Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп’ютерних наук та штучного інтелекту
Кафедра комп’ютерних систем та робототехніки

«Затверджую»
в.о. завідуючого кафедри
комп’ютерних систем та робототехніки
_____ к. ф.-м. н., доцент Максим Хруслов
«___» червня 2025 р.

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «**МОДЕЛЬ МІКРОКОНТРОЛЕРНОГО КЕРУВАННЯ ДИСПЛЕЄМ НА БАЗІ HD44780**»

Спеціальність 151 –Автоматизація та комп'ютерно-інтегровані технології
Галузь знань 15 – Автоматизація та приладобудування
Освітня програма «Автоматизація та комп’ютерно-інтегровані технології»

Захищено на засіданні
Екзаменаційної комісії № 46
протокол № __ від __.06.2025 р.
Оцінка _____ / _____
Голова Екзаменаційної комісії
_____ **ЧУГАЙ А.М.**

Виконав:
Студент групи КУ– 41
ЛИТИНСЬКИЙ Владислав Максимович_____

Керівник: доцент ЗВО кафедри КСтаР, К.ф.-м.н, доцент КОТВИЦЬКИЙ Альберт Тадеушевич
_____

Рецензент: доцент ЗВО кафедри загальної фізики фізичного факультету, К.т.н., доцент ГРЕСЬ Валерія Юріївна_____

Харків – 2025

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і п’яти додатків. Загальний обсяг роботи складає 76 сторінок, із яких 49 сторінок основної частини з 26 рисунками, 4 таблицями, 10 найменуваннями списку використаних джерел та п’ятьма додатками.

Метою кваліфікаційної роботи є реалізація мікроконтролерного керування рідкокристалічним дисплеєм LCD1602 та розробка методики виводу символів кирилиці на дисплей.

Об’єкт дослідження – проектування та розробка алгоритмічних підходів керування при роботі із символьними рідкокристалічними індикаторами та дисплеями.

Предмет дослідження - методика мікроконтролерного керування LCD1602 на базі контролера HD44780.

Проблема, яка вирішується в кваліфікаційній роботі полягає в розробці методики виводу символів кирилиці на рідкокристалічний дисплей LCD1602.

Область застосування – промисловість, робототехніка, сфера малого бізнесу.

Ключові слова: LCD1602, Arduino IDE, Proteus, ATmega2560, контролер HD44780, рідкокристалічний дисплей.

ABSTRACT

An explanatory note to the master's attestation work is created in the introduction, three sections, conclusions, a list of sources used and four additional substances.

The total volume of work is 76 pages, of which 49 pages of the main part with 26 figures, 4 table, 10 names of the list of used sources and five additions.

The purpose of the qualification work is to implement microcontroller control of the LCD1602 liquid crystal display and to develop a methodology for displaying Cyrillic characters on the display.

The object of research is the design and development of algorithmic control approaches when working with character liquid crystal indicators and displays.

The subject of research is the method of microcontroller control of LCD1602 based on the HD44780 controller.

The problem to be solved in the qualification work is to develop a methodology for displaying Cyrillic characters on the LCD1602 liquid crystal display.

The scope of application is industry, robotics, small business.

Keywords: LCD1602, Arduino IDE, Proteus, ATmega2560, HD44780 controller, liquid crystal display.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

ACG	–	Address for CG RAM;
ADD	–	Address Data Display;
ASCII	–	American Standard Code for Information Interchange;
B	–	Blink of cursor;
C	–	Cursor;
CGRAM	–	Character Generator ROM;
D	–	Decrement;
DB	–	Data Bus;
DL	–	Data Length;
DDRAM	–	Display Data RAM;
E	–	Enable;
F	–	Character font;
I	–	Increment;
L	–	Left;
LCD	–	Liquid Crystal Display;
N	–	Number of display line;
S	–	Shift;
R	–	Right;
RS	–	Register Select;
R/W	–	Read/Write;
UTF	–	Unicode Transformation Format;
VDD	–	Voltage Drain Drain;
VSS	–	Voltage Source Supply;

ВСТУП

Актуальність теми. Останніми роками на ринку електроніки стрімко зростає кількість нових мікроконтролерів від різноманітних виробників, таких як STMicroelectronics (STM32), Espressif Systems (ESP32, ESP8266), Raspberry Pi (RP2040), Nordic Semiconductor (nRF52), GigaDevice (GD32) та Arduino (AVR). Їхня популярність пояснюється гнучкістю, доступністю та широкими можливостями інтеграції різноманітних периферійних пристроїв (дисплеї, сенсори, модулі зв'язку тощо). Проте для деяких із цих мікроконтролерів ще не існує готових бібліотек для роботи із базовими компонентами зокрема такими, як LCD-дисплеї.

У такому випадку програмування на мові С є необхідним і часто одним з можливих підходів, що дозволяє безпосередньо керувати периферією. Крім того, код, написаний на мові С без використання сторонніх бібліотек, зазвичай займає менше пам'яті та виконується швидше, що критично для мікроконтролерів з обмеженими ресурсами.

Рідкокристалічний дисплей LCD1602 залишається надзвичайно поширеним і актуальним завдяки своїй простій конструкції, низькій вартості та широкій доступності. Його активно застосовують у навчальних лабораторіях, хобі-проєктах, побутовій електроніці, системах моніторингу та керування. Одним із основних недоліків цього дисплея є відсутність вбудованої підтримки кирилиці в стандартній прошивці контролера HD44780, що ускладнює його використання в україномовних застосуваннях.

Тому розробка методики виведення кириличних символів на дисплей LCD1602 засобами мови програмування С є актуальною та практично значущою задачею — як у навчальному процесі, так і в реальних технічних проєктах.

Мета роботи: Реалізувати мікроконтролерне керування рідкокристалічним дисплеєм LCD1602 та розробити методику виводу символів кирилиці на дисплей.

Об’єкт роботи: Проектування та розробка алгоритмічних підходів керування при роботі із символьними рідкокристалічними індикаторами та дисплеями.

Предмет роботи: Методика мікроконтролерного керування LCD1602 на базі контролера HD44780.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ ОСНОВИ РОБОТИ З РІДКОКРИСТАЛІЧНИМ ДИСПЛЕЄМ LCD1602 НА БАЗІ КОНТРОЛЕРА HD44780

1.1 Історична довідка

Контролер Hitachi HD44780 - це контролер на базі рідкокристалічного LCD дисплея з алфавітно-цифровою матрицею, розробленою компанією Hitachi в 1980-х роках. В контролер HD44780 на апаратному рівні закладено певний набір символів, що включає:

- символи ASCII таблиці, зокрема літери латинського алфавіту;
- японські символи Kana та інші спеціальні символи; [1]

На основі вбудованого драйверу у контролер HD44780, пристрій може відображати до 80 символів. Окрім цього дисплея сконструйованим компанією Hitachi існують деякі аналоги на ринку, які подібні до цього.

Наприклад: ряд моделей на базі - KS0066, KS0076, KS0070, S6A0069 від фірми Samsung, SED1278 від Epson та інші аналоги, що присутні [2]. Через це численні дисплеї від сторонніх виробників сумісні з його 16-контактним інтерфейсом і набором інструкцій, що робить його популярним серед інших і дешевим для рідкокристалічного дисплея.

HD44780 є популярним серед інших та має широкий загал використання у громадському транспорті (метро, тролейбус, трамвай), у промислових виробах й побутових пристроях (пристрої розумного дому (термостат, розумний таймер), пральна машина, електронний годинник) та інші пристрої, де він застосовується.

1.2 Документація LCD 1602 (на базі HD44780)

1.2.1 Будова дисплею

Рідкокристалічний дисплей LCD1602 на базі контролера HD44780 має дисплей розмірністю 16x2, де 16 -кількість символів, яку можна вивести на одну строку; 2- кількість строк, що використовуються для виводу інформації на нього. Також можна побачити на ньому, що стандартний розмір символів є 5x8 точок.

На корпусі LCD1602 один за одним розташовані 16 пінів, які можна використовувати для роботи з ним. Перелік пінів, які є [3]:

Таблиця 1.1

Призначення пінів LCD1602

№	Назва піна	Призначення
1	VSS	Живлення контролера (-)
2	VDD	Живлення контроллера (+)
3	V0(Contrast Adjustment)	Управління контрастністю дисплея
4	RS(Register Select)	Визначення передачі команди/даних (якщо RS дорівнює рівню логічній одиниці відправляємо дані на дисплеї, а якщо RS дорівнює рівню логічному нулю відправляємо команди для керуванням дисплеєм)
5	R/W(Read/Write)	Режим читання/запису(Якщо RW = 0, то активований режим запису у дисплей, RW = 1, то активований режим читання або зчитування інформації з дисплею.
6	E(Enable)	Імпульс, що визначає сигнал для читання/запису з пінів DB0-DB7, RS та R/W
7	DB0-DB3(Data Bus 0-3)	Молодші біти для 8-бітного інтерфейса
8	DB4-DB7(Data Bus 4-7)	Старші біти для 4 або 8-бітного інтерфейса
9	A(Anode)	Анод (+) живлення підсвічування
10	K(Cathode)	Катод (-) живлення підсвічування

1.2.2 Відеопам’ять та адресація дисплея

Дисплей LCD1602 на базі контролера HD44780 має відеопам’ять, що вміщує 80 символів. Адресація дисплея розпочинається із першого рядка, 0-го символа (0x00) і так до 15-го символа (0x0F) першого рядка. Приклад адресації перших 16 символів на двох рядках наявні нижче з даташиту [4]

Так само і для другого рядка, але адресація там буде розпочинатися із 64 символа адресу осередка пам’яті (0x40) та закінчуватиметься 79 символом (0x4F).Приклад адресації перших 16 символів на двох рядках наявні нижче на рис 1.1 з даташиту

9. Display Address

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Line 2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Рисунок 1.1 – Адресація символів дисплея у пам’яті DDRAM.

Описана адресація вище це саме ті 16 символів на двох рядках, які ми бачимо візуально. А інші символи, вони сховані у внутрішній пам’яті дисплея. Вони доступні для запису, але дисплей не виведе їх.

1.2.3 Ініціалізація дисплею та режими роботи

Перед початком роботи з дисплеєм LCD1602 пропонується обрати режим роботи із ним. У контролер HD44780 закладено такі 2 режими роботи, а саме 4-бітний та 8-бітний. Абстрактно між ними різниці ніякої не відчувається, адже ці 2 режими (4-бітний та 8-бітний) виконують свою функцію передачі даних у рідкокристалічний дисплей.

Але все ж таки різниця між ними є. Мінуси та плюси використання наведено нижче.

- Мінуси використання 8-бітного режиму;

  Якщо ми будемо використовувати 8-бітний режим роботи із дисплеєм, то під час збирання та налаштування схеми разом з макетною платою та мікроконтролером може знадобитися на 4 піни більше. Що в свою чергу збільшує використання кількості проводів, адже це не зовсім ергономічно зі сторони використання та підключення даних проводів від LCD дисплея до мікроконтролера.

- Плюси використання 8-бітного режиму;

  В свою чергу також є свої плюси у використанні 8-бітного режиму передачі даних. Під час написання функції запису та передачі даних або команди, ми

до дисплею від разу передаємо 8-біт інформації без розділу на півбайта та зсуву його байтів.

- Плюси використання 4-бітного режиму;

На мою думку великим плюсом використання 4-бітного режиму це саме зручність підключення від дисплея до мікроконтролера, що з іншого боку зменшує кількість використовуваних проводів для підключення на 4 штуки менше.

- Мінуси використання 4-бітного режиму

Як було вже сказано вище, що етапи передачі даних до дисплея із використанням 4-бітного або 8-бітного режимів суттєво не відрізняються між ними.

А в реалізації функції запису та передачі даних або команд є різниця, тому що потрібно передати 8-бітів, а в 4-бітному режиму, спочатку потрібно зсунути окрему кількість біт до старших, і разом із цим передати молодші біти, що з іншого боку збільшує складність ефективності алгоритму.

Обов’язково перед початком із дисплеєм потрібно провести деякі етапи ініціалізації незалежно від обраного режиму роботи LCD1602 для нормального функціонування його роботи.

Якщо подивитись, як відбуваються етапи ініціалізації у різних режимах роботи(4-бітний та 8-бітний), що пропонує даташит рис 1.2 нижче, то різниці майже немає.

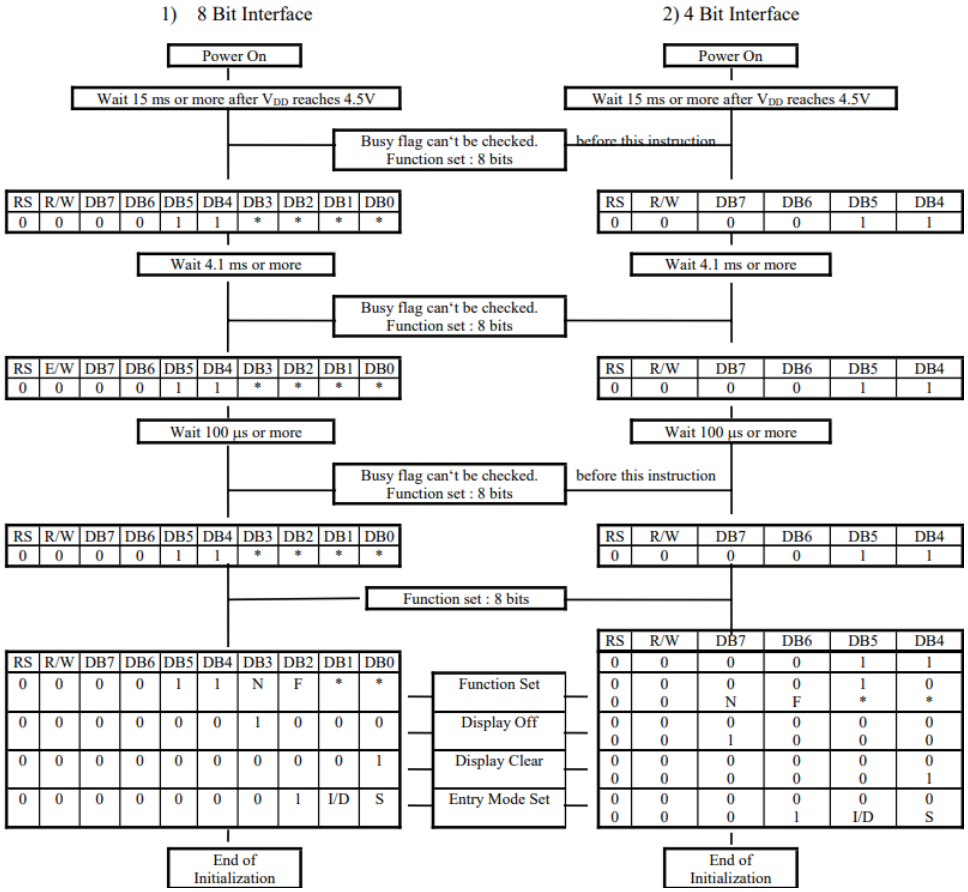

Рисунок 1.2 – Етапи ініціалізації у 4-бітному та 8-бітному режимі роботи.

Етапи ініціалізації у 4 бітному режимі:

1. Подати живлення на дисплей у розмірі 4.5 v, чекаємо приблизно 15мс
2. На відповідні біти D4-D7 повинні записати команду 3 до дисплея у двійковому вигляді(0b0011) та виставляємо затримку 4.1 мс або більше
3. Таку операцію потрібно повторити, ще 2 рази, тільки затримка буде 100 мкс або більше
4. Далі відправляємо на ці біти команду 2 до дисплея у двійковому вигляді (0b0010)
5. За допомогою відповідних команд, які закладено на програмному рівні у контролері HD44780, налаштовуємо дисплей, як нам потрібно. Наприклад : кількість строк на дисплеї, режим роботи, розмір шрифта (5x8 або 5x10 точок) та інші.

Етапи ініціалізації у 8 бітному режимі:

1. Подати живлення на дисплей у розмірі 4.5 v, чекаємо приблизно 15мс;
2. На відповідні біти D0-D3 повинні записати команду 48 до дисплея у двійковому вигляді(0b00110000) та виставляємо затримку 4.1 мс або більше;
3. Таку операцію потрібно повторити, ще 2 рази, тільки затримка буде 100 мкс або більше;
4. За допомогою відповідних команд, що запрограмовані у контролері HD44780, налаштовуємо дисплей,як нам потрібно. Наприклад : кількість строк на дисплеї, режим роботи, розмір шрифта (5x8 або 5x10 точок) та інші.

1.2.4 Вбудовані команди LCD1602

Контролер HD44780 дисплея LCD1602 має вбудовані команди, що полегшують його керування. Повний перелік команд наведено в таблиці 1.2, що містяться в даташиті [4].

Таблиця 1.2

Команди для керування LCD1602

№	Назва команди	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Опис команди
1	Clear Display	0	0	0	0	0	0	0	1	Виконує очистку дисплея.
2	Return Home	0	0	0	0	0	0	1	0	Повертає курсор у початкове положення на 1 рядок в 1-шу позицію на дисплеї.
3	Entry mode set	0	0	0	0	0	1	I/D	S	Налаштовує зміщення курсора або дисплея з урахуванням напрямку зсуву, якщо є інформація на дисплеї.

