МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«Новосибирский государственный технический университет»

Кафедра электронных приборов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **УТВЕРЖДАЮ** | | | |
| Заведующий кафедрой,  д. т. н., доцент | | | |
|  | | А. М. Семенов | |
| « » |  | | 20 г. |

|  |
| --- |
| Жеребцов Артем Алексеевич |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

|  |
| --- |
| **Разработка программной части тестового стенда для аналого-цифрового** |
| **Модуля лазерного детектора метана** |

Факультет радиотехники и электроники

Направление подготовки 11.03.04 — электроника и наноэлектроника

|  |  |  |  |
| --- | --- | --- | --- |
| Автор работы |  |  | А. А. Жеребцов, гр. РЭ3-11 |
|  |  |  |  |
| Руководитель работы |  |  | н. с., А. Ю. Ивойлов |
|  |  |  |  |
| Руководитель от НГТУ |  |  | к. т. н., доцент С. А. Чипурнов |
|  |  |  |  |
| Нормоконтроль |  |  | к. т. н., доцент С. А. Чипурнов |

Новосибирск 2025

МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«Новосибирский государственный технический университет»

Кафедра электронных приборов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **УТВЕРЖДАЮ** | | | |
| Заведующий кафедрой,  д. т. н., профессор | | | |
|  | | А. М. Семенов | |
| « » |  | | 20 г. |

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

студенту *Жеребцову Артему Алексеевичу*

Направление подготовки 11.03.04 — *электроника и наноэлектроника*

*Факультет радиотехники и электроники*

Тема  *Разработка программной части тестового стенда для аналого-цифрового модуля лазерного детектора метана*

Исходные данные (или цель работы): Разработать программное обеспечение на языке C и C++ для микроконтроллера STM32 блока измерения и диагностики тестового стенда для проверки работоспособности критических узлов тестируемой платы лазерного детектора метана. Выбрать микроконтроллер и среду разработки программного обеспечения. Провести тестирование на испытательном стенде.

Структурные части работы:

Задание согласовано и принято к исполнению.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Научный руководитель** |  | **Руководитель от НГТУ** |  | **Автор выпускной**  **квалификационной работы** |
| Ивойлов Андрей Юрьевич  (фамилия, имя, отчество) |  | Чипурнов Сергей  Алексеевич  (фамилия, имя, отчество) |  | Жеребцов Артем  Алексеевич  (фамилия, имя, отчество) |
| н.с. |  | к.т.н., доцент |  | РЭФ, РЭ3-11 |
|  |  |  |  |  |
| (подпись, дата) |  | (подпись, дата) |  | (подпись, дата) |

Тема утверждена приказом по НГТУ № 0 от « 0 » марта 2025г.

ВКР сдана в ГЭК № , тема сверена с данными приказа

(подпись секретаря государственной экзаменационной комиссии по защите ВКР, дата)

(фамилия, имя, отчество секретаря государственной экзаменационной комиссии по защите ВКР)

РЕФЕРАТ

Пояснительная записка 0 с., включая приложение, 0 рис., 0 табл., 0 ист., 0 прил., 0 лист ил. мат.

СИСТЕМА ДИАГНОСТИКИ, СИСТЕМА СБОРА ДАННЫХ, СИСТЕМА УПРАВЛЕНИЯ, СИСТЕМА ПЕРЕДАЧИ ДАННЫХ, СИСТЕМА ОБРАБОТКИ

Объектом разработки является программное обеспечение (ПО) на персональный компьютер (ПК) и прошивка для микроконтроллера STM32, являющийся частью платы тестового стенда. Стенд производит проверку ключевых модулей аналого-цифрового модуля лазерного детектора метана.

Цель работы — разработка программы управления для тестового стенда на языке C. Разработка протокола для взаимодействия микроконтроллера стенда с компьютером и исследуемой платой, разработка ПО на ПК, которое будет производить автоматизированную проверку, выводить статус выполнения, строить график на языке C++.

В процессе работы был исследован алгоритм работы аналого-цифрового модуля лазерного детектора метана. На этой основе был сформирован порядок тестирования ключевых модулей этого устройства, для точного выявления дефектов и брака.

В результате работы был описан алгоритм работы платы тестового стенда. Разработаны модули прошивки МК, для работы с платами цифровых входов, цифровых выходов, аналоговых входов. Разработан графический интерфейс для ПК. Проведено тестирование модулей на испытательном стенде, составлена документация.

Содержание

Стр.

[Введение 8](#_Toc196419409)

[1 Литературный обзор 11](#_Toc196419410)

[1.1 Тестовые стенды 11](#_Toc196419411)

[1.2 Принципы контактирования с тестируемыми платами 11](#_Toc196419412)

[1.2.1 Ручной метод 11](#_Toc196419413)

[1.2.2 Метод «ложе гвоздей» 12](#_Toc196419414)

[1.2.3 Метод «летающие щупы» 13](#_Toc196419415)

[1.2.4 Метод «летающих матриц» 14](#_Toc196419416)

[1.2.5 Анализ методов контактирования 15](#_Toc196419417)

[1.3 Программное обеспечение тестовых стендов 17](#_Toc196419418)

[1.3.1 Низкоуровневое программное обеспечение 17](#_Toc196419419)

[1.3.2 Уровни управления и сбора данных 17](#_Toc196419420)

[1.3.3 Уровни обработки и анализа данных 18](#_Toc196419421)

[1.4 Обзор метода «ложе гвоздей» с точки зрения программного обеспечения 18](#_Toc196419422)

[2 Практическая часть 21](#_Toc196419423)

[2.1 Лазерный газоанализатор метана 21](#_Toc196419424)

[2.2 Методика разработки 22](#_Toc196419425)

[2.2.1 Выбор 32-разрядного микроконтроллера 23](#_Toc196419426)

[2.2.2 Среды разработки 26](#_Toc196419427)

[2.3 Функционал тестового стенда 28](#_Toc196419428)

[2.4 Конфигурация выводов микроконтроллера 32](#_Toc196419429)

[2.5 Разработка прошивки микроконтроллера 34](#_Toc196419430)

[2.6 Модуль инициализации системы и управления основным циклом выполнения 37](#_Toc196419431)

[2.7 Модуль обработки протокольных пакетов и диспетчеризации команд 40](#_Toc196419432)

[2.8 Модуль управления аппаратной логикой и взаимодействием с периферией 45](#_Toc196419433)

[2.9 Конструкция тестового стенда 46](#_Toc196419434)

[Заключение 48](#_Toc196419435)

[Список использованных источников 49](#_Toc196419436)

[Приложение А 50](#_Toc196419437)

[Приложение Б 51](#_Toc196419438)

[Приложение В 52](#_Toc196419439)

[Приложение Г 53](#_Toc196419440)

[Приложение Д 54](#_Toc196419441)

[Приложение Е 55](#_Toc196419442)

[Приложение И 56](#_Toc196419443)

[Приложение К 57](#_Toc196419444)

[Приложение Л 58](#_Toc196419445)

[Приложение М 59](#_Toc196419446)

[Приложение Н 60](#_Toc196419447)

[Приложение П 61](#_Toc196419448)

[Приложение Р 62](#_Toc196419449)

[Приложение С 63](#_Toc196419450)

[Приложение Т 64](#_Toc196419451)

[Приложение У 65](#_Toc196419452)

[Приложение Ф 66](#_Toc196419453)

[Приложение Х 67](#_Toc196419454)

[Приложение Ц 68](#_Toc196419455)

[Приложение Ш 69](#_Toc196419456)

[Приложение Щ 70](#_Toc196419457)

[Приложение Э 71](#_Toc196419458)

[Приложение Ю 72](#_Toc196419459)

[Приложение Я 73](#_Toc196419460)

Введение

При разработке электроники важным этапом является тестирование устройства. На данном этапе инженеры-схемотехники, разработавшие электрическую схему, рассчитавшие необходимые компоненты, а также расстановку этих компонентов на печатной плате и трассировку печатной платы, осуществляют проверку критически важных узлов и модулей. Инженеры-программисты, в свою очередь, проверяют работоспособность прошивки, анализируют работу всего прибора, выявляя баги и ошибки на стадии тестирования. Всё это делается для того, чтобы отсеять бракованные платы, не прошедшие установленные проверки. Проверенные приборы затем собираются в корпус и высылаются заказчику.

Часто инженерам приходится тестировать готовые платы в большом объёме перед сдачей заказчику. Время, отведённое на проверку, зависит от сложности разработки платы, а также от специфики используемых компонентов и технологий. Задачу может усложнить и тот факт, что производство данного устройства планируется организовать серийно, что делает оптимизацию этапа проверки особенно актуальной. В таких случаях автоматизация тестирования становится необходимой для снижения затрат времени, затрат мощности и повышения точности выявления дефектов.

Одним из решений данной проблемы является разработка автоматизированного стенда, который проводит проверку плат в автоматическом режиме с минимальным участием человека. Такой стенд способен выполнять последовательность тестовых процедур, фиксировать результаты измерений и сразу же сигнализировать о выявленных ошибках. Это позволяет не только сократить время тестирования, но и значительно уменьшить вероятность человеческой ошибки. Такой подход способствует налаживанию серийного и даже массового производства приборов, что, в свою очередь, повышает прибыль компании за счёт оптимизации производственных процессов и повышения качества выпускаемой продукции.

