Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2024 р

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «**Метод керування електронним навантаженням за протоколом SPI**»

Захищено на засіданні
Атестаційної комісії № 42
протокол № __ від __.06.2024 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **СКОБ Ю. О.**
(підпис) (прізвище та ініціали)

Виконав:
студент 4 курсу, групи КУ– 41
Галузь знань: 15 – Автоматизація та приладобудування
Спеціальність: 151 – «Автоматизація та комп'ютерно-інтегровані технології»
Шульга Руслан Володимирович_____
(прізвище, ім'я та по батькові) (підпис)

Керівник: к.ф.-м.н., доц. ЗВО
КОТВИЦЬКИЙ Альберт Тадеушевич
(прізвище, ім'я та по батькові)
_____
(підпис)

Рецензент: к.т.н., доц. ЗВО каф. електроніки та управляючих систем
РЕВА Сергій Миколайович_____
(прізвище, ім'я та по батькові) (підпис)

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків. Загальний обсяг роботи складає 65 сторінки, з яких 48 сторінок основної частини з 34 рисунками, 8 таблиць, 10 найменуваннями списку використаних джерел та додатками.

Метою кваліфікаційної роботи: підвищення ефективності управління електронним навантаженням за допомогою протоколу SPI для покращення продуктивності та надійності мікроконтролерних систем.

Об'єкт дослідження: процеси керування електронними навантаженнями у мікроконтролерних системах.

Предмет дослідження: протокол SPI та його застосування для керування електронними навантаженнями, включаючи апаратну підтримку протоколу у мікроконтролерах ATmega328 та ATmega2560.

Проблема, яка вирішується в кваліфікаційній роботі, полягає у визначенні ефективних методів керування електронними навантаженнями, використовуючи SPI, для підвищення надійності та продуктивності системи.

Область застосування: розробка вбудованих систем та мікроконтролерних пристроїв для автоматизації та приладобудування. Розроблений метод може широко використовуватися в сфері автоматизації, IT-бізнесу та вбудованих систем.

Ключові слова: SPI, мікроконтролери, ATmega328, ATmega2560, електронне навантаження, керування, протокол, апаратна підтримка, симуляція, автоматизація.

ABSTRACT

The explanatory note to the bachelor's qualification work consists of an introduction, three chapters, conclusions, a list of references, and appendices. The total volume of the work is 65 pages, including 48 pages of the main part with 34 figures, 8 table, 10 references, and appendices.

The purpose of the qualification: Enhancing the efficiency of electronic load control using the SPI protocol to improve the performance and reliability of microcontroller systems.

Object of research: processes of controlling electronic loads in microcontroller systems.

Subject of research: the SPI protocol and its application for controlling electronic loads, including hardware support for the protocol in ATmega328 and ATmega2560 microcontrollers.

The problem addressed in the qualification work is to identify effective methods for controlling electronic loads using SPI to enhance the reliability and performance of the system.

Field of application: development of embedded systems and microcontroller devices for automation and instrumentation. The developed method can be widely used in the field of automation, IT business, and embedded systems.

Keywords: SPI, microcontrollers, ATmega328, ATmega2560, electronic load, control, protocol, hardware support, simulation, automation.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

SPI	-	Serial Peripheral Interface (Послідовний периферійний інтерфейс)
MOSI	-	Master Out Slave In (Вихід майстра, вхід підлеглого)
MISO	-	Master In Slave Out (Вхід майстра, вихід підлеглого)
SCK	-	Serial Clock (Серійний годинник)
SS	-	Slave Select (Вибір підлеглого)
PWM	-	Pulse Width Modulation (Широтно-імпульсна модуляція)
SSR	-	Solid State Relay (Твердотільне реле)
BJT	-	Bipolar Junction Transistor (Біполярний перехідний транзистор)
FET	-	Field-Effect Transistor (Польовий транзистор)
MOSFET	-	Metal-Oxide-Semiconductor Field-Effect Transistor (Метал-оксид-напівпровідниковий польовий транзистор)
EEPROM	-	Electrically Erasable Programmable Read-Only Memory (Електрично стираєма програмована постійна пам'ять)
ADC	-	Analog-to-Digital Converter (Аналого-цифровий перетворювач)
DAC	-	Digital-to-Analog Converter (Цифро-аналоговий перетворювач)
I2C	-	Inter-Integrated Circuit (Міжінтегральний інтерфейс)
UART	-	Universal Asynchronous Receiver/Transmitter (Універсальний асинхронний приймач/передавач)
CAN	-	Controller Area Network (Мережа контролерів)
RS232	-	Recommended Standard 232 (Рекомендований стандарт 232)
GPIO	-	General Purpose Input/Output (Загальний вхід/вихід)
PCB	-	Printed Circuit Board (Друкована плата)

ВСТУП

У сучасному технологічному світі, де швидкість і точність є вирішальними, управління електронними навантаженнями стає все більш складним завданням. Передові інтерфейси, як-от Serial Peripheral Interface (SPI), відіграють ключову роль у керуванні різними пристроями в автоматизованих системах, забезпечуючи високу швидкість передачі даних та надійність управління. Однак, зі зростанням комплексності систем, підвищуються вимоги до точності та ефективності методик управління.

Актуальність дослідження полягає у розробці нових методик, що дозволять оптимізувати процеси управління електронними навантаженнями через SPI, забезпечуючи максимальну продуктивність системи з мінімальними затратами ресурсів. Це особливо важливо в областях, де потрібна висока надійність і точність, наприклад, в медичних приладах або в системах діагностики.

Метою цієї роботи є підвищення ефективності управління електронним навантаженням за допомогою протоколу SPI для покращення продуктивності та надійності мікроконтролерних систем.

Предметом дослідження є протокол SPI та його застосування для керування електронними навантаженнями, включаючи апаратну підтримку протоколу у мікроконтролерах ATmega328 та ATmega2560.

Задачами дослідження є:

1. Аналіз існуючих методик керування електронними навантаженнями та їхньої ефективності.

2. Експериментальна оцінка розроблених методів на основі аналізу результатів тестування.

3. Визначення параметрів для максимальної ефективності управління в залежності від типу навантаження.

РОЗДІЛ 1.
АНАЛІЗ МЕТОДИК КЕРУВАННЯ ЕЛЕКТРОННИМ НАВАНТАЖЕННЯМ

1.1 Електронне навантаження

Електронне навантаження — це термін, який використовується для опису будь-якого пристрою або компонента в електронній системі, який споживає електричну енергію. Воно може включати в себе широкий спектр елементів, від простих резисторів, які розсіюють енергію у формі тепла, до складних мікропроцесорів, що виконують обчислення та обробку даних.

Електронні навантаження можуть бути активними або пасивними:

- Пасивні навантаження (наприклад, резистори та конденсатори) використовуються для контролю потоку електричної енергії без будь-якої зміни або підсилення сигналу.
- Активні навантаження (наприклад, транзистори та інтегральні схеми) виконують функції обробки або підсилення сигналів і можуть вимагати зовнішніх джерел живлення для функціонування.

1.2 Протокол для управління електронним навантаженням

Протокол для управління електронним навантаженням — це набір правил і процедур, що визначають формат та порядок передачі даних між керуючим пристроєм (наприклад, комп'ютером або мікроконтролером) і електронним навантаженням. Протоколи забезпечують узгоджену комунікацію, дозволяючи керуючому пристрою відправляти команди і отримувати зворотний зв'язок від навантаження.

Основні елементи протоколу управління електронним навантаженням включають:

Формат повідомлень: Визначає структуру даних, які передаються між пристроями. Це може включати заголовки, тіла повідомлень, контрольні суми тощо.

Команди: Набір команд, які керуючий пристрій може відправляти електронному навантаженню для виконання певних дій, таких як встановлення струму, напруги, режиму роботи тощо.

Відповіді: Формат і зміст відповідей, які електронне навантаження надсилає у відповідь на команди. Це можуть бути підтвердження отримання команди, поточні значення параметрів, повідомлення про помилки тощо.

Синхронізація: Механізми для забезпечення синхронного обміну даними між пристроями, включаючи тайм-аути, повторні запити та інші методи забезпечення цілісності даних.

Аутентифікація та безпека: Процедури для забезпечення безпеки комунікації, включаючи аутентифікацію пристроїв і шифрування даних.

Прикладами таких протоколів можуть бути стандартні інтерфейси, такі як UART, I2C, SPI, або спеціалізовані протоколи, розроблені виробниками електронних навантажень.

Приклади протоколів управління електронними навантаженнями, кожен з яких має свої переваги та обмеження, що дозволяє вибрати оптимальний варіант залежно від конкретних вимог і умов застосування див. рисунок 1.1

		0	1	2	3	4	5	6	7
1	Протокол / Стандарт	UART	I2C	SPI	1-провідний	CAN	ЛІН	RS-485	RS-232
2	Макс. Вузол	2	127 або 1023	Залежить від шпильок SS	2 ^ 48	128	16	256	2
3	Макс. Бітрейт [kbps]	-	5000	До 10000	16.3	1000	19.2	До 10000	128
4	Макс. Довжина [м]	-	Занижено	Занижено	300	500	40	1330	15
5	Переваги	1. Повний дуплекс 2. використовує лише два дроти 3. Може забезпечувати як	1. Використовує лише два дроти 2. Multi-master і multi slave 3. Більше рабів не	1. Повний дуплекс 2. Дуже прості драйвери 3. Більш надійний. ніж UART	1. Потрібен лише один провід 2. Конфігурація паразитної потужності	1. Дуже міцний 2. Багатопідпорядкований 3. Виявлення	1. Дуже дешево 2. Потрібен лише один провід 3. Може мати до 15 збереження	1. Висока досяжна швидкість передачі даних 2. Висока досяжна відстань	1. Дешево
6	Недоліки	1. Можна підключити лише 2 пристрої 2. Контролер повинен	1. Повільніше порівняно з SPI 2. Більш складне обладнання, ніж SPI 3. Слейви повинні	1. Використовує три або більше проводів (слідів) 2. Більше рабів збільшує кількість	1. Тільки пристрій Maxim має цей послідовний зв'язок	1. Досить дорого	1. Низька швидкість передачі даних	1. Більше енергоспоживання 2. Досить складне обладнання	1. Низька швидкість передачі даних 2. Сучасні пристрої рідко мають це підключення або не
7	примітки	1. Конфігурована швидкість передачі даних 2. Можна використовувати	1. Потрібні навантажувальні резистори 2. Базовий зв'язок IC між IC	1. Базовий зв'язок IC між IC 2. Дані дисплея/зображення з низькою	1. Вимагає підтягування	1. Лінія електропередачі з опором 120 Ом 2. Використовує диференціальну	1. Використовується переважно в автомобільній електроніці	1. Використовує одну або дві диференціальні пари 2. Зв'язок між	1. Його часто можна побачити в багатьох пристроях попереднього покоління

Рисунок 1.1 – Приклади протоколів управління електронними навантаженнями

1.3 Протокол SPI

Serial Peripheral Interface (SPI) — це синхронний протокол передачі даних, що використовується для зв'язку між мікроконтролерами та різними периферійними пристроями, такими як сенсори, пам'ять та інші мікросхеми. SPI був розроблений компанією Motorola у 1980-х роках для забезпечення швидкого обміну даними між внутрішніми компонентами.

Основні характеристики SPI:

Режими Роботи:

1. Режим 0 (CPOL=0, CPHA=0): Годинниковий сигнал починається з низького рівня, а передача даних відбувається при переході сигналу з низького рівня на високий (рисунок 1.2 Mode 0).
2. Режим 1 (CPOL=0, CPHA=1): Годинниковий сигнал починається з низького рівня, а передача даних відбувається при переході сигналу з високого рівня на низький (рисунок 1.2 Mode 1).
3. Режим 2 (CPOL=1, CPHA=0): Годинниковий сигнал починається з високого рівня, а передача даних відбувається при переході сигналу з високого рівня на низький (рисунок 1.2 Mode 2).
4. Режим 3 (CPOL=1, CPHA=1): Годинниковий сигнал починається з високого рівня, а передача даних відбувається при переході сигналу з низького рівня на високий (рисунок 1.2 Mode 3).