4	Display ON/OFF Control	0	0	0	0	1	D	C	B	Керує станами дисплея вмикаючи/вимикаючи,застосовує курсор та його миготіння.
5	Cursor or Display shift	0	0	0	1	S/C	R/L	0	0	Застосовує керування зсувом дисплея та курсора в певному напрямку.
6	Function Set	0	0	1	DL	N	F	0	0	Налаштовує на дисплеї режим передачі даних(4-бітний, 8-бітний), кількість рядків під час роботи із дисплеєм, а також застосовує шрифт символів(5x8 , 5x10 точок)
7	Set CGRAM address	0	1	ACG						Виводить створені спеціальні символи у відеопам’ять дисплею із CGRAM де вони знаходяться у комірках пам’яті ACG Приклад застосування команди є Розділі 2, пункт 2.2.8.
8	Set DDRAM address	1	ADD							Встановлює курсор на довільну позицію та рядок , передаючи ADD адресу для встановлення курсора. Використання команди наявно у Розділі 2, пункт 2.2.7.

Опис комбінацій усіх команд записано у примітці нижче до кожної з них. Команди, що є в даташиті, але не увійшли до таблиці 1.2 використовуються безпосередньо, що не потребують уваги в їхній реалізації.

Наприклад команди Write / Read Data їх не потрібно реалізовувати. За допомогою регістра R/W(Read/Write) на дисплеї ми можемо змінювати його стан з використанням макетної плати, підключивши провід до аноду або катоду.

Якщо провід підключений до катоду, то стан RW = 0, тобто у нас буде активований режим запису у дисплей, а якщо ми під'єднаємо його катоду стан RW = 1, то буде застосований режим читання або зчитування інформації з дисплею.

Примітка:

- Команда № 3 (див. таблицю 1.2). Використовуємо I (інкремент) = 1 або D (декремент) = 0 для визначення напрямку зсуву курсора вперед або назад на одну позицію. Застосовуємо S (зсув) = 1, для зсуву курсора якщо у нас на дисплеї є дані, якщо S (зсув) = 0, то курсор ми не будемо зсувати;

  Комбінації команди №3 (binary/hex):

  0000 0111/0x07- зсуває курсор разом із текстом(якщо він є);

  0000 0110/0x06 -пересуває курсор, але текст залишається на місці;

  0000 0100/0x04 -не пересуває текст і курсор:

- Команда № 4. D (дисплей) = 1 або D = 0 застосовуємо для увімкнення/вимкнення дисплея, C (курсор) = 1 або 0 використовується для увімкнення або вимкнення курсора на рідкокристалічному дисплеї, B (блимання курсору) =1 або 0 експлуатуємо для керування миготінням курсора на дисплеї;

  Комбінації команди №4 (binary/hex):

  0000 1111/0x0F -увімкнений дисплей, курсор та включене миготіння курсора;

  0000 1110/0x0E-увімкнений дисплей, курсор та виключене миготіння курсора;

0000 1100/0x0C -увімкнений дисплей, вимкнений курсор та миготіння курсора;

0000 1000/0x08 - вимкнений дисплей, курсор та миготіння курсора

- Команда №5. S (зміщення дисплея) =1, C (зміщення курсора) = 0 використовуючи це ми обираємо, що будемо зміщувати дисплей або курсор, також при налаштуванні команди ми обираємо напрям зсуву дисплея або курсора R (вправо) =1, L (вліво) = 0;

  Комбінації команди №5 (binary/hex):

  0001 1100/0x1C -зсув тексту на дисплеї вправо;

  0001 1000/0x18- зсув тексту на дисплеї вліво;

  0001 0000/0x10 - зсув курсора на дисплеї вліво;

  0001 0100/0x14 - зсув курсора на дисплеї вправо;

- Команди № 6. Під час проведення ініціалізації нам потрібно виконати налаштування на дисплеї:

  1.Обрати режим роботи дисплея. 8-бітний режим - DL=1 або 4-бітний режим - DL=0;

  2.Вибрати кількість рядків, що будуть використовуватись під час роботи із ним. 2 рядки – N=1, 1 рядок N=0;

  3.А також можна вибрати розмір шрифта символа, якщо це потрібно за замовчуванням шрифт вже встановлений 5x8 точок. 5x8 точок – F =1, 5x10 точок – F = 0;

  Комбінації команди №6 (binary/hex):

  8 – бітний режим:

  0011 1100/0x3C -встановлення використання 2 строк, шрифт символу 5x10;

  0011 1000/0x38 - встановлення використання 2 строк, шрифт символу 5x8;

  0011 0000/0x30 - встановлення використання 1 строки, шрифт символу 5x8;

  0000 1101/0x0D - встановлення використання 1 строки, шрифт символу 5x10;

  4 – бітний режим:

  0010 0000/0x20 - встановлення використання 1 строки, шрифт символу 5x8;

0010 1000/0x28 - встановлення використання 2 строк, шрифт символу 5x8;

0010 1100/0x2C- встановлення використання 2 строк, шрифт символу 5x10;

0000 1001/0x09 - встановлення використання 1 строки, шрифт символу 5x10

1.2.5 Символи

Рідкокристалічний дисплей LCD1602 на базі контролера HD44780 має в собі вже вбудовані символи латинського алфавіту, цифри, японські символи алфавіту Катакана та символи грецького алфавіту й деякі спеціальні символи на кшталт скобка, кома, точка та інші.

Незалежно від обраного режиму (4-бітний та 8-бітний) LCD1602 має можливість додатково додавати та зберігати й виводити на екран 8 спеціальних своїх символів. Це є обмеженням для дисплея такого типу.

Для спеціальних символів використовується CGRAM (див розділ ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ) пам’ять, яка має розмір 64 байти, якраз для зберігання спеціальних символів.

Під час створення символів у CGRAM виділяється місце комірок ACG (Address for CGRAM) (див рис 1.3) з адресами від 0 до 7, де будуть зберігатись спеціальні символи. Старші 3 біти(DB5-DB7) у нас ігноруються при створенні символа, тому для створення одного символа використовуються 5 молодших бітів (DB0-DB4). Пікселі символа, що мають вигляд матриці 5x8 у двійковому вигляді, відмічені 0 та 1, де 0 -не активний піксель, а 1- активний піксель.

Після створення спеціального символа йому присвоюється командою (див. рис 1.3) адрес у CGRAM пам’яті, де він знаходиться у адресовій частині ACG та зберігається у ній. Аби його вивести потрібно відправити команду (Send_byte_4bit_mode(«номер символа (0 – 7) , що закладено у CGRAM »,1), де 1 позначає що ми відправляємо дані до дисплея) на дисплеї аби вивести у DDRAM пам’ять, щоб побачити утворений нами символ.

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Set CG RAM address	0	0	0	1	ACG					

Рисунок 1.3 – Присвоєння CGRAM адреси для утвореного символа.

Висновки до розділу 1

Підсумовуючи розділ 1 можна зробити висновок, що контролер Hitachi HD44780 - контролер на базі рідкокристалічного LCD дисплея, розроблений компанією Hitachi у 1980-х роках. На світовому ринку є подібні аналоги, які нічим не відрізняються від оригінала, який був виготовлений японською компанією.

На корпусі LCD1602 розміщаються 16 пінів, які використовуються для керування дисплеєм. Для передачі даних до дисплея використовується 4-бітний або 8-бітний режим. Плюси та мінуси використання режимів наведено нижче:

- Плюси:

  8-бітний режим: Під час написання функції запису та передачі даних або команди, ми до дисплею від разу передаємо 8-біт інформації без розділу на півбайта та зсуву його байтів.

  4-бітний режим: Зручність підключення від дисплея до мікроконтролера, що з іншого боку зменшує кількість використовуваних проводів для підключення на 4 штуки менше.

- Мінуси:

  4-бітний режим: В реалізації функції запису та передачі даних або команд є різниця, тому що потрібно передати 8-бітів, а в 4-бітному режиму, спочатку потрібно зсунути окрему кількість біт до старших.

  8-бітний режим: Під час збирання та налаштування схеми разом з макетною платою та мікроконтролером може знадобитися на 4 піни більше.

Дисплей LCD1602 має в собі вже закладені на програмному рівні символи латинського алфавіту, цифри, японські символи алфавіту Катакана та символи грецького алфавіту й деякі по типу кома, скобка та інші вбудовані символи. Сам дисплей має можливість створювати власні та зберігати у CGRAM пам’яті символи. Одночасно виводити та створювати може 8 власних символів. Це є обмеженням для дисплея такого типу.

РОЗДІЛ 2
МЕТОД КЕРУВАННЯ ТА ЙОГО РЕАЛІЗАЦІЯ

2.1 Підготовка до роботи

2.1.1 Програмна модель

В першу чергу, як почати роботу із налаштування схеми й написання програми для керування рідкокристалічним дисплеєм LCD1602 на базі контролера HD44780. Потрібно визначитись як буде візуально виглядати програмна модель для використання її в подальшому процесі розробки програми. Реалізована програмна модель має вигляд блокової схеми, що відображає зв'язок між функціями програмної моделі (див рис 2.1).

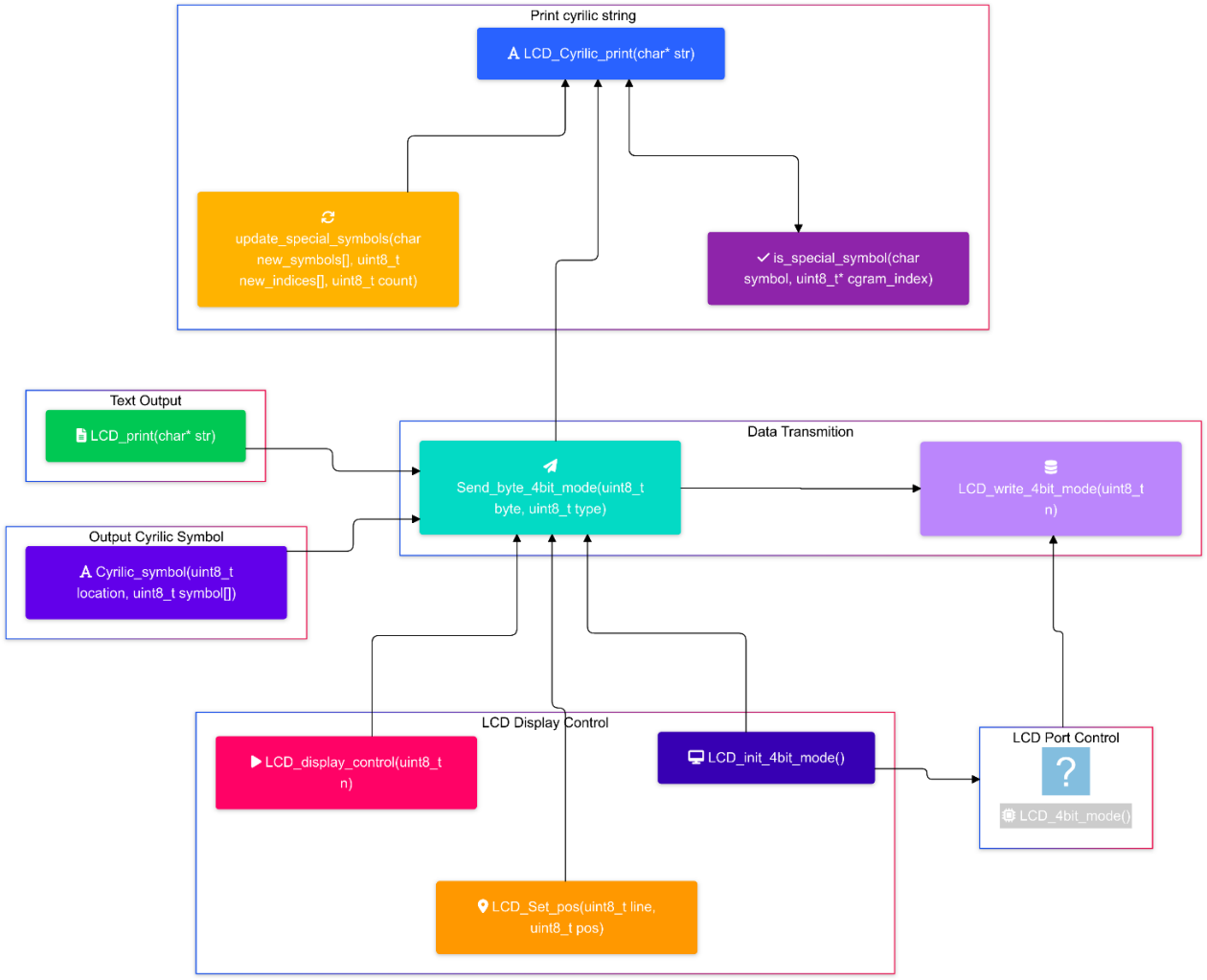

Рисунок 2.1 – Блокова схема програмної моделі.

Із схеми можна побачити виділені області в яких знаходяться ті функції, що будуть використовуватись для реалізації програмної моделі. У мене на схемі є такі області:

1. Data transmition на схемі відповідає за запис та передачу даних у дисплеї, тому мають значення такі функції LCD_write_4bit_mode(uint8_t n), Send_byte_4bit_mode(uint8_t byte, uint8_t type) що реалізують режим запису в дисплей, передаючи до нього дані;
2. LCD display control. Ця ділянка схеми відповідає за управління дисплеєм.Для цієї області мають значення такі функції: LCD_init_4bit_mode() – ініціалізація дисплея, LCD_display_control(uint8_t n) – включення/вимкнення дисплея, LCD_Set_pos(uint8_t line, uint8_t pos) Виставлення курсора у довільну позицію на дисплеї;
3. LCD Port ця область призначена для зберігання пінів, що використовуються для підключення від дисплея до мікроконтролера ATmega2560. Функція LCD_4bit_mode(), що знаходиться в цій виділеній області схеми зберігає в собі піни для використання 4-бітного режиму передачі даних;
4. Text output. Ця область на схемі програмної моделі відповідає за вивід рядку на дисплей, тому значення має ця функція LCD_print(char *str), що призначена для неї;
5. Output cyrilic symbol на схемі має застосування для виводу символів кирилиці, що не вбудовані у дисплей LCD1602. За це відповідає функція Cyrilic_symbol(uint8_t location, uint8_t symbol[]), що прив’язана до неї.
6. Print cyrilic string відповідає за вирішення головної проблеми, яка описана у меті роботи (див. Вступ), а саме виведення символів кирилиці на дисплей.

   Для цього використовуються такі функції: update_special_symbols(char new_symbols[], uint8_t new_indices[], uint8_t count),is_special_symbol(char symbol, uint8_t *cgram_index), LCD_Cyrilic_print(char *str).

Наявні стрілочки, що проведені від однієї функції до іншої між областями на схемі, свідчить про зв’язок та застосування функції під час її реалізації.

2.1.2 Процес підключення схеми

Насамперед перед роботою із апаратним обладнанням проводять такі підготовчі заходи:

1. Зібрати та правильно налаштувати схему із дотриманням правил безпеки з електронними приладами та мікросхем, аби під час користуванням LCD1602 не було ніяких неприємних ситуацій;
2. Обов’язково потрібно перевірити на наявність оголених проводів, якщо такі є потрібно їх замінити на інші;
3. Міцно втиснути елементи у макетну плату;
4. Для світлодіодів потрібно обов’язково встановлювати резистор для зменшення кількості струму що він буде пропускати через себе, при цьому збільшуючи опір струму, який туди потрапляє у нього;
5. Перевірити на справність мікроконтролера.

Апаратна схема з якою я буду працювати під час виконання дипломної бакалаврської роботи складається з даних таких елементів, які придбав.

1. Мікроконтролер ATMega2560. Для підключення мікроконтролера мною були використані такі піни [5] (див. рис. 2.2):
    - 5V, GND(GROUND) для надання живлення мікроконтролеру ATMega2560;
    - Піни D2, D3(PE4, PE5) використовуються для керування регістрами RS, E на дисплеї LCD1602;
    - Піни даних D4,D5,D6,D7(PG5, PE3, PH3, PH4) експлуатуються для застосування 4-бітного режиму на дисплеї керуючи даними для виводу на дисплей;
2. LCD1602 на базі контролера HD44780. Під час підключення його я застосував такі регістри:
    - VSS, VDD для подачі живлення на нього, тому VSS підключено до GROUND, а VDD до POWER на макетній платі;

- V0 для керування контрастністю дисплея. Цей регістр був підключений на середній вивід потенціометра;
- Регістри RS, E застосовуються для підключення від дисплея до пінів D2, D3(PE4, PE5) мікроконтролера ATMega2560;
- Регістри DB4-DB7, що підключені до пінів ATMega2560 використовуються D4,D5,D6,D7(PG5, PE3, PH3, PH4) для роботи із 4-бітним режимом передачі даних;
- Регістр A підключений до POWER, а K – до GROUND на макетній платі надаючи живлення підсвічування дисплея;

3. Змінний резистор B10K підключений до макетної плати на одному рівні із LCD1602 на 2 крайніх ніжках підключено GROUND й POWER, а до середнього виводу резистора під'єднано провід для керування контрастністю.

   Керуючи ручкою потенціометра ми регулюємо подачу струму збільшуючи чи зменшуючи контрастність дисплею
4. Проводи ПАПА-ПАПА застосовувались для підключення регістрів дисплея й змінного регістра на макетній платі, підключення до пінів від дисплея до мікроконтролера, й підключення живлення дисплея та ATMega2560;

Схему у зібраному вигляді можна побачити нижче на рис 2.2 із підключенням елементів, що описав вище.

