.7 Розробка алгоритму дистанційного керування

Для дистанційного керування БК, користувачеві знадобиться встановити на свій мобільний пристрій додаток, який по зв’язку Bluetooth передає у БК спеціальні команди(посилки), які потім виконуються. Для прийняття команд(посилок) у БК мається модуль зв’язку Bluetooth, який передає прийняті команди(посилки) у COM порт плати Arduino. Звідти функція слухач приймає і розпізнає команду(посилку), після чого викликає відповідну до команди(посилки) функцію її виконання.

Формування вхідних, відносно БК, команд(посилок) здійснюється додатком, а вихідних − самим БК. Команда(посилка) має наступну послідовність елементів:

- ідентифікатор початку передачі команди;

- назва команди;
- ідентифікатор початку даних;
- дані.

Серед команд будуть, які передаватимуться з віддаленого пристрою, будуть наступні: виконати швидкий рецепт, виконати збережений рецепт, зберегти рецепт, видалити рецепт.

2.8 Розробка загального алгоритму

Після розробки всіх компонентів скетчу програми, тепер треба об’єднати їх у єдиний алгоритм. Загальний алгоритм складатиметься з трьох кроків:

а) перевірка несправностей зовнішніх модулів. Для того щоб виконувати рецепти користувача, треба ПЗ повинно переконатися, що зовнішні модулі справні. Критичними для правильної роботи ПЗ є модуль годинника реального часу і датчик температури. При несправності або відсутності годинника часу повертає наступну дату «01-01-2001, 01:01:01, Sun», а несправний датчик температури, повертає значення «-127». Тому якщо при перевірці значень, що повертаються модулями, будуть зафіксовані одне чи інше значення, то ПЗ сповістить користувача, що є несправність і заблокує елементи екрану, які відповідають за запуск рецепта на виконання. У разі коли несправність зафіксовано при виконанні рецепта, то ПЗ призупинить своє виконання і сповістить про це користувача;

б) перевірка отримання команди(посилки) з віддаленого пристрою;

в) якщо рецепт налаштовано, то виконується рецепт, інакше налаштовуємо рецепт.

Тоді загальний алгоритм набуде наступного вигляду(див. рис. 5).

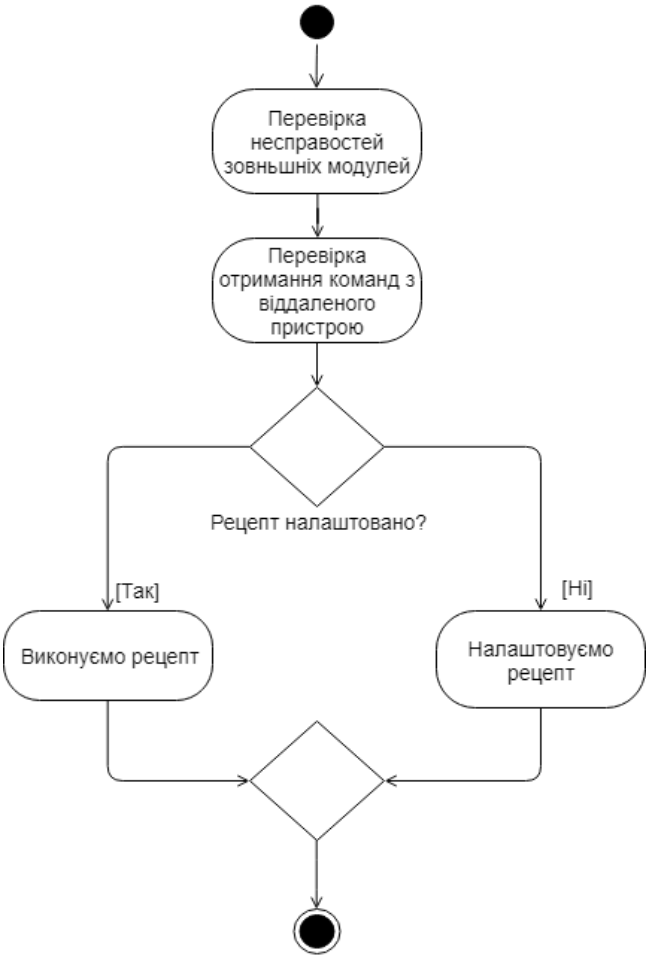

Рисунок 5 – Діаграма діяльності загального алгоритму програми

3 ПРОЕКТНІ ТА ТЕХНІЧНІ РІШЕННЯ

3.1 Опис бібліотек, що використовуються

Для більш швидкої розробки ПЗ доцільно буде скористатися готовими бібліотеками під зовнішні модулі. Бібліотеки містять всі необхідні інструменти, для поліпшення ходу розробки. Деякі необхідні бібліотеки, вже містяться у середовищі розробки, а деякі потрібно шукати на просторах всесвітньої павутини, наприклад для роботи з карткою пам’яті у середовищі розробки Arduino IDE мається бібліотека SD. Загалом знадобиться знайти бібліотеки для роботи з дисплеєм, датчиком температури і модулем годинника реального часу.

3.1.1 Бібліотека для праці з модулем RTC

Для праці з модулем RTC буде використовуватися бібліотека iarduino_RTC.

Бібліотека дозволяє читати і записувати час RTC модулів на базі чіпів: DS1302, DS1307, DS3231.

Перевагою даної бібліотеки є зручна реалізація отримання часу.

Дана бібліотека може використовувати як апаратну, так і програмну реалізацію шини I2C[11][1)].

3.1.2 Бібліотеки для праці з датчиком температури

Для знімання показників температури будуть використовуватися бібліотеки OneWire і DallasTemperature.

Термодачик використовує протокол 1-Wire.

[1)] [11] Универсальная библиотека iarduino_RTC для RTC DS1302, DS1307, DS3231 к Arduino. URL: https://iarduino.ru/file/235.html. (дата звернення 09.04.2019).

