

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего
образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация автоматизированных
систем специального назначения»
Специализация: «Эксплуатация вычислительных машин, комплексов,
систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ДИСПЛЕЙНЫЙ МОДУЛЬ СИСТЕМЫ УПРАВЛЕНИЯ
ТЕПЛОВОЗА

Студент Вен 11.06.24 Зелепухин Андрей Сергеевич
(подпись) (дата) (ФИО полностью)

Руководитель Б 11.06.24 Бицашев Р.А.
(подпись) (дата) (фамилия, инициалы)

Нормоконтролёр А 11.06.2024 Бычков А.С.
(подпись) (дата) (фамилия, инициалы)

Рецензент зав. каф. Программирования 14.06.2024 Морозов А.И.
(должность, место работы) ПензГУ (подпись) (дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой М Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой Отлично
(протокол заседания ГЭК от 19.06.2024 № 5)

Секретарь ГЭК Г Гурий Е.И.
(подпись) (фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
Пензенский государственный университет
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
Вычислительная техника

«Утверждаю»
Заведующий кафедрой
Митрохин М.А.
«19» апреля 2024 г.

ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Зелепухин Андрей Сергеевич гр. 19ВОЭ1 факультета вычислительной техники.

2. Тема работы: «Дисплейный модуль системы управления тепловоза».

Тема утверждена приказом ПГУ № 405/о от 19.04.2024 г.

3. Руководитель работы к.т.н., доцент каф. ВТ «ПГУ» Бикташев Р.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.).

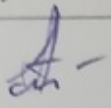
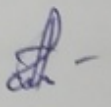
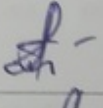
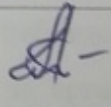
Назначение разработки – Программа предназначена для внедрения в дисплейный модуль системы управления тепловоза. В программе необходимо предусмотреть методы защиты устройства от экстремальных температур и потери питания, а также сбор и анализ ключевых параметров узлов транспорта.

Программа должна быть разработана с помощью среды STM32CubeIDE с помощью инструментов, имеющих лицензию свободного распространения.

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной области, определить функциональный состав программы, выбрать инструменты реализации программы. Необходимо разработать алгоритм работы программы, разработать UML диаграммы с последующим написанием программного кода.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	19.04.2024 – 26.04.2024	
2.	Выбор инструментов для разработки программы	10	26.04.2024 – 3.05.2024	
3.	Проектирование и составление UML диаграмм для реализации программы	20	3.05.2024 - 10.05.2024	
4.	Реализация программы	30	10.05.2024 - 31.05.2024	
5.	Оформление выпускной работы	30	31.05.2024 - 05.06.2024	

Дата выдачи задания 19 «апреля» 2024 года.

Руководитель дипломного проекта



19.04.24

Бикташев Р.А.

(подпись, дата)

Задание к исполнению принял студент



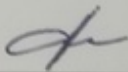
19.04.24

Зелепухин А.С.

(подпись, дата)

Работу к защите допустить

Декан факультета



Фионова Л.Р.

14.06.2024

Реферат

Работа содержит 111 листов, 12 рисунков, 13 таблиц, 8 источников, 3 приложения

ПРОГРАММА, МИКРОКОНТРОЛЛЕР, STM32, STM32CUBEIDE, C

Цель ВКР – разработка программы для микроконтроллера в составе дисплейного модуля системы управления тепловоза.

В работе произведен анализ и подбор инструментов разработки для реализации приложения. Выполнено проектирование, в результате которого были разработаны UML диаграммы разрабатываемого ПО.

Разработаны функции для сбора и фильтрации данных АЦП и датчиков; анализа состояния электропитания; управления состоянием вентиляторов, источника автономного питания и обогрева; управления яркостью подсветки экрана и клавиатуры; опроса состояния клавиш клавиатуры; передачи данных между микроконтроллером и материнской платой.

В ходе выполнения ВКР была разработана программа, которая удовлетворяет поставленной задаче.

В разделе безопасности жизнедеятельности были описаны требования к рабочему месту для минимизации воздействия вредных факторов на организм человека.

В экономическом разделе был произведен расчет затрат на разработку проекта.

Графическая часть выпускной квалификационной работы содержит UML-диаграммы.

					ПГУ 1.090501.03.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Зелепухин А.С.		11.06.24			
Провер.		Бикташев Р.А.		11.06.24			
Н. контр.		Бычков А.С.		11.06.24			
Утв.		Митрахин М.А.		11.06.24			
					Дисплейный модуль системы управления тепловоза		
					Лит.	Лист	Листов
						2	111
					ФВТ, каф. ВТ гр.19В031		

Содержание

Перечень сокращений и обозначений.....	6
Введение.....	7
1 Анализ предметной области.....	8
1.1 Анализ темы.....	8
1.2 Анализ аналогов.....	8
1.2.1 Многофункциональные дисплеи серии BC2050.....	9
1.2.2 Многофункциональные дисплеи серии BC2000 / BC2900.....	9
1.2.3 Многофункциональные дисплеи серии BC59xx.....	10
1.2.4 Многофункциональные дисплеи серии BC50xx.....	11
1.2.5 Многофункциональные дисплеи серии BC69xx.....	11
1.2.6 Многофункциональные дисплеи серии BC60xx.....	12
1.2.7 Многофункциональные дисплеи серии BC7.....	12
1.2.8 Результаты анализа аналогов.....	13
1.3 Постановка задачи.....	14
2 Выбор инструментов разработки.....	16
2.1 Выбор среды разработки.....	16
2.2 Выбор языка программирования.....	17
3 Архитектура приложения.....	19
3.1 Архитектура аппаратного решения.....	19
3.2 Выбор архитектуры приложения.....	20
3.3 Общий принцип работы программы.....	20
3.4 Описание пользовательских модулей.....	22
3.4.1 Модуль «adc».....	22
3.4.2 Модуль «brightness».....	23
3.4.3 Модуль «fan».....	24
3.4.4 Модуль «matrix_keyboard».....	25
3.4.5 Модуль «mp_communication_protocol».....	25

3.4.6 Модуль «polet_bat_handle».....	27
3.4.7 Модуль «power_control».....	28
3.4.8 Модуль «temperature_control».....	29
3.4.9 Модуль «warm_up».....	30
4 Разработка приложения.....	32
4.1 Разработка основной функции программы.....	32
4.1.1 Инициализация периферии.....	32
4.1.2 Инициализация модулей.....	34
4.1.3 Цикл обработки данных и принятия решений.....	39
4.2 Описание основных функций модулей.....	42
4.2.1 Модуль «adc».....	42
4.2.2 Модуль «brightness».....	44
4.2.3 Модуль «fan».....	45
4.2.4 Модуль «matrix_keyboard».....	46
4.2.5 Модуль «mp_communication_protocol».....	49
4.2.6 Модуль «polet_bat_handle».....	49
4.2.7 Модуль «power_control».....	50
4.2.8 Модуль «temperature_control».....	51
4.2.9 Модуль «warm_up».....	53
5 Техничко-экономическое обоснование.....	54
5.1 Расчет единовременных затрат на оборудование.....	54
5.2 Расчет затрат на заработную плату.....	55
5.3 Общий расчет затрат.....	55
5.4 Расчет стоимости устройства.....	55
6 Безопасность жизнедеятельности.....	57
6.1 Классификация вредных факторов.....	57
6.2 Требования к температуре.....	58
6.3 Требования к влажности воздуха.....	58
6.4 Требования к подвижности воздуха.....	58

6.5 Требования к электробезопасности.....	58
6.6 Требования к оборудованию рабочего места.....	59
Заключение.....	60
Список используемых источников.....	61
Приложение А – UML диаграммы.....	62
Приложение Б – Листинг программы.....	69
Приложение В – Презентация.....	105

Перечень сокращений и обозначений

В данном проекте применяются следующие сокращения и обозначения

АЦП – аналого-цифровой преобразователь

Дисплейный модуль – дисплейный модуль системы управление тепловоза

ИАП – источник автономного питания

НДФЛ – налог на доходы физических лиц

ПК – персональный компьютер

ПКИ – покупные комплектующие изделия

ПО – программное обеспечение

РФ – Российская Федерация

СанПиН – санитарно-эпидемические нормы

ТЭО – технико-экономическое обоснование

UML (Unified Modeling Language) – унифицированный язык моделирования,
язык графического описания для объектного моделирования

					ПГУ 1.090501.03.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Важным вопросом любого материального производства является вопрос транспортировки материалов и готовой продукции. Ведущее положение в грузообороте занимает железнодорожный транспорт. Примером этому служит статистика различных видов грузооборота транспорта в РФ за 2023 год [1], в котором из 5,5 трлн т-км грузооборот железнодорожного транспорта составил 2,6 трлн т-км (рисунок 1).

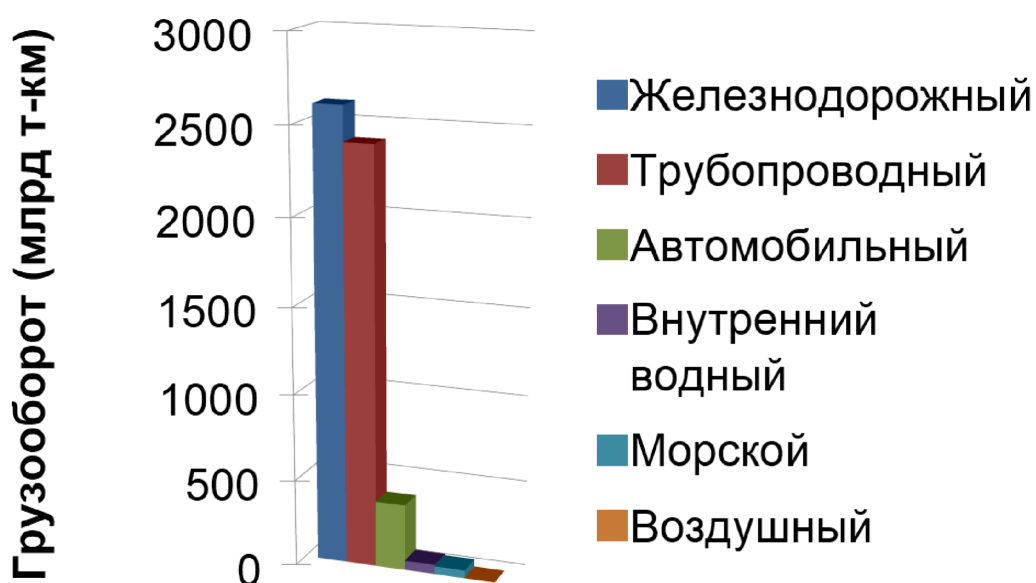


Рисунок 1 – Статистика грузооборота транспорта в РФ за 2023 год

Ключевой проблемой эксплуатации железнодорожного транспорта оператором является необходимость анализа множества параметров ключевых узлов локомотива. Для упрощения данной задачи служат бортовые промышленные компьютеры, осуществляющие сбор, анализ и вывод рабочих параметров в режиме реального времени.

1 Анализ предметной области

1.1 Анализ темы

Наиболее важными функциями бортового компьютера являются:

- сбор, анализ и вывод параметров ключевых узлов транспорта;
- вывод предупредительных сообщений при возникновении поломок или выхода значений параметров за пределы их нормальных значений.

Сбор, анализ и отображение параметров ключевых узлов транспорта позволяют снизить нагрузку на оператора, тем самым сделав работу более комфортной.

Своевременное отображение предупредительных сообщений о неисправностях позволяет принять соответствующие меры для минимизации повреждения транспорта, что, в долгосрочной перспективе, уменьшает денежные затраты на обслуживание транспорта.

Из вышеперечисленного следует, разрабатываемый бортовой компьютер будет иметь спрос на любых предприятиях, прямо занимающихся железнодорожной транспортировкой.

1.2 Анализ аналогов

Единственным общераспространенным аналогом разрабатываемого устройства являются многофункциональные дисплеи, разработанные немецкой компанией GERSYS.

Ниже представлены характеристики различных серий многофункциональных дисплеев GERSYS, выполняющих функции, аналогичные разрабатываемому устройству. Особое внимание стоит уделить характеристикам многофункциональных дисплеев серии BC7, так как дисплейный модуль планируется внедрить на российский рынок в качестве прямого конкурента данного изделия.

1.2.1 Многофункциональные дисплеи серии BC2050

Характеристики многофункционального дисплея серии BC2050 представлены в таблице 1.

Таблица 1 – Характеристики дисплея серии BC2050

Характеристика	Значение характеристики
Процессор	ЕARM Cortex A9 800 МГц, 32 бит
Клавиатура	Клавиатура (32 клавиши) Сенсорный экран
Дисплей	7" TFT дисплей, WVGA 800x480 Автоматическая система регулирования яркости подсветки
Диапазон температуры окружающего воздуха	От -40 до +70
Интерфейсы	Ethernet 10/100 МБит, USB 2.0
Входное питание	24 В - 110 В (+/- 40%)
Габариты	235x158x47 мм

К недостаткам данного изделия относятся слабый процессор и малое количество доступных интерфейсов. Однако, в отличие от большинства других серий, серия BC2050 имеет преимущество в виде встроенной клавиатуры, что позволяет оператору работать с дисплеем в холодную погоду без необходимости снимать перчатки [2].

1.2.2 Многофункциональные дисплеи серии BC2000 / BC2900

Характеристики многофункциональных дисплеев серий BC2000 / BC2900 представлены в таблице 2.

Таблица 2 – Характеристики дисплея серий BC2000 / BC2900

Характеристика	Значение характеристики
Процессор	X86 Intel ATOM E3825 2 ядра, 1,33 ГГц
Клавиатура	Клавиатура (32 клавиши) Сенсорный экран
Дисплей	8,4" TFT дисплей, VGA 640x480 Автоматическая система регулирования яркости подсветки
Диапазон температуры окружающего воздуха	От -40 до +70

Характеристика	Значение характеристики
Интерфейсы	RS422/RS485, RS232, Ethernet, USB, Mini-PCle
Входное питание	24 В - 110 В (+/- 40%)
Габариты	275x192x66 мм

Дисплеи серий ВС2000 / ВС2900 выделяются на фоне дисплеев серии ВС2050 за счет более быстродействующего процессора и наличия большего количества доступных интерфейсов. Также, дисплеи данной серии, как и дисплеи серии ВС2050 имеют встроенную клавиатуру [3].

1.2.3 Многофункциональные дисплеи серии ВС59хх

Характеристики многофункционального дисплея серии ВС59хх представлены в таблице 3.

Таблица 3 – Характеристики дисплея серии ВС59хх

Характеристика	Значение характеристики
Процессор	X86 Intel Atom Bay Trail 4 ядра, 1,9 ГГц
Клавиатура	Сенсорный экран
Дисплей	12,1" TFT дисплей, XGA 1024x768 Автоматическая система регулирования яркости подсветки
Диапазон температуры окружающего воздуха	От -40 до +70
Интерфейсы	RS422/RS485, RS232, Ethernet, USB, Mini-PCle
Входное питание	24 В - 110 В (+/- 40%)
Габариты	360x240x91 мм

Дисплеи данной серии отличаются от рассмотренных ранее в первую очередь быстродействием процессора. Также, немаловажным фактором является наличие интерфейсов RS422/RS485 и RS232, часто применяемых в промышленных системах. К явным минусам стоит отнести отсутствие клавиатуры [4].

