Київський столичний університет імені Бориса Грінченка

Факультет інформаційних технологій та математики

Кафедра комп’ютерних наук

«Допущено до захисту»
Завідувач кафедри
комп’ютерних наук,
доктор технічних наук, професор

_____ А. П. Бондарчук

« ___ » _____ 2025_ р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня «Магістр»
Спеціальність 122 Комп’ютерні науки
Освітня програма 122.00.02 Інформаційно-аналітичні системи

Тема роботи: Багаторозрядна цифрова індикація у вбудованих системах

Виконав

студент групи _ІАСм-1-24-1.4д_
(шифр академічної групи)

Кушнір Олег Володимирович
(прізвище, ім’я, по батькові)

_____
(підпис)

Науковий керівник

Доктор технічних наук.
Професор.
(науковий ступінь, наукове звання)

Бушма О. В.
(прізвище, ініціали)

_____
(підпис)

Київ – 2025

Київський столичний університет імені Бориса Грінченка

Факультет інформаційних технологій та математики

Кафедра комп'ютерних наук

«Затверджую»
Завідувач кафедри
комп'ютерних наук,
кандидат технічних наук, доцент
(науковий ступінь, наукове звання)

Машкіна І.В.
(підпис) (прізвище, ініціали)

ЗАВДАННЯ НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

студенту групи _ІАСм-1-24-1.4д_

Кушнір Олег Володимирович
(прізвище, ім'я, по батькові)

Тема роботи Багаторозрядна цифрова індикація у вбудованих системах

1. Вихідні дані: Сучасні вбудовані системи, в яких використовуються багаторозрядні цифрові індикатори для відображення параметрів роботи, кодів помилок та службової інформації

2. Основні завдання: Сформувати вимоги до апаратно-програмного комплексу багаторозрядної цифрової індикації для вбудованих систем на базі Arduino Nano; Розробити варіант структурної схеми системи індикації, що включає Arduino Nano, модуль індикації та семисегментні індикатори, із виділенням трьох апаратних реалізацій модуля; Синтезувати електричні принципові схеми трьох варіантів модуля індикації на базі Arduino Nano у середовищі моделювання (Proteus) із урахуванням вимог до живлення та захисту; Реалізувати програмне забезпечення Arduino Nano, що забезпечує динамічне відображення чисел і тестових шаблонів на семисегментних індикаторах; Створити віртуальні стенди в середовищі Proteus для моделювання роботи трьох варіантів модуля індикації; Розробити програму експериментальних досліджень, яка включає тести на коректність відображення, частоту оновлення, стабільність

індикації та оцінку енергоспоживання для конфігурацій; Виконати серію експериментів на віртуальних стендах; Здійснити порівняльний аналіз трьох підходів до побудови багаторозрядної індикації; Сформулювати практичні рекомендації щодо вибору варіанта керування багаторозрядною цифровою індикацією;

3. Пояснювальна записка: Обсяг – до 131 стор. формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.

4. Графічні матеріали: презентація.

5. Додатки: не передбачено.

6. Строк подання роботи на кафедру « ___ » _____ 20 __ р.

Науковий керівник
__Доктор технічних наук. Професор_
(науковий ступінь, наукове звання)
____Бушма. О. В.__ ____
(прізвище, ініціали, підпис)

Завдання прийняв до виконання
« ___ » _____ 20__ р.
_____
(підпис студента)

Анотація кваліфікаційної роботи

Кваліфікаційна робота: 139с., рис. 39, табл 9, посилання 46

Актуальність:

Сучасний розвиток вбудованих систем охоплює широкий спектр застосувань: від промислової автоматизації до побутової електроніки та медичних пристроїв. Багаторозрядна цифрова індикація залишається ключовим елементом взаємодії користувача з системою, забезпечуючи зручне, наочне та оперативне відображення інформації. Зростаючі вимоги до енергоефективності, мініатюризації, надійності та інтеграції в складні інформаційні системи вимагають пошуку нових технічних рішень для реалізації багаторозрядної індикації у вбудованих пристроях. Розвиток мікроконтролерної техніки, спеціалізованих драйверів та алгоритмів програмного керування відкриває нові можливості оптимізації систем цифрової індикації в умовах обмежених апаратних ресурсів та складних експлуатаційних середовищ.

Об’єкт дослідження:

Багаторозрядна цифрова індикація у вбудованих систем.

Предмет дослідження:

Методи, алгоритми, апаратні компоненти та програмні засоби, що застосовуються для реалізації багаторозрядної цифрової індикації у вбудованих системах з урахуванням енергетичних, апаратних та функціональних обмежень.

Мета дослідження:

Розробити та обґрунтувати методи, технічні та програмні рішення для оптимізації багаторозрядної цифрової індикації у вбудованих системах з метою забезпечення високої енергоефективності, точності, надійності та швидкодії в умовах обмежених ресурсів.

Завдання:

- Сформувати вимоги до апаратно-програмного комплексу багаторозрядної цифрової індикації для вбудованих систем на базі Arduino Nano
- Розробити варіант структурної схеми системи індикації, що включає Arduino Nano, модуль індикації та семисегментні індикатори, із виділенням трьох апаратних реалізацій модуля
- Синтезувати електричні принципові схеми трьох варіантів модуля індикації на базі Arduino Nano у середовищі моделювання (Proteus) із урахуванням вимог до живлення та захисту.
- Реалізувати програмне забезпечення Arduino Nano, що забезпечує динамічне відображення чисел і тестових шаблонів на семисегментних індикаторах
- Створити віртуальні стенди в середовищі Proteus для моделювання роботи трьох варіантів модуля індикації
- Розробити програму експериментальних досліджень, яка включає тести на коректність відображення, частоту оновлення, стабільність індикації та оцінку енергоспоживання для конфігурацій
- Виконати серію експериментів на віртуальних стендах
- Здійснити порівняльний аналіз трьох підходів до побудови багаторозрядної індикації
- Сформулювати практичні рекомендації щодо вибору варіанта керування багаторозрядною цифровою індикацією

Методи дослідження: У роботі використано аналіз і узагальнення літератури з теми вбудованих систем та цифрової індикації. Застосовано порівняння різних варіантів схем на базі Arduino Nano, MAX7219, 74HC595 та HT16K33. Проведено комп’ютерне моделювання в середовищі Proteus з вимірюванням стабільності індикації й енергоспоживання, результати оформлено у вигляді таблиць і графіків.

Наукова новизна дослідження: У роботі в одному комплексі розглянуто та порівняно кілька підходів до керування багаторозрядною індикацією (MAX7219,

74HC595, HT16K33) саме для малопотужних вбудованих систем. Запропоновано прості критерії вибору варіанта (енергоспоживання, кількість виводів, складність реалізації, можливість розширення). На основі цих критеріїв побудовано й перевірено апаратно-програмний комплекс, який дає змогу обґрунтовано обрати оптимальне рішення.

Практичне значення дослідження: Розроблені схеми та програмні приклади можуть бути безпосередньо використані під час створення вбудованих пристроїв з багаторозрядною індикацією на базі Arduino Nano. Отримані результати та рекомендації допомагають зменшити енергоспоживання, раціонально використовувати виводи мікроконтролера та спростити подальше розширення пристрою. Матеріали роботи можуть бути також використані у навчальних лабораторних і курсових роботах.

Ключові слова:

Багаторозрядна індикація, вбудовані системи, мікроконтролери, мультиплексування, регістри зсуву, драйвери індикаторів, енергоефективність, програмна підтримка.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Arduino Nano	Мікроконтролерна плата на базі ATmega328P, основна апаратна платформа в роботі.
ATmega328P	8-бітний мікроконтролер сімейства AVR, що використовується на платі Arduino Nano.
AVR	Архітектура 8-бітних мікроконтролерів компанії Microchip (раніше Atmel).
GPIO	General Purpose Input/Output — універсальні цифрові лінії вводу-виводу мікроконтролера.
DIO	Digital Input/Output — цифровий вхід/вихід.
I²C	Дводротова послідовна шина обміну даними (Serial Bus), використовується, зокрема, для зв’язку Arduino Nano з HT16K33.
SPI	Serial Peripheral Interface — послідовний інтерфейс «ведучий–ведений», застосовується для обміну з MAX7219 та іншими модулями.
UART	Universal Asynchronous Receiver/Transmitter — інтерфейс

	асинхронного послідовного зв’язку (наприклад, Arduino–ПК).
PWM (ШІМ)	Pulse Width Modulation / широтно-імпульсна модуляція; спосіб керування яскравістю світлодіодів та потужністю навантаження.
LED	Light Emitting Diode — світлодіод, базовий елемент індикатора.
LCD	Liquid Crystal Display — рідкокристалічний індикатор.
VFD	Vacuum Fluorescent Display — вакуумно-люмінесцентний індикатор.
MAX7219	Спеціалізований драйвер для керування до 8 семисегментними індикаторами або LED-матрицями, з апаратним мультиплексуванням і обмеженням струму.
74HC595	8-бітний послідовно-паралельний регістр зсуву з вихідним latch; використовується для каскадного керування сегментами індикаторів.
HT16K33	I²C-драйвер світлодіодних матриць та семисегментних індикаторів з вбудованим мультиплексуванням і генератором частоти опитування.
TM1637	Спеціалізований драйвер для чотирирозрядних семисегментних

	індикаторів із власним послідовним протоколом.
IoT	Internet of Things — Інтернет речей, мережа взаємопов’язаних вбудованих пристроїв.
Proteus	Програмний пакет для моделювання електронних схем і мікроконтролерів, використаний для створення віртуальних стендів.
IDE Arduino	Integrated Development Environment Arduino — середовище розробки для написання, компіляції та завантаження скетчів на Arduino.
GND	Загальний провід («земля») електричної схеми.
VCC	Лінія живлення цифрових мікросхем (у роботі переважно +5 В).
DIN	Data In — вхід послідовних даних мікросхеми MAX7219.
CLK	Clock — вхід тактових імпульсів (для SPI чи регістрів зсуву).
LOAD / CS	LOAD / Chip Select — лінія вибору мікросхеми MAX7219 та фіксації (latch) переданих даних.
SER (DS)	Serial Data — послідовний вхід даних регістра 74HC595.

SH_CP	Shift Clock Pulse — тактовий вхід зсуву регістра 74HC595.
ST_CP	Storage Clock Pulse — тактовий вхід запам’ятовування вихідного регістра 74HC595 (latch).
SDA	Serial Data — лінія даних шини I²C.
SCL	Serial Clock — лінія тактування шини I²C.
GPIO-expander	Розширювач портів вводу-виводу — мікросхема, що збільшує кількість доступних GPIO мікроконтролера.
Мультиплексування розрядів	Метод послідовного ввімкнення окремих розрядів індикатора з високою частотою, щоб користувач бачив суцільне зображення без мерехтіння.
Регістр зсуву	Послідовно-паралельний пристрій, який дозволяє керувати багатьма лініями, використовуючи кілька послідовних входів.
Драйвер індикаторів	Спеціалізована мікросхема для керування світлодіодною індикацією, яка бере на себе частину функцій мікроконтролера.
Семисегментний індикатор	Світлодіодний індикатор, що складається з семи сегментів (та, як правило, десяткової точки) для відображення цифр.

Багаторозрядна індикація	Відображення числової інформації на кількох послідовних розрядах (цифрових позиціях).
Віртуальний стенд	Модель апаратної схеми у середовищі симуляції, що дозволяє проводити експерименти без фізичного макета.

ЗМІСТ

ВСТУП

Коли ми дивимось на електролічильник, кухонні ваги, лабораторний блок живлення чи простий термометр, зазвичай бачимо одне й те саме – кілька цифр на невеликому індикаторі. За цими цифрами ховається вся «розмова» пристрою з користувачем: поточне значення, режим роботи, код помилки, лічильник подій тощо. Якщо індикації немає або вона зроблена незручно, навіть цілком розумний пристрій перетворюється на незрозумілу чорну коробку. Сучасні вбудовані системи стали меншими, дешевшими й економнішими, але це не спрощує завдання. Доводиться рахувати кожен міліампер, кожен вивід мікроконтролера й думати, як часто оновлювати дані на екрані, щоб і не мерехтіло, і не перевантажувало систему. Це особливо помітно в пристроях, які працюють цілодобово, живляться від акумулятора або входять до складу більшої IoT-системи.

У роботі розбирається багаторозрядна цифрова індикація у вбудованих системах: як вивести кілька цифр чи простих символів так, щоб індикатор працював стабільно, був читабельним і не забирав зайвих ресурсів мікроконтролера. Описано конкретні апаратні рішення (регістрі зсуву, драйвери, схеми підключення) та програмні прийоми, які дозволяють реалізувати таку індикацію на типових мікроконтролерних платформах. Фактично тут розглядається доволі приземлене питання: як саме у вбудованому пристрої показувати числа, коли й виводів обмаль, і споживання струму потрібно тримати в рамках. Йдеться не про те, щоб під’єднати перший-ліпший індикатор, а про те, щоб порівняти кілька варіантів, подивитися, де кожен із них «просідає», де має перевагу, і в яких умовах його справді є сенс застосовувати. В одному випадку на перший план виходить економія енергії, в іншому — кількість необхідних виводів, в третьому — додаткове навантаження на мікроконтролер і зручність роботи з даними, і саме з огляду на ці речі формується остаточний вибір технічного рішення.

У процесі підготовки роботи довелося зробити кілька логічних кроків:

1. подивитися, які зараз існують способи побудови багаторозрядної індикації у вбудованих системах, які індикатори застосовують і як їх зазвичай підключають;

2. розібратися з апаратною частиною: регістри зсуву, різні драйвери індикаторів, варіанти мультиплексування розрядів;

3. окремо пройтися по програмних алгоритмах, які дозволяють керувати індикацією на «зажатому» по ресурсах мікроконтролері;

4. звернути увагу на сучасні тенденції: енергоефективні рішення, інтеграцію з IoT, більш «розумні» або адаптивні схеми відображення.

Підходи, які описані й протестовані в роботі, можна застосувати не лише в навчальних прикладах. Вони придатні для розробки нових вбудованих пристроїв, автоматизованих систем контролю, медичних приладів, систем моніторингу, вузлів у мережах IoT, де потрібно показати кілька параметрів простою й наочною індикацією.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТА СУЧАСНІ ПІДХОДИ ДО РЕАЛІЗАЦІЇ БАГАТОРОЗРЯДНОЇ ЦИФРОВОЇ ІНДИКАЦІЇ У ВБУДОВАНИХ СИСТЕМАХ

1.1 Актуальність багаторозрядної цифрової індикації у вбудованих системах

Щодня ми дивимося на цифри: час на електронному годиннику, покази лічильника, температуру в котлі чи кондиціонері. За цими простими цифрами ховається невелика електронна “начинка” – вбудована система, яка все вимірює, рахує й керує. Вона може стояти в пральній машині, у системі сигналізації, в медичному моніторі або в промисловому контролері. Людині при цьому важливо не знати всі технічні деталі, а просто бачити: пристрій працює, помиляється чи потребує уваги. Саме тому в будь-якій вбудованій системі потрібен зрозумілий спосіб показати стан пристрою, і найчастіше це робиться через цифрову індикацію. Коли треба відобразити одне число, завдання здається простим. Але в реальних пристроях одного розряду замало. Потрібно вивести кілька параметрів, коди помилок, час, режим роботи. Тому використовують багаторозрядні індикатори. На них з’являються значення тиску чи пульсу в медичному апараті, обсяг спожитої електроенергії в лічильнику, залишок часу до завершення програми в пральній машині. З боку користувача це виглядає як звичайний “рядок цифр”, але для розробника за цим стоїть окрема схема, таблиці відповідностей символів до сегментів і чимало коду.

Проблеми починаються, коли розрядів стає багато. Наприклад, 8-розрядний семисегментний індикатор у статичному режимі потребує 56 керуючих ліній. Для типового мікроконтролера на кшталт Arduino Nano це просто недосяжно – у нього стільки виводів немає. У реальних схемах такий підхід швидко впирається в обмеження, тому зазвичай переходять до інших рішень. Тому в реальних пристроях таке пряме підключення зазвичай не використовують. Коли виходів

мікроконтролера не вистачає, починають шукати інші варіанти. Найчастіше це мультиплексування розрядів, додавання зсувних регістрів (наприклад, 74HC595) або винесення індикації на окремі мікросхеми-драйвери, такі як MAX7219 чи HT16K33. У такій схемі контролер уже не “смикає” кожен сегмент окремо, а просто передає дані на допоміжні мікросхеми, а вони самі розподіляють сигнали по світлодіодах.

До апаратних обмежень додаються вимоги до зручності для людини. Цифри мають бути читабельними, без помітного мерехтіння й “розмазування”. Частоту оновлення потрібно підібрати так, щоб око сприймало картинку як стійку, а не як проблиски. Багато пристроїв живляться від батарей, тому зайві міліампери, витрачені на підсвічування індикатора, відразу скорочують час роботи. Під час власних експериментів з Arduino Nano я не раз стикався з ситуацією, коли красиво написаний код працював, але індикатор або тьмянів, або миготів, якщо невдало вибрати затримки й порядок опитування розрядів. Це добре показує, що індикація — це не лише про схему, а й про практичне налаштування. Окремий напрямок, де багаторозрядна індикація відчувається особливо, — це пристрої Інтернету речей. Маленький вузол десь у щитку чи на трубі не завжди має окремий екран або можливість постійно підключатися до смартфона. Невеликий цифровий дисплей часто залишається єдиним “вікном” у його стан: на ньому показуються поточні й сумарні покази, службові коди, іноді – рівень сигналу або стан мережі. Такі вузли одночасно обмежені по енергії, пам’яті, кількості виводів і тому особливо вимогливі до того, як саме організована індикація.

Сучасні мікроконтролери, з якими доводиться працювати на практиці (ESP32, STM32, Raspberry Pi Pico й інші), додають ще один вимір. У сучасних мікроконтролерів всередині багато всього зайвого на перший погляд: апаратні таймери, кілька шин SPI та I²C, DMA й інші вузли, які можна під’єднати до задачі індикації. Коли дивишся на перелік можливостей у даташиті, виникає спокуса “увімкнути все й одразу” і побудувати дуже хитру систему відображення. З практики видно, що такий спосіб швидко виявляється незручним. Тож виходить, що помітна частина часу й ресурсів мікроконтролера йде не на обробку корисних

даних, а просто на підтримку роботи індикатора. Схема стає складнішою, деталей і з’єднань більше, а користувач на панелі все одно бачить лише кілька цифр.

У цій роботі я розглядаю не загальну теорію індикації, а конкретну практичну задачу. Потрібно побудувати багаторозрядний індикатор так, щоб плата Arduino Nano могла з ним працювати при обмеженій кількості виводів і щоб схема не витрачала зайвий струм. При цьому індикація має залишатися зрозумілою для користувача й досить швидко оновлювати дані. Далі я послідовно випробовую кілька способів керування семисегментними індикаторами: пряме підключення, схему з 74HC595, варіант із драйвером MAX7219 та рішення на базі HT16K33. Для кожного з підходів окремо оцінюю, скільки ліній потрібно задіяти, з якою частотою оновлюються покази та яким є орієнтовне енергоспоживання. У підсумку це дозволяє не просто вибрати зручний варіант для конкретної схеми, а й побачити, які саме компроміси стоять за, на перший погляд, простою цифровою індикацією.

1.2 Основні поняття та класифікація цифрової індикації

У більшості сучасних приладів, де всередині працює мікроконтролер, без цифрової індикації вже не обходяться. Саме вона показує користувачеві числа, коди, повідомлення про стан чи помилки. Коли розрядів кілька, індикатор уже не просто показує одне число, а цілу “історію” про стан пристрою. На такому табло можна одночасно бачити поточне значення, код помилки, режим роботи тощо. Це зручно і в звичайній техніці на кухні, і в точних приладах, де важлива кожна цифра.

Під багаторозрядною цифровою індикацією я далі розумітиму “ланцюжок” окремих індикаторів, де кожен розряд відповідає за свою цифру або символ. Разом вони дають можливість показувати досить великі числа або значення з дрібною частиною. Наприклад, у вольтметрі можна не обмежуватися написом «10 В», а вивести 3–4 розряди і побачити, що там насправді 10,37 В, а не “десь біля десяти”.

Під час проектування індикації доводиться зважати на практичні речі: скільки місця є на передній панелі, чи працює прилад від мережі чи від батарей, які саме параметри треба показувати й з якою точністю. Від цих умов часто залежить і вибір типу дисплея. У реальних проєктах вибір зазвичай невеликий: або сегментні індикатори, або матриця, або якийсь маленький дисплей. Найчастіше перемагають звичайні семисегментні — вони коштують недорого, добре видно цифри й підключення не викликає особливого головного болю. Один розряд такого індикатора складається з семи окремих світлодіодних “паличок”. Змінюючи їхні комбінації, можна отримати цифри від 0 до 9. Досить часто додають ще одну “крапку”, щоб позначати десятковий роздільник. Завдяки цій простій будові семисегментні індикатори зустрічаються майже всюди: від духовок і ваг до мультиметрів, блоків живлення та медичних моніторів.

У багаторозрядній індикації можна виділити кілька “зрізів”, за якими зазвичай порівнюють різні рішення. Один із них — це сама технологія відображення. Тут маємо класичні світлодіодні індикатори (LED), рідкокристалічні панелі (LCD), вакуумно-люмінесцентні індикатори (VFD), невеликі OLED-дисплеї та інші варіанти. Інший важливий параметр — спосіб керування: індикатор може працювати в паралельному, послідовному або мультиплексованому режимі. Додається ще й суто практичний момент: скільки розрядів потрібно та як саме вся ця конструкція буде “підв’язана” до виводів мікроконтролера.

Найочевидніший варіант, з якого зазвичай починають, — пряме паралельне підключення. У цьому випадку кожен сегмент кожного розряду під’єднаний до окремого контакту. Для невеликої кількості цифр це ще працює, але в багаторозрядних системах число ліній дуже швидко зростає, і мікроконтролер банально не має стільки виводів, щоб усе це обслужити. Для багаторозрядних систем це майже відразу створює проблеми: портів мікроконтролера просто не вистачає. Послідовні методи, навпаки, дозволяють обійтися кількома лініями зв’язку — наприклад, за рахунок шин SPI, I²C або регістрів зсуву. У цьому

випадку апаратна частина ускладнюється за рахунок додаткових мікросхем, зате мікроконтролер звільняє багато виводів.

Окремо варто згадати мультиплексування. Ідея в тому, що всі розряди підключені до спільних ліній сегментів, а активується в кожен момент часу лише один розряд. Якщо швидко “перемикати” їх по колу, людське око бачить суцільну картинку, хоча насправді в кожну мить світиться тільки частина індикатора.Завдяки мультиплексуванню один і той самий набір ліній може обслуговувати кілька розрядів навіть на простих платформах на кшталт Arduino Nano чи ESP32. Головне — не завалити систему надто малою частотою оновлення та необережними затримками в коді, інакше індикатор почне помітно миготіти або “пливти”.

Ще одна корисна річ у таких системах — таблиці кодування символів (lookup tables). У них для кожної цифри або букви зберігається готовий шаблон: які саме сегменти треба ввімкнути. Це знімає з програміста багато рутини: не потрібно щоразу вручну виставляти біти, достатньо один раз скласти таблицю, а далі просто брати з неї потрібний шаблон. До того ж, за потреби можна швидко підлаштувати індикацію під інший алфавіт чи набір спеціальних знаків, не переписуючи всю логіку роботи індикатора.

Щоб не залишатися на рівні абстракцій, можна навести кілька типових прикладів використання багаторозрядної індикації:

- медицина – кардіомонітори, дефібрилятори, апарати контролю стану пацієнта;
- побутова техніка – мікрохвильові печі, пральні машини, електронні ваги, кухонні таймери;
- транспорт – приладові панелі автомобілів, прості бортові комп’ютери та модулі діагностики;
- промисловість – табло обліку енергоспоживання, модулі керування автоматизованими лініями, цифрові датчики тиску й температури.
-

Бачимо, що багаторозрядна індикація в реальних пристроях — це не якийсь “косметичний додаток”, а цілком робочий вузол. Без неї важко нормально показати результати вимірювань і поточний стан системи так, щоб користувач усе зрозумів з першого погляду. Кожен раз розробнику доводиться зважувати кілька речей одночасно: скільки енергії можна віддати на підсвічування індикатора, скільки місця він займає на платі та на передній панелі, з якою швидкістю треба оновлювати покази і який ресурс по строку служби мають мати обрані компоненти.

Через це питання “який саме тип багаторозрядної індикації вибрати” виникає ще на старті проєктування вбудованої системи. Від цього рішення залежить не тільки електрична схема й спосіб підключення, а й те, чи буде готовий пристрій зручним у користуванні, надійним у роботі й чи не доведеться надто часто повертатися до нього з ремонтом або доопрацюваннями.

1.3 Особливості роботи вбудованих систем, що впливають на реалізацію цифрової індикації

Питання багаторозрядної індикації завжди впирається в те, на якій “залізячці” все це працює. Вбудовані системи — це не універсальні ПК, а невеликі пристрої, які роблять під конкретні задачі: поміряти, ввімкнути, показати кілька параметрів тощо. Через це при проєктуванні постійно доводиться дивитися, скільки є виводів, який запас по живленню, скільки пам’яті доступно і чи потрібна робота в жорсткому режимі реального часу. Одне з перших обмежень, яке “вилазить” у практиці, — це кількість ліній введення-виведення.У багатьох мікроконтролерах, які ставлять у побутову чи промислову електроніку (той самий AVR ATmega328P на платах Arduino Uno або Nano), реально є близько 20 ліній GPIO. Якщо семисегментний індикатор підключати безпосередньо, один розряд забирає щонайменше сім ліній (без десяткової крапки), і вже кілька розрядів легко вибирають майже весь доступний запас. Друге суттєве обмеження стосується

пам’яті. Обсяги RAM і FLASH у мікроконтролерах невеликі, тому код, який відповідає за індикацію, доводиться писати економно. Зазвичай для цього використовують попередньо складені таблиці кодування символів, прості процедури роботи з портами та буфери відображення, щоб зайвий раз не звертатися до периферії. Окремо постає питання споживання енергії. Чимало вбудованих пристроїв живляться від батарей чи невеликих джерел живлення.Світлодіодні індикатори дають доволі помітне навантаження на живлення, особливо коли одночасно засвічено кілька розрядів або цифри типу «8», де працює більшість сегментів. Тому доводиться обмежувати яскравість, стежити за загальним струмом споживання, використовувати енергозберігаючі режими мікроконтролера і, де можливо, підключати спеціалізовані драйвери. У таких драйверах частина роботи виконується апаратно, тож вони частково розвантажують як порти мікроконтролера, так і вузол живлення.

Окреме питання — робота в режимі реального часу. У вимірювальних приладах, медичній апаратурі чи системах керування значення на індикаторі мають змінюватися без помітних “зависань”. Це змушує розподіляти пріоритети між основним алгоритмом і оновленням індикації. На практиці для цього зазвичай задіюють апаратні таймери, переривання, буфери відображення, а в складніших системах — ще й DMA, щоб передавати дані без постійної участі центрального ядра. Умови роботи теж відіграють свою роль. У промислових установках індикатори часто працюють при підвищеній температурі, в умовах вібрацій, пилу й перепадів напруги. Тому під час вибору конкретного типу індикатора доводиться зважати на те, наскільки стабільно він світить з часом, як тримає контраст і чи здатен працювати у широкому температурному діапазоні без помітної деградації. Це безпосередньо впливає на вибір елементної бази для системи індикації.

Сучасні вбудовані пристрої рідко обмежуються лише індикацією. Поруч працюють модулі зв’язку, блоки збору даних, вузли діагностики, контури автоматичного керування. Усі ці частини мають бути погоджені між собою, щоб індикація не “гальмувала” основні процеси й не створювала зайвих перешкод на живленні або шинах даних. У підсумку вибір схеми багаторозрядної індикації стає

окремим інженерним завданням. Тут уже мало просто “під’єднати індикатор” – потрібно зважити, скільки саме розрядів потрібно, який тип індикаторів обрано, що вміє конкретний мікроконтролер, який є запас по живленню і в яких умовах працюватиме пристрій.

Саме від цього залежить, чи вистачить простого підключення, чи доведеться додавати регістри зсуву, спеціалізовані драйвери або комбіновану схему. Від архітектури апаратної частини, у свою чергу, залежать і програмні рішення: які алгоритми оновлення індикації використовувати, як організувати мультиплексування та підтримати стабільну роботу багаторозрядного індикатора в конкретній вбудованій системі.

1.4 Методи реалізації багаторозрядної цифрової індикації

Коли мова заходить про багаторозрядну індикацію, перше питання, яке постає перед розробником: як саме підключати й “годувати” усі ці розряди. На схемі це виглядає як кілька однакових індикаторів, але способів організувати їх роботу насправді кілька. Вибір методу впливає і на кількість потрібних виводів мікроконтролера, і на енергоспоживання, і на складність програми.

1.4.1 Статичне керування індикацією

Найпростіше для розуміння багаторозрядну індикацію можна реалізувати статичним способом. У цьому випадку кожен сегмент кожного розряду підвішують безпосередньо на окремий вивід мікроконтролера. Сегменти світяться постійно, тому картинка на індикаторі не залежить від частоти оновлення й не виникає жодного мерехтіння, як це буває при динамічному керуванні. З програмної точки зору такий варіант теж максимально прямолінійний: щоб вивести цифру, достатньо виставити потрібні стани на порті згідно з маскою

сегментів. Саме тому статичне керування часто обирають на перших етапах вивчення мікроконтролерів або там, де головне — стабільна картинка на індикаторі, а не економія ресурсів. Проблема починається, коли розрядів стає більше. Кількість потрібних ліній керування росте дуже швидко. Для 4-розрядного семисегментного індикатора без крапки вже треба 28 виводів (4 × 7), а з десятковими крапками — 32. Для типового мікроконтролера на кшталт Arduino Nano на ATmega328P це практично межа його можливостей: більша частина портів буде “з’їдена” індикацією (Рис.1.1), і майже не залишиться місця для кнопок, датчиків чи інших вузлів.

У результаті статичне керування має сенс там, де індикаторів небагато і немає жорстких обмежень по кількості виводів та споживаному струму. Це можуть бути прості лабораторні макети, навчальні стенди чи допоміжна індикація, де важливіше не оптимізація, а наочність і простота реалізації.

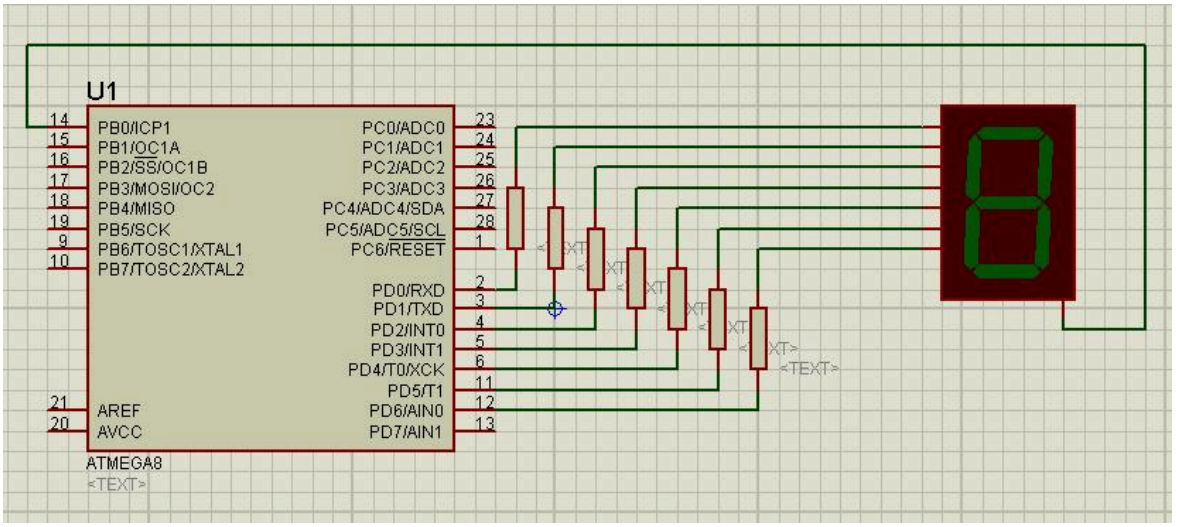

Рис. 1.1. Схема підключення статичного керування

1.4.2 Динамічне мультиплексування

У багаторозрядних індикаторах зараз найчастіше використовують динамічне мультиплексування. По-простому це виглядає так: у кожен момент часу реально горить тільки одна цифра. Мікроконтролер ставить на спільні сегментні лінії

потрібну комбінацію, вмикає один розряд, потім вимикає його і переходить до наступного. Якщо робити це з великою швидкістю, для ока всі розряди виглядають так, ніби світяться одночасно.