В данном проекте необходимо разработать прошивку на языке программирования Си для микроконтроллера STM32F411CEU6, который является центральным компонентом схемы. Микроконтроллер управляет работой тестового стенда, осуществляет сбор, передачу и обработку данных с тестируемой платы аналого-цифрового модуля лазерного детектора метана. Помимо реализации основных функций, прошивка должна обеспечивать:

* Надёжное управление периферийными устройствами стенда;
* Своевременную обработку сигналов и данных;
* Взаимодействие с внешними устройствами через стандартизированные интерфейсы;
* Ведение журнала ошибок и событий для дальнейшего анализа и устранения неисправностей.

Необходимо написать ПО с графическим интерфейсом на языке программирования C++ для управления работы прошивки данного МК, для отправки, получения и анализа данных, на основе которых будет в удобном пользователю виде представлен статус выполнения и результат выполнения команд.

Оптимальное распределение задач между аппаратной и программной частями системы позволяет добиться высокой точности тестирования и ускорить процесс проверки, что особенно важно при серийном производстве. Внедрение автоматизированного тестирования способствует сокращению себестоимости конечного продукта и повышению его качества, что в конечном итоге удовлетворяет потребности заказчика и усиливает конкурентные преимущества компании.

**Актуальность темы.** Тестовый стенд является важным прибором для тестирования аналого-цифрового модуля лазерного детектора метана. Он призван оптимизировать процесс проверки печатных плат газоанализатора, оперативно выявляя бракованные изделия, с указанием модулей, не прошедших проверку. В отсутствие тестового стенда время, затраченное на выпуск приборов, будет кратно выше, что негативно скажется с финансовой точки зрения для предприятия-изготовителя, а значит актуально данной работы – высока.

**Новизна темы.** Существующие на рынке универсальные тестовые стенды не подходят к применению для плат газоанализатора в связи со спецификой проекта и его сложностью. Необходима разработка узкоспециализированного тестового стенда, которая учтет все нюансы и тонкости платы детектора метана.

**Постановка задачи.** Разработать ПО для микроконтроллера STM32 тестового стенда для аналого-цифрового модуля лазерного детектора метана, разработать ПО с графическим интерфейсом на ПК. Работа включает в себя написание модулей на языке программирования C и C++ для сбора, анализа и передачи состояний модулей исследуемой платы газоанализатора метана на персональный компьютер, а так же составление документации и отладку написанных модулей на испытательном стенде.

# Литературный обзор

## Тестовые стенды

Печатные платы (ПП) являются неотъемлемой частью практически всех электронных устройств, от бытовых приборов до высокотехнологичных систем. Эффективность их производства и надежность напрямую зависят от качества изготовления, что, в свою очередь, требует применения различных методов тестирования и проверки ПП на соответствие заявленным характеристикам. Для этого используется множество тестовых стендов, которые различаются по принципам работы, оснащению и сложности. В данном обзоре рассматриваются возможные варианты тестовых стендов для проверки ПП, а также обосновывается выбор одного из направлений работы.

Разработка автоматизированных систем проверки печатных плат зависит от конкретных решаемых задач. Критерии отбраковки могут варьироваться по степени строгости, но в целом их можно свести к трём основным направлениям: а) проверка соответствия монтажных соединений электрической схеме (функциональный контроль); б) анализ параметров монтажа на соответствие установленным нормам (параметрический контроль); в) оценка достаточности надёжности платы для эксплуатации (диагностический контроль).

## Принципы контактирования с тестируемыми платами

Существует несколько методов электрического контактирования печатных плат: ручной, «ложе гвоздей», «летающие щупы» и «летающие матрицы».

### Ручной метод

При ручном методе оператор вручную проверяет электрические цепи, используя щупы. Однако из-за человеческого фактора около 25% дефектов остаются незамеченными, что делает этот способ наименее надёжным.



1. — Иллюстрация ручного метода

### Метод «ложе гвоздей»

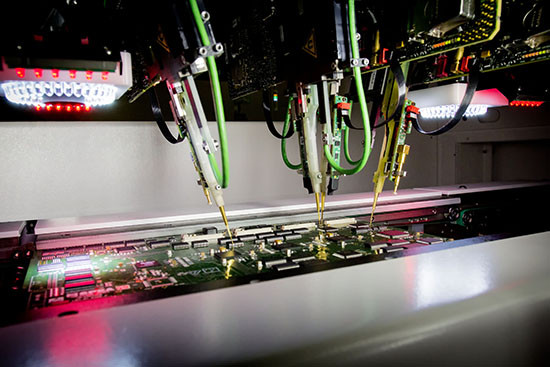
Метод «ложе гвоздей» предполагает использование матрицы подпружиненных контактов, размещённых в узлах координатной сетки. Для обеспечения контакта с конкретной платой применяется перфорированная маска или специальный тестовый адаптер с индивидуально расположенными зондами. Все точки платы контактируются одновременно, а скорость проверки определяется быстродействием переключающих элементов. Этот метод обеспечивает высокую производительность, но требует значительных затрат на переналадку при смене типа платы.



1. — Иллюстрация метода «Ложе гвоздей»

### Метод «летающие щупы»

Метод «летающих щупов» использует тестовую установку с несколькими подвижными головками, каждая из которых оснащена зондом и может перемещаться по трём осям. Контактирование выполняется последовательно по заданной программе, позволяя подавать сигналы или измерять параметры цепей. Такой подход обеспечивает быструю переналадку благодаря использованию CAD/CAM-данных, но обладает невысокой производительностью из-за последовательного характера тестирования.

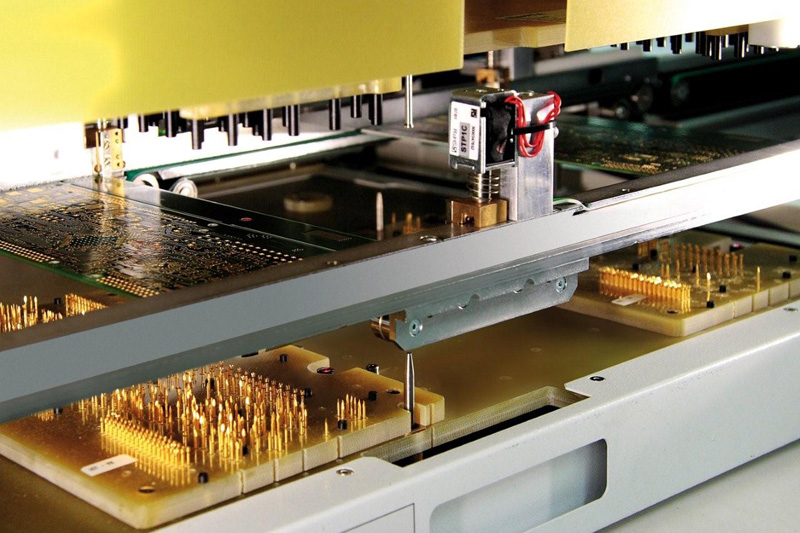


1. — Иллюстрация метода «Летающие щупы»

### Метод «летающих матриц»

Метод «летающих матриц» подразумевает, что на каждой каретке располагается матрица щупов, причём каждый из них имеет независимый привод по оси. Матрицы перемещаются с высокой скоростью на небольшие расстояния, а наиболее близкий зонд выполняет измерения. Благодаря этому производительность возрастает на порядок по сравнению с методом «летающих щупов».

Разновидностью метода «летающих матриц» является метод «Скорпион», при котором контактные щупы сосредоточены в нескольких модулях, расположенных с двух сторон платы. Эти модули оснащены подвижными щупами с большим рабочим ходом, которые последовательно тестируют зоны платы. Данная технология позволяет проверять трёхмерные печатные платы и электронные модули с высокими компонентами, что делает её незаменимой в таких задачах.



1. — Иллюстрация метода «Летающие матрицы»

### Анализ методов контактирования

Для наглядности преимуществ и недостатков четырех перечисленных методов, приведена таблица 1.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1. — Преимущества и недостатки методов электрического контактирования | | | | | |
| Метод | Вероятность  ошибок | Производи-  тельность | Разработка  изделия | Разработка  ПО | Стоимость |
| Ручной | Высокая | Низкая | Не требуется | Не требуется | Низкая |
| Ложе гвоздей | Низкая | Высокая | Требуется | Сложная | Средняя |
| Летающие щупы | Низкая | Средняя | Не требуется | Простая | Высокая |
| Летающие матрицы | Низкая | Высокая | Требуется | Средняя | Высокая |

Выбор оборудования для электрического тестирования должен основываться на нескольких ключевых факторах: требуемой производительности, разнообразии выпускаемой продукции и уровне сложности печатных плат.

Ручной метод, хотя и является наименее затратным с точки зрения капитальных вложений, оправдан в основном для лабораторных исследований и прототипного производства. Однако надёжность такого контроля оставляет желать лучшего.