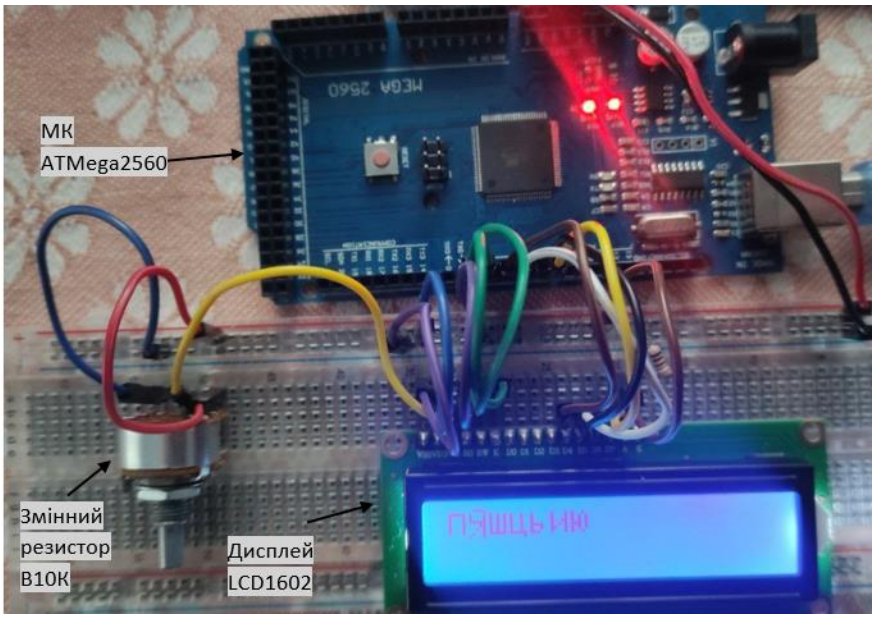

Рисунок 2.2 – Зібрана схема для роботи.

Але на цьому зображенні не зовсім зрозуміло що до чого підключено. Я зібрав монтажну схему підключення у додатку Wokwi для більш чіткого ілюстративного розуміння, що й куди підключається на цій схемі.

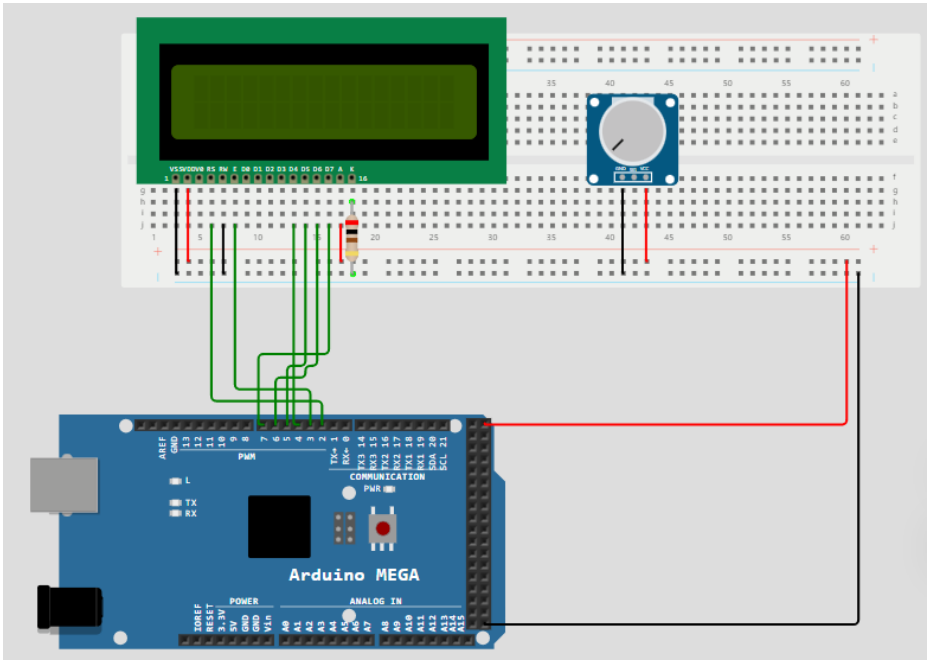

Рисунок 2.3 – Монтажна схема у Wokwi.

Налаштувавши монтажну схему у цьому застосунку для збирання електронних схем, можна детально побачити, куди який провід підключений до мікроконтролера від макетної плати.

2.2 Опис програмної моделі керування LCD1602

Функції що були реалізовані, для LCD1602 на базі контролера HD44780 складається з таких функцій, які необхідні для керування ним. Їхній перелік та дії, які виконують під час роботи та функціонування із ним наведено нижче у таблиці 2.1.

Таблиця 2.1

Перелік функції у програмній моделі

№	Назва функції	Значення
1	void LCD_4bit_mode()	Функція ініціалізує відповідні піни(DB4-DB7), що підключені від дисплея LCD1602 до мікроконтролера для використання 4-бітного режиму передачі даних до нього.
2	void LCD_write_4bit_mode(uint8_t n)	Функція використовується у якості режиму запису під час ініціалізації роботи дисплея відправляючи певні біти даних (DB4-DB7) на дисплей, підтверджуючи конфігурацію, використання 4-бітного режим передачі даних.
3	void Send_byte_4bit_mode(uint8_t data, uint8_t type)	Функція призначена для відправки даних або команд на дисплей, і є частиною реалізації режиму запису.
4	void LCD_init_4bit_mode()	Функція виконує дію ініціалізації роботи дисплею при використанні 4-бітного режиму.
5	void LCD_print(char *str)	Функція реалізує запис та вивід строки тексту у дисплей.
6	void LCD_display_control(uint8_t n)	Функція керує дисплеєм включаючи та вимикаючи його.
7	void LCD_Set_pos(uint8_t line, uint8_t pos)	Функція дозволяє виставляти курсор у певній позиції на дисплеї згідно із адресом комірки на ньому.
8	void Cyrilic_symbol(uint8_t location, uint8_t symbol[])	Функція має можливість створювати свої власні та спеціальні символи кирилиці, із вказанням комірки у CGRAM пам’яті (0 - 7), де він буде зберігатись.
9	void update_special_symbols(char new_symbols[], uint8_t new_indices[], uint8_t count)	Функція, що оновлює масив спеціальних, для запису нових
10	uint8_t is_special_symbol(char symbol, uint8_t *cgram_index)	Функція створена для перевірки, чи є символ спеціальним
11	void LCD_Cyrilic_print(char *str)	Функція призначена для виводу строки тексту кирилиці разом із символами латиниці на дисплеї.

Для написання функцій керування LCD1602 на базі контролера HD44780, мною було прийнято рішення використовувати мову програмування C [6,7]. Вона використовується для написання програм, керуючи мікроконтролером та іншими пристроями на машинному рівні, тобто низькому рівні абстракції. Детальний опис та реалізацію функцій буде розглядатись у подальших підрозділах.

2.2.1 Функція LCD_4bit_mode()

Перед початком реалізації першої функції, а саме LCD_4bit_mode(). На початку коду(див. Додаток Г) були ініціалізовані деякі значення за замовчуванням, що будуть використовуватись під час написання інших функцій.

У мене ініціалізованими значеннями за замовчуванням є піни LCD1602 RS(Register Select) і E(Enable) для них прописані макроси PORT, що дозволяють керувати станами цих пінів. Ці піни знадобляться під час реалізації функцій запису та відправки команди або даних.

Функція void LCD_4bit_mode() - це функція без параметрів, де в тіло функції записуються піни RS, E та DB4-DB7, що підключаються від рідкокристалічного дисплею до мікроконтролера Arduino Mega 2560. Реалізована функція void LCD_4bit_mode() наявна в Додатку Г.

Під час роботи із дисплеєм ми будемо застосовувати 4-бітний режим передачі даних, адже ми використовуємо піни на дисплеї DB4-DB7.

2.2.2 Функція LCD_write_4bit_mode(uint8_t n)

Функція void LCD_write_4bit_mode(uint8_t n) використовується для реалізації режиму запису даних 4-бітного режиму у дисплей. Параметр uint8_t n, без знакове цілочисельне число типу int у 8 бітному форматі, що записується у біти регістрів даних DB4-DB7.

Для її реалізації я користувався даташитом, де була подібна на кшталт схема, яка побудована з використанням сигналів логічних рівнів, які використовується для керування рідкокристалічним дисплеєм у цьому режимі. Схема реалізації режиму запису є нижче на рис 2.4.

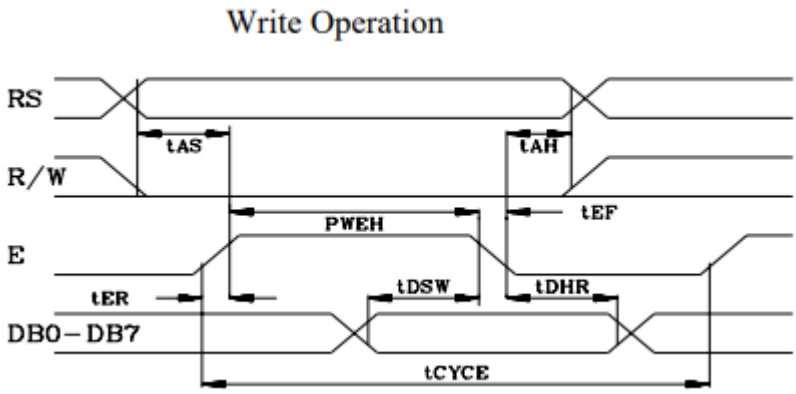

Рисунок 2.4 – Схема режиму запису.

Згідно із схеми мені для побудови функції знадобяться такі піни як - RS, R/W, E, DB4-DB7. Структура даної функції представлена у Додатку Г.

Опис функції:

1. Спочатку нам потрібно очистити регістри даних DB4-DB7, тобто виставити значення цих пінів у стан логічного нуля за допомогою макросу PORTX;
2. Далі потрібно перевірити стан кожного з портів DB4–DB7, які з’єднані з мікроконтролером. Для цього, використовуючи шістнадцяткові маски та параметр n (8-бітне число), застосовуємо логічну операцію «І», щоб визначити, чи біт дорівнює 1. Якщо так — відповідний пін встановлюємо в логічну одиницю. Таку перевірку виконуємо для кожного піна.

   Наприклад: функція прийняла значення 3, у двійковому форматі це буде 0011. І це значення ми подаємо для перевірки умови:

   Перший біт(DB4): 0001 & 0001(0x01) -> 0001, виставляємо біт у логічний стан одиниці;

   Другий біт (DB5): 0010 & 0010(0x02) -> 0010, виставляємо біт у логічний стан одиниці;

   Третій біт (DB6): 0000 & 0100(0x04) -> 0000, виставляємо біт у логічний стан нуля;

   Четвертий біт (DB7) : 0000 & 1000(0x08) -> 0000, виставляємо біт у логічний стан нуля

3. Наступним кроком, є генерація імпульсів регістру E(Enable). Виставивши його у стан, логічної одиниці ми подаємо сигнал дисплею, що дані готуються до прийому та зробили затримку. Подавши стан логічного нуля, ми даємо дисплею знати, що ми завершили відправку даних.

2.2.3 Функція Send_byte_4bit_mode(uint8_t data, uint8_t type)

Функція void Send_byte_4bit_mode(uint8_t data, uint8_t type) є продовженням реалізації режиму запису у дисплеї. Вона має 2 параметри, які приймає під час виклику функції.

Параметр Uint8_t data приймає без знакове цілочисельне число типу int у 8 бітному форматі, а uint8_t type приймає значення 0 або 1 взалежності, що ми будемо відправляти до дисплея команду або дані. Функція представлена в Додатку Г.

Опис функції:

1. Напочатку функції ми перевіряємо умову відправки до дисплея. Керуючи станом регістру RS(Register Select) ми можемо визначити, що будемо відправляти. Якщо RS дорівнює 1, то значить будемо відправляти дані, якщо RS дорівнює нулю – команду відправляємо до дисплею;
2. Окрім цього дисплей повинен приймати цілий байт, а в 4-бітному режимі передачі даних одразу зробити неможливо. Тому спочатку за допомогою функції void LCD_write_4bit_mode(uint8_t n), передаємо старші 4 біти (зсунувши байт вправо на 4 біти), а потім окремо передаємо молодші 4 біти на регістри DB4–DB7, що використовуються в цьому режимі.

2.2.4 Функція LCD_init_4bit_mode()

Функція LCD_init_4bit_mode(),функція яка не приймає ніяких параметрів, а використовується виключно в цілях ініціалізації рідкокристалічного дисплею LCD1602 на базі контролера HD44780. Структура реалізованої функції продемонстровано в Додатку Г .

Для реалізації методу ініціалізації дисплея додатково використовуються функції LCD_write_4bit_mode(uint8_t n) та Send_byte_4bit_mode(uint8_t byte, uint8_t type) згідно із даташиту. LCD_write_4bit_mode(uint8_t n) викликається в

ції функції для підтвердження того, що ми саме будемо використовувати 4-бітний режим роботи передачі даних до дисплею.

A Send_byte_4bit_mode(uint8_t byte, uint8_t type) застосовується для початкових налаштувань в ньому наприклад, яким режимом користуватися під час роботи із ним, скільки строк на дисплеї буде застосовувати та інші.

На рис 2.5 нижче можна додатково побачити як співвідноситься схема реалізації ініціалізації дисплея з даташиту та код функції LCD_init_4bit_mode() для покрокового розуміння, як вона реалізована.

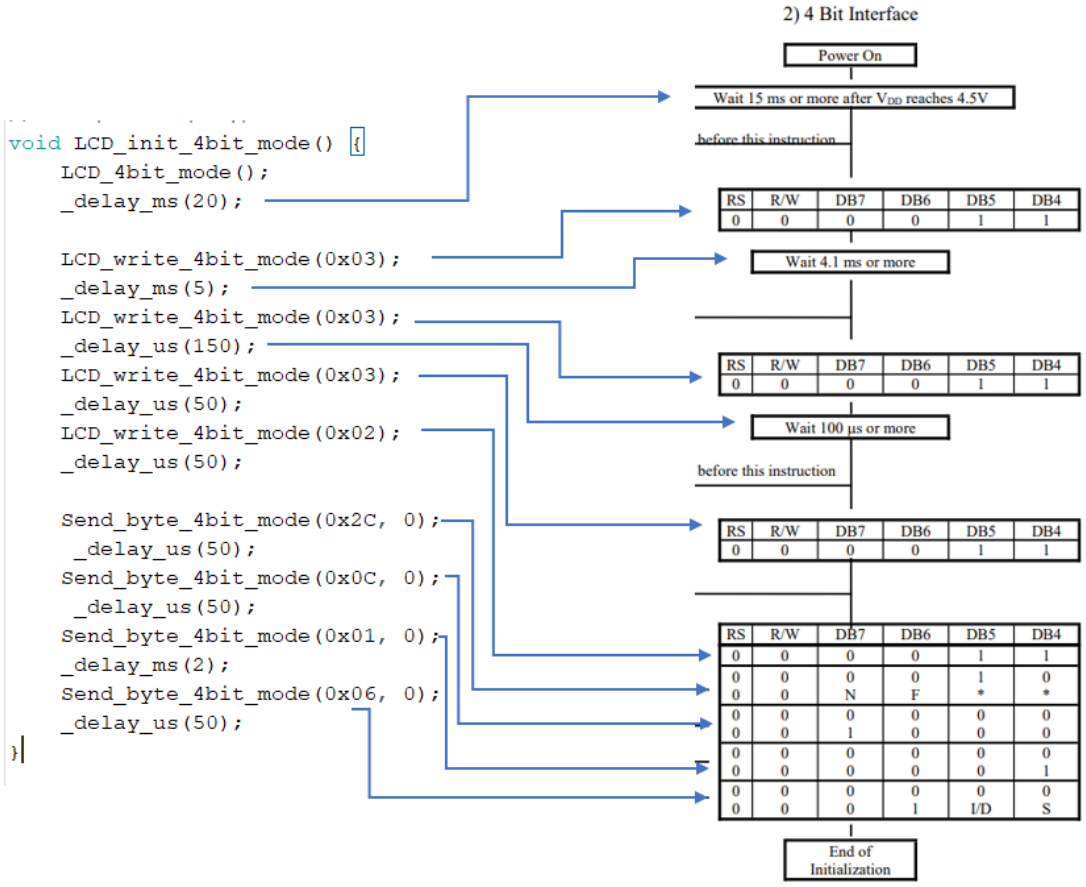

Рисунок 2.5 – Покрокова реалізація функції LCD_init_4bit_mode().

2.2.5 Функція LCD_print(char *str)

Функція void LCD_print(char *str) призначена для виводу строк тексту на дисплеї. Вона приймає параметр типу char, що посилається на масив строк str.

Вигляд реалізованої функції є у Додатку Г.

Опис функції:

1. Напочатку функції ініціалізуємо без знакову цілочисельну змінну «c» типу int у 8 бітному форматі, присвоївши значення нулю. Ця змінна буде використовуватись у масиві як індекс для послідовного доступу кожного символу рядка;
2. У циклі while за допомогою умови if перевіряємо, чи передано символ на дисплей. Якщо ні - продовжуємо перебувати в циклі, передаючи його через функцію Send_byte_4bit_mode(str[c], 1) і переходимо до наступного символу (збільшуючи c). Коли доходимо до нульового байта (термінального нуля) цикл завершується.