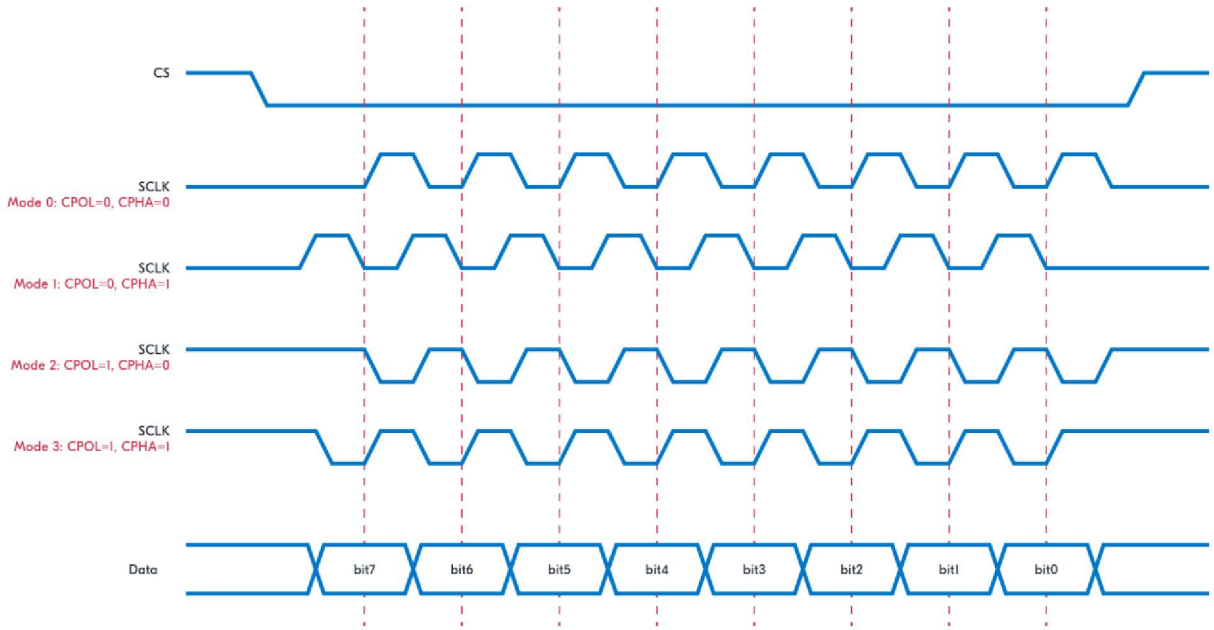

Рисунок 1.2 – режими роботи SPI

Вибір режиму залежить від конкретних вимог периферійних пристроїв, з якими спілкується мікроконтролер. Різні пристрої можуть вимагати різних режимів, тому важливо звертати увагу на документацію до периферійних компонентів для вибору правильного режиму роботи.

Швидкість передачі даних:

Технічні Обмеження:

1. **Частота Годинника:** Основним фактором, що визначає швидкість передачі даних у SPI, є частота годинника SCK (Serial Clock). Максимально можлива частота залежить від характеристик мікроконтролера та периферійних пристроїв. Деякі мікроконтролери та чіпи підтримують частоту годинника до декількох десятків МГц.

2. **Підтримка Процесором та Периферією:** Максимальна швидкість передачі може бути обмежена можливостями найповільнішого пристрою в ланцюжку SPI. Наприклад, якщо периферійний пристрій може обробляти дані лише на частоті до 5 МГц, то немає сенсу налаштовувати мікроконтролер на вищу частоту.

Практичні Обмеження:

1. **Довжина і якість з'єднання:** Довші проводи можуть спричиняти вищі рівні електромагнітних завад та втрат сигналу, що обмежує максимальну частоту годинника та знижує надійність передачі даних.

2. **Електромагнітні завади:** У середовищах з високим рівнем електромагнітних завад може знадобитися знизити швидкість передачі даних або застосувати додаткові методи захисту сигналу, наприклад, використання екранованих кабелів.

Приклади Максимальних Швидкостей:

- Деякі SPI інтерфейси мікроконтролерів, таких як STM32, можуть працювати з частотою годинника до 42 МГц.
- SPI в мікроконтролерах від Atmel можуть підтримувати швидкість до 10-20 МГц, залежно від конкретної моделі.
- Пристрої пам'яті, такі як SD-карти, часто використовують SPI зі швидкістю близько 25 МГц в стандартному режимі.

Швидкість передачі даних у SPI можна адаптувати під конкретні потреби проекту, вибираючи відповідну частоту годинника і забезпечуючи адекватну якість з'єднань. Це робить SPI дуже гнучким рішенням для багатьох застосувань у вбудованих системах.

Майстер та підлеглі:

Майстер (Master):

Майстер у SPI-з'єднанні відіграє центральну роль. Він відповідальний за:

1. **Генерацію Годинникового Сигналу (SCK):** Майстер генерує годинниковий сигнал, який синхронізує передачу даних на всіх підлеглих пристроях. Це означає, що частоту та полярність годинника контролює майстер, і всі підлеглі пристрої повинні працювати відповідно до цього сигналу.

2. **Вибір Підлеглих (Slave Select, SS):** Майстер використовує окремі лінії Slave Select для керування тим, який із підлеглих пристроїв активний у даний момент. Це дозволяє майстру комунікувати з декількома підлеглими на одній шині, активуючи їх одного за одним.

3. **Ініціація Передачі Даних:** Майстер відповідальний за ініціацію та контроль за всім процесом передачі даних. Він може одночасно відправляти дані на підлеглого пристрою (через лінію MOSI) та отримувати дані від нього (через лінію MISO).

Підлеглі (Slaves):

Підлеглі у SPI-системі — це пристрої, які керуються майстром:

1. **Приймання Годинникового Сигналу:** Підлеглі синхронізують свої операції з передачі даних з годинниковим сигналом, який генерує майстер.

2. **Реагування на Вибір Майстром:** Підлеглі активуються та деактивуються за допомогою ліній SS, які контролюються майстром. Коли підлеглий активований, він може спілкуватися з майстром, відповідаючи на запити даних або приймаючи дані від майстра.

3. **Двонаправлена Передача Даних:** Під час активації, підлеглий може отримувати дані з лінії MOSI та відправляти дані до майстра через лінію MISO, все це залежно від потреб системи.

1.4 Як SPI протокол використовується для керування електронними навантаженнями?

1. Основні компоненти SPI

SPI складається з чотирьох основних ліній:

- **MOSI (Master Out Slave In):** Лінія для передачі даних від майстра до підлеглого.
- **MISO (Master In Slave Out):** Лінія для передачі даних від підлеглого до майстра.
- **SCLK (Serial Clock):** Годинниковий сигнал, що синхронізує передачу даних.

- **SS (Slave Select):** Лінія вибору підлеглого, що активує певний пристрій для комунікації.

2. Процесс обміну данними

- **Ініціалізація зв'язку:** Майстер (наприклад, мікроконтролер) активує підлеглого (електронне навантаження) за допомогою лінії SS.
- **Передача команд:** Майстер відправляє команди по лінії MOSI, синхронізуючи їх з годинниковим сигналом на лінії SCLK.
- **Прийом даних:** Підлеглий відповідає, відправляючи дані по лінії MISO. Ці дані можуть бути зворотним зв'язком про стан навантаження або результатами вимірювань.

3. Конфігурація електронного навантаження

- **Встановлення параметрів:** Майстер може відправляти команди для встановлення параметрів навантаження, таких як струм, напруга, потужність або опір.
- **Перемикання режимів:** Команди можуть використовуватися для перемикання між різними режимами роботи електронного навантаження (CC, CV, CP, CR).

1.5 Аналіз та вибір існуючих пристроїв, які керуються по SPI.

1. Цифро-аналогові перетворювачі (DAC)

Цифро-аналогові перетворювачі використовуються для перетворення цифрових сигналів у аналогові. DAC є важливими у аудіосистемах, інструментальній техніці та будь-яких інших застосуваннях, де потрібно контролювати аналогові величини з цифрових систем. Прикладом є MCP4922 від Microchip, який надає двоканальний вихід і має вбудований інтерфейс SPI.

2. Аналогово-цифрові перетворювачі (ADC)

ADC використовуються для перетворення аналогових сигналів, таких як температура, тиск, світловий рівень, у цифрові дані, які може обробляти мікропроцесор. ADC, як AD7793 від Analog Devices, забезпечує високу точність і широко використовується в промислових додатках.

3. Пам'ять

SPI також широко використовується для зв'язку з чіпами пам'яті, включаючи EEPROM і flash-пам'ять. Ці пристрої, як 25 Series SPI Flash, надають великі обсяги зберігання та швидкість передачі даних, ідеальні для зберігання коду програм, конфігураційних файлів або даних користувача.

4. Датчики

Багато сучасних сенсорів використовують SPI для передачі точних вимірювань до мікроконтролерів. До таких сенсорів можуть входити гіроскопи, акселерометри, та інші сенсори довкілля. Bosch Sensortec BMP280, наприклад, це барометричний сенсор, який надає дані про температуру та тиск через SPI.

5. Інтегровані схеми управління

Драйвери моторів, контролери світлодіодів та інші інтегровані контролери часто включають підтримку SPI, що дозволяє забезпечити фіне налаштування та контроль через програмне забезпечення. Прикладом є драйвери моторів, як L6470 від STMicroelectronics, який може бути керований через SPI для точного контролю швидкості та положення крокового мотора.

1.6 Постановка задачі дослідження

Цей розділ визначає ключові дослідницькі задачі, які будуть вирішуватися в рамках кваліфікаційної роботи на тему "Методика керування електронним навантаженням за протоколом SPI". Ці задачі спрямовані на покращення існуючих методик і розробку нових підходів до керування за допомогою SPI, зокрема на основі аналізу недоліків існуючих систем та впровадження сучасних технологічних рішень.

Задачі дослідження включають:

1. **Аналіз існуючих методик керування електронними навантаженнями:**

- Детальний аналіз технічних характеристик і принципів роботи існуючих методик.
- Визначення основних проблем і недоліків поточних методів у контексті різних застосувань.

2. **Експериментальна перевірка розробленої методики:**
   - Побудова прототипу системи на основі мікроконтролерів з використанням SPI.
   - Тестування прототипу в лабораторних умовах для визначення його ефективності та надійності.
3. **Аналіз результатів експериментів та оптимізація методики:**
   - Вивчення зібраних даних для оцінки продуктивності та можливих поліпшень.
   - Рефінансування параметрів системи на основі отриманих результатів.
4. **Підготовка рекомендацій для впровадження в практику:**
   - Розробка методичних вказівок для застосування нової методики в промислових та інших практичних умовах.
   - Визначення можливих напрямків подальших досліджень на основі аналізу та експериментів.

Висновки за розділом 1

У цьому розділі я дослідив історію, розвиток і поточне застосування протоколу SPI, а також існуючі методики керування електронними навантаженнями. Цей огляд надав фундаментальне розуміння технологій та визначив основні виклики та обмеження, які стоять перед сучасними системами керування.

Основні висновки включають:

1. **Розвиток SPI:** Протокол SPI продемонстрував високу ефективність у комунікаціях між мікроконтролерами та периферійними пристроями, завдяки чому знайшов широке застосування у різноманітних

галузях. Його гнучкість і швидкість забезпечують важливу перевагу для реалізації високопродуктивних систем.

2. **Аналіз існуючих методів керування:** Хоча існуючі методи, такі як PWM та цифрові потенціометри, є ефективними для певних застосувань, вони мають низку обмежень, зокрема високе енергоспоживання та складнощі в масштабуванні.

3. **Проблеми та можливості для покращення:** Існуючі системи часто стикаються з проблемами у масштабуванні, затримками в реакціях та недостатньою ефективністю в умовах змінних навантажень. Новітні дослідження та розробки, зокрема використання машинного навчання та адаптивних систем, можуть пропонувати значні покращення.

4. **Напрями подальших досліджень:** Огляд літератури підкреслив необхідність подальшого вивчення та розробки нових методів керування, які б могли краще відповідати потребам сучасних високотехнологічних застосувань.

Ці висновки визначають напрями подальшої роботи у розробці нових методів керування електронними навантаженнями за допомогою SPI, які будуть розглянуті у наступних розділах роботи. Метою є не лише вирішення існуючих проблем, а й розробка більш ефективних, адаптивних та енергоефективних систем.

РОЗДІЛ 2.
СТВОРЕННЯ СИМУЛЯЦІЇ МІКРОКОНТРОЛЕРНОГО КЕРУВАННЯ ЗА ПРОТОКОЛОМ SPI

В даному розділі розглянуто процес створення симуляції мікроконтролерного керування електронним навантаженням за допомогою протоколу SPI (Serial Peripheral Interface). Симуляція є важливим етапом дослідження, оскільки дозволяє верифікувати коректність реалізації керування та оптимізувати налаштування системи до безпосереднього впровадження в апаратному середовищі.

2.1 Вибір платформи та інструментів

Для створення симуляції було обрано мікроконтролери ATmega328 та ATmega2560, які широко використовуються у вбудованих системах і мають апаратну підтримку протоколу SPI. Основними інструментами для симуляції є програмне середовище Arduino IDE та Proteus Design Suite, що дозволяють моделювати роботу мікроконтролерів та периферійних пристроїв.

Характеристики SPI в ATmega328

1. **Режими роботи**:
    - Master (ведучий).
    - Slave (ведений).
2. **Реєстри конфігурації**:
    - **SPCR** (SPI Control Register): регістр керування SPI.
    - **SPSR** (SPI Status Register): регістр статусу SPI.
    - **SPDR** (SPI Data Register): регістр даних SPI.
3. **Швидкість передачі**:

- Максимальна частота тактового сигналу SCK: до Fosc/2 (де Fosc — частота генератора).
- Вибір попереднього ділення частоти: Fosc/4, Fosc/16, Fosc/64, Fosc/128.

4. **Підтримка SPI режимів**:
   - Чотири режими: режим 0 (CPOL=0, CPHA=0), режим 1 (CPOL=0, CPHA=1), режим 2 (CPOL=1, CPHA=0), режим 3 (CPOL=1, CPHA=1).
5. **Входи/виходи**:
   - MOSI: PB3 (порт B, біт 3).
   - MISO: PB4 (порт B, біт 4).
   - SCK: PB5 (порт B, біт 5).
   - SS: PB2 (порт B, біт 2).

Характеристики SPI в ATmega2560

1. **Режими роботи**:
   - Master (ведучий).
   - Slave (ведений).
2. **Реєстри конфігурації**:
   - **SPCR** (SPI Control Register): регістр керування SPI.
   - **SPSR** (SPI Status Register): регістр статусу SPI.
   - **SPDR** (SPI Data Register): регістр даних SPI.
3. **Швидкість передачі**:
   - Максимальна частота тактового сигналу SCK: до Fosc/2 (де Fosc — частота генератора).