Протокол 1-Wire − це протокол, який використовується для управління пристроями, які виробляються компанією Dallas Semiconductor (нині Maxim). Хоча 1-Wire є пропрієтарним протоколом і торговою маркою Dallas, програмісти, які використовують драйвери 1-Wire, не зобов'язані робити за це ніяких виплат.

Мережа 1-Wire, яку Dallas називає MicroLan(торгова марка), складається з одного ведучого пристрою, до якого за допомогою єдиної лінії передачі даних з'єднання з одним або декілька ведених пристроїв. Цю лінію, крім іншого, можна використовувати для електроживлення ведених пристроїв (ситуацію, коли пристрої підживлюються через шину 1-Wire, називають «паразитним живленням»).

Серед інших пристроїв, що працюють через протокол 1-Wire, особливо популярні температурні датчики − вони недорогі, прості у використанні і дозволяють безпосередньо зчитувати відкалібровані цифрові температурні дані. Крім того, вони терпимі до довгих проводах, якими часто доводиться користуватися при створенні ланцюга з Arduino.

DallasTemperature – бібліотека для праці з датчиками температури Dallas[12][1)].

3.1.3 Бібліотека для праці з кольоровим дисплеєм

Для праці з кольоровим дисплеєм буде використовуватися бібліотека TFT_HX8357. Це окрема бібліотека, яка містить графічні функції плюс драйвери TFT HX8357B, HX8357C, ILI9481 і ILI9486. Бібліотека була отримана з Adafruit_GFX і бібліотеки драйверів з додаткові фрагменти коду інших авторів. Додані нові функції, зокрема, вони містять пропорційні шрифти крім оригінального шрифту Adafruit. Шрифти більшого розміру кодуються для скорочення довжини їх флеш-сліду і швидкості рендерингу.

[1)] [12] Arduino: Библиотеки/OneWire – Онлайн справочник. URL: http://wikihandbk.com/wiki/Arduino:Библиотеки/OneWire. (дата звернення 09.04.2019).

За відгуками працює швидше бібліотеки UTFT.

3.1.4 Бібліотеки для праці з пам’яттю пристрою

Для зберігання рецептів на з’ємний носій будуть використовуватися бібліотеки SD і SPI.

Бібліотека SD дозволяє зчитувати і записувати інформацію на SD-карту пам'яті (наприклад, на платі розширення Arduino Ethernet). Вона заснована на бібліотеці sdfatlib (автор William Greiman). Бібліотека підтримує роботу зі стандартними картами пам'яті типу SD і SDHC, відформатовані в файлову систему FAT16 або FAT32. При роботі з картою пам'яті необхідно використовувати короткі імена файлів у форматі 8.3 (8 символів − ім'я файлу, 3 символу − розширення). Функції бібліотеки SD в якості параметра можуть приймати не тільки ім'я файлу, але і шлях до нього. При цьому в якості роздільника між каталогами використовується прямий слеш (наприклад, "directory/filename.txt"). Додавання косою риси перед ім'ям файлу необов'язково, оскільки робочої Директорією завжди є кореневий каталог карти пам'яті (таким чином, ім'я "/file.txt" еквівалентно "file.txt"). Починаючи з версії 1.0, в бібліотеці реалізована можливість одночасного відкриття декількох файлів.

Взаємодія між мікроконтролером і SD-картою пам'яті здійснюється по шині SPI, що об'єднує в собі виводи 11, 12 і 13 (на більшості плат Ардуіно), або 50, 51 і 52 (на Arduino Mega). Крім перерахованих, ще один вивід повинен використовуватися для активізації SD-карти. Для цього може використовуватися як апаратний вивід SS − виводу 10 (на більшості плат Ардуіно) або виводу 53 (на Arduino Mega), так і будь-який інший вивід, зазначений при виклику методу SD.begin(). Зверніть увагу, що для коректної роботи бібліо-

теки SD, вивід SS повинен бути налаштований як вихід, навіть в тих випадках, коли він не використовується[13][1)].

Бібліотека SPI дозволяє Ардуіно взаємодіяти з різними SPI-пристроями, виступаючи при цьому в ролі ведучого пристрою.

Послідовний периферійний інтерфейс (SPI) – це синхронний протокол послідовної передачі даних, який використовується для зв'язку мікроконтролера з одним або декількома периферійними пристроями. Інтерфейс SPI відрізняється відносно високою швидкістю і призначений для зв'язку близько розташованих пристроїв. Він також може використовуватися для взаємодії двох мікроконтролерів.

Згідно з протоколом SPI, одне з взаємодіючих пристроїв (зазвичай мікроконтролер) завжди є провідним і контролює ведені периферійні пристрої. Як правило, всі взаємодіючі пристрої об'єднані трьома загальними лініями:

- MISO (Master In Slave Out) − лінія для передачі даних від відомого пристрою (Slave) до ведучого (Master);
- MOSI (Master Out Slave In) − лінія для передачі даних від провідного пристрою (Master) до веденим (Slave);
- SCK (Serial Clock) − тактові імпульси, що генеруються провідним пристроєм (Master) для синхронізації процесу передачі даних.

Крім перерахованих, на кожен пристрій відводиться окрема лінія:

- SS (Slave Select) − вивід, присутній на кожному відомому пристрої. Він призначений для активізації Майстром того чи іншого периферійного пристрою.

Периферійний пристрій (Slave) взаємодіє з провідним (Master) тоді, коли на виведення SS присутній низький рівень сигналу. В іншому випадку дані від Master-пристрої будуть ігноруватися. Така архітектура дозволяє взає-

[1)] [13] SD Программирование Ардуино. URL: https://doc.arduino.ua/ru/prog/SD. (дата звернення 09.04.2019).

модіяти з декількома SPI-пристроями, підключеними до однієї і тієї ж шині: MISO, MOSI і SCK[14][1)].