1.2.4 Многофункциональные дисплеи серии BC50xx

Характеристики многофункционального дисплея серии BC50xx представлены в таблице 4.

Таблица 4 – Характеристики дисплея серии BC50xx

Характеристика	Значение характеристики
Процессор	ARM Cortex A9 4 ядра, 800 МГц
Клавиатура	Сенсорный экран
Дисплей	12,1" TFT дисплей, XGA 1024x768 Автоматическая система регулирования яркости подсветки
Диапазон температуры окружающего воздуха	От -40 до +70
Интерфейсы	RS422/RS485, RS232, Ethernet, USB, Mini-PCIe
Входное питание	24 В - 110 В (+/- 40%)
Габариты	360x240x49 мм

Дисплеи данной серии по своим преимуществам и недостаткам похожи на дисплеи серии BC59xx, за исключением дополнительного недостатка в виде слабого быстродействия процессора [4].

1.2.5 Многофункциональные дисплеи серии BC69xx

Характеристики многофункционального дисплея серии BC69xx представлены в таблице 5.

Таблица 5 – Характеристики дисплея серии BC69xx

Характеристика	Значение характеристики
Процессор	X86 Intel Atom Bay Trail 4 ядра, 1,9 ГГц
Клавиатура	Сенсорный экран
Дисплей	15" TFT дисплей, XGA 1024x768 Автоматическая система регулирования яркости подсветки
Диапазон температуры окружающего воздуха	От -40 до +70
Интерфейсы	RS422/RS485, RS232, Ethernet, USB, Mini-PCIe
Входное питание	24 В - 110 В (+/- 40%)
Габариты	415x303x96 мм

Основные преимущества и недостатки дисплеев данной серии аналогичны преимуществам и недостаткам серии ВС59хх, за исключением больших габаритов дисплеев серии ВС69хх. Данная особенность не может быть отнесена к недостаткам, однако, она однозначно ограничивает область применения данного многофункционального дисплея [5].

1.2.6 Многофункциональные дисплеи серии ВС60хх

Характеристики многофункционального дисплея серии ВС60хх представлены в таблице 6.

Таблица 6 – Характеристики дисплея серии ВС60хх

Характеристика	Значение характеристики
Процессор	ARM Cortex A9 4 ядра, 800 МГц
Клавиатура	Сенсорный экран
Дисплей	15" TFT дисплей, XGA 1024x768 Автоматическая система регулирования яркости подсветки
Диапазон температуры окружающего воздуха	От -40 до +70
Интерфейсы	RS422/RS485, RS232, Ethernet, USB, Mini-PCle
Входное питание	24 В - 110 В (+/- 40%)
Габариты	415x303x54 мм

Основные преимущества и недостатки дисплеев данной серии аналогичны дисплеям серии ВС69хх [5].

1.2.7 Многофункциональные дисплеи серии ВС7

Характеристики многофункционального дисплея серии ВС7 представлены в таблице 7.

Таблица 7 – Характеристики дисплея серии ВС7

Характеристика	Значение характеристики
Процессор	Е3940 4 ядра, 1,8 ГГц
Клавиатура	Клавиатура (3 клавиши) Сенсорный экран
Дисплей	10,4" TFT дисплей, VGA 640x480 Автоматическая система регулирования яркости подсветки
Диапазон температуры окружающего воздуха	От -25 до +70 (версия Т3) От -40 до +70 (версия Тх)
Интерфейсы	Ethernet, USB, RS232, RS422, MVB, CAN, Audio
Входное питание	24 В - 110 В (+/- 40%)
Габариты	310x214x45 мм

Основные преимущества и недостатки дисплеев данной серии аналогичны дисплеям серии ВС59хх. Отличительной особенностью данного устройства является наличие 3 встроенных клавиш. Данные клавиши позволяют оператору взаимодействовать с устройством при низких температурах не снимая перчаток, однако, не позволяют осуществлять управление устройством в полной мере [6].

1.2.8 Результаты анализа аналогов

Сравнение характеристик дисплейного модуля с характеристиками дисплеев серии ВС7 представлено в таблице 8.

Таблица 8 – Сравнение характеристик дисплейного модуля с характеристиками дисплеев серии ВС7

Наименование характеристики	Характеристики серии ВС7	Характеристики дисплейного модуля
Процессор	Е3940 4 ядра, 1,8 ГГц	Intel Celeron J1900 4 ядра, 2.42 ГГц
Клавиатура	Клавиатура (3 клавиши) Сенсорный экран	Клавиатура (27 клавиш)
Дисплей	10,4" TFT дисплей, VGA 640x480 Автоматическая система регулирования яркости подсветки	10,4" TFT LCD дисплей, 640x480 Автоматическая система регулирования яркости подсветки
Диапазон температуры	От -25 до +70 (версия Т3) От -40 до +70 (версия Тх)	От -40 до +70

Наименование характеристики	Характеристики серии BC7	Характеристики дисплейного модуля
окружающего воздуха		
Интерфейсы	Ethernet, USB, RS232, RS422, MVB, CAN, Audio	RS422/RS485, RS232, Ethernet, CAN, USB, Audio
Входное питание	24 В - 110 В (+/- 40%)	36-160 В Источник автономного питания
Габариты	310x214x45 мм	310x214x101

В результате анализа вышеперечисленных характеристик был сделан вывод о том, что дисплейный модуль не уступает аналогичным продуктам. Также, устройство выгодно выделяется на фоне аналогов наличием в своем составе ИАП, позволяющего корректно завершить работу устройства в условиях повреждения линии питания.

Разработка данного устройства является актуальной и перспективной деятельностью, так как спрос на подобные устройства велик, а аналоги, производимые на российском рынке, отсутствуют.

1.3 Постановка задачи

Задачей данного дипломного проекта является разработка программы для управления работой дисплейного модуля. Данная программа будет исполняться на микроконтроллере серии stm32f4. Приложение должно обладать следующим функционалом:

- сбор и фильтрация сигналов, поступающих на АЦП;
- управление уровнем яркости подсветки экрана и клавиатуры дисплейного модуля на основе показаний датчика, пользовательских настройках и текущих режимах электропитания и температуры;
- управление охлаждающими вентиляторами;
- сбор и анализ информации о нажатых клавишах клавиатуры;
- осуществление приема и передачи между микроконтроллером и материнской платой. При приеме данных в соответствии с полученным сообщением необходимо производить изменение пользовательских настроек, управляющих работой дисплейного модуля. При передаче данных необходимо

производить сбор и отправку наиболее важных параметров работы дисплейного модуля;

- управление работой ИАП;
- управление режимом электропитания устройства;
- сбор и анализ показаний температурного датчика;
- управление устройством подогрева.

					ПГУ 1.090501.03.001 ПЗ	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

2 Выбор инструментов разработки

Основными требованиями к выбору среды разработки при выполнении данной работы выбраны:

- наличие инструмента для настройки периферии устройства;
- наличие встроенных библиотек для работы с микроконтроллерами серии stm32;
- наличие встроенного отладчика.

Основными требованиями к выбору языка программирования при выполнении данной работы выбрана поддержка данного языка выбранной средой разработки.

2.1 Выбор среды разработки

Для разработки программного обеспечения для микроконтроллеров семейства stm32 чаще всего используются следующие среды разработки: STM32CubeIDE и Keil MDK [7].

Среда разработки STM32CubeIDE является официальной средой разработки от компании STMicroelectronics и поддерживает языки программирования C и C++. Среда предоставляет широкий набор инструментов, в который входят:

- встроенная библиотека для работы с микроконтроллерами семейства stm32;
- поддержка графического инструмент генерации проекта STM32CubeMX;
- встроенный инструмент отладки;
- инструмент для поиска документации по микроконтроллерам компании STMicroelectronics.

Среда разработки Keil MDK направлена на разработку программного обеспечения для микроконтроллеров на базе ARM Cortex-M и поддерживает языки программирования C и C++. Основными преимуществами среды являются:

- встроенные инструменты для отладки и имитации;

– поддержка библиотек для работы с микроконтроллерами семейства stm32 от компании STMicroelectronics.

Различия между рассматриваемыми средами разработки представлены в таблице 9.

Таблица 9 – Сравнение STM32CubeIDE и Keil MDK

Наименование характеристики	STM32CubeIDE	Keil MDK
Языки программирования	C, C++	C, C++
Инструмент для настройки периферии	+	-
Поддержка расширений среды	+	+
Наличие/Поддержка встроенных библиотек для работы с stm32	+	+
Наличие инструмента для поиска документации по контроллерам семейства stm32	+	-
Встроенный отладчик	+	+
Встроенный инструмент имитации	-	+

Проанализировав преимущества и недостатки представленных IDE, было принято решение выбрать STM32CubeIDE, так как данная среда предоставляет наиболее удобный набор инструментов для разработки и дальнейшего сопровождения проекта.

2.2 Выбор языка программирования

Для программирования микроконтроллеров семейства stm32 используются следующие языки программирования: C, C++. Сравнение данных языков представлено в таблице 10.

Таблица 10 – Сравнение языков С и С++

Наименование характеристики	С	С++
Класс языка	Процедурный	Объектно-ориентированный
Перегрузка функций и операторов	Поддерживается	Не поддерживается
Динамическое выделение памяти	Поддерживается	Поддерживается, имеет больше функций, чем в языке С

Проанализировав преимущества данных языков, было принято решение выбрать язык С. Данное решение обусловлено следующими причинами:

- проект не требует использования объектно-ориентированного подхода;
- проект не требует использования функций для динамического выделения памяти вследствие отсутствия использования объектно-ориентированного подхода и многозадачной операционной системы;
- программа на языке С имеет небольшие преимущества в плане скорости выполнения и размера кода [8].

3 Архитектура приложения

3.1 Архитектура аппаратного решения

Схематичное изображение взаимодействия частей дисплейного модуля изображено на рисунке 2. Пунктирными линиями обозначены устройства, сгруппированные по общему функциональному признаку. Стрелками «Управляющий сигнал» указаны потоки передачи сигналов, контролирурующих работу частей дисплейного модуля. Стрелками «Обмен данными» указаны потоки передачи всех остальных сигналов.

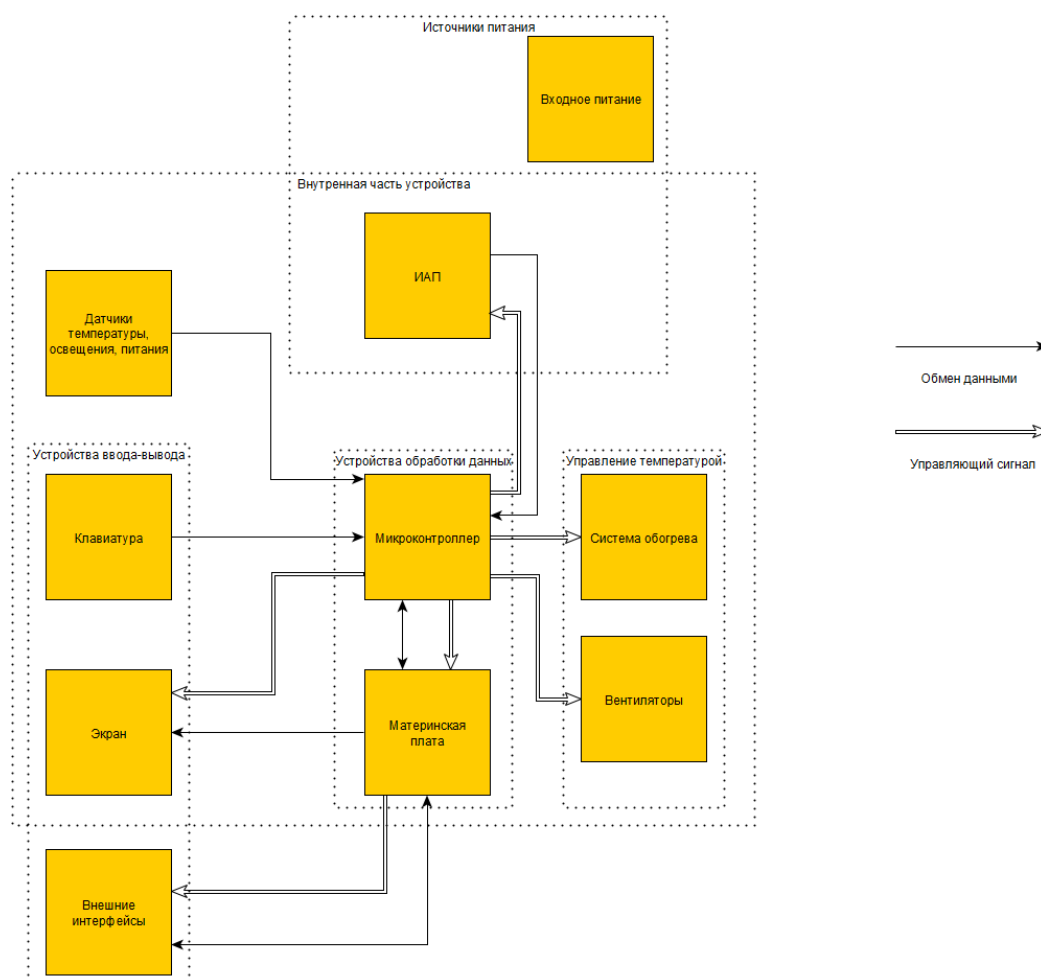


Рисунок 2 – Схематичное изображение взаимодействия частей дисплейного модуля

3.2 Выбор архитектуры приложения

В качестве архитектуры приложения выбрана монолитная архитектура. Монолитная архитектура – это вид структурной организации приложения, в которой все компоненты приложения крепко связаны между собой.

Данная архитектура была выбрана из-за следующих преимуществ перед другими архитектурами:

- простота отладки приложения, так как все действия происходят в одном потоке;
- высокая производительность приложения из-за отсутствия временных затрат на межпроцессорное взаимодействие;
- отсутствие сложных механизмов синхронизации из-за отсутствия межпроцессорного взаимодействия и совместно используемой среды данных.

3.3 Общий принцип работы программы

Проект, сгенерированный в среде STM32CubeIDE, по умолчанию содержит в себе файлы «main.c» и «stm32f4xx_it.c». Файл «main.c» содержит в себе функцию «main», являющуюся точкой входа основной программы и производящей вызов функций инициализации модулей программы в соответствии с настройками, заданными в STM32CubeMX. Файл «stm32f4xx_it.c» содержит в себе обработчики всех немаскируемых и включенных маскируемых прерываний.

При подаче питания на микроконтроллер происходит переход на начало функции «main» и выполнение функций инициализации различных модулей микроконтроллера: тактового генератора, выводов микроконтроллера, периферии. Далее производится настройка пользовательских модулей, направленных на выполнение различных функций, описанных в постановке задач. Данная настройка производится при помощи функций инициализации, содержащихся в пользовательских модулях. Затем при помощи функций обработки данных пользовательских модулей программа осуществляет управление устройством в бесконечном цикле.

Для обеспечения соблюдения временных интервалов между вызовами пользовательских функций используется проверка состояния флага. Данный флаг устанавливается в обработчике событий, который вызывается каждую миллисекунду.