- Вибір попереднього ділення частоти: Fosc/4, Fosc/16, Fosc/64, Fosc/128.

4. **Підтримка SPI режимів**:
    - Чотири режими: режим 0 (CPOL=0, CPHA=0), режим 1 (CPOL=0, CPHA=1), режим 2 (CPOL=1, CPHA=0), режим 3 (CPOL=1, CPHA=1).
5. **Входи/виходи**:
    - MOSI: PB2 (порт B, біт 2).
    - MISO: PB3 (порт B, біт 3).
    - SCK: PB1 (порт B, біт 1).
    - SS: PB0 (порт B, біт 0).

2.1.1 Порівняння ATmega328 та ATmega2560

Хоча характеристики SPI інтерфейсу в мікроконтролерах ATmega328 та ATmega2560 дуже схожі, відмінності полягають в різних розташуваннях виводів на мікроконтролерах. Обидва мікроконтролери підтримують ті ж самі режими роботи SPI, однакові швидкості передачі та конфігураційні реєстри.

SPI в мікроконтролерах ATmega328 та ATmega2560 має подібні характеристики, що дозволяє використовувати однакові алгоритми керування для обох пристроїв. Основні відмінності полягають у фізичному розташуванні контактів, що необхідно враховувати при проектуванні апаратної частини.

2.1.2 Створення симуляції в Proteus

1. **Створення проекту** (рисунок. 2.1)**:**

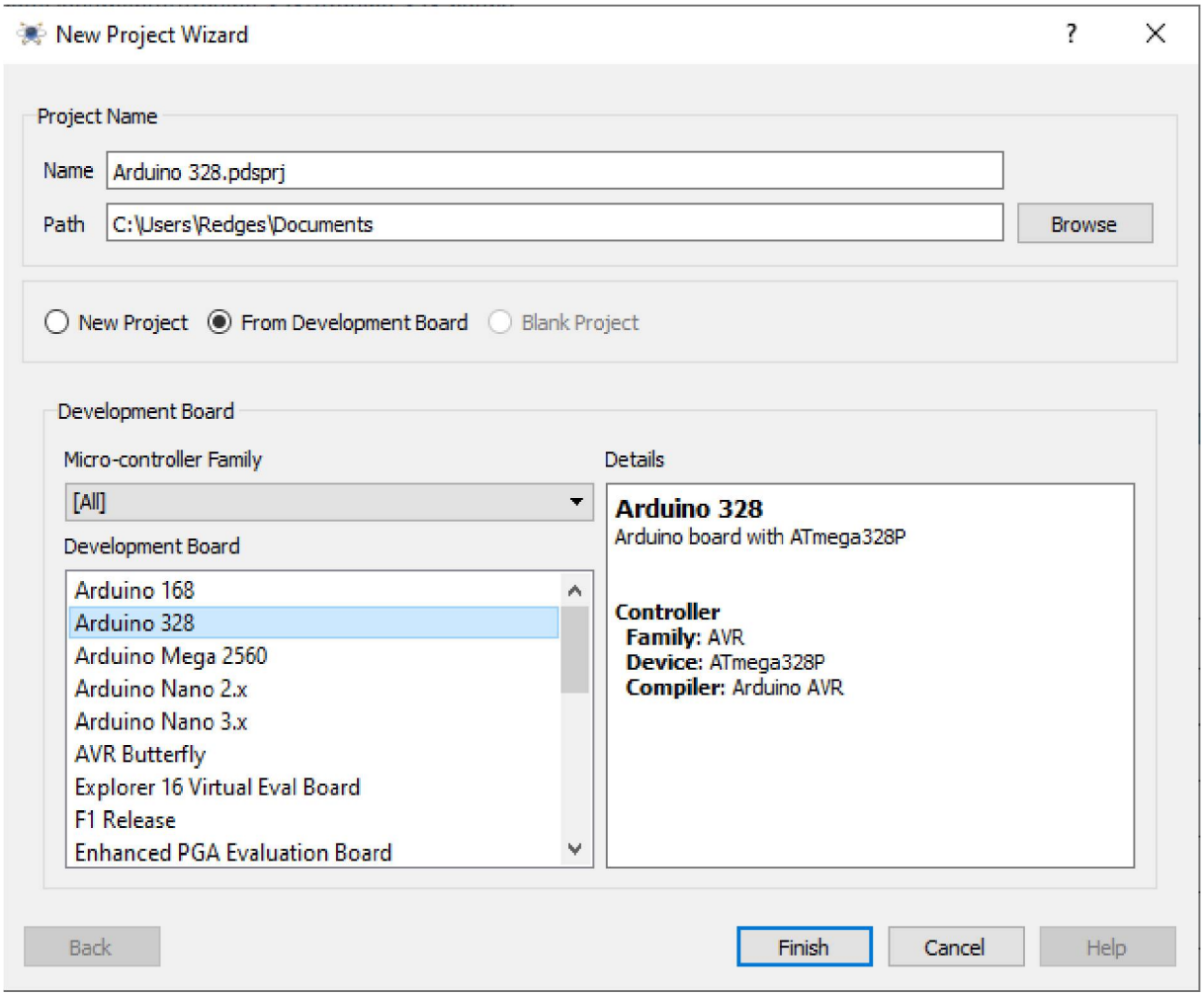

Рисунок 2.1 – Створення проекту в Proteus

Додавання до схеми двох мікроконтролерів ATmega328 та ATmega2560(рисунки 2.2, 2.3)

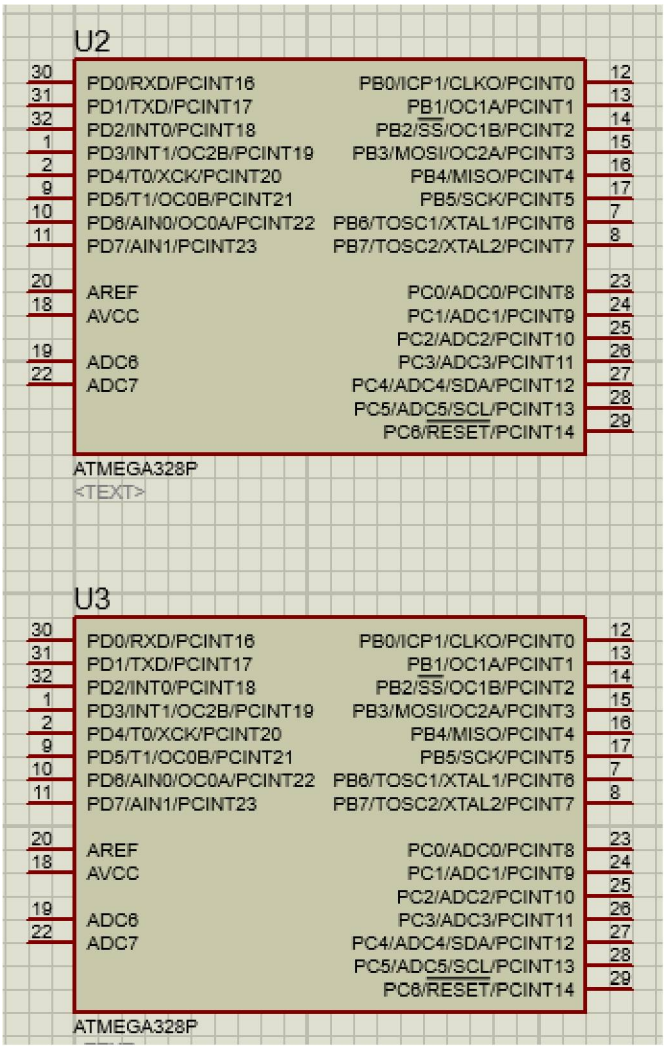

Рисунок 2.2 – додовання до схеми двох мікроконтролерів ATmega328

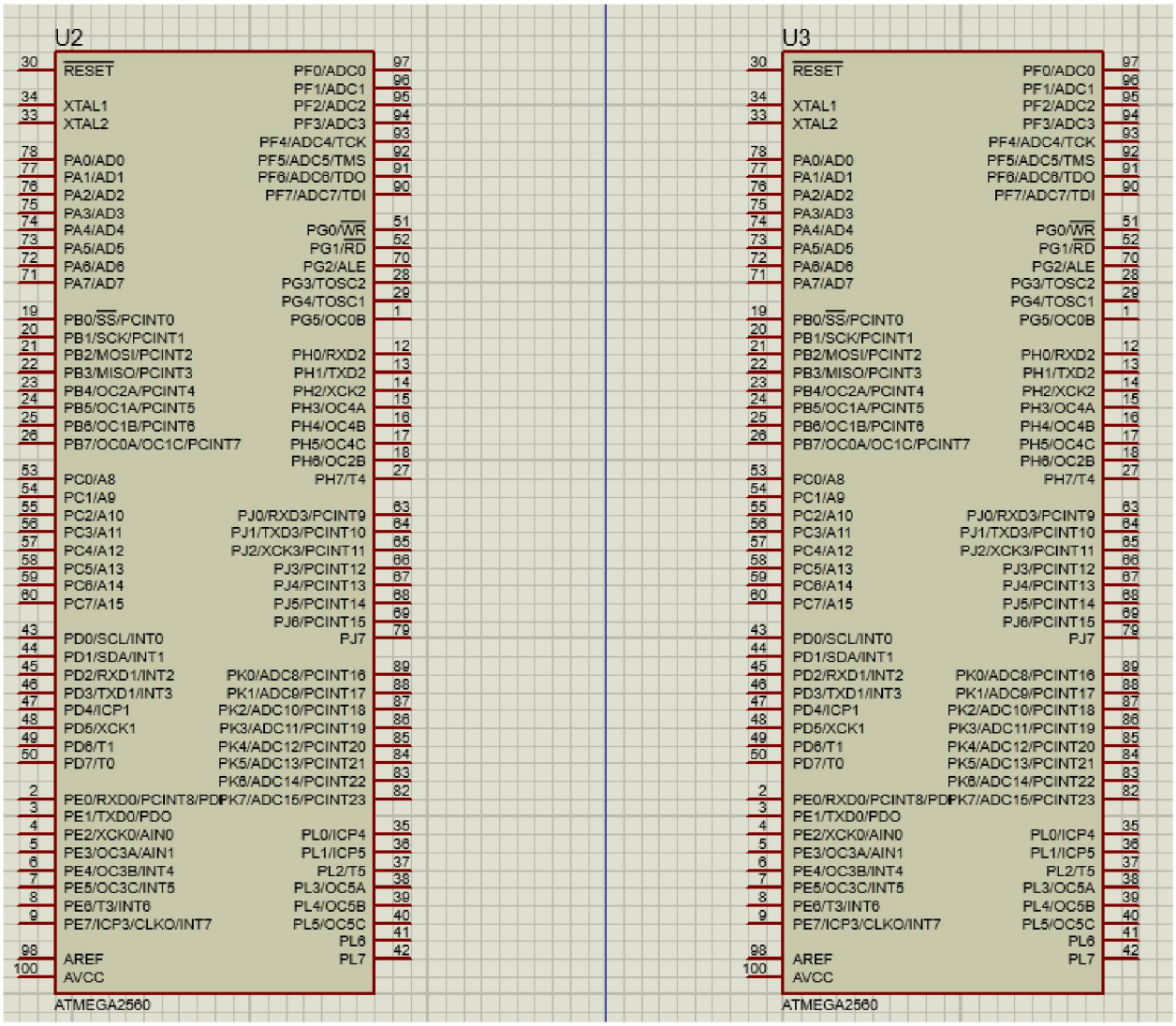

Рисунок 2.3 – додовання до схеми двох мікроконтролерів ATmega2560

Підключення SPI пристроїв:

1. З'єднання ліній MOSI, MISO, SCK та SS (Slave Select) між мікроконтролерами (рисунки 2.4, 2.5)

