Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК Информатика и управление

КАФЕДРА ИУК5 Системы обработки информации

КУРСОВОЙ ПРОЕКТ

НА ТЕМУ:

Разработка системы автоматического управления шаговым двигателем на микроконтроллере

Студент группы ИУК5-72Б _____ Р.В. Ли
(подпись, дата) (И.О. Фамилия)

Руководитель курсового проекта _____ _____
(подпись, дата) (И.О. Фамилия)

Калуга, 2025

Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУК5
_____(_____)
«__» _____2025г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине ***Элементы управления в автоматизированных системах обработки информации и управления***
Студент группы ***ИУК5-72Б Ли Роман Владиславович***
(фамилия, имя, отчество)
Тема курсового проекта ***Разработка системы автоматического управления шаговым двигателем на микроконтроллере***
Направленность КП ***учебная***
Источник тематики ***кафедра ИУК5***

Задание

-Написать программу управления шагового двигателя и светодиодиодов, оповещающих о начале вращения вала двигателя. Время вращения вала двигателя составляет 1 минута. Светодиоды горят до включения двигателя. Светодиоды должны начать выключаться поочередно по мере работы двигателя с задержкой 2 секунды

Оформление курсовой работы

Расчетно-пояснительная записка на листах формата А4.
Перечень графического материала КР (плакаты, схемы, чертежи и т.п.):

Дата выдачи задания «___» _____2025 г.

Руководитель _____. _____
(подпись, дата) (И.О. Фамилия)

Студент _____ _____
(подпись, дата) (И.О. Фамилия)

Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

КАЛЕНДАРНЫЙ ПЛАН
на выполнение курсового проекта

по дисциплине ***Элементы управления в автоматизированных системах обработки информации и управления***

Студент группы ***ИУК5-72Б Ли Роман Владиславович***
(фамилия, имя, отчество)

Тема курсовой работы ***Разработка системы автоматического управления шаговым двигателем на микроконтроллере***

№	Наименование этапов	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Руководитель КП	Куратор
1	Задание на выполнение курсовой работы	1 – 4 недели			
2	Выполнение основных расчетов и эскизов графической части	5 – 7 недели			
3	Выполнение и окончательное оформление графической части и расчетно-пояснительной записки	9 - 10-я недели			
4	Защита курсовой работы	11 - 14-я недели			

Студент _____ Руководитель _____
(подпись, дата) (подпись, дата)

СОДЕРЖАНИЕ

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 6

1.1 Наименование 6

1.2. Основание для разработки 6

1.3. Исполнитель 6

1.4. Цель разработки 6

1.5 Содержание работы 7

1.5.1. Задачи, подлежащие решению 7

1.5.2. Требования к архитектуре системы автоматического управления шаговым двигателем на микроконтроллере 7

1.5.3. Требования к составу системы автоматического управления шаговым двигателем на микроконтроллере 8

1.5.4. Требования к прикладным программам 8

1.5.5. Требования к входным/выходным данным 8

1.5.6. Требования к временным характеристикам 9

1.5.7 Требования к составу технических средств 9

1.6. Этапы разработки 9

1.7 Техническая документация, предъявляемая по окончании работы 9

1.8 Дополнительные условия 9

2. НАУЧНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 10

2.1 Постановка задачи проектирования 10

2.3 Анализ аналогов 13

2.4. Перечень задач, подлежащих решению в процессе разработки 14

2.5. Обоснование выбора инструментов и платформы для разработки 15

3. ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ 16

3.1. Разработка структуры приложения 16

3.2. Разработка алгоритмов обработки информации 19

3.3. Реализация функционирующего приложения 19

3.4. Разработка интерфейса взаимодействия пользователя с системой 20

4. ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 22

4.1. Тестирование и отладка макета рабочей программы 22

4.2. Разработка руководства пользователя и руководства программиста (администратора) 23

ЗАКЛЮЧЕНИЕ 26

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 28

ПРИЛОЖЕНИЕ 29

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

По дисциплине «Элементы управления в автоматизированных системах обработки информации и управления»

1.1 Наименование

«Элементы управления в автоматизированных системах обработки информации и управления»

Система предназначена для управления шагового двигателя и нескольких светодиодов

1.2. Основание для разработки

Разработка ведется в рамках курсового проекта по дисциплине «Элементы управления в автоматизированных системах обработки информации и управления».