Диаграмма деятельности функции main приведено на рисунке 3.

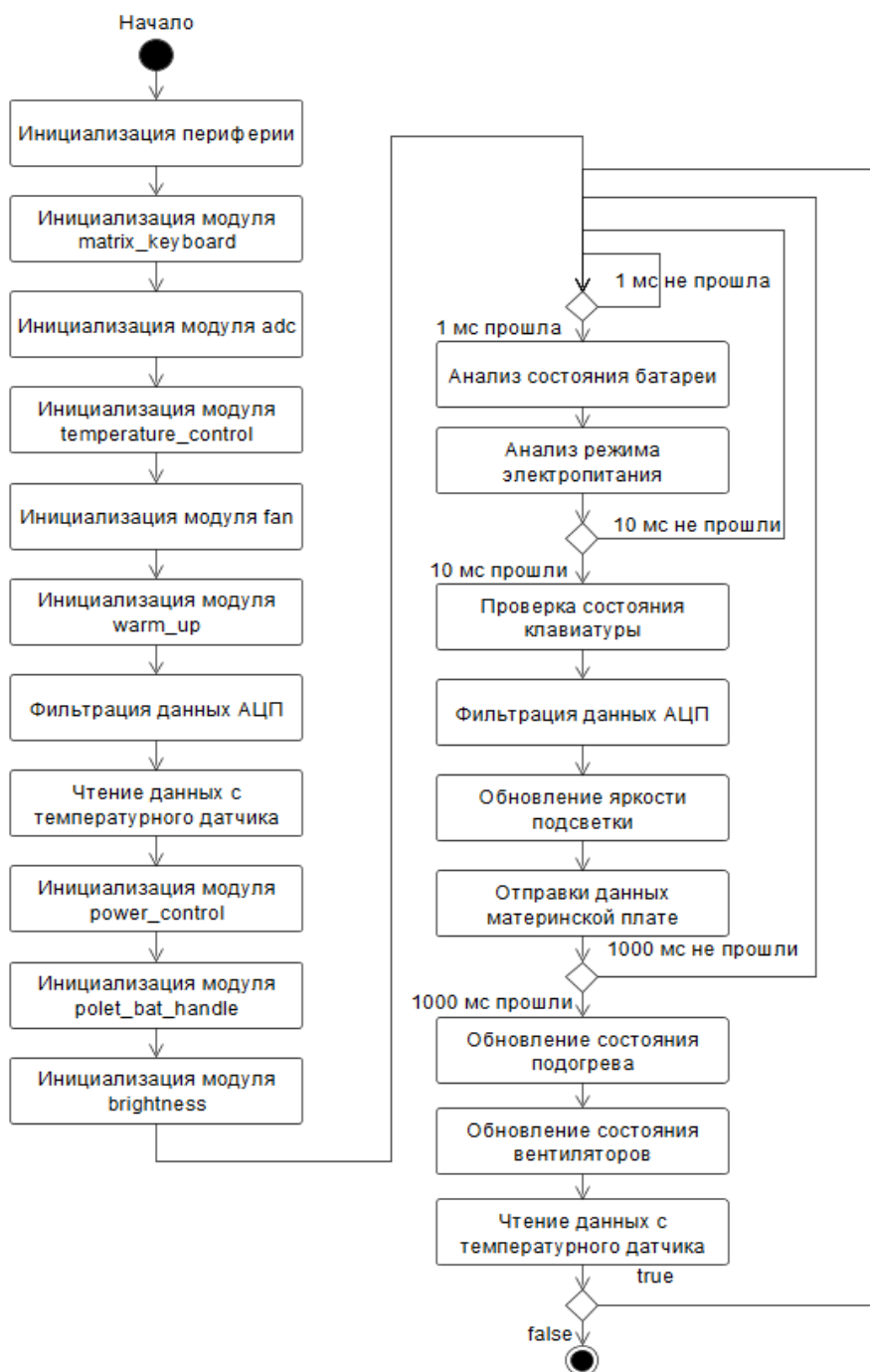


Рисунок 3 – Диаграмма деятельности: функция «main»

В случае срабатывания прерывания управление передается одному из обработчиков прерываний модуля «stm32f4xx_it.c». Обработчик прерываний вызывает необходимые для выполнения функции пользовательских модулей, после чего возвращает управление основной программе.

3.4 Описание пользовательских модулей

Пользовательские модули представляют собой исполняемые файлы, содержащие функции, направленные на работу конкретных частей устройства и обеспечивают выполнение пунктов, описанных в подразделе «постановка задач».

3.4.1 Модуль «adc»

Диаграмма компонентов для модуля «adc» приведена на рисунке 4.

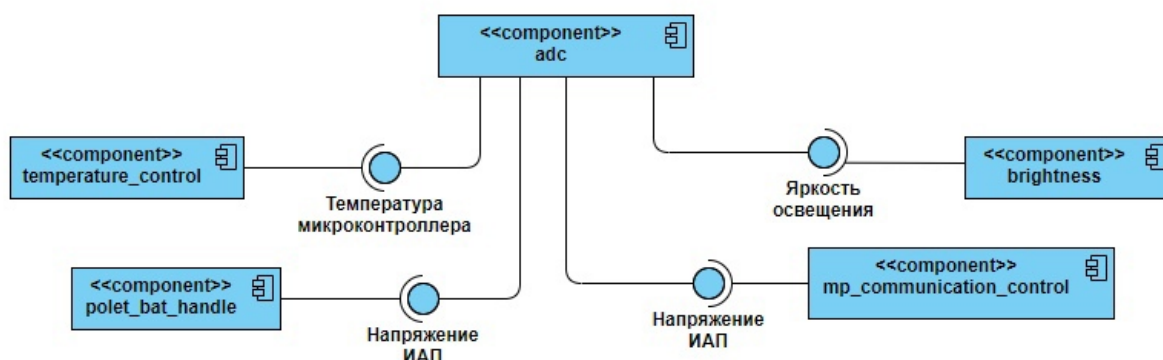


Рисунок 4 – Диаграмма компонентов: модуль «adc»

Модуль «adc» производит управление встроенным в микроконтроллер АЦП. К функциям данного модуля относятся:

- сбор данных с каналов АЦП;
- фильтрация данных, полученных с каналов АЦП.

Данный модуль передает другим модулям следующие данные:

- напряжение ИАП;
- яркость освещения, полученная с датчика освещения;
- температура, полученная со встроенного в микроконтроллер датчика температуры.

3.4.2 Модуль «brightness»

Диаграмма компонентов для модуля «brightness» приведена на рисунке 5.

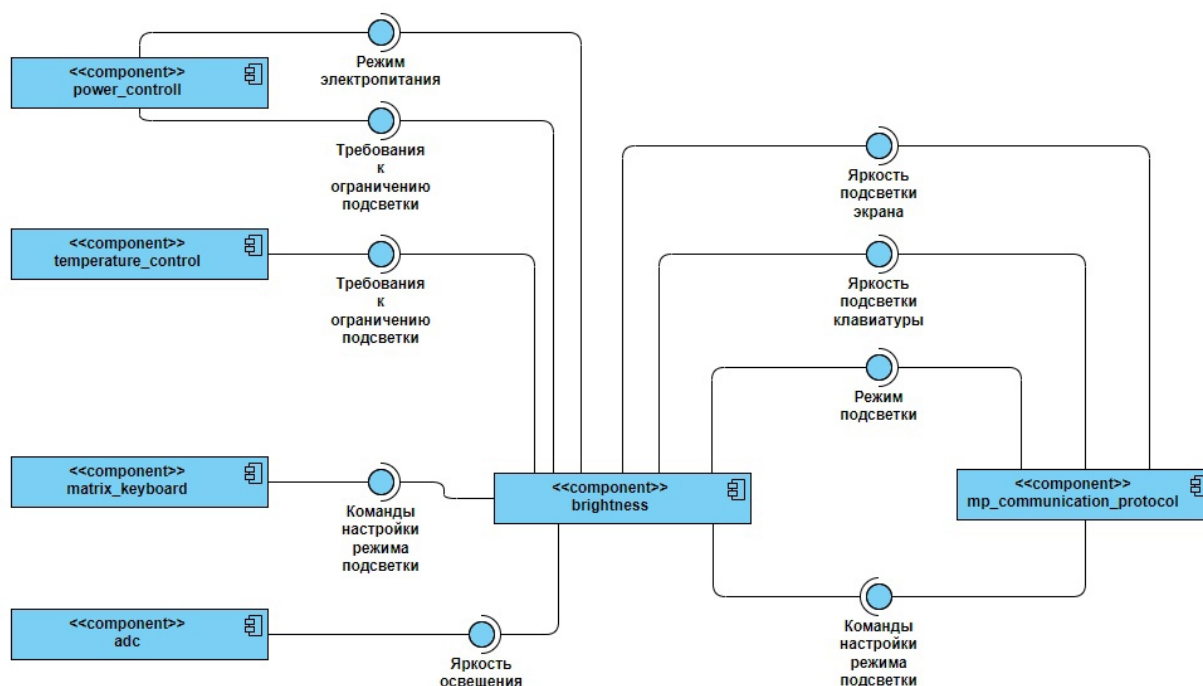


Рисунок 5 – Диаграмма компонентов: модуль «brightness»

Модуль «brightness» производит управление подсветкой экрана и клавиатуры. К функциям данного модуля относятся:

- обработка требований изменения текущего режима подсветки;
- анализ требований к ограничению максимальной яркости подсветки на основе электропитания и температуры;
- расчет требуемой яркости подсветки экрана и клавиатуры на основе показаний датчика освещения;
- плавное изменение яркости подсветки экрана и клавиатуры.

Данный модуль получает от других модулей следующие данные:

- данные с датчика освещения;
- команды настройки режима подсветки;
- требования к ограничению подсветки;

– режим электропитания.

Данный модуль передает другим модулям следующие данные:

- текущий режим подсветки;
- текущее значение яркости подсветки экрана;
- текущее значение яркости подсветки клавиатуры.

3.4.3 Модуль «fan»

Модуль «fan» производит управление вентиляторами. К функциям данного модуля относится анализ требований включения/отключения вентиляторов на основе температуры и электропитания;

Данный модуль получает от других модулей следующие данные:

- текущая температура;
- требования к ограничению работы вентиляторов.

Данный модуль передает другим модулям текущее состояние вентиляторов.

Диаграмма компонентов для модуля «fan» приведена на рисунке 6.

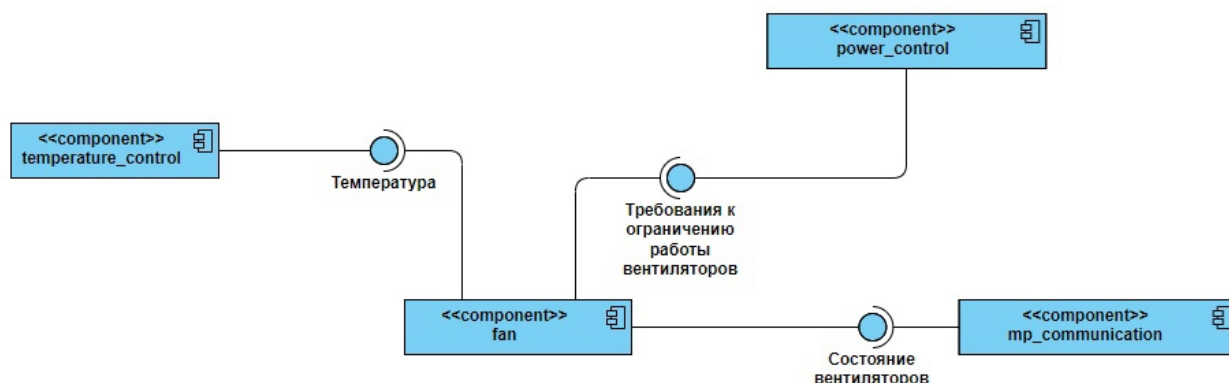


Рисунок 6 – Диаграмма компонентов: модуль «fan»

3.4.4 Модуль «matrix_keyboard»

Модуль «matrix_keyboard» производит управление матричной клавиатурой.

К функциям данного модуля относятся:

- анализ текущего состояния клавиш клавиатуры;
- выявление залипших клавиш клавиатуры;
- анализ требований к изменению режима электропитания на основе состояния клавиши управления питанием;
- анализ требований к изменению режима подсветки экрана на основе состояния клавиш управления подсветкой.

Данный модуль передает другим модулям следующие данные:

- команды настройки режима подсветки;
- команды настройки режима электропитания;
- комбинация нажатых клавиш, которую необходимо передать материнской плате.

Диаграмма компонентов для модуля «matrix_keyboard» приведена на рисунке 7.

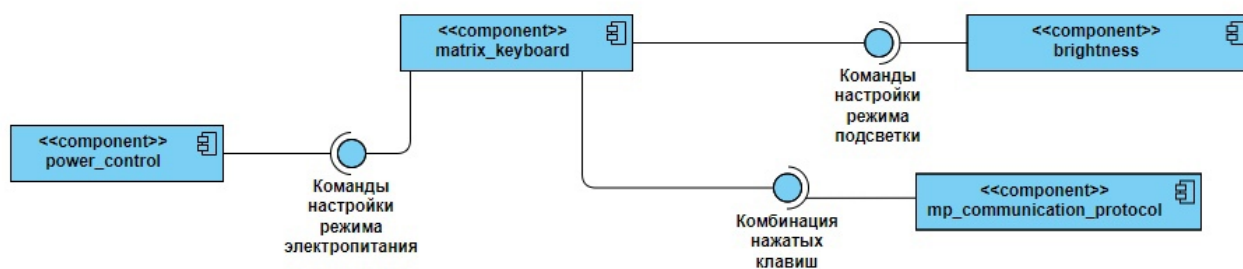


Рисунок 7 – Диаграмма компонентов: модуль «matrix_keyboard»

3.4.5 Модуль «mp_communication_protocol»

Диаграмма компонентов для модуля «mp_communication_protocol» приведена на рисунке 8.