2.2.6 Функція LCD_display_control(uint8_t n)

Функція void LCD_display_control(uint8_t n) призначена для станом дисплея ввімкнення та вимкнення його за допомогою параметра «n», що приймає значення 0 або 1. Структура функції представлена в Додатку Г.

Якщо функція void LCD_display_control(uint8_t n) приймає значення 1, тоді ми до дисплея відправляємо команду вмикання дисплея Send_byte_4bit_mode(0x0C, 0), яка вбудованою у контролер. А якщо параметр «n» дорівнює 0, тоді виконується та відправляється команда Send_byte_4bit_mode(0x08, 0) до дисплея, що вимикає його.

2.2.7 Функція LCD_Set_pos(uint8_t line, uint8_t pos)

Функція void LCD_Set_pos(uint8_t line, uint8_t pos) використовується для встановлення курсора у довільному місці на дисплеї, якщо він сам включений. Параметр функції «line» відповідає за те, на якому рядку ми встановимо курсор, приймаючи значення 0 або 1. Тоді «pos» застосовується для встановлення його(курсора) на іншому місці рядка. Реалізована функція наявна в Додатку Г.

Опис функції:

Для встановлення курсора у певний рядок та місце ми будемо визначати його передаючи адрес комірки до дисплею. Ми скористаємося такою формулою address = (line*0x40 + pos)|0x80, де у нас line- рядок дисплея (значення 0 або 1) та pos – позиція на рядку. Пояснити та довести цю формулу можна таким чином.

На дисплеї перший елемент адреси першого рядка має адресу 0x00(0 в десятковому вигляді), тоді на другому рядку він вже буде мати іншу адресу 0x40 (64 в десятковому вигляді), тож можна зробити припущення, що рядках адреса між ними відрізняється на 0x40.

А позиція на рядку нічим не змінна, вона відповідає кількості символів на одній строці дисплея від 0 до 15 Тоді треба рядок помножити на 0x40 та додати значення позиції (0 - 15). Окремо для цього треба встановити маску 0x80 у старший біт логічним оператором АБО, що буде передавати адресу у DDRAM пам'ять.

Адже адреса 0x80 (1000 0000 - у двійковому форматі, де в 1 виставлене значення регістр даних DB7) є маскою у шістнадцятковому вигляді, що використовується як команда для встановлення адреси у DDRAM пам’яті, куди відправляється значення адреси ADD (Address Data Display) до дисплея.

Вигляд та опис команди можна переглянути в таблиці 1.2 та з даташиту [4] у секції 13 Instruction Set. Частково команда з даташиту має такий вигляд на рис 2.6 нижче.

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DESCRIPTION
Set DD RAM address	0	0	1	ADD							Set DD RAM address. DD RAM data is sent and received after this setting

Рисунок 2.6 - Фрагмент з даташиту.

Як застосовується встановлення курсора представлено на прикладі нижче на рис 2.7.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
Line 2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Рисунок 2.7 – Приклад застосування.

2.2.8 Функція Cyrilic_symbol(uint8_t location, uint8_t symbol[])

Функція void Cyrilic_symbol(uint8_t location, uint8_t symbol[]) застосовується для створення своїх власних символів Параметр «location» вказує на те, де буде та в якій комірці (від 0 – 7) буде знаходитись символ у CGRAM пам’яті, а «symbol[]» приймає до функції масив спеціального символу.

На дисплеї одночасно зберігати та виводити 8 спеціальних символів, що є недоліком для LCD1602. Структурний вигляд функції продемонстровано в Додатку Г.

Опис функції:

1. Спочатку виділяємо місце для зберігання кастомних символів у CGRAM пам’яті від 0 до 7, адже ми можемо зберігати та одночасно виводити у DDRAM пам’ять дисплея (відеопам’ять) тільки 8 символів.
2. Команда Send_byte_4bit_mode(0x40 | (location << 3), 0) встановлює адресу символу у CGRAM пам’ять дисплея. Маска 0x40 (0100 0000 у двійковому вигляді) вказує, що ми звертаємось до CGRAM для запису спеціальних символів. Докладніше цю команду описано в таблиці 1.2 та в розділі 13 із даташиту [4], де показано на рисунку 2.8

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DESCRIPTION
Set CG RAM address	0	0	0	1	ACG						Set CG RAM address. CG RAM data is sent and received after this setting.

Рисунок 2.8 – Фрагмент із даташиту.

Змінну «location» зсуваємо на 3 біти вліво, для визначення початкової адреси для запису у CGRAM. Використовуючи логічне побітове «АБО» визначаємо остаточну адресу у ній. Приклад використання команди є нижче на рис 2.9.

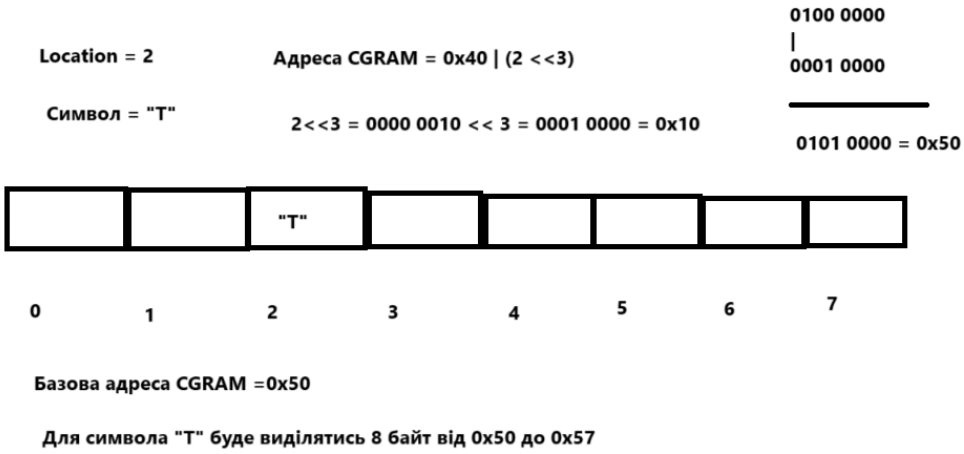

Рисунок 2.9 – Приклад застосування.

Після визначення фактичного місцеположення символу у CGRAM, для кожного символу буде виділятись по 8 байт.

3. Далі ми проходимось у циклі по всьому масиву із 8 байт символу і відправляємо у вигляді команди для виводу даних на дисплей.

2.2.9 Функція update_special_symbols(char new_symbols[], uint8_t new_indices[], uint8_t count)

Функція void update_special_symbols(char new_symbols[], uint8_t new_indices[], uint8_t count) призначена для оновлення символів, що виводяться на дисплеї, зберігаючи їх у CGRAM пам’яті дисплея.

Параметр char new_symbols[] приймає в себе масив рядків, що застосовується для додавання або оновлення символів. uint8_t new_indices[] у якості параметра одержує значення індексів CGRAM, що відповідає спеціальним символам, що зберігаються в цьому масиві.

А параметр uint8_t count приймає значення для функції, що підраховує кількість елементів, у масивах new_symbols та new_indices.Спочатку перед тим як реалізовувати функцію оновлення символа на дисплеї, треба ввести додаткові змінні:

- char special_symbols[8] – масив для зберігання спеціальних символів;
- uint8_t cgram_indices[8] – масив для зберігання індексів символів у CGRAM символів;

- uint8_t special_symbols_count, змінна, що зберігає кількість спеціальних символів;

Структура функції представлена в Додатку Г.

Опис функції:

1. Використовуючи цикл ми проходимось по масиву special_symbols, що ініціалізували вище для зберігання символів та масиву new_symbols, що приймає значення у якості параметра. Ми оновлюємо символи на інші у CGRAM пам’яті дисплея, порівнюючи їх;
2. Так само проходимось по масивам cgram_indices й new_indices оновлюючи індекси у CGRAM пам’яті, де зберігаються символи. Тим само змінюючи кількість та оновлюючи символи змінною special_symbols_count та count порівнюючи їх кількість;

2.2.10 Функція is_special_symbol(char symbol, uint8_t *cgram_index)

Функція uint8_t is_special_symbol(char symbol, uint8_t *cgram_index) застосовується для перевірки створеного мною символа у CGRAM пам’яті, на те, що він є спеціальним. Параметр char symbol приймає для функції символ, який потрібно перевірити на наявність його у комірці CGRAM пам’яті. А uint8_t *cgram_index є вказівником на індекс спеціального символа, що знаходиться в ній (в CGRAM пам’яті). Структурно функція представлена в Додатку Г.

Опис функції:

1. Ми проходимось в циклі по масиву special_symbols[i], використовуючи змінну special_symbols_count, що ми ініціалізували раніше, для межі ітерації в циклі та визначення кількості елементів в масиві.
2. Далі перевіряємо умову використовуючи масив special_symbols[i],який ми ініцілізували раніше, для зберігання спеціальних символів та змінну symbol, що є параметром функції для порівняння символа, що передаємо та символа, що є в масиві.

Якщо символ знайдено в масиві, тоді записуємо індекс цього створеного символу у CGRAM пам’яті та повертаємо значення «1», а якщо не знайдено, - повертаємо «0».

2.2.11 Функція LCD_Cyrilic_print(char *str)

Функція void LCD_Cyrilic_print(char *str) використовується для виводу рядку тексту символів кирилиці разом із латиницею ASCII-символів, що прописано у меті дипломної роботи. Параметр char *str приймає рядок, який будемо виводити на рідкокристалічний дисплей. Реалізована функція наявна в Додатку Г.

Опис функції:

1. Ми проходимось в циклі по кожному символу, що ми будемо передавати, доки не буде досягнуто термінального нуля.
2. Далі ми перевіряємо умову, якщо цей символ є спеціальним у CGRAM пам’яті, використовуючи функцію uint8_t is_special_symbol(char symbol, uint8_t *cgram_index) яку ми написали раніше. Якщо цей символ знайдено, ми відправляємо його до дисплея у якості даних.
3. Також нам треба перевірити, чи переданий символ не є символом ASCII, що є вшитим у контролер HD44780. Якщо умова виконана, ми відправляємо його до дисплея у якості даних. А якщо ні, то ми ігноруємо невідомий символ.

Висновки до розділу 2

Підсумовуючи розділ 2, для роботи із апараним обладнанням я використав Arduino Mega 2560, рідкокристалічний дисплей LCD1602 на базі контролера HD44780, макетна плата, потенціометр, та проводи типу ПАПА-ПАПА. Програмна модель була написана на мові програмуванні С, для керування можливостями дисплея LCD1602 на машинному рівні.

Для управління дисплеєм програмна модель складається із таких функцій:

- void LCD_4bit_mode() - Функція ініціалізує відповідні піни(DB4-DB7), що підключені від дисплея LCD1602 до мікроконтролера для використання 4-бітного режиму передачі даних до нього.

- void LCD_write_4bit_mode(uint8_t n) - Функція використовується у якості режиму запису під час ініціалізації роботи дисплея відправляючи певні біти даних (DB4-DB7) на дисплей, підтверджуючи конфігурацію, використання 4-бітного режим передачі даних.
- void Send_byte_4bit_mode(uint8_t byte, uint8_t type) - Функція призначена для відправки даних або команд на дисплей, і є частиною реалізації режиму запису.
- void LCD_init_4bit_mode() - Функція виконує дію ініціалізації роботи дисплею при використанні 4-бітного режиму.
- void LCD_print(char *str) - Функція реалізує запис та вивід строки тексту у дисплей.
- void LCD_display_control(uint8_t n) - Функція керує дисплеєм включаючи та вимикаючи його.
- void LCD_Set_pos(uint8_t line, uint8_t pos) - Функція дозволяє виставляти курсор у певній позиції на дисплеї згідно із адресом комірки на ньому.
- void Cyrilic_symbol(uint8_t location, uint8_t symbol[]) - Функція має можливість створювати свої власні та спеціальні символи кирилиці, із вказанням комірки у CGRAM пам’яті (0 - 7), де він буде зберігатись.
- void update_special_symbols(char new_symbols[], uint8_t new_indices[], uint8_t count) - Функція, що оновлює масив спеціальних, для запису нових
- uint8_t is_special_symbol(char symbol, uint8_t *cgram_index) - Функція створена для перевірки, чи є символ спеціальним.
- void LCD_Cyrilic_print(char *str) - Функція призначена для виводу строки тексту кирилиці разом із символами латиниці на дисплеї.

Під час перевірки результатів виконання функцій, що взаємодіють із дисплеєм, а саме перевіряючи функцію, що виводить символи кирилиці із латиницею, ніяких помилок виявлено не було, тому можна вважати, що програмна модель працює коректно та правильно.

РОЗДІЛ 3
ТЕСТУВАННЯ ТА РЕКОМАНДАЦІЇ ЩОДО ВИКОРИСТАННЯ

3.1 Тестування можливостей програми в Proteus,Wokwi та на апаратному обладнанні

Тестування створеної мною програмної моделі для рідкокристалічного дисплея LCD1602 на базі контролера HD44780 будуть відбуватись у програмному середовищі для симулювання Proteus, у онлайн-додатку симуляторі Wokwi та на апаратному обладнанні для перевірки правильності роботи програмної моделі.

3.1.1 Підготовка до тестування в Proteus

Proteus – це пакет програм типу САПР(Системного автоматизованого проектування та розрахунку) для конструювання та проектування автоматизованих схем [8]. В ній також передбачена робота із мікроконтролерами Arduino, STM32 та інших мікроконтролерів за допомогою вже готових вбудованих бібліотек.

Насамперед перед початком тестування нам необхідно створити проект, де воно буде виконуватись.

Із вбудованих бібліотек програмного середовища Proteus нам необхідно взяти та підключити такі компоненти:

1. ATMEGA2560 - мікроконтролер, де буде виконуватись програма керуванням рідкокристалічним дисплеєм;
2. LM016L – прототип рідкокристалічного дисплея LCD1602 на базі контролера HD44780, яким буде відбуватись керування у тестуванні;
3. POT-HG – змінний резистор, що буде регулювати контрастність на ньому.

Тепер треба, усі потрібні нам компоненти підключити до разом. Спочатку на дисплеї регістр VSS під’єднаємо до GROUND (земля), а VDD до POWER (Живлення).

Далі регістр RW, що відповідає за режим запису/читання треба підключити до GROUND, тобто на дисплеї буде застосований режим запису.

Тепер регістри RS(Register Select) і E(Enable) підключити до пінів Input/Output мікроконтролера ATMEGA2560 D2, D3, у специфікації Arduino вони мають назву PE4, PE5. Для використання 4-бітного режиму регістри DB4-DB7 одразу підключаємо до пінів мікроконтролера D4, D5, D6, D7 (PG5, PE3, PH3, PH4 – назви у специфікації Arduino).

Змінний резистор або потенціометр застосовуємо для керуванням контрастністю дисплея і треба підключити дві крайні ніжки або GROUND або до POWER, різниці майже ніякої немає, а середню підімкнути до регістру VEE. Підключена схема разом з усіма компонентами виглядає таким чином на рис 3.1 нижче.

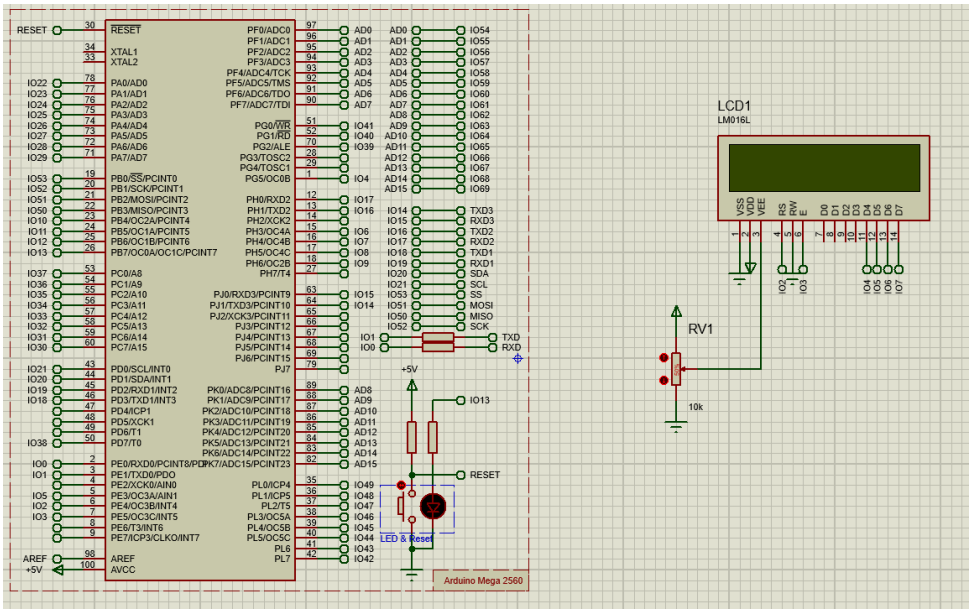

Рисунок 3.1 – Підключена схема в середовищі Proteus.