Планируется разработать программное обеспечение для управления шаговым двигателем и светодиодов для отображения стадии работы мотора

Заказчик: КФ МГТУ им. Н. Э. Баумана.

1.3. Исполнитель

Ли Р.В. студент КФ МГТУ им. Н. Э. Баумана группы ИУК5-72Б

Ответственный за приемку работы: Комиссия кафедры ИУК5.

Плановый срок начала работы: 1 сентября 2025 года.

Плановый срок окончания работы: 26 декабря 2025 года.

1.4. Цель разработки

Разработать программное обеспечение для управления шагового двигателя и светодиодов, оповещающих о начале вращения вала двигателя. Время вращения вала двигателя составляет 1 минута. Светодиоды должны включаться по очереди с задержкой 5 секунд, по мере прекращения работы двигателя.

1.5 Содержание работы

1.5.1. Задачи, подлежащие решению

1. Проектирование архитектуры программного кода программы
   - Выбор программных средств и библиотек для разработки программного обеспечения
2. Разработка алгоритма работы шагового двигателя и светодиодов
   - Подготовка к работе и установка интегрированной среды разработки Atmel Studio
   - Реализация алгоритма работы шагового двигателя на языке программирования С в среде Atmel Studio
3. Разработка схемотехники
   - Установка интегрированной среды проектирования Proteus
   - Создать схему, отладить программное обеспечения для микроконтроллера в режиме симуляции

1.5.2. Требования к архитектуре системы автоматического управления шаговым двигателем на микроконтроллере

- Уровень аппаратных средств - ATmega8L, драйвер шагового двигателя (A4988/DRV8825), силовые ключи для светодиодов. А также, регистр сдвига 74HC595
- Уровень прикладного ПО - основная программа управления с конечным автоматом состояний
- Уровень интерфейса - Serial Monitor для мониторинга и отладки

1.5.3. Требования к составу системы автоматического управления шаговым двигателем на микроконтроллере

Программное обеспечение должно состоять из следующих модулей:

- Модуль настройки и инициализация необходимых переменных и объектов программы
- Модуль запуска алгоритма работы

1.5.4. Требования к прикладным программам

Основная программа управления:

- Циклическое выполнение с периодом не более 1 мс
- Поддержка неблокирующих временных задержек на основе millis()
- Обработка состояний конечного автомата
- Контроль временных характеристик выполнения операций

1.5.5. Требования к входным/выходным данным

Входные данные:

- Тактовые импульсы - цифровой сигнал на STEP-вход драйвера двигателя
- Сигнал направления - цифровой сигнал на DIR-вход драйвера (HIGH/LOW)
- Временные метки - системное время микроконтроллера (millis())
- Состояние кнопки сброса - цифровой вход для принудительного остановки

Выходные данные:

- STEP pin: импульсы длительностью 1-10 мкс
- DIR pin: установка направления вращения
- ENABLE pin: активация драйвера

1.5.6. Требования к временным характеристикам

- Время вращения двигателя: 60 секунд
- Период включения светодиодов: 5 секунд
- Задержка перед началом индикации: 45 секунд от старта системы

- Длительность цикла: 65 секунд (двигатель + индикация + таймаут)

1.5.7 Требования к составу технических средств

1.6. Этапы разработки

1. Анализ требований и проектирование системы
2. Проектирование архитектуры приложения
3. Разработка программного кода
4. Разработка схемотехники
5. Интеграция и настройка системы автоматического управления шаговым двигателем на микроконтроллере
6. Тестирования и отладка
7. Разработка документации
8. Защита курсовой работы

1.7 Техническая документация, предъявляемая по окончании работы

- Техническое задание
- Расчётно-пояснительная записка
- Графические материалы
- Руководство пользователя
- Исходный код с комментариями
- Презентация проекта

1.8 Дополнительные условия

- Возможность дальнейшего расширения
- Система должна корректно работать при сбоях.

2. НАУЧНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

2.1 Постановка задачи проектирования

Актуальность и обоснование проекта

Современный этап развития технологий характеризуется повсеместной автоматизацией и роботизацией процессов в промышленности, медицине, бытовой технике и научных исследованиях. Ключевым элементом большинства автоматизированных систем, требующих точного позиционирования и управления движением, являются шаговые двигатели. Их уникальное свойство — возможность дискретного углового перемещения вала и точного контроля положения без использования обратной связи по положению — делает их незаменимыми в таких устройствах, как станки с ЧПУ, 3D-принтеры, роботизированные манипуляторы, медицинские дозаторы и офисная техника.

Однако эффективность и функциональность шагового двигателя целиком определяются системой управления. Чтобы эффективно использовать шаговый двигатель и обеспечить его пошаговое управление, необходимо применять комплексную систему управления, чувствительную к различным электрическим параметрам. Традиционные подходы, основанные на применении специализированных контроллеров или схем на дискретной логике, обладают рядом существенных недостатков: они малогибки, обладают ограниченным функционалом и сложны в модификации. При изменении характеристик нагрузки (напряжения, силы тока или механических параметров) требуется перепроектирование всей системы управления. В этой связи применение микроконтроллеров (МК) для управления шаговыми двигателями представляется наиболее перспективным направлением, позволяющим преодолеть указанные ограничения.

Использование МК позволяет не только значительно упростить аппаратную часть системы, но и реализовать сложные алгоритмы управления, такие как плавный разгон и торможение, компенсацию резонансных явлений и адаптацию параметров в реальном времени. Кроме того, микроконтроллерная

платформа обеспечивает легкую интеграцию устройства в более сложные системы управления через стандартные интерфейсы связи (UART, I2C, Ethernet), что соответствует принципам Industry 4.0 и Интернета Вещей (IoT).

Таким образом, разработка системы автоматического управления шаговым двигателем на базе микроконтроллера является актуальной задачей, имеющей значительный практический потенциал для применения в различных отраслях науки и техники.

Формулировка задачи

Проектирование и разработка программного обеспечения для управления шаговым двигателем, включает в себя следующие задачи:

- Провести аналитический обзор существующих систем управления шаговыми двигателями и обосновать выбор элементной базы (микроконтроллера, драйвера, типа шагового двигателя).
- Разработать структурную и принципиальную электрическую схему устройства.
- Разработать программное обеспечение для микроконтроллера, включающее:
- Низкоуровневые драйверы для управления драйвером двигателя.
- Собрать действующий макет системы и провести его экспериментальные испытания.

Ограничения и требования

- Аппаратная платформа: Микроконтроллер ATmega8L обеспечивающий генерацию управляющих импульсов с высокой точностью.
- Силовая часть: Драйвер шагового двигателя с поддержкой интерфейса STEP и контролем тока обмоток.

2.2 Описание предметной области

Разработка системы автоматического управления шаговым двигателем на микроконтроллере представляет собой комплексную задачу, находящуюся на стыке нескольких научно-технических дисциплин. Данная предметная область

охватывает вопросы микропроцессорной техники, силовой электроники, теории автоматического управления и мехатроники.

В основе исследования лежит проектирование интеллектуальной системы управления, где микроконтроллер выполняет функции вычислительного ядра, а специализированный драйвер обеспечивает преобразование управляющих сигналов в мощные токовые импульсы для питания обмоток двигателя. Особенностью данной предметной области является необходимость точного согласования цифровых алгоритмов управления с силовыми электронными компонентами и электромеханическими характеристиками двигателя.

Ключевой аспект исследования связан с разработкой алгоритмов управления, обеспечивающих точное позиционирование и плавное движение ротора двигателя. Это требует решения задач по генерации управляющих импульсов с переменной частотой, реализации алгоритмов разгона и торможения, а также компенсации нелинейных эффектов, характерных для шаговых двигателей.

Важное место в предметной области занимает проектирование аппаратной части системы, включающей выбор оптимальной элементной базы, проектирование схем питания и защитных цепей, обеспечение электромагнитной совместимости компонентов. Особое внимание уделяется вопросам теплового режима силовых элементов и энергоэффективности системы в целом.

Программная реализация системы требует разработки многоуровневой архитектуры программного обеспечения, включающей низкоуровневые драйверы для работы с периферией микроконтроллера, алгоритмы управления движением и пользовательский интерфейс для взаимодействия с оператором.