Оборудование с подвижными зондами выгодно отличается простой переналадки, что делает его оптимальным для мелкосерийного производства с широким ассортиментом плат. Тем не менее, производительность систем с «летающими щупами» недостаточна для массового производства.

Максимальную скорость проверки обеспечивают матричные тестеры, где основной лимитирующий фактор – скорость установки платы на контактное поле и время обработки измерительных сигналов. Такие системы используют коммутаторы на транзисторных ключах, что позволяет ускорить процесс тестирования. Однако применение релейных коммутаторов снижает производительность на порядок, из-за чего этот подход практически вышел из употребления.

Использование матричных тестеров требует значительных затрат на переналадку при смене типа платы, особенно если необходимо изготовление сменных адаптеров из-за несовпадения шагов контактирования с шагом контактного поля. В связи с этим подобные тестеры целесообразно применять в серийном и массовом производстве, где разнообразие выпускаемой продукции невелико.

Оптимальным решением, сочетающим универсальность и высокую производительность, является метод «летающих матриц». Он обеспечивает достаточную скорость проверки и подходит как для мелкосерийного, так и для серийного производства, независимо от ассортимента плат.

При выборе оборудования, помимо производительности и стоимости, следует учитывать тип и износостойкость контактных зондов, точность системы базирования, а также программное обеспечение, входящее в комплект. Именно ПО во многом определяет быстродействие установки и время её переналадки.

## Программное обеспечение тестовых стендов

Программное обеспечение (ПО) для прошивки тестового стенда, предназначенного для проверки печатных плат, можно разделить на несколько категорий в зависимости от уровня управления, типа взаимодействия с оборудованием и целей тестирования.

### Низкоуровневое программное обеспечение

На низком уровне функционирует встроенное ПО (firmware), которое управляет работой микроконтроллеров и отвечает за перемещение зондов, подачу управляющих сигналов, измерение отклика, коммутацию цепей и базовую обработку данных. Такое ПО обычно разрабатывается на языках C, C++ или Assembly и выполняется на встроенных процессорах тестера. В дополнение к этому работают драйверы оборудования, которые обеспечивают взаимодействие тестового стенда с операционной системой компьютера. Они создают интерфейсы для команд управления и передачи данных, используя API оборудования или стандартные протоколы, такие как USB, RS-232, GPIB и Ethernet.

### Уровни управления и сбора данных

Следующим уровнем являются программы, предназначенные для настройки и управления процессом тестирования. В эту категорию входят графические интерфейсы оператора (GUI-программы), позволяющие загружать тестовые программы, визуализировать результаты тестирования, настраивать измерительные параметры и формировать отчеты. Такие приложения разрабатываются на C#, Python, Java, а также с использованием C++ и LabVIEW для визуального программирования. Дополнительно могут использоваться скриптовые системы управления тестированием, которые поддерживают автоматическое выполнение сценариев, написанных на Python или TCL.

### Уровни обработки и анализа данных

После проведения тестов необходимо обработать и проанализировать полученные данные. Для этого применяются специализированные системы, выполняющие обработку измеренных параметров, сравнение с эталонными значениями и выявление брака. Эти программы могут использовать статистический анализ и формировать отчеты, работая на Python или MATLAB. Помимо анализа, важно обеспечить хранение данных и их интеграцию с производственными процессами. Для этого используются базы данных (SQL и NoSQL) и облачные сервисы (Microsoft Azure, AWS IoT, Google Cloud).

В современных системах тестирования активно применяется автоматическая генерация тестовых программ на основе CAD- и CAM-данных печатных плат. Такое ПО анализирует схемотехническую и трассировочную информацию, автоматически формируя последовательность тестирования, работая с форматами Gerber, ODB++ и IPC-2581. Некоторые системы используют алгоритмы машинного обучения для оптимизации тестов, сокращая их количество без потери качества.

## Обзор метода «ложе гвоздей» с точки зрения программного обеспечения

Программное обеспечение для микроконтроллера, встроенного в тестовый стенд с методом «ложе гвоздей», выполняет несколько взаимосвязанных функций, обеспечивая передачу, обработку и управление исследуемой печатной платой. Основная задача прошивки МК – организация взаимодействия между контактной матрицей тестового адаптера и исследуемой платой, а также обеспечение точного измерения электрических параметров и своевременной передачи данных внешней системе управления или оператору. При использовании метода «ложе гвоздей» тестовый стенд оборудован матрицей подпружиненных контактов, которая позволяет одновременно контактировать с заданными точками печатной платы, что требует высокой точности управления коммутацией. Для этого программное обеспечение МК реализует алгоритмы управления матрицей контактов, инициируя замыкание определённых групп контактов посредством цифровых или аналоговых переключателей, что позволяет подавать тестовые сигналы на плату и проводить измерения.

После активации нужной контактной группы происходит подача тестовых сигналов, измерение параметров, таких как напряжение, ток, сопротивление и другие характеристики, с использованием встроенных аналого-цифровых преобразователей (АЦП) и/или внешних измерительных модулей. Полученные данные проходят первичную обработку непосредственно на микроконтроллере, где осуществляется фильтрация шумов, усреднение значений и сравнение с заранее заданными эталонными параметрами. Программное обеспечение анализирует результаты тестирования, выявляет возможные отклонения от нормы, а затем формирует отчёт, который передаётся во внешнюю систему управления через один из стандартных интерфейсов связи, например, UART, USB, SPI или Ethernet. Такой подход обеспечивает оперативное принятие решения о качестве тестируемой печатной платы.

Кроме того, прошивка микроконтроллера включает функции самодиагностики и мониторинга состояния тестового стенда, что позволяет обнаруживать возможные неисправности в самом оборудовании, контролировать корректность работы измерительных цепей и своевременно информировать оператора о необходимости проведения технического обслуживания. Программная архитектура МК может быть реализована с использованием многозадачности, например, с применением операционных систем реального времени (RTOS), что позволяет одновременно выполнять несколько задач: управление коммутацией, сбор данных, их предварительную обработку и коммуникацию с внешними устройствами.

Для разработки такого ПО используются языки низкого уровня, такие как C, C++ и Assembly, что обеспечивает максимальную производительность и надежность работы в условиях ограниченных вычислительных ресурсов микроконтроллера. Также важным аспектом является гибкость системы: возможность легкой перенастройки и обновления программной прошивки в зависимости от типа тестируемых печатных плат и изменяющихся требований производства. Таким образом, ПО для микроконтроллера тестового стенда с методом «ложе гвоздей» объединяет в себе функции управления контактной матрицей, проведения точных измерений, первичной обработки данных и обмена информацией, что обеспечивает высокую точность и эффективность тестирования печатных плат в автоматизированном режиме.

# Практическая часть

## Лазерный газоанализатор метана

Разрабатываемый в данном проекте тестовый стенд необходим для тестирования критически важных модулей и контрольных точек ПП лазерного газоанализатора, предназначенного для точного определения концентрации метана в воздухе с использованием специализированного алгоритма.

Внедрение стенда позволит обеспечить стабильное качество производства и сократить время на диагностику плат.

Устройство ПП газоанализатора состоит из двух ключевых модулей — Плату аналого-цифрового модуля (АЦМ) и внешний оптический блок

Оптический блок содержит:

* Два канала предусилителей: а) аналитический — регистрирует отраженное лазерное излучение; б) реперный — обеспечивает эталонный сигнал для калибровки;
* Лазерный модуль состоящий из: а) лазерный диод (источник излучения); б) элемент Пельтье (термостабилизация); в) термистор (контроль температуры);
* Оптические компоненты: объектив, коллиматор, оптоволоконные линии.

АЦМ включает следующие компоненты:

* Измерительный блок: а) три 18-битных АЦП для обработки сигналов реперного, аналитического и резервного каналов; б) один 16-битный АЦП, интегрированный в микроконтроллер (для термисторного канала); в) два 16-битных цифро-аналоговых преобразователей (управление током лазера и элемента Пельтье);
* Цифровая часть: а) программируемая логическая интегральная схема (ПЛИС) — управляет цифро-аналоговый преобразователь (ЦАП), собирает данные с АЦП и выполняет их обработку; б) МК — рассчитывает концентрацию метана и обеспечивает связь через интерфейсы Ethernet и RS-232;
* Интерфейсный модуль: а) разъемы и схемы согласования для подключения внешних устройств; б) индикаторы статуса питания, режима работы, состояния оптического блока и GPS.

## Методика разработки

Объектами разработки является прошивка для МК тестового стенда и ПО с графическим интерфейсом на ПК для управления прошивкой МК. Перед началом проведения работ нужно выбрать микропроцессор, который будет центральным звеном всего проекта. От его выбора зависит большое количество факторов, таких как:

* Производительность — тактовая частота, разрядность и архитектура (8, 16, 32 бит) определяют скорость обработки данных;
* Объём памяти – важно учитывать объём оперативной (RAM) и флеш-памяти для хранения данных и кода программы;
* Наличие и количество периферийных интерфейсов – поддержка UART, SPI, I2C, CAN, USB, Ethernet и других интерфейсов для связи с другими устройствами;
* Совместимость с выбранной средой разработки – поддержка компиляторов и отладочных инструментов, например, Keil, IAR, GCC, STM32CubeIDE;
* Доступность и стоимость – цена микропроцессора и его наличие на рынке;
* Поддержка аналоговых и цифровых входов/выходов – наличие ADC, DAC, GPIO и других встроенных модулей.