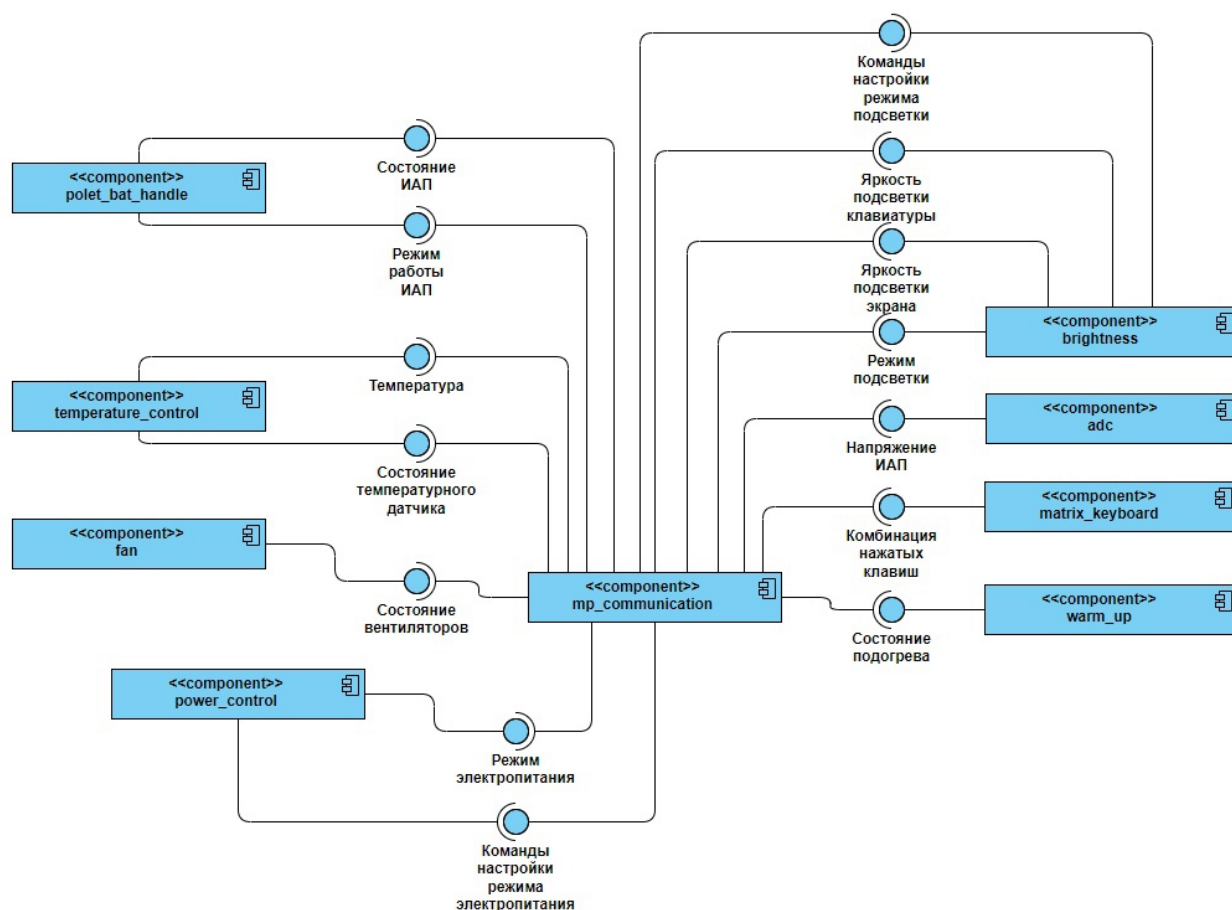


Рисунок 8 – Диаграмма компонентов: модуль «mp_communication_protocol»

Модуль «mp_communication_protocol» производит управление коммуникацией микроконтроллера с материнской платой. К функциям данного модуля относятся:

- передача параметров различных модулей устройства материнской плате;
- анализ требований изменения настроек устройства на основе сообщений, полученных от материнской платы.

Данный модуль получает от других модулей следующие данные:

- яркость подсветки экрана;
- яркость подсветки клавиатуры;
- текущий режим подсветки;
- комбинация нажатых клавиш клавиатуры;
- режим работы ИАП;

- состояние ИАП;
- режим электропитания;
- состояние подогрева;
- состояние вентиляторов;
- состояние температурного датчика;
- текущая температура;
- напряжение ИАП.

Данный модуль передает другим модулям следующие данные:

- команды настройки режима подсветки;
- команды настройки режима электропитания.

3.4.6 Модуль «polet_bat_handle»

Диаграмма компонентов для модуля «polet_bat_handle» приведена на рисунке 9.

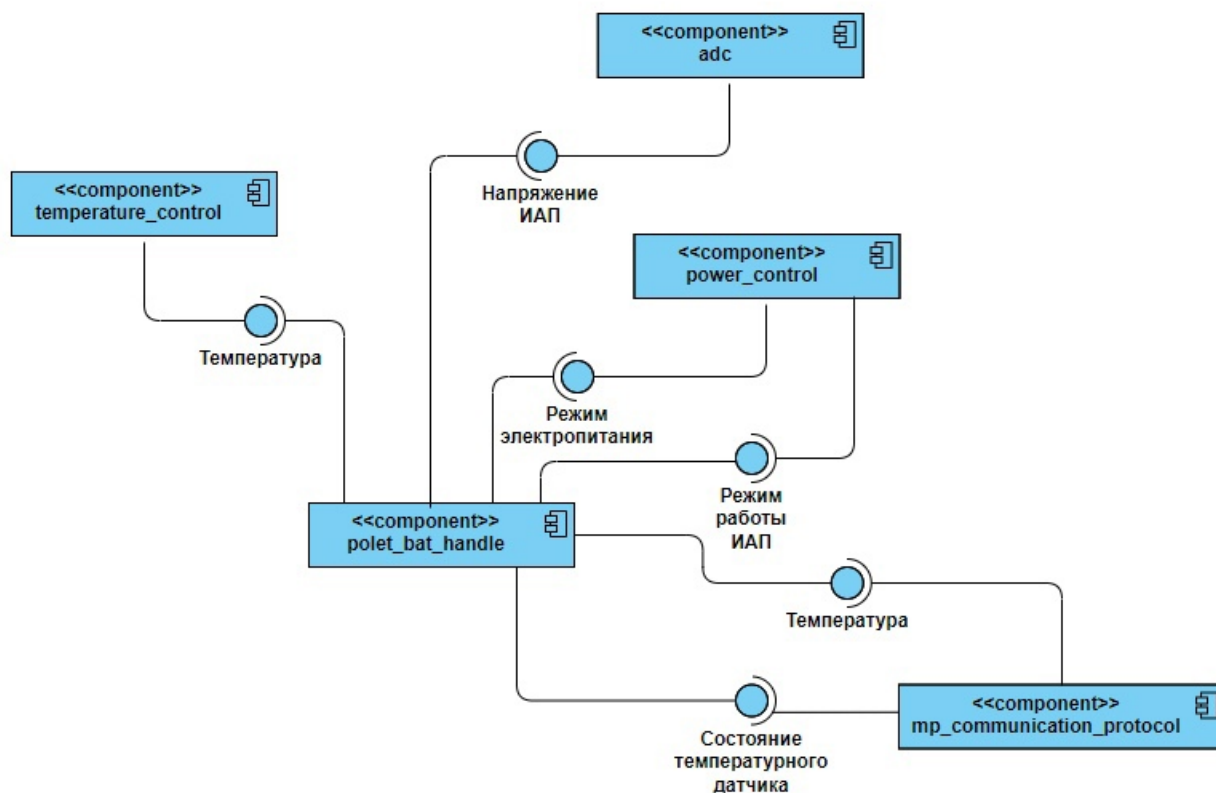


Рисунок 9 – Диаграмма компонентов: модуль «polet_bat_handle»

Модуль «polet_bat_handle» производит управление ИАП. К функциям данного модуля относятся:

- анализ исправности ИАП;
- перевод ИАП в режим зарядки;
- перевод ИАП в режим автономного питания.

Данный модуль получает от других модулей следующие данные:

- напряжение ИАП;
- текущее значение температуры;
- текущий режим электропитания.

Данный модуль передает другим модулям следующие данные:

- текущий режим работы ИАП;
- состояние ИАП.

3.4.7 Модуль «power_control»

Модуль «power_control» производит управление электропитанием устройства. К функциям данного модуля относятся:

- анализ наличия электропитания устройства;
- управление функциями энергосбережения при отсутствии электропитания.

Данный модуль получает от других модулей следующие данные:

- команды настройки режима электропитания;
- режим работы ИАП;
- температура.

Данный модуль передает другим модулям следующие данные:

- требования к ограничению подсветки;
- текущий режим электропитания;
- требования к ограничению подсветки;
- требования к ограничению работы вентиляторов;
- требования к ограничению работы подогрева.

Диаграмма компонентов для модуля «power_control» приведена на рисунке 10.

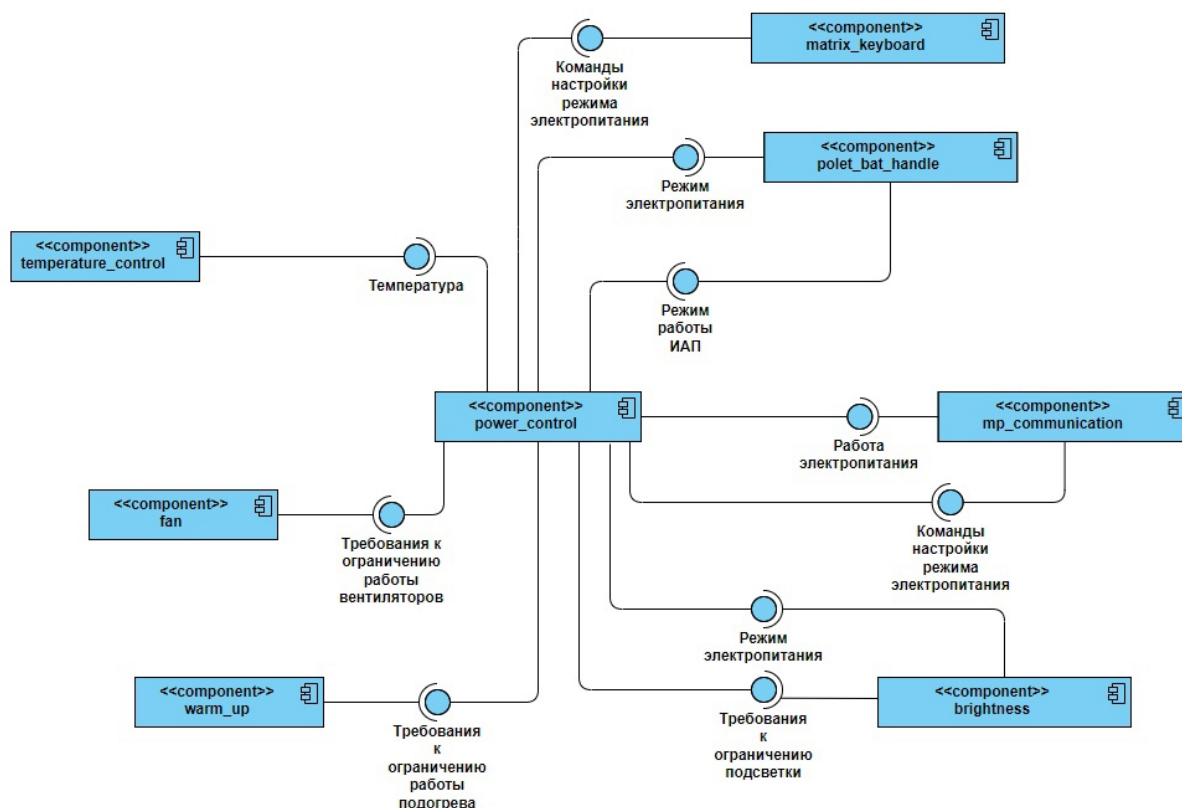


Рисунок 10 – Диаграмма компонентов: модуль «power_control»

3.4.8 Модуль «temperature_control»

Модуль «temperature_control» производит управление данными о температуре. К функциям данного модуля относятся:

- анализ состояния температурного датчика;
- опрос температурного датчика.

Данный модуль получает от других модулей показания со встроенного в микроконтроллер температурного датчика.

Данный модуль передает другим модулям следующие данные:

- текущее значение температуры;
- состояние температурного датчика;
- требования к ограничению подсветки.

Диаграмма компонентов для модуля «temperature_control» приведена на рисунке 11.

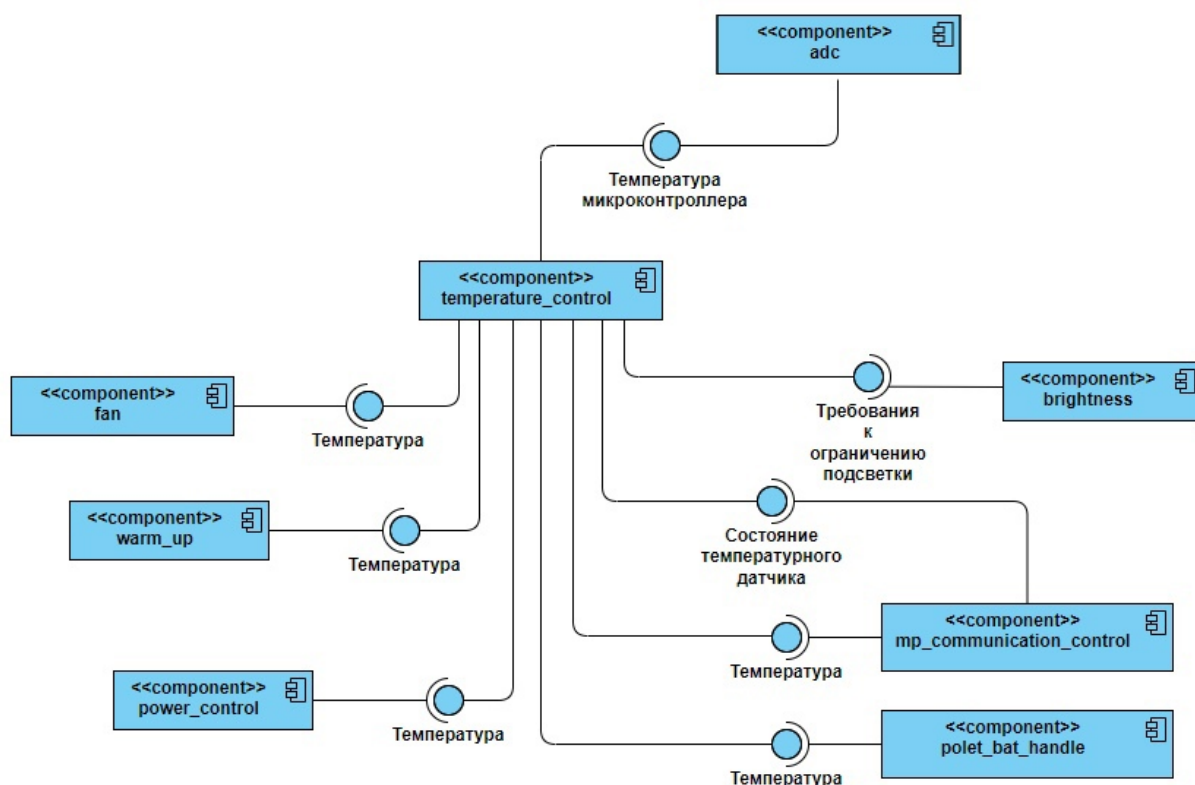


Рисунок 11 – Диаграмма компонентов: модуль «temperature_control»

3.4.9 Модуль «warm_up»

Модуль «warm_up» производит управление подогревом устройства. К функциям данного модуля относится анализ требований к включению/отключению подогрева.

Данный модуль получает от других модулей следующие данные:

- текущее значение температуры;
- требования к ограничению работы подогрева.

Данный модуль передает другим модулям текущее состояние подогрева.

Диаграмма компонентов для модуля «warm_up» приведена на рисунке 12.