Предметная область характеризуется тесной взаимосвязью теоретических аспектов автоматического управления с практическими задачами схемотехнического проектирования и программирования встроенных систем. Это определяет необходимость применения системного подхода, учитывающего

как функциональные требования к системе управления, так и эксплуатационные характеристики проектируемого устройства.

Таким образом, разработка системы управления шаговым двигателем представляет собой актуальную научно-техническую задачу, требующую комплексного решения вопросов аппаратной и программной реализации, оптимизации алгоритмов управления и обеспечения надежности работы системы в различных режимах эксплуатации.

2.3 Анализ аналогов

Для реализации системы автоматического управления шаговым двигателем были рассмотрены четыре распространённые платформы: ATmega, STM32, PIC и ESP32. Все они позволяют реализовать управление шаговым двигателем, однако отличаются по производительности, периферии, сложности разработки и ориентируются на разные классы задач.

ATmega

Микроконтроллеры ATmega относятся к 8-битному семейству AVR и широко применяются в учебных и любительских проектах, где ключевыми требованиями являются простота освоения и предсказуемость разработки. Архитектура ATmega хорошо документирована, поддерживается большим количеством примеров и библиотек, а также совместима с популярной экосистемой Arduino. Это позволяет без существенных трудностей реализовывать управление шаговыми двигателями через драйверы A4988, DRV8825 и аналогичные, включая алгоритмы разгона и торможения. Благодаря этому микроконтроллеры ATmega являются удобным и обоснованным выбором для реализации логики управления и пользовательского интерфейса в рамках курсового проекта.

STM32

Микроконтроллеры STM32 (ARM Cortex-M) обладают более высокой производительностью, точными таймерами и развитой периферией, что делает их эффективными для сложных задач мотор-контроля и многодвигательных

систем. Однако использование STM32 требует более глубоких знаний встроенного программирования и работы с низкоуровневыми библиотеками, что усложняет разработку учебного проекта по сравнению с Arduino.

PIC

Семейство PIC ориентировано на недорогие встраиваемые решения и отличается надёжными портами ввода-вывода и хорошей поддержкой работы в режиме реального времени. В то же время по удобству разработки, наличию готовых библиотек и примеров для управления шаговыми двигателями PIC уступает платформе Arduino, что увеличивает трудоёмкость проекта на уровне курсовой работы.

ESP32

ESP32 представляет собой высокопроизводительный 32-битный микроконтроллер с встроенными Wi-Fi и Bluetooth, позволяющий реализовывать беспроводное управление шаговым двигателем и web-интерфейс без дополнительных модулей связи. Однако наличие двухъядерной архитектуры и сетевого стека усложняет программирование и отладку, а для типовой учебной системы позиционирования возможности ESP32 зачастую избыточны по сравнению с более простой и доступной платформой Arduino.

Обоснование выбора ATmega8L

Микроконтроллер ATmega выбран в качестве целевой платформы благодаря простой и наглядной архитектуре, хорошей документации и широкому распространению в учебных и лабораторных работах. Семейство ATmega обеспечивает достаточную производительность и набор периферийных модулей (таймеры, прерывания, GPIO) для реализации алгоритмов управления шаговыми двигателями, включая плавный разгон и торможение. Использование ATmega позволяет детально контролировать работу аппаратных ресурсов и таймингов, что важно при анализе точности позиционирования. При этом сложность программной и аппаратной конфигурации остается умеренной по сравнению с более современными и многофункциональными

микроконтроллерами, такими как STM32, PIC или ESP32, что делает ATmega обоснованным выбором для курсового проекта.

2.4. Перечень задач, подлежащих решению в процессе разработки

В процессе разработки системы будут решаться следующие задачи:

Анализ и уточнение требований

- Изучение потребностей пользователей и требований к функционалу
- Оценка текущих решений на рынке, выявление недостатков и недостающих функций
- Формулировка детализированных технических требований к каждому компоненту системы.

Разработка программного кода

- Проектирование архитектуры системы с учетом всех необходимых модулей и интеграций
- Выбор технологий и инструментов для реализации системы
- Создание схемы взаимодействия компонентов

Разработка схемотехники

- Проектирование схемотехники
- Обработка ошибок и исключений

Тестирование системы

- Проверка работоспособности системы в различных условиях
- Проверка производительности и отзывчивости

Документирование системы

- Составление технической документации, описывающей архитектуру и функциональность системы
- Разработка пользовательской документации для пользователей

Запуск системы и поддержка

- Запуск системы в эксплуатацию.
- Обеспечение технической поддержки и обновлений в процессе использования системы.