Правильный выбор микропроцессора определяет среду разработки, используемые библиотеки, а так же надёжность и эффективность работы тестового стенда и удобство его дальнейшей модернизации.

В проекте будет использован МК STM32F411CEU6 на готовой мини-отладочной плате «Black Pill». Средой разработки выбрана STM32CubeIDE. Прошивка будет производиться через программатор ST-LINK V2, через разъем подключения SWD. Для эффективной разработки прошивки в этой работе используется система контроля версий git, предоставляемое интернет-порталом «GitHub». Далее идет аргументирование выбора МК а так же описание всех сред разработок, в которых проводились работы.

### Выбор 32-разрядного микроконтроллера

Выбор 32-разрядного микроконтроллера вместо 8-разрядного или 16-разрядного обусловлен его высокой производительностью, большей памятью, поддержкой современных интерфейсов (USB, Ethernet, CAN), возможностью многозадачности и работы с RTOS, а также энергоэффективностью. Он позволяет обрабатывать данные быстрее, облегчает разработку сложных алгоритмов и даёт запас для масштабирования проекта. При этом стоимость 32-разрядных МК сопоставима с 8- и 16-разрядными, что делает их более предпочтительными для тестового стенда.

На сегодняшний день существует большое множество 32-битных микроконтроллеров, используемых в приборах различной сложности — от бытовых приборов до устройств с высокими вычислительными мощностями, которые нередко выпускаются в серийных или даже промышленных масштабах. Одним из самых популярных и зарекомендовавших себя семейств 32-битных микроконтроллеров являются чипы STM32 от компании STMicroelectronics. Эти микроконтроллеры широко применяются благодаря своей надежности, производительности и доступности.

Каждый микроконтроллер STM32 состоит из ядра процессора на базе архитектуры ARM с сокращённым набором инструкций (RISC), что обеспечивает высокую производительность при низком энергопотреблении. Кроме того, STM32 оснащены статической RAM-памятью, флеш-памятью для хранения программного кода, отладочными интерфейсами (такими как SWD и JTAG) и разнообразными периферийными модулями, что делает их универсальными для широкого спектра задач.

Так как в плате аналого-цифрового модуля лазерного детектора метана используется микроконтроллер модели STM32H743VIT, то разумно использовать микроконтроллер той же фирмы, в силу совместимости архитектур и периферии. Такой выбор позволит упростить разработку ПО тестового стенда. Но и несомненно благодаря благоприятной среде разработки, включающей такие инструменты, как STM32CubeIDE, STM32CubeMX и обширные библиотеки HAL (Hardware Abstraction Layer) и LL (Low layer), а также поддержку сообществом и доступность документации. Эти факторы значительно упрощают процесс разработки, отладки и внедрения проектов.

В рамках данной работы необходимо выбрать одну из линеек микроконтроллеров STM32, который обладает всей необходимой периферией для реализации поставленных задач. Проанализировав существующие серии STM32, я выделил линейку F4. Серия STM32F4 представляет собой высокопроизводительные микроконтроллеры, которые предлагают богатый набор периферии, большие объемы памяти и широкие возможности для реализации сложных проектов. Ядро Cortex-M4, используемое в этой серии, поддерживает набор инструкций DSP (цифровая обработка сигналов) и FPU (блок вычислений с плавающей точкой), что позволяет с запасом выполнить все поставленные задачи, включая обработку данных и управление сложными алгоритмами.

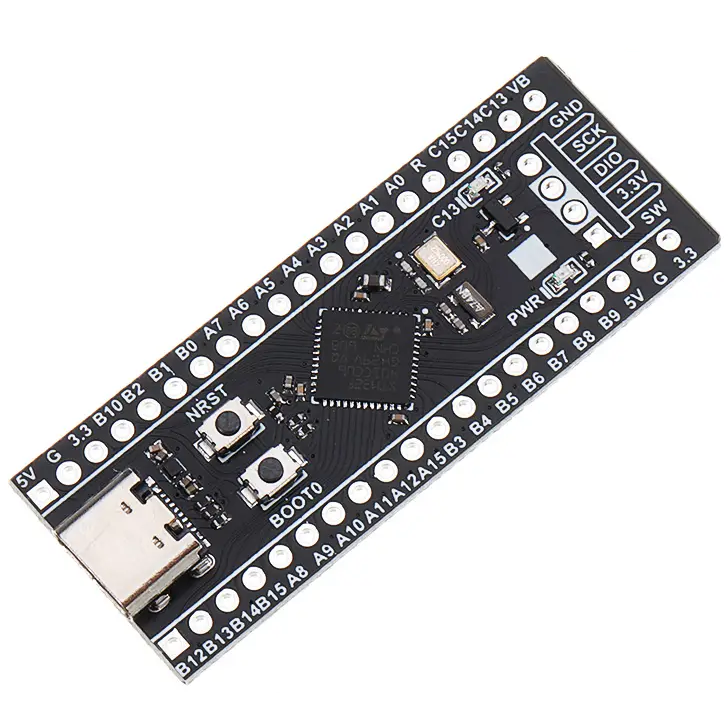
На борту микроконтроллеров серии F4 также присутствуют до 16 каналов 12-битного АЦП, что обеспечивает высокую точность измерений аналоговых сигналов, и необходимые коммуникационные интерфейсы, такие как USART, SPI, I2C, Ethernet и USB OTG. Эти интерфейсы позволяют легко интегрировать устройство в различные системы и обеспечивают гибкость в проектировании.

Однако для реализации текущего проекта не требуется всех мощностей, которые предлагает серия F4. Поэтому выбор пал на более компактный и экономичный вариант — микроконтроллер STM32F411CEU6. Этот чип сочетает в себе оптимальный баланс производительности, энергопотребления и стоимости. Он оснащен ядром Cortex-M4 с тактовой частотой до 100 МГц, что обеспечивает достаточную вычислительную мощность для выполнения задач проекта. Кроме того, STM32F411CEU6 имеет 512 КБ флеш-памяти и 128 КБ оперативной памяти, что достаточно для хранения программного кода и данных.

Важным преимуществом STM32F411CEU6 является его периферия: 12-битный АЦП с поддержкой до 16 каналов, несколько интерфейсов USART, SPI, I2C, а также USB OTG, что позволяет реализовать взаимодействие с внешними устройствами и передачу данных. Наличие аппаратной поддержки DMA (Direct Memory Access) значительно снижает нагрузку на процессор при работе с периферией, что особенно важно для задач, требующих высокой скорости обработки данных.

Кроме того, STM32F411CEU6 поддерживает различные режимы энергосбережения, что делает его подходящим для проектов, где важно минимизировать энергопотребление. Это особенно актуально для портативных устройств или систем, работающих от автономных источников питания.

Не так давно большим спросом пользовались мини-отладочные платы с чипом STM32F103C8T6 на базе процессора Cortex-M3, которые часто называют «Blue Pill» за счет их дешевизны. Такие платы были отличными вариантами для встраивания в проекты за счет компактного корпуса, в котором находились все выводы, разъем для питания USB и разъем JTAG для программирования производительного процессора. На сегодняшний день, актуальность данного модуля разработки упала, по причине нарастания мощностей микропроцессоров, поэтому на замену «Blue Pill» пришел его усовершенствованный вариант «Black Pill», с микроконтроллером STMF411CEU6. Сохранив те же компактные размеры, производитель поставил в плату высокопроизводительный чип, такой модуль имеет высокую популярность.



— Отладочная плата «Black Pill»

Таким образом, отладочная плата «Black Pill» с микроконтроллером STM32F411CEU6 полностью удовлетворяет всем требованиям проекта. Он обеспечивает необходимую производительность, обладает достаточным объемом памяти и поддерживает все необходимые интерфейсы. При этом его стоимость и энергопотребление находятся на оптимальном уровне, что делает его экономически выгодным выбором для реализации поставленных задач.

### Среды разработки

До начала разработки прошивки для МК требуется выбор среды разработки. Он зависит от совместимости с микроконтроллером, удобства работы, доступности инструментов отладки и поддержки необходимых библиотек. Среди доступных решений можно рассмотреть STM32CubeIDE, Keil MDK, IAR, Arduino IDE и другие. Выбор среды влияет на скорость разработки, удобство настройки периферии. Оптимальным решением для микроконтроллеров STM32 является STM32CubeIDE, так как это официальная бесплатная среда от STMicroelectronics, поддерживающая генерацию кода, встроенную отладку и работу с HAL/LL библиотеками и имеет гибкую систему отладки с поддержкой SWD/JTAG, что выгодно отличает её от более простых IDE.

**STM32CubeIDE —** комплексная платформа для разработки ПО, выпущенная компанией STMicroelectronics. Она предназначена для создания приложений под микроконтроллеры серии STM32 и объединяет в себе все необходимые инструменты: редактор, компилятор, отладчик и утилиты для тестирования. Решение полностью бесплатно и оптимизировано для проектов, где требуется точная работа с аппаратными компонентами.