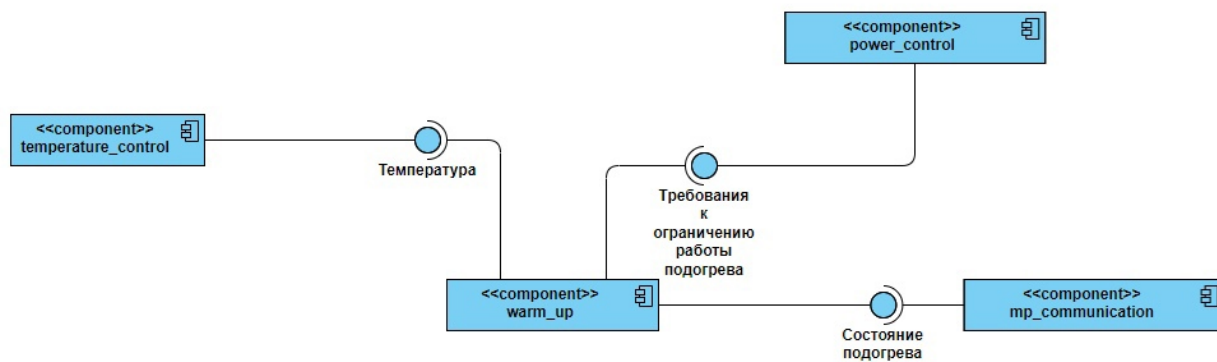


Рисунок 12 – Диаграмма компонентов: модуль «warm_up»

4 Разработка приложения

4.1 Разработка основной функции программы

В соответствии с принципом работы, описанным в подразделе 3.3 инициализация периферии и пользовательских модулей, а также вызов функций обработки происходит в функции «main».

4.1.1 Инициализация периферии

Код, производящий инициализацию периферии, представлен ниже:

```
/* MCU Configuration-----  
-----*/  
  
/* Reset of all peripherals, Initializes the Flash interface  
and the SysTick. */  
HAL_Init();  
  
/* USER CODE BEGIN Init */  
HAL_Delay(3000);  
/* USER CODE END Init */  
  
/* Configure the system clock */  
SystemClock_Config();  
  
/* USER CODE BEGIN SysInit */  
  
/* USER CODE END SysInit */  
  
/* Initialize all configured peripherals */  
MX_GPIO_Init();  
MX_DMA_Init();  
MX_ADC1_Init();  
MX_RTC_Init();  
MX_TIM1_Init();  
MX_TIM2_Init();
```

```

MX_TIM3_Init();
MX_TIM4_Init();
MX_USART3_UART_Init();
MX_IWDG_Init();
MX_USB_DEVICE_Init();
MX_SPI2_Init();
/* USER CODE BEGIN 2 */

```

В данном фрагменте кода производятся:

- инициализация библиотеки HAL функцией «HAL_Init»;
- временная задержка, направленная на завершение переходных процессов в периферии микроконтроллера;
- настройка режима работы источника синхросигнала при помощи функции «SystemClock_Config»;
- настройка портов ввода-вывода осуществляется при помощи функции «MX_GPIO_Init»;
- настройка прямого доступа к памяти осуществляется при помощи функции «MX_DMA_Init»;
- настройка АЦП осуществляется при помощи функции «MX_ADC1_Init»;
- настройка часов реального времени осуществляется при помощи функции «MX_RTC_Init»;
- настройка таймеров осуществляется при помощи функций «MX_TIM1_Init», «MX_TIM2_Init», «MX_TIM3_Init», «MX_TIM4_Init»;
- настройка интерфейса UART осуществляется при помощи функции «MX_USART3_UART_Init»;
- настройка сторожевого таймера осуществляется при помощи функции «MX_IWDG_Init»;
- настройка интерфейса USB осуществляется при помощи функции «MX_USB_DEVICE_Init»;
- настройка интерфейса SPI осуществляется при помощи функции «MX_SPI2_Init».

4.1.2 Инициализация Модулей

После инициализации периферии происходит инициализация модулей. Для каждого из модулей происходит передача необходимых ему параметров.

Код, производящий инициализацию модуля «matrix_keyboard», представлен ниже:

```
tmp_pin_array[0].pin = ROW_1_Pin;
tmp_pin_array[0].port = ROW_1_GPIO_Port;
tmp_pin_array[1].pin = ROW_2_Pin;
tmp_pin_array[1].port = ROW_2_GPIO_Port;
tmp_pin_array[2].pin = ROW_3_Pin;
tmp_pin_array[2].port = ROW_3_GPIO_Port;
tmp_pin_array[3].pin = ROW_4_Pin;
tmp_pin_array[3].port = ROW_4_GPIO_Port;
tmp_pin_array[4].pin = ROW_5_Pin;
tmp_pin_array[4].port = ROW_5_GPIO_Port;
tmp_pin_array[5].pin = COLUMN_1_Pin;
tmp_pin_array[5].port = COLUMN_1_GPIO_Port;
tmp_pin_array[6].pin = COLUMN_2_Pin;
tmp_pin_array[6].port = COLUMN_2_GPIO_Port;
tmp_pin_array[7].pin = COLUMN_3_Pin;
tmp_pin_array[7].port = COLUMN_3_GPIO_Port;
tmp_pin_array[8].pin = COLUMN_4_Pin;
tmp_pin_array[8].port = COLUMN_4_GPIO_Port;
tmp_pin_array[9].pin = COLUMN_5_Pin;
tmp_pin_array[9].port = COLUMN_5_GPIO_Port;
tmp_pin_array[10].pin = COLUMN_6_Pin;
tmp_pin_array[10].port = COLUMN_6_GPIO_Port;
Matrix_Keyboard_Init(tmp_pin_array);
```

В качестве параметров в модуль «matrix_keyboard» передается массив, содержащий информацию о контактах микроконтроллера, подключенных к клавиатуре.

Код, производящий инициализацию модуля «adc» представлен ниже:

```
ADC_Handle_Init(&hadc1, TRUE);
```

В качестве параметров функции инициализации передается дескриптор периферии АЦП и логическое значение, отвечающее за включение или отключение прерываний данного модуля. В самой функции инициализации производится настройка коэффициентов фильтрации и запуск работы АЦП в связке с прямым доступом к памяти.

Код, производящий инициализацию модулей управления температуры («temperature_control», «fan» и «warm_up»), представлен ниже:

```
__HAL_SPI_ENABLE(&hspi2);
Temperature_Control_Init(&hspi2, DOUT_SPI1_CS_TEMP_GPIO_Port,
DOUT_SPI1_CS_TEMP_Pin);
```

```
FAN_Init(&htim3, TIM_CHANNEL_1, TIM_CHANNEL_2, 0);
FAN_Off(0);
FAN_Off(1);
```

```
WARM_UP_Init(DOUT_HEATING_GPIO_Port, DOUT_HEATING_Pin);
```

```
for (uint8_t counter = 0; counter < 50; counter++)
{
    HAL_Delay(10);
    ADC_Handle_Update_Value();
}
for (uint8_t counter = 0; counter < 20; counter++)
{
```

```
    HAL_Delay(250);
```

```
    ADC_Handle_Update_Value();
```

```
    Temperature_Control_Update_MC_Data();
```

```
    Temperature_Control_Update_Sensor_Data();
```

```
    WARM_UP_Analyze_1s(Temperature_Control_Get_Temp(),
Temperature_Control_Get_Sensor_Status());
```

```
#ifndef IWDG_DISABLE
```

```

        HAL_IWDG_Refresh(&hiwdg);
    #endif
}
Temperature_Control_Set_Regular_K();

```

В качестве параметров к функции «Temperature_Control_Init» передаются дескриптор периферии SPI и контакты, используемые для работы интерфейса SPI. Далее производится инициализация модуля «fan». В качестве параметров передаются дескриптор таймера и используемые каналы таймера. После инициализации модуля производится отключение вентиляторов.

После этого в двух последовательных циклах производится обновление значений температуры, полученных от внешнего и встроенного датчиков температуры при помощи вызова функций запроса обновления данных от модулей «temperature_control» и «adc» соответственно. Вызов данных функций множество раз гарантирует, что по завершению работы циклов будут получены отфильтрованные температурные значения с минимальной погрешностью.

Особенностью фильтрации данных внешнего температурного датчика является то, что изначальная фильтрация производится с коэффициентами, позволяющим за указанное выше количество циклов получить текущие температурные данные с определенной погрешностью, а затем происходит изменение коэффициента, обеспечивающее менее быструю, но более точную фильтрацию данных.

Код, производящий инициализацию модулей управления питанием («power_control» и «temperature_control»), представлен ниже:

```

tmp_pin_array[0].pin = DIN_110V_Pin;
tmp_pin_array[0].port = DIN_110V_GPIO_Port;
tmp_pin_array[1].pin = DIN_MBOARD_IS_ON_Pin;
tmp_pin_array[1].port = DIN_MBOARD_IS_ON_GPIO_Port;
tmp_pin_array[2].pin = DOUT_MBOARD_POWER_EN_Pin;
tmp_pin_array[2].port = DOUT_MBOARD_POWER_EN_GPIO_Port;
tmp_pin_array[3].pin = DOUT_AUDIO_EN_Pin;
tmp_pin_array[3].port = DOUT_AUDIO_EN_GPIO_Port;

```

```

Power_Control_Init(Temperature_Control_Get_Temp(),
tmp_pin_array, Temperature_Control_Get_Sensor_Status());

tmp_pin_array[2].pin = DIN_BATTERY_CHARGE_STATUS_Pin;
tmp_pin_array[2].port = DIN_BATTERY_CHARGE_STATUS_GPIO_Port;
tmp_pin_array[3].pin = DOUT_BATTERY_CHARGE_EN_Pin;
tmp_pin_array[3].port = DOUT_BATTERY_CHARGE_EN_GPIO_Port;
tmp_pin_array[4].pin = DOUT_BATTERY_BOOST_EN_Pin;
tmp_pin_array[4].port = DOUT_BATTERY_BOOST_EN_GPIO_Port;
tmp_pin_array[5].pin = DIN_BATTERY_FAULT_Pin;
tmp_pin_array[5].port = DIN_BATTERY_FAULT_GPIO_Port;
tmp_pin_array[6].pin = GPIO_PIN_7;
tmp_pin_array[6].port = GPIOC;

Polet_BAT_Init(Temperature_Control_Get_Temp(),
Temperature_Control_Get_Sensor_Status(), tmp_pin_array);

```

В качестве параметров в данные функции передаются не только используемые модулями контакты, но и показания температурных датчиков, что позволяет данным модулям сразу после инициализации контролировать требования для разрешения включения различных узлов устройства.

Код, производящий инициализацию модуля управления подсветкой («brightness»), представлен ниже:

```

tmp_pin_array[0].pin = DOUT_LCD_EN_Pin;
tmp_pin_array[0].port = DOUT_LCD_EN_GPIO_Port;

Brightness_Init(tmp_pin_array[0], Polet_BAT_Get_Status(),
ADC_Handle_Get_Value(ADC_Channel_Bright), 3,
(CtrlVar_GetValue_u8(Prm_u8_ManualBright) &
CV_BRIGHT_MANUAL_EN_MASK_INV), Temperature_Control_Get_Temp());
if ((CtrlVar_GetValue_u8(Prm_u8_ManualBright) &
CV_BRIGHT_MANUAL_EN_MASK) != 0)
{
    Brightness_Set_Manual_Mode(Brightness_Switch_Mode_MP,
CtrlVar_GetValue_u8(Prm_u8_ManualBright) &
CV_BRIGHT_MANUAL_EN_MASK_INV);
}

```

```
Brightness_SaveMode();
}
```

В качестве аргументов данный модуль принимает не только используемые им контакты, но и множество флагов, настраивающих изначальный режим работы подсветки.

В последнюю очередь производится инициализация модуля «mp_communication_protocol»:

```
USB_HID_Init();
```

Порядок инициализации всех указанных выше модулей выбран не случайно, так как большинство модулей для начала своей работы требуют параметры, получаемые из других модулей.

В качестве значений, заполняемых в массив передаваемых контактов, используются определения, объявленные при помощи макроса define. При этом, модули производят перенастройку данных контактов внутри функций инициализации. Данное решение было принято для упрощения дальнейшего сопровождения: при изменении схематехнической привязки контактов к модулям используется переименование объявлений, а не перенастройка режима работы портов ввода-вывода при помощи среды STM32CubeIDE.

При возникновении необходимости создания другого исполнения устройства создаются новые определения для указанных выше объявлений. Созданные определения включаются в файл «main.h» внутрь условных директив #if, #elif, #endif. Пример использования данных директив представлен ниже:

```
#if (EXECUTION == 0)
//В данный блок вставляются define'ы для EXECUTION = 0
#define (ROW_1_Pin) (GPIO_Pin_1)
#define (ROW_1_GPIO_Port) (GPIOA)
#elif (EXECUTION == 1)
//В данный блок вставляются define'ы для EXECUTION = 1
#define (ROW_1_Pin) (GPIO_Pin_2)
#define (ROW_1_GPIO_Port) (GPIOB)
#endif
```

4.1.3 Цикл обработки данных и принятия решений

После инициализации модулей программа переходит в бесконечный цикл, в котором происходит обработка данных и принятие решений. В самом начале данного цикла происходит проверка состояния флага «one_ms_is_elapsed_flag». Данный флаг выставляется каждую миллисекунду работы программы при помощи обработчика событий. Основным назначением данного флага является формирование временных интервалов для отслеживания времени между вызовами функций из цикла обработки данных и принятия решений. Функции, которые необходимо вызывать через одинаковые промежутки времени, группируются между проверками значений переменных «timer_1ms», «timer_10ms», «timer_100ms». Данные переменные инкрементируются каждые 1, 10 и 100 миллисекунд соответственно и используются для формирования интервалов в 1, 10, 100 и 500 миллисекунд.

Код, выполняющийся каждую миллисекунду работы устройства, представлен ниже:

```
Polet_BAT_Analyze_STAT();  
Power_Control_Perform_Transition(Matrix_Keyboard_Get_FuncBTN_State(M  
AXRIX_KEYBOARD_FUNC_POWER_SWITCH), Temperature_Control_Get_Temp(),  
Polet_BAT_Get_Status(), Temperature_Control_Get_Sensor_Status());
```

Данные функции выполняют анализ состояния ИАП и входного питания. Выполнение их каждую секунду гарантирует своевременную реакцию на изменение в режиме питания.

Код, выполняющийся каждые 10 миллисекунд работы устройства, представлен ниже:

```
Buzzer_Timer_10ms();  
  
Matrix_Keyboard_Check_All_Buttons();  
  
ADC_Handle_Update_Value();
```

```

Brightness_Update_Data(Matrix_Keyboard_Get_FuncBTN_State(MAXRIX_KEYB
OARD_FUNC_BRIGHTNESS_INC),
Matrix_Keyboard_Get_FuncBTN_State(MAXRIX_KEYBOARD_FUNC_BRIGHTNESS_DE
C), ADC_Handle_Get_Value(ADC_Channel_Bright),
Polet_BAT_Get_Status(),
Temperature_Control_Get_Temp());

CtrlVar_Upd_Next_Val();

USB_HID_10ms();

```

Данные функции, в основном, выполняют анализ параметров устройства, которые возможно замерить за короткий промежуток времени.

Код, выполняющийся каждые 100 миллисекунд работы устройства, представлен ниже:

```

if (led_state == LED_NO_BAT)
{
    HAL_GPIO_TogglePin(DOUT_LED_TEST_GPIO_Port, DOUT_LED_TEST_Pin);
}

```

Данный код используется для отладки устройства — он заставляет светодиод мигать при отсутствии или некорректной работе ИАП.