Один із головних плюсів такого підходу – економія виводів. Для чотирьох розрядів зазвичай достатньо 4 ліній для вибору розряду і 8 ліній для сегментів, разом виходить 12 виводів, а не кілька десятків, як у статичній схемі. Якщо розрядів вісім, картина все одно залишається прийнятною: 8 ліній керування розрядами і ті самі 8 сегментних, тобто 16 виводів — з цим більшість популярних мікроконтролерів уже нормально справляються. Щоб картинка на індикаторі виглядала стабільною, потрібно правильно підібрати частоту оновлення. З практики зручним діапазоном є приблизно 100–400 Гц. Якщо робити оновлення повільнішим, починає бути помітним мерехтіння, особливо при боковому зорі. Занадто висока частота теж не ідеальна: зростають вимоги до елементної бази, з’являються зайві електромагнітні перешкоди й додаткове навантаження на мікроконтролер. Програмно динамічна індикація виглядає складніше за статичну. Потрібно організувати постійний цикл оновлення: по таймеру або в перериванні переключати активний розряд і підставляти відповідні дані (часто це окрема функція або задача в системі реального часу). Проте сучасні бібліотеки, середовища розробки та приклади коду значно спрощують це завдання навіть для початківців(Рис.1.2).

Саме тому динамічне мультиплексування сьогодні дуже часто можна зустріти в побутових приладах, лічильниках, індустріальних панелях та компактних медичних пристроях: воно дозволяє застосувати багаторозрядну індикацію з обмеженою кількістю виводів і при цьому зберегти нормальну читабельність та якість відображення.

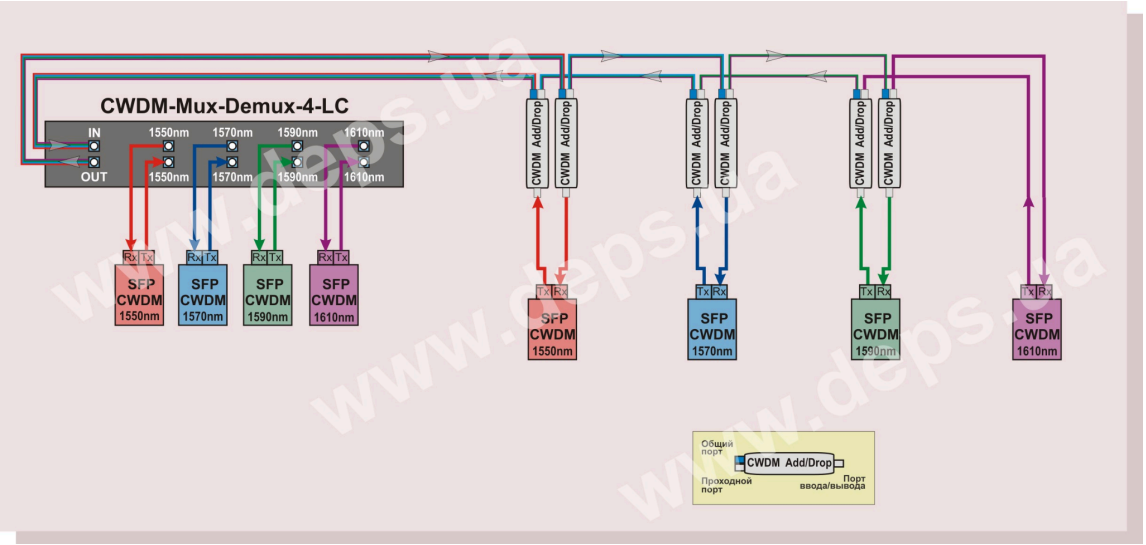

Рис. 1.2. Схема динамічного мультиплексування

1.4.3 Застосування регістрів зсуву

Коли розрядів стає багато, одного лише мультиплексування іноді вже не вистачає, особливо якщо хочеться ще більше заощадити виводи мікроконтролера. У таких схемах часто підключають регістр зсуву, найвідоміший варіант – мікросхема 74HC595.

Ідея роботи проста: мікроконтролер передає біти не на кожен вихід окремо, а послідовно в невеликий внутрішній буфер регістра. Для цього зазвичай потрібно всього три лінії: одна для даних, одна для тактових імпульсів і ще одна для “защіпки” (строба). Після того як байт (або кілька байтів) повністю записані, регістр одним моментом виставляє ці дані на свої паралельні виходи. Великий плюс такого підходу в тому, що регістри можна послідовно з’єднувати між собою, утворюючи ланцюжок. Кількість вихідних ліній при цьому легко нарощується — додаємо ще один 74HC595 і отримуємо плюс вісім виходів — а от число керуючих виводів мікроконтролера залишається тим самим. Це дозволяє будувати досить великі системи індикації й при цьому майже не витрачати додаткові виводи мікроконтролера. Виходи додаються за рахунок самих регістрів, а не за рахунок портів мікроконтролера . Є, щоправда, й своя ціна: робота з регістрами зсуву

ускладнює програмну частину. Потрібно формувати бітові маски для передачі, правильно “розкласти” сегменти по бітах, дотримуватися послідовності сигналів, стежити за затримками. Для невеликої кількості розрядів це не виглядає проблемою, але в швидкодіючих системах з великими індикаторними полями ці нюанси вже доводиться враховувати дуже уважно.

На практиці регістри зсуву часто використовують у інформаційних табло, системах моніторингу з великою кількістю каналів, на автоматизованих виробничих лініях — всюди, де потрібно показати багато даних одночасно, але немає бажання або можливості витрачати десятки портів мікроконтролера.

1.4.4 Використання спеціалізованих драйверів

У багатьох схемах немає великого змісту тягнути окремі проводи до кожного сегмента й добудовувати все транзисторами. Зручніше поставити одну мікросхему-драйвер, яка бере керування індикатором на себе. По суті, вона виступає “посередником”: мікроконтролер передає їй числа або коди, які треба показати, а вже драйвер розбирається, які сегменти увімкнути і який розряд активувати.

У цій роботі я використовую, зокрема, MAX7219. До нього можна під’єднати до восьми семисегментних індикаторів, а з боку мікроконтролера вистачає кількох послідовних ліній. Всередині мікросхеми вже налаштовані мультиплексування, обмеження струму та регулювання яскравості, тому в програмі не потрібно окремо прораховувати послідовність увімкнення кожного сегмента.Тому в програмі фактично залишається тільки передати число, а про те, як саме його “розкласти” по сегментах, мікроконтролер подбає сам. MAX7219 особливо активно застосовується у промислових панелях, годинниках, інформаційних табло та багатьох аматорських проєктах(Рис.1.3).

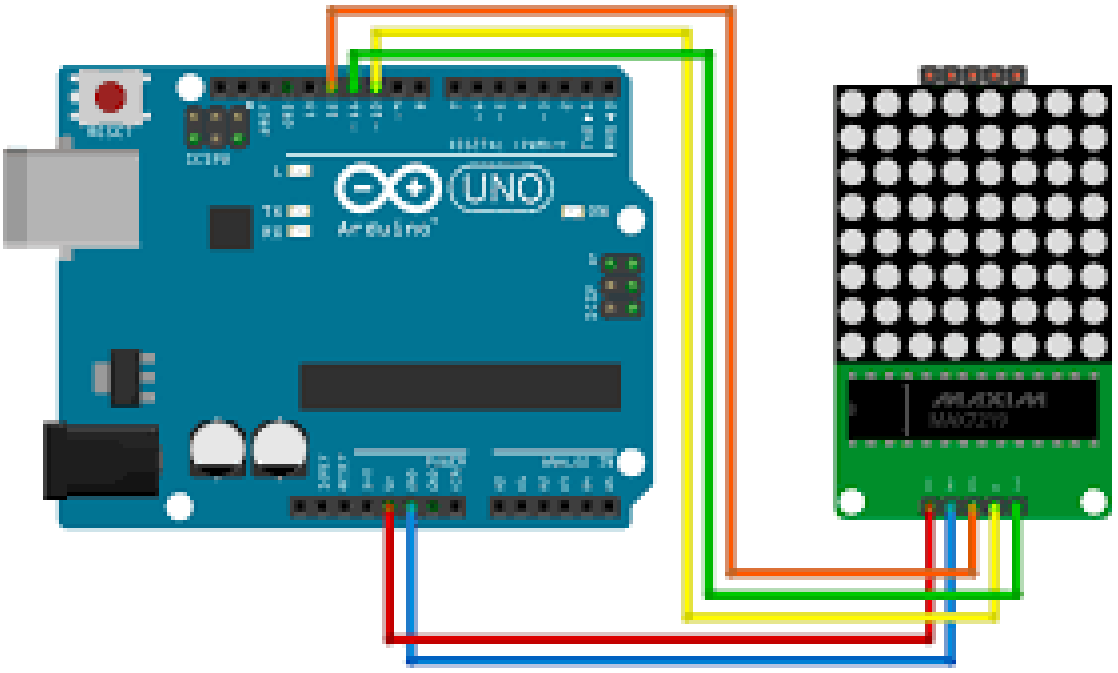

Рис. 1.3. Схема підключення MAX7219

Ще один драйвер, з яким я працював, — це TM1637. Зазвичай він уже стоїть на маленькій платі разом із чотирма семисегментними індикаторами, тож усе виглядає як готовий модуль. Щоб під’єднати його до мікроконтролера, мені вистачало двох проводів: один для передавання даних, другий – для тактування (Рис.1.4). Саме через таку простоту такі модулі часто бачимо в електронних годинниках, кухонних таймерах та подібній побутовій техніці.

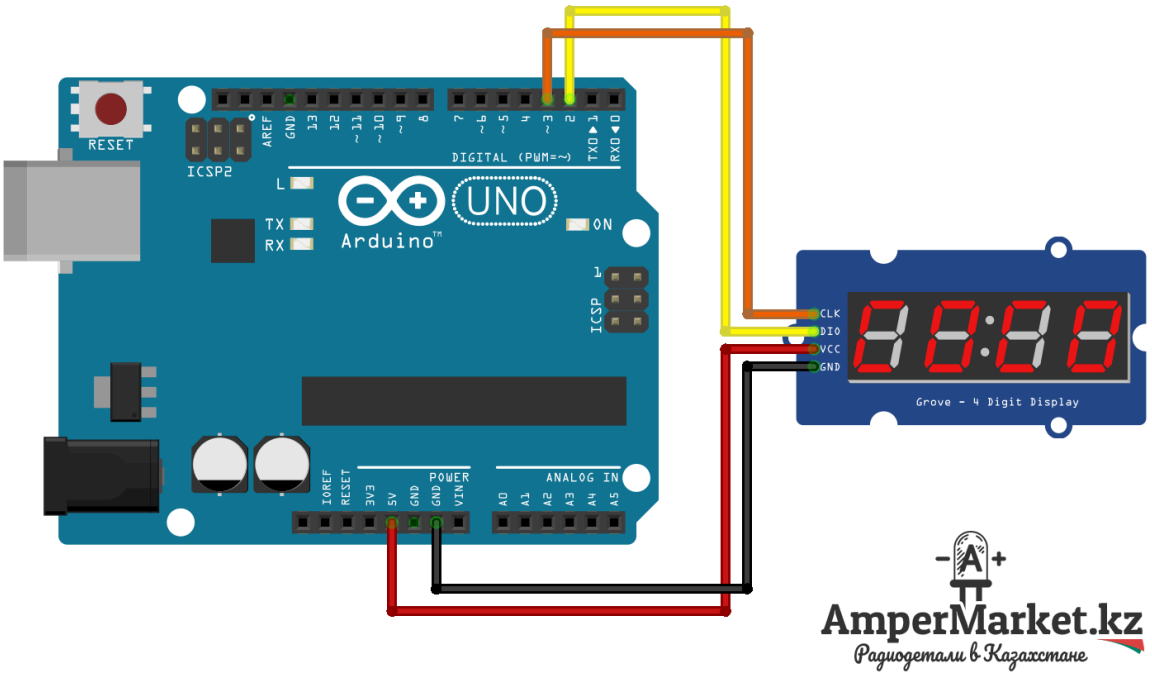

Рис. 1.4. Схема підключення TM1637

Ще один контролер, який я використовую в роботі, – це HT16K33. Він спілкується з мікроконтролером через звичайну шину I²C, тому для підключення достатньо двох ліній зв’язку разом із живленням. На такому чипі можна побудувати як індикацію на семисегментних індикаторах, так і керування невеликою світлодіодною матрицею. Окремо зручно, що всередині вже є налаштування яскравості та режими, які зменшують помітне мерехтіння. Такі драйвери дають змогу зібрати складнішу індикацію без великої кількості додаткових деталей навколо мікроконтролера. Вони частину роботи беруть на себе, допомагають не “роздувати” споживання енергії й роблять поведінку пристрою більш стабільною під час тривалої роботи, що важливо для серійних виробів і тих систем, до яких не хочеться постійно повертатися з ремонтом.

1.4.5 Апаратна підтримка сучасних мікроконтролерів

У новіших мікроконтролерних платформах можливості для роботи з індикацією стали помітно ширшими. Такі контролери, як STM32, ESP32,

Raspberry Pi Pico, окрім стандартних портів вводу-виводу, мають цілу низку корисної периферії: апаратні таймери, генератори PWM, DMA-контролери. Усе це можна залучити, щоб частину роботи з мультиплексування перекласти з програми на «залізо». Наприклад, за допомогою DMA можна налаштувати автоматичну передачу заздалегідь підготовлених масивів даних до портів, майже без участі основного ядра. А таймери в цей час формують імпульси для послідовного оновлення розрядів, тримаючи потрібну частоту оновлення. У результаті основна програма менше відволікається на обслуговування індикації, частота оновлення залишається стабільною навіть тоді, коли мікроконтролер зайнятий іншими обчисленнями, а споживання енергії та узгодженість роботи індикаторів з іншими модулями системи виходять на більш керований рівень.

1.4.6 Комбіновані методи реалізації

У реальних проєктах рідко обмежуються одним-єдиним методом керування індикацією. Часто використовують комбінацію підходів. Наприклад, у великому інформаційному табло частина розрядів може обслуговуватись через регістри зсуву, центральний блок – через спеціалізований драйвер, а сам мікроконтролер стежить за синхронізацією всієї системи й задає загальний ритм оновлення. Такий змішаний підхід дає більше свободи при проєктуванні: десь можна зекономити на залізі, десь – спростити монтаж, а десь – підняти надійність, не ускладнюючи схему без потреби. Це особливо помітно в промисловій автоматизації, лабораторному обладнанні, енергетичних щитах та сучасних IoT-пристроях, де одночасно важливі й обсяг відображуваної інформації, і гнучкість побудови всієї системи.

1.5 Програмна підтримка багаторозрядної цифрової індикації

Коли говоримо про багаторозрядну індикацію, важливо думати не тільки про схему, а й про те, як її обслуговує програма. Спосіб підключення може бути різним, але в будь-якому випадку саме код готує числа до відображення, оновлює розряди, стежить за зайнятістю мікроконтролера і за тим, щоб картинка на індикаторі не “сіпалась”, а виглядала стабільно. Починати зазвичай доводиться з найпростішого — перетворення чисел на набір сегментів. Цифри від 0 до 9 отримуються різними комбінаціями семи сегментів. Щоб не виписувати ці комбінації щоразу вручну, у програмі заводять невелику таблицю: для кожної цифри там лежить готова бітова маска, яка показує, які сегменти треба ввімкнути. Далі алгоритм прямолінійний: число розбивається на окремі цифри, для кожної з них із таблиці береться свій “шаблон”, і ці дані вже передаються в модуль, що керує індикацією. Після цього постає питання, як саме перемикати розряди. Динамічне мультиплексування зазвичай оформлюють або як періодичну функцію, або як обробник переривання від таймера. Через однакові проміжки часу активується наступний розряд, на спільні лінії сегментів виставляються його дані, потім черга переходить далі. Для 4-розрядного індикатора, наприклад, можна налаштувати оновлення кожного розряду раз на 250 мкс — у підсумку виходить близько 1 кГц по циклу, і при такій швидкості око вже сприймає всі розряди як рівномірно підсвічені, без помітного мерехтіння.

Оскільки мікроконтролер зазвичай не займається тільки індикацією, доводиться думати й про економію обчислювальних ресурсів. Паралельно можуть виконуватися читання датчиків, обробка вимірювань, обмін по UART чи I²C, керування реле, двигунами тощо. Тому код, який відповідає за індикацію, бажано робити неблокуючим: не використовувати довгі delay, а працювати через таймери, переривання, кільцеві буфери. Зручно мати окремий буфер відображення, куди

основна програма записує підготовлені цифри, а задача індикації просто читає з нього й виводить на розряди.

Якщо в схемі використовуються регістри зсуву, наприклад 74HC595, до логіки додається ще один шар. Потрібно не лише визначити, які сегменти мають світитись, а й “загорнути” це в послідовний бітовий потік для передавання через SPI-подібний інтерфейс. Для цього зазвичай пишуть окрему функцію, яка формує байт (або кілька байтів) для активного розряду, подає біти в певному порядку й синхронізує їх з тактовими імпульсами. Якщо регістрів кілька й вони з’єднані “ланцюжком”, доводиться враховувати і час, потрібний на передачу всієї послідовності, і паузу на стабілізацію виходів після строба.

У випадку зі спеціалізованими драйверами (MAX7219, TM1637, HT16K33) програмна частина, навпаки, стає простішою. Такі мікросхеми мають власну внутрішню пам’ять відображення: кожен розряд відповідає певному регістру всередині драйвера. Завдання програми зводиться до того, щоб через SPI, I²C або свій двопровідний протокол передати нові значення цих регістрів. Драйвер далі сам бере на себе мультиплексування, формування струмів сегментів і підтримання обраної яскравості.

Окреме питання — регулювання яскравості. Тут можливі два варіанти: або скористатися готовими можливостями самого драйвера (як це робиться в MAX7219 чи HT16K33), або реалізувати керування яскравістю через широтно-імпульсну модуляцію (PWM) на лініях живлення чи керування. Зменшуючи тривалість імпульсу, можна знизити середній струм через індикатор, а отже, трохи “полегшити” блок живлення й продовжити ресурс світлодіодів.

Ще один програмний момент — спеціальні символи: десяткова крапка, символи помилки, літерні позначення, умовні індикатори режимів. Для них у таблиці сегментів зазвичай відводять окремі маски або додаткові біти (наприклад, окремий біт для крапки). Це дозволяє гнучко поєднувати цифри й додаткові індикатори, не ускладнюючи основну логіку.

Коли система стає складнішою, а задач багато, зручно переходити до операційної системи реального часу (FreeRTOS, Zephyr тощо). В цьому випадку оновлення індикації оформлюють як окрему задачу з власним пріоритетом, а інші задачі займаються опитуванням датчиків, обробкою даних, зв’язком тощо. Такий підхід дозволяє чітко розвести функції, уникнути блокувань і легше масштабувати проєкт.

Корисний прийом, подвійна буферизація. Один буфер використовується для виведення (з нього “читає” задача індикації), інший у цей час заповнюється новими даними. Коли оновлення завершене, буфери просто міняються місцями. Це дає змогу уникнути ситуацій, коли частина розрядів уже оновлена, а частина ще показує старі значення. Саме грамотна програмна частина робить систему індикації живою: допомагає повністю використати можливості апаратури, не перевантажити мікроконтролер, зберегти стабільність відображення й адекватне споживання енергії. Добре продумана взаємодія між “залізом” і кодом дозволяє реалізувати багаторозрядну індикацію так, щоб вона природно вписувалася в роботу всього вбудованого пристрою, а не заважала йому.

1.6 Апаратна підтримка багаторозрядної цифрової індикації у сучасних вбудованих системах

Апаратна частина індикації – це те, з чим доводиться мати справу руками: які саме індикатори взяти, як їх живити, через що підключити до мікроконтролера. Від цих рішень залежить, чи буде система працювати стабільно, наскільки вона вийде “ненажерливою” по струму і чи не стане плата схожою на клубок проводів. Сучасні компоненти дають багато варіантів, тому перед початком розробки доводиться визначитися з базовою схемою.

Перший елемент, про який завжди думаєш, – це сам індикатор. У більшості простих пристроїв це звичайні семисегментні модулі. Вони складаються із семи світлодіодних “паличок”, з яких складається цифра, і часто мають окрему крапку

для дробової частини. Вони коштують недорого, цифри на них читаються без напруження, тому їх можна зустріти і в звичайній побутовій техніці, і на щитках керування, і в лабораторних блоках живлення. Коли ж одних цифр уже недостатньо й потрібно виводити літери або короткі підписи, використовують 14- чи 16-сегментні індикатори.У них сегментів більше, і за рахунок цього виходять літери, службові символи й прості піктограми. Для ще складніших задач (табло, інформаційні панелі) часто використовують матриці 8×8, 16×16 та подібні – там вже можна виводити будь-який символ або маленьке “зображення” за рахунок окремих світлодіодів у клітинках.

Окреме питання, як правильно обмежити струм через сегменти. Світлодіоди не люблять “пряме” підключення до джерела живлення. Якщо не ставити обмеження, вони легко перегріваються і швидко виходять з ладу. У найпростішому варіанті на кожен сегмент ставлять резистор. Наприклад, при живленні від 5 В і падінні напруги на світлодіоді близько 2 В резистор 150–220 Ом дає струм порядку 10–20 мА – цього зазвичай достатньо, щоб сегмент світився нормально й не перегрівався. Але коли розрядів багато, сумарний струм індикації стає помітним, і це треба враховувати при виборі блоку живлення.

Ще один момент, навантаження на порти мікроконтролера. Якщо напряму вішати багато сегментів, можна легко перевищити допустимий струм на вивід і “вбити” мікросхему. Щоб цього уникнути, між мікроконтролером і індикаторами часто ставлять транзисторні ключі. Це можуть бути окремі NPN/PNP-транзистори (Рис.1.5) або MOSFET’и, які вмикають спільні аноди/катоди розрядів чи цілі групи сегментів. У цьому випадку порт мікроконтролера керує лише базою чи затвором, а основний струм тече через транзистор.

Коли ключів потрібно багато, зручно використати готову “збірку” на кшталт ULN2803. Це мікросхема, в якій в одному корпусі вже є кілька транзисторних каскадів з потрібними резисторами. Що спрощує розводку плати й дозволяє підключати одразу кілька ліній індикації, не думаючи про кожен окремий транзистор.

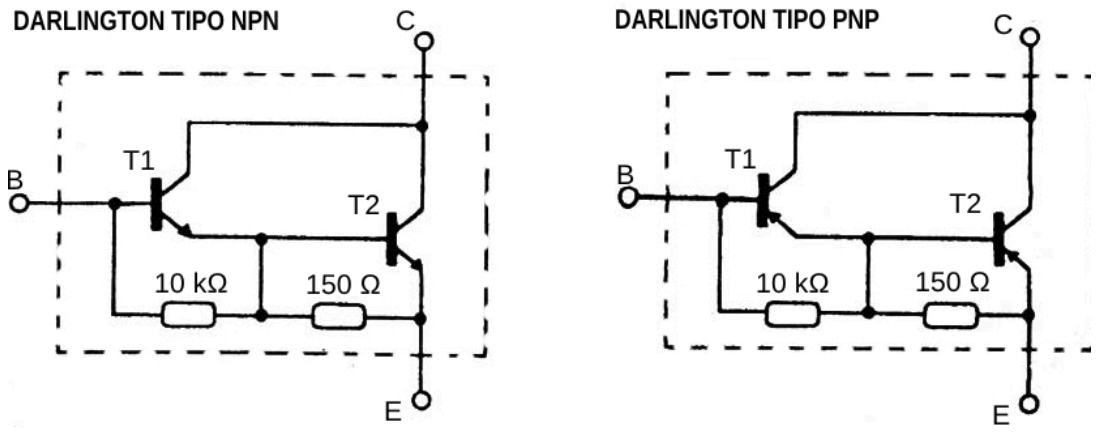

Рис. 1.5. Схема живлення через ULN2803 транзистори

Окремий вузол у багаторозрядній індикації – це мікросхеми, які допомагають “перемикати” розряди. До них належать мультиплексори та демультиплексори. Такі мікросхеми фактично працюють як “перемикачі”: частину ліній можна зробити спільними, а не тягнути окремий провід до кожного розряду чи сегмента. У підсумку на платі менше доріжок і деталей, а схема виглядає простіше й легше в налагодженні.

Щоб ще не ускладнювати все транзисторами й додатковими ключами, у багатьох випадках просто ставлять готовий драйвер індикатора. Усередині такої мікросхеми вже закладені базові речі — перемикання розрядів, обмеження струму, налаштування яскравості та прості захисні функції, тож це не треба реалізовувати окремо в схемі. Це дозволяє не “винаходити велосипед” щоразу з нуля.

Найчастіше в практиці зустрічаються, зокрема:

- MAX7219 – драйвер для восьми семисегментних розрядів із послідовним інтерфейсом, сумісним зі SPI. Кілька таких мікросхем можна об’єднувати в ланцюжок, отримуючи до 64 розрядів при мінімальній кількості ліній керування. Усередині чипа вже є 16 рівнів яскравості та всі необхідні обмеження по струму.

- TM1637 найчастіше застосовують у вузлах з чотирма семисегментними LED-індикаторами. Мікросхема сама опитує розряди, може змінювати яскравість, а з боку мікроконтролера їй потрібно всього дві лінії — DIO та CLK. Дані передаються послідовно по простому двопровідному протоколу, тому кількість задіяних виводів залишається мінімальною.
- HT16K33 зручно використовувати там, де одного-двох індикаторів уже мало і потрібно вивести більше інформації. Він під'єднується до мікроконтролера по стандартній шині I²C, тому окремі лінії керування під нього виділяти не потрібно – вистачає тієї ж пари проводів, що й для іншої периферії. Один такий контролер може керувати кількома семисегментними індикаторами або невеликою світлодіодною матрицею (у сумі до 128 світлодіодів). Усередині мікросхеми є власні регістри відображення, реалізоване апаратне сканування розрядів і зміна яскравості за допомогою ШІМ, тож більшість "рутинної" роботи з індикацією він бере на себе.

Коли в схемі використовують подібні драйвери, індикаторна частина стає простішою в реалізації: зменшується кількість дрібних елементів, легше розвести плату й менше шансів "помилитися доріжкою". Частину рутинних дій із оновлення індикації бере на себе сам драйвер, а мікроконтролер може більше ресурсів віддати основним задачам пристрою, а не постійному обслуговуванню сегментів. На етапі підключення індикаторів дуже швидко з'являється питання живлення. Якщо одночасно засвітити багато сегментів, струм помітно зростає – не мікроампери, а вже десятки, а інколи й сотні міліампер. Якщо блок живлення або стабілізатор підібрані "впритул", напруга починає просідати, індикатори тьмяніють, а деякі деталі на платі відчутно гріються. Тому для схем із багаторозрядною індикацією вузол живлення зазвичай продумують окремо: закладають запас по струму, дивляться, який тип стабілізації підійде, і додають хоча б елементарний захист від перевантаження. Коли мова йде про пристрої, що працюють від батарейки або невеликого акумулятора, вимоги ще жорсткіші. Тут уже не влаштовує індикатор, який "з'їдає" десятки міліампер тільки на підсвітку.

У таких випадках часто переходять на рідкокристалічні (LCD) дисплеї. Вони можуть працювати на дуже малих струмах, тому добре підходять для портативних приладів, носимих пристроїв чи простих медичних датчиків, які повинні працювати довго без підзарядки.

Окрім класичних семисегментних і LCD-індикаторів, у реальних проєктах усе частіше зустрічаються й інші типи дисплеїв. Наприклад, OLED дає яскраву і контрастну картинку, яку добре видно під різними кутами, тож його ставлять там, де треба показати не лише числа, а й невеликі піктограми чи просту графіку. Електронно-паперові (e-ink) дисплеї, навпаки, майже не споживають енергію, поки зображення не змінюється, тому їх зручно використовувати в пристроях, де інформація оновлюється рідко, але має залишатися читабельною годинами або днями.

У готовому пристрої індикація завжди “зав’язана” на інші вузли. Вона підключена до того ж мікроконтролера, сидить на тих самих шинах UART / I²C / SPI, живиться від спільного джерела разом із датчиками та іншою електронікою. Якщо недооцінити, скільки струму забирають індикатори, легко отримати ситуацію, коли напруга просідає, з’являються зайві завади або мікроконтролер починає працювати нестабільно через затримки й помилки в роботі програми. Тому, коли проектується апаратна частина багаторозрядної індикації, доводиться одночасно думати про кілька речей: який тип індикатора вибрати, скільки виводів реально є в мікроконтролера, скільки струму може видати система живлення, наскільки надійним має бути пристрій і скільки він у підсумку може коштувати. Якщо ці моменти узгоджені між собою, індикація працює передбачувано й довго, а не тільки “гарно виглядає” в схемі на папері.

1.7 Перспективи розвитку багаторозрядної цифрової індикації у вбудованих системах

За останні роки вбудовані системи помітно змінилися, і разом із ними змінилася й індикація. Багаторозрядні індикатори вже важко сприймати як другорядну деталь на схемі. У вбудованих пристроях індикація — це не просто прикраса.Те, як саме організована індикація, впливає одразу на кілька речей. Те, як зроблена індикація, напряму впливає і на людину, і на сам прилад. Користувач або з першого погляду розуміє, що йому показують цифри, або кожного разу “вгадується”, де що написано. Те саме з живленням: дисплей може тихо тягнути струм постійно, і тоді про нормальну автономну роботу вже не йдеться, пристрій усе час тягнеться до зарядки.Зараз більшість невеликих приладів живиться від батарей чи компактних акумуляторів, тому про економію енергії доводиться думати ще на самому початку проєкту, а не коли схема й плата вже готові. Якщо індикатор їсть зайвий струм, час роботи помітно падає, навіть якщо решта схеми досить ощадлива. Через це розробники часто дивляться в бік рідкокристалічних (LCD) і електронно-паперових (e-ink) дисплеїв. У ситуаціях, коли зображення змінюється рідко, такі екрани можуть дуже довго тримати картинку при мінімальному споживанні. Це підходить для приладів, де значення оновлюються не щомиті, а, наприклад, кілька разів на годину чи ще рідше. Паралельно дуже широко пішли в хід органічні світлодіодні дисплеї OLED. Вони дають яскраву, контрастну картинку, яку нормально видно під різними кутами, навіть якщо дивитися збоку, а не прямо. Тому такі екрани зараз ставлять і в телефони, і в елементи розумного дому, і в невеликі вимірювальні чи діагностичні прилади, і в різну носиму електроніку. Проблема з їх повільним “старінням” усе ще є, але з кожним новим поколінням панелей виробники трохи підтягують матеріали та саму технологію. У результаті для багатьох реальних пристроїв строк служби OLED-дисплея вже вважають цілком прийнятним. Окремо варто згадати системи,

які працюють у складі Інтернету речей. Там індикація все частіше поєднується з мережевими можливостями: той самий дисплей і показує дані користувачу, і дозволяє на місці подивитися стан вузла, помилки чи службову інформацію, не підключаючись щоразу до комп'ютера або сервера. З іншого – зручний спосіб прямо на місці подивитися стан модуля, помилки чи сервісні коди без підключення ноутбука або сервера. У таких приладах дисплей часто виконує подвійну роботу. З одного боку, він показує користувачу потрібні величини — напругу, температуру, лічильник енергії тощо. З іншого боку, на ньому ж зручно подивитися службову інформацію: чи є зв'язок, чи немає помилок, який зараз режим роботи. Подібний підхід багато де вже використовується: у вузлах розумного міста, лічильниках, модулях енергомоніторингу або клімат-контролю, де багаторозрядна індикація дозволяє буквально за кілька секунд оцінити стан пристрою прямо на місці. Паралельно з такими простішими рішеннями з'являються й більш "розумні" варіанти індикації, які намагаються враховувати контекст. У частині систем починають застосовувати елементи штучного інтелекту: спосіб відображення змінюється залежно від освітлення, важливості показника чи того, наскільки завантажений оператор. У медичних приладах це може виглядати так: звичайні параметри виводяться звичайним шрифтом, а критичні значення додатково підсвічуються, виділяються кольором або виносяться в окрему область екрана, щоб їх не доводилося шукати серед великої кількості цифр. Паралельно йде процес мініатюризації. Індикаторні вузли стають меншими, і їх усе частіше вбудовують у сенсорні модулі, переносну електроніку, імплантовані медичні пристрої та портативні монітори. Є рішення, де матриця індикації поєднана з сенсорним шаром, тобто одна й та сама поверхня і показує інформацію, і реагує на дотик. Для промислових панелей чи транспортних систем, де вільного місця традиційно мало, це дає відчутний виграш. Окремий напрям задають гнучкі, прозорі та вигнуті дисплеї. Вони дозволяють розміщувати індикацію там, де раніше її взагалі не планували: на вигнутих панелях автомобіля, частинах корпусу побутової техніки або на елементах конструкції, які одночасно виконують декоративну й робочу функцію.