2.5. Обоснование выбора инструментов и платформы для разработки

Для реализации системы были выбраны следующие инструменты:

1. **Язык программирования C**

   В Atemel Studio используется язык программирования C.

2. **Проектирование электронных схем с помощью САПР Proteus**

   Proteus является одной из лучших САПР для курсового проектирования, так как позволяет запустить и отладить прошивку без использования настоящей электроники.

3. **Библиотека CheapStepper для работы с шаговым двигателем**

   Библиотека CheapStepper создана специально для работы с популярным (и самым доступным для студентов) комплектом: шаговый двигатель 28BYJ-48 и драйвер ULN2003. Её главное преимущество в том, что она исправляет «врожденные» недостатки стандартной библиотеки Stepper.h при работе именно с этим мотором.

4. **Среда разработка**

   Atmel Studio — это интегрированная среда разработки (IDE), предназначенная для программирования микроконтроллеров семейства AVR, включая ATmega. Она предоставляет полный набор инструментов для написания кода на языках C и C++, его компиляции, отладки и прошивки микроконтроллера. Среда поддерживает аппаратную и программную отладку, работу с регистрами и периферией на низком уровне, а также подключение внешних библиотек и конфигурацию проекта под конкретную модель микроконтроллера, что делает Atmel Studio удобным инструментом для разработки и отладки встроенных систем.

Вывод: Выбранные инструменты позволят эффективно реализовать функционал управления шаговым двигателем с помощью микроконтроллера ATmega.

3. ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ

3.1. Разработка аппаратной и программной структуры системы

В основе проектируемой системы лежит микроконтроллер ATmega8L, управляемый программным обеспечением, реализующим алгоритмы автоматического управления шаговым двигателем. В качестве объекта управления выбран дешевый униполярный шаговый двигатель 28BYJ-48 в сочетании с драйвером ULN2003. Для визуализации этапов работы системы и обратной связи с пользователем используются четыре светодиода. Такое аппаратное решение обеспечивает универсальность, простоту сборки и позволяет полностью реализовать заложенный функционал.

Структурная схема устройства

Система состоит из следующих основных компонентов:

- Микроконтроллер ATmega8L – вычислительный центр системы, на котором выполняется управляющая программа
- Драйвер шагового двигателя ULN2003 – силовой интерфейс, преобразующий низковольтные цифровые сигналы в токи для питания обмоток двигателя
- Шаговый двигатель 28BYJ-48 – объект управления, осуществляющий дискретное угловое перемещение вала
- Светодиоды индикации этапов работы (LED0-LED3) – визуальная обратная связь о состоянии системы

Микроконтроллер производит формирование управляющих сигналов на основе заложенного алгоритма, подавая их на входы драйвера ULN2003. Драйвер преобразует сигналы в токи, необходимые для питания обмоток двигателя.

Описание программного алгоритма

Управляющая программа реализована на языке C в среде Atmel Studio. Основная логика работы программы строится на неблокирующем конечном автомате состояний с поддержкой многозадачности.

Первым делом происходит инициализация всех необходимых переменных для последующего их использования в программном коде. После определения переменных запускается основной цикл while(), реализующий фазу вращения двигателя и индикации. Чтобы точно отследить работу программы и подать ток на диоды требуется вычислять время с момента запуска алгоритма, что обеспечивают две переменные elapsedTime и remainingTime. По истечении 60 секунд двигатель останавливается, все светодиоды выключаются, и система переходит в режим перезапуска. Такой алгоритм позволяет просто реализовать управление шаговым двигателем.