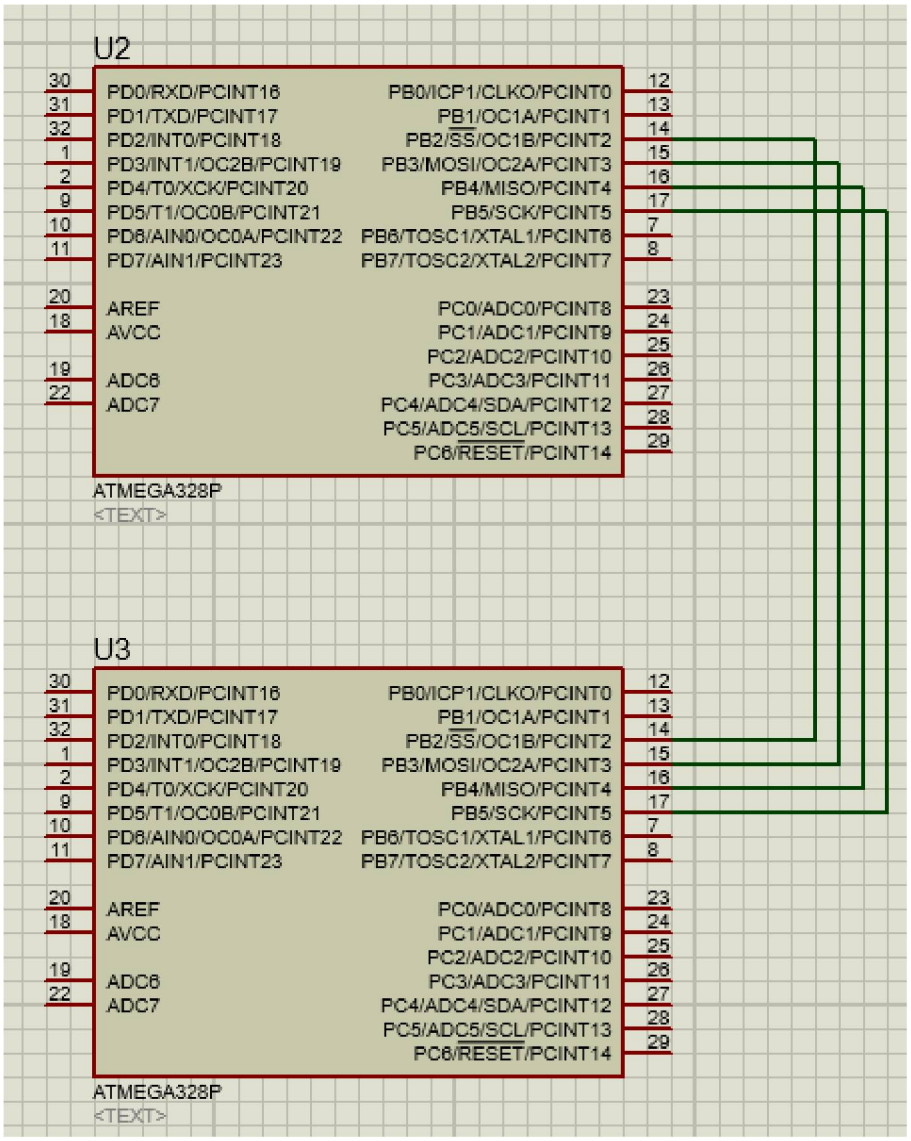

Рисунок 2.4 - З'єднання ліній MOSI, MISO, SCK та SS (Slave Select) між мікроконтролерами ATmega328

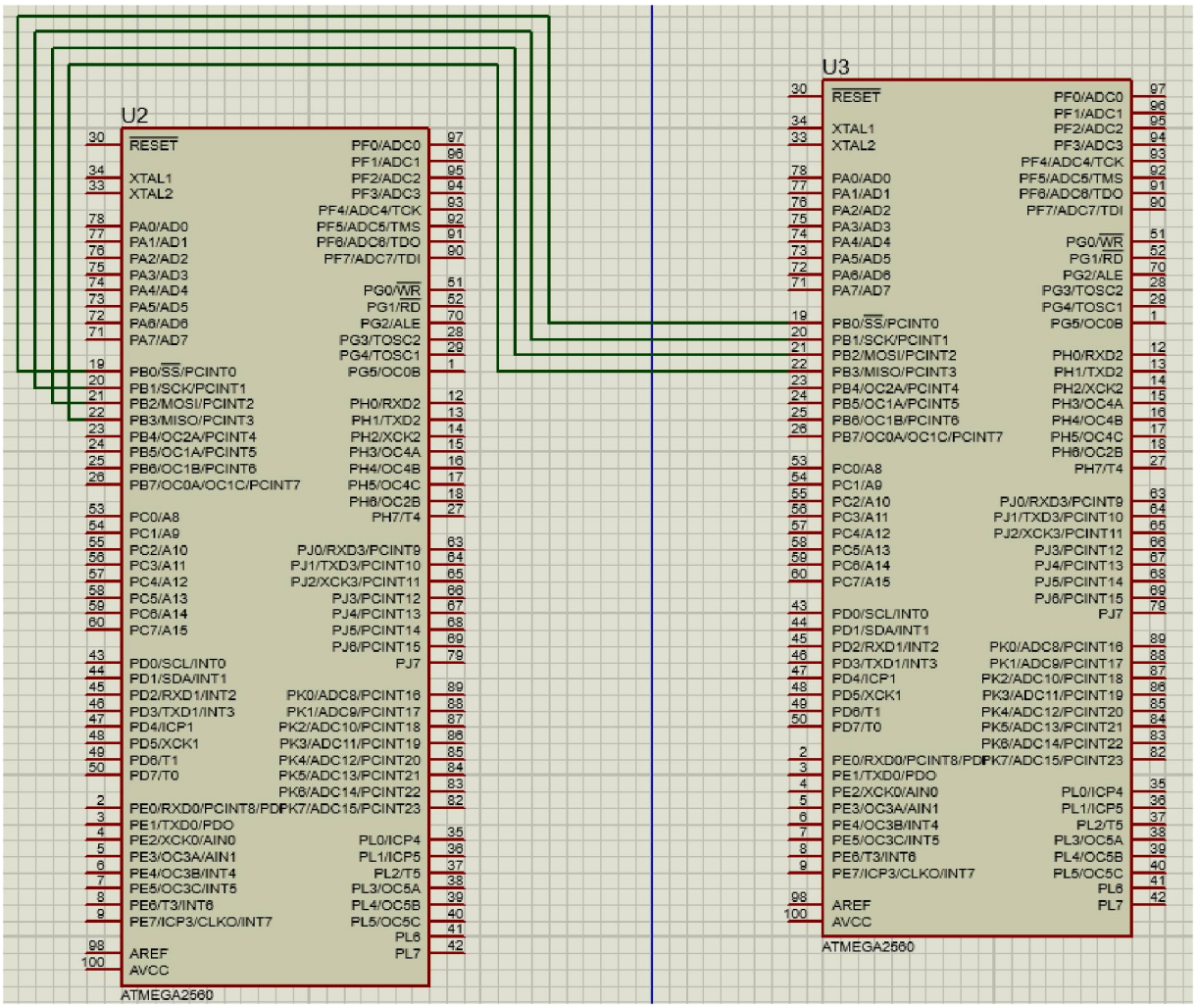

Рисунок 2.5 - З'єднання ліній MOSI, MISO, SCK та SS (Slave Select) між мікроконтролерами ATmega2560

2.1.3 Побудова симуляції на мові програмування C

Основні функції та регістри для SPI

Ініціалізація SPI (Master) рисунок 2.6:

```
void SPI_MasterInit(void) {
    // Встановлюємо MOSI і SCK як виходи, всі інші як входи
    DDRB = (1<<PB3)|(1<<PB5)|(1<<PB2);
    // Увімкнути SPI, встановити як Master, встановити частоту SCK
    SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}
```

Рисунок 2.6 - Ініціалізація SPI (Master)

Ініціалізація SPI (Slave) рисунок 2.7:

```
void SPI_SlaveInit(void) {
    // Встановлюємо MISO як вихід, всі інші як входи
    DDRB = (1<<PB4);
    // Увімкнути SPI
    SPCR = (1<<SPE);
}
```

Рисунок 2.7 - Ініціалізація SPI (Slave)

Передача байта даних (Master/Slave) рисунок 2.8:

```
void SPI_Transmit(uint8_t data) {
    // Записати дані в регістр
    SPDR = data;
    // Очікувати завершення передачі
    while(!(SPSR & (1<<SPIF)));
}

uint8_t SPI_Receive(void) {
    // Очікувати завершення передачі
    while(!(SPSR & (1<<SPIF)));
    // Повернути отримані дані з регістра
    return SPDR;
}
```

Рисунок 2.8 - Передача байта даних (Master/Slave)

Встановлення порядку передачі бітів рисунок 2.9:

```
void SPI_setBitOrder(uint8_t bitOrder) {
    if(bitOrder == 0) {
        SPCR &= ~(1<<DORD); // MSBFIRST
    } else {
        SPCR |= (1<<DORD); // LSBFIRST
    }
}
```

Рисунок 2.9 - Встановлення порядку передачі бітів

Встановлення режиму передачі даних рисунок 2.10:

```
void SPI_setDataMode(uint8_t mode) {
    SPCR = (SPCR & ~((1<<CPOL) | (1<<CPHA))) | mode;
}
```

Рисунок 2.10 - Встановлення режиму передачі даних

Режими:

- SPI_MODE0: 0
- SPI_MODE1: (1<<CPHA)
- SPI_MODE2: (1<<CPOL)
- SPI_MODE3: (1<<CPOL) | (1<<CPHA)

Встановлення дільника частоти для SPI годинника риснок 2.11:

```
void SPI_setClockDivider(uint8_t divider) {
    SPCR = (SPCR & ~((1<<SPR1) | (1<<SPR0))) | (divider & 0x03);
    SPSR = (SPSR & ~(1<<SPI2X)) | ((divider >> 2) & 0x01);
}
```

Рисунок 2.11 - Встановлення дільника частоти для SPI годинника

Можливі значення для divider:

- SPI_CLOCK_DIV2: 0x04
- SPI_CLOCK_DIV4: 0x00
- SPI_CLOCK_DIV8: 0x05
- SPI_CLOCK_DIV16: 0x01

- SPI_CLOCK_DIV32: 0x06
- SPI_CLOCK_DIV64: 0x02
- SPI_CLOCK_DIV128: 0x03

Передача і прийом байта (одночасно) рисунок 2.12:

```
uint8_t SPI_Transfer(uint8_t data) {
    // Записати дані в регістр
    SPDR = data;
    // Очікувати завершення передачі
    while(!(SPSR & (1<<SPIF)));
    // Повернути отримані дані з регістра
    return SPDR;
}
```

Рисунок 2.12 - Передача і прийом байта (одночасно)

Використання функцій

Приклад ініціалізації SPI в режимі Master і передачі байта даних рисунок 2.13:

```
int main(void) {
    // Ініціалізуємо SPI як Master
    SPI_MasterInit();

    while (1) {
        // Передаємо дані 0x42
        SPI_Transmit(0x42);
        _delay_ms(1000);
    }

    return 0;
}
```

Рисунок 2.13 - Приклад ініціалізації SPI в режимі Master і передачі байта даних

Приклад ініціалізації SPI в режимі Slave і прийому байта даних див. рисунок 2.14:

```
int main(void) {
    // Ініціалізуємо SPI як Slave
    SPI_SlaveInit();

    while (1) {
        // Приймаємо дані
        uint8_t data = SPI_Receive();
        // Обробляємо отримані дані (наприклад, зберігаємо чи передаємо далі)
    }

    return 0;
}
```

Рисунок 2.14 - Приклад ініціалізації SPI в режимі Slave і прийому байта даних

Повна программа:

Для Master мікроконтролера див. рисунок 2.15.

```
#include <stdint.h>
#include <avr/io.h>
#include <util/delay.h>

void SPI_init() {
    // Set MOSI, SCK, and SS as output
    DDRB |= (1<<PB3) | (1<<PB5) | (1<<PB2);
    // Enable SPI, set as master, and set clock rate
    SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
}

void SPI_transfer(uint8_t data) {
    // Start transmission
    SPDR = data;
    // Wait for transmission complete
    while(!(SPSR & (1<<SPIF)));
}

int main(void) {
    // Initialize SPI
    SPI_init();
    // Set SS pin as output
    DDRB |= (1<<PB2);

    while(1) {
        // Activate connection with slave
        PORTB &= ~(1<<PB2);
        // Send byte of data
        SPI_transfer(0x42);
        // Deactivate connection with slave
        PORTB |= (1<<PB2);
        // Delay
        _delay_ms(1000);
    }

    return 0;
}
```

Рисунок 2.15 – Налаштування SPI майстра

Для Slave мікроконтролера див. рисунок 2.16.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

volatile uint8_t receivedData;

void SPI_init() {
    // Set MISO as output
    DDRB |= (1<<PB4);
    // Enable SPI and SPI interrupt, set as slave
    SPCR = (1<<SPE) | (1<<SPIE);
}

ISR(SPI_STC_vect) {
    // Receive data
    receivedData = SPDR;
    // Optionally handle the received data here
}

int main(void) {
    // Initialize SPI as slave
    SPI_init();
    // Set SS pin as input
    DDRB &= ~(1<<PB2);