Усі ці зміни відбиваються й на апаратній частині. Виробники мікросхем рухаються в бік універсальніших драйверів індикації, які можуть однаково нормально працювати і з класичними семисегментними дисплеями, і з різними варіантами матричних чи комбінованих індикаторів. Такі чипи беруть на себе основну рутину: мультиплексування, обмеження струму, керування яскравістю, іноді навіть просту діагностику ліній. Для мікроконтролера це означає менше навантаження та більше ресурсів на основні функції пристрою. З програмного боку рух іде в бік більш акуратних і продуманих алгоритмів. Зараз поступово з’являються більш продумані програмні прийоми. Частина сучасних рішень для індикації просто прибирає типові недоліки мультиплексування: зменшує помітне мерехтіння, підбирає стабільний режим роботи сегментів. Частина алгоритмів просто стежить за тим, щоб яскравість індикаторів помітно не змінювалась, коли “гуляє” напруга живлення. Те саме і з частотою оновлення: якщо дані майже не змінюються, немає сенсу постійно перезаписувати індикатор на максимальній швидкості, це лише зайве навантажує систему. Найбільше такі речі помітні там, де система працює в реальному часі. Наприклад, у вимірювальних приладах, які сьогодні стоять у холодному цеху, а завтра — біля гарячого обладнання. При зміні температури світлодіоди й LCD-дисплеї поводяться трохи по-іншому, і індикація мусить це “переварити”, щоб не поплила яскравість чи контраст. Якщо подивитися ширше, розвиток багаторозрядної індикації рухається разом із розвитком самих вбудованих систем. З одного боку, намагаються менше витрачати енергії й витиснути максимум із батареї, з іншого — показати на невеликому екрані більше корисних даних і при цьому не заплутати користувача. Паралельно змінюється і елементна база, і підходи до програмування. У результаті навіть відносно прості пристрої з кількома розрядами на панелі зараз намагаються проєктувати так, щоб індикаторний вузол можна було легше підлаштувати під реальні умови роботи, а не переробляти щоразу з нуля.

1.8. Висновки до розділу 1

У першому розділі зібрана теоретична база, без якої далі працювати з багаторозрядною індикацією було б важко. Спочатку йдеться про те, де взагалі зараз зустрічаються такі індикатори: від побутових приладів і промислових щитків до медичної апаратури та простих IoT-пристроїв. Після цього вводяться основні терміни й робиться поділ індикаторів на основні групи: сегментні, матричні, графічні, LCD, LED, VFD, OLED. Окремо виділено саме семисегментні індикатори, бо далі в роботі вони є головним об’єктом аналізу. Далі розділ переходить до того, які обмеження задає типова вбудована платформа: скільки є GPIO, яка ситуація з пам’яттю, чи потрібно працювати в режимі реального часу і які є рамки по енергоспоживанню. Пояснюється, як ці фактори впливають на те, яку схему керування індикатором можна дозволити. На цьому фоні порівнюються основні підходи до реалізації багаторозрядної індикації: просте статичне підключення, динамічне мультиплексування, варіанти з регістрами зсуву (74HC595) і спеціалізованими драйверами MAX7219, TM1637, HT16K33, а також комбіновані схеми, де поєднано кілька методів. Окремим блоком коротко описано програмну сторону: таблиці кодування символів, організацію циклів оновлення індикації, використання таймерів і переривань, керування яскравістю, буферизацію даних. Поруч із цим розглянуто базові апаратні рішення: як обмежують струм через сегменти, коли потрібні транзисторні ключі, які типові варіанти живлення застосовують для індикаторів. Наприкінці розділ показує основні тенденції розвитку багаторозрядної індикації у вбудованих системах і пояснює, чому в подальших розділах увага зосереджується саме на порівнянні схем на базі Arduino Nano з використанням MAX7219, 74HC595 та HT16K33.

РОЗДІЛ 2. ПРОЄКТУВАННЯ АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ БАГАТОРОЗРЯДНОЇ ЦИФРОВОЇ ІНДИКАЦІЇ

2.1 Постановка завдання розробки

У вбудованих пристроях індикація цифр нікуди не зникла. Попри появу недорогих графічних LCD та OLED, звичайні багаторозрядні семисегментні індикатори й далі часто ставлять там, де важливі простота, надійність, нормальна читабельність і невисока ціна. До таких випадків належить і модуль індикації, який я роблю в цій роботі. Досить швидко вилізає типова проблема: багаторозрядний семисегментник вимагає багато ліній керування. Якщо кожен сегмент кожного розряду вішати прямо на виводи мікроконтролера, кількість задіяних контактів росте дуже швидко. Уже кілька розрядів “з’їдають” більшість портів, і для датчиків, кнопок чи інших модулів місця майже не лишається. Для недорогих плат на кшталт Arduino Nano це стає відчутним обмеженням. Під час роботи над дипломом саме з цим довелося зіткнутися в першу чергу: треба зробити зручну багаторозрядну індикацію, але не заблокувати собі всі виводи мікроконтролера й не вийти за розумні межі по споживанню струму. Тому завдання формулюється не як “просто підключити індикатор”, а як знайти та реалізувати такий спосіб керування, який дозволить мати достатню кількість розрядів, економно використовувати GPIO і не робити схему надто “ненажерливою”.

У рамках цієї роботи я ставлю собі таку практичну задачу:

- спроєктувати модуль багаторозрядної цифрової індикації на базі Arduino Nano;

- реалізувати його щонайменше у трьох апаратних варіантах:
  - варіант зі мікросхемою-драйвером HT16K33
  - варіант зі спеціалізованим драйвером MAX7219;
  - варіант із каскадом регістрів зсуву 74HC595 з подальшим мультиплексуванням розрядів;
- написати код для обох варіантів так, щоб індикатор працював без помітного мерехтіння, а цифри було нормально видно;
- підготуватися до подальшого порівняння цих схем: наскільки їх складно зібрати й запрограмувати, скільки ресурсів мікроконтролера вони забирають, як вони поводяться зі споживанням струму і чи легко їх потім розширювати.

2.2. Опис апаратної частини

Апаратна частина багаторозрядної індикації — це все “залізо”, яке змушує цифри з’являтися на дисплеї. Сюди входить мікроконтролер, мікросхеми керування, самі індикатори, вузол живлення, елементи захисту та обв’язка. Якщо ці компоненти підібрані невдало або погано узгоджені між собою, починаються типові проблеми: індикатор мерехтить, гріються елементи, просідає напруга, цифри світяться нерівномірно чи погано читаються.

У цій дипломній роботі апаратну частину я будував із кількома простими вимогами:

- щоб схему можна було повторити в навчальній лабораторії;
- щоб на одному стенді показати три різні способи керування індикацією (через MAX7219, через 74HC595 і через HT16K33);
- щоб цю ж схему можна було проганяти як у середовищі моделювання, так і на реальному стенді, порівнюючи результати.

Далі по черзі розглядаються основні елементи цієї апаратної частини і те, які особливості їх використання виявилися важливими саме для багаторозрядної цифрової індикації.

2.2.1. Вибір мікроконтролерної платформи

У цій роботі основою служить плата Arduino Nano. Це невелика плата на мікроконтролері ATmega328P: за можливостями вона близька до Arduino Uno, але займає менше місця, тож її простіше вписати в корпус чи поставити на невеликий стенд.

Для задач індикації цього цілком вистачає. На Nano є достатньо цифрових входів/виходів, щоб спробувати кілька варіантів підключення індикаторів. Крім того, плата має апаратний SPI (лінії SCK, MOSI, MISO) — через нього зручно працювати з драйвером MAX7219. Є й шина I²C (лінії SDA, SCL), яку я використовую для підключення контролера HT16K33. А для каскаду 74HC595 окремий апаратний модуль взагалі не потрібен: простий послідовний інтерфейс можна реалізувати програмно на будь-якій парі виводів.

З погляду апаратних ресурсів Arduino Nano має(Рис.2.1.):

- 14 цифрових ліній (D0–D13), частина з яких може використовуватися як виходи ШІМ;
- 8 аналогових входів (A0–A7), які при потребі також можна використовувати як цифрові;
- вбудований стабілізатор живлення з можливістю подачі як 5 В, так і вищої напруги на вхід VIN;
- USB-інтерфейс (через mini-USB / micro-USB, залежно від ревізії) для прошивки та живлення.

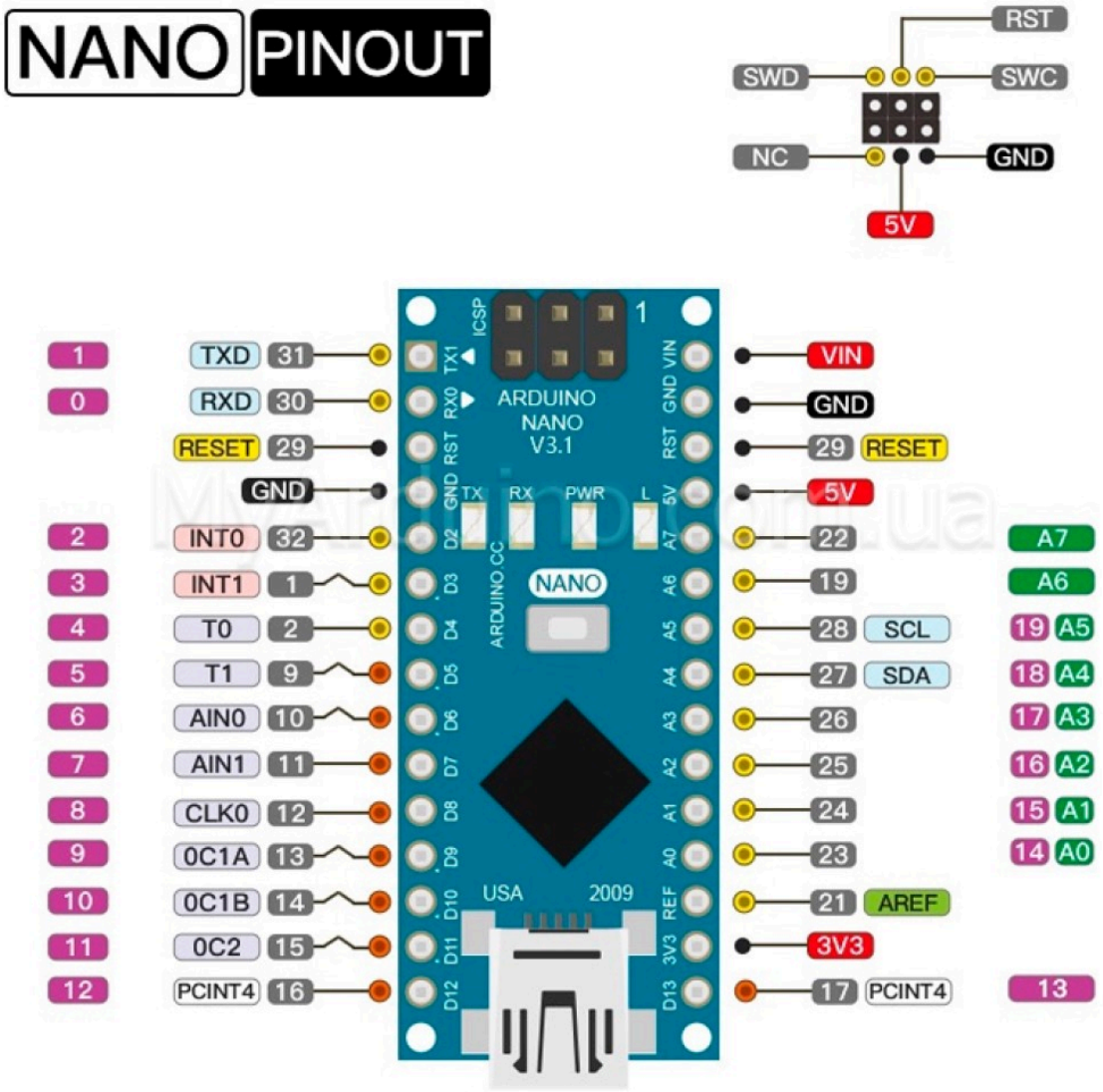

Рис.2.1. Розпіновка Arduino Nano

Обмежена кількість виводів Arduino Nano створює реалістичні умови задачі: якщо підключати багаторозрядну індикацію «напряму», значна частина портів буде зайнята сегментами й розрядами. Саме це й мотивує використання спеціалізованих драйверів (MAX7219, HT16K33) та регістрів зсуву (74HC595), які дозволяють організувати багаторозрядну індикацію за допомогою невеликої кількості ліній мікроконтролера.

2.2.2. Семисегментні індикатори та їх типи

Основним елементом відображення у проєкті є семисегментні світлодіодні індикатори. Один розряд семисегментного індикатора — це, по суті, набір із семи окремих світлодіодних “штрихів” (A–G) і, за потреби, ще десяткової крапки (DP). Змінюючи, які саме сегменти засвічені, можна показати цифри від 0 до 9 і кілька найпростіших символів.

Основні типи семисегментних індикаторів(Рис.2.2.):

- з загальним катодом (common cathode) — усі катоди сегментів об’єднані;
- з загальним анодом (common anode) — об’єднані аноди.

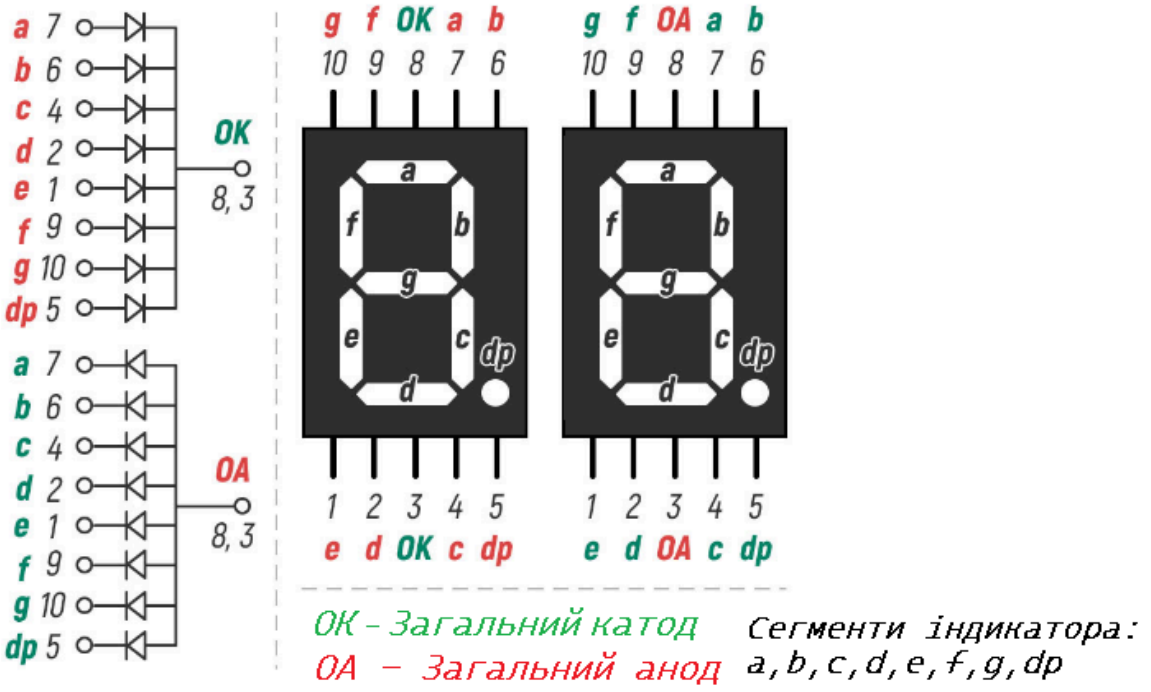

Рис.2.2. Розпіновка семисегментного індикатора

У дипломному проєкті використовуються індикатори з загальним катодом, оскільки вони:

- добре сумісні з поширеними модулями на MAX7219 і HT16K33, розрахованими саме на такий тип підключення;
- зручні для використання з транзисторними ключами в 74HC595.

Зазвичай окремі семисегментні індикатори не ставлять по одному, а збирають їх у спільний блок на кілька розрядів — наприклад, 4, 8 чи більше. У такому блоці вже можна нормально показувати цілі числа, а за потреби й значення з десятковою крапкою. Коли планується такий модуль, доводиться одразу думати про кілька речей:

- однаково підключити сегменти в усіх розрядах;
- правильно підібрати резистори обмеження струму;
- забезпечити акуратну розводку загальних шин живлення та «землі», щоб уникнути паразитних падінь напруги й мерехтіння.

2.2.3. Апаратна реалізація варіанта з MAX7219

У першому варіанті індикація зроблена на MAX7219. По суті, це окремий чип, до якого підключають кілька додаткових деталей та індикатори, де MAX7219 і семисегментні індикатори стоять на одній платі й одразу з’єднані між собою.

Основні елементи цього варіанта:

- Arduino Nano;
- мікросхема або модуль MAX7219;
- багаторозрядний 7-сегментний індикатор (до 8 розрядів від одного драйвера);
- резистор Rset для встановлення струму сегментів;
- конденсатори для фільтрації живлення.

Для обміну даними між Arduino Nano та MAX7219 використовуються три лінії:

- DIN (Data In) — послідовні дані;
- CLK (Clock) — тактовий сигнал;
- LOAD/CS (Chip Select / Load) — сигнал вибору мікросхеми та фіксації даних.

Ці сигнали підключаються до відповідних цифрових пінів Arduino Nano (наприклад, D10–D13, залежно від обраної конфігурації). Рівні напруги сумісні (5 В), тому не потрібні додаткові перетворювачі рівнів.

Виходи MAX7219 використовуються так:

- DIG0–DIG7 — керування окремими розрядами (цифрами) індикатора;
- SEG A–G, DP — керування сегментами всіх розрядів.

Струм через сегменти задається резистором Rset, підключеним до відповідного виводу MAX7219. Це спрощує розрахунок режимів роботи й захищає LED-сегменти від перевантаження.

Такий варіант схеми є компактним і технологічно простим: Arduino Nano передає лише дані й команди, а всі низькорівневі задачі (мультиплексування, обмеження струму, регулювання яскравості) вирішуються усередині MAX7219.

2.2.4. Апаратна реалізація варіанта з 74HC595

Другий варіант базується на регістрах зсуву 74HC595. Цей тип мікросхем, дозволяють за допомогою декількох ліній мікроконтролера, керувати великою кількістю виходів.

У в цей варіанта входять:

- Arduino Nano;
- каскад із двох або більше регістрів 74HC595;
- семисегментні індикатори з загальним катодом;
- транзисторні ключі для комутації розрядів (npn-транзистори з базовими резисторами);
- резистори обмеження струму для сегментів;
- конденсатори по живленню.

Зв’язок Arduino Nano з каскадом 74HC595 забезпечується трьома сигналами:

- SER / DS — вхід послідовних даних;
- SRCLK / SH_CP — тактовий сигнал зсуву;

- RCLK / ST_CP / LATCH — сигнал фіксації даних на виходах.

Логіка підключення така:

- частина виходів 74HC595 використовується для керування сегментами (спільні лінії для всіх розрядів: SEG A–G, DP);
- інша частина виходів керує базами транзисторів, які по черзі підключають до спільної шини відповідний розряд індикатора;
- транзистори можуть вмикати «землю» або «плюс» для розряду — залежно від вибраної схеми включення, але в цьому проєкті розглядається варіант із загальним катодом.

У цьому варіанті важливо правильно підібрати:

- номінали резисторів у колі сегментів, щоб обмежити струм кожного світлодіода;
- базові резистори транзисторів, щоб забезпечити надійне відкривання транзисторів без перевантаження виходів 74HC595 і Arduino Nano.

Основною особливістю такої апаратної конфігурації є те, що мультиплексування розрядів реалізується програмно: Arduino Nano по черзі активує транзистор відповідного розряду, записує в 74HC595 шаблон сегментів, витримує коротку затримку, вимикає розряд і переходить до наступного. Від правильності цього процесу залежить відсутність мерехтіння та рівномірність яскравості.

2.2.5. Апаратна реалізація варіанта з HT16K33

Третій варіант модулю індикації використовує HT16K33 — спеціалізований I²C-драйвер світлодіодних індикаторів.

До його апаратної частини входять:

- Arduino Nano;
- мікросхема або готовий модуль HT16K33;
- один або кілька багаторозрядних семисегментних індикаторів;

- резистори обмеження струму (якщо вони не інтегровані в модуль);
- розв’язувальні конденсатори.

Зв’язок між Arduino Nano і HT16K33 здійснюється по шині I²C, для якої використовуються:

- SDA — лінія даних;
- SCL — лінія такту.

На Arduino Nano шина I²C уже виведена на окремі ніжки: A4 відповідає за SDA, A5 — за SCL. Тому HT16K33 підключається досить просто, за стандартною схемою, без додаткових “хитрощів”. На цих же двох лініях можуть висіти й інші модулі, а сам HT16K33 відрізняється від них своєю I²C-адресою. Виходи HT16K33 організовані у вигляді матриці ліній. Їх можна налаштувати так, щоб вони керували або світлодіодною матрицею, або набором семисегментних індикаторів, залежно від того, що потрібно в конкретній схемі. Внутрішня логіка драйвера:

- забезпечує апаратне мультиплексування розрядів;
- формує ШІМ для регулювання яскравості;
- контролює режим роботи виходів.

Таким чином, Arduino Nano у цьому варіанті фактично тільки «оновлює вміст буфера», який HT16K33 потім самостійно відображає на індикаторах.

2.2.6. Елементи живлення, узгодження та захисту

Надійність всієї системи дуже залежить від того, як зроблене живлення і захист. Якщо тут зекономити або “забути дрібниці”, проблеми потім вилізуть саме на індикації.

1. Живлення.

○ Усі основні вузли — Arduino Nano, MAX7219, 74HC595, HT16K33 і самі індикатори — працюють від 5 В. Nano можна живити або від USB, або від зовнішнього блока через VIN. Щоб живлення було більш стабільним, біля кожної мікросхеми варто поставити по керамічному конденсатору приблизно 100 нФ, а на

вході живлення — ще один, електролітичний, більшої ємності (наприклад, кілька десятків–сотень мікрофарад).

2. Струм через сегменти.

○ У схемі з MAX7219 струм сегментів задається резистором Rset і далі вже контролюється всередині самого драйвера.

○ Якщо використовується 74HC595 або HT16K33, без додаткових резисторів уже не обійтися: у коло кожного сегмента (або, як мінімум, кожної групи) ставляться обмежувальні резистори, щоб не перевантажувати виходи й не спалити індикатор.

○ Номінали резисторів підбираються з урахуванням бажаної яскравості та допустимих струмів для конкретних індикаторів.

3. Загальна «земля» та узгодження рівнів.

○ У всіх варіантах мікроконтролер і модулі індикації мають спільну «землю» (GND).

○ Важливо забезпечити надійні з’єднання GND, особливо при використанні макетних плат і довших проводів, щоб запобігти нестабільності роботи та мерехтінню індикаторів.

4. Захист від короткого замикання та перевищення струму.

○ У навчальних стендах зазвичай використовується захист на рівні блока живлення (обмеження максимального струму);

○ у реальних промислових рішеннях доцільно додавати запобіжники або інші елементи захисту на вході живлення.

2.2.7. Можливості розширення та модернізації апаратної частини

Проєктована апаратна частина не є жорстко фіксованою. Вона спроєктована так, щоб:

- у варіанті з MAX7219 можна було каскадувати кілька драйверів, збільшуючи кількість розрядів без значного ускладнення схеми;

- У схемі з 74HC595 можна просто підчепити ще один регістр у ланцюжок і отримати додаткові виходи — під нові сегменти, індикатори чи окремі світлодіоди.
- HT16K33 масштабується по-іншому: на тій самій I²C-шині можна повісити кілька таких драйверів з різними адресами й таким чином зібрати більшу, комбіновану панель індикації

2.3. Структурна схема системи індикації

Коли проєктують електронний пристрій, зручно спочатку дивитися на нього не з боку окремих мікросхем і доріжок, а “згори” — як на набір блоків. Один блок відповідає за живлення, інший за вимірювання, ще один за обробку даних, окремо стоїть індикація тощо. Такий підхід по суті і є структурним: систему ділять на зрозумілі шматки й уже потім розбираються, як ці частини пов’язані між собою.

Багаторозрядна індикація у вбудованому пристрої виконує начебто просту, але дуже помітну роль — показує числа користувачу: значення лічильників, дані з датчиків (температура, напруга, час) або службові коди. Якщо цей блок працює некоректно, для користувача все інше залишається “закритим”: прилад щось робить усередині, але що саме, не видно. У цій роботі індикаційна частина будується навколо мікроконтролера Arduino Nano та багаторозрядних семисегментних індикаторів. Для реалізації керування індикацією розглядаються три різні варіанти апаратного модуля:

- варіант на драйвері MAX7219;
- варіант на каскаді регістрів зсуву 74HC595 з транзисторними ключами;
- варіант на I²C-драйвері HT16K33.

Ці варіанти роблять одне й те саме — керують багаторозрядною індикацією, просто кожен по-своєму. На структурній схемі це добре видно: вона показує

загальну картину і дозволяє наочно порівняти, як саме організований кожен підхід.

2.3.1. Загальна структура системи

Найпростішу систему індикації в цій роботі можна уявити як кілька основних “цеглинок”, які разом працюють на результат (Рис.2.3.).

- Блок живлення. Його завдання просте: подати стабільні 5 В на Arduino Nano, мікросхеми керування індикаторами та самі світлодіодні індикатори, щоб усе це не просідало по напрузі й працювало без збоїв.
- Мікроконтролер Arduino Nano. «Мозок» системи. Він виконує програму, обробляє дані, формує числа, які необхідно відобразити, і надсилає відповідні команди до модуля індикації.
- Модуль багаторозрядної індикації. Спеціалізований блок, який перетворює інформацію від Arduino на сигнали для сегментів і розрядів. Саме на ньому ми будемо концентрувати увагу. У роботі розглянуто три його реалізації.
- Семисегментні індикатори. Набір із декількох цифр, кожна з яких складається із семи світлодіодних сегментів та, за потреби, десяткової точки. Кожна цифра відповідає одному розряду (одній позиції) числа.
- Захисні та узгоджувальні елементи. Резистори обмеження струму, транзисторні ключі для комутації розрядів, конденсатори для фільтрації живлення. Вони не «розумні», але без них пристрій або не працюватиме, або працюватиме ненадійно.

Рис.2.3. Блок-схема загальної структури системи

Тобто, незалежно від того, який саме чип ми ставимо між Arduino та індикаторами, загальний принцип не змінюється: Arduino не керує напряму кожним світлодіодом, а спілкується з окремим «драйвером», який бере на себе рутинну роботу з увімкненням сегментів.

2.3.2. Варіант модуля індикації на драйвері MAX7219

MAX7219 — це спеціальний драйвер, який придумали саме для керування семисегментними індикаторами та світлодіодними матрицями. Ідея проста: замість того, щоб Arduino постійно “ганяв” окремі сегменти, уся чорнова робота віддається цій мікросхемі (рис. 2.4).

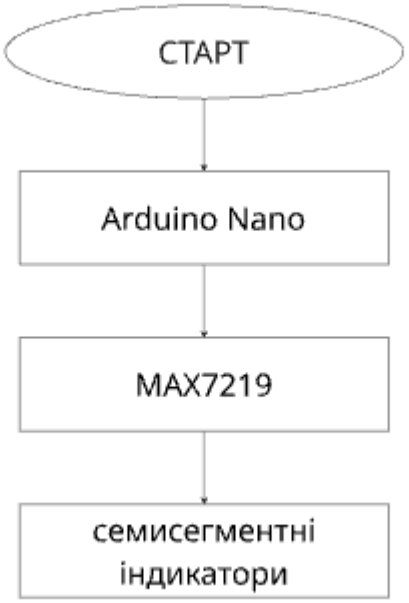

Рис.2.4. Блок-схема індикації на драйвері MAX7219

Між Arduino Nano і MAX7219 йде послідовний обмін даними, по кількох проводах: лінія даних, тактовий сигнал і сигнал фіксації. Arduino передає туди байти, а всередині MAX7219 є своя пам’ять — по суті, набір регістрів, де зберігається, що має світитися на кожному розряді. Далі драйвер сам робить мультиплексування: дуже швидко перебирає розряди по колу, тому для ока всі цифри виглядають засвіченими одночасно.

Яскравість можна змінювати через окрему команду (внутрішній регістр інтенсивності), а одна мікросхема без додаткових хитрощів тягне до 8 розрядів. Якщо пояснювати зовсім просто:

1. Arduino один раз “каже”, що показувати,
2. MAX7219 вже сам вирішує, які сегменти та з якою яскравістю вмикати, щоб на індикаторі вийшло потрібне число.

2.3.3. Варіант модуля індикації на каскаді регістрів зсуву 74HC595

Другий варіант модуля індикації побудовано на регістрах зсуву 74HC595. Це вже не «розумний драйвер», а універсальні мікросхеми, які дозволяють отримати багато виходів із невеликої кількості виводів мікроконтролера (Рис.2.5.).

Рис.2.5. Блок-схема індикації на каскаді регістрів зсуву 74HC595

Тут логіка така:

- кожен 74HC595 має 8 виходів. Якщо з’єднати кілька таких мікросхем «у ланцюжок» (каскад), можна керувати, наприклад, 16, 24 або 32 лініями, використовуючи всього 3–4 виводи Arduino;
- дані подаються послідовно — по одному біту за такт, а потім «застигають» на виходах завдяки сигналу LATCH;
- частина виходів керує сегментами (які саме частини цифри світяться), а частина — транзисторними ключами, які по черзі вмикають різні розряди індикатора.

На відміну від MAX7219, тут немає вбудованого мультиплексування. Воно цілком реалізується програмно:

- Arduino сам вирішує, який розряд зараз активувати;
- сам виставляє потрібні біти в 74HC595, щоб увімкнути потрібні сегменти;
- сам обирає затримки між перемиканнями розрядів.

Такий підхід дає багато свободи: можна наростити велику індикаторну панель, додавати розряди, сегменти, світлодіоди. Але за це доводиться платити — більше роботи лягає на програму, і потрібно добре розуміти, як саме працює мультиплексування, щоб індикатор не мерехтів і не зависав. Якщо пояснити простіше, без формул: мікросхему 74HC595 можна уявити як розгалужувач. Мікроконтролер говорить одним “каналом” даних, а 74HC595 уже розкладає ці біти по багатьох виходах. Але вирішує, яку саме “лампочку” коли ввімкнути, все одно мікроконтролер — він має чітко й вчасно подати потрібну послідовність бітів.

2.3.4. Варіант модуля індикації на I²C-драйвері HT16K33

Третій варіант — використання HT16K33, це спеціалізований драйвер світлодіодної індикації, який працює по шині I²C (Рис.2.6.).

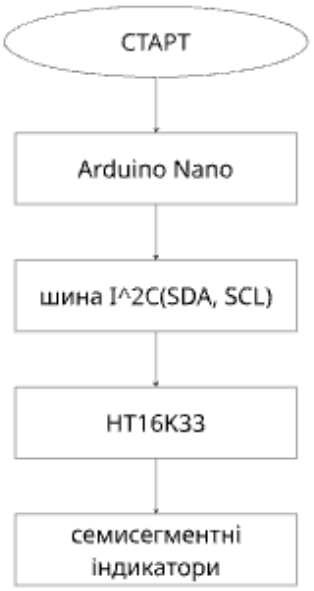

Рис.2.6. Блок-схема індикації на I²C-драйвері HT16K33

Основні риси HT16K33:

- використовує всього дві лінії зв’язку з Arduino (SDA — дані, SCL — такт), по яких можна підключити ще багато інших пристроїв;
- має велику кількість виходів (типово до 8×16 світлодіодів), яких достатньо для кількох багаторозрядних індикаторів;
- всередині мікросхеми вже є власне мультиплексування: вона сама по черзі перемикає розряди з потрібною швидкістю, без участі програми;
- у чипі передбачене керування яскравістю — за рахунок вбудованого ШІМ, тож рівень підсвічування можна змінювати без додаткових схем;
- дозволяє підключати кілька таких драйверів до однієї і тієї ж пари ліній SDA/SCL, змінюючи їх I²C-адреси. Це спрощує масштабування.

Якщо порівняти з MAX7219, HT16K33 близький за ідеологією: це теж «розумний драйвер», але він працює через інший інтерфейс (I²C) і розрахований на іншу топологію підключення. Для Arduino це виглядає як запис байтів у певні «комірки пам’яті» HT16K33, після чого драйвер сам організовує роботу індикаторів. HT16K33 — це ще один «посередник» між Arduino і світлодіодами, але він підключається до двох універсальних ліній (I²C), які Arduino і так має «завжди під рукою». Це дозволяє не витрачати додаткові виводи й легко розширювати систему.

2.3.5. Порівняння структурних рішень

Якщо подивитися зверху, у всіх трьох випадках маємо те саме: багаторозрядний індикатор, яким керує Arduino.

Але всередині ці модулі побудовані по-різному, і це вже впливає на інше, скільки мороки з проєктуванням, наскільки легко все розширювати й наскільки складний код потрібен для керування індикацією. Водночас архітектура кожного модуля має свої особливості:

- MAX7219

- мінімальна кількість зовнішніх компонентів;
- простий інтерфейс, близький до SPI;
- внутрішнє мультиплексування та регулювання яскравості;
- обмеження на кількість розрядів (до 8 без каскадування);
- добре підходить для компактних, простих пристроїв.