3.2. Разработка алгоритма системы

Работа системы начинается с инициализации портов:

```
DDRC |= (1 << DS1) | (1 << SHCP1) | (1 << STCP1) | (1 << DS2) | (1 << SHCP2) | (1 << STCP2);
DDRD |= (1 << DS3) | (1 << SHCP3) | (1 << STCP3) | (1 << DS4) | (1 << SHCP4) | (1 << STCP4);;
DDRB = 0x0F;

uint8_t diodeState1 = 0xFF;
uint8_t diodeState2 = 0xFF;
uint8_t diodeState3 = 0xFF;
uint8_t diodeState4 = 0xFF;
```

Основной алгоритм управления реализуется с помощью цикла while() и выполняет следующие действия:

1. Вычисление прошедшего времени:

- elapsedTime — время, прошедшее с момента запуска (в миллисекундах)

- remainingTime — оставшееся время до конца 60-секундного цикла

2. Фаза вращения двигателя (60 секунд):
   - Включение светодиодов LED0-LED30
   - Двигатель непрерывно вращается с использованием функции stepMotor()
   - Светодиоды LED0-LED30 выключаются каждые 2 секунды по мере вращения двигателя.

4. Завершение цикла:
   - По истечении 60 секунд двигатель останавливается
   - Все светодиоды выключаются
   - Система переходит в режим перезапуска

3.3. Разработка аппаратной архитектуры системы

Cirkit Designer — приложение для проектирования схемы установки.

4. ПРОИЗВОДСТВЕННО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

4.1. Тестирование аппаратной и программной частей системы

Процесс отладки разработанного устройства проводился в несколько этапов, включающих проверку монтажа, верификацию электрических параметров и функциональное тестирование программного обеспечения.

4.1.1 Аппаратная отладка

На первом этапе производился визуальный контроль правильности подключения компонентов согласно принципиальной электрической схеме. Особое внимание уделялось:

- Корректности подключения фаз шагового двигателя к драйверу
- Соблюдению полярности светодиодов и номиналов токоограничивающих резисторов
- Качеству соединений на макетной плате для исключения вибрации контактов.

После подачи питания через USB-интерфейс (5В) производился замер напряжения на шине питания драйвера двигателя. Убедившись в отсутствии коротких замыканий и перегрева компонентов, был выполнен переход к программной отладке.

4.1.2 Программная отладка

Отладка кода выполнялась в среде Atmel Studio симулитора Proteus.

- Проверка инициализации: При старте системы в консоль должно выводиться сообщение System ready, что подтверждает корректную работу функции setup().
- Тестирование логики таймера: Проверялась корректность работы функции millis(). Для этого отслеживались метки времени, выводимые в Serial Monitor («Time remaining: X seconds»). Ошибка накопления времени за 60 секунд составила менее 10 мс, что является допустимым для данной задачи.

4.2. Разработка руководства пользователя и программиста

Для эксплуатации и сопровождения системы разработаны инструкции для двух категорий пользователей.

4.2.1 Руководство пользователя

Назначение: система предназначена для демонстрации автоматического управления шаговым двигателем с циклическим режимом работы и световой индикацией.

Порядок работы:

1. Подключение: подключите питание от схемы к источнику питания (USB-порт компьютера или блок питания 5В)
2. Запуск:
3. Рабочий цикл:
   - двигатель начнет вращение автоматически;
   - каждые две секунды будут потухать светодиоды LED0-LED29
4. Завершение и перезапуск: по истечении 60 секунд двигатель остановится, все индикаторы погаснут. Через 5 секунд паузы система автоматически перезапустится.

4.2.2 Руководство программиста

Требования к среде:

- Среда разработки: Atmel Studio

Структура программы:

Программа построена на базе конечного автомата. Основные параметры вынесены в дефайны и глобальные переменные для удобства настройки:

- #define … — назначение пинов для светодиодов.
- shiftOut() - функция управления светодиодами через регистр сдвига 74HC595.

- stepMotor() - функция управления пошаговым двигателем.

4.3. Экспериментальные данные тестирования процедур и функциональных задач

Для подтверждения работоспособности системы были проведены контрольные испытания. Результаты зафиксированы в таблице протокола испытаний. (см. Таблица 1)

Таблица 1. Тестирование системы

№ п/п	Проверяемая функция	Ожидаемый результат	Фактический результат	Статус
1	Инициализация	При подаче питания в Serial Monitor выводится "System ready", двигатель начинает вращение.	Сообщение получено, вал двигателя начал вращение.	Успех
2	Вращение и реверс	Двигатель вращается плавно. После 4096 шагов направление меняется на противоположное.	Смена направления зафиксирована визуально и подтверждена логом "Direction changed".	Успех
3	Тайминг индикации	Светодиоды потухают каждые 2 секунды	Включение светодиодов происходит синхронно с таймером в Serial Monitor (погрешность <0.5 сек).	Успех
4	Остановка цикла	На 60-й секунде двигатель останавливается, светодиоды гаснут.	Остановка произошла ровно по истечении таймераremainingTime = 0.	Успех