Одна из сильных сторон среды — тесная интеграция с STM32CubeMX, утилитой для настройки периферийных модулей микроконтроллера. С её помощью можно графически задавать параметры тактовых сигналов, портов ввода-вывода, интерфейсов связи (UART, SPI, I2C) и других систем, после чего платформа автоматически формирует базовый код на языке C. Это ускоряет настройку проектов и снижает риск ошибок при ручном написании конфигураций.

STM32CubeIDE совместима со всеми моделями STM32, включая устройства на архитектурах Cortex-M0, M0+, M3, M4, M7 и M33. Для компиляции используется инструментарий GNU (GCC), поддерживающий языки C и C++, что позволяет задействовать современные подходы в программировании. Встроенный отладчик работает с протоколами SWD и JTAG, обеспечивая пошаговое выполнение кода, мониторинг переменных, регистров и памяти. Дополнительные функции включают анализ производительности (замер времени выполнения операций) и трассировку, для которой потребуется внешний аппаратный модуль, например, ST-Link.

Преимущества данной платформы — минимальные требования к ручной настройке, доступ к низкоуровневым библиотекам (HAL и LL) и возможность быстрого старта за счёт автоматической генерации кода.

**Visual Studio** – мощная интегрированная среда разработки (IDE), созданная Microsoft, которая поддерживает множество языков программирования включая язык C, на базе которого проводилась разработка ПО. Visual Studio широко используется разработчиками для создания настольных, мобильных, веб- и облачных приложений, а также для разработки программного обеспечения для встраиваемых систем. Она предоставляет удобные инструменты для редактирования кода, отладки и тестирования, что делает процесс разработки более эффективным.

В данном проекте эта среда разработки используется для разработки протокола передачи данных между ПК и МК на языке C.

## Функционал тестового стенда

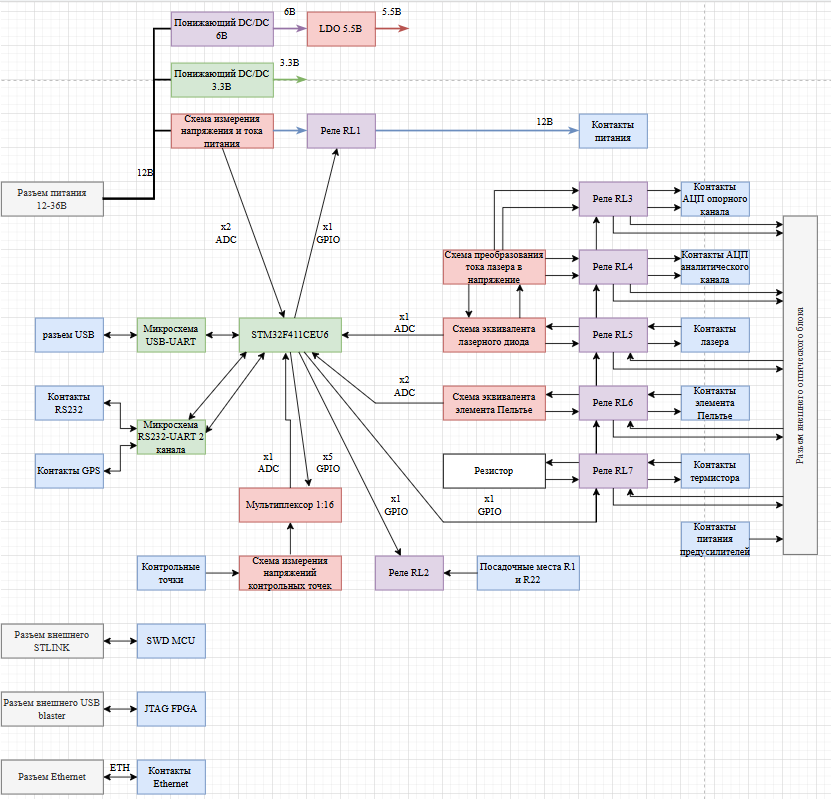
Перед написанием кода необходимо определить какие модули и контрольные точки необходимо тестировать на ПП газоанализатора, чтобы можно было утверждать, что плата исправна и не имеет дефектов, влияющих на ее работу. К таким функциям относится :

* Измерение напряжения в контрольных точках платы: 15 точек, напряжения от -6 до +6В;
* «Замыкание» посадочных мест резисторов R1 и R22;
* Прошивка ПЛИС;
* Прошивка МК;
* Подача и снятие питания прибора;
* Измерение напряжения и тока питания прибора;
* Подключение/отключение эквивалента лазерного модуля (лазерный диод, элемент Пельтье и термистор);
* Измерение тока через эквивалент лазерного диода;
* Преобразование тока лазера в напряжение для подачи на АЦП прибора;
* Измерение тока и напряжения на эквиваленте элемента Пельтье;
* Имитация GPS-модуля;
* Связь с ПК по интерфейсу USB-UART;
* Тестирование связи с прибором по интерфейсу RS232.

Для обеспечения выполнения вышеперечисленных функций, необходимо затратить ресурсы МК, такие как:

* 6 каналов АЦП: а) 2 канала для снятия напряжения и тока питания; б) 1 канал для снятие формы тока лазерного диода; в) 2 для снятия напряжения и тока потребления элемента Пельтье; г) 1 канал для измерения напряжения в 15 контрольных точках.
* 3 Универсальных асинхронных приёмопередатчиков (UART): а) 1 UART для микросхемы USB-UART; б) 2 UART для микросхемы RS232-UART (2 канала)
* 8 каналов интерфейса ввода/вывода общего назначения (GPIO): а) 3 канала для управления переключателями реле RL; б) 5 каналов для управления мультиплексором 1:16.
* Поддержка интерфейсов для прошивки: SWD (MCU) и JTAG (FPGA).

Таким образом, прошивка для МК включает в себя вышеперечисленную периферию. На рисунке 2.2 приведена функциональная схема платы тестового стенда.



— Функциональная схема платы тестового стенда

Функциональная схема (рисунок 2.2) тестового стенда для газоанализатора демонстрирует принцип работы, основанный на контроле, измерении и управлении различными модулями платы прибора с помощью микроконтроллера STM32F411CEU6. Питание стенда обеспечивается от внешнего источника с диапазоном напряжений 12–36 В. Это напряжение поступает на понижающий DC/DC-преобразователь, где понижается до 6 В. Далее формируются стабилизированные напряжения 5.5 В (через LDO) и 3.3 В (через дополнительный преобразователь), которые используются для питания логических цепей и самого микроконтроллера.

Подача питания на тестируемое устройство осуществляется через реле RL1, которое управляется через GPIO микроконтроллера. Для контроля состояния прибора реализовано измерение тока и напряжения питания с использованием двух каналов АЦП. Основные функции тестирования заключаются в следующем: измерение напряжений в контрольных точках, проверка цепей с посадочными местами резисторов R1 и R22, прошивка ПЛИС и микроконтроллера, управление питанием устройства, измерение потребляемых токов и напряжений, симуляция периферийных модулей (лазерного диода, элемента Пельтье, термистора), а также проверка интерфейсов связи (USB-UART, RS232, GPS).

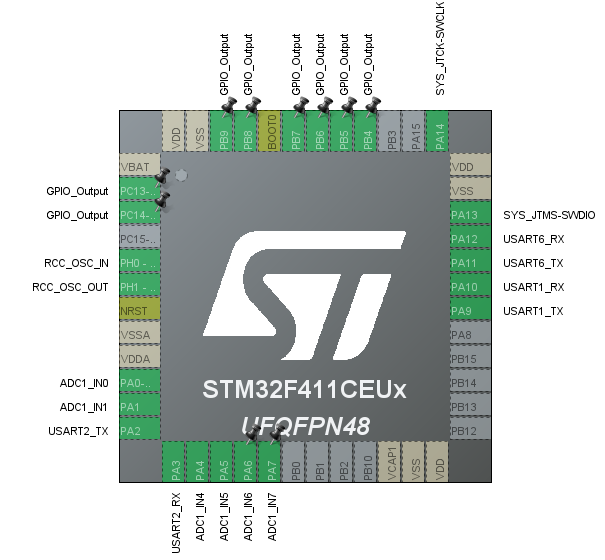
Для измерения напряжений в 15 контрольных точках используется мультиплексор 1:16, подключённый к одному каналу АЦП. Микроконтроллер поочерёдно переключает входы мультиплексора и считывает значения напряжений. Управление реле RL2 позволяет замыкать цепи резисторов R1 и R22, что необходимо для проверки их установки и контактирования. Для симуляции внешних компонентов прибора используются реле RL3–RL7. RL3 подключает опорный канал, RL4 – канал АЦП, RL5 – эквивалент лазерного диода, RL6 – элемент Пельтье, а RL7 – термистор. Все эти цепи позволяют провести полноценную проверку взаимодействия прибора с внешними модулями.