- 74HC595
  - універсальне розширення кількості виходів;
  - повний контроль над мультиплексуванням із боку програми;
  - більше навантаження на мікроконтролер, необхідність точно підбирати таймінги;
  - зручно масштабувати до дуже великого числа розрядів;
  - цікавий навчальний варіант, який дає глибоке розуміння роботи індикації «на низькому рівні».
- HT16K33
  - підключення по шині I²C з використанням лише двох ліній;
  - апаратне мультиплексування та ШІМ-керування яскравістю;
  - можливість підключення кількох драйверів на одну шину;
  - гнучке розширення без значного ускладнення схеми;
  - добре підходить, коли Arduino вже зайнятий іншими модулями, а вільних виводів небагато.

2.4. Алгоритм роботи системи індикації

Щоб багаторозрядна цифрова індикація здавалася користувачу «просто цифрами, що світяться», усередині пристрою постійно виконується досить складний алгоритм.

У роботі вся логіка крутиться навколо мікроконтролера. Він постійно оновлює дані, розкладає числа на сегменти й надсилає ці біти в мікросхеми

керування. Далі вже самі драйвери по черзі вмикають розряди та сегменти, задаючи потрібну частоту оновлення й рівень яскравості.

У цьому підрозділі спочатку подається загальна схема роботи такої системи, а потім окремо розбираються три варіанти модуля індикації:

- на драйвері MAX7219;
- на регістрах зсуву 74HC595;
- на I²C-драйвері HT16K33.

2.4.1. Загальна послідовність роботи

Незалежно від обраного апаратного рішення, логіка роботи системи індикації складається з кількох типових етапів(Рис.2.7.):

1. Після подачі живлення спочатку треба «розбудити» всю систему. Мікроконтролер виставляє режими роботи своїх пінів, вмикає потрібні інтерфейси (SPI, I2C або просто кілька цифрових ліній, якими далі керує вручну) і надсилає стартові команди в MAX7219, HT16K33 чи каскад 74HC595, щоб ці мікросхеми перейшли в робочий стан.

2. Формування значень, які потрібно показати.

У програмі обчислюються числа, що підлягають відображенню: це можуть бути покази датчиків, лічильники, коди стану, час тощо.

3. Розбиття числа на окремі розряди.

Багаторозрядне число розкладається на окремі цифри (зліва направо або навпаки), які будуть показані на індикаторі в кожному розряді.

4. Перетворення цифр на шаблони сегментів.

Кожна цифра замінюється на набір увімкнених сегментів семисегментного індикатора шляхом використання спеціальної таблиці кодування (lookup table).

5. Передача шаблонів у мікросхему керування індикацією.

Підготовлені байти передаються в MAX7219, HT16K33 або 74HC595 через відповідний інтерфейс.

6. Мультиплексування та оновлення індикації.

Внутрішня логіка драйвера (MAX7219, HT16K33) або програмний код (у випадку 74HC595) забезпечують швидке циклічне перемикання розрядів так, щоб людина сприймала їх як одночасно засвічені.

7. Регулювання яскравості та енергоспоживання.

За потреби налаштовуються параметри яскравості, частоти оновлення, а також реалізуються режими вимикання/затемнення індикації для економії енергії.

Рис.2.7. Блок-схема загальної послідовності роботи

2.4.2. Підготовка числових даних до відображення

Перш ніж щось «засвітити» на індикаторі, мікроконтролер має отримати та підготувати відповідні числові дані. Джерела цих даних можуть бути різними:

- простий внутрішній лічильник — секундомір, таймер зворотного відліку, просто лічба імпульсів;
- покази датчиків: температура, напруга, освітленість, рівень сигналу й подібні величини;
- результати обчислень у самій програмі — суми, різниці, відсотки, частота, середні значення;
- тестові чи демонстраційні набори, які зручно вмикати під час налагодження: прокручування 0000, 1111, 2222…, виведення 8888 для перевірки всіх сегментів.

Після цього виконується розбиття числа на розряди. Коли треба щось показати на індикаторі, спочатку число розбивають на окремі цифри. Наприклад, 1234 перетворюють у набір 1, 2, 3, 4. У коді це зазвичай роблять або через ділення на 10 та взяття остачі, або простіше — переводять число в рядок і беруть символи по одному.

Далі постає питання, як це влізе у кількість розрядів. Якщо індикатор має 4 розряди, а значення, наприклад, 25, треба вирішити, що саме показувати: 0025, 25 з вимкненими старшими розрядами чи якийсь інший варіант вирівнювання. Якщо ж число довше за доступну кількість розрядів (наприклад, 123456 на 4-розрядному індикаторі), доводиться або відкидати частину цифр, або робити прокрутку чи поетапне відображення.

2.4.3. Кодування цифр у шаблони сегментів

Семисегментний індикатор не вміє «знати», що таке цифра 3 чи 7, – він лише має сім (або вісім, з десятковою точкою) сегментів-світлодіодів, які можна вмикати або вимикати. Тому наступний крок алгоритму — перетворення цифри на шаблон сегментів.

Для цього використовується таблиця кодування (lookup table), в якій:

- для кожної цифри від 0 до 9 зберігається один байт (8 біт);
- кожен біт у цьому байті відповідає одному сегменту: A, B, C, D, E, F, G і, за потреби, десятковій крапці;
- наприклад, для 0 в масці будуть увімкнені всі сегменти, крім середнього (G),
- для 1 — тільки два “праві” вертикальні сегменти, решта вимкнені.
- Далі програма просто бере потрібну маску з таблиці й відправляє її в драйвер або прямо на лінії сегментів.

У коді це зазвичай оформлено як масив констант, наприклад(Рис.2.8.):

```
const uint8_t digitMap[10] = {
  0b00111111, // 0
  0b00000110, // 1
  0b01011011, // 2
  0b01001111, // 3
  0b01100110, // 4
  0b01101101, // 5
  0b01111101, // 6
  0b00000111, // 7
  0b01111111, // 8
  0b01101111  // 9
};
```

Рис.2.8. масив констант

Алгоритм на цьому етапі:

1. Беремо підготовлений масив цифр [d0, d1, d2, d3].
2. Для кожного елемента di підставляємо відповідний байт з таблиці кодування.
3. Отримуємо масив шаблонів сегментів, готовий для передавання в драйвер.

За потреби тут же можуть додаватися службові символи:

- увімкнення десяткової точки (наприклад, для показу 12.34);
- відображення знака мінус для від’ємних значень;
- спеціальні символи типу «– – – –», «Err» тощо (для них також додаються окремі шаблони кодування).

2.4.4. Алгоритм обміну з MAX7219

У варіанті з MAX7219 увесь подальший процес значно спрощується, тому що більшість операцій бере на себе сам драйвер.

Алгоритм взаємодії з MAX7219 можна описати так:

1. Ініціалізація MAX7219.

○ надсилаються команди, які вимикають тестовий режим, задають робочу кількість розрядів, встановлюють початкову яскравість;

○ очищується внутрішня пам’ять розрядів (усі індикатори гаснуть).

2. Оновлення індикації.

Для кожного розряду (позиції на індикаторі):

○ вибирається адреса розряду (наприклад, від 1 до 8);

○ береться відповідний байт із масиву шаблонів сегментів;

○ по послідовному інтерфейсу посилається два байти: «адреса розряду» + «дані для цього розряду»;

○ після передавання формується імпульс на лінії LOAD/CS, який фіксує отримані дані.

3. Внутрішнє мультиплексування.

MAX7219 самостійно, без участі Arduino, з потрібною частотою:

○ перемикає активний розряд;

○ вмикає для цього розряду сегменти відповідно до збереженого стану;

○ забезпечує достатню частоту оновлення, щоб людина не бачила мерехтіння.

4. Зміна яскравості.

При необхідності Arduino може в будь-який момент надіслати команду, яка змінює регістр яскравості, не змінюючи при цьому самі дані індикації.

З точки зору алгоритму програми на Arduino, робота з MAX7219 схожа на оновлення «маленького екрана» по адресу: ми просто записуємо, які символи мають бути в кожному розряді, а внутрішня логіка драйвера далі «крутить» їх сама.

2.4.5. Алгоритм обміну з 74HC595 та програмне мультиплексування

Алгоритм роботи з 74HC595 відрізняється тим, що:

- по-перше, регістри зсуву самі по собі не вміють мультиплексувати розряди;
- по-друге, вони лише зберігають і видають стани на виходи, а всю «розумну» роботу виконує мікроконтролер.

Алгоритм можна розбити на два рівні:

1. Запис даних у каскад 74HC595.
   - Arduino формує послідовність бітів, яка відповідає поточному стану всіх сегментів і розрядів;
   - по лінії DATA відправляється кожен біт;
   - по лінії CLOCK формується тактовий імпульс, що «зсуває» біт у регістр;
   - після надсилання всіх бітів подається імпульс на LATCH, який переписує ці біти на виходи регістрів.
2. Мультиплексування розрядів у основному циклі.

Тут алгоритм виглядає так:

- у програмі визначено поточний «активний розряд» (наприклад, від 0 до 7);
- для активного розряду обирається відповідний шаблон сегментів;
- формуються дані для сегментів (які сегменти мають світитися) та дані для транзисторів (який саме розряд зараз увімкнути);
- ці дані послідовно записуються через каскад 74HC595;
- для цього розряду витримується коротка пауза (затримка в кілька сотень мікросекунд);
- активний розряд вимикається, і змінна активного розряду переходить до наступного значення.

Цикл швидко повторюється для всіх розрядів. Якщо робити це досить швидко (із частотою десятки–сотні герц для кожного розряду), людське око сприймає індикацію як статичну, без мерехтіння.

Тут важливо:

- правильно підібрати затримки: якщо вони будуть надто великими, індикатор помітно мерехтітиме;
- дотримуватися пропорцій між часом увімкнення й вимкнення розрядів, щоб не було «пересвічування» одного розряду й «тьмяності» іншого;
- забезпечити достатньо швидке виконання інших частин програми, щоб вони не «зривали» мультиплексування.

Таким чином, у варіанті з 74HC595 основне навантаження за часом і логікою лягає саме на програму Arduino.

2.4.6. Алгоритм обміну з HT16K33

У випадку з HT16K33 алгоритм поєднує в собі особливості MAX7219 (апаратне мультиплексування та яскравість) і гнучкість шини I²C.

Загальна логіка така:

1. Ініціалізація шини I²C.

Arduino вмикає I²C-інтерфейс (у термінології Arduino — Wire), задає частоту роботи шини та готується до спілкування з HT16K33 за певною I²C-адресою.

2. Ініціалізація HT16K33.

Надсилаються службові команди, які:

- вмикають генератор мультиплексування;
- вмикають виходи драйвера;
- встановлюють базовий рівень яскравості (через регістр ШІМ);
- очищують усі виходи (гаситься попередня індикація).

3. Формування буфера відображення.

У пам’яті Arduino створюється буфер, який за структурою відповідає внутрішній

пам’яті HT16K33. У цей буфер записуються біти, що визначають стан кожного сегмента й кожного розряду.

4. Передача буфера в HT16K33 через I²C.

- Arduino відкриває сесію передачі (start condition);
- надсилає адресу HT16K33 на шині I²C;
- вказує стартовий адрес пам’яті драйвера, куди потрібно записати дані;
- послідовно передає байти буфера;
- завершує передачу (stop condition).

5. Внутрішнє мультиплексування та регулювання яскравості.

Після того, як дані записані у внутрішню пам’ять, HT16K33 самостійно:

- перемикає розряди з необхідною частотою;
- керує яскравістю через вбудований ШІМ;
- синхронізує роботу всіх виходів.

Для Arduino це виглядає так: «записати новий вміст екрана» у вигляді послідовності байтів. Далі драйвер працює сам, звільняючи мікроконтролер від низькорівневої роботи.

2.4.7. Регулювання яскравості та енергоспоживання

Окремо доводиться думати і про яскравість індикаторів, і про те, скільки вони споживають. Це не дрібниця: у вбудованих системах пристрій часто живиться від батарейки, акумулятора чи навіть порту USB, тож зайвий світ індикаторів прямо б’є по часу роботи.

Можна виділити такі механізми:

- MAX7219
  - має окремий регістр яскравості;
  - Arduino може встановити один із кількох рівнів, змінюючи піковий струм сегментів;
  - зменшення яскравості автоматично знижує споживання енергії.

- 74HC595
  - не має «вбудованого» регулювання яскравості;
  - яскравість можна змінювати програмно, регулюючи коефіцієнт заповнення (долю часу, коли розряд увімкнений);
  - для цього інколи вводять додаткові цикли — наприклад, на кожен розряд відводиться певний період, протягом частини якого він світиться, а частини — вимикається.
  - так само можна зменшити загальну частоту оновлення, якщо дозволяє задача.

- HT16K33
  - має вбудований ШІМ-контролер яскравості;
  - Arduino змінює яскравість шляхом запису в спеціальний регістр, не змінюючи при цьому логіку індикації;
  - драйвер сам коригує Імпульсні співвідношення, знижуючи споживання.

Крім того, у всіх трьох випадках можна додатково:

- вимикати індикацію, якщо дані не змінюються або пристрій перебуває в режимі очікування;
- частково гасити розряди, які в даний момент не несуть корисної інформації (наприклад, старші нулі).

2.4.8. Обробка спеціальних режимів роботи

Реальний пристрій рідко обмежується показом лише «спокійних» чисел. У практиці часто необхідно реалізувати спеціальні режими індикації, які також є частиною алгоритму:

- Режим тесту індикаторів.

На початку роботи (або за окремою командою) кожен сегмент усіх розрядів

вмикається одночасно (наприклад, індикація 8888 або всі сегменти + точки). Це дозволяє користувачу швидко переконатися, що жоден сегмент не «згорів» і не відключився.

- Режим сигналізації помилки.

У разі некоректних даних або внутрішньої помилки системи індикація може переходити в спеціальний стан: миготіння, відображення «Err», «----», чергування кодів помилки тощо. Алгоритм у цьому разі включає додатковий рівень логіки, який визначає, який саме шаблон показувати і з якою частотою миготіти.

- Режим економії енергії.

При тривалому простої або при роботі від батареї програма може знижувати частоту оновлення, зменшувати яскравість або повністю вимикати індикацію, поки користувач не натисне кнопку чи не відбудеться подія.

- Режим демонстрації / тестових шаблонів.

Для наочної демонстрації роботи системи індикація може програвати наперед задані послідовності, наприклад «біжучі» цифри або ефекти «заповнення» сегментів. Це корисно також для перевірки стабільності роботи мультиплексування.

Усі ці режими реалізуються на рівні алгоритму програми, але спираються на ті ж самі базові механізми: формування чисел, кодування в сегменти, передавання даних у драйвери та оновлення індикації.

2.5. Електрична принципова схема системи

Електрична принципова схема, вже не загальна картинка, а фактично «карта проводів». На ній видно, куди саме підключена Arduino Nano, де стоять MAX7219, 74HC595, HT16K33, як до них під’єднані семисегментні індикатори, звідки подається живлення, де розміщені резистори та елементи захисту. Тобто якщо структурна схема просто показує, які вузли між собою пов’язані, то принципова говорить уже про інше: який вивід до якого контакту йде і що стоїть між ними по дорозі.

У дипломній роботі реалізовано й досліджено три варіанти електричної схеми модуля індикації:

- варіант на драйвері MAX7219;
- варіант на каскаді 74HC595 із транзисторними ключами;
- варіант на I²C-драйвері HT16K33.

Усі три варіанти побудовані на базі Arduino Nano, що явно відображено на принципових схемах, змодельованих у середовищі Proteus.

2.5.1. Загальні підходи до побудови принципових схем

Перед тим як розглядати кожен варіант окремо, важливо виділити кілька спільних моментів, які присутні на всіх принципових схемах:

1. Спільна шина живлення та «землі».

Усі мікросхеми (Arduino Nano, MAX7219, 74HC595, HT16K33) живляться від 5 В, а їхні контакти GND обов'язково з'єднані між собою. На схемі чітко показано, що:

- вивід 5V Arduino Nano є джерелом живлення для логічних мікросхем та індикаторів (у симуляції Proteus це може бути окреме джерело 5 В, до якого під'єднаний Nano);
- усі позначення GND об'єднані в одну загальну точку.

2. Розв'язувальні конденсатори.

Біля виводів живлення кожної мікросхеми ставлять невеликі конденсатори, зазвичай близько 100 нФ, просто між VCC і GND. Вони підхоплюють короткі стрибки струму і згладжують живлення, що особливо помітно, коли індикатор швидко перемикає розряди при мультиплексуванні.

3. Резистори обмеження струму для LED-сегментів.

Світлодіодні сегменти семисегментних індикаторів вимагають обмеження струму. У схемах це реалізовано:

- або одним резистором Rset (для MAX7219), який задає струм для всіх сегментів;

○ або окремими резисторами в колі сегментів (варіанти з 74HC595 та HT16K33).

4. Однозначне маркування виводів Arduino Nano.

На схемах чітко вказано, до яких саме пінів Nano підключені сигнали керування: D10, D11, D12, D13 для інтерфейсу типу SPI, A4/A5 для I²C, будь-які вільні пінів для керування 74HC595 тощо. Це дозволяє легко відтворити схему на реальній макетній платі.

Далі розглянемо кожен варіант принципової схеми детальніше.

2.5.2. Принципова схема варіанта з Arduino Nano та MAX7219

На рисунку 2.4, показано електричне з'єднання між:

- платою Arduino Nano;
- мікросхемою MAX7219;
- багаторозрядним семисегментним індикатором (до 8 цифр).

1. Підключення Arduino Nano до MAX7219.

На схемі видно, що три цифрові виводи Arduino Nano використовуються для обміну даними з MAX7219:

- умовно, D10 – сигнал LOAD (CS);
- D11 – сигнал DIN (MOSI – лінія даних);
- D13 – сигнал CLK (SCK – тактовий сигнал).

Номери пінів можуть бути змінені програмно, але на схемі вони обрані так, щоб відповідати типовій SPI-розводці Nano. Важливо, що:

- усі ці сигнали односторонні: від Arduino Nano до MAX7219;
- рівні напруги узгоджені (5 В), тому додаткові логічні перетворювачі не потрібні.

2. Підключення MAX7219 до індикатора.

Далі на схемі відображено, як MAX7219 підключається до семисегментних індикаторів:

- виводи DIG0–DIG7 мікросхеми йдуть до загальних катодів (або анодів, залежно від конфігурації) кожного розряду індикатора;
- виводи SEG A–G, DP підключені до відповідних сегментів усіх розрядів.

Таким чином, кожен сегмент кожної цифри вмикається комбінацією: «активний розряд (DIGx) + потрібний сегмент (SEG y)».

3. Ланцюг обмеження струму.

На принциповій схемі присутній резистор Rset, підключений між спеціальним виводом Iset MAX7219 та шиною живлення. Його номінал задає максимальний струм через сегменти. Це дозволяє:

- уникнути перегріву індикаторів;
- забезпечити близьку яскравість усіх сегментів без підбору окремих резисторів.

4. Живлення та конденсатори.

У MAX7219 ніжки VCC і GND йдуть на 5 В і землю. Поруч між ними зазвичай ставлять один маленький конденсатор 100 нФ і ще електроліт 10–47 мкФ — вони трохи згладжують стрибки струму, коли драйвер швидко перемикає розряди.

У результаті схема з MAX7219 виходить досить проста: Arduino Nano з’єднаний із мікросхемою трьома сигналами (DATA, CLK, CS), а самі семисегментні індикатори вже “розкидані” від MAX7219 по лініях DIG і SEG.

2.5.3. Принципова схема варіанта з Arduino Nano та 74HC595

На рисунку 2.5, показано більш «розгалужене» рішення, у якому мікросхеми 74HC595 розширюють кількість виходів Arduino Nano.

1. Підключення Arduino Nano до каскаду 74HC595.

На схемі видно три основні сигнали:

- SER (DS) підключений до одного з цифрових виводів Arduino Nano, наприклад D8;

- SRCLK (SH_CP) – до іншого виводу, наприклад D9;
- RCLK (ST_CP / LATCH) – до третього, наприклад D10.

Цього достатньо, щоб послідовно завантажувати дані в кілька з’єднаних між собою 74HC595.

Лінія Q7’ першого 74HC595 з’єднана із входом SER наступного, утворюючи каскад, в якому байти даних зсуваються «ланцюжком» через усі мікросхеми.

2. Підключення виходів 74HC595 до сегментів.

Частина виходів каскаду 74HC595 використовується для керування сегментами:

- до виходів Q0–Q6 можуть бути підключені сегменти A–G;
- до одного з виходів — сегмент DP, якщо використовується десяткова точка.

Між виходами 74HC595 та сегментами індикаторів встановлюються резистори обмеження струму, номінал яких вибирається за довідковими даними для конкретних світлодіодних індикаторів і бажаної яскравості.

Ці сегментні лінії є спільними для всіх розрядів: тобто сегмент A для всіх цифр підключений до одного виходу 74HC595, сегмент B — до іншого, і так далі.

3. Комутація розрядів через транзисторні ключі.

Окремі виходи 74HC595 або Arduino Nano (залежно від обраної конфігурації) підключені до баз транзисторів, які керують загальними лініями розрядів (DIG1, DIG2, DIG3, …). На схемі видно:

- npn-транзистори (наприклад, типу 2N2222, BC547 тощо), кожний із яких відповідає за один розряд;
- резистори в базових ланцюгах, що обмежують струм бази та захищають як мікросхему 74HC595, так і сам транзистор.

Колектори транзисторів підключені до загальних катодів відповідних розрядів індикатора, емітери — до GND (при схемі з загальним катодом), що дозволяє «вмикати» той чи інший розряд, подаючи на базу транзистора логічну «1».

4. Живлення та додаткові елементи.

Виводи VCC та GND мікросхем 74HC595 підключені до 5 В і спільної «землі». Біля них розміщені розв’язувальні конденсатори. За потреби використовується сигнал MR (reset), який у простому варіанті може бути підтягнутий до VCC через резистор, щоб уникнути випадкового скидання регістрів.

У підсумку принципова схема з 74HC595 виходить найбільш насиченою: багато ліній сегментів, транзисторні ключі, резистори. Проте саме цей варіант найкраще демонструє всі внутрішні процеси мультиплексування й керування індикацією, що важливо в навчальному контексті.

2.5.4. Принципова схема варіанта з Arduino Nano та HT16K33

На рисунку 2.6, наведено варіант, у якому використовується I²C-драйвер HT16K33. Ця схема за кількістю з’єднань виглядає простіше, але логіка роботи драйвера всередині нього значно складніша.

1. Підключення Arduino Nano до HT16K33 по I²C.

На принциповій схемі чітко позначено:

- вивід A4 Arduino Nano підключено до лінії SDA HT16K33;
- вивід A5 Arduino Nano підключено до лінії SCL HT16K33.

Між цими лініями й шиною 5 В зазвичай встановлюються підтягуючі резистори (pull-up), які забезпечують коректний логічний рівень на шині I²C. У багатьох готових модулях HT16K33 ці резистори вже впаяні, що також можна врахувати в схемі.

2. Підключення виходів HT16K33 до індикаторів.

На схемі видно, що виходи драйвера HT16K33 утворюють матрицю ліній (наприклад, ROW0–ROW7, COL0–COL15). Вони підключаються до:

- загальних ліній розрядів індикатора;
- ліній сегментів.

Залежно від конфігурації, яку задає програма, ті або інші виходи використовуються як «рядки» та «стовпці» для відображення:

- семисегментних цифр;
- додаткових сегментів і точок;
- (у загальному випадку) навіть матричних індикаторів.

Як і у попередніх варіантах, у колах сегментів або груп сегментів встановлюються резистори обмеження струму. У готових модулях HT16K33 вони часто вже інтегровані, і це відображається на принциповій схемі відповідним символом або приміткою.

3. Живлення та конденсатори.

Вивід OE (Output Enable), якщо потрібно, можна використати як окремий вимикач виходів: подали потрібний рівень — індикатор погас, змінили рівень — знову світиться. У найпростішому варіанті цей вивід просто залишають у стані “увімкнено” і далі до нього не повертаються.

Основний плюс такої схеми в іншому: Arduino Nano спілкується з модулем індикації лише по двох лініях — SDA та SCL. Уся подальша робота з розрядами й сегментами відбувається вже всередині HT16K33, тож на платі менше проводів і менше шансів щось переплутати під час монтажу.

2.5.5. Порівняльний аналіз принципових схем

Порівнюючи три варіанти електричних принципових схем, можна зробити кілька практичних висновків.

1. Складність схеми.

- Варіант з MAX7219 — один із найпростіших з погляду розводки: небагато з’єднань, мінімум додаткових елементів, зрозумілі зв’язки DIG×SEG.
- У схемі з 74HC595 проводів і деталей найбільше: багато ліній на сегменти, транзистори, резистори. Вона добре показує, що відбувається

“всередині” індикації, зате вимагає акуратного монтажу і уважності до кожного виводу.

- Варіант із HT16K33 виглядає простіше: з боку Arduino є лише дві лінії I²C, а далі вже кілька виходів прямо на індикатор. Усю складну частину — мультиплексування, керування яскравістю, внутрішні регістри — бере на себе сам драйвер.

2. Вимоги до виводів Arduino Nano.

- MAX7219 потребує лише 3 лінії керування (DIN, CLK, LOAD).
- 74HC595, у базовій конфігурації, також потребує 3 лінії, але за рахунок програмного мультиплексування можуть знадобитися додаткові піни для керування розрядами (якщо частина з них не виведена з 74HC595).
- HT16K33 використовує 2 лінії I²C (A4, A5), які можна спільно використовувати з іншими I²C-пристроями.

3. Гнучкість та масштабованість.

- MAX7219 легко каскадується, але кожна мікросхема обмежена 8 розрядами.
- 74HC595 дозволяє масштабувати схему майже необмежено — додаючи нові регістри, можна керувати десятками й навіть сотнями ліній.
- HT16K33 дає змогу підключати кілька драйверів до однієї I²C-шини, змінюючи адреси, що спрощує побудову великих індикаторних полів.

4. Зручність налагодження.

- У варіанті з MAX7219 простіше відділити проблеми апаратні від програмних, оскільки драйвер є добре відлагодженим рішенням.
- У схемі з 74HC595 неправильна робота може бути наслідком як помилки в коді (таймінги, алгоритм мультиплексування), так і помилки в схемі (неправильні номінали резисторів, некоректні транзистори).
- HT16K33 потребує уваги до I²C-конфігурації, але у випадку коректного підключення працює досить надійно.

2.6. Висновок розділу 2

У другому розділі було зібрано «кістяк» всієї системи багаторозрядної індикації: обрано елементну базу, пояснено, чому я працюю саме з Arduino Nano, і на її основі побудувано три варіанти модуля індикації – з MAX7219, каскадом 74HC595 та драйвером HT16K33. Для кожного варіанта було намальовано структурну та принципову схеми, показано, куди саме під’єднуються піни Arduino, які потрібні резистори та живлення, і як організовано обмін даними. Окремо описав алгоритм відображення чисел: від розбиття значення на розряди до перетворення цифр у набір сегментів і передавання цих шаблонів у вибраний драйвер. У результаті отримано три працюючі конфігурації, які економно використовують виводи мікроконтролера, допускають розширення кількості розрядів і можуть бути безпосередньо зібрані у середовищі Proteus.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ РОБОТИ МОДУЛЯ БАГАТОРОЗРЯДНОЇ ІНДИКАЦІЇ

3.1. Організація експериментального дослідження

Експеримент будувався так, щоб побачити систему і “на екрані”, і в залізі. Програма індикації писалася під Arduino Nano, саме її й приймаємо як цільову платформу для вбудованого пристрою. Далі ця ж логіка запускалася у середовищі Proteus, де замість плати використовувалася бібліотечна модель ATmega328P. У Proteus збиралася схема, яка повторює реальний варіант: Arduino Nano, драйвер MAX7219 або каскад 74HC595 і багаторозрядний семисегментний індикатор. Там уже зручно дивитися часові діаграми, частоту оновлення, оцінювати навантаження по струму, не ризикуючи реальною платою.Варіант з MAX7219, де мультиплексування та струми сегментів бере на себе сам драйвер; варіант з кількома 74HC595, де всі перемикання розрядів та формування даних для сегментів виконує програма на Arduino,та варіант з HT16K33, д де вона, використовується швидше як орієнтир.

Така побудова експерименту дає можливість не просто переконатися, що всі три схеми взагалі працюють у різних режимах, а й зіставити їх між собою: де індикація стабільніша, де сильніше “ламається” від неправильних таймінгів і що з цього реально можна ставити у вбудовані пристрої з обмеженими ресурсами.

3.2. Програмне забезпечення системи

Програмна частина для багаторозрядної індикації тут відіграє не меншу роль, ніж сама схема. Від коду залежить, як часто оновлюються розряди, у якому порядку йдуть дані, чи є мерехтіння, як обробляються числа, що потрібно показати на екрані. Я по черзі розбираю, з яких частин складається програма під Arduino Nano і як вона працює з кожним із трьох варіантів індикації: MAX7219,

74HC595 та HT16K33. Окремо описується, як організовано обмін по SPI для MAX7219, послідовна передача даних у каскад 74HC595 та робота по I²C із HT16K33, а також які функції формують масиви даних для індикатора і хто «відповідає» за регулярне оновлення відображення під час роботи пристрою.

3.2.1. Загальна структура програмного забезпечення

Програмна частина умовно ділиться на кілька блоків, які працюють разом для трьох варіантів індикації — MAX7219, 74HC595 і HT16K33(Рис.3.1.):

1. Модуль ініціалізації. Налаштовує піни Arduino Nano, вмикає потрібні інтерфейси: SPI для MAX7219, лінії даних/такту для 74HC595, шину I²C для HT16K33. Тут же виконується початкове налаштування самих драйверів.
2. Модуль кодування символів. Містить таблиці сегментів для цифр і службових позначень. Одні й ті самі дані потім можуть віддаватися в MAX7219, 74HC595 або HT16K33.
3. Модуль передачі даних. Відповідає за надсилання байтів у вибраний драйвер: послідовна передача для MAX7219 і 74HC595, обмін по I²C для HT16K33.
4. Модуль мультиплексування (актуальний тільки для 74HC595). Забезпечує швидке перемикання розрядів у програмному режимі. Для MAX7219 і HT16K33 мультиплексування виконується всередині мікросхем.
5. Основний цикл програми. Оновлює значення для відображення, запускає тестові режими, показує прикладні дані (лічильники, виміряні параметри) і викликає потрібні функції для оновлення індикації.

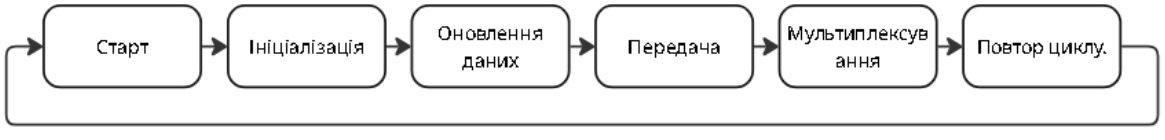

Рис.3.1. Загальна структура програмного забезпечення

3.2.2. Модуль ініціалізації системи

Цей модуль готує все “залізо” до роботи. Для MAX7219 усе досить просто: використовується SPI-подібний інтерфейс, кілька команд на конфігурацію — і драйвер готовий показувати цифри. У випадку з 74HC595 доводиться вручну налаштовувати щонайменше три лінії: дані, тактовий сигнал і строб (щоб зафіксувати байт на виходах). Для HT16K33 схема інша: піднімається шина I²C, задається адреса чипа, вмикається його внутрішній генератор, режим відображення та початковий рівень яскравості.

У спрощеному вигляді під час старту програма робить таке:

- переводить потрібні піни Arduino в режим виходу;
- виставляє початковий стан керуючих ліній (строб для 74HC595, сигнали для MAX7219);
- запускає SPI (для MAX7219) та I²C (для HT16K33).
- надсилає початкові команди: очистити індикатор, задати яскравість, увімкнути потрібні розряди.

```
  digitalWrite(CS_PIN, HIGH);
}

void initMAX7219() {
  sendCommand(0x09, 0x00);
  sendCommand(0x0A, 0x04);
  sendCommand(0x0B, 0x07);
  sendCommand(0x0C, 0x01);
  sendCommand(0x0F, 0x00);
}

void clearDisplay() {
```

Рис.3.2. Фрагмент коду ініціалізації драйвера MAX7219

3.2.3. Модуль передачі даних у MAX7219

Щоб щось з’явилося на індикаторі, Arduino має передати в MAX7219 пару байтів:

- перший байт — це адреса регістра (який саме розряд або службовий регістр ми змінюємо);
- другий байт — це власне дані (шаблон сегментів або параметр конфігурації).