    // Enable global interrupts
    sei();

    while (1) {
        // Main loop remains empty
    }

    return 0;
}
```

Рисунок 2.16 – Налаштування SPI slave

Для відслідковування усього процесу підключимо SPI DEBUGGER рисунок 2.17

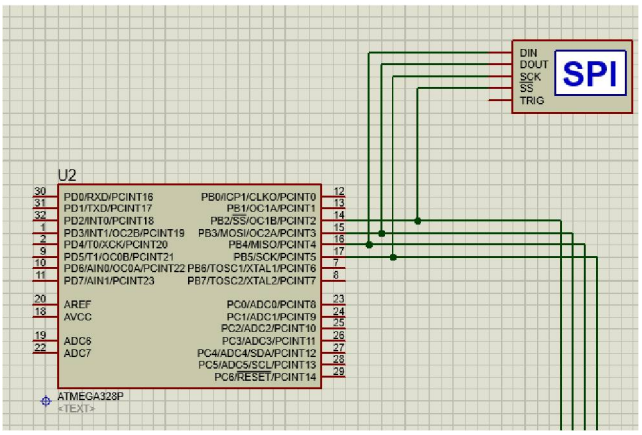

Рисунок 2.17 – Підключення SPI DEBUGGER

Запуск симуляції виконуємо натискаючи на синій трикутничок рисунок 2.18

Рисунок 2.18 – запуск симуляції

Вигляд працюючої симуляції рисунок 2.19

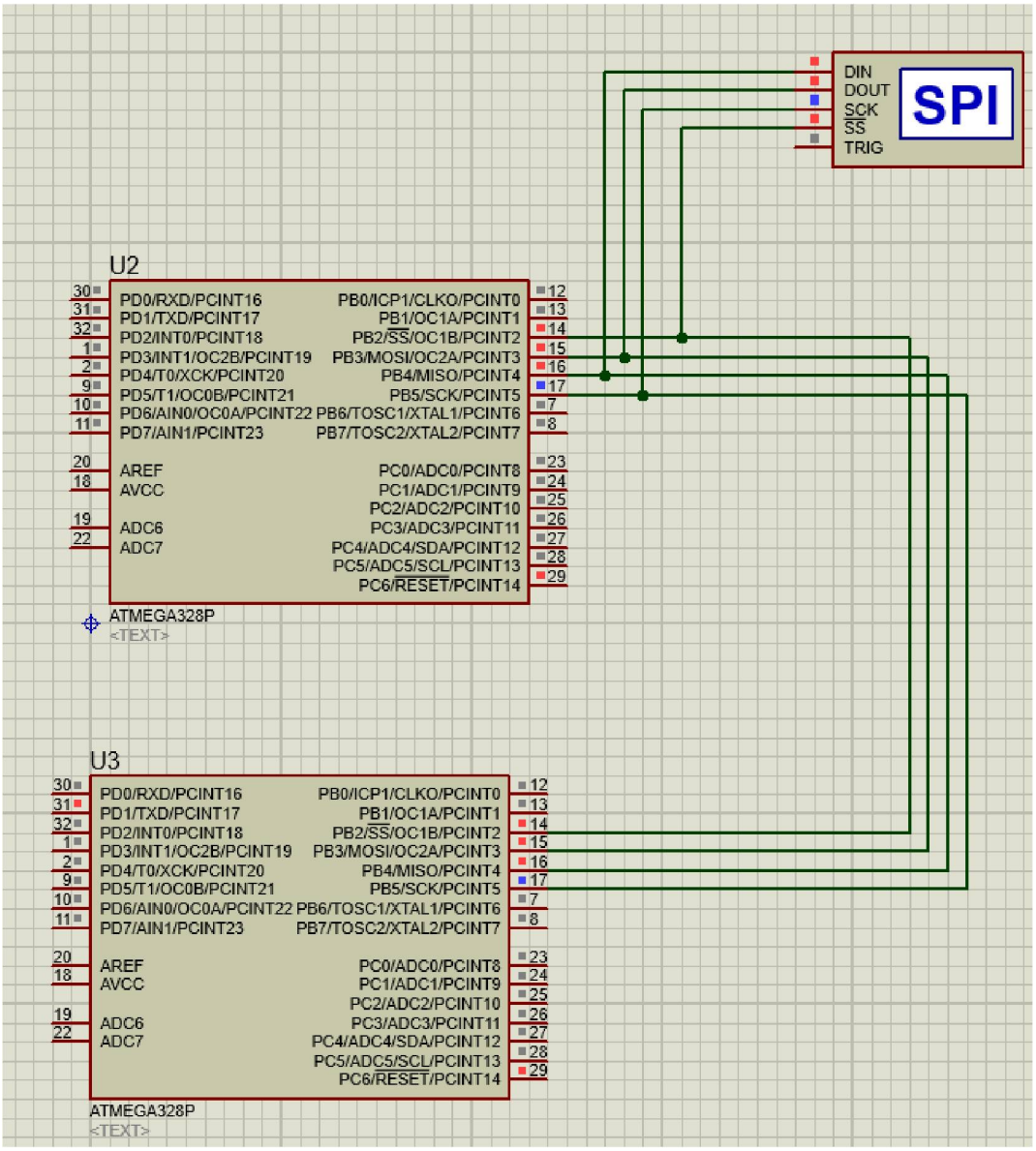

Рисунок 2.19 – симуляція

Вигляд SPI DUBUGGER де ми можемо пабачити наші данні які передаються по SPI рисунок 2.20

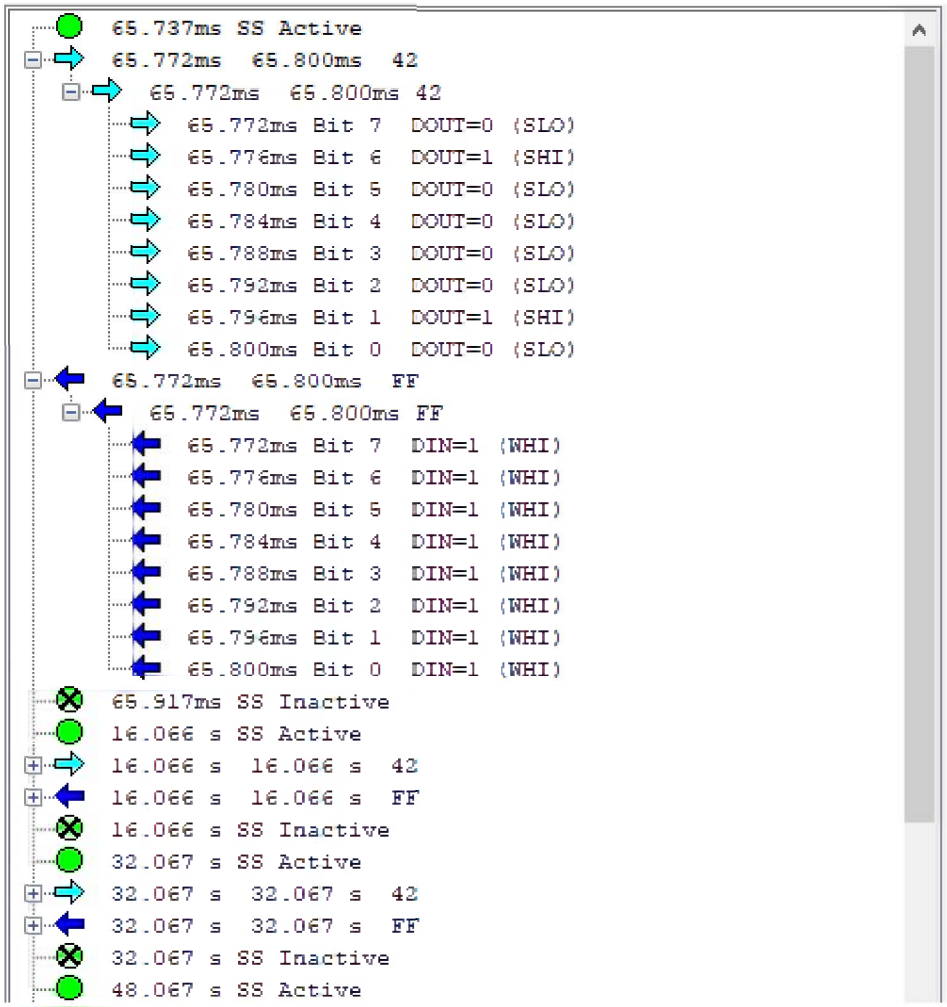

Рисунок 2.20 – SPI DEBUGGER

За отриманими результатами бачимо що симуляція успішно працює.

2.1.4 Побудова SPI з’єднання на реальних пристроях Arduino UNO які керуються мікроконтролером ATmega328p

Піни SPI на Arduino Uno

1. **MISO (Master In Slave Out)**:
   - Пін: **12**

- Функція: Прийом даних від Slave на Master.

2. **MOSI (Master Out Slave In)**:
   - Пін: **11**
   - Функція: Передача даних від Master до Slave.
3. **SCK (Serial Clock)**:
   - Пін: **13**
   - Функція: Передача тактового сигналу від Master до Slave.
4. **SS (Slave Select)**:
   - Пін: **10**
   - Функція: Вибір Slave, з яким Master буде спілкуватися. Цей пін використовується для вибору конкретного пристрою на SPI шині, якщо підключено декілька Slave.

Приклад з’єднання двох Arduino Uno по SPI у програмі **Fritzing** рисунок 2.21

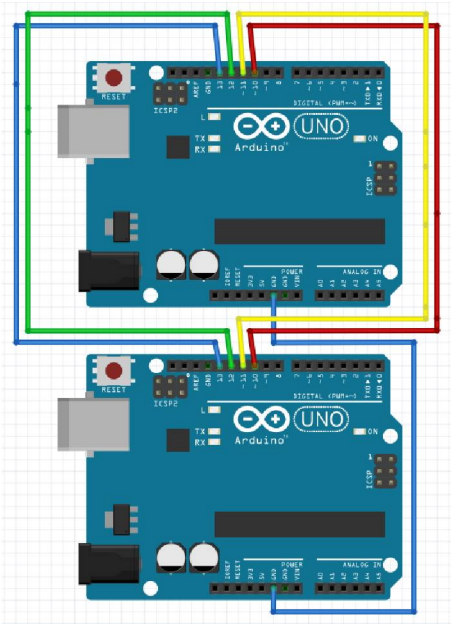

Рисунок 2.21 – з’єднання двох Arduino UNO по SPI

Приклад з’єднання на реальних пристроях рисунок 2.22

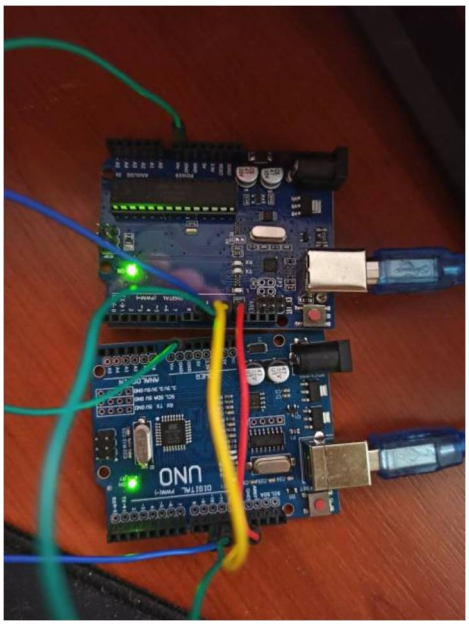

Рисунок 2.22 – З’єднання двох Arduino UNO

2.2 Підключення зсувового регістру 74HC595 по SPI.

2.2.1 Схема зсувого регістру 74HC595

Розглянемо схему зсувового регістру 74HC595 рисунок 2.23

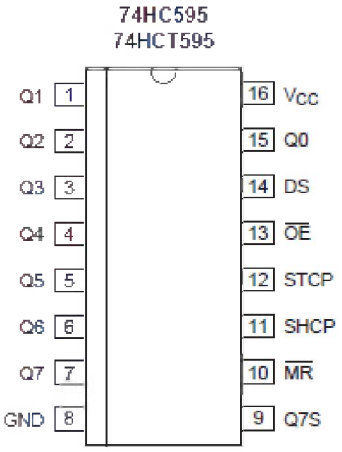

Рисунок. 2.23 – схема сдвигового регістру 74HC595

Схема по якій ми будемо підключати виглядає так див. Таблиця 2.1 та рисунок 2.24:

Таблиця 2.1

Схема відповідності підключення

Arduino	74HC595
GND	GND
5V	VCC
D11	SER (DS)
D13	SRCLK (SH_CP)
D10	RCLK (ST_CP)
GND	OE
5v	MR
Q0 – Q7	Керовані пристрої

2.2.2 Підключення зсувового регістру 74HC595 до Arduino

1. **Q0-Q7** - виходи, до яких підключаються керовані пристрої (LED, реле тощо).
2. **GND** - до GND Arduino.
3. **VCC** - до 5V Arduino.
4. **SER (DS)** - до MOSI (D11 на Arduino).
5. **SRCLK (SH_CP)** - до SCK (D13 на Arduino).

6. **RCLK (ST_CP)** - до будь-якого цифрового виходу Arduino (наприклад, D10).
7. **OE (Output Enable)** - до GND.
8. **MR (Master Reset)** - до VCC.

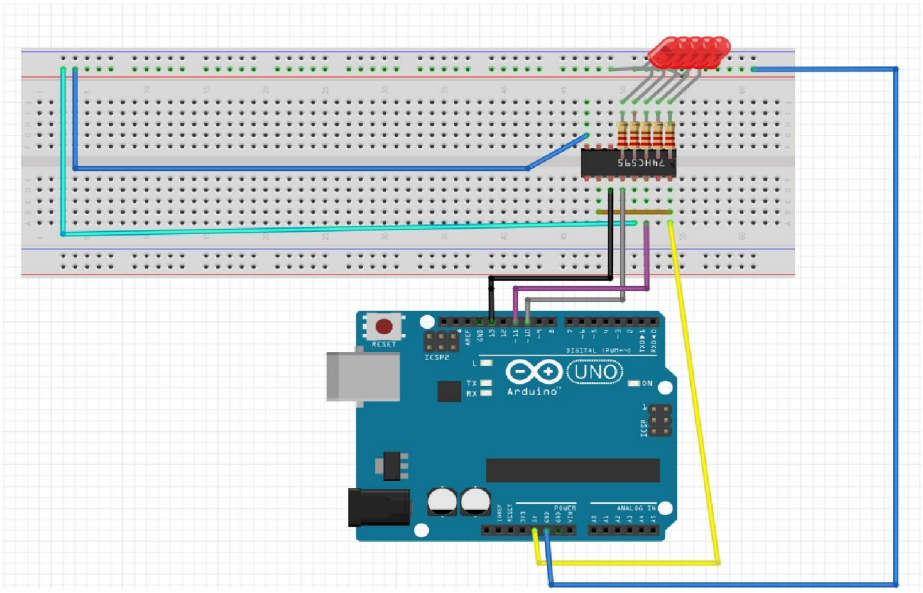

Рисунок 2.24 – Схема підключення зсувового регістру 74HC595 до Arduino UNO

Приклад коду для управління зсувовим регістром 74HC595 рисунок 2.25