Для преобразования тока лазерного диода в напряжение, пригодное для подачи на АЦП, реализована специализированная схема, подключенная к отдельному каналу АЦП. Коммуникационные интерфейсы представлены USB-UART для связи с ПК, RS232 и разъёмом GPS, имитирующим работу соответствующего модуля. Кроме того, предусмотрена поддержка Ethernet и интерфейсы для программирования и отладки: SWD – для микроконтроллера и JTAG – для ПЛИС.

Таким образом, функциональная схема представляет собой многофункциональный, гибкий и автоматизированный стенд, обеспечивающий полную проверку электрических параметров, работоспособности интерфейсов и симуляции работы устройства в условиях, максимально приближенных к реальной эксплуатации.

## Конфигурация выводов микроконтроллера

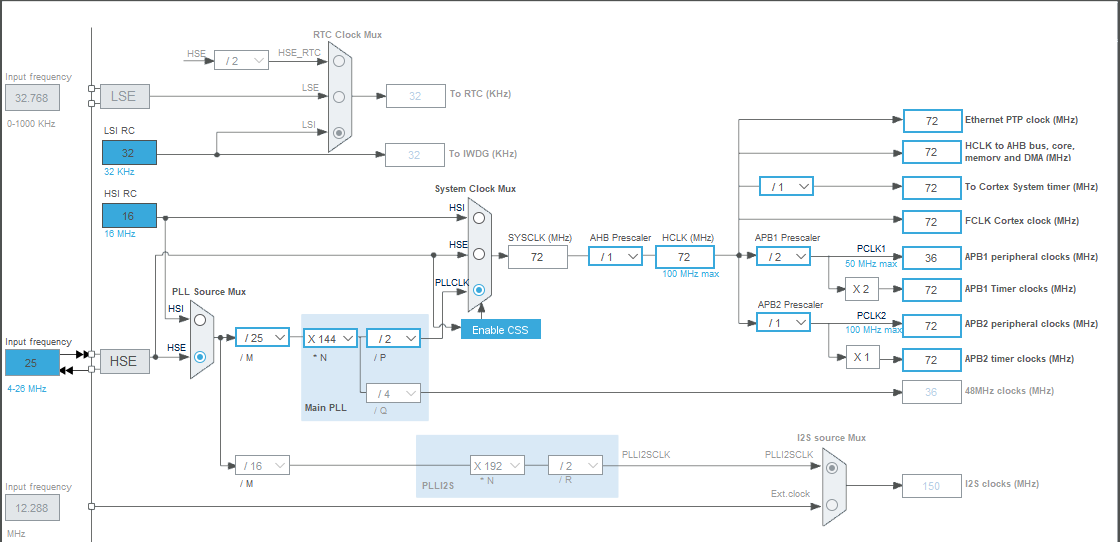
Для корректной работы проектируемого устройства необходимо выполнить настройку выводов микроконтроллера STM32F411CEU6 в соответствии с его функциональными задачами, описанными в предыдущем пункте, рисунок 2.3:



— Конфигурация выводов микроконтроллера

* Цифровые выходы управления (GPIO) — для подачи управляющих сигналов на исполнительные устройства такие как: реле и мультиплексор 1:16. Для этого задействованы следующие цифровые выводы: PB4, PB5, PB6, PB7, PB8, PB9, PC13, PC14 — настроены в режиме GPIO Output Push-Pull. Где PB4-PB8 — используются для управления мультиплексором 1:16. PB9, PC13 и PC14 — управляют состояниями реле.
* Аналоговые входы (ADC) — для измерения напряжений и токов на контрольных точках и эквивалентных схемах используется ADC1. Используются следующие выводы с функцией аналогового входа: ADC1\_IN0 (PA0), ADC1\_IN1 (PA1), ADC1\_IN4 (PA4), ADC1\_IN5 (PA5), ADC1\_IN6 (PA6), ADC1\_IN7 (PA7). Где ADC1\_IN0 — Для измерения формы тока лазерного диода, ADC1\_IN1 — подается на мультиплексор, ADC1\_IN4 и ADC1\_IN5 — для измерения напряжения и тока питания соответственно, ADC1\_IN6 и ADC1\_IN7 — для измерения падения напряжения элемента Пельтье.
* Интерфейс связи (UART) — для связи микроконтроллера с внешними устройствами через интерфейс UART используется USART1, USART2 и USART6. В данной конфигурации задействованы выводы: PA10 (USART1\_RX) и PA9 (USART1\_TX) — для UART-RS-232, PA2 (USART2\_TX) и PA3 (USART2\_RX) — для UART-USB, PA12 (USART6\_RX) и PA11 (USART6\_TX) — для UART\_GPS. Все 3 UART обеспечивают двунаправменную асинхронную последовательную передачу данных.
* Подключение отладчика ST-LINK V2 — для программирования и отладки через ST-LINK V2 используется интерфейс SWD (Serial Wire Debug): PA13 — SWDIO (SYS\_JTMS-SWDIO) и PA14 — SWCLK (SYS\_JTCK-SWCLK)
* Тактирование — тактовый генератор микроконтроллера работает от внешнего кварцевого резонатора с максимальной частотой 25 Мгц, установленного на отладочной плате «Black Pill». Резонатор подключается к выводам PH0 (RCC\_OSC\_IN) и PH1 (RCC\_OSC\_OUT), обеспечивая стабильную частоту тактирования для всей системы.

Назначение выводов и настройка частоты МК производилось в среде разработки STM32CubeIDE. В этой среде есть удобная программисту возможность визуально-наглядной настройки частоты отдельных элементов МК. Схема тактирования представлена на рисунке 2.4. Проект сгенерирован на языке C.



— Схема тактирования микроконтроллера

## Разработка прошивки микроконтроллера

Прошивка микроконтроллера в рамках данного проекта организована модульно. Такой подход позволяет не только облегчить разработку и отладку системы, но и обеспечивает наглядность структуры программы, её читаемость и удобство масштабирования в будущем.

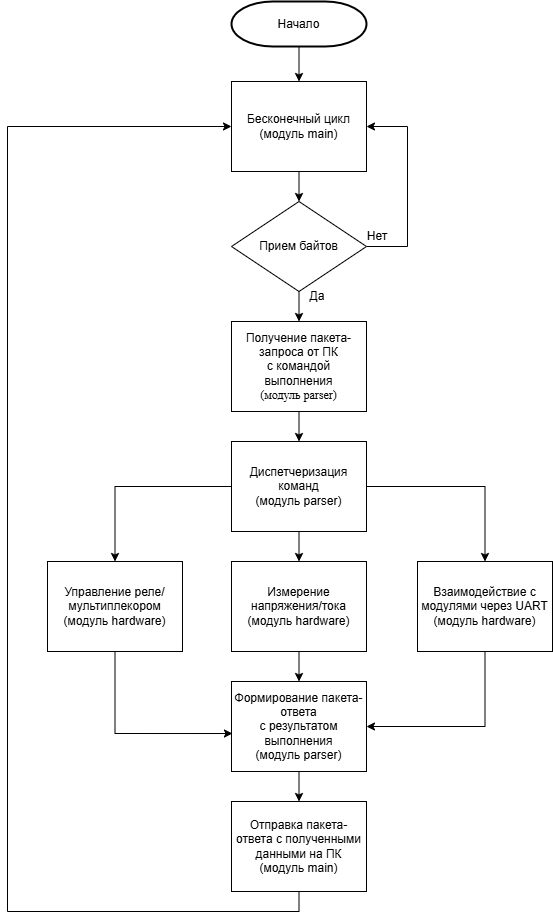
Структура программы построена на базе HAL (Hardware Abstraction Layer) библиотеки от STMicroelectronics, которая обеспечивает удобный и надёжный доступ к низкоуровневым функциям периферии, а так же были использованы библиотеки CMSIS и LL там, где можно оптимизировать код.

В проекте использовался компилятор GCC.

Логика работы микроконтроллера разделена на три основных модуля, каждый из которых выполняет строго определённый набор задач:

1. Модуль инициализации системы и управления основным циклом выполнения (модуль main);
2. Модуль обработки протокольных пакетов и диспетчеризации команд (модуль parser);
3. Модуль управления аппаратной логикой и взаимодействием с периферией (модуль hardware).

Каждый из этих модулей играет ключевую роль в общей архитектуре прошивки, рисунок 2.5. Далее подробный обзор их назначения и алгоритма работы.



— Обобщенная блок-схема алгоритма работы прошивки

## Модуль инициализации системы и управления основным циклом выполнения

Главный модуль main содержит точку входа программы и реализует базовую инициализацию всех используемых периферийных устройств: GPIO, UART, ADC, системного тактирования. Также он содержит основной бесконечный цикл, в котором происходит приём и обработка входящих UART-сообщений в асинхронном режиме, рисунок 2.6.