У програмі для цього використовується окрема функція типу sendToMAX7219(addr, data) (або sendCommand()) (Рис3.3.), Її роботу можна описати покроково так:

1. Формування байтів на стороні Arduino Nano.

Спочатку в змінні готуються два значення:

- байт адреси регістра MAX7219 (наприклад, 0x01…0x08 для розрядів або службові адреси 0x09, 0x0A тощо);
- байт даних, де бітами задається, які сегменти мають світитися, або яке значення параметра потрібно встановити.

2. Вибір мікросхеми MAX7219.

Лінія LOAD (вона ж CS) опускається в низький рівень (LOAD = LOW).

Це сигнал для MAX7219, що зараз почнеться нова передача і дані треба приймати саме цій мікросхемі.

3. Передача адреси й даних по послідовному інтерфейсу.

Далі по лініях DIN та CLK послідовно надсилаються:

- спочатку байт адреси регістра;
- одразу за ним байт даних.

Передача йде по бітах, від старшого до молодшого (MSB → LSB). Після кожного біта Arduino змінює стан CLK, і MAX7219 «зсуває» отриманий біт усередину свого внутрішнього регістра.

4. Фіксація даних у внутрішніх регістрах.

Коли всі 16 біт передані, лінія LOAD повертається в високий рівень (LOAD = HIGH).

У цей момент MAX7219 «захлопує» внутрішній регістр: щойно отримані адреса й дані записуються у відповідний внутрішній регістр, а індикатор оновлює свій стан.

5. Повернення в основну програму.

Після цього функція просто завершується, керування повертається в основний цикл, де за потреби така передача повторюється для інших розрядів або регістрів конфігурації.

Таким чином, весь обмін зводиться до короткої послідовності дій: підготувати два байти, «притиснути» LOAD, відправити 16 біт по DIN/CLK, «відпустити» LOAD. Уся решта — як саме зберігати ці дані, як мультиплексувати розряди й керувати сегментами — уже робота самого драйвера MAX7219.

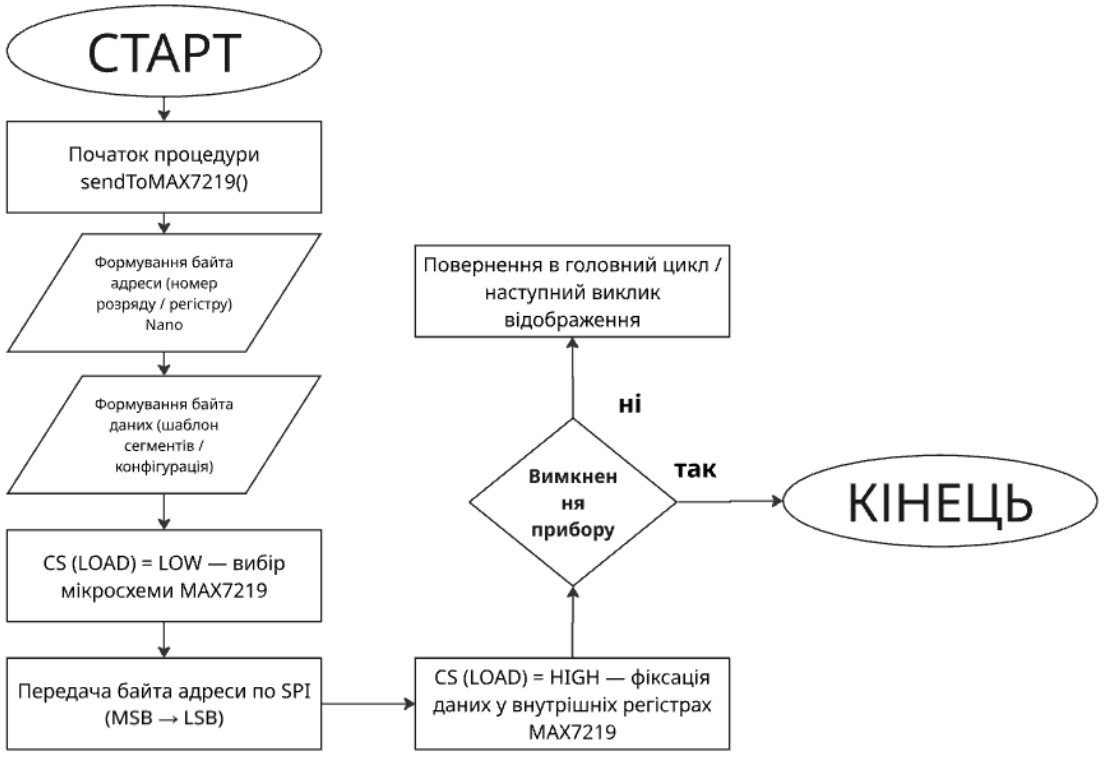

Рис.3.3. Передача даних у MAX7219

MAX7219 мультиплексує автоматично, тому цей пункт актуальний лише для варіанту з регістратором.

Мультиплексування включає:

- оновлення даних сегментів;
- вибір активного розряду через транзисторний ключ;
- швидке перемикання з частотою 200–500 Гц;
- компенсацію яскравості через регулювання часу активності.

3.2.4. Модуль передачі даних у 74HC595

У варіанті з 74HC595 ніякого «розумного» драйвера немає – саму картинку для індикатора повністю формує Arduino, а регістр зсуву тільки приймає біти й тримає їх на своїх виходах. Для обміну використовуються три основні лінії:

- лінія даних (SER / DATA),
- тактовий вхід (SRCLK / CLK),
- вхід фіксації стану виходів (RCLK / LATCH).

Якщо 74HC595 кілька (каскад), усі вони отримують спільні сигнали такту й «защіпки», а дані проходять через них послідовно(Рис.3.4.).

У програмі це оформлено як окрема процедура (умовно sendTo595()), яка щоразу виконує одну й ту саму послідовність дій:

1. Підготовка даних.

Спочатку формується байт (або кілька байтів) для сегментів і розрядів. Наприклад, один байт описує, які сегменти мають світитися, інший – який саме розряд зараз активний. Якщо є два регістри в каскаді, потрібно підготувати два байти; якщо три – відповідно три.

2. Скидання «защіпки».

Перед передачею даних сигнал LATCH виставляється в нуль. Це означає, що 74HC595 поки що не змінює свої виходи, а тільки приймає біти «всередину» зсувного регістра.

3. Послідовне зсування бітів.

Далі йде цикл, у якому по одному біту передається в каскад регістрів:

- на лінію DATA виставляється значення поточного біта (0 або 1);
- на вхід CLK подається короткий імпульс: спочатку LOW→HIGH, потім назад у LOW;
- при фронті тактового сигналу біт «зсувається» всередину 74HC595, а попередні біти рухаються далі по ланцюжку.

4. Цей крок повторюється стільки разів, скільки біт потрібно передати: 8 для одного регістра, 16 для двох, 24 — для трьох і т.д. У коді це звичайний цикл for по кількості бітів або виклик shiftOut().

5. Фіксація стану виходів.

Після того як усі біти «загнані» в регістри, лінія LATCH переводиться в стан HIGH. У цей момент вміст внутрішнього зсувного регістра копіюється на виходи 74HC595, і на індикаторі з’являється новий візерунок сегментів/розрядів.

6. Повернення в основний цикл.

Процедура закінчується, керування повертається в головний цикл програми або в модуль мультиплексування, який через деякий час знову викличе sendTo595() з новим набором бітів.

Рис.3.4. Передачі даних у 74HC595

3.2.5. Модуль передачі даних у HT16K33

Для HT16K33 передача даних виглядає інакше, ніж у MAX7219 чи 74HC595. Тут мікроконтролер не «мигає» розрядами сам і не ганяє біти по SPI, а просто записує вбудовану пам’ять відображення драйвера по шині I²C. Далі HT16K33 сам виконує мультиплексування й керує сегментами.

У HT16K33 є внутрішня RAM дисплея: послідовність байтів, де кожен біт відповідає певному сегменту (Рис.3.5.).

Схема проста:

- вибираємо I²C-адресу чипа (типово 0x70, 0x71 тощо, залежно від підключення A0…A2);
- задаємо початкову адресу в його пам’яті відображення (зазвичай 0x00);
- по черзі надсилаємо байти, які описують стан сегментів усіх розрядів.

Як тільки байти записані в RAM HT16K33, драйвер автоматично оновлює індикатор.

Послідовність роботи процедури sendToHT16K33()

У програмі це оформлюється як окрема функція, яка робить таке:

1. Формування буфера відображення.

На основі потрібних цифр (наприклад, 1-2-3-4) і таблиці сегментів готується масив байтів.

Кожен байт або пара байтів описує, які сегменти мають світитися в певному розряді.

2. Початок I²C-транзакції.

Викликається Wire.beginTransmission(адреса_HT16K33).

Arduino бере на себе формування стартового сигналу на шині SDA/SCL.

3. Вказівка стартової адреси пам’яті.

Першим байтом у передачі надсилається номер комірки RAM, з якої починаємо запис (зазвичай 0x00).

HT16K33 сприймає це як «курсор», з якого далі автоінкрементує адресу.

4. Послідовне надсилання даних сегментів.

Далі в циклі відправляються всі підготовлені байти буфера:

Wire.write(buffer[i]);

Для кожного наступного байта HT16K33 сам переходить до наступної адреси в пам’яті.

5. Завершення передачі.

Викликається Wire.endTransmission().

На шині формується STOP-умова, а драйвер оновлює відображення згідно з новим вмістом своєї RAM.

6. Повернення в основний цикл.

Функція закінчується, керування повертається в loop(). Через певний час (наприклад, у кожному циклі або по таймеру) процедура знову викликається, але вже з новими даними.

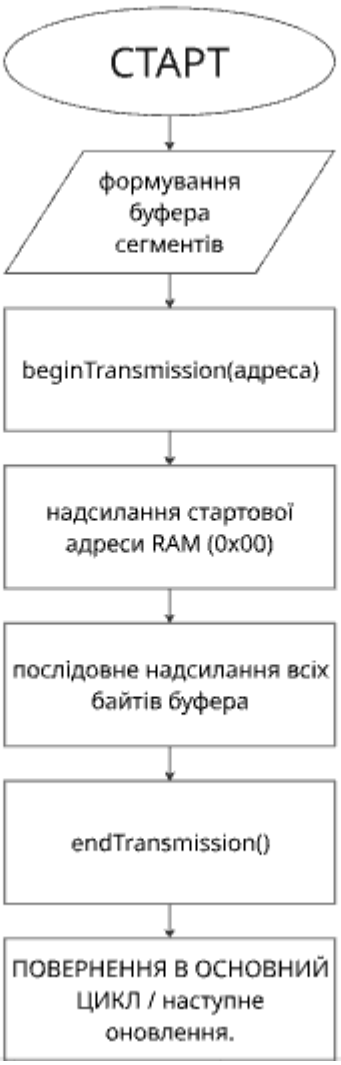

Рис.3.5. Передачі даних у HT16K33

3.2.6. Основний цикл роботи

У циклі loop() виконується (Рис.3.6.):

- отримання нового значення (лічильник, сенсор, ввід);
- розкладання числа на цифри;
- передача сегментних шаблонів;
- оновлення індикатора;
- тестування сегментів (за потреби).

```
  initMAX7219();
}

void loop() {
  static unsigned long lastUpdate = 0;
  static int counter = 0;

  unsigned long now = millis();

  if (now - lastUpdate > 200) {
    lastUpdate = now;
    counter++;
    if (counter > 99999999) {
      counter = 0;
    }
  }

  byte digits[8];
  splitNumber(counter, digits);

  updateDisplay(digits);
}
```

Рис.3.6. цикл loop()

3.2.7. Завершальна інтеграція програмно-апаратних засобів

Після того як окремі шматки програми вже написані й перевірені окремо (ініціалізація портів, робота з MAX7219, варіант із 74HC595, варіант із HT16K33, розкладання числа на окремі цифри, мультиплексування), їх потрібно «зібрати докупи».

На цьому кроці важливо подивитись, як система поводиться вже не по частинах, а в цілому: Arduino Nano, драйвер, індикатор і живлення разом.

1. Перевірка, чи правильно показуються цифри

Спочатку роблять найпростішу річ — дивляться, чи індикатор взагалі вміє чесно показати цифри.

Зазвичай набирають невеликий тестовий скетч, у якому по кроках відбувається таке:

1. На всі розряди виводиться 0, потім 1, потім 2 … до 9.
2. Далі виводяться кілька «довгих» чисел:
   - 00001234;
   - 00009999;
   - 12345678 (якщо є 8 розрядів);
   - число з усіма сегментами, наприклад 8888.

Для кожного з трьох варіантів підключення (через MAX7219, через 74HC595, через HT16K33) дивляться, щоб:

- цифри не мінялися місцями між розрядами;
- не було переплутаних сегментів (наприклад, замість 3 світиться щось схоже на 9);
- не з’являлися «обірвані» цифри, коли один сегмент не загоряється зовсім.

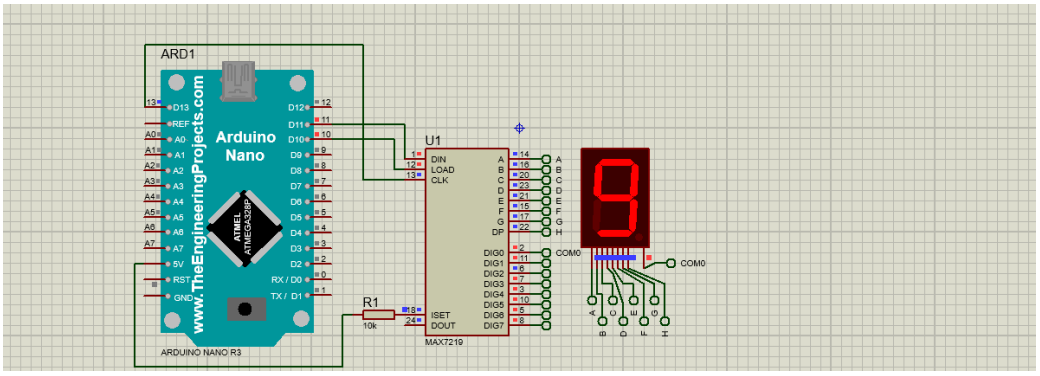

Рис.3.7. Семисегментний індикатор підключений через MAX7219

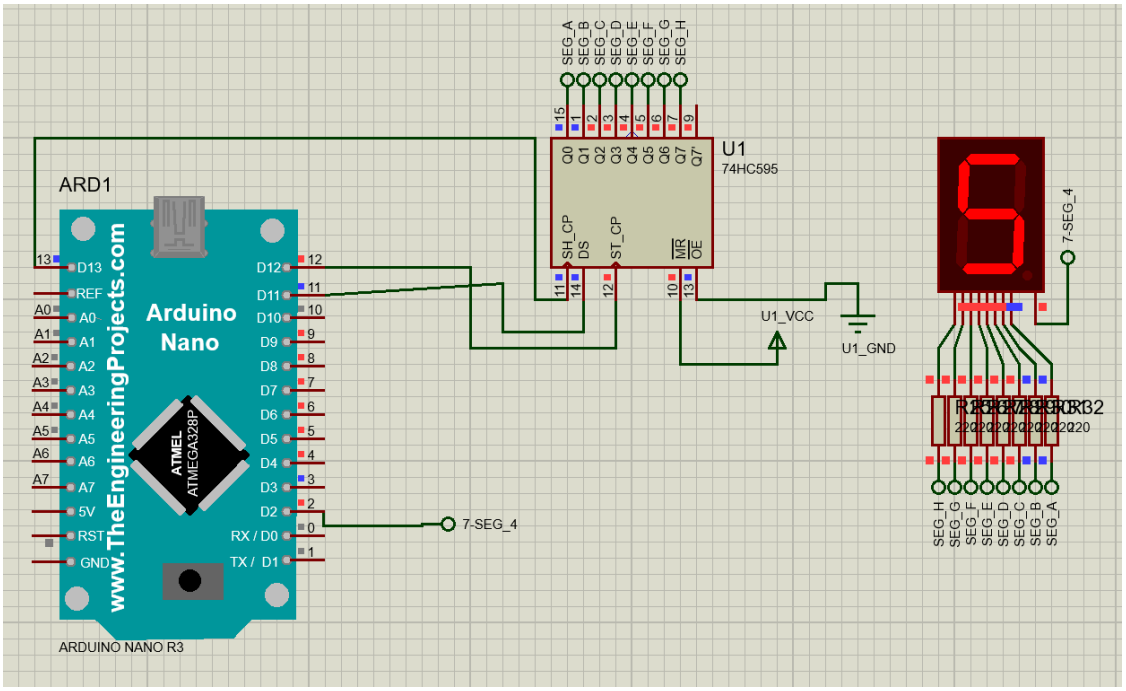

Рис.3.8. Семисегментний індикатор підключений через 74HC595

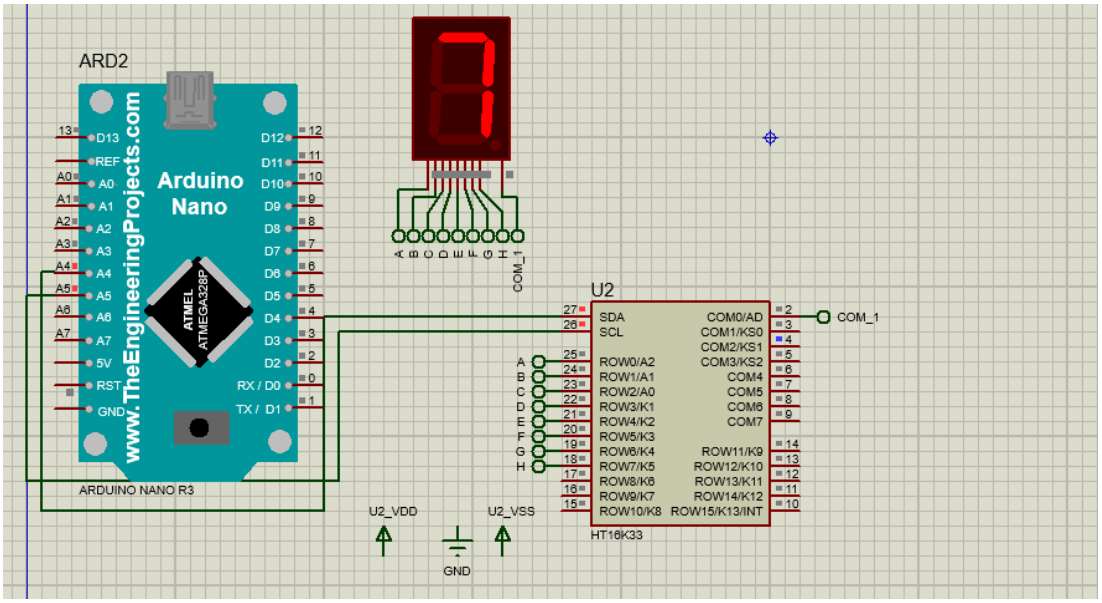

Рис.3.9. Семисегментний індикатор підключений через HT16K33

2. Перевірка часових діаграм (timing)

Коли з цифрами все добре, переходять до «невидимої» частини — сигналів на лініях даних.

У Proteus підключають віртуальний осцилограф або логічний аналізатор до потрібних виводів і дивляться, що реально відбувається під час передачі даних.

- Для MAX7219 аналізують лінії DIN, CLK і CS (LOAD):

дивляться, щоб на DIN дані встигали виставитися до фронту CLK,

щоб імпульси такту не були надто вузькі,

щоб CS опускався перед передачею і піднімався тільки після того, як пройшли два байти (адреса + дані).

- Для 74HC595 контролюють три сигнали: DS (дані), SH_CP (зсув / такт), ST_CP (защіпка / фіксація).

Спочатку на DS подається біти майбутнього байта,

кожен фронт SH_CP «просуває» їх усередині регістра,

а ST_CP одним імпульсом «перекидає» весь набір бітів на виходи.

На діаграмі легко побачити помилку: якщо ST_CP спрацьовує занадто рано, на виходи потрапляють неповні дані.

- Для HT16K33 дивляться послідовність на шині I²C — лінії SDA і SCL:

чи є старт-умова (S), байт адреси з підтвердженням ACK, потім байти даних, і наприкінці стоп-умова (P);

чи не «зливаються» імпульси на SCL, чи не обривається сигнал на SDA посеред байта.

Якщо в якихось місцях імпульси виходять занадто короткими або сигнали перекриваються не в той момент, це видно на осцилограмі, і можна підкоригувати затримки в коді.

3. Частота оновлення і мерехтіння

Далі оцінюють, наскільки часто оновлюється індикація і чи не бачить око мерехтіння.

- У MAX7219 мультиплексування робить сам драйвер. Arduino просто періодично змінює вміст його внутрішніх регістрів. Тут головне — не відправляти команди занадто часто, щоб не «засипати» чип постійними оновленнями без потреби.
- У 74HC595 усе навпаки: за мультиплексування відповідає сама програма. В основному циклі або в перериванні мікроконтролер:
  - обирає, який розряд зараз буде активний;
  - відправляє через 74HC595 шаблон сегментів для цієї цифри;
  - через невелику затримку переходить до наступного розряду.

Якщо робити це повільно — розряди починають помітно блимати. Якщо занадто швидко — зростає навантаження на процесор.

- У HT16K33 мультиплексування і ШІМ яскравості також реалізовані всередині драйвера. Програма лише оновлює його «екранну пам’ять» (display RAM). Тут важливо, щоб оновлення відбувалося досить часто для плавної зміни показів, але без зайвих повторів.

У моделюванні просто дивляться на індикатор: якщо цифри «тремтять», залишають шлейф або «розсипаються» під час змін — щось не так із частотою оновлення, і цикл потрібно переробити.

4. Яскравість і споживання енергії

Наступне питання — наскільки сильно світиться індикатор і скільки для цього потрібно струму.

- У MAX7219 яскравість задається через регістр INTENSITY. У коді по черзі пробують декілька рівнів і дивляться, при якому індикатор ще добре читається, але вже не «сліпить» і не навантажує живлення максимально.

- У 74HC595 окремого регістра немає, тому яскравість фактично залежить від того, як довго кожен розряд залишається увімкненим у циклі мультиплексування.

Можна, наприклад, зменшити затримку активного стану або додати короткі паузи між розрядами — це знижує середній струм і трохи затемнює індикатор.

- У HT16K33 рівень яскравості задається через його внутрішні регістри ШІМ. Там є кілька фіксованих ступенів, які також підбираються експериментально.

Завдання цього етапу — знайти режим, у якому індикатор:

- добре читається з робочої відстані;
- не світить «на повну потужність» без потреби;
- не перевантажує блок живлення, коли засвічено відразу багато сегментів.

5. Поведінка при зміні живлення

Останній момент, який дивляться під час інтеграції, — як система веде себе при різних умовах живлення.

У симуляції можна:

- трохи зменшити напругу живлення;
- увімкнути на індикаторі «важкі» шаблони (наприклад, одночасно багато цифр 8);
- збільшити частоту оновлення.

Потім спостерігають, чи не:

- «зникають» окремі сегменти;
- змінюється яскравість між розрядами;
- починає спотворюватися форма цифр.

Для MAX7219, 74HC595 і HT16K33 це актуально однаково: усі вони в реальному пристрої живляться від того самого джерела, що й решта схеми. Тому в дипломі важливо показати, що питання живлення не ігнорується, а принаймні базово проглянуте ще на етапі моделювання.

3.3. Методика проведення експерименті

Методику експерименту будували так, щоб однаково перевірити всі варіанти модуля індикації на Arduino Nano і мати можливість чесно їх порівняти. Для кожного типу модуля — на MAX7219, на каскаді 74HC595 і на HT16K33 — послідовно проганяли одні й ті самі тести. Схеми збиралися з різною кількістю розрядів: спочатку з одним семисегментним індикатором, потім з чотирма, а далі з вісьмома. Для MAX7219 і 74HC595 усі режими спершу відпрацьовували в Proteus, де зручно дивитися часові діаграми й поведінку індикатора. Варіант із HT16K33 додатково перевіряли на реальному стенді з Arduino Nano. Результати кожного запуску фіксували у вигляді скріншотів та зводили в таблиці, щоб далі було простіше порівнювати варіанти між собою.

3.3.1. Побудова віртуального стенду в середовищі Proteus

Для варіантів на MAX7219 і 74HC595 усі тести спершу робили в Proteus. Узяли стандартну бібліотечну модель ATmega328P (плата Arduino Uno, що за можливостями відповідає Arduino Nano) і на її основі зібрали дві типові схеми.

У першій схемі до контролера під’єднано драйвер MAX7219 і семисегментні індикатори. У другій — той самий контролер, але вже каскад 74HC595, транзисторні ключі по розрядах і семисегментні індикатори. Для кожної з цих схем проганяли три варіанти: один індикатор, чотири та вісім розрядів. Умови всюди однакові: живлення 5 В, індикатори з загальним катодом, однакові

резистори обмеження струму. Додатково в схемах ставили віртуальний осцилограф, щоб дивитися часові діаграми, і амперметр — для оцінки споживаного струму.

3.3.2. Конфігурації індикаторів (4 та 8 розрядів)

Щоб побачити, як змінюється робота модуля індикації, тести запускали в кількох варіантах кількості розрядів, усі тести виконувалися у трьох конфігураціях:

- 4×7-сегментні індикатори (4 розряди).

Типовий випадок для реальних пристроїв: лічильники, таймери, прості панелі відображення. На цьому рівні вже добре видно, як працює мультиплексування: чи однаково світяться всі розряди, наскільки схема «чутлива» до затримок у коді і чи не з’являється помітне мерехтіння при оновленні даних.

- 8×7-сегментні індикатори (8 розрядів).

Конфігурація з максимальною кількістю розрядів для одного MAX7219/HT16K33 і показовий випадок для 74HC595. Саме тут найкраще видно обмеження по частоті оновлення, навантаженню на мікроконтролер і енергоспоживанню.

Для кожної з цих конфігурацій виконувалися всі подальші тести. Результати для 4 та 8 розрядів у рамках одного тесту оформлювалися у вигляді окремих скріншотів.

3.3.3. Тест 1 – відображення цифр 0–9

Мета цього тесту – перевірити базову коректність відображення цифр для кожного модуля індикації в конфігураціях на 1, 4 та 8 розрядів.

3.3.3.1. Тест 1 для модуля на MAX7219

У прошивці для MAX7219 реалізовано послідовний показ цифр від 0 до 9 з фіксованою затримкою між змінами. Для кожної конфігурації (4, 8 розрядів):

У конфігурації 4×7-сегментні індикатори активувались чотири розряди (DIG0–DIG3). У режимі Тесту 1 усі чотири розряди відображали одне й те саме число (поточну цифру 0–9). Це дозволило оцінити не лише правильність кодування, а й рівномірність яскравості між розрядами. За результатами моделювання всі цифри в усіх чотирьох позиціях мали однакову яскравість, не спостерігалося «провалів» чи затримок в окремих розрядах. MAX7219 коректно виконував внутрішнє мультиплексування, тож при зміні цифр усі розряди оновлювалися синхронно, без видимого «підморгування» чи «виїдання» сегментів (Рис 3.10.).

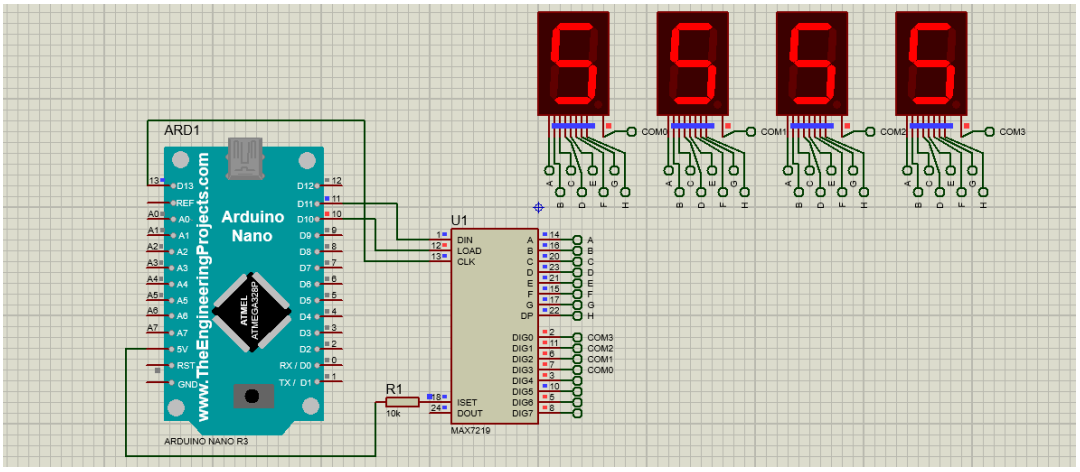

Рис 3.10. Arduino Nano з MAX7219 на 4 7-розрядних індикатори, відображення цифр 0–9

Найбільш показовою є конфігурація 8×7-сегментні індикатори, у якій використовуються всі доступні розряди MAX7219. У цьому режимі також реалізовувався одночасний показ однієї цифри у всіх восьми позиціях з послідовною зміною від 0 до 9.Під час моделювання видно, що навіть коли MAX7219 «зайнятий» усіма розрядами, цифри малюються правильно: форма однакова, сегменти не плутаються, яскравість на око рівна по всьому індикатору.

Жодних збоїв чи дивних артефактів не з’являється. Вбудоване мультиплексування працює досить швидко, тому навіть при 8 розрядах індикація виглядає майже статичною, без помітного мерехтіння (рис. 3.11).

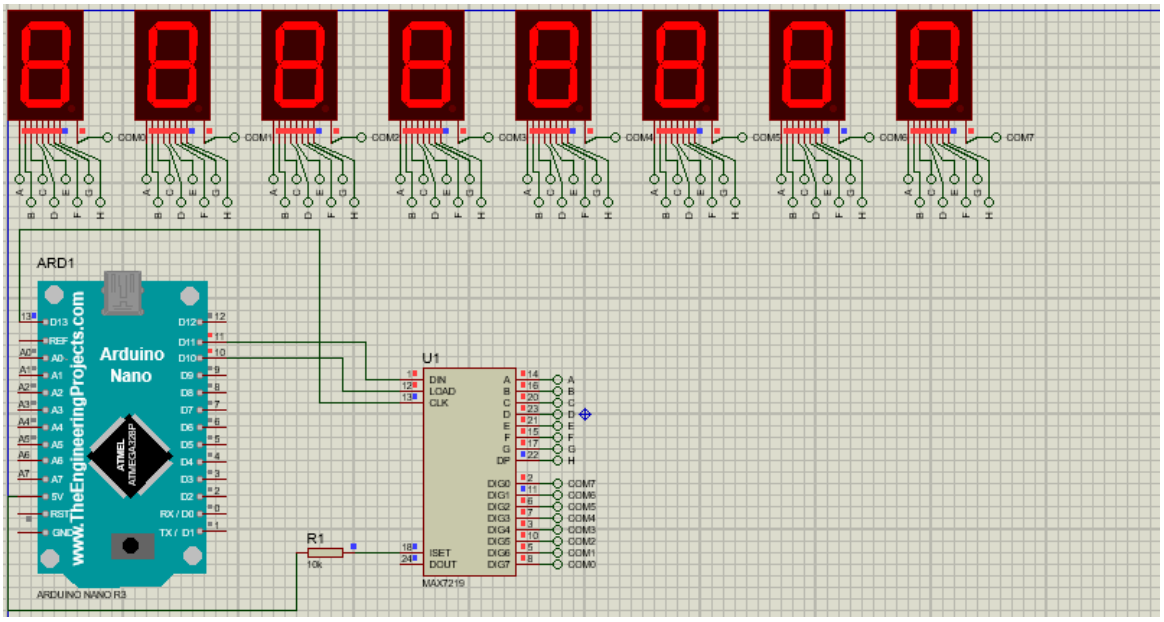

Рис 3.11. Arduino Nano з MAX7219 на 8 7-розрядних індикатори, відображення цифр 0–9

Таблиця 3.1 – Arduino Nano з MAX7219 тест, на відображення коректності

№	Кількість розрядів	Опис конфігурації	Коректність відображення цифр 0–9	Рівномірність яскравості між розрядами	Артефакти при переходах (0–9
1	4 розряди	4×7-сегментні індикатори, програмне мультиплексування	Коректна у всіх розрядах	Візуально однакова за оптимальних затримок	Не виявлено
2	8 розрядів	8×7-сегментні індикатори, повне навантаження по розрядах	Коректна при належному налаштуванні таймінгів	Можлива невелика нерівномірність за невдалих затримок	Не виявлено

3.3.3.2. Тест 1 для модуля на 74HC595

Для конфігурації з 74HC595 той самий алгоритм послідовного показу 0–9 використовується спільно з програмним мультиплексуванням розрядів. Тест виконується для 4 та 8 розрядів.