3.2 Складові ПЗ

Для більш зручної розробки, константи, змінні і функції, які будуть створені, містяться на декількох файлах .ino. Це потрібно, щоб розбити функції по категоріям їх відповідальності у роботі ПЗ(див. рис. 6).

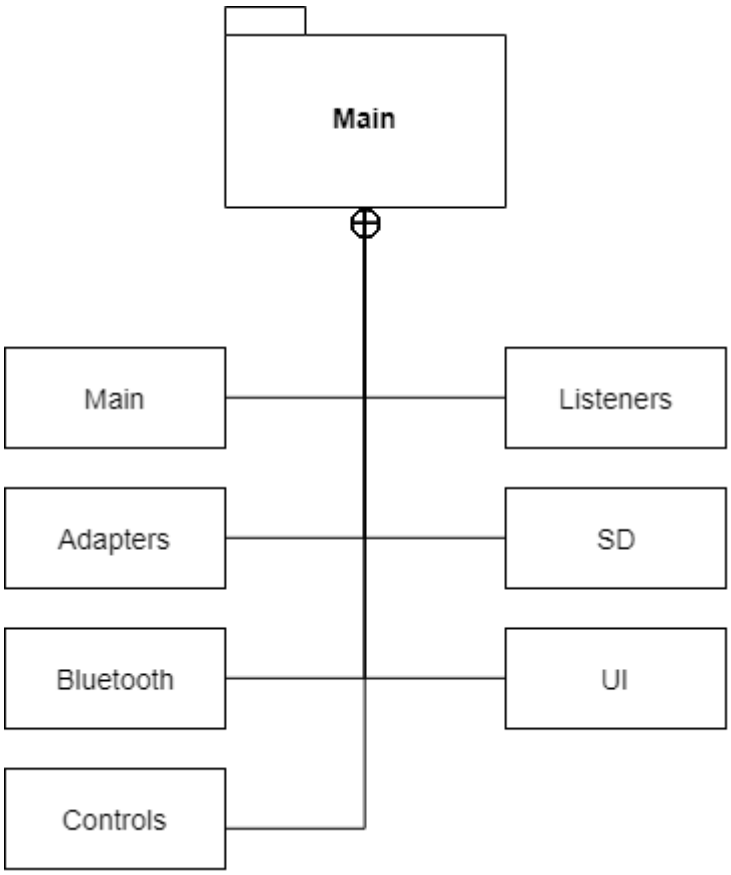

Рисунок 6 – Діаграма пакетів ПЗ

Головний скетч ПЗ матиме ім’я Main.ino. В цьому файлі будуть зберігатися посилання на бібліотеки, які будуть використовуватися, константи,

[1)] [14] SPI Программирование Ардуино. URL: https://doc.arduino.ua/ru/prog/SPI. (дата звернення 09.04.2019).

змінні та дві функції void setup() і void loop(), які повинні бути у кожній програмі створеній для Arduino. Під цей файл повинна виділятися директорія з тим же ім’ям.

Всі інші функції, як було сказано, будуть розподілені по файлах Adapters.ino, Bluetooth.ino, Controls.ino, Listeners.ino, SD.ino і UI.ino.

Розглянемо для яких функцій потрібен файл UI.ino. У цьому файлі знаходитимуться всі функції, які відповідатимуть за відображення екранів, елементів екрану тощо.

Файл SD.ino матиме функції призначені для роботи з карткою пам’яті, а саме функції для збереження і видалення рецептів користувача, функцію зміни імені файлів рецепта і перевірки наявності картки пам’яті, а також функцію оновлення внутрішнього списку збережених рецептів, який буде використовуватися програмою.

Controls.ino – це файл, який зберігатиме функції для роботи із зовнішніми модулями такими, як годинник реального часу, реле помпи, звуковий п’єзоелемент. Зокрема у цьому файлі будуть міститися функції, які реалізують алгоритми процесу затирання, паузи затирання і процесу кип’ятіння.

Файл Bluetooth.ino призначений для розміщення у ньому функцій, які будуть відстежувати прибуття команд(посилок) з віддаленого пристрою, розпаковки команди(посилки) і її виконання.

Файли Listeners.ino і Adapters.ino, будуть зберігати функції слухачі, елементів екранів, і функції обробники події натиску кнопок клавіатури, відповідно.

3.3 Змінні і константи

У скетчі програми маються константи для вказівки пінів, екранів, основного шрифту і кольорової теми:

- ONE_WIRE_BUS_PIN – пін підключення шини OneWire датчика температури;
- BUZZER_PW_PIN – пін підключення живлення звукового п’єзоелементу;
- PUMP_PIN – пін зміни стану реле помпи;
- HEAT_PIN – пін зміни стану реле ТЕНу;
- BUTTON_UP_PIN – пін під’єднання кнопки «Вгору»;
- BUTTON_DOWN_PIN – пін під’єднання кнопки «Вниз»;
- BUTTON_PLUS_PIN – пін під’єднання кнопки «Плюс»;
- BUTTON_MINUS_PIN – пін під’єднання кнопки «Мінус»;
- MAIN_SCR – головний екран;
- SETTINGS_SCR – екран налаштувань ПЗ;
- CHECK_RECIPE_SCR – екран перевірки рецепта перед виконанням;
- RECIPES_LIST_SCR – екран перегляду списку збережених рецептів;
- VIEW_QUICK_RECIPE_SCR – екран перегляду швидкого рецепта;
- VIEW_SAVE_RECIPE_SCR – екран перегляду збереженого рецепта;
- FONT_SIZE – розмір шрифту;
- FONT_COLOR – колір шрифту;
- BACK_COLOR – колір фону екранів;
- SELECTION_COLOR – колір обраного елементу екрану.