```
#include <avr/io.h>
#include <util/delay.h>

#define LATCH_PIN PB2

void SPI_init(void) {
    // Set MOSI, SCK and SS as output, others as input
    DDRB |= (1<<PB3) | (1<<PB5) | (1<<LATCH_PIN);
    // Enable SPI, set as Master, and set clock rate fck/16
    SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
}

void SPI_send(uint8_t data) {
    // Start transmission
    SPDR = data;
    // Wait for transmission complete
    while(!(SPSR & (1<<SPIF)));
}

void shiftOut(uint8_t data) {
    // Latch low
    PORTB &= ~(1<<LATCH_PIN);
    // Send data via SPI
    SPI_send(data);
    // Latch high
    PORTB |= (1<<LATCH_PIN);
}

int main(void) {
    // Initialize SPI
    SPI_init();
    // Set LATCH_PIN as output
    DDRB |= (1<<LATCH_PIN);

    while (1) {
        // Example: LED shift
        for (uint8_t i = 0; i < 256; i++) {
            shiftOut(i);
            _delay_ms(500); // Delay for visualization
        }
    }

    return 0;
}
```

Рисунок 2.25 - Приклад коду для управління зсувовим регістром 74HC595

Демонсстрація роботи схеми див. рисунок 2.26

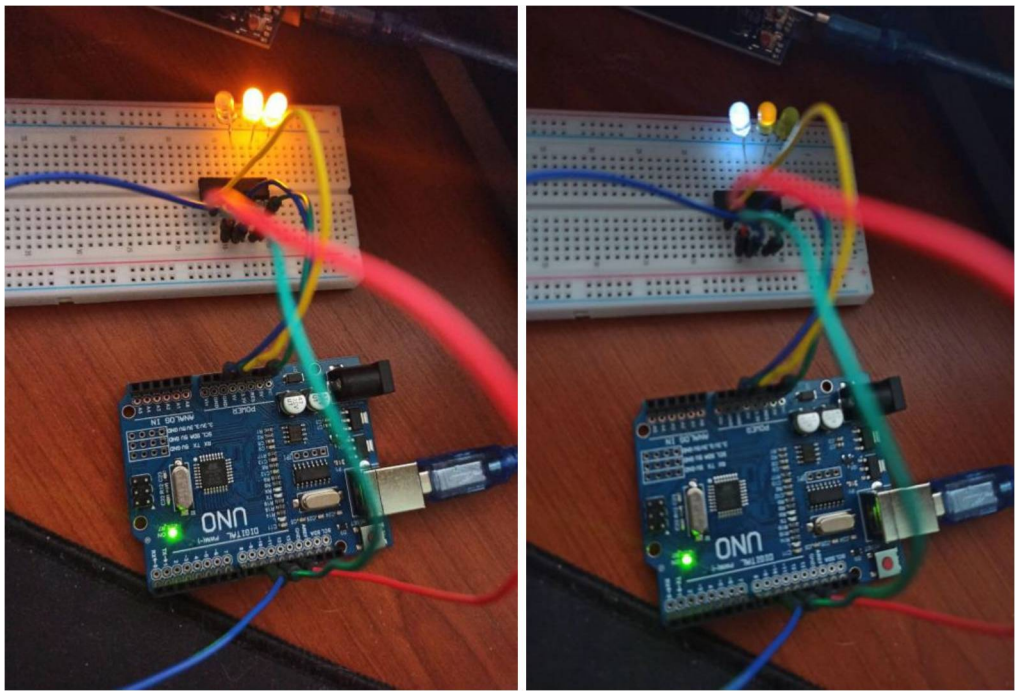

Рисунок 2.26 – демонстрація роботи схеми

Висновки за розділом 2

У цьому розділі було розглянуто процес створення симуляції мікроконтролерного керування електронним навантаженням за допомогою протоколу SPI. Симуляція дозволяє верифікувати коректність реалізації керування та оптимізувати налаштування системи до безпосереднього впровадження в апаратному середовищі.

1. **Вибір платформи та інструментів:**
    - Для створення симуляції було обрано мікроконтролери ATmega328 та ATmega2560, які широко використовуються у вбудованих системах і мають апаратну підтримку протоколу SPI.
    - Основними інструментами для симуляції є програмне середовище Arduino IDE та Proteus Design Suite, що дозволяють моделювати роботу мікроконтролерів та периферійних пристроїв.
2. **Створення симуляції в Proteus:**

- Симуляція в Proteus дозволяє моделювати роботу мікроконтролерів та периферійних пристроїв, забезпечуючи візуалізацію процесу передачі даних по SPI.
- Симуляція успішно працює, що підтверджується результатами, отриманими з SPI Debugger.

3. **Побудова SPI з'єднання на реальних пристроях:**
   - Підключення мікроконтролерів на реальних пристроях Arduino UNO, що керуються мікроконтролером ATmega328p.
   - Схема підключення зсувового регістру 74HC595 до Arduino дозволяє керувати виходами регістру за допомогою SPI.
4. **Ефективність та надійність реалізованої системи:**
   - Симуляція та реальне тестування підтвердили коректність та ефективність керування електронним навантаженням за допомогою протоколу SPI.
   - Реалізована система забезпечує швидку та надійну передачу даних, що дозволяє ефективно керувати електронними навантаженнями.

РОЗДІЛ 3.
АНАЛІЗ ОТРИМАНИХ ДАНИХ, ВИЯВЛЕННЯ ТЕНДЕНЦІЙ ТА ЕФЕКТИВНОСТІ МЕТОДІВ

3.1 Методика тестування та характеристики, які будуть порівнюватись

Для оцінки ефективності керування електронним навантаженням за допомогою протоколу SPI були визначені наступні характеристики:

1. Час відгуку системи: час, який проходить від моменту подання команди на зміну стану електронного навантаження до моменту фактичного виконання цієї команди. Він включає час, необхідний для передачі даних через SPI, обробки цієї інформації мікроконтролером та здійснення фізичної зміни стану навантаження (наприклад, увімкнення чи вимкнення світлодіода).

2. Точність керування: Відхилення від заданих параметрів (струм, напруга).

3. Енергоспоживання системи при різних режимах роботи.

4. Визначення параметрів для максимальної ефективності управління в залежності від типу навантаження

5. Швидкість передачі даних: Максимальна швидкість, досягнута при передачі даних через SPI.

3.2 Проведення тестування та висвітлення результатів

Для проведення тестування були використані наступні методи:

- **Лабораторні експерименти:** Вимірювання часу відгуку та точності керування при різних налаштуваннях електронного навантаження.

- **Симуляції:** Використання програмних симуляцій для оцінки енергоспоживання та стабільності роботи системи.
- **Аналіз логів:** Збір та аналіз логів для оцінки швидкості передачі даних та стабільності системи.

Результати тестування:

1. **Час відгуку системи рисунок 3.1:**

Дільник частоти SPI	Частота SPI	Середній час відгуку (мікросекунди)	Висновок
DIV4	4 МГц	15 мкс	Найшвидший час відгуку, але може бути нестабільним на довгих проводах або при перешкодах.
DIV16	1 МГц	20 мкс	Збалансована швидкість і стабільність.
DIV64	250 кГц	44 мкс	Найбільш стабільний, але найповільніший час відгуку.

Рисунок 3.1 - Час відгуку системи при різних частотах

При частоті SPI в 4 МГц, тривалість одного такту становить 1/4 МГц = 0.25 мкс. Для передачі одного байта даних (8 бітів) потрібно 8 тактів, що дає 0.25 мкс * 8 = 2 мкс для передачі одного байта.

При частоті SPI в 1 МГц, тривалість одного такту становить 1/1 МГц = 1 мкс. Для передачі одного байта 1 мкс * 8 = 8 мкс.

Однак, якщо загальний час відгуку для 4 МГц становить 15 мкс, а для 1 МГц – 20 мкс, то це означає, що основна частина часу відгуку припадає на інші компоненти системи, такі як обробка команд мікроконтролера і програмні затримки, а не лише на час передачі даних.

2. **Точність керування:**

(очікувана вхідна напруга для Arduino UNO 5 В)

- Вхідна напруга 4.95 (В) в межах норми див. рисунок 3.2 допустиме відхилення +-5 %.

Рисунок 3.2 – вхідна напруга на реальному пристрої.

(очікувана вихідна напруга 2.0 – 3.2 В)

- Вихідна напруга 3.02 (В) в межах норми див. рисунок 3.3

Рисунок 3.3 – вихідна напруга на реальному пристрої.

3. **Енергоспоживання:**

   виміряний струм

   Arduino UNO без навантаження – 0.1 А див. рисунок 3.4

Рисунок 3.4 – Arduino UNO без навантаження, з одним світлодіодом та с двума світлодіодами

4. **Визначення параметрів для максимальної ефективності управління в залежності від типу навантаження див. рисунок 3.5**

Параметри Для Максимальної Ефективності Управління

	Тип навантаження	Напруга живлення	Опір/Індуктивність/Ємність	Частота PWM	Струм (А)	Потужність (Вт)	Час відгуку	Точність керування	Вплив SPI
1	Аналогові навантаження	5 В	100 Ом	-	0.05	0.25	50 мкс	Висока	Високошвидкісна передача даних для швидкого та точного вимірювання.
2	Індуктивні навантаження	5 В	10 мГн	1 кГц	0.5	2.5	200 мкс	Середня	Швидке перемикання, зменшення втрат, точне налаштування
3	Ємкісні навантаження	5 В	100 мкФ	1 кГц	0.5	2.5	100 мкс	Висока	Стабільність частоти PWM, зниження втрат, висока швидкість передачі даних.

Рисунок 3.5 - Визначення параметрів для максимальної ефективності управління

5. **Швидкість передачі даних при різних частотах SCK див. рисунки (3.6, 3.7, 3.8, 3.9, 3.10, 3.11):**

    - Максимальна швидкість передачі даних через SPI: 20 Мбіт/с.

	Частота (Hz)	Передано байтів	Час передачі (мікросекунди)
1	125000	10	660
2	125000	50	3280
3	125000	100	6560
4	125000	1000	65540
5	125000	2000	131080
6	125000	5000	327688
7	125000	10000	655368
8	125000	15000	983044
9	125000	20000	1310724
10	125000	25000	1638408
11	125000	30000	1966088

Рисунок. 3.6 - Швидкість передачі даних при частоті 125000 Hz

12	500000	10	180
13	500000	50	884
14	500000	100	1756
15	500000	1000	17508
16	500000	2000	35024
17	500000	5000	87568
18	500000	10000	175124
19	500000	15000	262696
20	500000	20000	350256
21	500000	25000	437824
22	500000	30000	525388

Рисунок 3.7 - Швидкість передачі даних при частоті 500000 Hz

23	1000000	10	100
24	1000000	50	484
25	1000000	100	960
26	1000000	1000	9544
27	1000000	2000	19072
28	1000000	5000	47676
29	1000000	10000	95340
30	1000000	15000	143028
31	1000000	20000	190692
32	1000000	25000	238360
33	1000000	30000	286028

Рисунок 3.8 - Швидкість передачі даних при частоті 1000000 Hz

34	2000000	10	60
35	2000000	50	280
36	2000000	100	560
37	2000000	1000	5532
38	2000000	2000	11056
39	2000000	5000	27648
40	2000000	10000	55296
41	2000000	15000	82952
42	2000000	20000	110592
43	2000000	25000	138240
44	2000000	30000	165900

Рисунок 3.9 - Швидкість передачі даних при частоті 2000000 Hz

45	4000000	10	40
46	4000000	50	180
47	4000000	100	352
48	4000000	1000	3524
49	4000000	2000	7048
50	4000000	5000	17612
51	4000000	10000	35224
52	4000000	15000	52840
53	4000000	20000	70452
54	4000000	25000	88064
55	4000000	30000	105672

Рисунок 3.10 - Швидкість передачі даних при частоті 4000000 Hz

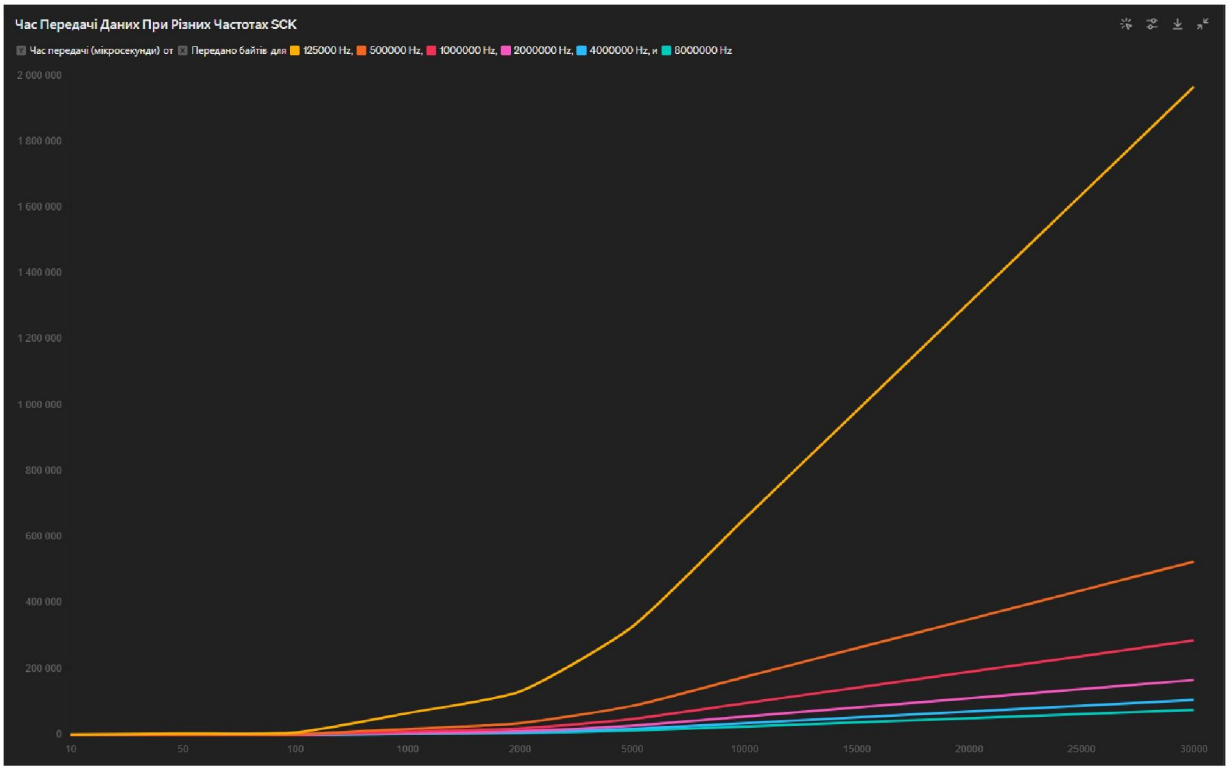

Рисунок 3.11 - Швидкість передачі даних при частоті 8000000 Hz

3.3 Виявлення тенденцій та висновки

На основі проведеного аналізу було виявлено наступні тенденції:

- Час відгуку системи був виміряний для різних частот SPI. Результати показали, що при збільшенні частоти SPI (наприклад, з 1 МГц до 4 МГц) час передачі даних значно скорочується.

- Точність керування була оцінена шляхом вимірювання відхилень від заданих параметрів напруги та струму. Результати показали, що вхідна та вихідна напруга на реальних пристроях знаходиться в межах норми, що свідчить про високу точність керування.

- Час відгуку системи: Показав стабільно низькі значення, що свідчить про високу швидкість обробки запитів і відповідність вимогам реального часу.

Висновки за розділом 3

- **Основні знахідки:** Аналіз показав, що протокол SPI є ефективним для керування електронними навантаженнями, забезпечуючи високу швидкість передачі даних та точність керування.

- **Рекомендації:** Для систем, де необхідна висока точність та швидкість керування, рекомендується використовувати метод PWM. Для великих навантажень краще застосовувати транзисторне керування або твердотільні реле.

- **Майбутні дослідження:** Пропонується дослідити можливості інтеграції штучного інтелекту для адаптивного керування навантаженням, а також розробити методики підвищення енергоефективності систем.

ВИСНОВКИ

В даній кваліфікаційній роботі було проведено аналіз та порівняння різних методів керування електронними навантаженнями за допомогою протоколу SPI. У роботі було досліджено різні підходи до використання SPI в мікроконтролерних системах та проведено аналіз наявних популярних технологічних рішень в області керування електронними навантаженнями.

Було зібрано дані щодо різних характеристик цих підходів, таких як час відгуку системи, точність керування, енергоспоживання та стабільність роботи. На основі отриманих даних була розроблена рекомендаційна модель для вибору методу керування електронними навантаженнями в залежності від конкретних вимог та обмежень проекту.

Виявлено, що кожен з розглянутих методів має свої переваги та недоліки, і вибір оптимального методу залежить від конкретних вимог проекту та його контексту. Зокрема:

Для практичного застосування рекомендаційної моделі важливо враховувати специфіку проекту, його потреби та обмеження. Такий підхід дозволить забезпечити оптимальне використання ресурсів та досягнення поставлених цілей проекту з мінімальними витратами.

Загалом, кваліфікаційна робота показала високу ефективність використання протоколу SPI для керування електронними навантаженнями в мікроконтролерних системах. Впроваджені методи дозволяють підвищити надійність та продуктивність системи, забезпечуючи високу точність та стабільність керування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adafruit Learning System. Overview | SPI Wiring & Test | Adafruit Learning System. URL: https://learn.adafruit.com/ (дата звернення – 24.04.2024).

2. Arduino Workshop: A Hands-On Introduction with 65 Projects / John Boxall. San Francisco: No Starch Press, 2013. 392 с.

3. Designing SOCs with Configured Cores: Unleashing the Tensilica Xtensa and Diamond Cores / Steve Leibson. Amsterdam: Elsevier, 2007. 392 с.

4. Embedded Systems: Introduction to the MSP432 Microcontroller, Volume 1 / Jonathan W. Valvano. 6th ed. Austin: CreateSpace Independent Publishing Platform, 2016. 486 с.

5. Interfacing Microchip Serial EEPROMs with SPI Systems / Microchip Technology Inc. (AN991). Chandler: Microchip Technology Inc., 2004. 16 с.

6. M. Ali Mazidi, Sarmad Naimi, Sepehr Naimi. The AVR Microcontroller and Embedded Systems: Using Assembly and C. Harlow: Pearson, 2014. 832 с.