У конфігурації 4×7-сегментні індикатори до схеми додано транзисторні ключі розрядів, а мікроконтролер циклічно опитує чотири позиції: для кожного розряду через 74HC595 подається відповідний шаблон сегментів, активується потрібний ключ і після короткої паузи виконується перехід до наступного розряду. За правильно підібраних затримок усі чотири розряди показували однакову цифру (поточне значення 0–9), видимого мерехтіння не було, відмінностей за яскравістю між розрядами не спостерігалося. Це підтверджує працездатність реалізованого алгоритму мультиплексування для невеликої кількості розрядів (Рис 3.12.).

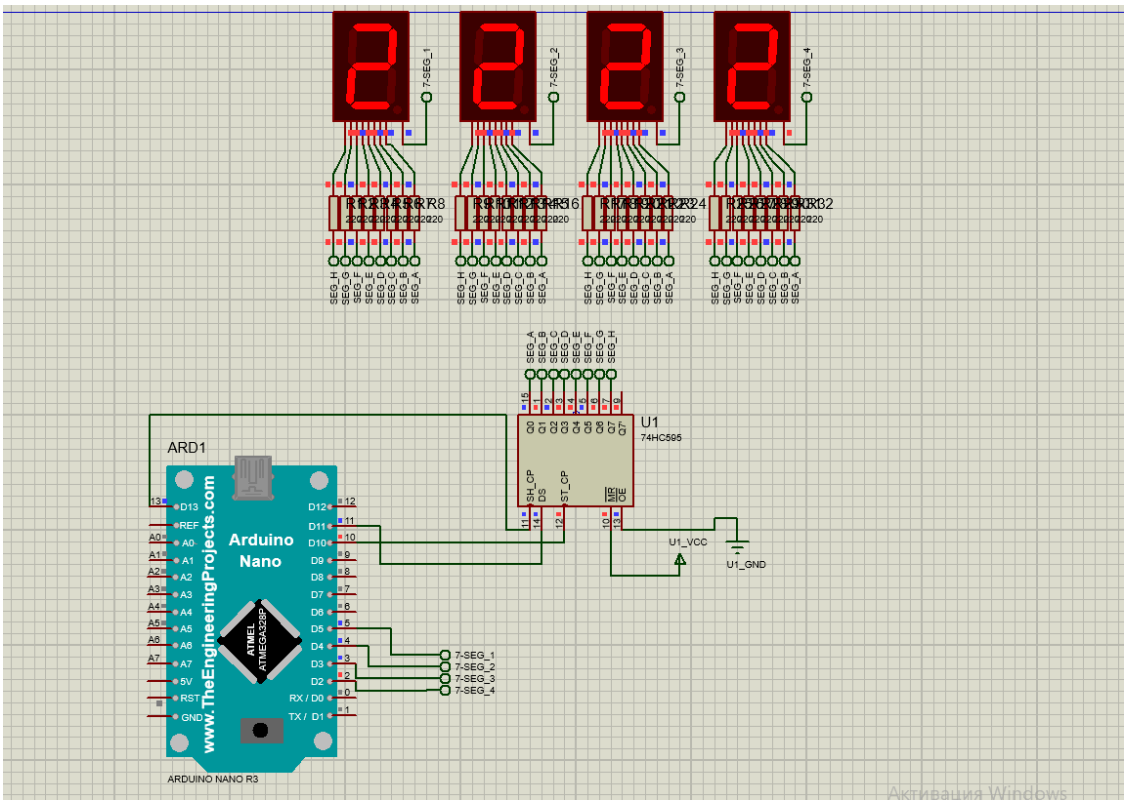

Рис 3.12. Arduino Nano з 74HC595 на 4 7-розрядних індикатори, відображення цифр 0–9

Найбільш показовою виявилася конфігурація 8×7-сегментні індикатори, у якій один або кілька каскадованих 74HC595 обслуговують уже вісім розрядів. При оптимальних значеннях затримок усі вісім розрядів коректно відображали цифри 0–9, а артефактів типу «пересмикування» сегментів або спотворення форми символів не зафіксовано. Водночас у цій конфігурації чітко проявляється чутливість системи до налаштувань таймінгів: при спробі збільшити затримки або додати в основний цикл додаткову логіку з’являються перші ознаки мерехтіння та невелика нерівномірність яскравості між розрядами. Таким чином, Тест 1 показав, що зв’язка Arduino Nano + 74HC595 забезпечує правильне відображення цифр 0–9 в усіх трьох конфігураціях, але для 8-розрядної індикації вимагає більш уважного підбору частоти мультиплексування й обмеження фонових обчислень у мікроконтролері (Рис 3.13.).

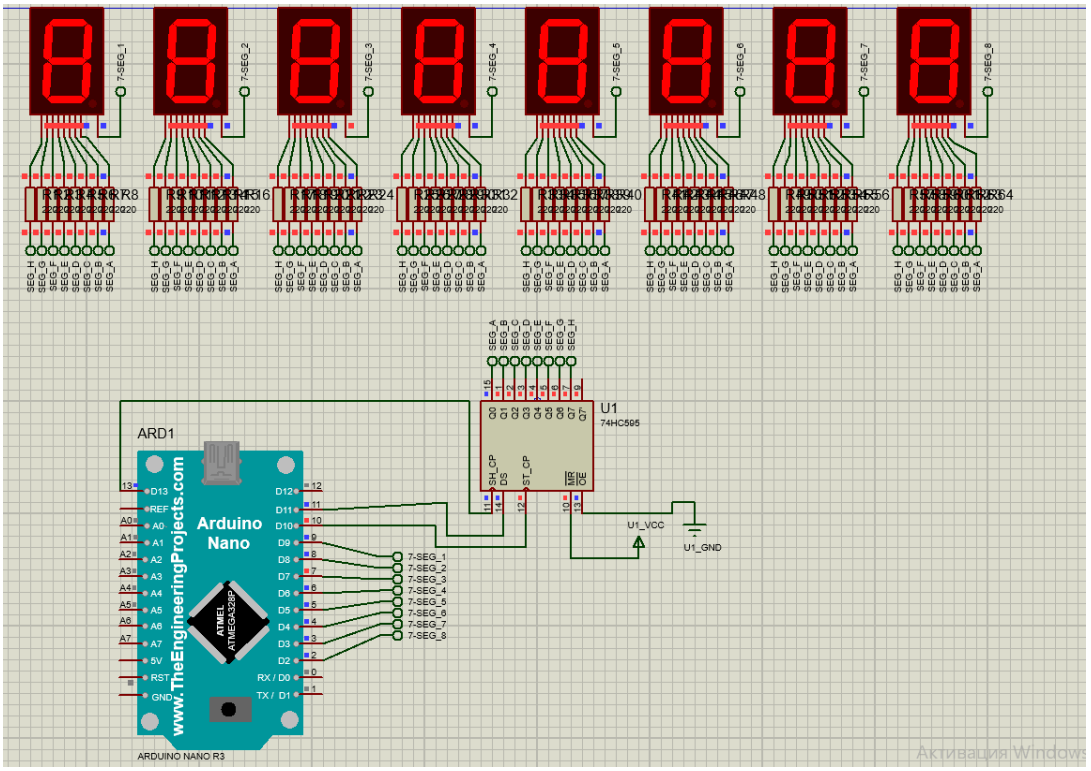

Рис 3.13. Arduino Nano з 74HC595 на 8 7-розрядних індикатори, відображення цифр 0–9

Таблиця 3.2 – Arduino Nano з 74HC595 тест, на відображення коректності

№	Кількість розрядів	Опис конфігурації	Коректність відображення цифр 0–9	Рівномірність яскравості між розрядами	Артефакти при переходах (0–9
1	4 розряди	4×7-сегментні індикатори, програмне мультиплексування	Коректна у всіх розрядах	Візуально однакова за оптимальних затримок	Не виявлено
2	8 розрядів	8×7-сегментні індикатори, повне навантаження по розрядах	Коректна при належному налаштуванні таймінгів	Можлива невелика нерівномірність за невдалих затримок	Не виявлено

3.3.3.3. Тест 1 для модуля на HT16K33

У тесті 1 для модуля на HT16K33 перевірялася найпростіша річ — чи вміє він нормально показувати цифри 0–9 на всіх розрядах. Для цього зібрали стенд: Arduino Nano, підключений до HT16K33 по шині I²C (лінії SDA і SCL під'єднані до A4 і A5), та модуль із семисегментними індикаторами. Після цього на Arduino було завантажено програму, яка через однаковий проміжок часу змінює цифру і виводить її одразу на всі розряди — спочатку 0, потім 1, 2, …, 9 і знову по колу.

Для конфігурації 4×7-сегментні індикатори було підтверджено, що всі цифри 0–9 відображаються без помилок, форма символів відповідає еталонній, а яскравість усіх чотирьох розрядів візуально однакова. Завдяки вбудованому в HT16K33 апаратному мультиплексуванню та власному генератору тактових імпульсів індикація залишалася стабільною навіть при зменшенні інтервалу між зміною цифр, мерехтіння або «провалів» яскравості не спостерігалося (Рис 3.14.).

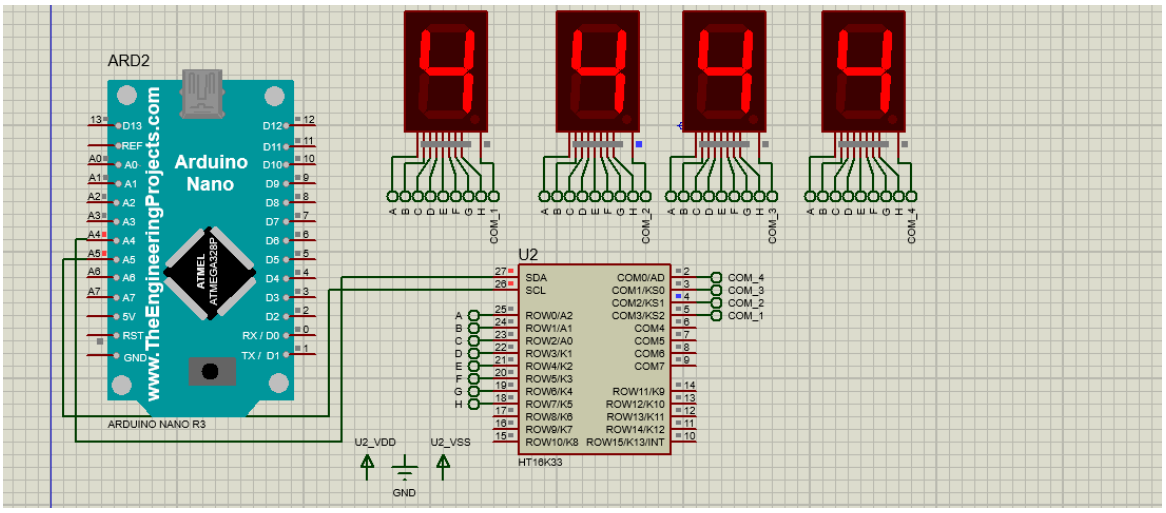

Рис 3.14. Arduino Nano з HT16K33 на 4 7-розрядних індикатори, відображення цифр 0–9

У варіанті з вісьмома розрядами до HT16K33 під’єднано вісім семисегментних індикаторів. Код на Arduino Nano при цьому майже не чіпався: контролер просто періодично надсилає по I²C нові значення цифр, а сам HT16K33 всередині вже розподіляє сегменти й сканує розряди. У тесті з послідовністю 0–9 усі вісім позицій змінювалися синхронно, без різниці в яскравості між початком і кінцем індикатора. Коли на всі розряди виводили «8», індикатор не перегрівався і крайні цифри не ставали тьмянішими — це показує, що обмеження струму й апаратне мультиплексування працюють коректно. Окремо видно, що навантаження на Arduino Nano тут менше, ніж у варіанті з 74HC595: замість постійно «крутити» мультиплексування, мікроконтролер лише формує I²C-пакети з новими числами (рис. 3.15).

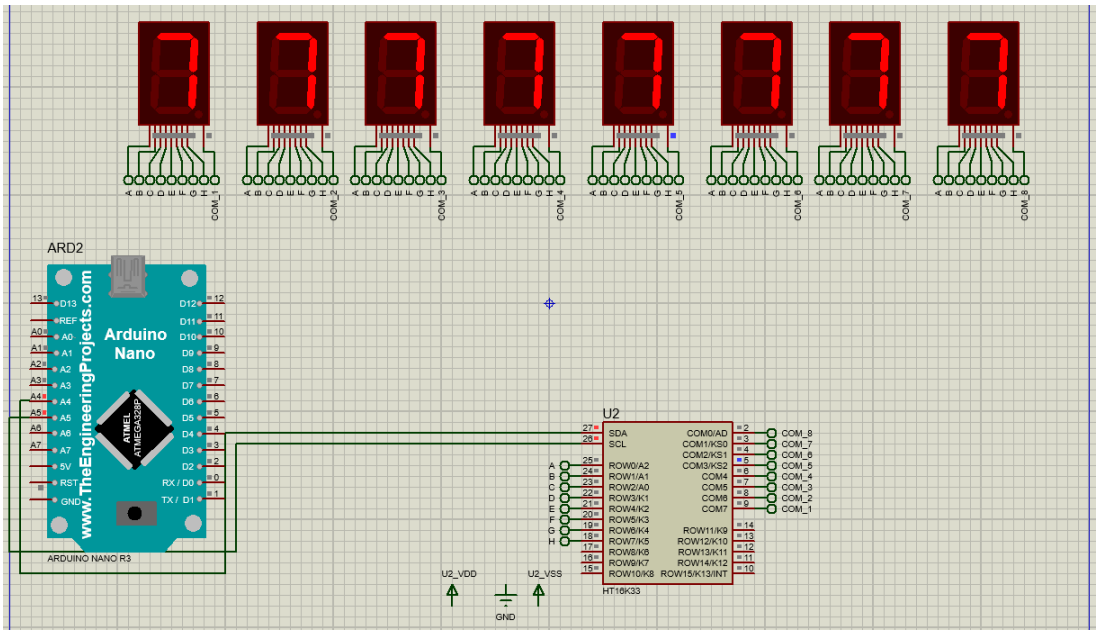

Рис 3.15. Arduino Nano з HT16K33 на 8 7-розрядних індикатори, відображення цифр 0–9

Таблиця 3.3 – Arduino Nano з HT16K33 тест, на відображення коректності

№	Кількість розрядів	Опис конфігурації	Коректність відображення цифр 0–9	Артефакти при переходах (0–9, 9–0)
1	4 розряди	4×7-сегментні індикатори, HT16K33 + Arduino Nano по I²C	Яскравість візуально однакова в усіх розрядах	Не виявлено
2	8 розрядів	8×7-сегментні індикатори, той самий HT16K33	Яскравість рівномірна по всій довжині індикатора, крайні розряди не тьмяніють	Не виявлено

3.3.3.4. Висновок тесту 1

Шкала (для всіх трьох параметрів):

- 5 – відмінно / без зауважень
- 4 – добре, але залежить від налаштувань
- 3 – помітні дрібні недоліки
- (2, 1 можна не використовувати, бо у твоїх даних усе працює досить добре)

У першій таблиці дивимося, як поводяться три варіанти керування чотирирозрядною індикацією:

- MAX7219 показує себе найпростішим варіантом: цифри 0–9 відображаються без помилок, усі чотири розряди світяться однаково, нічого не смикається й не мерехтить. Для користувача це виглядає як спокійний, “статичний” індикатор.
- 74HC595 теж правильно виводить всі цифри, але тут більше залежить від прошивки. Якщо затримки в коді підібрані вдало — яскравість по розрядах майже однакова. Якщо ні — окремі розряди можуть світитися трохи по-іншому, хоча явних збоїв чи “битих” сегментів немає.
- HT16K33 на 4 розрядах працює приблизно так само, як і MAX7219. Цифри показує правильно, усі розряди світяться однаково, при пробігу 0–9 нічого не смикається й не мерехтить.

Тож для чотирьох розрядів загалом підходять усі три варіанти, але MAX7219 і HT16K33 дають нормальний результат майже одразу “з коробки”. А от з 74HC595 треба трохи повозитися в коді з затримками, щоб картинка виглядала так само рівно.

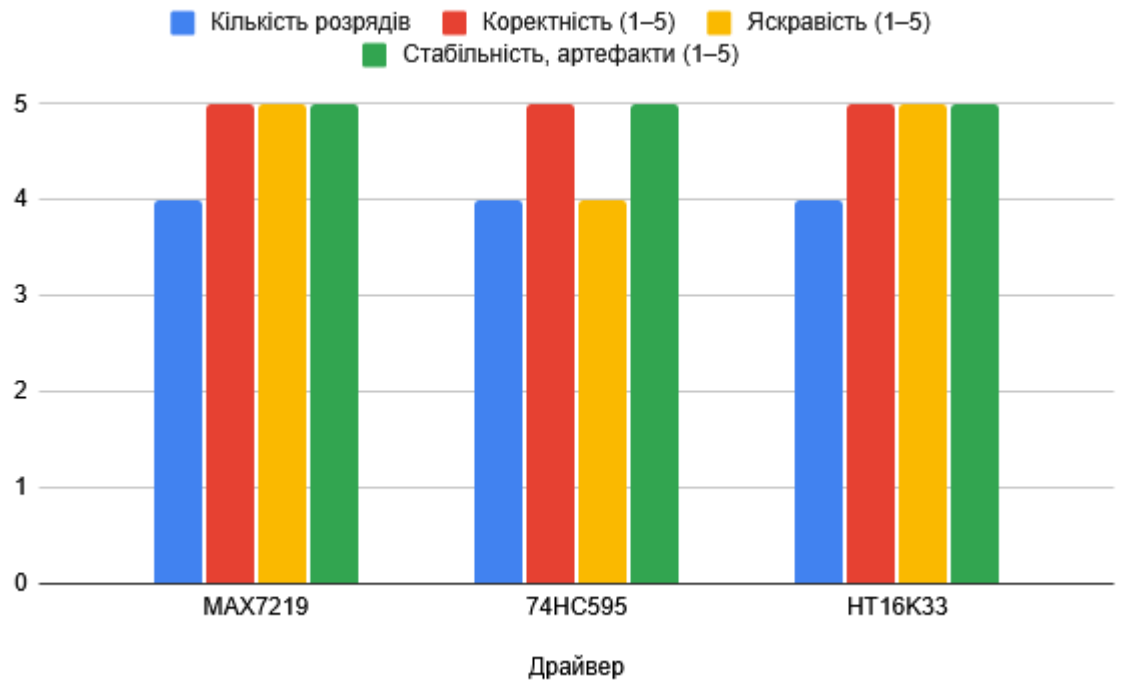

Рис.3.16. Загальна коректність відображення 4-розрядів семисегментних індикаторів

Друга таблиця показує, що змінюється, коли замість 4 розрядів ми виводимо вже 8. Тут різниця між підходами стає помітнішою (Рис.3.17.).

- **MAX7219** навіть на 8 розрядах зберігає ту саму якість: цифри відображаються правильно, яскравість однакова по всій довжині індикатора, ніяких артефактів при переходах немає. Внутрішнє апаратне мультиплексування робить свою справу, тому індикація виглядає “суцільною”, без тремтіння.
- **74HC595** у конфігурації з 8 розрядами вже чутливіший до налаштувань. За правильних таймінгів цифри показуються коректно, але, якщо затримки підібрані не дуже вдало, може з’являтися невелика нерівномірність яскравості між розрядами. Тобто схема працює, але стає видно, що все мультиплексування “на плечах” мікроконтролера та прошивки.
- **HT16K33** на 8 розрядах тримає рівень: цифри від 0 до 9 відображаються правильно, яскравість рівномірна по всіх сегментах, крайні

розряди не тьмяніють навіть коли засвічено багато сегментів (наприклад, одні “вісімки”). Видимих артефактів при швидких переходах не спостерігається.

У підсумку для 8-розрядної індикації можна сказати так: MAX7219 і HT16K33 добре тягнуть повний індикатор без додаткових “танців з бубном”, тоді як варіант на 74HC595 працює, але вже чітко показує залежність від якості програмної реалізації мультиплексування. Це і відображено у діаграмах оцінками за коректність, яскравість та стабільність.

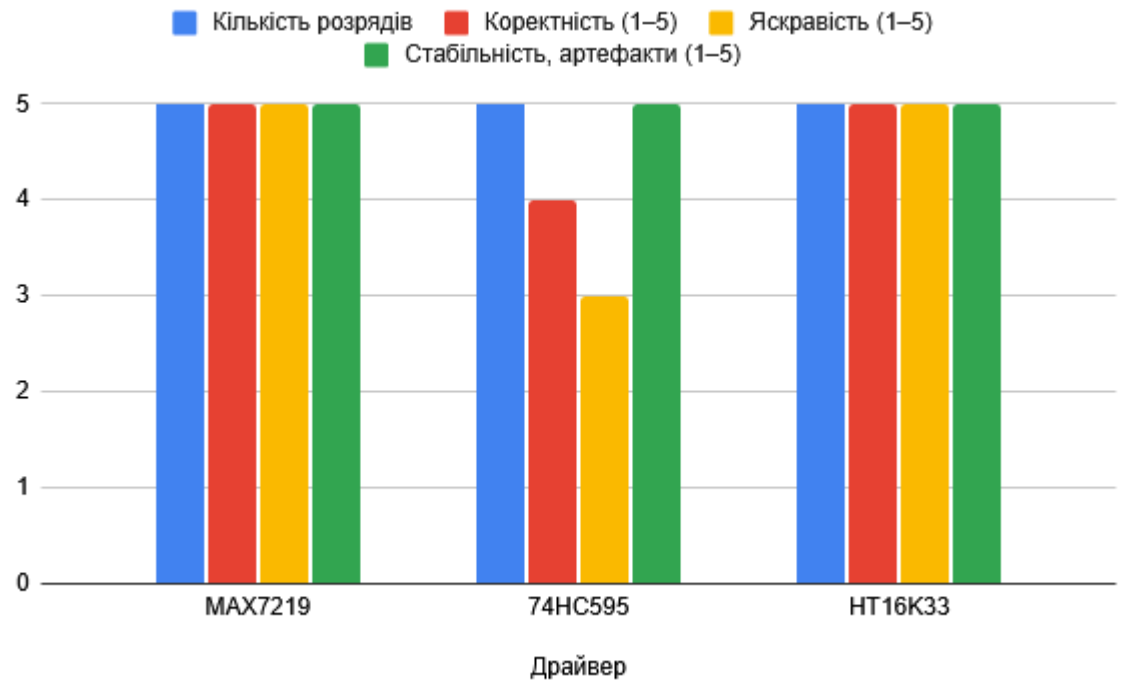

Рис.3.17. Загальна коректність відображення 8-розрядів семисегментних індикаторів

3.3.4. Тест 2 – часові діаграми та частота оновлення

У цьому тесті досліджуються часові характеристики роботи модулів індикації, зокрема частота оновлення та потенційне мерехтіння. Для прикладу, у підпункті 3.3.5 для кожної схеми (MAX7219, 74HC595, HT16K33) та конфігурацій

1, 4 і 8 розрядів передбачається по 3 осцилограми (усього 9 скріншотів) і по 1 таблиці з числовими параметрами.

3.3.4.1. Тест 2 для MAX7219

У Тесті 2 для модуля на базі MAX7219 досліджувались часові діаграми сигналів керування та фактична частота оновлення індикації. В середовищі Proteus до ліній DIN, CLK та LOAD/CS були підключені канали віртуального осцилографа. Для кожної з трьох конфігурацій (1, 4 та 8 семисегментних індикаторів) на індикаторі запускалася та сама тестова програма з поцикловим виведенням цифр 0–9, після чого аналізувались моменти передавання даних у MAX7219 та поведінка індикатора між цими циклами. Осцилограми показали, що пакети даних на лініях DIN/CLK з’являються лише в момент зміни значення (переходу до наступної цифри), після чого MAX7219 самостійно забезпечує внутрішнє мультиплексування розрядів із сталою високою частотою, яка не залежить від кількості задіяних індикаторів.

Для конфігурації 4×7-сегментні індикатори на осцилограмах чітко видно, що характер сигналів на лініях DIN, CLK і LOAD не змінюється порівняно з однорозрядним варіантом: при переході до наступної цифри формується короткий пакет послідовних імпульсів CLK із відповідними станами DIN, після чого коротким імпульсом активується LOAD, і нові дані фіксуються у внутрішніх регістрах MAX7219. Подальше оновлення всіх чотирьох розрядів здійснюється вже всередині драйвера, тому на керуючих лініях у проміжку між змінами цифр панує «тиша», а індикація залишається повністю стабільною. Візуальне спостереження за індикатором підтвердило, що при такому режимі частота внутрішнього мультиплексування є достатньо високою: усі чотири розряди світяться рівномірно, мерехтіння не сприймається (Рис.3.18.).

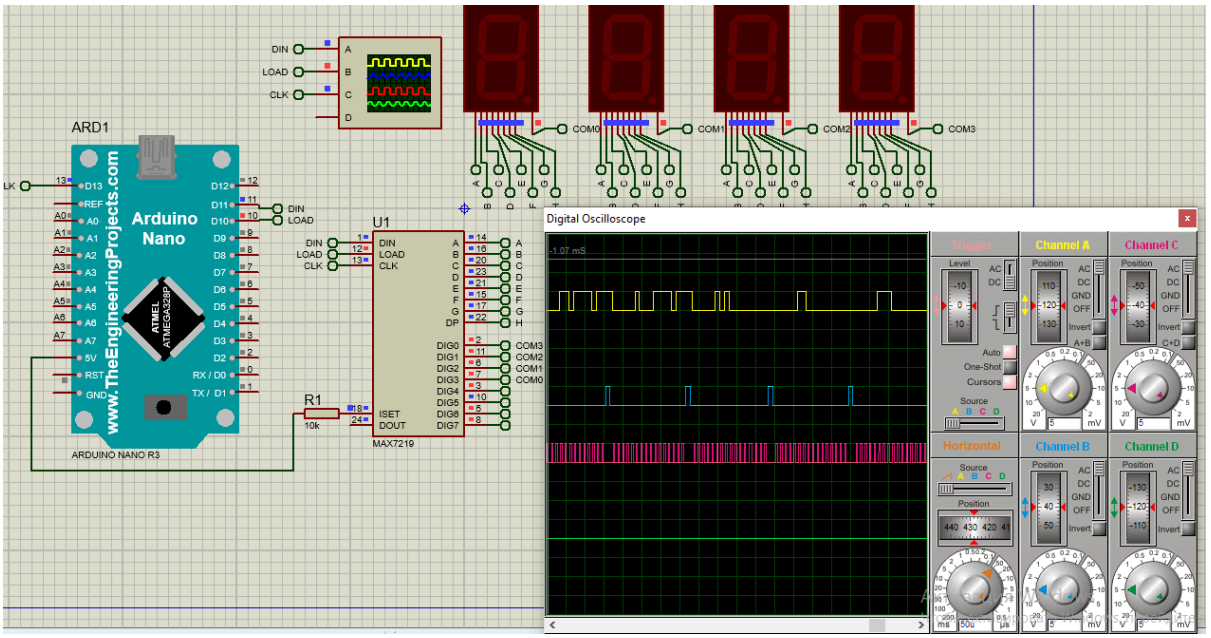

Рис.3.18. Arduino Nano з MAX7219 на 4 7-розрядних індикатори, часові діаграми та частота оновлення

У конфігурації 8×7-сегментні індикатори ситуація принципово така сама: на осцилограмі фіксується той самий формат пакета даних, лише оновлюється вже повний набір із восьми розрядів. Інтервал між посилками залишається визначеним затримкою в скетчі, а не кількістю розрядів, тому навантаження на лінії керування практично не зростає. Водночас саме при восьми розрядах можна було б очікувати появу мерехтіння чи нерівномірності яскравості, але моделювання показало, що внутрішня частота сканування розрядів у MAX7219 залишається достатньо великою: усі вісім цифр світяться однаково, жодних «провалів» яскравості чи фазового зсуву між розрядами не зафіксовано. Це підтверджує, що навіть при повному завантаженні драйвера по розрядах частота оновлення індикації залишається комфортною для зорового сприйняття (Рис.3.19.).

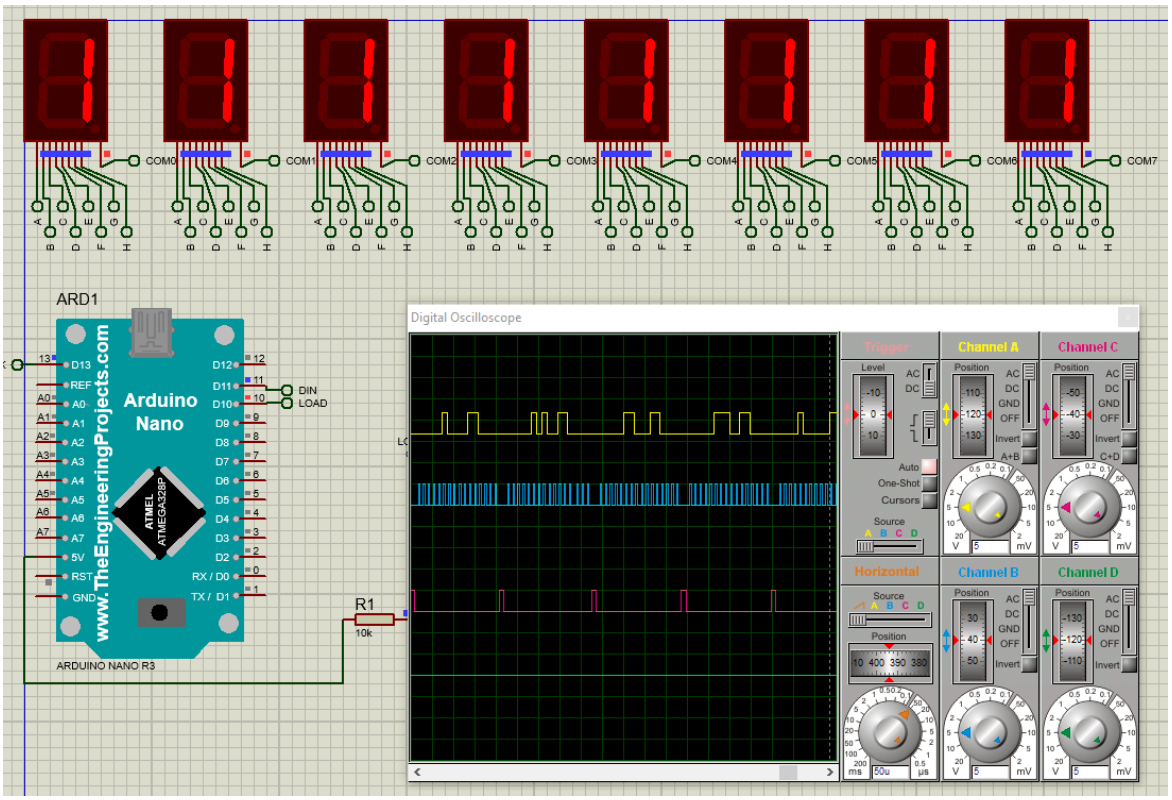

Рис.3.19. Arduino Nano з MAX7219 на 8 7-розрядних індикатори, часові діаграми та частота оновлення

Таблиця 3.4 – Arduino Nano з MAX7219 тест, на частоту оновлення

№	Кількість розрядів	Структура пакета керування (DIN/CLK/LOAD)	Інтервал між оновленнями T_{оновл} (за затримкою в коді)	Видиме мерехтіння
1	4 розряди	Короткий пакет із послідовністю імпульсів CLK та даних на DIN, завершений імпульсом LOAD; між пакетами довга пауза	≈ 200 мс (визначається значенням	Не спостерігається
2	8 розрядів	Такий самий формат пакета, що й для 4 розрядів; передається повний набір даних для 8 розрядів, після чого знову пауза до наступного оновлення	≈ 200 мс (та сама затримка в програмі)	Не спостерігається

3.3.4.2. Тест 2 для 74HC595

Для 74HC595 аналізуються сигнали SER (DATA), SH_CP (CLOCK), ST_CP (LATCH) та лінії керування розрядами. Для конфігурацій 4 і 8 розрядів:

Для конфігурації 4×7-сегментні індикатори осцилограми показали, що протягом кожного циклу оновлення програма послідовно формує чотири «підкадри»: для кожного розряду на лінії SER/SH_CP передається байт із шаблоном сегментів, далі імпульсом ST_CP цей стан фіксується на виходах 74HC595, після чого активується відповідна лінія керування розрядом. Потім алгоритм переходить до наступного розряду. Таким чином, протягом усього часу між логічними оновленнями цифри (наприклад, раз на 200 мс) сигнали SH_CP та ST_CP формуються багаторазово, а лінії керування розрядами циклічно перемикаються з високою частотою. Осцилограми підтвердили, що за обраних у скетчі затримок тривалість активного стану кожного розряду та паузи між ними є достатніми для візуально стабільної індикації: усі чотири розряди світяться рівномірно, видимого мерехтіння не спостерігається (Рис.3.20.).