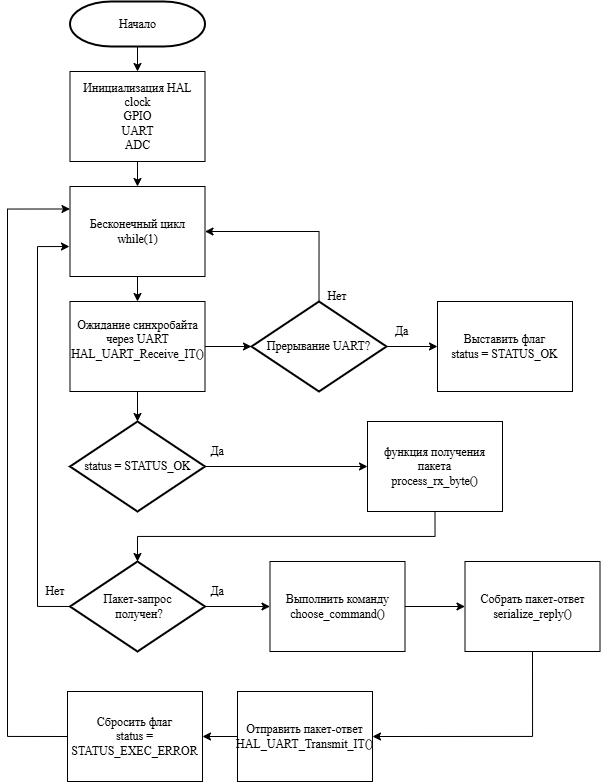
Программа начинается с вызова стандартной функции HAL\_Init(), которая настраивает базовые системные таймеры и прерывания. Затем вызывается функция SystemClock\_Config(), которая отвечает за настройку тактового генератора.

После настройки тактирования осуществляется инициализация всех необходимых периферий: UART (USART1, USART2, USART6), аналогово-цифрового преобразователя (ADC), а также цифровых входов/выходов (GPIO). Все эти шаги позволяют подготовить микроконтроллер к полноценной работе в режиме реального времени.

После завершения этапа инициализации программа входит в бесконечный цикл while(1), в рамках которого происходит основной обмен информацией с внешними устройствами. На каждом цикле вызывается функция HAL\_UART\_Receive\_IT(), инициирующая приём одного байта данных по интерфейсу UART в асинхронном режиме. Как только байт принят, вызывается функция обратного вызова (прерывание UART) HAL\_UART\_RxCpltCallback(), которая устанавливает флаг status в значение STATUS\_OK, сигнализируя основному циклу о готовности к обработке данных. Далее, если флаг установлен, вызывается функция process\_rx\_byte(), которая реализует поэтапный разбор входящего пакета по определённому протоколу.

Если все байты успешно разобраны и собран полный корректный пакет (функция возвращает PARSER\_DONE), вызывается функция choose\_command(), которая определяет, какую команду необходимо выполнить. На этом этапе производится вызов соответствующих функций из модуля аппаратной логики, например: включение реле, измерение напряжения, тестирование нагрузки и т.д. После вызывается функция serialize\_reply(), которая формирует окончательный пакет: добавляет заголовки, вычисляет контрольную сумму CRC и записывает всё это в буфер. Полученный пакет передаётся обратно по UART с помощью функции HAL\_UART\_Transmit\_IT().

В случае, если при разборе пакета была обнаружена ошибка (например, некорректная CRC или структура), функция process\_rx\_byte() возвращает PARSER\_ERROR, и в этом случае состояние парсера сбрасывается в STATE\_SYNC, чтобы подготовиться к приёму нового пакета. Независимо от результата обработки, в конце цикла значение переменной status принудительно сбрасывается, что предотвращает повторную обработку одного и того же байта.



— Блок-схема модуля инициализации системы и управления основным циклом выполнения

Исходный код модуля main приведен в приложении А

## Модуль обработки протокольных пакетов и диспетчеризации команд

Модуль parser отвечает за обработку принятых байтов, поступивших по UART в микроконтроллер от ПК через переходник UART-USB. Модуль проводит диспетчеризацию команд, а также формирует байты с результатом выполнения команд и измеренными данными для передачи по UART в ПК.

Для корректной обработки и передачи байтов, разработан протокол обмена данными между ПК и МК, представляющий собой простой, но надёжный механизм взаимодействия на основе последовательной передачи данных. Протокол обеспечивает выполнение команд по установке и получению параметров с последующим формированием ответов, а также контроль целостности пакетов при помощи контрольной суммы CRC16. Каждый пакет имеет чётко определённую структуру, состоящую из синхробайта, длины полезной части, кода команды, данных команды и контрольной суммы, таблица 2.1.

Коммуникация начинается с отправки запроса, содержащего один байт синхронизации (0xAA), за которым следуют два байта, обозначающих размер полезной части пакета, включающей код команды, возможные данные и два байта CRC. Далее следует сам код команды, который определяет тип запроса — например, получение значения параметра или его установка. Команда сопровождается данными, специфичными для данного типа запроса. В конце пакета передаётся CRC, рассчитанная на основе всех байтов начиная с кода команды.

|  |  |  |
| --- | --- | --- |
| * + - * 1. — Содержание пакета | | |
| Байты | Размер (в байтах) | Описание |
| 0 | 1 | Синхробайт (0xAA) |
| 1 – 2 | 2 | Размер полезных данных:   * код команды (1 байт); * данные команды (N байт в диапазоне [0;65529]; * CRC16 сумма (2 байта). |
| 3 | 1 | Код команды |
| 4 – … | N | Данные |
| (K-2) – (K-1) | 2 | CRC16 сумма |

K — размер пакета, максимум которого 65535 байт.

После получения корректного пакета микроконтроллер анализирует команду и в зависимости от её кода выполняет соответствующее действие, таблица 2.2. Например, при получении команды с кодом 0 осуществляется чтение значения определённого параметра (по его коду), а команда с кодом 1 используется для установки нового значения параметра. По завершении операции микроконтроллер формирует ответный пакет, структура которого аналогична запросу, таблица 2.3. В нём содержится оригинальный код команды, код ошибки (в случае успешного выполнения — 0), и, если необходимо, возвращаемое значение.

Система команд поддерживает как минимальные по объёму запросы (например, только код параметра), так и более расширенные — с установкой значений в виде 4-байтных слов. Размер передаваемых данных может достигать до 65535 байт, что позволяет передавать значительные объёмы информации, если потребуется.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| * + - * 1. — Набор команд | | | | | |
| Код команды | Описание | Данные команды | | | Размер данных (байт) |
| Байты | Размер (байт) | Описание |
| 0 | Получение параметра | 0 | 1 | Код параметра | 1 |
|  | | | | | |
| 1 | Установка параметра | 0 | 1 | Код параметра | 5 |
| 1 – 4 | 4 | Значение параметра |
|  | | | | | |
| В ответ на команду МК формирует пакет с кодом принятой команды (X) | | | | | |
| X | Ответ на команду | 0 | 1 | Код ошибки | 5 |
| 1 – 4 | 4 | Ответное значение |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| * + - * 1. — Набор ответов | | | | | |
| Код команды | Описание | Данные команды | | | Размер данных (байт) |
| Байты | Размер (байт) | Описание |
| 0 | Ответ на запрос параметра | 0 | 1 | Код ошибки | 5 |
| 1 – 4 | 4 | Значение параметра |
|  | | | | | |
| 1 | Ответ на установку параметра | 0 | 1 | Код ошибки | 5 |
| 1 – 4 | 4 | Значение параметра |

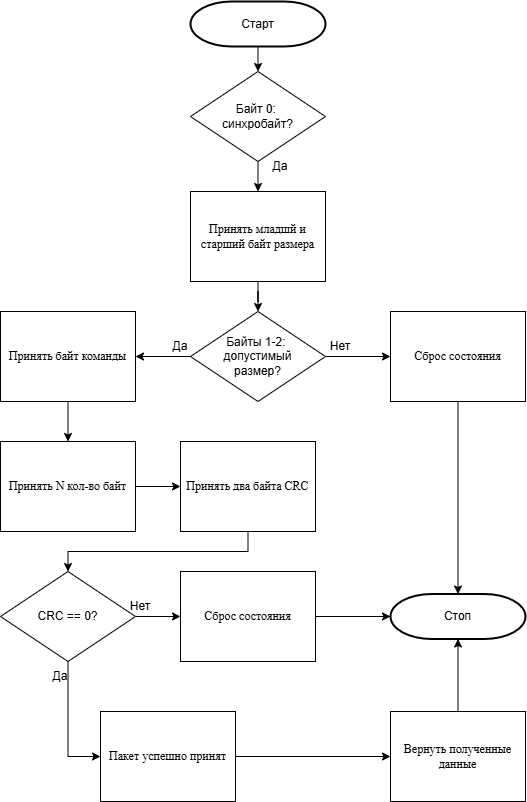
Также предусмотрен набор кодов ошибок, передаваемых в ответных пакетах, таблица 2.4. Это позволяет легко диагностировать сбои и реагировать на них в программной логике верхнего уровня.

|  |  |
| --- | --- |
| * + - * 1. — Коды ошибок | |
| Код ошибки | Описание |
| 0 | Нет ошибки |
| 1 | Ошибка выполнения команды |
| 2 | Несуществующая команда |
| 3 | Превышено время выполнения команды |
| 4 | Ошибка размера данных команды |

Пример для установки параметра в приложении А.