№ п/п	Проверяемая функция	Ожидаемый результат	Фактический результат	Статус
5	Автоматический перезапуск	После паузы 5 секунд цикл начинается заново с обнулением переменных.	Функцияге startSystem()отработала корректно, новый цикл запущен.	Успех

Вывод: В ходе производственно-технологического этапа была проведена сборка макета и комплексная отладка программного обеспечения. Тестирование показало, что разработанная система полностью соответствует техническому заданию: обеспечивается стабильное управление шаговым двигателем, корректная отработка временных интервалов и безотказное функционирование логики световой индикации.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была успешно решена задача разработки системы автоматического управления шаговым двигателем на базе микроконтроллера. Выполненный проект представляет собой законченное аппаратно-программное решение, прошедшее путь от анализа технического задания до экспериментального тестирования макета.

При проектировании системы был проведен анализ существующих подходов к управлению шаговыми приводами, описанных в технической литературе и аналогичных проектах.

В результате было создано программное обеспечение и собран действующий макет, обеспечивающий цикличное вращение двигателя с автоматическим реверсом и интеллектуальной световой индикацией. Экспериментальные испытания подтвердили стабильность работы системы во всех режимах.

Практическая значимость работы заключается в создании гибкого модуля управления, который, в отличие от жесткой логики промышленных контроллеров начального уровня, легко адаптируется под новые задачи (изменение скорости, логики индикации или траектории движения) путем простой корректировки программного кода.

Таким образом, цель работы достигнута, а выбранные методы проектирования доказали свою эффективность и преимущество перед рассмотренными аналогами.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ATmega8 Datasheet. Atmel Corporation. 8-bit AVR Microcontroller ATmega8. — Official technical documentation.
2. Мортон К. Микроконтроллеры AVR: вводный курс. — М.: ДМК Пресс, 2018.
3. Юферев Л. В. Программирование микроконтроллеров AVR на языке C. — СПб.: БХВ-Петербург, 2019.

Барретт С., Пак Д. Микроконтроллеры AVR: архитектура, программирование и применение. — М.: Вильямс, 2017.

4. WinAVR User Manual. Документация к компилятору avr-gcc и утилитам для разработки под AVR.
5. Atmel Studio User Guide. Microchip Technology Inc. — Руководство пользователя среды разработки Atmel Studio.
6. Монк С. Программирование встроенных систем. — М.: Питер, 2016.
7. Application Notes AVR. Atmel Corporation — Методические материалы и примеры применения микроконтроллеров AVR.

