Министерство науки и высшего образования Российской Федерации

**Муромский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Владимирский государственный университет   
имени Александра Григорьевича и Николая Григорьевича Столетовых»**

(МИВлГУ)

Факультет ИТР

Кафедра ИС

КУРСОВАЯ

РАБОТА

по Прикладная разработка на Java

(наименование дисциплины)

Тема Разработка драйвера для матричного умножителя

Руководитель

Щаников С.А

(оценка) (фамилия, инициалы)

Члены комиссии

(подпись) (дата)

Студент ИС-122

(подпись) (Ф.И.О.) (группа)

Колоколов К.А

(подпись) (Ф.И.О.) (фамилия, инициалы)

(подпись) (дата)

Муром 2025

БЛАНК ЗАДАНИЯ

В курсовой работе отражена разработка драйвера для матричного умножителя, проведены тестирование и отладка, приведены примеры работы.

Ил. 2. Библ. 6.

Министерство науки и высшего образования Российской Федерации

**Муромский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Владимирский государственный университет   
имени Александра Григорьевича и Николая Григорьевича Столетовых»**

(МИВлГУ)

Факультет ИТР

Кафедра ИС

ПОЯСНИТЕЛЬНАЯ

ЗАПИСКА

Тема: Разработка драйвера для матричного умножителя

МИВУ.09.03.02-09.000 ПЗ

Муром 2025

Оглавление

[ВВЕДЕНИЕ 6](#_Toc199258879)

[1 Анализ технического задания 7](#_Toc199258880)

[2 Проектирование программы 9](#_Toc199258881)

[3 Разработка программы 11](#_Toc199258882)

[4 Тестирование 12](#_Toc199258883)

[ЗАКЛЮЧЕНИЕ 14](#_Toc199258884)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 15](#_Toc199258885)

ВВЕДЕНИЕ

Современные микрокомпьютеры, такие как Raspberry Pi, находят широкое применение в задачах управления различными периферийными устройствами. Работа с ЦАП/АЦП, цифровыми ключами и мультиплексорами требует наличия драйвера, способного взаимодействовать с аппаратной частью напрямую через интерфейсы GPIO и SPI.

Цель данной курсовой работы — разработать драйвер, реализующий управление цифровыми ключами, аналоговыми преобразователями и коммутацией сигналов. Драйвер реализуется на языке программирования C, с использованием библиотеки WiringPi. Разработка ориентирована на практическое применение в проектах, связанных с нейроморфными вычислениями и аналоговой матричной памятью (MVM — Multiply Vector Matrix).

1 Анализ технического задания

Курсовая работа предполагает создание драйвера управления периферийными устройствами на базе Raspberry Pi. Основной задачей является реализация программного модуля, способного управлять цифровыми ключами, аналогово-цифровыми и цифро-аналоговыми преобразователями, а также мультиплексорами, используя интерфейсы GPIO и SPI.

Согласно техническому заданию, драйвер должен предоставлять следующие возможности. Во-первых, он должен обеспечивать возможность конфигурирования цифровых ключей, подключённых через сдвиговые регистры 74HC595. Эти ключи используются для управления состоянием элементов матрицы, имитирующей поведение аналоговой памяти. Во-вторых, требуется реализация взаимодействия с ЦАП и АЦП, которые связаны с Raspberry Pi по SPI-шине. Передача данных через SPI должна быть реализована на низком уровне, с возможностью задавать рабочий режим (SPI mode 0 или 1), скорость и последовательность передачи. Дополнительно необходимо реализовать управление SPDT-коммутацией для настройки направлений сигналов и подключения конкретных линий ввода/вывода.

Также одной из целей является обеспечение нескольких логических режимов работы, включая чтение (read), запись (write) и режим матричного умножения (MVM), который имитирует работу аналоговой нейроморфной памяти. Для корректной работы драйвера необходимо обеспечить правильную последовательность переключения режимов, подачи управляющих сигналов и считывания данных с учётом временных задержек.

Кроме реализации программной части, требуется предусмотреть возможность как компиляции в виде библиотеки, пригодной для использования в сторонних приложениях, так и создания исполняемого файла для автономного запуска и отладки. Все элементы должны быть реализованы на языке C с использованием библиотеки WiringPi для работы с GPIO и системных вызовов Linux для работы с SPI.

Таким образом, техническое задание определяет создание низкоуровневого модуля, способного управлять несколькими типами аналогово-цифровых компонентов в рамках единой логики, с возможностью гибкого масштабирования и расширения в будущих проектах.

2 Проектирование программы

На этапе проектирования программы было принято решение организовать модульную структуру исходного кода, что позволяет упростить поддержку и расширение функциональности драйвера. Архитектура проекта предполагает разделение на отдельные логические компоненты, каждый из которых отвечает за конкретную функциональность. Основной модуль управления — rpi\_modes.c — включает в себя общую логику работы драйвера и реализует конкретные режимы взаимодействия с аппаратными компонентами. Внутри этого модуля используется структура RPI\_modes, которая содержит два основных элемента: структуру для работы с SPI-интерфейсом и структуру для управления сдвиговыми регистрами и ключами.

Подключаемый модуль r595hc реализует управление регистрами 74HC595, обеспечивая управление включением и выключением ключей, коммутацией сигналов и активацией необходимых цепей. Модуль MVM\_SPI предоставляет интерфейс для работы с шиной SPI, включая функции отправки и приёма данных, а также вспомогательные функции управления ЦАП (DAC), АЦП (ADC) и SPDT-ключами.