7. Serial Peripheral Interface (SPI) / SparkFun Electronics. URL: https://learn.sparkfun.com/ (дата звернення – 24.04.2024).

8. Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems / Jan Axelson. 2nd ed. Madison: Lakeview Research, 2007. 424 с.

9. 8-bit serial-in, serial or parallel-out shift register with output latches / Nexperia. (Datasheet 74HC595). Nijmegen: Nexperia, 2018. 21 с.

10. 7/8-Bit Single/Dual SPI Digital Potentiometers / Microchip Technology Inc. (Datasheet MCP4131). Chandler: Microchip Technology Inc., 2019. 25 с.

ДОДАТКИ

Додаток А

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

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **бакалавр**
галузь знань:**15 – Автоматизація та приладобудування**
спеціальність:**151 – Автоматизація та комп'ютерно-інтегровані технології**

ЗАТВЕРДЖУЮ
Завідувач кафедри
теоретичної
та прикладної системотехніки
д.т.н., проф. Шматков С. І.
«21» грудня 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Шульги Руслана Володимировича

1. Тема роботи «**Метод керування електронним навантаженням за протоколом SPI**»

керівник роботи Котвицький Альберт Тадеушевич, к.ф.-м.н, доцент каф.ТПС
затверджені наказом по університету від «03» травня 2024року № 4101-5/909
2.Строк подання студентом роботи 31 травня 2024року
3. Перелік питань, які потрібно розробити
1) Що таке SPI (Serial Peripheral Interface) та його характеристики?
2) Які існують методики керування електронними навантаженнями?
3) Як SPI протокол використовується для керування електронними навантаженнями?
4) Апаратна підтримка протоколу SPI у мікроконтролерах ATmega328 та ATmega2560.
5) Аналіз та вибір існуючих пристроїв, які керуються по SPI.
6) Створення проекту мікроконтролерного керування навантаженням за допомогою SPI пристрою.
7) Які переваги та недоліки має керування електронним навантаженням за допомогою SPI?
8) Які можливі покращення та оптимізації можна запропонувати для методики керування?

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Вступ: Визначення проблеми, цілей та значення дослідження.	21.12.2023 - 25.01.2024
2	Огляд літератури: Аналіз наукових публікацій з теми SPI та електронних навантажень.	19.12.2023 - 2.01.2024
3	Налаштування режимів роботи SPI протоколу в мікроконтролерах ATmega328 та ATmega2560.	2.01.2024 - 2.02.2024
4	Дослідження та аналіз: Вивчення механізмів керування електронними навантаженнями за протоколом SPI.	2.01.2024 - 2.02.2024
5	Створення симуляції мікроконтролерного керування за протоколом SPI	2.01.2024 - 2.02.2024
5	Обговорення результатів: Аналіз отриманих даних, виявлення тенденцій та ефективності методів.	3.02.2024 - 30.03.2024
6	Висновки: Формулювання загальних висновків та рекомендацій за дослідженням.	3.03.2024 - 30.04.2024
7	Рекомендації: Пропозиції щодо покращення методик керування та оптимізації процесів.	31.03.2024 - 27.05.2024
8	Підготовка до захисту: Оформлення результатів дослідження для презентації та захисту.	22.05.2024 – 05.06.2024

5. Дата видачі завдання 21.12.2023

Студент Р.В. Шульга _____
ініціали, прізвище підпис

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

Додаток Б

Технічне завдання
на розробку програмного виробу
«Система керування електронним навантаженням за протоколом SPI»

Назва розділу	Назва і зміст підрозділу
1. Введення	1.1. Назва програмного виробу – Метод керування електронним навантаженням за протоколом SPI. 1.2. Галузь застосування – управління проектами, IT-індустрія і розробка ПЗ.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології. 2.2. Завдання на дипломну роботу бакалавра, затверджено наказом ХНУ імені В. Н. Каразіна № 4101-5/909 від «03» травня 2024 р. (представит як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3. Призначення розробки	3.1. Мета розробки - дослідження та розробка методики керування електронним навантаженням за допомогою SPI протоколу з використанням мікроконтролерів ATmega328 і ATmega2560. 3.2. Застосування розробки - програмна і апаратна реалізація керування електронними навантаженнями в індустріальних та наукових дослідженнях для покращення точності та ефективності протоколу SPI. 3.3. Початкові дані для розробки: • Аналіз наукових публікацій з теми SPI та електронних навантажень. • Вивчення апаратної підтримки протоколу SPI в мікроконтролерах ATmega328 і ATmega2560. • Розробка та тестування пристроїв, які керуються за допомогою SPI.
4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: • Програма повинна мати можливість виконувати налаштування та управління режимами роботи SPI в мікроконтролерах ATmega328 та ATmega2560. • Необхідно розробити симуляцію мікроконтролерного керування за протоколом SPI. • Програма має надавати детальний аналіз і порівняння існуючих методів керування електронними навантаженнями. 4.2. Вимоги до надійності: • Розроблені методики мають бути стабільними та повторюваними при однакових умовах тестування. 4.3. Вимоги до умов експлуатації:

	• Програма має бути адаптована для використання в лабораторних умовах, що включає стійкість до високих навантажень та тривалого використання. 4.4. Вимоги до складу і параметрів технічних засобів: • Система має бути сумісна з персональними комп'ютерами та лабораторним обладнанням, що використовується для налаштування та тестування мікроконтролерів. 4.5. Вимоги до інформаційної та програмної сумісності: • Забезпечити можливість інтеграції з існуючими програмними та апаратними рішеннями, які вже використовуються в університеті та наукових лабораторіях. 4.6. Вимоги до маркування та упаковки: • Не потрібно спеціального маркування або упаковки. 4.7. Вимоги до транспортування і зберігання: • Вимоги відсутні, оскільки продукт буде розповсюджуватись в цифровій формі. 4.8. Спеціальні вимоги: • Не пред'являються.
5. Вимоги до програмної документації.	Програмною документацією до виробу «Система автоматизації розробки програмного забезпечення на основі використання архітектурних підходів» вважати: 1) Справжнє Технічне завдання на розробку програмного вироби (представити у вигляді Додатку Б до пояснювальної записки до дипломної роботи). 2) Програму і методику випробувань розробленого програмного вироби (представити у вигляді Додатку В до пояснювальної записки до дипломної роботи). 3) Опис програмного вироби (представити в розділі 3 пояснювальної записки до дипломної роботи). 4) Текст програми (представити в Додатку Г до пояснювальної записки до дипломної роботи).
6. Техніко-економічні показники	Вимоги до розрахунку техніко-економічних показників не потрібні.

<table>
<tr>
<td>7. Стадії і етапи розробки</td>
<td>
<table>
<tr><th>№ з/п</th><th>Назви етапів роботи</th><th>Термін виконання етапів роботи</th></tr>
<tr><td>1</td><td>Вступ: Визначення проблеми, цілей та значення дослідження.</td><td>21.12.2023 - 25.01.2024</td></tr>
<tr><td>2</td><td>Огляд літератури: Аналіз наукових публікацій з теми SPI та електронних навантажень.</td><td>19.12.2023 - 2.01.2024</td></tr>
<tr><td>3</td><td>Налаштування режимів роботи SPI протоколу в мікроконтролерах ATmega328 та ATmega2560.</td><td>2.01.2024 - 2.02.2024</td></tr>
<tr><td>4</td><td>Дослідження та аналіз: Вивчення механізмів керування електронними навантаженнями за протоколом SPI.</td><td>2.01.2024 - 2.02.2024</td></tr>
<tr><td>5</td><td>Створення симуляції мікроконтролерного керування за протоколом SPI</td><td>2.01.2024 - 2.02.2024</td></tr>
<tr><td>5</td><td>Обговорення результатів: Аналіз отриманих даних, виявлення тенденцій та ефективності методів.</td><td>3.02.2024 - 30.03.2024</td></tr>
<tr><td>6</td><td>Висновки: Формулювання загальних висновків та рекомендацій за дослідженням.</td><td>3.03.2024 - 30.04.2024</td></tr>
<tr><td>7</td><td>Рекомендації: Пропозиції щодо покращення методик керування та оптимізації процесів.</td><td>31.03.2024 - 27.05.2024</td></tr>
<tr><td>8</td><td>Підготовка до захисту: Оформлення результатів дослідження для презентації та захисту.</td><td>22.05.2024 – 05.06.2024</td></tr>
</table>
</td>
</tr>
<tr>
<td>8. Порядок контролю і приймання</td>
<td>1) Перевірку ходу розробки програмного виробу керівнику робіт виконувати раз в 2 тижні.

2) Випробування програмного продукту провести відповідно до програми та методики випробувань на базі комп’ютерного класу.

3) Захист розробленої моделі провести на засіданні Атестаційної комісії.

4) Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику на CD R компакт-диску.</td>
</tr>
</table>

Виконавець
студент групи КУ-41
Шульга Р.В.
_____

Замовник
доцент кафедри ТПС
Котвицький А.Т
_____

Додаток В

Програма і методика випробувань
програмного вироби
«Метод керування електронним навантаженням за протоколом SPI»

1. **Об'єкт випробувань**

1.1. Найменування випробуваного програмного виробу: "Система керування електронним навантаженням за протоколом SPI".

1.2. Область застосування: автоматизація промислових процесів, контроль і моніторинг електронних систем.

2. **Мета випробувань**

Розробка та оптимізація системи керування для покращення ефективності використання електронних навантажень за допомогою протоколу SPI.

3. **Загальні положення**

3.1. Підстави для проведення випробувань: Наказ ректора університету про проведення наукових випробувань.

3.2. Місце і тривалість випробувань: Випробування проводяться в лабораторії кафедри протягом одного академічного семестру.

3.3. Обсяг випробувань: Повне тестування всіх функцій програми згідно з вимогами Технічного завдання.

3.4. Організації, які беруть участь у випробуваннях: Кафедра, наукові керівники, студенти та інші зацікавлені особи.4. Вимоги до програми або програмного виробу

4. Вимоги до програми або програмного виробу

4.1. Вимоги до функціональних характеристик:

- Забезпечення налаштування параметрів SPI.
- Моніторинг стану електронних навантажень.

4.2. Вимоги до надійності: Програма має забезпечити стабільну роботу при тривалому використанні.

4.3. Вимоги до умов експлуатації: Лабораторні умови.

4.4. Вимоги до складу і параметрів технічних засобів: Необхідне обладнання для налаштування і тестування мікроконтролерів.

4.5. Вимоги до інформаційної та програмної сумісності: Сумісність з використовуваними мікроконтролерами та вторинним обладнанням.

4.6. Вимоги до маркування та упаковки: Не потребує спеціального маркування.

4.7. Вимоги до транспортування і зберігання: Не застосовуються.

4.8. Спеціальні вимоги: Не висуваються.

5. Вимоги до програмної документації

Склад програмної документації, що подається на випробування, включає:

1) Технічне завдання на розробку програмного вироби (представлено в Додатку Б до пояснювальної записки до дипломної роботи).

2) Ця Програма і методика випробувань розробленого програмного вироби (представлена в Додатку В до пояснювальної записки до дипломної роботи).

3) Опис програмного вироби (представлено в розділі 3 пояснювальної записки до дипломної роботи).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Персональні комп'ютери, мікроконтролери.

6.2 Порядок проведення випробувань

6.2.1. Перевірка програмної документації.

Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в технічному завданні документації.

6.2.2. Перевірка якості програмної документації.

Перевірку здійснювати за критерієм відповідності вимогам єдиної системи програмної документації (ЄСПД).

6.2.3. Перевірка виконання програми.

Тест 1: Побудова симуляції мікроконтролерного керування за протоколом SPI з використанням мікроконтролерів ATmega328 та його працездатність (рисунки В.1, В.2, В.3).

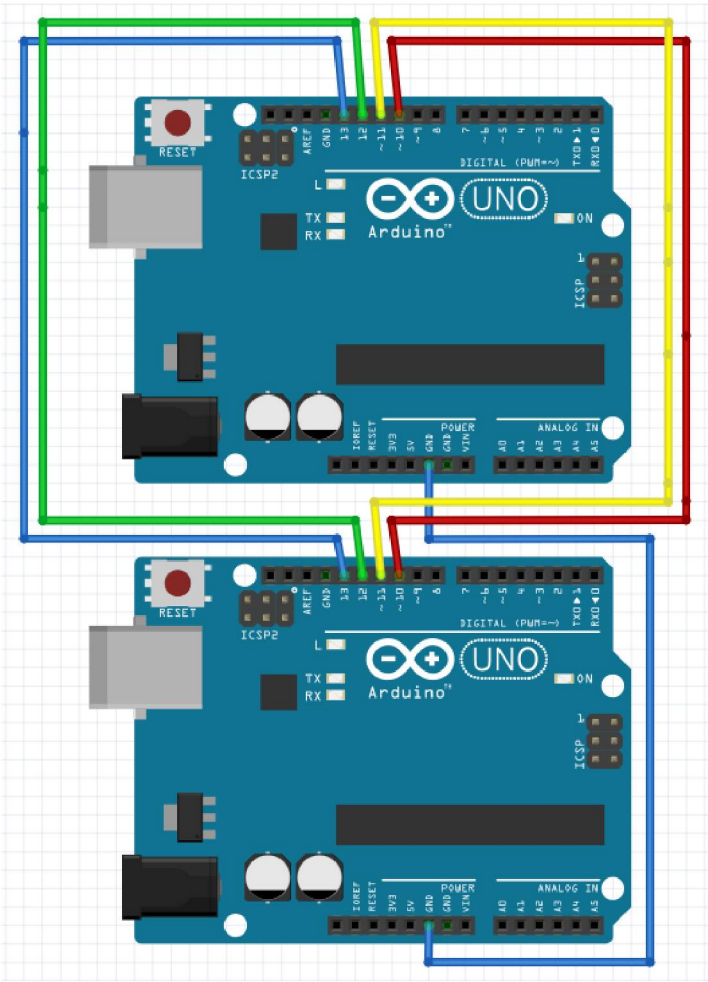

Рисунок. В.1 Тест 1

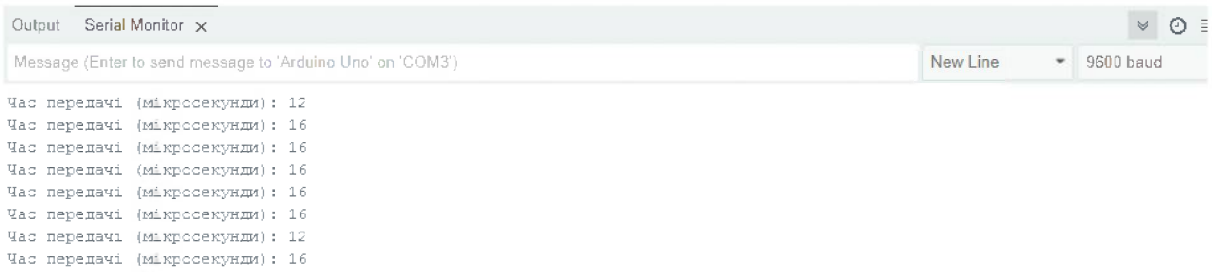

Рисунок. В.2 Тест 1

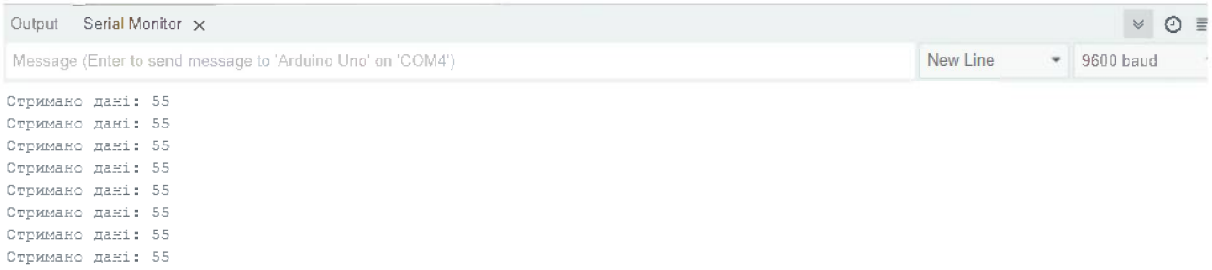

Рисунок. В.3 Тест 1

Висновок: : Перевірка створення симуляції мікроконтролерного керування за протоколом SPI з використанням мікроконтролерів ATmega328 пройшла успішно.

Тест 2: Перевірка створення Сі-програми для симуляції мікроконтролерного керування за протоколом SPI з використанням мікроконтролерів ATmega328 (рисунки В.4, В.5).