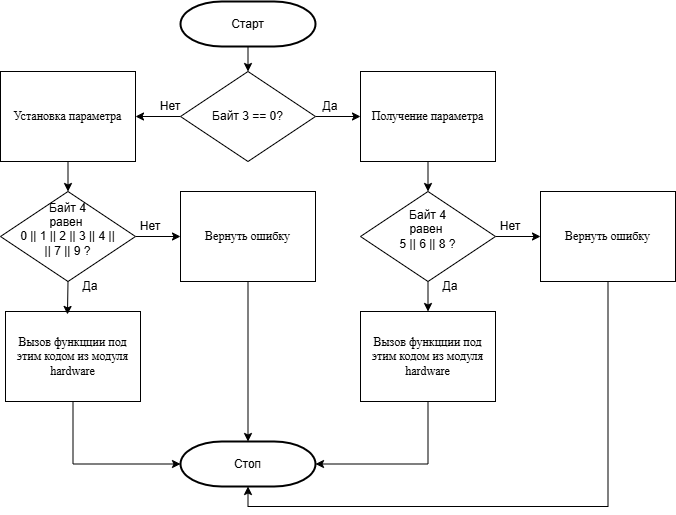
|  |  |  |  |
| --- | --- | --- | --- |
| * + - * 1. — Пример для установки параметра УБРАТЬ В ПРИЛОЖЕНИЕ | | | |
| Байт | Значение | Описание | |
| Запрос (AA 08 00 01 01 04 00 00 00 3D 3A) 11 Байт | | | |
| 0 | 0xAA | Синхробайт | |
| 1 | 0x08 | Младший байт размера полезных данных | В данном случаем = 8, т.к. Код команды – 1 байт, данные команды – 5 байт, CRC – 2 байта. |
| 2 | 0x00 | Старший байт размера полезных данных |
| 3 | 0x01 | Код команды (1 – установка параметра). | |
| 4 | 0x01 | Данные команды | Код параметра |
| 5 | 0x04 | Устанавливаемое значение. |
| 6 | 0x00 |
| 7 | 0x00 |
| 8 | 0x00 |
| 9 | 0x3D | Младший байт CRC | |
| 10 | 0x3A | Старший байт CRC | |
| Ответ (AA 08 00 01 00 02 00 00 00 00 72) 11 Байт | | | |
| 0 | 0xAA | Синхробайт | |
| 1 | 0x08 | Младший байт размера полезных данных | В данном случаем = 8, т.к. Код команды – 1 байт, данные команды – 5 байт, CRC – 2 байта. |
| 2 | 0x00 | Старший байт размера полезных данных |
| 3 | 0x01 | Код команды (1 – ответ на код 1) | |
| 4 | 0x00 | Байт ошибки (0 – ошибок нет) | |
| 5 | 0x02 | Четыре байта возвращаемого значения (от младшего к старшему байту).  Если не было ошибок возвращается значение установленного параметра, в данном случае - 2. | |
| 6 | 0x00 |
| 7 | 0x00 |
| 8 | 0x00 |
| 9 | 0x00 | Младший байт CRC | |
| 10 | 0x72 | Старший байт CRC | |

Блок схема алгоритма получения байтов через UART приведена на рисунке 2.7.



— Блок схема функции process\_rx\_byte()

Далее после успешного принятия всего пакета, вызывается функция ветвления choose\_command(), рисунок 2.8, в которой в зависимости от принятый байтов кода команды и кода параметра, вызывается модуль hardware, в котором происходят действия, указанные в коде команды и коде параметра. Оттуда возвращается статус выполнения команды, а так же полученный результат, на основе которых функция serialize\_reply() формирует-ответ, в который передает байт ошибки и полученные данные. После этого модуль main отправляет сформированный пакет по UART побайтно.



— Блок-смеха функции choose\_command()

Таким образом, протокол построен по принципу запроса-ответа с минимальной избыточностью и высокой гибкостью. Исходный код модуля parser приведен в приложении А.

## Модуль управления аппаратной логикой и взаимодействием с периферией

Модуль hardware реализует взаимодействие микроконтроллера с аппаратной частью устройства. Его основная задача — выполнение команд, полученных по протоколу, путём непосредственного управления реле, считывания аналоговых сигналов (напряжений и токов), а также обмена данными с внешними устройствами через интерфейсы UART.

Одной из ключевых функций модуля является управление реле. Это реализовано через функции apply\_voltage\_relay\_X(), где X — номер реле. В зависимости от команды (открыть или закрыть), происходит установка соответствующего пина GPIO, подключённого к обмотке реле. После активации производится проверка состояния пина, чтобы удостовериться, что реле сработало. Если всё прошло успешно, в выходной буфер возвращается статус выполнения OK, иначе — код ошибки.

Также в модуле реализованы функции измерения напряжений на контрольных точках. Например, функции test\_voltage\_4\_point() и test\_voltage\_11\_point() позволяют выбрать нужную точку путём переключения мультиплексора, после чего производится серия измерений напряжения с использованием АЦП. Усреднённое значение переводится в милливольты и записывается в выходной буфер. В похожем стиле реализовано измерение тока (test\_voltage\_current()), в котором напряжение на шунте пересчитывается в силу тока.

Отдельно реализована функция test\_voltage\_peltie(), предназначенная для диагностики тока, проходящего через модуль Пельтье. В ходе работы этой функции осуществляется измерение двух напряжений, рассчитывается разность и на её основе определяется направление и величина тока. Это позволяет отслеживать правильную работу термоэлектрического элемента.

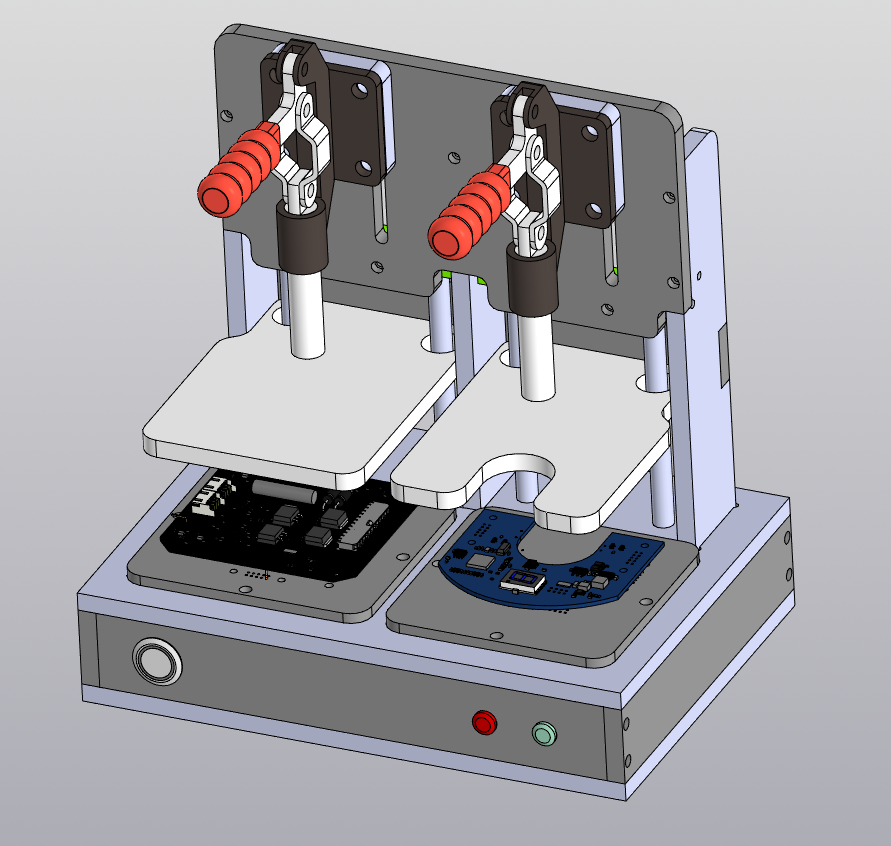
Модуль также содержит вспомогательные функции, такие как set\_pins() — для управления четырьмя цифровыми линиями, а также apply\_relay() — для подачи управляющего импульса на конкретный пин.

Немаловажной частью является реализация обмена по UART с внешними модулями. Например, функции massage\_rs232() и massage\_gps() осуществляют передачу и приём строк по соответствующим интерфейсам, после чего производится побайтовое сравнение отправленных и полученных данных. Если ответы совпадают, функция возвращает статус OK, иначе — сообщает об ошибке выполнения.

Таким образом, модуль hardware.c является «исполнителем» команд: он напрямую взаимодействует с физической частью устройства и выполняет всю низкоуровневую работу, оставляя другим модулям прошивки (parser и main) только обработку логики и структуры пакетов. Исходный код модуля parser приведен в приложении А.

## Конструкция тестового стенда

Общую концепцию тестового стенда можно понять по рисунку 2.3. Тестируемая печатная плата кладется на ложе погопинов (pogo-pin), подключенных к плате тестового стенда, и прижимается с помощью специального прижима (прижим будет один, а не два как на рисунках).



— Конструкция тестового стенда

Заключение

В результате выполнения выпускной квалификационной работы были достигнуты следующие результаты:

Список использованных источников



(необязательное)



(необязательное)



(необязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(необязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)



(обязательное)