Для завдання цих констант використовуємо директиву #define, ця директива визначає ідентифікатор і послідовність символів, якій буде заміщатися даний ідентифікатор при його виявленні в тексті програми. Ідентифікатор також називається ім'ям макросу, а процес заміщення називається підстановкою макросу. Стандартний вид директиви наступний:

```
#define ім’я_макросу послідовність_символів
```

Перед завантаженням скетчу, користувач повинен буде вказати у константи пінів, піни плати Arduino, до яких під’єднані зовнішні модулі.

Перейдемо огляду глобальних змінних, що використовуються у скетчі програми.

Змінні датчиків, екранів, тощо:

- TFT_HX8357 tft – дисплей БК;
- OneWire oneWire – шина OneWire датчика температури, потрібна для передачі у якості вхідного параметру при створенні об’єкту DallasTemperature;
- DallasTemperature dts – датчик температури;
- iarduino_RTC rtc – модуль годинника реального часу;
- unsigned long last_press – час у мілісекундах з останнього натиснення кнопки;
- bool isRecipeCreated – якщо приймає істину, то рецепт було створено, інакше – ні;
- byte currentScreen – змінна, у якій зберігається номер поточного екрану;
- byte currentScreenItem – змінна, у якій зберігається номер поточного елементу екрану.

Змінні, що містять параметри рецепта:

- int amountOfPumping – кількість разів прокачування помпи;
- int maltTemp – температура додання солоду, в градусах Цельсію;
- int pausesTemps[5] – температури пауз затирання, в хвилинах;
- unsigned int pausesTimes[5] – часи пауз затирання, в хвилинах;
- unsigned int hopsTimes[5] – часи додання хмелю, в хвилинах;
- int boilingTemp – температура кип’ятіння, в градусах Цельсію;
- unsigned int boilingTime – час кип’ятіння, в хвилинах.

Змінні для зміни назви рецепта:

- String recipeName – назва нового рецепта;
- char symbols[3][13] – символи, якими формується назва рецепта;
- String recipesNames[8] – перші вісім рецептів на картці пам’яті;
- String currentRecipeName – назва поточного рецепта.

Змінні, що містять час і дату:

- int endTimerH – години завершення роботи таймера;
- int endTimerM – хвилини завершення роботи таймера;
- int endTimerS – секунди завершення роботи таймера;
- int endHop[5][3] – часи додання хмелю, під час роботи таймера;
- int dd – змінна для налаштування поточного дня;
- int mm – змінна для налаштування поточного місяця;
- int yy – змінна для налаштування поточного року;
- int h – змінна для налаштування поточної години;
- int m – змінна для налаштування поточної хвилини;
- int s – змінна для налаштування поточної секунди;
- String currentDate – строкове відображення поточної дати і часу.

Змінні, що містять стан критичних модулей:

- bool isSDCardInit – якщо приймає істину, то ініціалізація картки пам’яті проведено успішно, інакше – ні;
- bool isDTSFine – якщо приймає істину, то датчик температури в нормі, інакше – ні;
- bool isRTCFine – якщо приймає істину, то модуль годинника реального часу в нормі, інакше – ні.

3.4 Обов’язкові функції програми: setup() і loop()

Функція setup() викликається, коли стартує скетч. Використовується для ініціалізації змінних, визначення режимів роботи висновків, запуску використовуваних бібліотек і т.д. Функція setup запускає тільки один раз, після кожної подачі живлення або скидання плати Arduino[15][1)].

```
void setup() {
```

[1)] [15] Setup Программирование Ардуино. URL: https://doc.arduino.ua/ru/prog/Setup. (дата звернення 09.04.2019).

```
    Serial1.begin(9600);

    pinMode(BUTTON_UP_PIN, INPUT_PULLUP);
    pinMode(BUTTON_DOWN_PIN, INPUT_PULLUP);
    pinMode(BUTTON_PLUS_PIN, INPUT_PULLUP);
    pinMode(BUTTON_MINUS_PIN, INPUT_PULLUP);
    pinMode(BUZZER_PW_PIN, OUTPUT);
    pinMode(PUMP_PIN, OUTPUT);
    pinMode(HEAT_PIN, OUTPUT);

    dts.begin();
    dts.getTempCByIndex(0);

    tft.init();
    tft.setRotation(3);
    currentScreenItem = 1;
    setScreen(MAIN_SCR);

    rtc.begin();

    if (SD.begin(53)) {
        isSDCardInit = true;
        refreshRecipes();
    }
    else {
        isSDCardInit = false;
    }
}
```

Оглянемо, що відбувається у функції setup(). Встановимо піни живлення входу та виходу, запустимо датчик температури та модуль годинника реального часу, ініціалізуємо кольоровий дисплей та картку пам’яті, запустимо Serial-порт для зчитування надісланих команд(посилок), встановимо головний екран і дамо значення змінній currentScreenItem.

Функція pinMode конфігурує режим роботи вказаного виводу, як вхід або як вихід. Перший параметр – це номер виводу, режим роботи якого треба буде конфігурувати. Другий параметр приймає значення INPUT, OUTPUT або INPUT_PULLUP. При вказані INPUT_PULLUP, на піні будуть задіяні внутрішні підтягуючи резистори.

Після виклику функції setup(), яка ініціалізує і встановлює початкові значення, функція loop() робить точнісінько те, що означає її назва, і крутиться в циклі, дозволяючи програмі здійснювати обчислення і реагувати на

них. Функція використовується для активного керування платою Arduino[16][11)].