3.1.2 Підготовка до тестування в Wokwi

Wokwi – це онлайн-додаток для симуляції роботи різних моделей мікроконтролерів таких як: Arduino, ESP32, Raspberry Pi Pico та інших типів мікроконтролерного обладнання. Ця платформа дозволяє користувачам симулювати різні пристрої для розробки своїх проектів. Для початку роботи користувачу на цьому ресурсі достатньо просто зареєструватись на цьому сайті [9]. І перше, що нас зустрічає на ньому це вибір моделі мікроконтролера для розробки проекту.

Рисунок 3.2 – Вибір мікроконтролера для розробки на Wokwi.

Припустимо ми обрали Arduino, тоді наступним кроком потрібно обрати модель мікроконтролера з яким будемо працювати під час реалізації проекту.

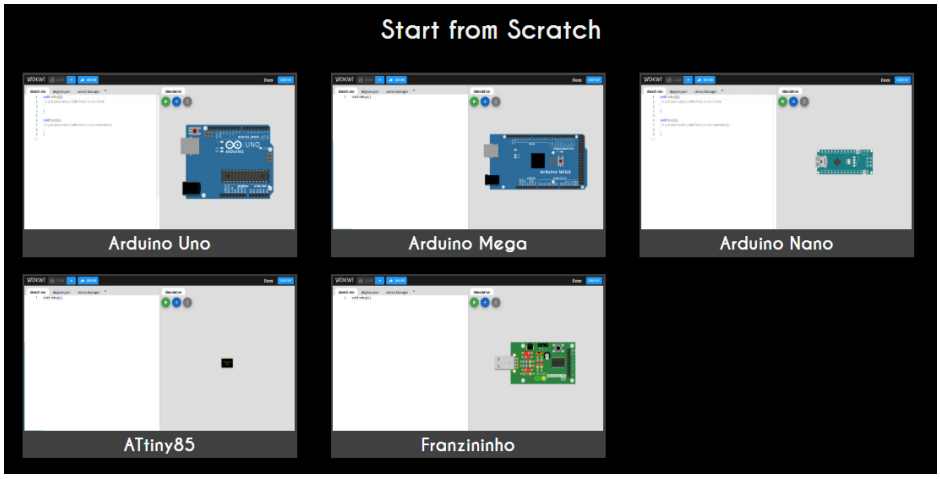

Рисунок 3.3-Вибір моделі МК Arduino для проекту.

Далі обираємо модель МК для проекту й виконуємо підключення тих компонентів, які будуть застосовуватись. В моєму випадку я працюю з мікроконтролером ATMega2560 та використовую дисплей LCD1602 для тестування програмної моделі. Елементи схеми, що використовується у моєму проекті, мають таке саме підключення , що й у проекті в Proteus (див 3.1.1).

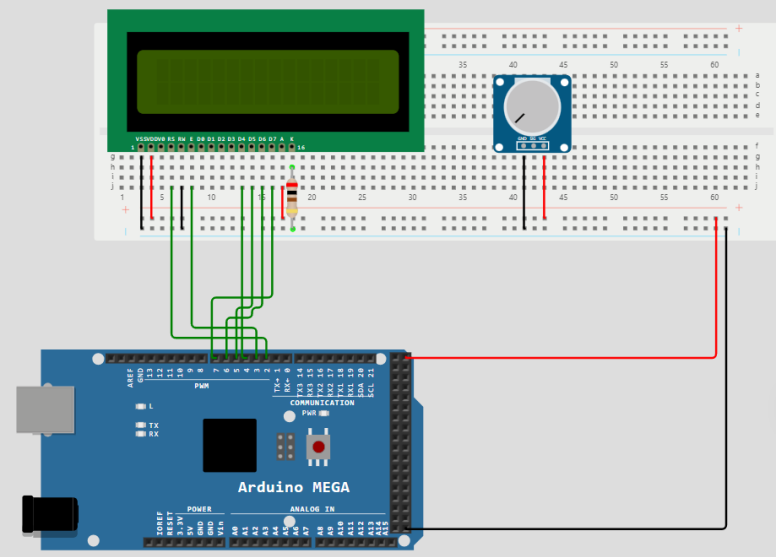

Рисунок 3.4 – Підключена схема у Wokwi.

3.2 Тестування ініціалізації та вивід інформації на дисплеї

Перед початком тестування функцій ініціалізації та виводу рядку тексту на дисплеї. Нам необхідно спочатку в середовищі розробки Arduino IDE потрібно експортувати вихідний код у скомпільований бінарний файл, щоб ми могли його використовувати будь-де зокрема й у програмному середовищі Proteus. Ось як воно виглядає на рис 3.5.

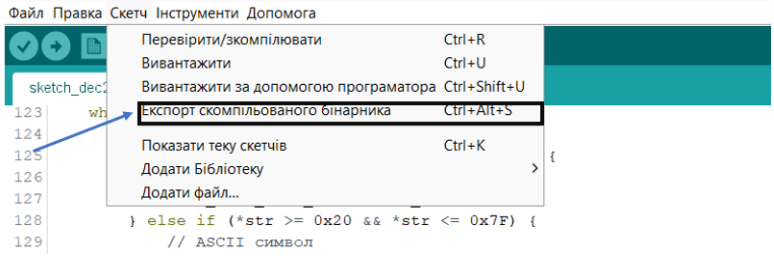

Рисунок 3.5 – Вбудована Arduino IDE функція експорт скомпільованого у бінарний файл.

Далі коли файл вже експортований, то нам необхідно у Proteus, обрати його щоб користуватись ним. Щоб це зробити необхідно натиснути на прототип ATMEGA2560, що додали із загальної бібліотеки для підключення схеми. Далі натискаємо на Program File і обираємо файл у форматі .hex.

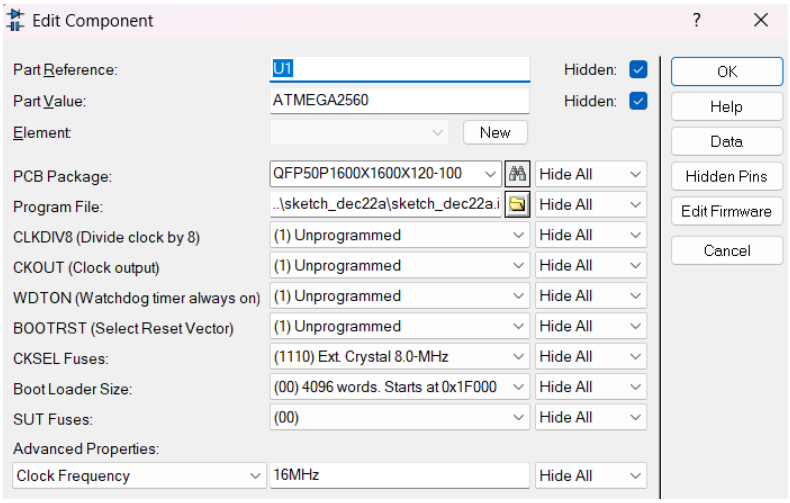

Рисунок 3.6 – Меню налаштування мікроконтролеру.

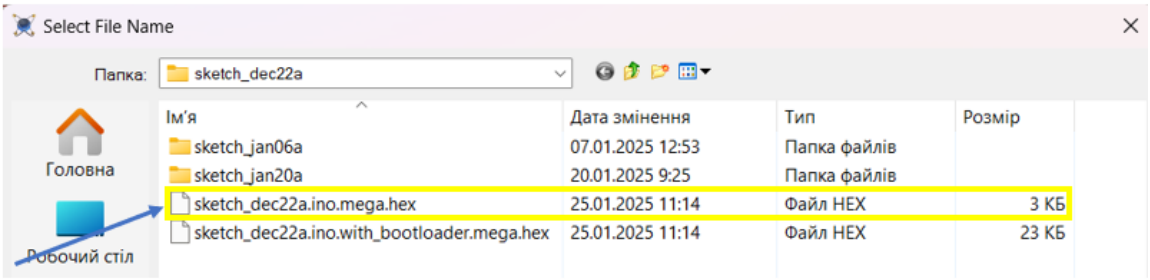

Рисунок 3.7 -Додавання файлу до ATmega2560 в середовищі Proteus.

Завантаживши експортований файл у бінарному вигляді до Proteus протестуємо функції ініціалізації та виводу рядка на дисплей.

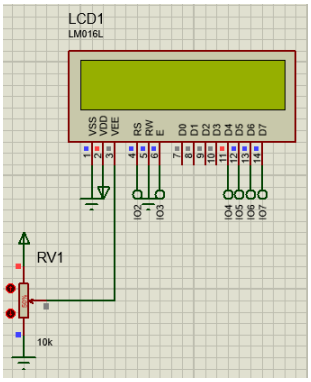

Рисунок 3.8 – Ініціалізація дисплею.

З рис 3.8 можна помітити, що дисплей загорівся та працює, тобто функція ініціалізації LCD_init_4bit_mode() функціонує правильно та коректно.

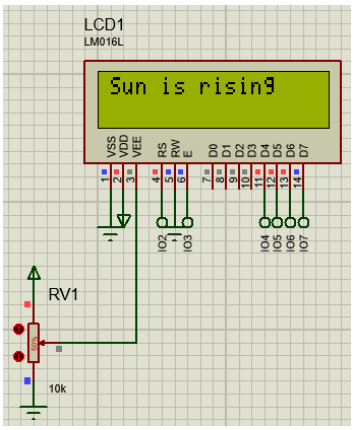

Рисунок 3.9 - Вивід рядку тексту на дисплеї.

На рис 3.9 дисплей вивів рядок тексту «Sun is rising», що можна стверджувати функція LCD_print() відпрацювала так як вона була задумана.

Далі виконаємо тестування у онлайн-ресурсі Wokwi. Для початку треба додати код програмної моделі, що розробили у Wokwi це зробити простіше, аніж у Proteus. У поле sketch.ino необхідно додати код, який буде зберігатись на час проведення тестування функцій програмної моделі.

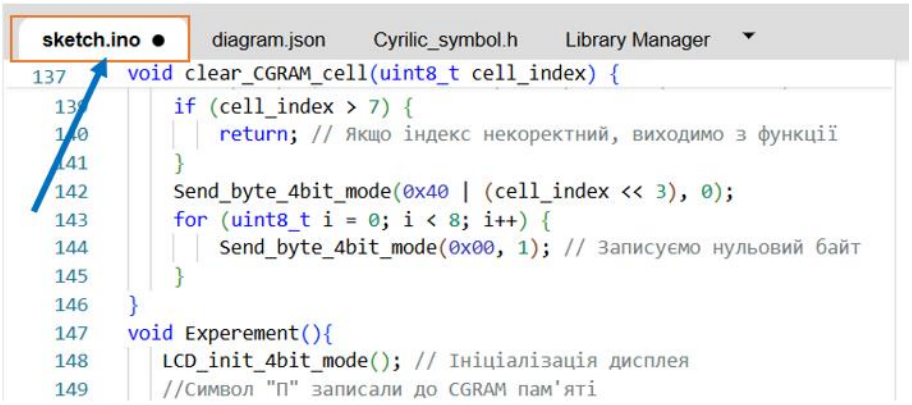

Рисунок 3.10 – Додавання коду у поле sketch.ino.

Виконаємо тестування функції ініціалізації та виводу рядку на дисплей.

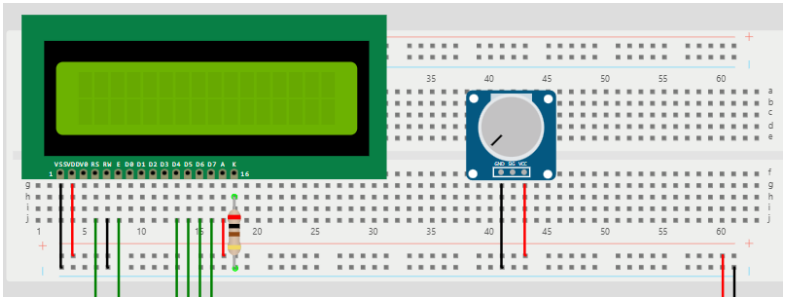

Рисунок 3.11 – Тестування функції ініціалізації.

З рис 3.11 видно, що дисплей у нас загорівся це значить, що тестування функція ініціалізації працює правильно й пройшло успішно.

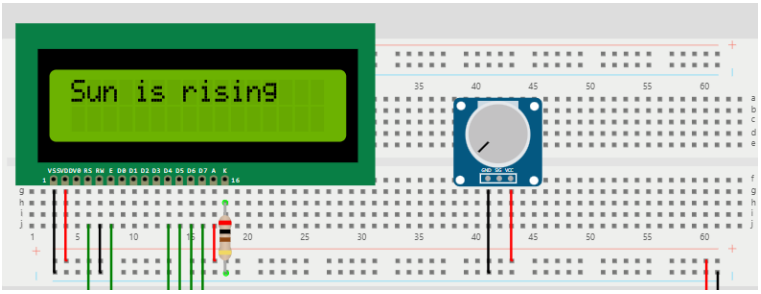

Рисунок 3.12 – Тестування функції виводу рядку.

Додавши рядок «Sun is rising» у якості параметра до функції LCD_print(), дисплей вивів його. Із результату тестування можна сказати, що функція працює коректно.

Тепер виконаємо тестування функцій ініціалізації та виводу рядка на дисплей. Для цього нам необхідно запустити код у Arduino IDE, що буде передавати його у пам’ять мікроконтролера ATmega2560, відтворюючи його на LCD1602.

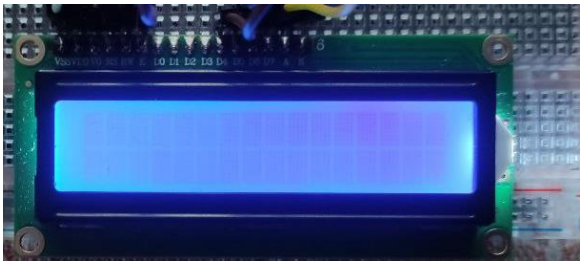

Рисунок 3.13 – Тестування ініціалізації дисплею.

З огляду на рис 3.13 можна дисплей у нас увімкнений це значить, що він пройшов ініціалізацію й функція працює коректно й правильно.

Рисунок 3.14 – Тестування функції виводу рядка.

Подавши рядок тексту «Sun is rising», він його вивів на дисплей, що свідчить проте, що функція виводу тексту на дисплей виконується правильно. Результати тестування інших функцій та скриншоти наведені у Додатку Д.

3.3 Аналіз та порівняння подібних рішень

Для аналізу та порівняння рішення проблеми виводу рядку символів кирилиці на дисплей, я знайшов у просторі мережі Інтернет подібну реалізовану методику, що вирішує дану проблему. Знайдений проект має назву LCD_1602_R_ALL (посилання на github репозиторій [10]).

3.3.1 Опис проекту

Проект LCD_1602_R_ALL розроблений для керування рідкокристалічним дисплеєм LCD1602 на базі контролера HD44780, з використанням ардуїно – сумісних плат мікроконтролерів (Наприклад Arduino Uno, Mega, Nano). Цей проект передбачає підключення дисплея на пряму до мікроконтролера й із застосуванням послідовної шини I2C для зв’язку інтегральної схеми та зчитування символів із монітора порта до нього.

Проект містить два заголовочні файли LCD_1602_R_ALL.h та font_LCD_1602_R.h. Файл LCD_1602_R_ALL.h містить код, що використовується для керування дисплеєм LCD1602 із можливістю виведення символів кирилиці українського алфавіту й працювати із різним типом підключення дисплея до мікроконтролеру – паралельно (звичайний тип підключення до мікроконтролера) й з модулем I2C. Також містяться такі функції та класи, що написані на мові програмування C++:

- Клас Symbol. Цей клас використовується для роботи із користувацькими символами кирилиці, приймаючи код символу у форматі Unicode;
- Клас LCD_1602_R. Цей клас успадковує функціонал бібліотек від LiquidCrystal_I2C або LiquidCrystal для визначення як підключено дисплей до мікроконтролера паралельно або із застосуванням модуля I2C. Також у класі містяться ряд функцій що, реалізують керування рідкокристалічним дисплеєм:

  1. Функції для роботи із текстом:
     - print(const wchar_t*), print(const char*), print(String) -Вивід рядок тексту кирилицею на дисплей;
     - printwc(wchar_t) – Перевірка символів на знак латиниці чи кирилиці;
     - ascii_win1251() та ascii_utf8() – Конвертація ASCII символів у код Unicode для застосування символів кирилиці;
  2. Функції для роботи із курсором:
     - setCursor(), getCursorCol(), getCursorRow() – зміна позиції курсора на дисплеї;
  3. Функції для роботи із спеціальними символами:
     - CharSetToLCD() – Додавання символів до CGRAM пам’яті дисплея;
     - ResetAllIndex() – скидання індексів символів під час заміни на інші, що знаходяться у CGRAM пам’яті;

Файл font_LCD_1602_R.h зберігає в собі створені символи кирилиці у вигляді масиву, що будуть використовуватись під час роботи із дисплеєм.

3.3.2 Порівняння методик виведення символів кирилиці на дисплей власного проекту та LCD_1602_R_ALL

Порівняння двох методик – власного проекту(LCD_Program) та проекту LCD_1602_R_ALL, відбувалось візуально без застосування тестування на

швидкодію методики за конкретний час, визначення складності алгоритму та інших засобів.

Адже якщо я б застосовував цей проект LCD_1602_R_ALL для проведення тестування на швидкодію, тоді для його запуску мені б скоріше знадобились інші програмні засоби для перевірки його працездатності.