Код, выполняющийся каждые 500 миллисекунд работы устройства, представлен ниже:

```

if (++timer_100ms >= 5)
{
    timer_100ms = 0;

    if (Polet_BAT_Is_Exist() == FALSE)
    {
        led_state = LED_NO_BAT;
    }
    else
    {
        if (USB_HID_CommunicationIsOK() == TRUE)

```

```

        {
            led_state = LED_OK;
        }
        else
        {
            led_state = LED_NO_COM;
        }
    }

    if (led_state == LED_OK)
    {
        HAL_GPIO_TogglePin(DOUT_LED_TEST_GPIO_Port,
DOUT_LED_TEST_Pin);
    }
    else
    {
        if (led_state == LED_NO_COM)
        {
            HAL_GPIO_WritePin(DOUT_LED_TEST_GPIO_Port,
DOUT_LED_TEST_Pin, GPIO_PIN_SET);
        }
    }

    USB_HID_Push_Errors();

    Brightness_Update_500ms();

```

В начале данного сегмента кода производится проверка работы ИАП и коммуникации с материнской платы. Данная проверка служит для отладки программы — исходя из результатов проверки производится настройка режима работы светодиода. После этого происходит вызов функции «USB_HID_Push_Errors». Данная функция производит опрос всех узлов устройства и записывает в стек сообщений информацию о всех ошибках в устройстве. При отправке сообщения производится передача кодовых сигналов из

данного стека в материнскую плату для поиска неисправностей в устройстве. Далее происходит обновление яркости подсветки экрана и клавиатуры.

4.2 Описание основных функций модулей

4.2.1 Модуль «adc»

Для инициализации модуля используется функция «ADC_Handle_Init». Данный модуль производит включение и настройку АЦП микроконтроллера, а также настройку прямого доступа к памяти для более быстрой передачи результатов опроса АЦП. Заголовок данной функции представлен ниже:

```
/*
 * @brief      Initialize module
 * @param      Pointer to used ADC_HandleTypeDef
 * @param      Interrupt disable flag value
 *
 *              TRUE          - DMA interrupts will be disabled
 *              other values  - DMA interrupts will not be
changed
 * @retval     HAL_StatusTypeDef value - result of init
 */
HAL_StatusTypeDef ADC_Handle_Init(ADC_HandleTypeDef *hadc, uint8_t
interrupt_disable_flag)
```

Для фильтрации данных, полученных от АЦП, используется функция «ADC_Handle_Update_Value». Фильтрация производится для всех включенных каналов АЦП. Заголовок данной функции представлен ниже:

```
/*
 * @brief      Update filtered values
 * @retval     None
 */
void ADC_Handle_Update_Value()
```

Для запроса фильтрованных данных от АЦП используется функция «ADC_Handle_Get_Value». Данная функция определяет запрашиваемый канал. Если запрашиваемый канал существует, то функция возвращает фильтрованное значение. В ином случае функция возвращает 0. Для канала, опрашивающего

датчик освещения, производится проверка фильтрованных данных. В случае выхода датчика освещения и получения данных, выходящих за пределы минимального и максимального значения, производится возврат минимального или максимального значения соответственно. Код данной функции представлен ниже:

```
uint16_t ADC_Handle_Get_Value(ADC_Channels channel_index)
{
    if (channel_index == ADC_Channel_Bright)
    {
        if (adc_handle.Channel[ADC_Channel_Bright].valueCodeFiltr <
min)
        {
            min =
adc_handle.Channel[ADC_Channel_Bright].valueCodeFiltr;
        }
        else
        {
            if
(adc_handle.Channel[ADC_Channel_Bright].valueCodeFiltr > max)
            {
                max =
adc_handle.Channel[ADC_Channel_Bright].valueCodeFiltr;
            }
        }
    }

    if (channel_index < ADC_Channel_last)
    {
        return adc_handle.Channel[channel_index].valueCodeFiltr;
    }

    return 0;
}
```

4.2.2 Модуль «brightness»

Для инициализации модуля используется функция «Brightness_Init». Данный модуль производит начальную настройку режима подсветки при помощи: текущих режимов температуры и питания, битов настройки стандартного режима освещения. При выполнении данной функции подсветка будет переведена в режим подсветки с фиксированной или автоматической яркостью. Более того, будут учтены требования к ограничению подсветки, полученные от температурного модуля и модуля питания. Заголовок данной функции представлен ниже:

```
/*
 * @brief      Initialize module
 * @param      Display enable pin data
 * @param      Battery status
 * @param      ADC value (12 bit value)
 * @param      Auto mode settings
 * @param      Manual mode settings
 * @param      Temperature value
 * @retval     None
 */
void Brightness_Init(Pin_Struct_Type en_pin, Polet_BAT_Status
bat_status, uint16_t adc_arg, uint8_t auto_arg, uint8_t manual_arg,
float temp_arg)
```

Для обновления значений, используемых при расчете требуемой яркости, используется функция «Brightness_Update_Data». Заголовок данной функции представлен ниже:

```
/*
 * @brief      Update required brightness
 * @param      "Brightness increment and backlight enable" button
state
 * @param      "Brightness decrement and backlight disable" button
state
 * @param      ADC value (12 bit value)
 * @param      Battery status
```

```

*   @param    Temperature value
*   @retval   None
*/
void Brightness_Update_Data(Button_State btn_bright_inc_flag,
Button_State btn_bright_dec_flag, uint16_t adc_value,
Polet_BAT_Status bat_status, float temperature)

```

Для получения текущих значений яркости подсветки экрана и клавиатуры используются функции «Brightness_Get_Display_Percent» и «Brightness_Get_Keyboard_Value» соответственно. Заголовки данных функций представлены ниже:

```

/*
*   @brief    Returns current display brightness
*   @retval   Display brightness value (in percent * 100)
*/
float Brightness_Get_Display_Percent()
/*
*   @brief    Returns current keyboard brightness
*   @retval   Keyboard brightness value (in percent * 100)
*/
float Brightness_Get_Keyboard_Value()

```

4.2.3 Модуль «fan»

Для инициализации модуля используется функция «FAN_Init». Данная функция производит начальную настройку режима работы таймеров и контактов, управляющих вентиляторами. Заголовок данной функции представлен ниже:

```

/*
*   @brief    Initialize module
*   @param    Timer handle
*   @param    FAN1 TIM channel value. This parameter can be one of
the following values:
*
*               @arg TIM_CHANNEL_1: TIM Channel 1 selected
*               @arg TIM_CHANNEL_2: TIM Channel 2 selected
*               @arg TIM_CHANNEL_3: TIM Channel 3 selected
*               @arg TIM_CHANNEL_4: TIM Channel 4 selected

```

```

* @param FAN2 TIM channel value. (See Fan1 channel param)
* @param Current temperature
* @retval None
*/
void FAN_Init(TIM_HandleTypeDef *htim, uint32_t fan1_channel,
uint32_t fan2_channel, float temperature)

```

Для управления режимом работы вентиляторов используется функция «FAN_Analyze_1s». Данная функция определяет требования к ограничению работы вентиляторов и необходимость их включения. Заголовок данной функции представлен ниже:

```

/*
* @brief Analyze current power and control fan. Should be
executed every 1 second
* @param Device power state
* @param Temperature value
* @param PowerControl disabled_by_user flag value
* @retval None
*/
void FAN_Analyze_1s(Power_Control_State state, float
temperature_arg, uint8_t disabledByUser)

```

4.2.4 Модуль «matrix_keyboard»

Для инициализации модуля используется функция «Matrix_Keyboard_Init». Данный модуль производит начальную настройку контактов, использующихся в клавиатуре. Заголовок данной функции представлен ниже:

```

/*
* @brief Initialize module
* @param Pins data array
* @note Pins data array should include:
MAXRIX_KEYBOARD_ROWS_AMOUNT rows pins and
MAXRIX_KEYBOARD_COLLIS_AMOUNT colls pins
* @retval None
*/
void Matrix_Keyboard_Init(Pin_Struct_Type *pins)

```

Для проверки состояния клавиш клавиатуры используется функция «Matrix_Keyboard_Check_All_Buttons». Данная функция производит перенастройку контакта в режим вывода сигнала и отправку сигнала при помощи функций «Matrix_Keyboard_Send_Signal_On_Row» и «Matrix_Keyboard_Send_Signal_On_Coll». Опрос строк и столбцов осуществляется при помощи функций «Matrix_Keyboard_Poll_Row» и «Matrix_Keyboard_Poll_Coll». Для перенастройки контакта обратно в режим ввода используются функции «Matrix_Keyboard_Clear_Row» и «Matrix_Keyboard_Clear_Coll». Заголовки данных функций представлены ниже:

```
/*
 * @brief      Update all button status and search pressed key
 * @retval     None
 */
void Matrix_Keyboard_Check_All_Buttons()
/*
 * @brief      Set row in output mode and switch it in reset state
 * @param      Selected row index
 * @retval     None
 */
void Matrix_Keyboard_Send_Signal_On_Row(uint8_t row_index)
/*
 * @brief      Set row in input mode
 * @param      Selected row index
 * @retval     None
 */
void Matrix_Keyboard_Clear_Row(uint8_t row_index)
/*
 * @brief      Set coll in output mode and switch it in reset state
 * @param      Selected coll index
 * @retval     None
 */
void Matrix_Keyboard_Send_Signal_On_Coll(uint8_t coll_index)
/*
```

```

*   @brief      Set coll in input mode
*   @param      Selected coll index
*   @retval     None
*/
void Matrix_Keyboard_Clear_Coll(uint8_t coll_index)
/*
*   @brief      Poll button on selected row
*   @param      Selected row index
*   @retval     None
*/
void Matrix_Keyboard_Poll_Coll(uint8_t coll_index)

*   @note       Pins data array should include:
MAXRIX_KEYBOARD_ROWS_AMOUNT rows pins and
MAXRIX_KEYBOARD_COLLS_AMOUNT colls pins
*   @retval     None
*/
void Matrix_Keyboard_Init(Pin_Struct_Type *pins)
/*
*   @brief      Poll button on selected row
*   @param      Selected row index
*   @retval     None
*/
void Matrix_Keyboard_Poll_Row(uint8_t row_index)

```

Для опроса всех строк и столбцов клавиатуры необходимо провести поиск нажатых клавиш. Для этого используется функция «Matrix_Keyboard_Search_Pressed_Key». При обнаружении более двух нажатых клавиш данная функция сообщает об ошибке, так как матричная клавиатура может выдавать ложные данные при зажатии более 2 клавиш одновременно. Заголовок данной функции представлен ниже:

```

/*
*   @brief      Update pressed key indexes
*   @retval     None
*/

```

```
void Matrix_Keyboard_Search_Pressed_Key()
```

4.2.5 Модуль «mp_communication_protocol»

Для контроля временных интервалов между отправками сообщений материнской плате используется функция «USB_HID_10ms». Данная функция ведет подсчет времени, прошедшего от последней отправки сообщения. Если данное время превышает максимально допустимое — производится отправка сообщения. Также, сообщение отправляется незамедлительно при наличии таких условий, как получение от материнской платы запроса или обновление состояния клавиатуры.

Для разбора полученного сообщения используется функция «MP_Communication_Protocol_Analyze_Rx_Message». Данная функция вызывается внутри обработчика событий получения сообщения.

Функция «USB_HID_Push_Errors» применяется для анализа состояния устройства и добавления в стек кодов ошибок информации о неисправностях. Коды ошибок, добавляемые данной функцией, забираются из стека при формировании сообщения.

4.2.6 Модуль «polet_bat_handle»

Для инициализации модуля используется функция «Polet_BAT_Init». Данная функция нужна для определения контактов, использующихся при взаимодействии с ИАП. Заголовок данной функции представлен ниже:

```
/*
 * @brief      Initialize Polet_BAT_Handle
 * @param      Current temperature
 * @param      Temperature sensor status
 * @param      Pins data array
 * @note       Pins data array should include: Power pin; MainBoard
Maintenance pin; STAT pin; CHR pin; EN pin; FAULT pin
 * @retval     None
 */
void Polet_BAT_Init(float temperature_arg, uint8_t sensor_status,
Pin_Struct_Type *pins)
```

Для анализа состояния ИАП используется функция «Polet_BAT_Perform_Transition». Данная функция проверяет условия изменения текущего состояния ИАП и, при необходимости, меняет его. ИАП может находиться в следующих режимах:

- пассивный режим;
- активный режим;
- режим зарядки ИАП;
- режим ошибки.

Заголовок данной функции представлен ниже:

```
/*
 * @brief    Perform transition between states
 * @retval   None
 */
void Polet_BAT_Perform_Transition()
```

4.2.7 Модуль «power_control»

Для инициализации модуля используется функция «Power_Control_Init». Данная функция нужна для определения контактов для управления питанием. Заголовок данной функции представлен ниже:

```
/*
 * @brief    Initialize Power_Control_Handle
 * @param    Current temperature value
 * @param    Pins data array
 * @note     Pins data array should include: Power pin; MainBoard
Maintenance pin; Display ~EN pin; MainBoard Power pin; Audio enable
pin
 * @param    Temperature sensor status
 * @retval   None
 */
void Power_Control_Init(float temperature_arg, Pin_Struct_Type*
pins, uint8_t temp_status)
```

Для анализа текущего режима питания используется функция «Power_Control_Perform_Transition». Заголовок данной функции представлен ниже:

```
/*
 * @brief      Check power status change conditions
 * @param      Power button state
 * @param      Current temperature value
 * @param      Battery state
 * @brief      Perform transition between states
 * @param      Temperature sensor status
 * @retval     None
 */
void Power_Control_Perform_Transition(Button_State btn_down_flag,
float temperature_arg, Polet_BAT_Status bat_status, uint8_t
temp_status)
```

4.2.8 Модуль «temperature_control»

Для инициализации модуля используется функция «Temperature_Control_Init». Данная функция нужна для определения контактов, использующихся при взаимодействии с внешним температурным датчиком и настройки допустимых значений температуры. Код данной функции представлен ниже:

```
/*
 * @brief      Initialize Temperature_Control_Handle
 * @param      Pointer to used SPI_HandleTypeDef
 * @param      Temperature sensor CS signal port
 * @param      Temperature sensor CS signal port
 * @retval     None
 */
HAL_StatusTypeDef Temperature_Control_Init(SPI_HandleTypeDef
*spi_handle, GPIO_TypeDef *port, uint16_t pin)
{
    temp_handle.cs.pin = pin;
    temp_handle.cs.port = port;
```

```

temp_handle.spi_handle = spi_handle;
temp_handle.spi_last_rx_data = 0;
temp_handle.spi_error_amount = SPI_ERROR_AMOUNT_MAXIMUM;

temp_handle.tempVal_status_too_high = TEMP_STATUS_TOO_HIGH_NOK;
temp_handle.tempVal_status_high = TEMP_STATUS_HIGH_NOK;
temp_handle.tempVal_status_low = TEMP_STATUS_LOW_NOK;
temp_handle.tempVal_status_too_low = TEMP_STATUS_TOO_LOW_NOK;

temp_handle.filtered_sensor_temp =
temp_handle.tempVal_status_too_low;
temp_handle.filtered_mc_temp =
temp_handle.tempVal_status_too_low;

Temperature_Control_Set_Init_K();

Temperature_Control_Configure_SPI();
uint16_t tmp_u16 = 0;
return Read_Sensor_Data(&tmp_u16);
}

```

Для запроса и фильтрации значений температуры от внутреннего датчика температуры используется функция «Temperature_Control_Update_MC_Data». Для запроса и фильтрации значений температуры от внутреннего датчика температуры используется функция «Temperature_Control_Update_Sensor_Data».