```c
#include <avr/io.h>
#include <util/delay.h>

#define SS_PIN PB2
#define F_CPU 16000000UL

void SPI_init(void) {
    // Set MOSI, SCK, and SS as Output
    DDRB |= (1<<PB3) | (1<<PB5) | (1<<SS_PIN);
    // Enable SPI, Set as Master, Set clock rate fck/16
    SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
    // Ensure SS is high
    PORTB |= (1<<SS_PIN);
}

uint8_t SPI_transfer(uint8_t data) {
    // Start transmission
    SPDR = data;
    // Wait for transmission complete
    while(!(SPSR & (1<<SPIF)));
    // Return received data
    return SPDR;
}

unsigned long micros(void) {
    return (unsigned long)(TCNT1);
}

void timer1_init() {
    // Set timer1 normal mode
    TCCR1A = 0;
    TCCR1B = (1 << CS10); // No prescaling
    TCNT1 = 0; // Initialize counter
}

int main(void) {
    // Initialize SPI
    SPI_init();
    // Initialize timer1
    timer1_init();

    uint8_t dataToSend = 0x55;
    uint8_t receivedData;

    while (1) {
        unsigned long startTime = micros();

        // Activate slave device
        PORTB &= ~(1<<SS_PIN);
        // Transfer and receive data
        receivedData = SPI_transfer(dataToSend);
        // Deactivate slave device
        PORTB |= (1<<SS_PIN);

        unsigned long endTime = micros();

        // Here you can handle the result if needed
        // For example, store it or use it in further logic

        // Delay for 1 second
        _delay_ms(1000);
    }

    return 0;
}
```

Рисунок. В.4 Тест 2

```c
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

volatile uint8_t receivedData;
volatile uint8_t dataReceived = 0;

void SPI_init(void) {
    // Set MISO as output
    DDRB |= (1<<PB4);
    // Enable SPI in slave mode
    SPCR = (1<<SPE);
    // Enable SPI interrupt
    SPCR |= (1<<SPIE);
}

ISR(SPI_STC_vect) {
    // Read received data
    receivedData = SPDR;
    // Set data received flag
    dataReceived = 1;
}

int main(void) {
    // Initialize SPI
    SPI_init();

    // Enable global interrupts
    sei();

    while (1) {
        if (dataReceived) {
            // Handle received data (e.g., send it back or process it)
            SPDR = receivedData; // Send back the received data

            // Clear data received flag
            dataReceived = 0;
        }
    }

    return 0;
}
```

Рисунок. В.5 Тест 2

Висновок: Перевірка створення Сі-програми для симуляції мікроконтролерного керування за протоколом SPI з використанням мікроконтролерів ATmega328 пройшла успішно.

Тест 3: Вимірювання швидкості передачі даних для симуляції мікроконтролерного керування за протоколом SPI з використанням мікроконтролерів ATmega328 при різних частотах SCK(Serial Clock) (рисунки В.6, В.7, В.8, В.9, В.10, В.11).

Частота (Гц)	Кількість байтів	Час передачі (мікросекунди)
125000	10	660
125000	50	3280
125000	100	6560
125000	1000	65540
125000	2000	131080
125000	5000	327688
125000	10000	655368
125000	15000	983044
125000	20000	1310724
125000	25000	1638408
125000	30000	1966088

Рисунок. В.6 Тест 3

500000	10	180
500000	50	884
500000	100	1756
500000	1000	17508
500000	2000	35024
500000	5000	87568
500000	10000	175124
500000	15000	262696
500000	20000	350256
500000	25000	437824
500000	30000	525388

Рисунок. В.7 Тест 3

1000000	10	100
1000000	50	484
1000000	100	960
1000000	1000	9544
1000000	2000	19072
1000000	5000	47676
1000000	10000	95340
1000000	15000	143028
1000000	20000	190692
1000000	25000	238360
1000000	30000	286028

Рисунок. В.8 Тест 3

2000000	10	60
2000000	50	280
2000000	100	560
2000000	1000	5532
2000000	2000	11056
2000000	5000	27648
2000000	10000	55296
2000000	15000	82952
2000000	20000	110592
2000000	25000	138240
2000000	30000	165900

Рисунок. В.9 Тест 3

4000000	10	40
4000000	50	180
4000000	100	352
4000000	1000	3524
4000000	2000	7048
4000000	5000	17612
4000000	10000	35224
4000000	15000	52840
4000000	20000	70452
4000000	25000	88064
4000000	30000	105672

Рисунок. В.10 Тест 3

8000000	10	28
8000000	50	132
8000000	100	256
8000000	1000	2516
8000000	2000	5036
8000000	5000	12584
8000000	10000	25172
8000000	15000	37752
8000000	20000	50336
8000000	25000	62916
8000000	30000	75500

Рисунок. В.11 Тест 3

Висновок: Вимірювання швидкості передачі даних для симуляції мікроконтролерного керування за протоколом SPI з використанням мікроконтролерів ATmega328 при різних частотах SCK(Serial Clock) пройшов успішно.

Висновки: при вдалому виконанні всіх 3 тестів випробування розробленого додатку вважаються успішними.

Виконавець

студент групи КУ-41

Шульга Р.В.