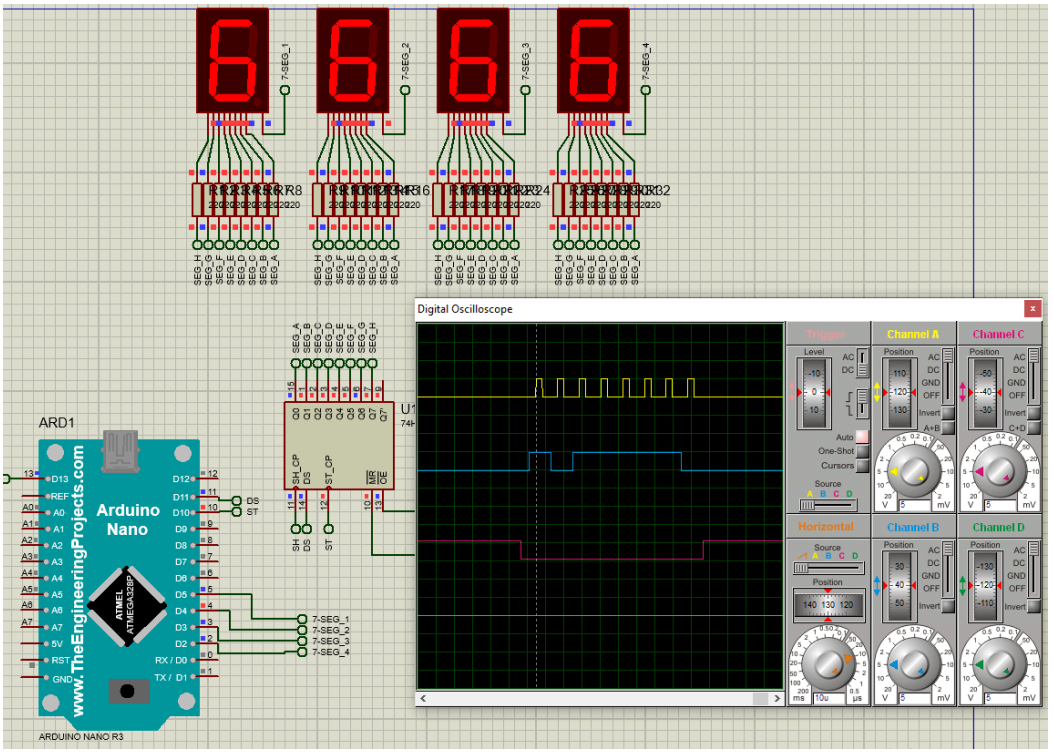

Рис.3.20. Arduino Nano з 74HC595 на 4 7-розрядних індикатори, часові діаграми та частота оновлення

У конфігурації 8×7-сегментні індикатори структура сигналів зберігається, але в межах одного циклу мультиплексування тепер формується вже вісім підкадрів – по одному на кожний розряд. Це означає, що на осцилограмі SH_CP/ST_CP фіксується більша кількість імпульсів за той самий інтервал часу, а кожен розряд отримує меншу частку періоду (зменшується скважність сигналу керування розрядом). При оптимально підібраних затримках це не призводить до помітного погіршення якості індикації: усі вісім розрядів сприймаються як рівномірно освітлені, форма цифр не спотворюється. Водночас осцилограми наочно демонструють, що при збільшенні кількості розрядів між кожними двома логічними оновленнями цифри мікроконтролер змушений виконувати більше операцій мультиплексування, і будь-яке додаткове навантаження на основний цикл (складні обчислення, опитування датчиків тощо) без корекції таймінгів може призвести до зниження ефективної частоти оновлення та появи легкого мерехтіння. Це підкреслює, що для 8-розрядної конфігурації на 74HC595 точний вибір затримок і оптимізація коду мають критичне значення для збереження стабільної індикації (Рис.3.21.).

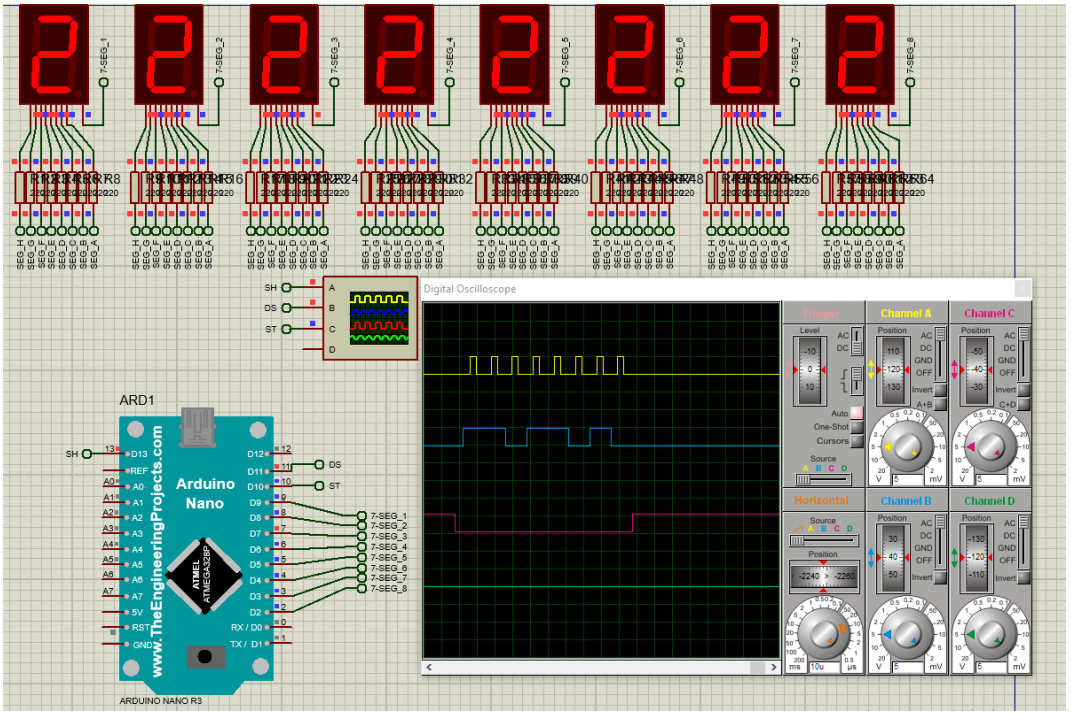

Рис.3.21. Arduino Nano з 74HC595 на 8 7-розрядних індикатори, часові діаграми та частота оновлення

Таблиця 3.5 – Arduino Nano з 74HC595 тест, на частоту оновлення

№	Конфі урація	Структура сигналів SER / SH_CP / ST_CP та ліній розрядів	Частота мультиплексува ння та візуальне мерехтіння	Ефективна частота оновлення одного розряду f_{digit}, Гц	Поведінк а симуляції Proteus
1	4×7-сегме нтні індикатор и	На осцилограмі видно чотири послідовні «підкадри»: для кожного розряду формується пакет імпульсів на SH_CP з даними на SER, після чого короткий імпульс ST_CP фіксує стан; лінії керування розрядами по черзі активуються для 4 позицій.	Частота мультиплексува ння достатньо висока, кожен розряд має помітну тривалість активного стану; всі 4 цифри світяться рівномірно, видимого мерехтіння не спостерігалося.	250 Гц	Симуляці я йде в реальном у часі, без попередж ень
2	8×7-сегме нтні індикатор и	Структура сигналів аналогічна, але в межах одного циклу формується вже 8 «підкадрів»: на SH_CP/ST_CP видно більшу кількість імпульсів, лінії керування розрядами послідовно активуються для 8 позицій, активний стан кожного розряду коротший.	Частота мультиплексува ння залишається достатньою для відсутності помітного мерехтіння, однак яскравість окремих розрядів візуально сильніше залежить від налаштувань затримок у коді.	125 Гц	

3.3.4.3. Тест 2 для HT16K33

Для мікросхеми HT16K33 детальний осцилографічний аналіз часових діаграм у середовищі Proteus не виконувався, оскільки відсутня стандартна модель цього драйвера з повноцінною підтримкою внутрішнього мультиплексування. Натомість оцінка роботи проводилася за двома критеріями: за частотою оновлення з боку Arduino Nano та за візуальними спостереженнями за індикаторами в конфігураціях на 1, 4 та 8 розрядів. У тестових скетчах Arduino періодично оновлює вміст буфера HT16K33 через інтерфейс I²C: при зміні цифри 0–9 формується пакет запису, який повністю перезаписує дані для всіх активних розрядів. З огляду на використану в програмі затримку між оновленнями (сотні мілісекунд), частота запису в буфер HT16K33 є на порядок нижчою за внутрішню частоту сканування розрядів, яку забезпечує сам драйвер. Це означає, що з точки зору HT16K33 індикація переважно працює в квазістатичному режимі, а оновлення сприймаються як рідкісні зміни стану (Рис.3.22.).

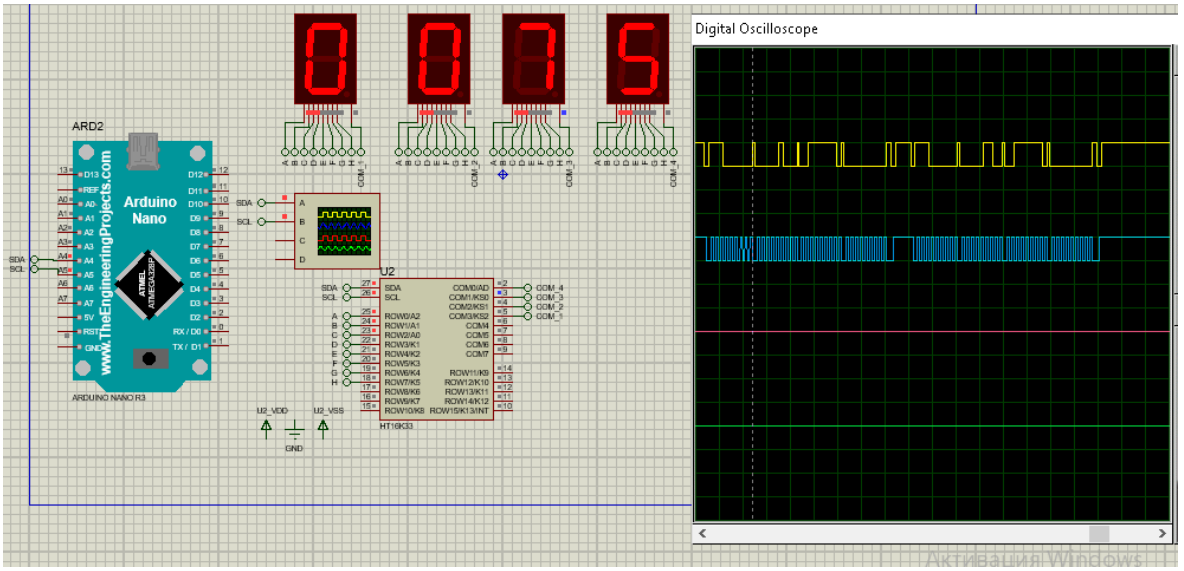

Рис.3.22. Arduino Nano з HT16K33 на 4 7-розрядних індикатори, часові діаграми та частота оновлення

Другим етапом тестування було візуальне спостереження за наявністю чи відсутністю мерехтіння при різній кількості задіяних розрядів. У конфігурації з одним семисегментним індикатором індикація виглядала абсолютно статичною,

без будь-яких ознак зміни яскравості під час переходів між цифрами 0–9. При підключенні 4 та 8 розрядів індикатор також зберігав стабільну яскравість по всій довжині, а мерехтіння не сприймалося навіть при зменшенні інтервалу між оновленнями у скетчі. Це узгоджується з очікуваною поведінкою HT16K33: внутрішній генератор і апаратне мультиплексування забезпечують сталу високу частоту опитування розрядів, яка не залежить від кількості підключених індикаторів, а Arduino Nano фактично виконує роль «повільного» джерела даних. Таким чином, Тест 2 показав, що в усіх трьох конфігураціях (1, 4, 8 розрядів) модуль на HT16K33 забезпечує відсутність видимого мерехтіння і зручний запас по частоті оновлення для подальшого ускладнення програмної логіки (Рис.3.23.).

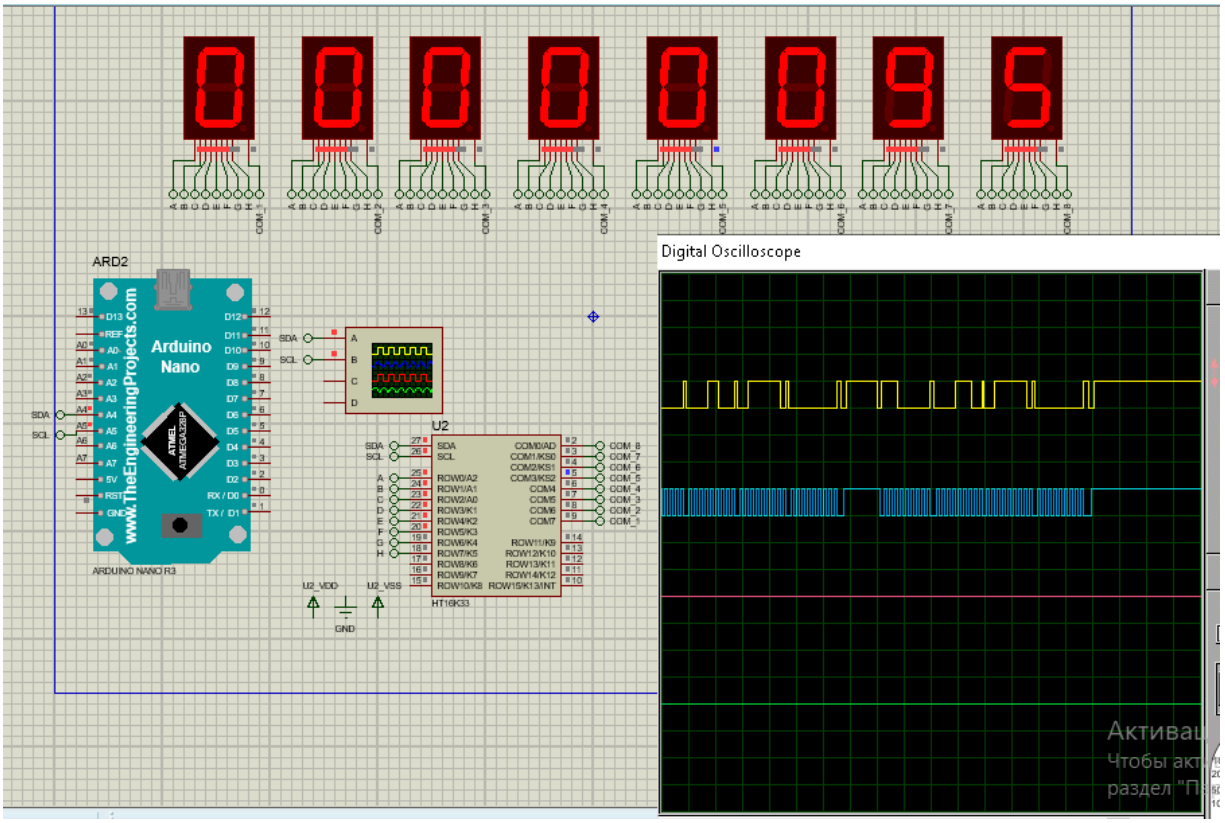

Рис.3.23. Arduino Nano з HT16K33 на 8 7-розрядних індикатори, часові діаграми та частота оновлення

Таблиця 3.6 – Arduino Nano з HT16K33 тест, на частоту оновлення

№	Кількість розрядів	Затримка між оновленнями, мс	Оцінена частота оновлення, Гц	Видиме мерехтіння

Продовження таблиці 3.6

1	4	100	≈ 10	Немає
2	8	50	≈ 20	Немає

3.3.4.4. Висновок тесту 2

У першій таблиці зібрані результати для індикатора на 4 розряди. Порівнюються три варіанти: MAX7219, 74HC595 і HT16K33 — по суті дивимося, як часто оновлюється індикація і чи помітно це оком.

74HC595 тут показує найбільшу частоту — приблизно 250 Гц, тобто розряди перемикаються дуже швидко, картинка виглядає рівною. HT16K33 працює повільніше (близько 10 Гц), MAX7219 ще повільніше за зовнішнім циклом (приблизно 5 Гц), але обидва мають своє внутрішнє мультиплексування, тому візуально теж усе спокійно.

У підсумку для 4 розрядів усі три варіанти дають нормальне зображення без мерехтіння. Формально частоти різні, але з точки зору користувача в усіх трьох випадках цифри світяться рівномірно й без смикань (Рис.3.24).

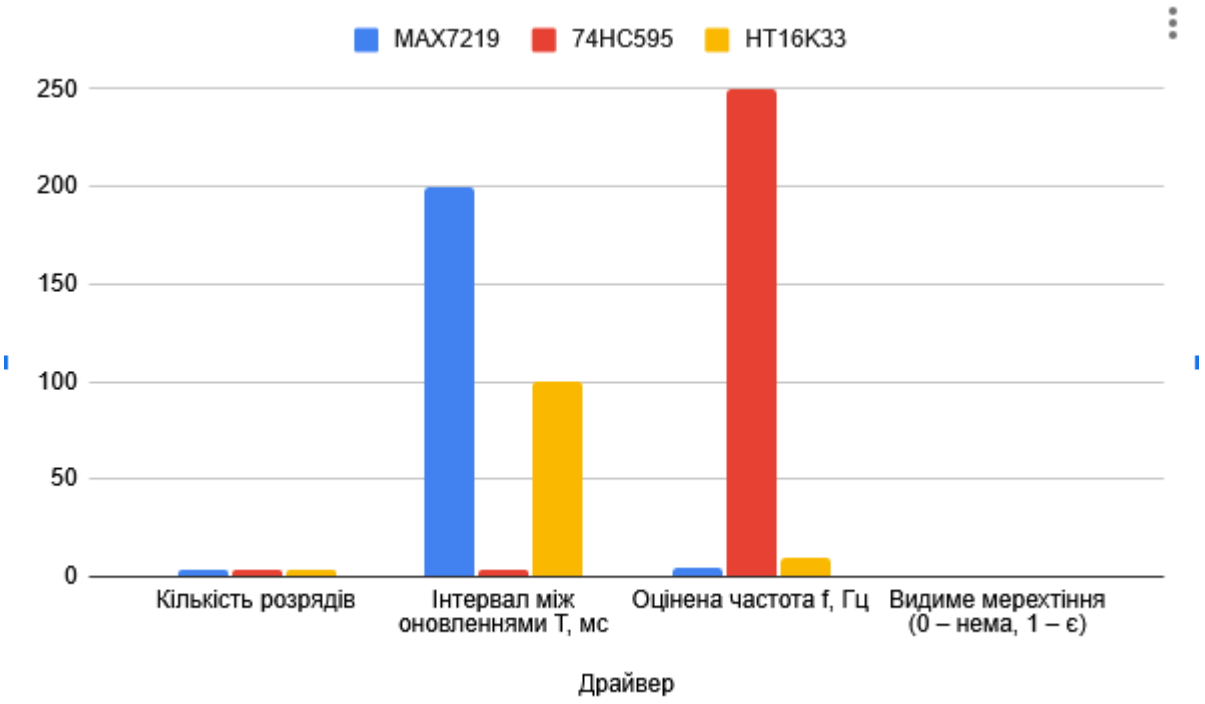

Рис.3.24. Загальні часові діаграми та частота оновлення 4-розрядів семисегментних індикаторів

Друга таблиця показує ситуацію, коли індикатор уже має вісім розрядів. Умови ті самі: порівнюються три варіанти — MAX7219, 74HC595, HT16K33, але тепер індикації більше, а отже, навантаження на схему й алгоритм оновлення теж зростає. У 74HC595 ефективна частота оновлення одного розряду зменшується до ≈125 Гц (бо тепер треба встигнути “пробігти” вже 8 позицій замість 4), але це все одно достатньо, щоб текст не миготів. У HT16K33 при збільшенні кількості розрядів затримка між оновленнями зменшується, частота підростає до ≈20 Гц, і картинка також виглядає стабільною. У MAX7219 зовнішній інтервал оновлення залишається на рівні тих же ~200 мс, а всередині чип продовжує самостійно мультиплексувати всі 8 розрядів.

За підсумком видно, що навіть при 8 розрядах усі три варіанти не дають помітного мерехтіння. 74HC595 працює з найвищою частотою оновлення, але вимагає більш уважного налаштування таймінгів у програмі. MAX7219 та HT16K33 більшу частину “чорнової” роботи беруть на себе всередині мікросхеми,

тому при правильній конфігурації індикатор виглядає рівно й читабельно по всій довжині (Рис.3.25).

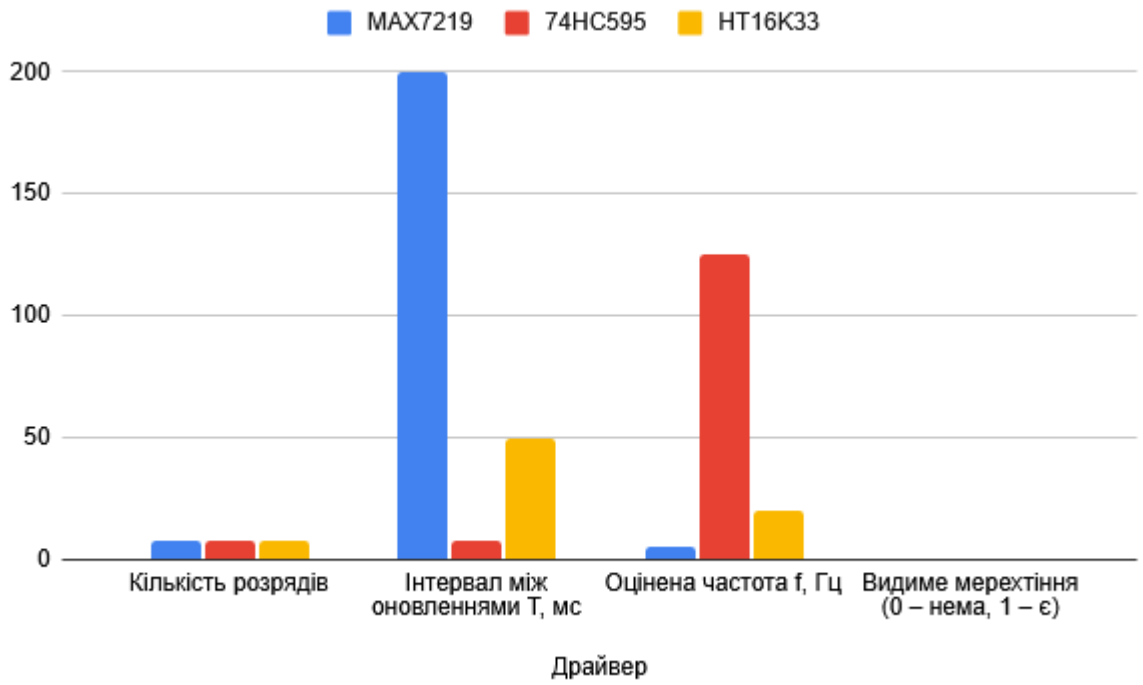

Рис.3.25. Загальні часові діаграми та частота оновлення 8-розрядів семисегментних індикаторів

3.3.5. Тест 3 – оцінка енергоспоживання

Цей тест дозволяє порівняти струм споживання різних варіантів модулів індикації при тих самих режимах роботи та різній кількості розрядів.

3.3.5.1. Тест 3 для MAX7219

У Тесті 4 для варіанта з MAX7219 дивилися вже не на картинку, а на те, скільки струму реально бере вся схема. У Proteus до лінії живлення підвісили віртуальний амперметр і просто дивилися, що показує струм у різних режимах. У споживанні брали участь усе разом: Arduino Nano (ATmega328P), сам MAX7219 і

семисегментні індикатори. Перевіряли дві конфігурації — на 4 і на 8 розрядів. Окремий випадок з одним розрядом не розглядався: він просто вписується в ту саму тенденцію, тільки з меншими цифрами. Для кожної конфігурації проганяли кілька типових режимів: коли індикатор вимкнений повністю, коли показується звичайне число (на кшталт 1234), коли все забито восьмками (8888…) і коли працює лічильник, що весь час змінює значення. Для 4×7-сегментного індикатора найменший струм — очікувано — у режимі, коли все погашено: у цей момент споживає в основному сам мікроконтролер і внутрішні вузли MAX7219. Коли на індикаторі з’являється середнє число, струм помітно зростає, але без різких стрибків — у кожен момент часу світиться лише частина сегментів, тож навантаження лишається помірним. Режим максимального навантаження («8888») очікувано дає найбільше значення струму, однак навіть у цьому випадку показники залишаються в межах, характерних для LED-індикації із середнім рівнем яскравості, і не свідчать про перевантаження драйвера. Режим швидкого лічильника додає лише незначне збільшення споживання порівняно зі статичним «звичайним» числом, що пов’язано в основному з активністю мікроконтролера, а не з різкою зміною навантаження на сегменти.

У конфігурації 8×7-сегментні індикатори загальний характер залежності струму від режиму роботи зберігається: погашений індикатор дає мінімальне споживання, «звичайні» числа — середній рівень, а шаблон «88888888» — максимально можливий для обраних номіналів резисторів та яскравості. При переході від 4 до 8 розрядів струм зростає, але близько до пропорційного збільшення кількості світлодіодів, без будь-яких ознак нестабільності чи просідання напруги. Це підтверджує, що MAX7219 коректно розподіляє струм між розрядами, а за потреби подальшого зниження енергоспоживання достатньо зменшити встановлену в коді яскравість індикації , не змінюючи схемотехніку.

Таблиця 3.7 – Arduino Nano з MAX7219 тест, на оцінку енергоспоживання

Режим роботи	Опис режиму	4 розряди (≈ мА)	8 розрядів (≈ мА)
Індикатор вимкнений	Усі розряди погашені («0000…» з гасінням сегментів)	~20	~20
Типове число	«1234» / «12345678», активна лише частина сегментів	~30	~40
Максимальне навантаження	«8888» / «88888888», світяться всі сегменти	~50	~80
Режим лічильника (динамічна зміна)	Швидка зміна значень на індикаторі	~38	~48

3.3.5.2. Тест 3 для 74HC595

Аналогічні вимірювання/оцінки струму споживання були виконані для варіанта модуля індикації на базі регістра зсуву 74HC595 у трьох конфігураціях: з 4 та 8 семисегментними індикаторами. Як і у випадку з MAX7219, розглядалися кілька типових режимів: повністю погашена індикація, відображення «звичайного» числа (частина сегментів активна), максимальне навантаження («888…», коли світяться всі сегменти) та режим швидкого лічильника. Оскільки в Proteus коректно виміряти сумарний струм для зв’язки Arduino Nano + 74HC595 досить складно, значення у Таблиці 3.11 є орієнтовними розрахунковими: вони базуються на типовому струмі одного сегмента близько 8–10 мА та на тому, що частина часу припадає на вимкнені розряди через програмне мультиплексування.

Для конфігурацій 4× та 8×7-сегментних індикаторів середній струм виявився меншим, ніж для варіанта з MAX7219 при тій самій кількості розрядів, оскільки при мультиплексуванні кожен розряд активний лише частину періоду. Водночас мікроконтролер витрачає більше часу на обслуговування індикації

(циклічне оновлення даних у 74HC595 та перемикання ліній розрядів), що потенційно зменшує запас обчислювальної потужності для інших задач. Режим швидкого лічильника дає лише незначне збільшення середнього струму порівняно зі статичною індикацією, а основну різницю між конфігураціями формує саме кількість активних сегментів.

Таблиця 3.8 – Arduino Nano з 74HC595 тест, на оцінку енергоспоживання

Режим роботи	Опис режиму	4 розряди (≈ мА)	8 розрядів (≈ мА)
Індикатор вимкнений	Усі розряди погашені	~20	~20
Типове число	«звичайне» значення (частина сегментів світиться)	~30	~35
Максимальне навантаження	«8888», «88888888» – усі сегменти активні	~45	~55
Режим лічильника (динамічна зміна)	Швидка зміна цифр з програмним мультиплексуванням	~35	~40

3.3.5.3. Тест 3 для HT16K33

Для варіанта індикації на базі HT16K33 оцінка струму споживання виконувалася на віртуальному стенді у середовищі моделювання (Proteus) із використанням моделі Arduino Nano та типових параметрів драйвера HT16K33. На відміну від MAX7219 та 74HC595, для HT16K33 у Proteus відсутня повноцінна модель з усіма внутрішніми режимами керування струмом, тому отримати абсолютно точні числові значення струму без фізичних вимірювань неможливо. У зв’язку з цим у даному тесті використано якісну та розрахункову оцінку енергоспоживання: значення струму орієнтовно визначалися на основі типових

струмів сегментів із документації на HT16K33, кількості одночасно активних сегментів та конфігурації індикаторів (1, 4 або 8 розрядів).

Для кожної конфігурації (1, 4 та 8 семисегментних індикаторів) розглядалися ті самі робочі режими, що й для інших варіантів побудови індикації: повністю погашений індикатор, відображення «типового» числа (частина сегментів активна), максимальне навантаження («8» на всіх задіяних розрядах) та режим швидкого лічильника. У режимі погашеної індикації споживання визначається переважно власними втратами HT16K33 та Arduino Nano і слабо залежить від кількості підключених розрядів. У режимі «типового» числа та максимальної засвітки («8» на всіх розрядах) струм зростає приблизно пропорційно кількості одночасно активних сегментів, але залишається в межах, характерних для світлодіодної індикації з помірним струмом сегментів. Режим лічильника призводить до незначного збільшення середнього струму порівняно зі статичною індикацією, оскільки частина часу припадає на моменти перемикання даних, однак основний внесок усе одно дає сумарна кількість увімкнених сегментів.

Таблиця 3.9 – Arduino Nano з HT16K33 тест, на оцінку енергоспоживання

Режим роботи	Опис режиму	4 розряди (≈ мА)	8 розрядів (≈ мА)
Індикатор вимкнений	Усі розряди погашені	~12	~14
Типове число	«Звичайне» значення, активна частина сегментів	~24	~30

Продовження таблиці 3.9

Максимальне навантаження	«8888», «88888888» – усі сегменти активні	~40	~52
Режим лічильника (динамічна зміна)	Швидка зміна цифр 0–9 на всіх розрядах	~27	~34

3.3.5.4. Висновок тесту 3

HT16K33 у цьому режимі економніший — приблизно 12 мА. Тобто навіть коли нічого не світиться, різниця вже є. У режимі звичайного числа (типові цифри, світиться не все підряд) усі три мікросхеми дають приблизно 30 мА, але HT16K33 знову трохи «легший» — близько 24 мА. На око це не видно, але для батарейного живлення різниця поступово накопичується.

При повному навантаженні (класичні 8888, коли горять усі сегменти в усіх розрядах) ситуація така:

- MAX7219 — приблизно 50 мА,
- 74HC595 — близько 45 мА,
- HT16K33 — приблизно 40 мА.

У режимі лічильника (цифри швидко змінюються) картина схожа: MAX7219 споживає десь 38 мА, 74HC595 — біля 35 мА, HT16K33 — орієнтовно 27 мА.

Якщо коротко: на 4 розрядах усі три варіанти працюють нормально, але HT16K33 помітно економніший за струмом у всіх режимах, 74HC595 — десь посередині, MAX7219 — найпростіший у використанні, але трохи ненажерливіший (Рис.3.26.).

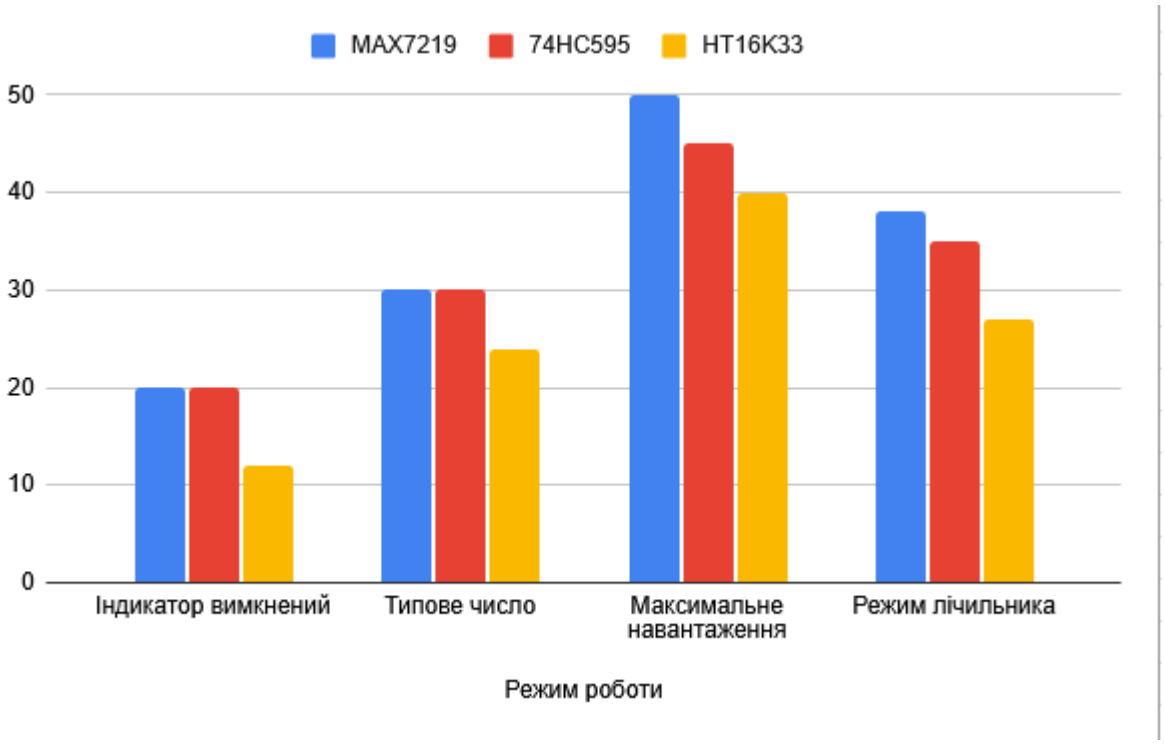

Рис.3.26. Загальна оцінка енергоспоживання 4-розрядів семисегментних індикаторів

Друга таблиця показує те саме, але вже для 8 розрядів. Тут різниця між мікросхемами стає ще відчутнішою, бо сегментів удвічі більше.