У функції loop() реалізовано загальний алгоритм роботи ПЗ, який був описаний у пункті 2.7.

```
void loop() {
    receiveParcel();

    if (millis() - timing > 1000) {
        dts.requestTemperatures();
        dts.getTempCByIndex(0) == -127 ?
            exceptionMessage(1): exceptionMessage(-1);

        !isSDCardInit ? exceptionMessage(3) :
            exceptionMessage(-3);

        currentDate = r
+tc.gettime("d-m-Y, H:i:s, D");
        currentDate.equals("01-01-2001, 01:01:01, Sun") ?
            exceptionMessage(2) : exceptionMessage(-2);

        tft.setCursor(5, 5, 1);
        tft.print(rtc.gettime("d-m-Y, H:i:s"));
        tft.setCursor(5, 15, 1);
        tft.print(dts.getTempCByIndex(0));
        tft.setCursor(35, 15, 1); tft.print("`C");
        timing = millis();
    }

    if (isRecipeCreated) {
        mashing();
        boiling();
    }
    else {
        createRecipe();
    }
}
```

Окрім реалізації загального алгоритму, у роботу функції було додано відображення, у лівому верхньому куті дисплею, поточної температури, дати і часу.

[1)] [16] Loop Программирование Ардуино. URL: https://doc.arduino.ua/ru/prog/Loop. (дата звернення 09.04.2019).

3.5 Технічні рішення при створенні меню налаштувань

Меню налаштувань складатиметься, як було сказано у пункті 2.4, з гілок екранів. Для виклику екрану буде використовуватися функція void setScreen(byte numOfScr), яка відтворюватиме екран, номер якого вказаний при виклику функції, у якості вхідного параметру. Також виклик цієї функції змінить значення змінної currentScreen на номер поточного екрану.

Не в останню чергу треба вирішити, як буде проходити етап обирання гілки меню і як чином користувач буде робити цей вибір. У схемі проекту БК, мається клавіатура з чотирма кнопкам. Нехай користувач натискаючи ці кнопки буде переміщуватися по елементам меню і змінювати їх стан. Поділимо кнопки на зони відповідальності: дві кнопки відповідатимуть за переміщення користувача у меню налаштувань, інші дві відповідатимуть за зміну значень параметрів рецепта, тощо.

Розглянемо першу пару. При натисканні першої кнопки, користувач буде перемикатися на наступний елемент екрану, назвемо цю кнопку «Вгору». А натискання на другу кнопку перемкне користувача на попередній елемент, ім’я у цієї кнопки буде «Вниз».

Тепер розглянемо наступну пару кнопок, що залишилися. Ці кнопки будуть використовуватися для зміни значення, наприклад параметру рецепта. Назвемо їх кнопки «Плюс» і «Мінус». Натиск на кнопку «Плюс» збільшить значення на одиницю, а натиск на «Мінус» − зменшить. Розширимо функціонал кнопки «Плюс» і додамо до її функцій підтвердження вибору, тобто коли користувач знаходитиметься на елементі екрану, який змінює екран, і натисне кнопку «Плюс», то цими діями користувач змінить поточний екран на наступний.

Натиск на ту чи іншу кнопку будемо вважати подією натиску. Ця подія виникатиме, коли користувач знаходиться на певному елементі екрану і натискатиме кнопку клавіатури. Для обробки цієї події будуть створені функції-обробники подій натиску кнопок на певному елементі екрану. Виклик об-

робників буде роботи функції слухачі, які відстежують поточне місце знаходження користувача у меню налаштувань.

Розглянемо на прикладі натиску кнопок на елементі екрану головного екрану. Дії які повинні відбутися після виникнення події натиску кнопок будуть прописані у функції void menuButtonClicked(). Розділимо огляд функції на дві частини. В першій частині буде огляд переходу від одного елементу екрану до іншого, в другій – підтвердження вибору.

```
if (isClicked(BUTTON_DOWN_PIN) && currentScreenItem < 3) {
    bool states[3];
    currentScreenItem++;

changeStates:
    switch (currentScreenItem) {
        case 1: states[0] = 1; states[1] = 0; states[2] = 0;
            break;
        case 2: states[0] = 0; states[1] = 1; states[2] = 0;
             break;
        case 3: states[0] = 0; states[1] = 0; states[2] = 1;
             break;
    }

    drawButton("QUICK RECIPE", 135, 110, 210, 30, states[0]);
    drawButton("MY RECIPES", 135, 145, 210, 30, states[1]);
    drawButton("SETTINGS", 135, 180, 210, 30, states[2]);
}

if (isClicked(BUTTON_UP_PIN) && currentScreenItem > 1) {
    currentScreenItem--;
    goto changeStates;
}
```

Спочатку перевіряється чи була натиснута кнопка «Вниз», викликом функції isClicked, де у якості вхідного параметри було передано номер піну до якого під’єднана ця кнопка, також перевіряється чи не було перевищено значення поточного елементу екрану currentScreenItem, яке не повинно перевищувати кількість елементів на головному екрані. Якщо перевірка пройдена, то значення поточного елементу екрана збільшується на одиницю. Потім в залежності від значення поточного елементу екрану змінюється стан елементів екрану, де поточний елемент стає активним, а інші два не активними. Стан елементів записується у масив states, де послідовність елементів масиву

співпадає з послідовністю елементів екрану. Завершальною дією є зміна кольору фону елементів екрану, за допомогою виклику функції drawButton, де останнім параметром передається стан елементу екрану.

Для того щоб повернутися до попереднього елементу екрана, треба виконати другу умову, де перевіряється чи була натиснута кнопка «Вгору» і чи значення поточного елементу екрану не менше за одиницю. Якщо умова виконується, то значення поточного елементу екрану зменшується на одиницю, після чого виконуються такі самі дії, що і в першій умові. Для того щоб не повторювати один й той самий код, було використано перехід до позначки changeStates.