Основная логика управления реализована в форме режимов, которые моделируют различные сценарии использования матрицы аналоговых компонентов. Каждый режим выполняет определённую последовательность операций: установка режима SPI, конфигурация ключей, установка напряжений на DAC, считывание аналоговых сигналов с ADC и возврат результатов.

2.2.3 Диаграмма классов

На рисунке 2 представлена упрощенная диаграмма классов, отражающая основные компоненты программы и их взаимосвязи.

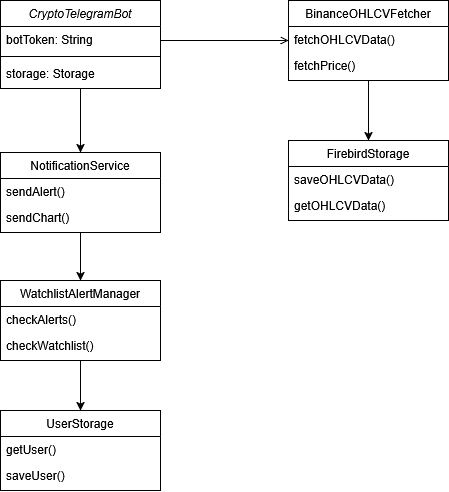


Рисунок 2 – Диаграмма классов

3 Разработка программы

Процесс разработки программы начался с реализации функции инициализации RPI\_modes\_init, в которой происходит настройка SPI-интерфейса, инициализация сдвиговых регистров, а также начальная конфигурация GPIO-выводов. Данная функция вызывается в начале работы программы и обеспечивает готовность всех компонентов к работе.

Следующим этапом стала реализация функций управления ключами и преобразователями. Для управления цифровыми ключами используется последовательная запись в сдвиговые регистры с последующей активацией нужных линий. Напряжение на DAC подаётся через SPI-последовательность, включающую команды активации и загрузки значения. После установки нужного напряжения сигнал подтверждается через линию LDAC, обеспечивая точную передачу значения на выход.

Для чтения аналоговых сигналов используется встроенный АЦП, взаимодействие с которым также реализовано через SPI. Сначала отправляется команда инициализации, затем производится считывание нескольких значений с усреднением для повышения точности измерений.

Особое внимание уделено режиму MVM, в котором на строку матрицы подаются аналоговые значения одновременно на все каналы, а с другой стороны считывается результирующее напряжение. Это позволяет моделировать процесс матричных умножений, применяемый в нейроморфных системах.

Программа компилируется как в виде динамической библиотеки (.so), так и как отдельное исполняемое приложение. Это позволяет использовать её как часть более крупной системы или запускать автономно для тестирования.

4 Тестирование

После завершения разработки программного обеспечения был проведён этап тестирования, направленный на проверку корректности работы драйвера в различных режимах. Тестирование проводилось на реальной аппаратной платформе Raspberry Pi с подключёнными компонентами: DAC, ADC, регистрами 74HC595 и цифровыми ключами.

Основной акцент при тестировании делался на проверку всех режимов работы: записи сигнала на элемент матрицы (mode\_7), чтения аналогового значения (mode\_9), одновременной работы с множеством элементов (mode\_mvm), а также упрощённой версии режима MVM (fast\_mvm). В каждом из этих режимов проверялась правильность установки управляющих сигналов на GPIO, передача значений по SPI и точность считывания результатов.

Для проверки режимов записи и считывания использовались предопределённые значения напряжения, подаваемые на ЦАП. Затем считывались соответствующие аналоговые значения через АЦП и сравнивались с ожидаемыми. Также проводилась проверка маскирования и включения нужных каналов в режиме работы с маской, что особенно важно для режима fast\_mvm. Кроме того, проверялась устойчивость работы драйвера при многократных запусках функций, включая повторную инициализацию, изменение направления сигнала и повторное считывание данных.

Результаты тестирования подтвердили работоспособность всех реализованных функций. Значения, полученные с АЦП, соответствовали ожидаемым в пределах допустимой погрешности. Управляющие сигналы на ключах и мультиплексорах устанавливались в нужной последовательности, что подтверждалось как логическим анализатором, так и визуальной индикацией.

Тестирование также показало, что драйвер устойчив к ошибкам конфигурации: при попытке установить недопустимые значения, например, выход за пределы допустимого диапазона строк WL, программа корректно выводила предупреждающее сообщение и не выполняла некорректную операцию. Это указывает на наличие базовой защиты от пользовательских ошибок и делает драйвер пригодным для использования в учебных и исследовательских проектах.

ЗАКЛЮЧЕНИЕ

Разработка драйвера управления аналоговыми компонентами для Raspberry Pi на языке C позволила глубже понять принципы работы с низкоуровневыми интерфейсами и микроконтроллерной периферией. Созданная программа обеспечивает надёжное и гибкое управление ЦАП, АЦП, цифровыми ключами и мультиплексорами, позволяя реализовать как базовые операции чтения и записи, так и более сложные режимы, такие как MVM.

Полученное решение может быть использовано в исследованиях и лабораторных установках, связанных с аналоговыми вычислениями и нейроморфными архитектурами. Благодаря модульной структуре драйвер может быть легко расширен для поддержки других устройств или адаптирован под разные аппаратные конфигурации. Работа также продемонстрировала эффективность использования языка C и библиотеки WiringPi для прямого управления периферийными устройствами в реальном времени.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. WiringPi Documentation – <http://wiringpi.com>
2. Raspberry Pi SPI Documentation – https://www.raspberrypi.com/documentation/computers/raspberry-pi.html
3. Datasheet 74HC595 – Texas Instruments
4. Datasheet MCP4922 (DAC) – Microchip
5. Datasheet MCP3008 (ADC) – Microchip