ПРИЛОЖЕНИЕ

Программный код

```
#define F_CPU 1000000UL

#include <avr/io.h>

#include <util/delay.h>



#define DS1    PC0

#define SHCP1  PC1

#define STCP1  PC2


#define DS2    PC3

#define SHCP2  PC4

#define STCP2  PC5


#define DS3    PD0

#define SHCP3  PD1

#define STCP3  PD2


#define DS4    PD3

#define SHCP4  PD4

#define STCP4  PD5


void shiftOut(uint8_t data, uint8_t DS, uint8_t SHCP, uint8_t STCP, volatile uint8_t*
DATA_PORT) {

       for (int i = 7; i >= 0; i--) {

              if (data & (1 << i))

              *DATA_PORT |= (1 << DS);

              else

              *DATA_PORT &= ~(1 << DS);



              *DATA_PORT |= (1 << SHCP);

              _delay_us(1);
```

```
                *DATA_PORT &= ~(1 << SHCP);

        }



        *DATA_PORT |= (1 << STCP);

        _delay_us(1);

        *DATA_PORT &= ~(1 << STCP);

}



void turnOffDiode(int diodeNumber, uint8_t DS, uint8_t SHCP, uint8_t STCP, volatile uint8_t*
DATA_PORT, uint8_t* state) {

        *state &= ~(1 << diodeNumber);

        shiftOut(*state, DS, SHCP, STCP, DATA_PORT);

}




void stepMotor(int step) {

        switch(step) {

                case 0: PORTB = 0b00000001; break;

                case 1: PORTB = 0b00000011; break;

                case 2: PORTB = 0b00000010; break;

                case 3: PORTB = 0b00000110; break;

                case 4: PORTB = 0b00000100; break;

                case 5: PORTB = 0b00001100; break;

                case 6: PORTB = 0b00001000; break;

                case 7: PORTB = 0b00001001; break;

        }

}



int main(void) {

        DDRC |= (1 << DS1) | (1 << SHCP1) | (1 << STCP1) | (1 << DS2) | (1 << SHCP2) | (1 <<
STCP2);

        DDRD |= (1 << DS3) | (1 << SHCP3) | (1 << STCP3) | (1 << DS4) | (1 << SHCP4) | (1 <<
STCP4);;

        DDRB = 0x0F;



        uint8_t diodeState1 = 0xFF;
```

```
uint8_t diodeState2 = 0xFF;
uint8_t diodeState3 = 0xFF;
uint8_t diodeState4 = 0xFF;

shiftOut(diodeState1, DS1, SHCP1, STCP1, &PORTC);
shiftOut(diodeState2, DS2, SHCP2, STCP2, &PORTC);
shiftOut(diodeState3, DS3, SHCP3, STCP3, &PORTD);
shiftOut(diodeState4, DS4, SHCP4, STCP4, &PORTD);

_delay_ms(1000);

diodeState1 = 0x00;
diodeState2 = 0x00;
diodeState3 = 0x00;
diodeState4 = 0x00;

shiftOut(diodeState1, DS1, SHCP1, STCP1, &PORTC);
shiftOut(diodeState2, DS2, SHCP2, STCP2, &PORTC);
shiftOut(diodeState3, DS3, SHCP3, STCP3, &PORTD);
shiftOut(diodeState4, DS4, SHCP4, STCP4, &PORTD);

_delay_ms(1000);

diodeState1 = 0xFF;
diodeState2 = 0xFF;
diodeState3 = 0xFF;
diodeState4 = 0xFF;

shiftOut(diodeState1, DS1, SHCP1, STCP1, &PORTC);
shiftOut(diodeState2, DS2, SHCP2, STCP2, &PORTC);
shiftOut(diodeState3, DS3, SHCP3, STCP3, &PORTD);
shiftOut(diodeState4, DS4, SHCP4, STCP4, &PORTD);

int step = 0;
```

```
	int totalSteps = 0;
	int diodesLit = 32;
	int moduleDioteCount = 0;

	while (1) {
		stepMotor(step);
		step = (step + 1) % 8;
		totalSteps ++;
		if (totalSteps % 400 == 0) {
			if (diodesLit > 24) {
				turnOffDiode(moduleDioteCount, DS1, SHCP1, STCP1, &PORTC,
&diodeState1);
			} else if (diodesLit > 16) {
				if (diodesLit == 24)
					moduleDioteCount = 0;
				turnOffDiode(moduleDioteCount, DS2, SHCP2, STCP2, &PORTC,
&diodeState2);
			} else if (diodesLit > 8) {
				if (diodesLit == 16)
					moduleDioteCount = 0;
					turnOffDiode(moduleDioteCount, DS3, SHCP3, STCP3,
&PORTD, &diodeState3);
			} else {
				if (diodesLit == 8)
					moduleDioteCount = 0;
				turnOffDiode(moduleDioteCount, DS4, SHCP4, STCP4, &PORTD,
&diodeState4);
			}
			moduleDioteCount++;
			diodesLit--;
		}
		_delay_ms(5);

		if (totalSteps == 12000) {
			diodeState1 = 0xFF;
			diodeState2 = 0xFF;
```

```
                        diodeState3 = 0xFF;
                        diodeState4 = 0xFF;

                        shiftOut(diodeState1, DS1, SHCP1, STCP1, &PORTC);
                        shiftOut(diodeState2, DS2, SHCP2, STCP2, &PORTC);
                        shiftOut(diodeState3, DS3, SHCP3, STCP3, &PORTD);
                        shiftOut(diodeState4, DS4, SHCP4, STCP4, &PORTD);

                        step = 0;
                        totalSteps = 0;
                        diodesLit = 32;
                        moduleDioteCount = 0;
                        _delay_ms(5000);
                }
        }
}
```