Для получения фильтрованного значения температуры используется функция «Temperature_Control_Get_Temp». Приоритетными считаются показания внешнего температурного датчика. При поломки внешнего температурного датчика возвращается значение, полученное от внутреннего температурного датчика.

Для получения текущего температурного режима используется функция «Temperature_Control_Get_Sensor_Status».

4.2.9 Модуль «warm_up»

Для инициализации модуля используется функция «WARM_UP_Init». Данная функция нужна для определения контакта, использующегося при взаимодействии с обогревом. Заголовок данной функции представлен ниже:

```
/*
 * @brief      Initialize module
 * @param      WARM UP port
 * @param      WARM UP pin
 * @retval     None
 */
void WARM_UP_Init(GPIO_TypeDef* port, uint16_t pin)
```

Для обновления текущего статуса обогрева используется функция «WARM_UP_Analyze_1s». Данная функция включает и отключает обогрев, основываясь на текущей температуре. Заголовок данной функции представлен ниже:

```
/*
 * @brief      Analyze current temperature and control warm up.
                Should be executed every 1 second
 * @param      Current temperature sensor data
 * @param      Temperature sensor status
 * @retval     None
 */
void WARM_UP_Analyze_1s(float temperature_arg, uint8_t temp_status)
```

5 Технико-экономическое обоснование

Технико-экономическое обоснование — это результат расчетов экономической целесообразности разработки проекта, включающий в себя учет все затраты, которые необходимо произвести для выполнения проекта. ТЭО включает в себя следующие пункты:

- расчет единовременных затрат на оборудование;
- расчет затрат на заработную плату;
- расчет стоимости устройства.

5.1 Расчет единовременных затрат на оборудование

Для разработки ПО необходимо приобрести вычислительную технику и лицензионное ПО. Перечень затрат на вычислительную технику и ПО приведен в таблице 11.

Таблица 11 – Единовременные затраты на оборудование

Наименование	Цена	Источник	Примечание
ПК MSI PRO DP180 13-222XRU [9S6-B0A721-291]	48999	DNS	IntelCore i3-13100 — 4 ядра, 3.4 ГГц; 8 Гб оперативной памяти; Встроенная видеокарта Intel UHD Graphics 730; 256 Гб SSD
Монитор MSI PRO MP243X	10999	DNS	
Клавиатура проводная Logitech K120 [920-002506/22]	1399	DNS	
Мышь проводная Logitech M100 [910-005006]	799	DNS	
Microsoft Windows 11 Pro	17699	DNS	

На основании таблицы 11 сумма единовременных затрат на оборудование составит 79895 рублей.

5.2 Расчет затрат на заработную плату

Затраты на разработку проекта включают в себя:

- сумму затрат на основную заработную плату сотрудников;
- сумму затрат на дополнительную заработную плату сотрудников, составляющую 15% от затрат на основную заработную плату. Перечень затрат на разработку программы приведен в таблице 12.

Таблица 12 – Затраты на разработку программы

Специалист	Количество, человек	Основная заработная плата, рублей/человек	Дополнительная заработная плата, рублей/человек	Всего, рублей
Руководитель проекта	1	80000	12000	92000
Прикладной программист	1	50000	7500	57500
Тестировщик	1	45000	6750	51750
Инженер-схемотехник	1	50000	7500	57500
Итого	4	225000	33750	258750

На основании таблицы 12 сумма затрат на разработку проекта составит 258750 рублей в месяц. Ориентировочное время разработки ПО составляет 6 месяцев

5.3 Общий расчет затрат

Итоговые затраты на разработку ПО равны сумме затрат на закупку необходимого оборудования (см. таблица 11) и затрат на выплату заработной платы сотрудникам (см. таблица 12). Итоговая сумма затрат составит 1632395 рублей.

5.4 Расчет стоимости устройства

Расчет стоимости одного устройства приведен в таблице 13.

Таблица 13 – Расчет стоимости устройства

Статья затрат	Сумма, рублей для серии	Примечание
Материал	4741,96	
ПКИ	178279,00	
Работы и услуги сторонних организаций производственного характера	3882,84	
Затраты на оплату труда основных производственных рабочих — всего, включая НДФЛ	15297,79	
Заработная плата основных производственных рабочих сборочное производство	4313,92	
Заработная плата основных производственных рабочих в механическом цехе	8988,51	При расчете применен коэффициент многостаночности
Дополнительная заработная плата	1995,36	15% от основной заработной платы
Социальные отчисления и налоги	3334,92	21,8% от затрат на оплату труда
Общепроизводственные расходы	26072,75	196% от основной заработной платы
Общехозяйственные расходы	19421,54	146% от основной заработной платы
Себестоимость	251030,81	
Прибыль	75309,24	30% от себестоимости
Цена	326340,05	

На основании таблицы 13 себестоимость устройства составляет 251030,81 рублей, а цена устройства составляет 326340,05 рублей. Получившаяся цена является конкурентоспособной, так как цена аналогов выше.

6 Безопасность жизнедеятельности

6.1 Классификация вредных факторов

Травмы и заболевания, связанные с процессом труда, возникают в результате воздействия на организм человека факторов производственной среды и трудового процесса. Для предотвращения травм и заболеваний введены нормативные акты, регламентирующие степень тех или иных факторов.

В соответствии с ГОСТ 12.0.003-2015 «Опасные и вредные производственные факторы» опасные производственные факторы разделяют на:

- факторы, обладающие свойствами физического воздействия на организм человека;
- факторы, обладающие свойствами химического воздействия на организм человека;
- факторы, обладающие свойствами биологического воздействия на организм человека;
- факторы, обладающие свойствами психофизиологического воздействия на организм человека.

Из вышеперечисленного на работоспособность и здоровье программиста оказывают влияние следующие факторы:

- факторы физического воздействия рабочего места: температура и относительная влажность воздуха, скоростью движения (подвижностью) воздуха относительно тела работающего;
- факторы физического воздействия, связанные с электрическим током и электромагнитными полями
- факторы биологического воздействия, связанные с физическими перегрузками организма, характеризующиеся рабочей позой;
- факторы психофизиологического воздействия, вызывающие нервно-психические перегрузки (информационная нагрузка и монотонность труда).

6.2 Требования к температуре

В соответствии с СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» температура должна варьироваться в пределах от 21°C до 23°C в холодное время года и от 22°C до 24°C в теплое время года. При этом допустимым диапазоном температур считается температура от 19°C до 24°C в холодное время года и температура от 20°C до 28°C в теплое время года.

6.3 Требования к влажности воздуха

В соответствии с СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» влажность воздуха должна варьироваться в пределах от 15% до 75% как в холодное, так и в теплое время года.

6.4 Требования к подвижности воздуха

В соответствии с СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» подвижность воздуха должна быть не более 0,1 м/с.

6.5 Требования к электробезопасности

Поражение электрическим током человека может привести к ожогам, разрывам мышц, остановки дыхания и сердцебиения. Поэтому, при взаимодействии с электрическим оборудованием необходимо соблюдать технику безопасности, включающую в себя следующие правила:

- необходимо поддерживать электрическое оборудование в пригодном состоянии;
- перед использованием электрического оборудования необходимо производить проверку целостности устройства;

- при наличии внешних повреждений не допускать взаимодействие с электрическим оборудованием;
- не взаимодействовать с электрическим оборудованием при попадании на него жидкостей или в условиях повышенной влажности воздуха;
- не допускать скопления большого количества пыли на оборудовании.

6.6 Требования к оборудованию рабочего места

В соответствии с СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» пространство для ног под столом должно быть не менее 600 мм в высоту, 500 мм в ширину и 450 мм в глубину. Высота рабочей поверхности стула должна регулироваться в пределах от 680 мм до 800 мм. Клавиатура должна располагаться на расстоянии от 100 до 300 мм от края стола

					ПГУ 1.090501.03.001 ПЗ	Лист
						59
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В процессе выполнения выпускной квалификационной работы была изучена предметная область, сформулировано техническое задание на разработку программы для микроконтроллера в составе дисплейного модуля системы управления тепловоза.

Был произведен анализ и подбор инструментов разработки для реализации программы. Выполнено проектирование, в результате которого были разработаны UML диаграммы разрабатываемого программного обеспечения.

Разработаны функции для сбора и фильтрации данных АЦП и датчиков; анализа состояния электропитания; управления состоянием вентиляторов, источника автономного питания и обогрева; управления яркостью подсветки экрана и клавиатуры; опроса состояния клавиш клавиатуры; передачи данных между микроконтроллером и материнской платой.

В разделе безопасности жизнедеятельности были описаны требования к рабочему месту для минимизации воздействия вредных факторов на организм человека.

В экономическом разделе был произведен расчет затрат на разработку программного обеспечения.

Таким образом, требования технического задания были выполнены в полном объеме и в установленные сроки.

					ПГУ 1.090501.03.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

Список используемых источников

1. Грузооборот транспорта в РФ в 2023 году – Росстат [Электронный ресурс]: URL: <https://company.rzd.ru/ru/9401/page/78314?id=214370> (дата обращения 19.04.2024).

2. TECHNISCHE DATEN BC2050 / 7" MIT TOUCHSCREEN MULTIFUNKTIONS-DISPLAY [Электронный ресурс]: URL: https://www.gersys.de/fileadmin/user_upload/gersys.de/03_Downloads/Datenblaetter_D_E/Gersys_Datenblatt_BC2050_2021-12_DE.pdf (дата обращения 19.04.2024).

3. TECHNISCHE DATEN BC2000 / BC2900 – 8,4" [Электронный ресурс]: URL: https://www.gersys.de/fileadmin/user_upload/gersys.de/03_Downloads/Datenblaetter_D_E/Gersys_Datenblatt_BC2000_BC2900_2021-12_DE.pdf (дата обращения 19.04.2024).

4. TECHNISCHE DATEN BC5-SERIE / 12,1" [Электронный ресурс]: URL: https://www.gersys.de/fileadmin/user_upload/gersys.de/03_Downloads/Datenblaetter_D_E/Gersys_Datenblatt_BC5_Serie_2021-12_DE.pdf (дата обращения 19.04.2024).

5. TECHNISCHE DATEN BC6-SERIE / 15" [Электронный ресурс]: URL: https://www.gersys.de/fileadmin/user_upload/gersys.de/03_Downloads/Datenblaetter_D_E/Gersys_Datenblatt_BC6_Serie_2021-12_DE.pdf (дата обращения 19.04.2024).

6. TECHNISCHE DATEN BC7-SERIE / 10,4" [Электронный ресурс]: URL: https://www.gersys.de/fileadmin/user_upload/gersys.de/03_Downloads/Datenblaetter_D_E/Gersys_Datenblatt_BC7_Serie_2021-12_DE.pdf (дата обращения 19.04.2024).

7. Выбор среды разработки для stm32: какую выбрать? [Электронный ресурс]: URL: <https://uchet-jkh.ru/i/vybor-sredy-razrabotki-dlya-stm32-kakuyu-vybrat/> (дата обращения 19.04.2024).

8. В чем разница между C и C++? / habr [Электронный ресурс]: URL: <https://habr.com/ru/sandbox/149138/> (дата обращения 19.04.2024).

Приложение А

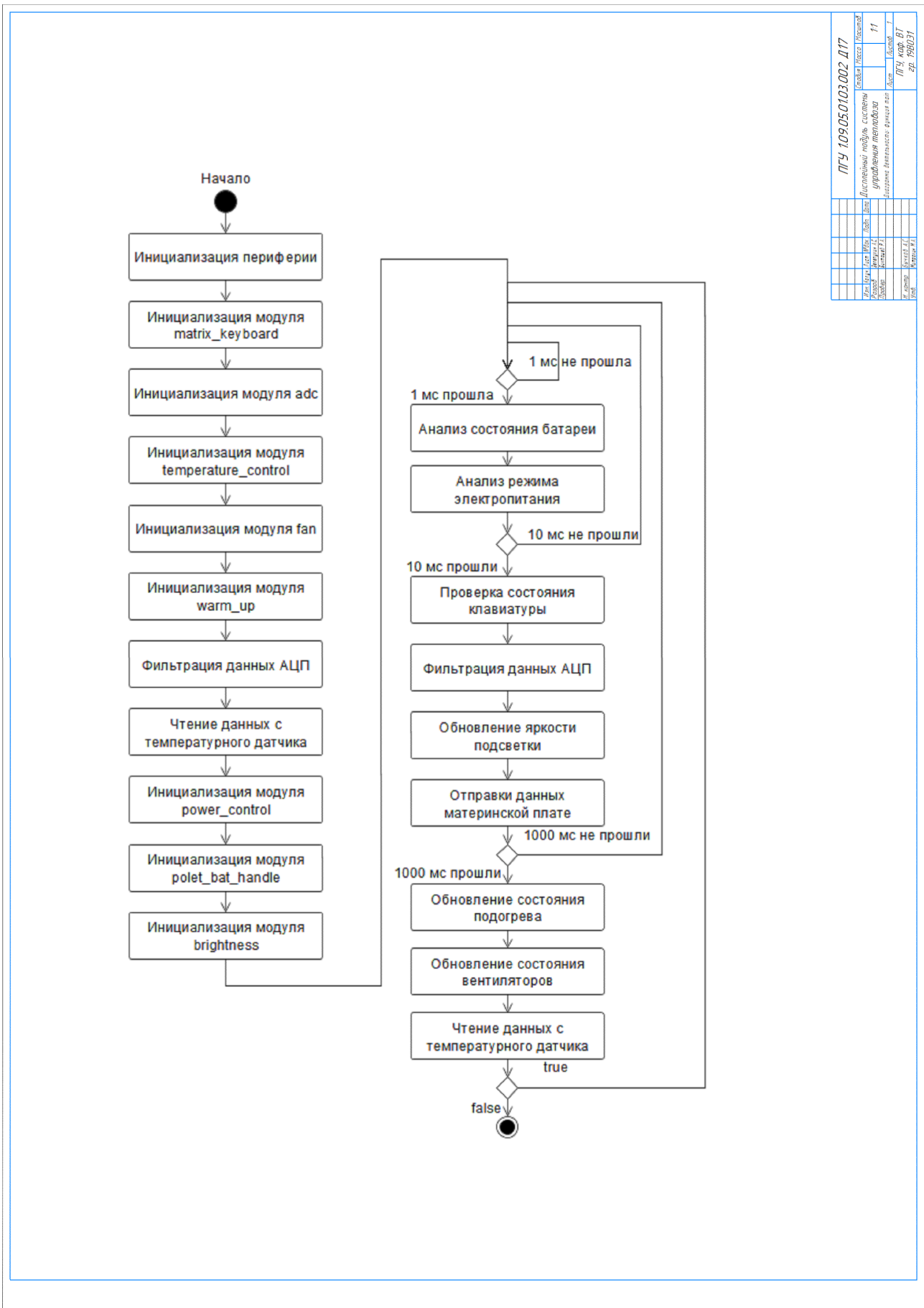
(обязательное)