Тепер розглянемо другу частину. В цій частині матиметься умова, виконання якої, призведе до заміни поточного екрану. Умова наступна:

```
if (isClicked(BUTTON_PLUS_PIN)) {
    switch (currentScreenItem) {
        case 1:
            setScreen(VIEW_QUICK_RECIPE_SCR);
            drawButton("Back", 5, 285, 70, 30, 1);
            currentScreenItem = 1;
            break;
        case 2:
            if (SD.begin(53)) {
                isSDCardInit = true;
                setScreen(RECIPES_LIST_SCR);
                drawButton("Back", 5, 285, 70, 30, 1);
                currentScreenItem = 1;
            }
            else {
                exceptionMessage(3);
            }
            break;
        case 3:
            setScreen(SETTINGS_SCR);
            drawButton("Back", 5, 285, 70, 30, 1);
            currentScreenItem = 1;
            break;
    }
}
```

Якщо була натиснута кнопка «Плюс», то викликається, відповідно до поточного елементу екрану, функція відтворення екранів setScreen, де у якості вхідного параметра передається номер екран, і значення поточного екрану перемикається на перший елемент наступного екрану. Слід звернути увагу на

те, що перехід до перегляду списика збережених рецептів здійснюється тоді, коли картка пам’яті ініціалізована, в іншому випадку, перехід не відбудеться і користувачеві буде надіслано на дисплей БК повідомлення про проблеми з карткою пам’яті, викликом функції exceptionMessage, де у якості вхідного параметру передається номер помилки.

Інші обробники подій схожі за своєю структурою і навіть можна виділити два типи обробників подій в цій програмі. Обробники подій на екрані і обробники подій на елементі екрану. Вище описана функція відноситься до першого типу.

Різниця типів складається в тому, що у обробнику подій елемента екрану передається у якості параметрів місце знаходження трьох суміжних елементів екрану і їх значення або текст, яке вони відображають. Розглянемо на прикладі подібну функцію, а саме обробник подій натиску кнопок на екрані перегляду рецепта при обраному елементі екрану з групи «Параметри рецепта»:

```
void recipeItemClicked(int *value1, int16_t x1, int16_t y1, int
*value2, int16_t x2, int16_t y2, int *value3, int16_t x3,
int16_t y3, int boundaryValue)
```

Ця функція нічого не повертає і має десять вхідних параметрів:

- int *value1 – значення, яке відображає попередній елемент екрану;
- int16_t x1 – координата по всі X лівого верхнього кута відображення попереднього елемента екрану;
- int16_t y1 – координата по всі Y лівого верхнього кута відображення попереднього елемента екрану;
- int *value2 – значення, яке відображає поточний елемент екрану;
- int16_t x2 – координата по всі X лівого верхнього кута відображення поточного елемента екрану;
- int16_t y2 – координата по всі Y лівого верхнього кута відображення поточного елемента екрану;

- int *value3 значення, яке відображає наступний елемент екрану;
- int16_t x3 – координата по всі Y лівого верхнього кута відображення наступного елемента екрану;
- int16_t y3 – координата по всі Y лівого верхнього кута відображення наступного елемента екрану;
- int boundaryValue – граничне значення, яке може приймати значення поточного елемента екрана.

При натисненні користувачем кнопок «Плюс» чи «Мінус», значення, яке закріплено за поточним елементом екрану, відповідно буде збільшено або зменшено на одиницю. Значення, яке було передано, обмежено у діапазоні від нуля до граничного значення(boundaryValue). Після зміни значення, поточний елемент екрану оновлюється на дисплеї БК. Нижче наведено код програми, що був описаний.

```
if (isClicked(BUTTON_PLUS_PIN) && (*value2) < boundaryValue) {
    (*value2)++;
    drawSelectedValue(*value2, x2, y2, 58, 33, 1)
}

if (isClicked(BUTTON_MINUS_PIN) && (*value2) > 0) {
    (*value2)--;
    drawSelectedValue(*value2, x2, y2, 58, 33, 1);
}
```

Переміщення користувача так само здійснюється натиском кнопки «Вгору», щоб перейти до наступного елементу екрана, і кнопки «Вниз», щоб перейти до попереднього елементу екрана. Якщо була натиснута будь-яка з цих кнопок, то значення змінної поточного елементу екрану currentSreenItem зміниться на одиницю, відповідно до тієї кнопки, що була натиснута. Після чого колір фон поточного елементу екрану стане дорівнювати кольору фону екрана, а наступний(чи попередній) елемент екрану змінить колір свого фону на колір зазначений у константі SELECTION_COLOR.

Тепер розглянемо приклад частину головної функції-слухача, яка діє на екрані перегляду списку збережених рецептів.

На цьому екрані присутні дев’ять елементів екрану. Для зручності кожну елементу надамо порядковий номер. Нумерація елементів цього екрану здійснюється наступним чином. Першим йде елемент екрану, який поверне користувача на попередній екран, потім у дві колонки зверху вниз елементи екрану, які направлять користувача до екрану перегляду відповідного збереженого рецепта.

Функція createRecipe дивиться на якому екрані і на якому елементі екрану користувач зупинився, на це їй вказують змінні currenScreen і currenScreenItem відповідно. Після чого викликається функція-обробник, яка реагує на натиснення кнопок клавіатури, у даному випадку змінює поточний екран і з ним значення змінної currentScreen.

```
void createRecipe() {
    switch (currentScreen) {
    ...
        case RECIPES_LIST_SCR:
            switch (currentScreenItem) {
                case 1: recipeNameClicked(0, -1, 0, -1, 0, -1,
40,
                    0, 0); break;
                case 2: recipeNameClicked(7, 170, 0, 80, 1, 110,
                    40, 40, 40); break;
                case 3: recipeNameClicked(0,   80, 1, 110, 2,
140,
                    40, 40, 40); break;
                case 4: recipeNameClicked(1, 110, 2, 140, 3,
170,
                    40, 40, 40); break;
                case 5: recipeNameClicked(2, 140, 3, 170, 4, 80,
                    40,  40, 270); break;
                case 6: recipeNameClicked(3, 170, 4,   80, 5,
110,
                    40, 270, 270); break;
                case 7: recipeNameClicked(4,  80, 5, 110,   6,
                    140,  270, 270, 270); break;
                case 8: recipeNameClicked(5, 110, 6, 140, 7,
170,
                    270, 270, 270); break;
                case 9: recipeNameClicked(6, 140, 7, 170, 255,
                    -1, 270, 270, 270); break;
            }
            checkSDCard();break;
    ...
}}
```

Таким чином описується кожний елемент екрану на кожному екрані, за умови що він був обраний користувачем.