Тому порівняння відбувались за такими критеріями, як – наявність кодування символів, автоматизації, кількість символів під час роботи із дисплеєм, наявність підтримки символів українського алфавіту. Узагальнене порівняння за такими критеріями записано у таблиці 3.1 нижче

Таблиця 3.1

Порівняння реалізації методик виводу символів кирилиці на дисплей.

Критерій	LCD_Program(мій проект)	LCD_1602_R_ALL
Кодування символів	Підтримка ASCII із обмеженим набором символів кирилиці через CGRAM пам’ять.	Підтримка конвертації Unicode символів разом із UTF-8 та Win-1251
Автоматизація	Ручне додавання символів у CGRAM пам’ять.	Ручне додавання символів у CGRAM пам’ять.
Кількість символів під час роботи із дисплеєм	Обмеженість використання 8 комірками пам’яті CGRAM за допомогою керування індексами.	Динамічне перевикористання пам’яті CGRAM із використання циклічного скиду
Підтримка символів алфавітів українського	Так, присутні символи українського алфавіту	Так є символи українського алфавіту

Окрім порівняння реалізації методик двох різних проектів (LCD_Program та LCD_1602_R_ALL), також було виявлено переваги та недоліки.

Переваги проекту LCD_Program:

- Використання прямого контролю підключення до мікроконтролю ATmega2560;
- Використання мінімальної кількості сторонніх бібліотек;

Недоліки проекту LCD_Program:

- Застосування ручного керування символів у CGRAM пам’яті;

- Обмежена підтримка кириличних символів без урахування конвертації для UTF-8 та Win-1251 символів;

Переваги LCD_1602_R_ALL:

- Стандартне використання функцій, що закладено у бібліотеку LCD_LIB_H для мови програмування C++;
- Підтримка застосування бібліотеки при різних типів підключення дисплею LCD1602 (паралельно та з модулем I2C);

Недоліки LCD_1602_R_ALL:

- Застосування додаткових ресурсів пам’яті для зберігання символів у CGRAM пам’яті;
- Обмеженість на кількісне використання кастомних символів(в залежності від налаштувань);

3.4 Рекомендації щодо використання LCD_Program

Програма LCD_Program може застосовуватись у невеликих та нескладних проектах з використанням Arduino - сумісних плат мікроконтролерів, онлайн-додатків та спеціальних програмних застосунків, що використовують вже готові бібліотеки із вбудованими платами МК (мікроконтролерів) та елементами електричних схем (резисторів, потенціометрів, світлодіодів та інших елементів). Для користувача, що буде застосовувати програмний застосунок LCD_Program можу дати поради для використання її у своїх особистих цілях.

Варіант 1: Застосування програми із використанням апаратного обладнання.

Якщо у користувача виникне потреба використовувати LCD_Program із апаратним обладнанням, то можна дати такі поради по використанню:

1. Придбати плату Arduino та рідкокристалічний дисплей LCD1602 разом із іншими елементами електронних схем, що можуть стати у нагоді;
2. Виконати підключення схеми. Опис підключення є розділі 2, пункт 2.1.2;
3. На персональному комп’ютері завантажити інтегроване середовище розробки Arduino IDE, для користування програмою LCD_Program та написання коду;

4. Підключити мікроконтролер до USB – порту комп’ютера;
5. Для користування програмою необхідно ознайомитись у розділі 2 із функціями, які використовуються для роботи із нею;
6. У роботі із символами кирилиці необхідно спочатку додати ті символи із файлу Cyrilic_symbol.h у CGRAM пам’ять, що буде застосовуватись. Далі треба ініціалізувати масив символів для зберігання їх у нього, а також масив індексів, де будуть зберігатись індекси, що відповідають тим коміркам пам’яті, де знаходяться створені символи у ній. Використовуючи функцію LCD_Cyrilic_print виводимо рядок символів кирилиці разом із символами латиниці;

   Приклад роботи із символами кирилиці:

```
int main() {
   LCD_init_4bit_mode();
  Cyrilic_symbol(1, symbol1);  // "Б" у CGRAM location 1
   Cyrilic_symbol(2, symbol10);  // "Ц" у CGRAM location 2
   Cyrilic_symbol(3, symbol3);  // "Ь" у CGRAM location 3
   Cyrilic_symbol(4, symbol4);  // "Ю" у CGRAM location 4
   Cyrilic_symbol(5, symbol5);  // "Я" у CGRAM location 5

   // Оновлення масиву спеціальних символів
   char new_symbols[] = {'Б', 'Ц', 'Ь', 'Ю', 'Я'};
   uint8_t new_indices[] = {1, 2, 3, 4, 5};
   update_special_symbols(new_symbols, new_indices, 5);
    LCD_Set_pos(0, 0);
    LCD_Cyrilic_print("СОНЦЕ");
   while (1) {}return 0;}
```

Варіант 2: Застосування програми із використанням програмних засобів чи онлайн додатків.

Цей варіант підходить для новачків, які тільки будуть вивчати програмування на мікроконтролерах. Тому замість апаратного обладнання можна використовувати такі альтернативи Wokwi, Proteus та інші додатки.

1. Завантажте Wokwi або Proteus на свій персональний комп'ютер;
2. Відкрийте програму та створіть проект та додайте до нього готові елементи електронних схем із вбудованих бібліотек;
3. З'єднайте елементи між собою із дотриманням рекомендацій пункту 2 варіанту 1;
4. У полі для писання програми вставте код програми LCD_Program та дотримуйтесь рекомендацій пунктів 5- 6 із варіанту 1;

Висновки до розділу 3

Підсумовуючи розділ 3 можна узагальнити все вище сказане, що під час проведення тестування програмної моделі у програмному застосунку Proteus, онлайн – додатку Wokwi та із застосуванням апаратного обладнання тести пройшли вдало, помилок та програмних несправностей невиявлено.

Також був проведений аналіз подібного проекту, що був знайдений на просторах мережі Інтернет. Під час порівняння проектів LCD_Program та LCD_1602_R_ALL були виявлені такі переваги та недоліки. Із суцільних переваг і недоліків моєї програмної моделі можна виділити наступні:

Переваги:

- Використання прямого контролю підключення до мікроконтролю ATmega2560;
- Використання мінімальної кількості сторонніх бібліотек;

Недоліки:

- Застосування ручного керування символів у CGRAM пам'яті;
- Обмежена підтримка кириличних символів без урахування конвертації для UTF-8 та Win-1251 символів;

Тоді як в проекті LCD_1602_R_ALL застосовується стандартне використання функцій, що закладено у бібліотеку LCD_LIB_H для мови

програмування C++, а також є підтримка застосування бібліотеки при різних типів підключення дисплею LCD1602 (паралельно та з модулем I2C).

Але із недоліків є застосування додаткових ресурсів пам’яті для зберігання символів у CGRAM пам’яті та обмеженість на кількісне використання кастомних символів.

Для користувачів, що будуть користуватись моїм проектом було написано ряд рекомендацій з яким можна ознайомитись у розділі 3, пункт 3.8.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було ознайомлено із будовою дисплея LCD1602 на базі контролера HD44780 та реалізовано мікроконтролерне керування дисплеєм й розроблено виведення символів кирилиці на дисплей.

Для реалізації мікроконтролерного керування та розробки методики виводу символів кирилиці на дисплей було самостійно розроблено програмну модель, візуально представивши процес її розробки, де зазначено ряд функцій, що будуть використовуватись.

Функції програмної моделі написані на мові програмування С. Їх тестування відбувалось із застосуванням апаратного обладнання, в програмі Proteus й онлайн-застосунку Wokwi. Під час тестування програмної моделі, помилок виконання у роботі функцій помічено не було це свідчить про те що, програмна модель працює коректно.

До цього всього було здійснене порівняння методики керування LCD_Program із іншим подібним проектом, зазначивши їхні переваги та недоліки. Для користувача були написані ряд рекомендацій із користуванням цією програмою у двох варіантах випадків - це застосування програми із використанням апаратного обладнання та програмних засобів чи онлайн додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hitachi HD44780 LCD controller [Електронний ресурс]. – режим доступу:URL:https://en.wikipedia.org/wiki/Hitachi_HD44780_LCD_controller (дата звернення - 13.01.2025).
2. HD44780 compatible LCD controllers [Електронний ресурс]. – режим доступу:URL:http://midibox.org/dokuwiki/offline/hd44780_compatible.html (дата звернення - 13.01.2025).
3. Interfacing 16×2 Character LCD Module with Arduino [Електронний ресурс]. – режим доступу: URL: https://lastminuteengineers.com/arduino-1602-character-lcd-tutorial/ (дата звернення - 14.01.2025)
4. DOT MATRIX LIQUID CRYSTAL DISPLAY MODULE HJ1602A Serial USER‘MANUAL [Електронний ресурс]. – режим доступу: URL: https://hades.mech.northwestern.edu/images/f/f7/LCD16x2_HJ1602A.pdf (дата звернення - 14.01.2025).
5. Котвицький А. Т. INTELLECTUAL CAPITAL IS THE FOUNDATION OF INNOVATIVE DEVELOPMENT ROBOTIC SYSTEMS [Електронний ресурс].–режим доступу: URL: https://desymp.promonograph.org/index.php/sge/issue/view/sge28-01/sge28-01 (дата звернення - 03.02.2025).
6. Brain W. Kernighan, Dennis M. Ritche The C programing, Second edition [Електронний ресурс].–режим доступу:URL: https://www.cs.sfu.ca/~ashriram/Courses/CS295/assets/books/C_Book_2nd.pdf (дата звернення - 03.02.2025)
7. Bayle J. C Programming for Arduino [Електронний ресурс].–режим доступу: URL: https://archive.org/details/cprogrammingfora0000juli (дата звернення - 15.02.2025).
8. Proteus Design [Електронний ресурс].–режим доступу: URL: https://uk.wikipedia.org/wiki/Proteus_Design (дата звернення - 20.02.2025).

9. Wokwi [Електронний ресурс]. – режим доступу:URL: https://wokwi.com/ (дата звернення - 23.02.2025).
10. Sergey S. LCD_1602_R_ALL [Електронний ресурс].–режим доступу: URL: https://surl.li/kxpxvq (дата звернення - 03.03.2025).

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Бакалавр**
Галузь знань: 15 – Автоматизація та приладобудування
Спеціальність: 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітня програма «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«02» жовтня 2024року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

ЛИТИНСЬКОГО Владислава Максимовича
(прізвище, ім'я, по батькові студента)

1. Тема роботи **МОДЕЛЬ МІКРОКОНТРОЛЕРНОГО КЕРУВАННЯ ДИСПЛЕЄМ НА БАЗІ HD44780**

керівник роботи **Котвицький Альберт Тадеушевич, к. ф.-м. н., доцент**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від ***16 квітня 2025 року № 4101-5/962***

2. Строк подання студентом роботи ***30 травня 2025 року***

3. Перелік питань, які потрібно розробити

1. Аналіз актуальності дисплея LCD1602
2. Вивчення технічної документації контролера HD44780.
3. Розробка програмної моделі мікроконтролерного керування дисплеєм LCD1602.
4. Реалізувати методику програмної моделі керування дисплеєм.
5. Виконати тестування розробленої методики та виправити наявні помилки
6. Проаналізувати та порівняти програмні рішення реалізації методики подібних проектів
7.Скласти рекомендації щодо користування програмою

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Затвердження теми роботи	02.10.2024-09.11.2024
2	Огляд та вивчення актуальності теми роботи	10.11.2024-24.11.2024
3	Аналіз наукової літератури та інформації щодо дисплея LCD1602 на базі контролера HD44780	25.11.2024-09.12.2024
4	Вивчення програмних можливостей дисплея LCD1602 на базі контролера HD44780	10.12.2024-24.12.2024
5	Аналіз методики керування дисплеєм	25.12.2024-02.01.2025
6	Розробка абстрактної програмної моделі керування дисплеєм	03.01.2025-17.01.2025
7	Реалізація програмної моделі на мові програмування С	18.01.2025-17.02.2025
8	Тестування методики моделі керування LCD1602 та усунення ймовірних помилок	18.02.2025-25.02.2025
9	Підготовка рекомендаційних порад користування програмною моделлю	26.02.2025-09.03.2025
10	Підготовка та оформлення звітних матеріалів. Написання розділів до кваліфікаційної роботи та оформлення списку використаної літератури	10.03.2025-14.04.2025
11	Оформлення пояснювальної записки кваліфікаційної роботи відповідно вимогам до звітів про НДР.	15.04.2025-18.05.2025
12	Представлення кваліфікаційної роботи керівнику та рецензенту	30.05.2025

5. Дата видачі завдання ***02 жовтня 2024 року.***

Студент В.М. Литинський _____
ініціали, прізвище підпис

Керівник роботи А.Т. Котвицький _____
ініціали, прізвище підпис

Додаток Б

Затверджую

_____

«_____»_____2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на розробку програмного виробу
«Модель мікроконтролерного керування дисплеєм на базі HD44780»

Назва розділу	Назва та зміст підрозділу
1. Вступ	1.1. Назва програмного виробу: Модель мікроконтролерного керування дисплеєм на базі HD44780 1.2. Галузь застосування: робототехніка, промисловість, комерційний проект
2.Підстава для розробки	2.1. Навчальний план за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології. 2.2. Завдання на кваліфікаційну роботу бакалавра затверджені наказом по університету № 4101-5/962 від 16 квітня 2025 року (представити як Додаток А до пояснювальної записки до дипломної роботи).
3.Призначення розробки	3.1. Метою розробки програмного виробу є керування дисплеєм з використанням мікроконтролера ATmega2560. 3.2. Керування рідкокристалічним дисплеєм за допомогою вбудованих команд, а також вивід символів кирилиці на дисплей із використанням реалізованого алгоритму виводу. 3.3. Вхідні дані для розробки: на вхід користувачу подається набір текстових команд із комбінацією символів, що виводяться на дисплей. 3.4. Вихідні дані для розробки: на дисплеї повинен відображатися текст або символ.
4.Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: повинна бути програмна та апаратна реалізація програмного виробу 4.2. Вимоги до надійності: немає 4.3. Вимоги до умов експлуатації: немає 4.4. Вимоги до складу параметрів технічних засобів: макетна плата, змінний резистор B10K для керування контрастністю дисплея, дроти «ПАПА-ПАПА» для з'єднання, дисплей LCD1602 на базі контролера HD44780, світлодіод, мікроконтролер ATmega2560, ПК та встановлена Arduino IDE

<table>
<tr><td></td><td colspan="2">4.5. Вимоги до інформаційної та програмної сумісності:
Програмна модель виключно працює на мікроконтролері ATmega2560
4.6. Вимоги до маркіровки та упаковки: немає
4.7. Вимоги до транспортування та зберігання: не зазначено
4.8. Спеціальні вимоги: немає</td></tr>
<tr><td>5. Вимоги до програмної документації.</td><td colspan="2">Програмною документацією до виробу «Модель мікроконтролерного керування дисплеєм на базі HD44780» вважати:
1) Справжнє Технічне завдання на розробку програмного виробу (зазначити у вигляді Додатку Б пояснювальної записки кваліфікаційної роботи)
2) Програму та методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до пояснювальної записки кваліфікаційної роботи)
3) Опис програмного виробу (у вигляді пунктів 2.2 – 2.2.11 Розділу 2 до пояснювальної записки кваліфікаційної роботи).
4) Текст програми (представити у вигляді Додатку Г пояснювальної записки до кваліфікаційної роботи).</td></tr>
<tr><td>6.Техніко-економічні показники</td><td colspan="2">Оцінку ефективності та економіко-технічних показників виконувати не потрібно</td></tr>
<tr><td rowspan="2">7.Стадії та етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>Від 02.10.2024
до 09.11.2024

Від 10.11.2024
до 24.11.2024

Від 25.11.2024
до 09.12.2024

Від 10.12.2024
до 24.12.2024

Від 25.12.2024
до 02.01.2025

Від 03.01.2025
до 17.01.2025

Від 18.01.2025</td><td>Затвердження теми роботи

Огляд та вивчення актуальності теми роботи

Аналіз наукової літератури та інформації щодо дисплея LCD1602 на базі контролера HD44780

Вивчення програмних можливостей дисплея LCD1602 на базі контролера HD44780

Аналіз методики керування дисплеєм

Розробка абстрактної програмної моделі керування дисплеєм</td></tr>
</table>

	до 17.02.2025 Від 18.02.2025 до 25.02.2025 Від 26.02.2025 до 09.03.2025 Від 10.03.2025 до 14.04.2025 Від 15.04.2025 до 18.05.2025 30.05.2025	Реалізація програмної моделі на мові програмування С Тестування методики моделі керування LCD1602 та усунення ймовірних помилок Підготовка рекомендаційних порад користування програмною моделлю Підготовка та оформлення звітних матеріалів. Написання розділів до кваліфікаційної роботи та оформлення списку використаної літератури Оформлення пояснювальної записки кваліфікаційної роботи відповідно вимогам до звітів про НДР. Представлення кваліфікаційної роботи керівнику та рецензенту
8.Порядок контролю та приймання	1) Перевірку ходу розроблення програмної моделі здійснювати 1 раз на 2 тижні. 2) Розроблені проектні документи подати в електронному вигляді та на паперових носіях в одному примірнику. 3) Захист та прийом дипломної роботи приймається на засіданні Атестаційної комісії	

Виконавець
студент групи КУ-41
Литинський В. М.