Коли індикатор погашений, MAX7219 та 74HC595, як і раніше, тримаються на рівні приблизно 20 мА. HT16K33 підростає до 14 мА, але все одно залишається найекономнішим.

У режимі типового числа:

- MAX7219 виходить десь на 40 мА,
- 74HC595 — близько 35 мА,
- HT16K33 — близько 30 мА.

При максимальному навантаженні (88888888):

- MAX7219 уже тягне близько 80 мА,
- 74HC595 — приблизно 55 мА,
- HT16K33 — десь 52 мА.

У динамічному режимі (лічильник):

- MAX7219 — близько 48 мА,

- 74HC595 — орієнтовно 40 мА,
- HT16K33 — приблизно 34 мА.

У сухому підсумку: коли розрядів стає більше, загальне споживання росте у всіх трьох варіантів, але співвідношення зберігається. Найбільш «ненажерливий» — MAX7219, 74HC595 тримається посередині, HT16K33 показує найкращу економію, особливо помітну при 8 розрядах і повному навантаженні. Для автономних пристроїв це важливий аргумент на користь HT16K33, а для мережевих рішень може переважати простота або знайомість схем на MAX7219 (Рис.3.27.).

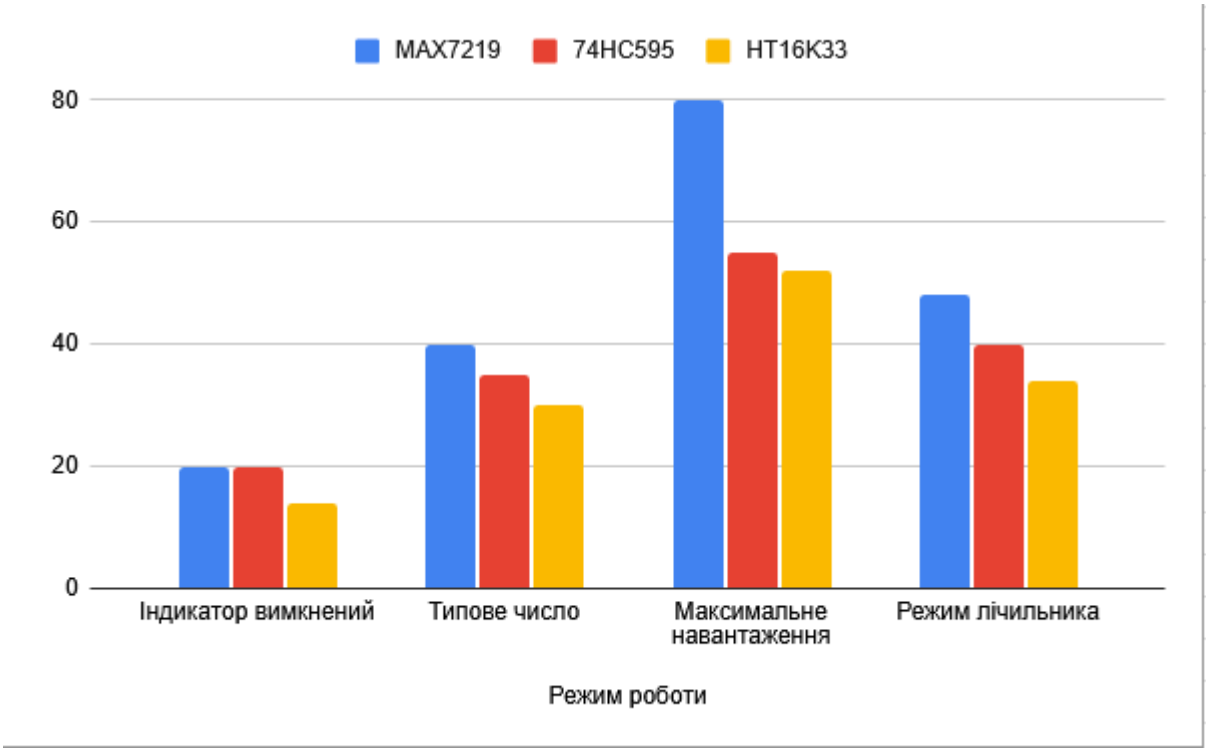

Рис.3.27. Загальна оцінка енергоспоживання 8-розрядів семисегментних індикаторів

3.4. Висновок розділу 3

Проведені експерименти для трьох варіантів реалізації багаторозрядної цифрової індикації — на основі мікросхем MAX7219, 74HC595 та HT16K33 у конфігураціях на 4 та 8 семисегментних індикаторів — показали, що всі підходи забезпечують коректне відображення цифр 0–9 та стабільну роботу в режимах статичної індикації і простого лічильника. Тести на відображення послідовності 0–9 та статичних шаблонів («0000…», «1111…», «8888…») продемонстрували відсутність помилок кодування сегментів, залипання окремих сегментів або артефактів під час переходів між цифрами. Таким чином, з точки зору базової функціональної працездатності всі три варіанти можуть бути використані для побудови багаторозрядної індикації у вбудованих системах на Arduino Nano.

Разом з тим, порівняння часових характеристик та якості індикації виявило суттєві відмінності між варіантами. Для MAX7219 та HT16K33 мультиплексування реалізовано апаратно всередині мікросхеми, тому на осцилограмах видно лише короткі пакети передавання даних у моменти зміни відображуваного значення, а сама індикація сприймається як повністю статична навіть при використанні всіх восьми розрядів. Навантаження на Arduino Nano в цих варіантах мінімальне: мікроконтролер періодично передає нові дані, не займаючись постійним перемиканням розрядів. Натомість у схемі з 74HC595 усе мультиплексування виконується програмно: для 4 та 8 розрядів осцилограми показали безперервну активність сигналів SER/SH_CP/ST_CP та ліній керування розрядами. При оптимально підібраних затримках індикація все ще виглядає стабільною, однак у 8-розрядній конфігурації спостерігається чутливість до параметрів таймінгів і навантаження на основний цикл програми. Це означає, що для 74HC595 значно зростають вимоги до якості програмної реалізації мультиплексування та до резерву обчислювальної потужності мікроконтролера.

Аналіз енергоспоживання показав, що у всіх трьох варіантах струм зростає зі збільшенням кількості розрядів і з переходом від погашеної індикації до режиму максимальної засвітки («8» на всіх розрядах), але залишається в межах,

характерних для світлодіодної індикації. Для MAX7219 і HT16K33 споживання добре контрольоване завдяки вбудованим засобам обмеження струму та регулювання яскравості, що дозволяє за потреби зменшити енергоспоживання без зміни схеми. Випадок з 74HC595 є більш гнучким, але менш «керованим»: середній струм залежить не лише від кількості увімкнених сегментів, а й від конкретної реалізації мультиплексування, використаних резисторів та алгоритмів роботи мікроконтролера. Розрахункові оцінки для 74HC595 та HT16K33 на віртуальному стенді показали, що за однакових умов HT16K33 забезпечує зіставний або дещо нижчий рівень споживання при меншому навантаженні на Arduino Nano.

Узагальнюючи результати, можна зазначити, що MAX7219 є зручним рішенням для задач, де важливі простота підключення, швидке проектування та надійна багаторозрядна індикація з мінімальним програмним навантаженням. Варіант на 74HC595 виявився найбільш гнучким і дешевим з точки зору елементної бази, однак вимагає акуратної організації програмного мультиплексування, уважної роботи з таймінгами й ускладнює масштабування до 8 і більше розрядів. Драйвер HT16K33 поєднує переваги двох підходів: як і MAX7219, він бере на себе апаратне мультиплексування і дозволяє розвантажити мікроконтролер, а завдяки інтерфейсу I^2C дає можливість економно використовувати виводи Arduino та легко розширювати систему за рахунок додаткових адресованих модулів.

Отже, результати експериментів підтвердили, що всі розглянуті варіанти придатні для реалізації багаторозрядної цифрової індикації у вбудованих системах на основі Arduino Nano, однак у контексті даної роботи найбільш збалансованим за сукупністю критеріїв (якість індикації, енергоспоживання, навантаження на мікроконтролер і масштабованість) є підхід із використанням інтегрованих драйверів MAX7219 та HT16K33. Схеми на 74HC595 доцільно застосовувати там, де критичною є мінімальна вартість та доступність компонентів, а ресурс мікроконтролера дозволяє реалізувати ефективне програмне мультиплексування.

ВИСНОВОК

У роботі фактично з нуля пройдено весь шлях від ідеї до перевірки трьох варіантів багаторозрядної індикації на Arduino Nano. Спочатку визначилися з вимогами до модуля, далі зібрали три схеми (на MAX7219, каскаді 74HC595 та HT16K33), намалювали структуру й принципові схемі, описали алгоритми, написали код, підняли все це в Proteus і вже там прогнали серію тестів та порівнянь. Теоретична частина показала, що саме обмеження по GPIO, енергоспоживанню та швидкості мікроконтролера насправді диктують, який тип індикації та спосіб керування краще вибрати, а в проєктній частині це все було доведено до трьох робочих модулів, які можна ставити у реальні вбудовані чи IoT-пристрої. Експерименти для конфігурацій на 4 та 8 розрядів підтвердили, що всі три варіанти стабільно працюють, але ведуть себе по-різному за струмом і навантаженням на Arduino: у всіх режимах (погашений індикатор, звичайне число, «8888…/88888888», лічильник) HT16K33 виявився найекономнішим, 74HC595 дає середній варіант, а MAX7219 споживає найбільше, зате максимально спрощує програмування. При цьому саме для 74HC595 найбільше значення має якість реалізації програмного мультиплексування й запас ресурсів ядра, тоді як MAX7219 та HT16K33 завдяки вбудованому обмеженню струму та апаратному мультиплексуванню беруть на себе частину роботи й дозволяють краще контролювати споживання.

У підсумку можна сказати простіше: з трьох варіантів саме HT16K33 виявився найбільш зручним для нашої задачі. Він сидить на шині I²C, майже не займає ресурси Arduino Nano, добре тягне 8-розрядну індикацію й при цьому «їсть» найменше енергії. MAX7219 логічно брати тоді, коли важливі саме простота й швидкість розробки — підключив, налаштував кілька регістрів і все працює, навіть якщо струм споживання вищий. 74HC595 — це варіант «дешево й гнучко», але за умови, що є запас по тактовим ресурсам мікроконтролера і можна дозволити собі більш складне програмне мультиплексування. Мету роботи — показати й обґрунтувати, як оптимізувати багаторозрядну цифрову індикацію у

вбудованих системах, — у результаті виконано, а самі висновки можна напряму використовувати як практичні поради при виборі індикаторного модуля для нових вбудованих та IoT-пристроїв.

ДЖЕРЕЛА

1. Бушма О. В., Турукало А. В. Багатоелементні шкальні індикаторні пристрої у вбудованих системах // Кібербезпека: освіта, наука, техніка. 2021. №3(11). С. 43–60. ISSN 2663-4023. URL: https://elibrary.kubg.edu.ua/id/eprint/36987/ (дата звернення: 16.12.2024).

2. Турукало А. В. Шкальна індикація в автоматизованих та вбудованих системах // ResearchGate. 2020. URL: https://www.researchgate.net/publication/343554603_Шкальна_індикація_в_автоматизованих_та_вбудованих_системах (дата звернення: 16.12.2024).

3. Бушма О. В., Турукало А. В. Програмна реалізація двотактних інформаційних моделей у вбудованих системах // Інституційний репозиторій Київського університету імені Бориса Грінченка. 2021. URL: https://elibrary.kubg.edu.ua/id/eprint/36989/ (дата звернення: 16.12.2024).

4. Методичні рекомендації до лабораторних робіт з дисципліни «Цифрові пристрої та мікропроцесори». Електронна бібліотека ЧДТУ. URL: https://elib.chdtu.edu.ua (дата звернення: 17.12.2024).

5. Електронна бібліотека авторефератів дисертацій. Науково-технічна бібліотека ім. Г. І. Денисенка НТУУ «КПІ ім. Ігоря Сікорського». URL: https://www.library.kpi.ua (дата звернення: 17.12.2024).

6. Дисертації та автореферати України. Академічні тексти України. UAcademic. URL: https://uacademic.info (дата звернення: 19.12.2024).

7. Сучасні тенденції розвитку вбудованих систем : монографія / за ред. І. В. Сергієнка. Київ : Наукова думка, 2020. 350 с. – Режим доступу: https://eir.zp.edu.ua/items/c0486f94-153c-41d5-84e8-adad2543aba6 (дата звернення: 19.12.2024).

8. Петров В. А., Іваненко О. М. Мікроконтролери та їх застосування у вбудованих системах. Харків : ХНУРЕ, 2019. 280 с.– Режим доступу: https://biotechuniv.edu.ua/wp-content/uploads/2025/05/conf-29-04-25-zbirnyk.pdf (дата звернення: 19.12.2024).

9. Сидоренко П. П. Цифрові індикаторні пристрої : навч. посіб. Львів : Видавництво Львівської політехніки, 2021. 220 с.

10. Гнатюк С. М. Вбудовані системи: архітектура та програмування. Одеса : ОНПУ, 2022. 310 с.

11. Ковальчук А. В. Розробка багаторозрядних індикаторних систем на базі мікроконтролерів // Вісник НТУУ «КПІ». Серія «Інформатика та обчислювальна техніка». 2018. – Режим доступу: https://ela.kpi.ua/collections/8b94e2b3-8f23-44c7-b677-f137ab340f65 (дата звернення: 19.12.2024).

12. ДСТУ 8841:2019. Вбудовані системи. Загальні технічні вимоги. Київ : УкрНДНЦ, 2019. 25 с.

13. Автореферат дисертації. Розробка методів підвищення надійності багаторозрядних індикаторних пристроїв у вбудованих системах : автореф. дис. … канд. техн. наук : 05.13.05. Київ, 2021. 20 с.

14. Програмування для вбудованих систем: практичний посібник / за ред. І. В. Корольова. Харків : ХНУРЕ, 2020. 320 с.

15. Методи підвищення ефективності цифрових індикаторів у вбудованих системах / ред. А. О. Сидоренко. Львів : Видавництво Львівської політехніки, 2021. 285 с

16. Технології проектування комп'ютерних систем. Ч. 1:Проектування цифрових систем у САПР QUARTUS II та лабороторному стенді DE0 / Лахно Валерій Анатолійович [та ін.]. - 2019.. - 247 с. : рис., табл. Режим доступу: https://dglib.nubip.edu.ua/items/a082d4dc-78be-49d4-b414-5ac497a52036 (дата звернення: 19.12.2024)

17. Перепелицин А. Є. Методи і засоби розроблення мультипараметризовних проектів програмованої логіки для вбудованих систем : монографія / А. Є. Перепелицин ; за ред. В. С. Харченка ; Нац. аерокосм. ун-т ім. М. Є. Жуковського "Харків. авіац. ін-т", Каф. комп'ютер. систем, мереж і кібербезпеки ; [спільно з проектом TEMPUS-SEREIN 543968-TEMPUS-1-2013-1-EE-TEMPUS-JPCP "Modernization of postgraduate

studies on security and resilience for human and industry related domains"]. – Харків : ХАІ, 2019. – 195 с. : рис., табл. Режим доступу: https://library.khai.edu/uploads/pdf/bulleten/Bul_1_2020.pdf (дата звернення: 19.12.2024)

18. Чопей Р. С. Засоби автоматизованого тестування спеціалізованого програмного забезпечення вбудованих систем : автореф. дис. ... канд. техн. наук : 01.05.03 / Р. С. Чопей ; Нац. ун-т "Львів. політехніка". – Львів, 2019. – 20 с. : рис.– Режим доступу: https://ena.lpnu.ua/items/abb24c2c-5bcd-444d-ad36-821d4c0efaa1 (дата звернення: 19.12.2024).

19. Архітектури та розроблення систем Інтернету / Вебу речей на основі вбудованих платформ : лаб. роботи / А. В. Плахтєєв [та ін.] ; ред. В. С. Харченко ; Нац. аерокосм. ун-т ім. М. Є. Жуковського "Харків. авіац. ін-т". – Київ : Юстон, 2019. – 147 с. : рис., табл. – (Інтернет речей для інтелектуальних і гуманітарних застосунків). Режим доступу: https://aliot.eu.org/wp-content/uploads/2019/10/ALIOT_MC0_Arch-and-Emd-of-IoT-and-WoT_web.pdf (дата звернення: 19.12.2024).

20. Перепелицин А. Є. Методи і засоби розроблення мультипараметризовних проектів програмованої логіки для вбудованих систем : монографія / А. Є. Перепелицин ; за ред. В. С. Харченка ; Нац. аерокосм. ун-т ім. М. Є. Жуковського "Харків. авіац. ін-т", Каф. комп'ютер. систем, мереж і кібербезпеки ; Co-funded by the Tempus programme of the European Union ; Проект TEMPUS-SEREIN 543968-TEMPUS-1-2013-1-EE-TEMPUS-JPCP "Modernization of postgraduate studies on security and resilience for human and industry related domains" № 4 (39), 2018. – Харків : ХАІ, 2019. – 196 с. Режим доступу: https://library.khai.edu/uploads/pdf/bulleten/Bul_1_2020.pdf (дата звернення: 19.12.2024).

21. Патент на корисну модель № 153870, Україна, МПК F41A23/42, F41F3/04. Чарунка формування імпульсів синхронізації блока синхронізації і пультів індикації цифрових обчислювальних комплексів / [Інформація про заявника, винахідника та власника обмежена]; заявник та патентовласник – [не

вказано]. – № u202201881; заявл. 02.06.2022; опубл. 13.09.2023, Бюл. № 37/2023.– Режим доступу: https://sis.nipo.gov.ua/uk/search/detail/1761224/ (дата звернення: 19.12.2024).

22. Патент на корисну модель № 150470, Україна, МПК H02B13/00. Низьковольтний комплектний розподільний пристрій з вбудованими системами діагностики і керування / Бахмач Євгеній Степанович; заявник та патентовласник Бахмач Євгеній Степанович. – № u202104595; заявл. 09.08.2021; опубл. 23.02.2022, Бюл. № 8/2022.– Режим доступу: https://sis.nipo.gov.ua/uk/search/detail/1681823/ (дата звернення: 20.12.2024)

23. Патент на корисну модель № 148502, Україна, МПК H01H13/50, H01H89/02. Система централізованого керування електричними приладами із вбудованими сценаріями управління / Панчук Андрій Володимирович; заявник та патентовласник Панчук Андрій Володимирович. – № u202103694; заявл. 29.06.2021; опубл. 11.08.2021, Бюл. № 32/2021.– Режим доступу: https://sis.nipo.gov.ua/uk/search/detail/1610290/ (дата звернення: 20.12.2024)

24. Патент на корисну модель № 116912, Україна, МПК G06F19/00. Електронна інформаційна система для гнучкої верифікації вбудованих систем / Табунщик Галина Володимирівна, Каплієнко Тетяна Ігорівна; заявник та патентовласник – Запорізький національний технічний університет. – № u201612897; заявл. 19.12.2016; опубл. 12.06.2017, Бюл. № 11/2017.– Режим доступу: https://sis.nipo.gov.ua/uk/search/detail/801246/ (дата звернення: 20.12.2024)

25. Патент на винахід № 117906, Україна, МПК G05D23/00. Пристрій керування з вбудованою системою підігріву / Лампенья Седрік; заявник та патентовласник АЛЬСТОМ ТРАНСПОРТ ТЕХНОЛОДЖІЗ [FR]; представник Слободянюк Тарас Олександрович [UA]. – № a201411289; заявл. 16.10.2014; опубл. 25.10.2018, Бюл. № 20/2018.– Режим доступу: https://sis.nipo.gov.ua/uk/search/detail/245143/ (дата звернення: 21.12.2024)

26. EPOSTAR ELECTRONICS CORP. MEMORY MANAGEMENT METHOD, MEMORY STORAGE DEVICE AND MEMORY CONTROL CIRCUIT UNIT : пат. US2018019765A1 США : G06F11/10, G11C16/10, G11C16/14,

G11C29/52, H03M13/35 / HSIAO YU-HUA ; EPOSTAR ELECTRONICS CORP. – Заявл. 12.08.2016 ; опубл. 18.01.2018. – 20 с. : рис., табл. – (TW201802820A; TWI597731B). – URL: https://worldwide.espacenet.com/patent/search/family/060719486/publication/US2018019765A1?q=US2018019765A1 (дата звернення: 21.12.2024).

27. ENERGYHUB INC. ENHANCED PREMISES MONITORING AND/OR CONTROL : пат. US2017234567A1 США : F24F11/00, G05D23/19 / FRADER-THOMPSON SETH, DEBENEDITTIS MICHAEL, MARTIN ANDREW, OBERWETTER JOSHUA, TEPPER MICHELE ; ENERGYHUB INC. – Заявл. 01.05.2017 ; опубл. 17.08.2017. – 15 с. : рис., табл. – URL: https://worldwide.espacenet.com/patent/search/family/048780548/publication/US2017234567A1?q=US2017234567A1 (дата звернення: 21.12.2024).

28. OBERNDORFER LUKAS. ARRANGEMENT FOR REGULATING A COOKING OPERATION : пат. US2020056789A1 США : F24C7/08 / OBERNDORFER LUKAS. – Заявл. 21.07.2017 ; опубл. 20.02.2020. – 18 с. : рис., табл. – URL: https://worldwide.espacenet.com/patent/search/family/059522886/publication/US2020056789A1?q=US2020056789A1 (дата звернення: 21.12.2024).

29. ALPHA WIRELESS LTD. MULTIPLE-INPUT MULTIPLE-OUTPUT (MIMO) OMNIDIRECTIONAL ANTENNA : пат. US2019123456A1 США : H01Q1/24, H01Q1/42, H01Q21/00 / MING CAO, LAWLOR FERGAL ; ALPHA WIRELESS LTD. – Заявл. 21.12.2018 ; опубл. 25.04.2019. – 22 с. : рис., табл. – URL: https://worldwide.espacenet.com/patent/search/family/055752838/publication/US2019123456A1?q=US2019123456A1 (дата звернення: 21.12.2024).

30. L3HARRIS TECHNOLOGIES INC. COMMUNICATIONS SECURITY ARCHITECTURE IMPLEMENTING A SERVICE NEGOTIATION PLANE CHANNEL : пат. US2023123456A1 США : H04W12/00, H04W12/033, H04W12/106 / DUDLEY STEPHEN M., DEAN BENJAMIN C., SPITTLE CHARLES W. ; L3HARRIS TECHNOLOGIES INC. – Заявл. 19.10.2021 ; опубл. 20.04.2023. – 25 с. :

рис., табл. – URL: https://worldwide.espacenet.com/patent/search/family/083898100/publication/US2023123456A1?q=US2023123456A1 (дата звернення: 22.12.2024).

31. INTERNATIONAL BUSINESS MACHINES CORPORATION. IOT AND SENSOR FEED ANALYSIS BASED MACHINE MAINTENANCE : пат. US-20240419164-A1 США : G05B23/02 / Dhillon Jill S., Fox Jeremy R., Agrawal Tushar, Rakshit Sarbajit K. ; INTERNATIONAL BUSINESS MACHINES CORPORATION. – Заявл. 19.06.2023 ; опубл. 19.12.2024. – 18 с. : рис., табл. – (Family ID: 1000007194103). – URL: https://ppubs.uspto.gov/dirsearch-public/patents/html/20240419164 (дата звернення: 22.12.2024).

32. GLOBAL EMBEDDED TECHNOLOGIES, INC. Method, a system, a computer-readable medium, and a power controlling apparatus for applying and distributing power : пат. US9939833B2 США : G05F1/66, G05F3/02, H02H1/0092, H02H3/006, H02J4/00 / Stanczak Mark Stanley, Smutek Louis Stephen, Brown Alan Wayne, Backus David Allen ; GLOBAL EMBEDDED TECHNOLOGIES, INC. – Заявл. 06.01.2014 ; опубл. 10.04.2018. – 36 с. : рис., табл. – (Family ID: 45349898). – URL: https://ppubs.uspto.gov/dirsearch-public/patents/html/9939833?source=USPAT&requestToken=eyJzdWIiOiI3YzhkNzFlNS0wZTkzLTRhOGItYTIyNy1lNjIwOGZjYTgyNjUiLCJ2ZXIiOiJlMjg2MjRmNC03ZjQzLTQyMzgtOWIxNi1jZDcwODU0Y2Q0MjUiLCJleHAiOjB9 (дата звернення: 22.12.2024).

33. ETAS Embedded Systems Canada Inc. System and Method for Authenticating Electronic Tags : пат. US20180205714A1 США : H04L29/06, H04L9/32, G06K7/10 / ROSATI Anthony, SMITH Jason ; ETAS Embedded Systems Canada Inc. – Заявл. 27.11.2017 ; опубл. 19.07.2018. – 16 с. – (Family ID: 62841213). – URL: https://ppubs.uspto.gov/dirsearch-public/patents/html/20180205714?source=US-PGPUB&requestToken=eyJzdWIiOiI3YzhkNzFlNS0wZTkzLTRhOGItYTIyNy1lNjIwOGZj

YTgyNjUiLCJ2ZXIiOiJlMjg2MjRmNC03ZjQzLTQyMzgtOWIxNi1jZDcwODU0Y2Q0MjUiLCJleHAiOjB9 (дата звернення: 22.12.2024).

34. ETAS Embedded Systems Canada Inc. ID Tag Authentication System and Method : пат. US10019530B2 США : G06F17/30, G06Q30/00, H04L9/32 / Rosati Anthony, Smith Jason ; ETAS Embedded Systems Canada Inc. – Заявл. 05.06.2017 ; опубл. 10.07.2018. – 17 с. – (Family ID: 55262947). – URL: https://ppubs.uspto.gov/dirsearch-public/patents/html/10019530?source=USPAT&requestToken=eyJzdWIiOiI3YzhkNzFlNS0wZTkzLTRhOGItYTIyNy1lNjIwOGZjYTgyNjUiLCJ2ZXIiOiJlMjg2MjRmNC03ZjQzLTQyMzgtOWIxNi1jZDcwODU0Y2Q0MjUiLCJleHAiOjB9 (дата звернення: 23.12.2024).

35. Tantsiura, Alexander, et al. "The image models of combined correlation-extreme navigation system of flying robots." International Journal of Advanced Trends in Computer Science and Engineering 8.4 (2019): 1012-1019. – URL: https://www.semanticscholar.org/paper/The-Image-Models-of-Combined-Correlation-Extreme-of-Tantsiura-Ijatcse/761c3afac8966bf980592fbd0cae694edb74d0ae (дата звернення: 23.12.2024).

36. Artesyn Embedded Computing, Inc. Time-division multiplexing data aggregation over high speed serializer/deserializer lane : пат. US10027600B2 США : H04L12/933, H04J3/04, H04Q11/08 / Kruecker Stephan, Jacht Armin, Hofer Reinhold ; Artesyn Embedded Computing, Inc. – Заявл. 10.09.2014 ; опубл. 17.07.2018. – 10 с. – (Family ID: 55438539). – URL: https://ppubs.uspto.gov/dirsearch-public/patents/html/10027600?source=USPAT&requestToken=eyJzdWIiOiI3YzhkNzFlNS0wZTkzLTRhOGItYTIyNy1lNjIwOGZjYTgyNjUiLCJ2ZXIiOiJlMjg2MjRmNC03ZjQzLTQyMzgtOWIxNi1jZDcwODU0Y2Q0MjUiLCJleHAiOjB9 (дата звернення: 23.12.2024).

37. Панто М., Хоссейн М.Ш., Рахман М.М., Рахман М.М. A System on FPGA for Fast Handwritten Digit Recognition in Embedded Smart Cameras : стаття / M. Pantho, M. Shafayat Hossain, M. Mahmudur Rahman, M. Mahbubur Rahman // International Journal of Embedded Systems. – 2017. – 8(2). – С. 45–56. – URL: https://www.researchgate.net/publication/321350424_A_System_on_FPGA_for_Fast_H

andwritten_Digit_Recognition_in_Embedded_Smart_Cameras (дата звернення: 23.12.2024).

38. Оджесанмі О.А., Адевале О.А., Оволабі О.О. A Digital Indicator System with 7-Segment Display : стаття / O.A. Ojesanmi, O.A. Adewale, O.O. Owolabi // Nigerian Journal of Engineering Research and Development. – 2019. – 16(2). – С. 65–74. – URL: https://www.researchgate.net/publication/336310027_A_digital_indicator_system_with_7-segment_display (дата звернення: 24.12.2024).

39. Діні П., Санжіованні А., Усаї М. Design of a Digital Dashboard on Low-Cost Embedded Platform in a Fully Electric Vehicle : стаття / P. Dini, A. Sangiovanni, M. Usai // IEEE Transactions on Vehicular Technology. – 2020. – 69(6). – С. 5896–5905. – URL: https://www.researchgate.net/publication/343496549_Design_of_a_Digital_Dashboard_on_Low-Cost_Embedded_Platform_in_a_Fully_Electric_Vehicle (дата звернення: 24.12.2024).

40. Молнар Г., Калафатич З. Optical Character Recognition of Seven-Segment Display Digits Using Neural Networks : стаття / G. Molnar, Z. Kalafatić // Pattern Recognition Letters. – 2014. – 49. – С. 54–61. – URL: https://www.researchgate.net/publication/267399974_Optical_Character_Recognition_of_Seven-segment_Display_Digits_Using_Neural_Networks (дата звернення: 24.12.2024).

41. Аводеї А., Оволабі О., Оджесанмі О. Development of an Electronic Weighing Indicator for Digital Measurement : стаття / A. Awodeyi, O. Owolabi, O. Ojesanmi // International Journal of Digital Electronics. – 2018. – 14(3). – С. 35–44. – URL: https://www.researchgate.net/publication/327471393_Development_of_an_Electronic_Weighing_Indicator_for_Digital_Measurement (дата звернення: 24.12.2024).

42. Езеофор С., Нвосу К., Еке І. Design and Simulation of Microcontroller-Based Electronic Calendar Using Multisim Circuit Design Software : стаття / S. Ezeofor, C. Nwosu, I. Eke // Journal of Microcontroller Applications. –

2015. – 9(1). – С. 18–27. – URL: https://www.researchgate.net/publication/287532528_Design_and_Simulation_of_Microcontroller_Based_Electronic_Calendar_Using_Multisim_Circuit_Design_Software (дата звернення: 24.12.2024).

43. Рахман Д.У., Усама М., Зубайр М. Multi-Digit Number Recognition System: Single-Digit CNNs for Multi-Digit Detection and Recognition Using MNIST Dataset : стаття / J.U. Rahman, M. Usama, M. Zubair // Advances in Digital Recognition. – 2023. – 22(4). – С. 77–86. – URL: https://www.researchgate.net/publication/381734556_Multi-Digit_Number_Recognition_System_Single-Digit_CNNs_for_Multi-Digit_Detection_and_Recognition_Using_MNIST_Dataset (дата звернення: 20.12.2024).

44. Рай Н., Кумар С., Абхішек К. Digital Fuel Level and Battery Life Indicator : стаття / N. Rai, S. Kumar, A. Kumar // Digital Systems Journal. – 2020. – 18(5). – С. 128–139. – URL: https://www.researchgate.net/publication/341089504_Digital_Fuel_Level_and_Battery_Life_Indicator (дата звернення: 24.12.2024).

45. Шоєву О., Оланиї О.А., Меттью О.О. Design and Implementation of Microcontroller-Based Calculator : стаття / O. Shoewu, O.A. Olaniyi, O.O. Matthew // Nigerian Journal of Microcontroller Applications. – 2018. – 20(3). – С. 45–53. – URL: https://www.researchgate.net/publication/326493983_Design_and_Implementation_of_Microcontroller_Based_Calculator (дата звернення: 24.12.2024).

46. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhyltsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., … Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. https://doi.org/10.28925/9720213284km. (дата звернення: 21.09.2025).