3.6 Технічні рішення при створені інструментів роботи з файлом рецепта

Для роботи з файлом рецепта було створено наступні функції:

- void getRecipe(String name) – зчитає з файлу рядок параметрів рецепта і розфасує їх у відповідні змінні;
- void putRecipe(String name) – створить файл рецепта, у який помістить рядок парметрів рецепта;
- void deleteRecipe(String name) – видалить вказаний рецепт та оновить список рецептів.

Розглянемо більш детально роботу першої функції. Ця функція має один вхідний параметр name, у який передається ім’я рецепта параметри, якого потрібно зчитати. Функція має наступні змінні:

- String buf – буфер, який міститиме параметр рецепта;
- char symbol – один зчитаний з файлу символ;
- int arr[18] – масив з усіма параметрами рецепта;
- byte num – вказівник на елемент масиву arr.

В першу чергу відкривається файл для зчитування. Якщо відкриття було проведено успішно, то зчитується кожен байт. Зчитаний байт буде занесена у змінну symbol. Ніяких завад у здійснені цієї операції не має тому, що в цьому випадку вказується код символу з таблиці ASCII. Отримані символи будуть заноситися доти, поки не зустрінеться кома. Після чого вміст буфера перетворюється у ціле число і заноситься у масив arr. Потім буфер очищається і вказівник масиву arr переміщується на наступний елемент. Так триває до тих пір, коли у файлі рецепта не закінчиться байтів для зчитування.

По закінченню зчитування, файл рецепта буде закрито. Заповнений параметрами рецепта масив arr, треба розфасувати по відповідним змінним. Кінцевим кроком є закриття файлу.

```
void getRecipe(String name) {
    String buf = "";
    char symbol = 0;
    int arr[18], num = 0;

    File file = SD.open(name);

    if (file) {
        while (file.available()) {
            symbol = file.read();
            delay(1);

            if (symbol != ',')
                buf += symbol;
            else {
                arr[num] = buf.toInt();
                buf = "";
                num++;
            }
        }
        file.close();
    }
    else
        file.close();

    for (int i = 0, j = 6, k = 11; i < 5, j < 11, k < 16; i++,
        j++, k++) {

        pausesTemps[i] = arr[i + 1];
        pausesTimes[i] = arr[j];
        hopsTimes[i]   = arr[k];
    }
    maltTemp = arr[0];
    boilingTemp = arr[16];
    boilingTime = arr[17];
    file.close();
}
```

Послідовність параметрів рецепта у рядку даних і у масиві arr було описано у пункті 2.5.

Тепер перейдемо до розгляду другої функції, а саме функції збереження файлу рецепта void putRecipe(String name), у яку вхідним параметром name передається ім’я нового рецепта. У функції оголошена строкова змінна data, у якій буде сформовано рядок параметрів рецепта. Після формування

рядку, з картки пам’яті буде видалений файл з ім’ям, яке дорівнює значенню вхідного параметру name, це робиться у випадках коли змінюються параметри вже збереженого рецепта. У разі якщо на картці пам’яті немає файлу з ім’ям, яке зазначено у параметрі name, то нічого не відбудеться.

```
void putRecipe(String name) {
    String data = "";
    data += maltTemp; data += ",";

    for (int i = 0; i < 5; i++) {
        data += pausesTemps[i]; data += ","; }
    for (int i = 0; i < 5; i++) {
         data += pausesTimes[i]; data += ","; }
    for (int i = 0; i < 5; i++) {
         data += hopsTimes[i]; data += ","; }

    data += boilingTemp; data += ",";
    data += boilingTime; data += ",";

    SD.remove(name);
    File file = SD.open(name, FILE_WRITE);

    if (file) {
        file.print(data);
        file.close();
    }
    else
        file.close();

    refreshRecipes();
}
```

Потім буде створено файл для запису, що символізує прапор FILE_WRITE. Після чого, якщо файл було відкрито, то записується рядок параметрів рецепта і закривається файл. Кінцевим кроком є оновлення списку рецептів викликом функції refreshRecipes.

3.7 Технічні рішення при створені інструментів для дистанційного керування БК

Для дистанційного керування було створено наступні функції:

- void receiveParcel() – приймає команди(посилки) від віддаленого пристрою;

- void getParameters(char buf[], byte shift) – розфасовує прийняті дані з віддаленого пристрою по відповідним змінним;
- void getName(char buf[]) – формує з прийнятих даних ім’я рецепта.

Розглянемо більш детально роботу першої функції. Ця функція очікує коли у COM порті з’являться байти даних для зчитування. Якщо дані з’явилися, то зчитується перший надісланий символ, і якщо цей символ такий самий, що і ідентифікатор початку передачі команди(див. п.2.6), то буфер cmd заповнюється наступними шістьма символами, після чого у строкову змінну parcelCmd записуються елементи буфера cmd. Таким чином отримується назва команди, яку необхідно виконати.

```
void receiveParcel() {
    char cmd[6];
    char val;

    while (Serial1.available() > 0) {
        val = Serial1.read();

        if (val == '/') {
            Serial1.readBytes(cmd, 6);
            parcelCmd = "";

            for (byte i = 0; i < 6; i++) {
                parcelCmd += cmd[i];
            }
        }

        if (val == '|') {
            ...
        }
    }
}
```

Потім після отримання ідентифікатору початку даних перевіряється отримана команда зі списком зарезервованих команд виконання і якщо є спів падіння, то створюється буфер для зчитування надісланих даних. Після чого заповнений буфер передається на обробку до функції getParameters або getName. І останнім кроком виконуються відповідні дії виконання надісланої команди.