_____

Замовник
К.ф.-м.н., доцент кафедри КСтаР
Котвицький А. Т.

_____

Додаток В

Затверджую

_____

«_____»_____2025р.

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ
програмного виробу «Модель мікроконтролерного керування дисплеєм на базі HD44780»

1. Об'єкт випробувань

Об’єктом випробувань є реалізація методики мікроконтролерного керування дисплеєм на базі HD44780.

2. Мета випробувань

Перевірка відповідності функціональності розробленого програмного виробу заявленим вимогам в технічному завданні (Додаток Б до пояснювальної записки)

3. Загальні положення

3.1 Підстави щодо випробувань

Підставою для випробувань є затвердження завдання на бакалаврську кваліфікаційну роботу.

3.2 Місце та тривалість випробувань

Приймальні випробування проводяться на базі власного обладнання вдома в період роботи атестаційної комісії.

3.3. Обсяг випробувань

Приймальні випробування програмного виробу проводяться в обсязі відповідно до цієї Програми та методики випробувань.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання за участю Замовника, Виконавця та інших присутніх осіб, призначених для прийому бакалаврської кваліфікаційної роботи.

4. Вимоги до програмного виробу

Програма повинна:

1. Демонструвати реалізацію методики моделі мікроконтролерного керування дисплеєм на базі HD44780;

2. Виводити на дисплей текст (кирилицею або латиницею) або символи;
3. Застосовувати лише один дисплей LCD1602 на базі HD44780;
4. Використовувати мікроконтролер моделі ATmega2560;
5. Працювати з використанням Arduino IDE;

5. Вимоги до програмної документації

Склад програмної документації, що подається на випробування включає:

1) Справжнє Технічне завдання на розробку програмного виробу (зазначити у вигляді Додатку Б пояснювальної записки кваліфікаційної роботи)
2) Програму та методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до пояснювальної записки кваліфікаційної роботи)
3) Опис програмного виробу (у вигляді пунктів 2.2 – 2.2.11 Розділу 2 до пояснювальної записки кваліфікаційної роботи).
4) Текст програми (представити у вигляді Додатку Г пояснювальної записки до кваліфікаційної роботи).

6. Засоби та порядок випробувань

6.1 Засоби випробувань

Для виконання випробувань необхідно мати такі апаратні засоби:

1. Мікроконтролер ATmega2560;
2. Змінний резистор B10K та макетна плата;
3. З’єднувальні дроти типу «ПАПА-ПАПА»;
4. Дисплей LCD1602 на базі контролера HD44780;
5. Світлодіод;

Для відтворення програми необхідно мати ПК з операційною системою Windows 11, USB-B кабель, що з’єднує мікроконтролер та ПК, а також інтегроване середовище розробки Arduino IDE.

6.2 Порядок проведення випробувань

Зазвичай випробовування проводяться у два етапи:

- 1-й етап ознайомчий;
- 2-й етап випробовування програмного виробу;

Перелік перевірок, що проводяться на 1-му етапі, включає наступні операції:

1. Перевірку комплектності програмної документації.

2. Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.

3. Перевірку комплектності складу технічних і програмних засобів.

4. Методику проведення перевірок на 1 етапі випробувань.

5. Перевірка якості програмної документації перевіряється на відповідність вимогам стандартів ЕСПД.

Перелік перевірок, що проводяться на другому етапі випробувань, включає наступні операції:

1. Перевірку працездатності розробленого програмного виробу;

Тест вважається виконаним, якщо функція відпрацьовала та результат виконання відображається на дисплей й є задовільним.

2. Перевірку ступеня виконання функціональних вимог до програмного виробу;

Програма працює відповідно до умов експлуатації операційної системи Microsoft Windows. Для роботи програмного виробу необхідний компілятор мови програмування C/C++ стандарту не нижче C98.

Порядок проведення випробувань:

1. Для початку треба запустити файл LCD_Program з розширенням ino в середовищі розробки Arduino IDE;
2. Далі необхідно треба підключити апаратні компоненти(мікроконтролер ATmega2560, макетна плата, дисплей LCD1602, змінний резистор B10K та світлодіод) до ПК, що знадобляться під час проведення випробувань;
3. В середовищі розробки Arduino IDE пропонується провести два випробування у вигляді функцій test1() та test2() в коді програми;
4. Спочатку викликати функцію test1(), а наступною за нею функцію test2() у області для користувача int main().
5. Запустити програму на виконання. На вкладці меню вікна Arduino IDE натиснути на стрілочку, що знаходиться поряд із галочкою, або навкладці меню вікна Arduino IDE натиснути на Скетч ->Перевірити/Зкомпілювати.
6. Функції тестів test1() та test2(), що були зкомпільовані без помилок завантажується до мікроконтролера ATmega2560.
7. Програмний виріб пройшов випробування, якщо обидва тести були виконані успішно, якщо на дисплеї LCD1602 видні результати випробовування.

Для проведення випробовувань пропонується test1() та test2().

Тест 1

В цьому тесті вимагається перевірити виконання програми виводу тексту латиницею на дисплей LCD1602, виконавши її в середовище розробки Arduino IDE.

```
void test1(){
   LCD_init_4bit_mode();
    LCD_Set_pos(0,0);
    LCD_print("Hello");
}
```

Рисунок В.1 - Код функції test1().

```
int main() {
    test1();
    test2();
    while (1) {

    }
    return 0;
```

Рисунок В.2 – Приклад виклику тесту у головній програмі.

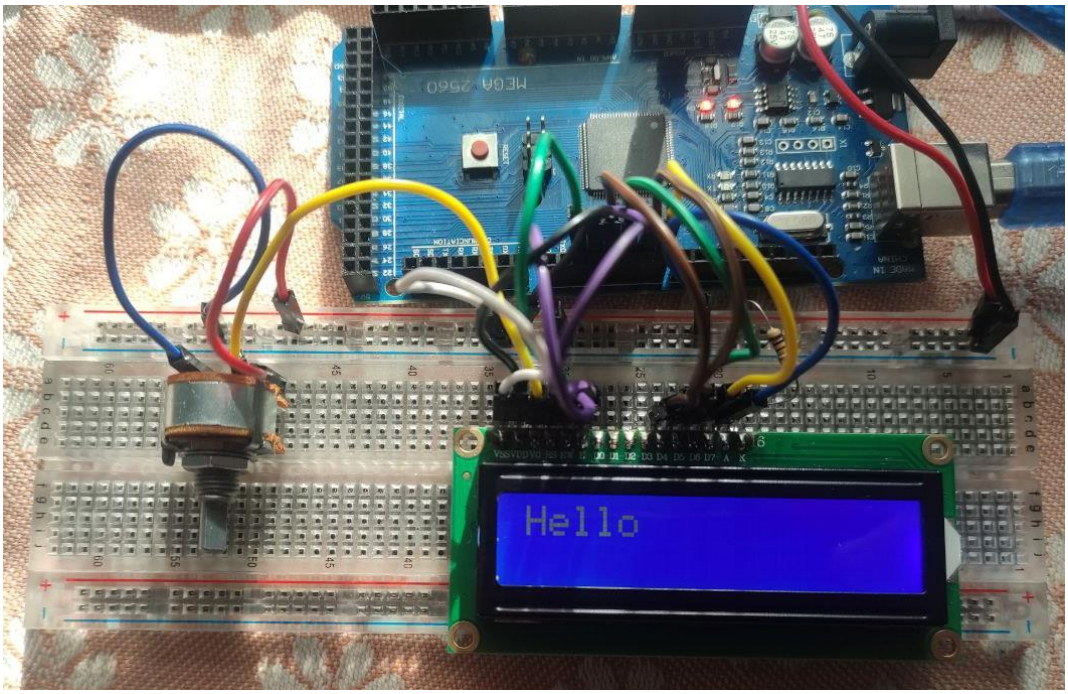

Рисунок В.3 – Результат виконання test1().

Якщо слово «Hello» вивелось на дисплей, то можна вважати, що Тест 1 проведено успішно. Якщо слово «Hello» не з’явилось на дисплеї, то потрібно перевірити помилки в коді або виявити несправності в апаратному обладнанні.

Тест 2

В цьому тесті пропонується перевірити виконання програми виводу тексту латиницею разом зі спеціальними створеними символами кирилиці на дисплей LCD1602, виконавши її в середовище розробки Arduino IDE.

```
void test2(){
   LCD_init_4bit_mode();
   Cyrilic_symbol(1, symbol1);
    Cyrilic_symbol(2, symbol5);
    Cyrilic_symbol(3, symbol8);
    Cyrilic_symbol(4, symbol4);
    Cyrilic_symbol(5, symbol5);

    char new_symbols[] = {'Б', 'Ж', 'Ї', 'Ю', 'Я'};
    uint8_t new_indices[] = {1, 2, 3, 4, 5};
    update_special_symbols(new_symbols, new_indices, 5);
     LCD_Set_pos(0, 0);
     LCD_Cyrilic_print("ЇЖАК");
}
```

Рисунок В.4 - Код функції test2().

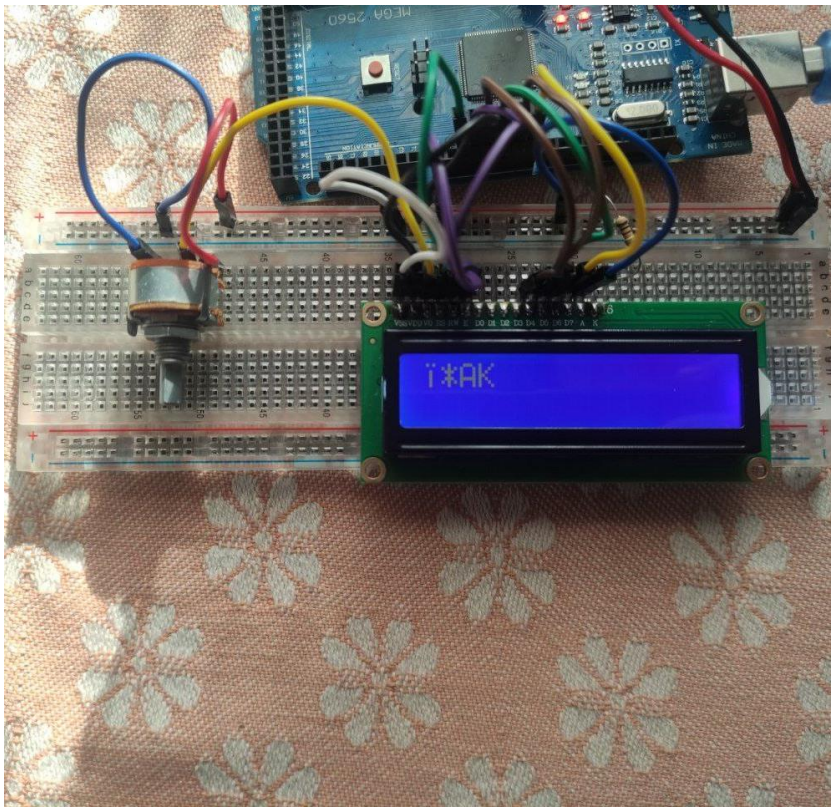

Рисунок В.5 - Результат виконання test2().

Якщо слово «ЇЖАК» вивелось на дисплей, то можна вважати, що Тест 2 виконано успішно. Якщо слово «ЇЖАК» не з’явилось на дисплеї, то потрібно перевірити помилки в коді або виявити несправності в апаратному обладнанні.

У підсумку можна зробити висновок, що Тест 1 та Тест 2 виконано вдало.
Виконавець
студент групи КУ-41
Литинський В. М.

_____

Додаток Г

Лістинг програми

Модель мікроконтролерного керування дисплеєм на базі HD44780

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "Cyrilic_symbol.h" // Підключаємо файл з символами

// Макроси для роботи з RS і E
#define E_High ((PORTE) |= (1 << PE5))
#define E_Low ((PORTE) &= ~(1 << PE5))
#define RS_High ((PORTE) |= (1 << PE4))
#define RS_Low ((PORTE) &= ~(1 << PE4))

// Ініціалізація портів для роботи з LCD
void LCD_4bit_mode() {
    DDRE |= (1 << PE4) | (1 << PE5);   // RS і E
    DDRG |= (1 << PG5);                // D4
    DDRE |= (1 << PE3);                // D5
    DDRH |= (1 << PH3) | (1 << PH4);   // D6 і D7
}

// Передача 4-бітних даних
void LCD_write_4bit_mode(uint8_t n) {
    PORTG &= ~(1 << PG5);
    PORTE &= ~(1 << PE3);
    PORTH &= ~(1 << PH3);
    PORTH &= ~(1 << PH4);
 E_High;
 _delay_us(1);
    if (n & 0x01) PORTG |= (1 << PG5);
    if (n & 0x02) PORTE |= (1 << PE3);
    if (n & 0x04) PORTH |= (1 << PH3);
    if (n & 0x08) PORTH |= (1 << PH4);

    E_Low;
    _delay_us(50);
}
```

```
// Відправка байта (команда чи дані)
void Send_byte_4bit_mode(uint8_t data, uint8_t type) {
    if (type)
        RS_High;
    else
        RS_Low;

    LCD_write_4bit_mode(data >> 4);   // Старші біти
    LCD_write_4bit_mode(data); // Молодші біти
}

// Ініціалізація дисплея
void LCD_init_4bit_mode() {
    LCD_4bit_mode();
    _delay_ms(20);

    LCD_write_4bit_mode(0x03);
    _delay_ms(5);
    LCD_write_4bit_mode(0x03);
    _delay_us(150);
    LCD_write_4bit_mode(0x03);
    _delay_us(50);
    LCD_write_4bit_mode(0x02);
    _delay_us(50);

    Send_byte_4bit_mode(0x2C, 0); // 4-бітний режим, 2 лінії
     _delay_us(50);
    Send_byte_4bit_mode(0x0C, 0); // Увімкнути дисплей, без курсора
     _delay_us(50);
    Send_byte_4bit_mode(0x01, 0); // Очистити дисплей
    _delay_ms(2);
    Send_byte_4bit_mode(0x06, 0); // Очистити дисплей
    _delay_us(50);
}

// Виведення тексту (ASCII символи)
void LCD_print(char *str) {
    uint8_t c = 0;
    while (str[c] != 0) {
        Send_byte_4bit_mode(str[c], 1);
        c++;
    }
}
```

```
void LCD_display_control(uint8_t n) {
    if (n == 1) {
        Send_byte_4bit_mode(0x0C, 0);
    } else if (n == 0) {
        Send_byte_4bit_mode(0x08, 0);
    }}

// Установка позиції курсора
void LCD_Set_pos(uint8_t line, uint8_t pos) {
    uint8_t address = (line * 0x40 + pos) | 0x80;
    Send_byte_4bit_mode(address, 0);
    _delay_us(50);
}

// Завантаження кириличного символу в CGRAM
void Cyrilic_symbol(uint8_t location, uint8_t symbol[]) {
    location &= 0x07; // Тільки 8 місць для символів (0-7)
    Send_byte_4bit_mode(0x40 | (location << 3), 0); // Установка CGRAM адреси

    for (uint8_t i = 0; i < 8; i++) {
        Send_byte_4bit_mode(symbol[i], 1);
    }
    _delay_us(100);
}

// Масиви для спеціальних символів та їх індексів у CGRAM
char special_symbols[8]; // Спеціальні символи
uint8_t cgram_indices[8]; // Відповідні індекси CGRAM
uint8_t special_symbols_count = 0; // Кількість спеціальних символів

// Функція для оновлення масиву спеціальних символів
void update_special_symbols(char new_symbols[], uint8_t new_indices[], uint8_t
count) {
    for (uint8_t i = 0; i < count; i++) {
        special_symbols[i] = new_symbols[i];
        cgram_indices[i] = new_indices[i];
    }
    special_symbols_count = count;
}
// Перевірка, чи символ є спеціальним
uint8_t is_special_symbol(char symbol, uint8_t *cgram_index) {
    for (uint8_t i = 0; i < special_symbols_count; i++) {
        if (special_symbols[i] == symbol) {
            *cgram_index = cgram_indices[i];
```

```
            return 1; // Символ знайдено
        }
    }
    return 0; // Символ не знайдено
}