Перейдемо до огляду функцій getParameters і getName.

Функція getParameters призначена для розсортування отриманих даних, які є параметрами рецепта, по відповідним змінним. Вона приймає два вхідних параметра буфер даних на обробку buf і зміщення вказівника переданого буфера shift. В тілі функції оголошено наступні змінні:

- String num – буфер, що міститиме строкове уявлення параметру рецепта;
- int arr[18] – масив з усіма параметрами рецепта;
- byte pointer – вказівник масиву arr;

Спочатку заповнюється строковий буфер num елементами переданого буфера buf, до тих пір, коли кількість елементів стане у буфері num дорівнюватиме трьом елементам. Після чого вміст буфера num конвертується у ціле число, записується у масив arr і буфер очищається. Ці дії триватимуть доти, поки вказівник у буферу buf не дійде до кінця цього буфера .

```
void getParameters(char buf[], byte shift) {
    String num; byte pointer = 0; int arr[18];

    for (byte i = shift, count = 0; i < 55 + shift; i++) {
        if (count == 3) {
            count = 0;
            arr[pointer] = num.toInt();
            num = "";
            pointer++;
        }

        if (i == 55 + shift) break;

        num += buf[i];
        count++;
    }

    for (int i = 0, j = 6, k = 11; i < 5, j < 11, k < 16; i++,
j++, k++) {
        pausesTemps[i] = arr[i + 1];
        pausesTimes[i] = arr[j];
        hopsTimes[i]   = arr[k];
    }
    maltTemp = arr[0];
    boilingTemp = arr[16];
    boilingTime = arr[17];
}
```

Потім заповнений масив всіх параметрів рецепта arr розфасовується по відповідним змінним.

Функція getName призначена для отримання назви рецепта з буфера прийнятих даних, який передається у якості вхідного параметру цієї функції.

```
void getName(char buf[]) {
    recipeName = "";

    for (byte i = 0; i < 8; i++) {
        recipeName += buf[i];
    }
}
```

Спочатку змінна назви рецепта recipeName очищається перед заповненням. Потім вона заповнюється вісьма символами з переданого буферу buf. Таким чином отримується ім’я рецепта.

ВИСНОВКИ

При виконанні проекту дипломної роботи було оглянуто та проаналізовано:

- технологічні процеси виготовлення пива;
- складові схеми БК;
- інструменти розробки ПЗ;

Розроблені:

- алгоритми технологічних процесів виготовлення пива;
- меню налаштувань;
- збереження рецептів користувача;
- модель дистанційного керування і інструменти її підтримки.

Результатом виконання проекту дипломної роботи стало – ПЗ для БК домашньої пивоварні на апаратній платформі Arduino, яке автоматизує технологічні процеси виготовлення пива, а саме процес затирання солоду і кип’ятіння сусла.

Розроблений програмний продукт в поліній мірі відповідає всім вимогам, що були пред’явлені у ТЗ(див. п.1.2) і має наступні характеристики:

- скетч програми використовує 66954 байт (26%) пам’яті пристрою;
- глобальні змінні використовують 2154 байт (26%) динамічної пам’яті.

Цільова платформа ПЗ – Arduino Mega 2560 з мікроконтролером ATmega2560.

В подальшому програмний продукт буде проходити етапи модернізації спрямовані на зменшення використання об’ємів пам’яті для розширення модельного ряду апаратних платформ Arduino, що підтримуються, а також планується доповнити програмний продукт мобільним додатком, для дистанційного керування БК домашньої пивоварні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Что такое Arduino?. URL: http://arduino-diy.com/arduino-chto-eto-takoye. (дата звернення 20.02.2019).
2. Как сварить пиво в домашних условиях классический рецепт. URL: https://alcofan.com/kak-svarit-pivo-v-domashnix-usloviyax-klassicheskij-recept.html. (дата звернення 02.03.2019).
3. Assembler – Энциклопедия языков программирования. URL: http://progopedia.ru/language/assembler/. (дата звернення 02.03.2019).
4. Языки программирования C и C++. URL: https://metanit.com/cpp/. (дата звернення 02.03.2019).
5. Lua – Энциклопедия языков программирования. URL: http://progopedia.ru/language/lua/. (дата звернення 02.03.2019).
6. Basic – Энциклопедия языков программирования. URL: http://progopedia.ru/language/basic. (дата звернення 02.03.2019).
7. PL/M – Энциклопедия языков программирования. URL: http://progopedia.ru/language/plm/. (дата звернення 05.03.2019).
8. Arduino IDE – програми для Windows. URL: https://programy.com.ua/ua/arduino_ide/. (дата звернення 05.03.2019).
9. Анализ сред программирования для МК. URL: http://www.gaw.ru/html.cgi/txt/soft/avr/Algorithm_Builder.htm. (дата звернення 05.03.2019).
10. Algorithm Builder. Графическая среда разработки программного обеспечения для микроконтроллеров AVR. (дата звернення 05.03.2019).
11. Универсальная библиотека iarduino_RTC для RTC DS1302, DS1307, DS3231 к Arduino. URL: https://iarduino.ru/file/235.html. (дата звернення 09.04.2019).

12. Arduino: Библиотеки/OneWire – Онлайн справочник. URL: http://wikihandbk.com/wiki/Arduino:Библиотеки/OneWire. (дата звернення 09.04.2019).
13. SD Программирование Ардуино. URL: https://doc.arduino.ua/ru/prog/SD. (дата звернення 09.04.2019).
14. SPI Программирование Ардуино. URL: https://doc.arduino.ua/ru/prog/SPI. (дата звернення 09.04.2019).
15. Setup Программирование Ардуино. URL: https://doc.arduino.ua/ru/prog/Setup. (дата звернення 09.04.2019).
16. Loop Программирование Ардуино. URL: https://doc.arduino.ua/ru/prog/Loop. (дата звернення 09.04.2019).