// Виведення рядка зі спеціальними символами
void LCD_Cyrilic_print(char *str) {
    while (*str) {
        uint8_t cgram_index;
        if (is_special_symbol(*str, &cgram_index)) {
            // Символ знайдено у CGRAM
            Send_byte_4bit_mode(cgram_index, 1);
        } else if (*str >= 0x20 && *str <= 0x7F) {
            // ASCII символ
            Send_byte_4bit_mode(*str, 1);
        }
        // Невідомий символ ігнорується (нічого не виводимо)
        str++;
    }
}
void test1(){
   LCD_init_4bit_mode();
    LCD_Set_pos(0,0);
    LCD_print("Hello");
}
void test2(){
   LCD_init_4bit_mode();
   Cyrilic_symbol(1, symbol1);
    Cyrilic_symbol(2, symbol5);
    Cyrilic_symbol(3, symbol8);
    Cyrilic_symbol(4, symbol4);
    Cyrilic_symbol(5, symbol5);

    char new_symbols[] = {'Б', 'Ж', 'Ї', 'Ю', 'Я'};
    uint8_t new_indices[] = {1, 2, 3, 4, 5};
    update_special_symbols(new_symbols, new_indices, 5);
     LCD_Set_pos(0, 0);
     LCD_Cyrilic_print("ЇЖАК");
}
int main() {
    //test1();
    test2();     while (1) {} return 0; }
```

Додаток Д

Результати тестування

Тестування функції вимкнення/включення дисплея

Виконаємо тестування функції у Proteus, Wokwi й на апаратному обладнанні. Результати тестування функції продемонстровано нижче.

Тестування у Proteus:

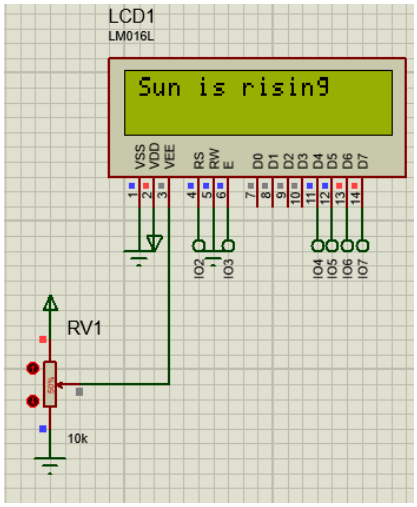

Рисунок Д.1 –Тестування вимкнення дисплея (результат до).

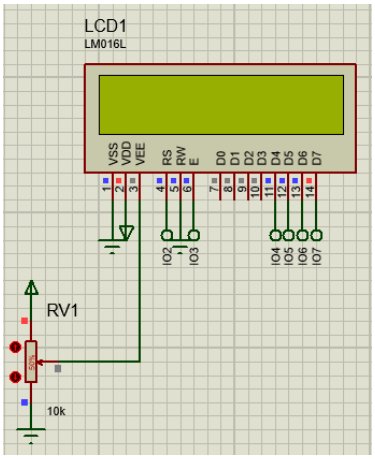

Рисунок Д.2 –Тестування вимкнення дисплея (результат після).

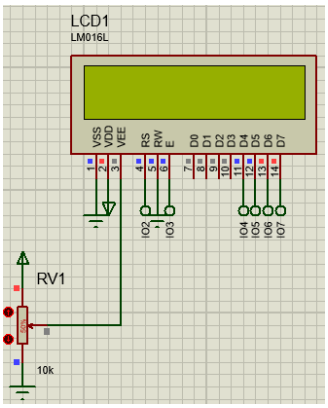

Рисунок Д.3 –Тестування включення дисплея (результат до).

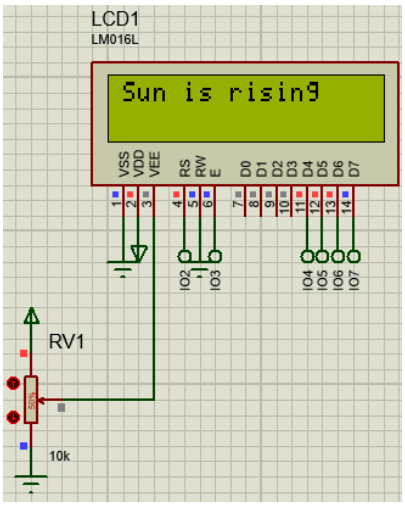

Рисунок Д.4 –Тестування включення дисплея (результат після).

Тестування у Wokwi:

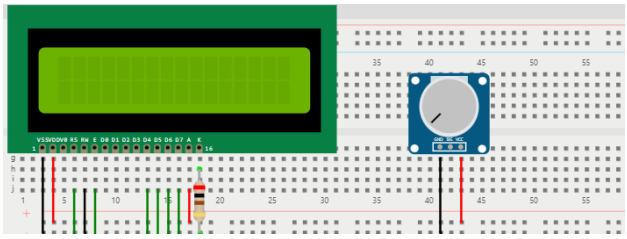

Рисунок Д.5 -Тестування увімкнення дисплея (результат до).

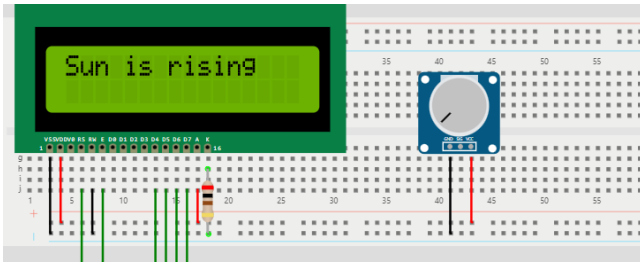

Рисунок Д.6 -Тестування увімкнення дисплея (результат після).

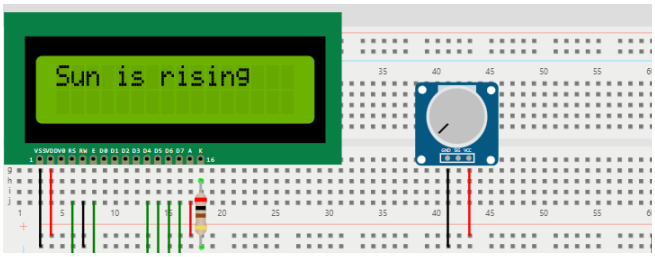

Рисунок Д.7 -Тестування вимкнення дисплея (результат до).

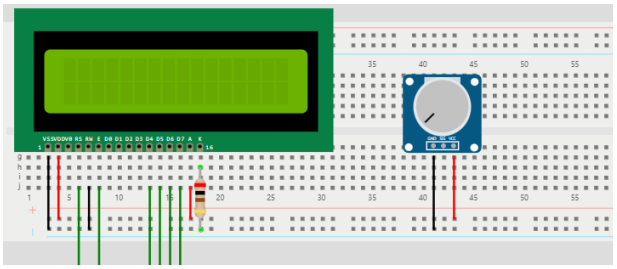

Рисунок Д.8 -Тестування вимкнення дисплея (результат після).

Тестування на апаратному обладнанні:

Рисунок Д.9 -Тестування вимкнення дисплея на апаратному обладнанні (результат до).

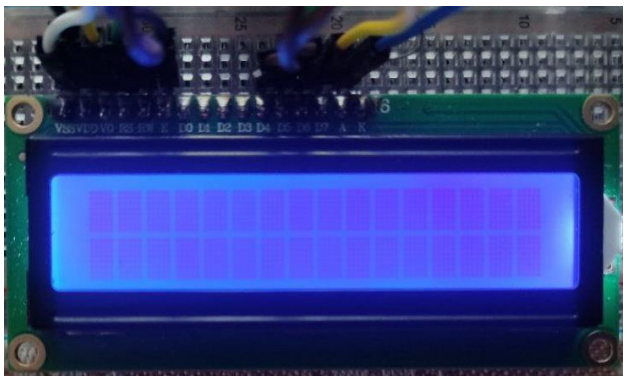

Рисунок Д.10 -Тестування вимкнення дисплея на апаратному обладнанні (результат після).

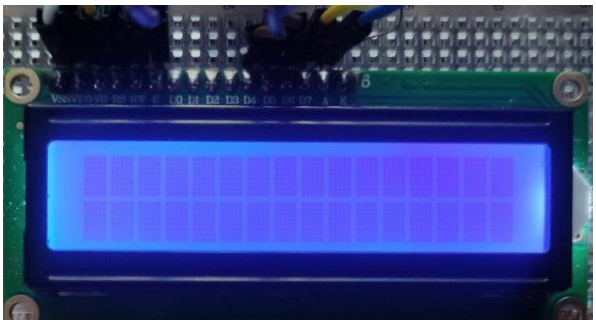

Рисунок Д.11 – Тестування увімкнення дисплея на апаратному обладнанні (результат до).

Рисунок Д.12 – Тестування увімкнення дисплея на апаратному обладнанні (результат після).

Провівши тестування функції вимкнення/увімкнення дисплея у Proteus (див рис Д.1 - Д.4), Wokwi (див рис. Д.5 – Д.8) та з використанням апаратного обладнання (див рис. Д.9 -Д.12) можна сказати, що функція працює правильно й коректно, а також помилок при її тестуванні не було помічено.

Тестування функції виставлення курсора у довільну позицію на дисплеї

Тестування у Proteus:

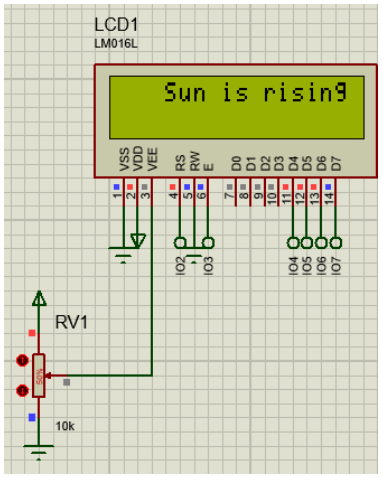

Рисунок Д.13 – Тестування функції виставлення курсора у Proteus.

Тестування у Wokwi:

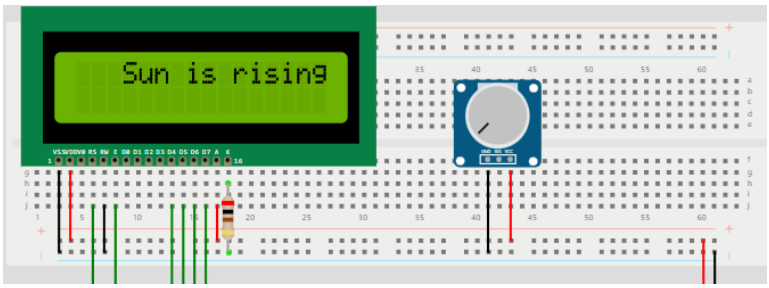

Рисунок Д.14 – Тестування функції виставлення курсора у Wokwi.

Тестування на апаратному обладнанні:

Рисунок Д.15 – Тестування функції виставлення курсора на апаратному обладнанні.

Провівши тестування функції у Proteus, Wokwi й на апаратному обладнанні, виставивши курсор на першому рядку у позиції чотири на ньому. З рис Д.13 – Д.15 можна проаналізувати, що функція коректно виконує свої дії.

Тестування функції створення власного символа
Тестування у Proteus:

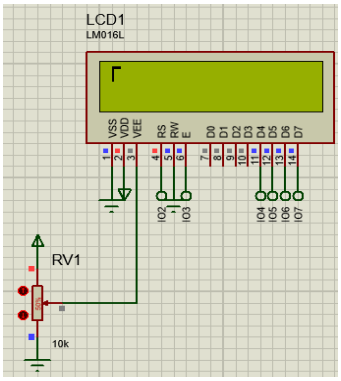

Рисунок Д.16 – Тестування функції створення зі символа у Proteus.

Тестування у Wokwi:

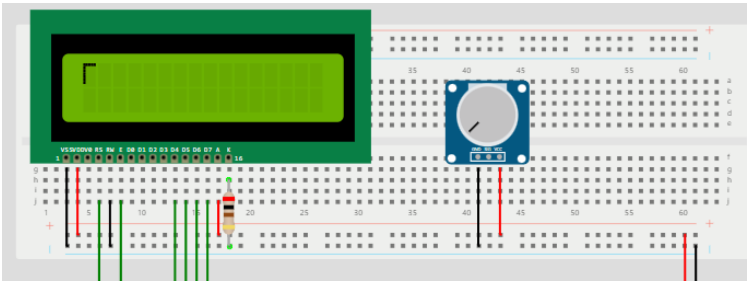

Рисунок Д.17 – Тестування функції створення зі символа у Wokwi.

Тестування на апаратному обладнанні:

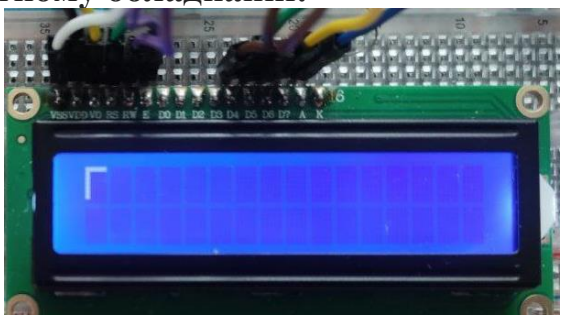

Рисунок Д.18 – Тестування функції створення зі символа на апаратному обладнанні.

Для тестування цієї функції було створено символ кирилиці літеру «Г» для перевірки виводу символа застосовано Proteus, Wokwi й апаратне обладнання. Підсумовуючи тестування функції створення спеціального символа, вона працює й виводить створений спеціальний символ правильно.

Тестування функції виводу рядку кирилиці із символами латиниці

Тестування у Proteus:

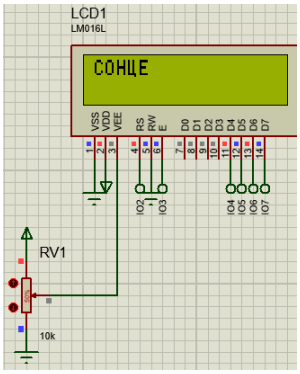

Рисунок Д.19 - Тестування функції виводу рядку кирилиці разом із символами латиниці у Proteus.

Тестування у Wokwi:

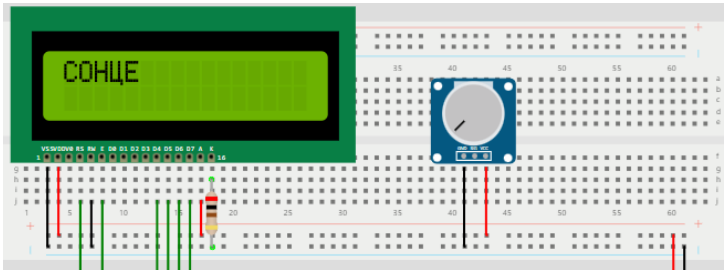

Рисунок Д.20 - Тестування функції виводу рядку кирилиці разом із символами латиниці у Wokwi.

Тестування на апаратному обладнанні:

Рисунок Д.21 – Тестування функції виводу рядку кирилиці разом із символами латиниці на апаратному обладнанні.

Провівши тестування цієї функції спочатку був ініціалізований масив для символів, що будуть використовуватись під час роботи із дисплеєм. Ці символи я додав у CGRAM пам’ять.
Далі використовую додаткові функції для визначення кількості символів у цьому масиві й вивожу рядок з використанням символів латиниці, тому що деякі із них є в кирилиці й я буду використовувати ті яких немає в латинському алфавіті для виводу на дисплей. За результатами тестів функція працює правильно й коректно.
Перед тим як переходити до тестування програмної моделі, було виявлено дивна поведінка програми у функції запису. Нижче наведена реалізована функція запису LCD_write_4bit_mode(uint8_t n) згідно із даташитом за рис 2.4 у розділі 2.

```
void LCD_write_4bit_mode(uint8_t n) {
   E_High;
    _delay_us(1);
    PORTG &= ~(1 << PG5);
    PORTE &= ~(1 << PE3);
    PORTH &= ~((1 << PH3) | (1 << PH4));  // Очистити порти даних

    if (n & 0x01) PORTG |= (1 << PG5);
    if (n & 0x02) PORTE |= (1 << PE3);
    if (n & 0x04) PORTH |= (1 << PH3);
    if (n & 0x08) PORTH |= (1 << PH4);

        E_Low;
    _delay_us(50);  // Загальна затримка для обробки команди
}
```

Спробуємо протестувати програмну модель на прикладі функцій виводу рядку тексту на дисплей в онлайн середовищі Wokwi.

```
int main() {
  LCD_init_4bit_mode();
```

```
  LCD_print("Ar");
    while (1) {

    };
    return 0;
}
```

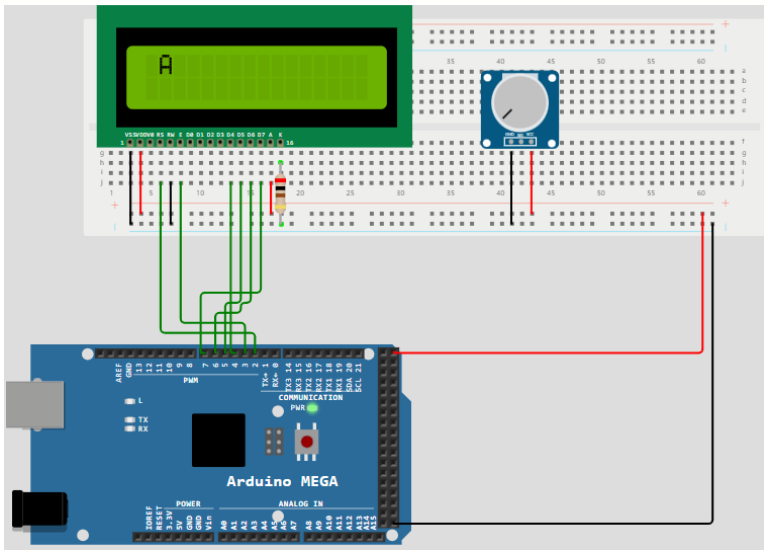

Рисунок Д.22 – Тестування.

З рис 3.36 як можна побачити на дисплей у нас вивелась тільки одна літера, замість двох що ми ввели у якості параметру в функціїї LCD_print(). За результатами тестування можна ствердити, що програмна модель працювала б в подальшому некоректно при реалізації інших функцій, які зараз є в програмній моделі. Тому функцію запису було змінено для правильної роботи програмної моделі. Змінена функція наявна у розділі 2, пункт 2.2.2.