Дисплейный модуль системы управления тепловоза

UML-диаграммы

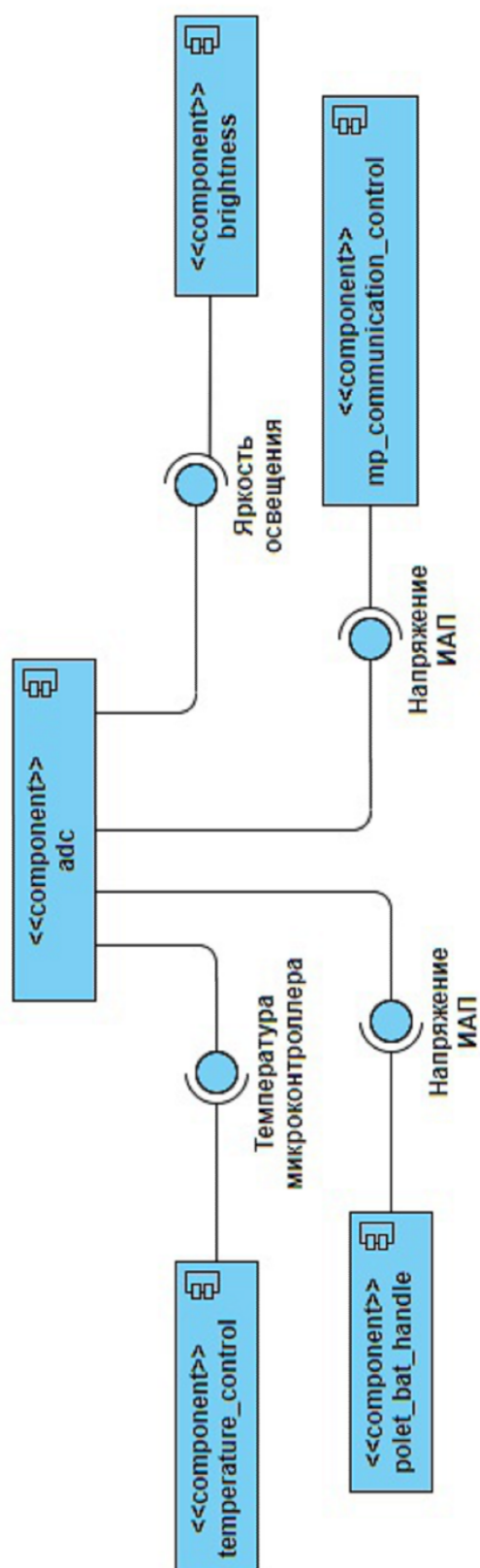
					ПГУ 1.090501.03.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – Диаграмма деятельности: функция «main»

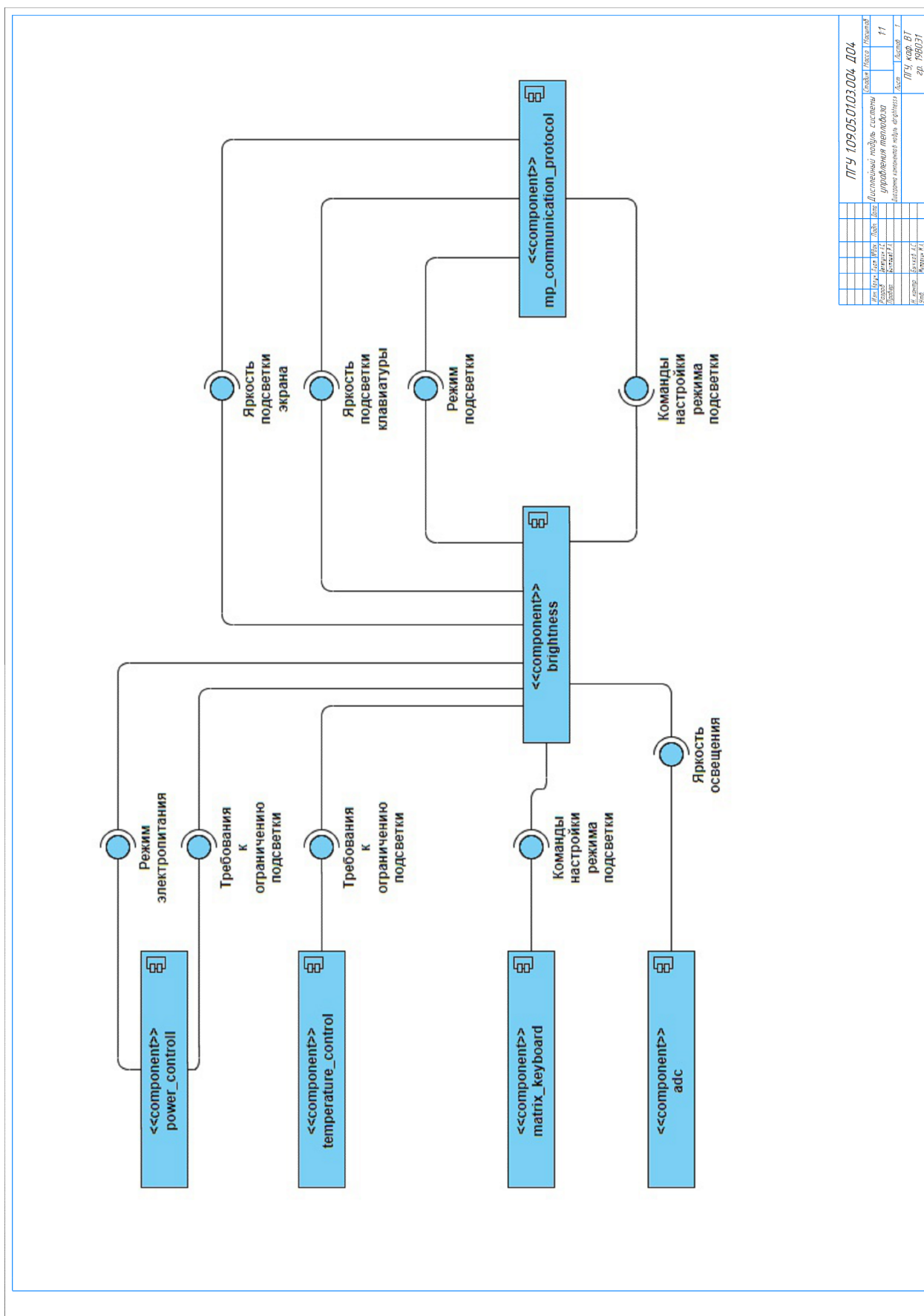


ПГУ 1.090501.03.002 Д17				Система: ПГУ 1.090501.03.002 Д17			
Диспетчерский модуль, система управления тепловым оборудованием				Версия: 1.1			
Диаграмма деятельности: функция main				ПГУ 1.090501.03.002 Д17			
				ар. 19.02.21			

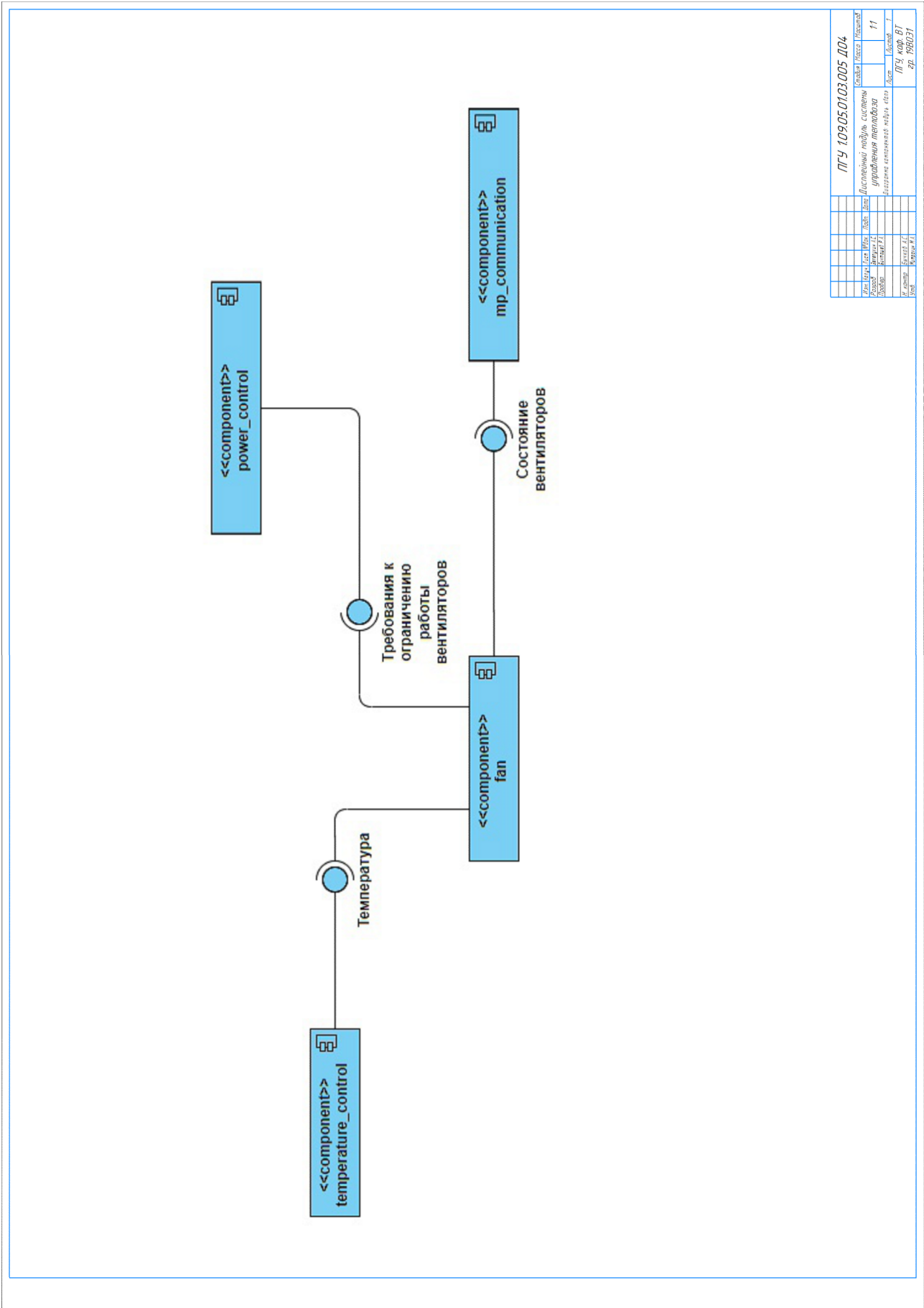
Приложение А.2 – Диаграмма компонентов: модуль «adc»

[illegible]

Приложение А.3 – Диаграмма компонентов: модуль «brightness»



Приложение А.4 – Диаграмма компонентов: модуль «fan»

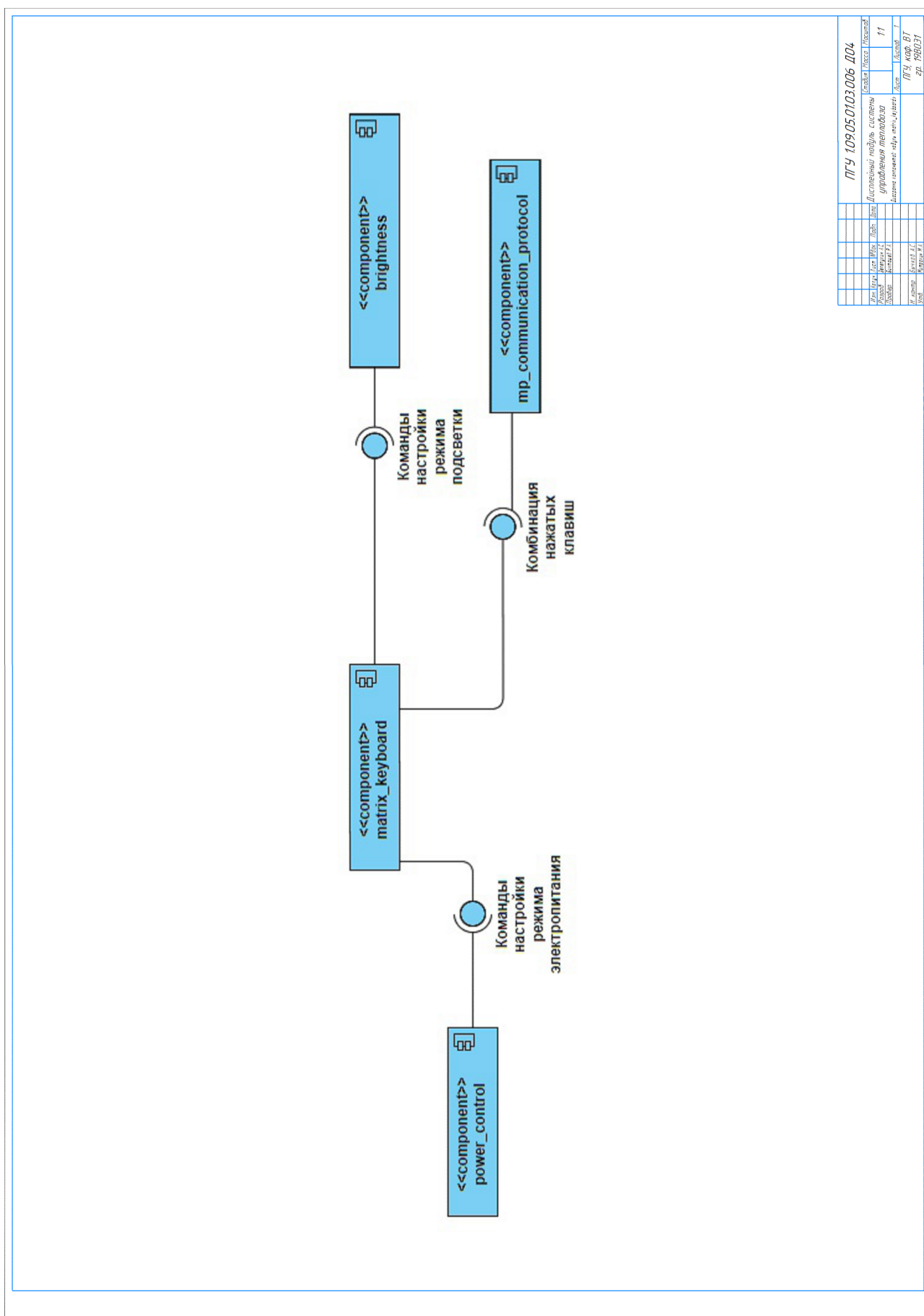


ПГУ 1.090501.03.005 Л04									
Диспетчерский модуль системы управления тепловыми узлами									
Диаграмма компонентов модуля	Лист	11	ПГУ мод. ВТ						
Дат. изм.	Лист	1	ар. 190021						
Дат. изм.	Лист	1							
Дат. изм.	Лист	1							
Дат. изм.	Лист	1							
Дат. изм.	Лист	1							
Дат. изм.	Лист	1							
Дат. изм.	Лист	1							

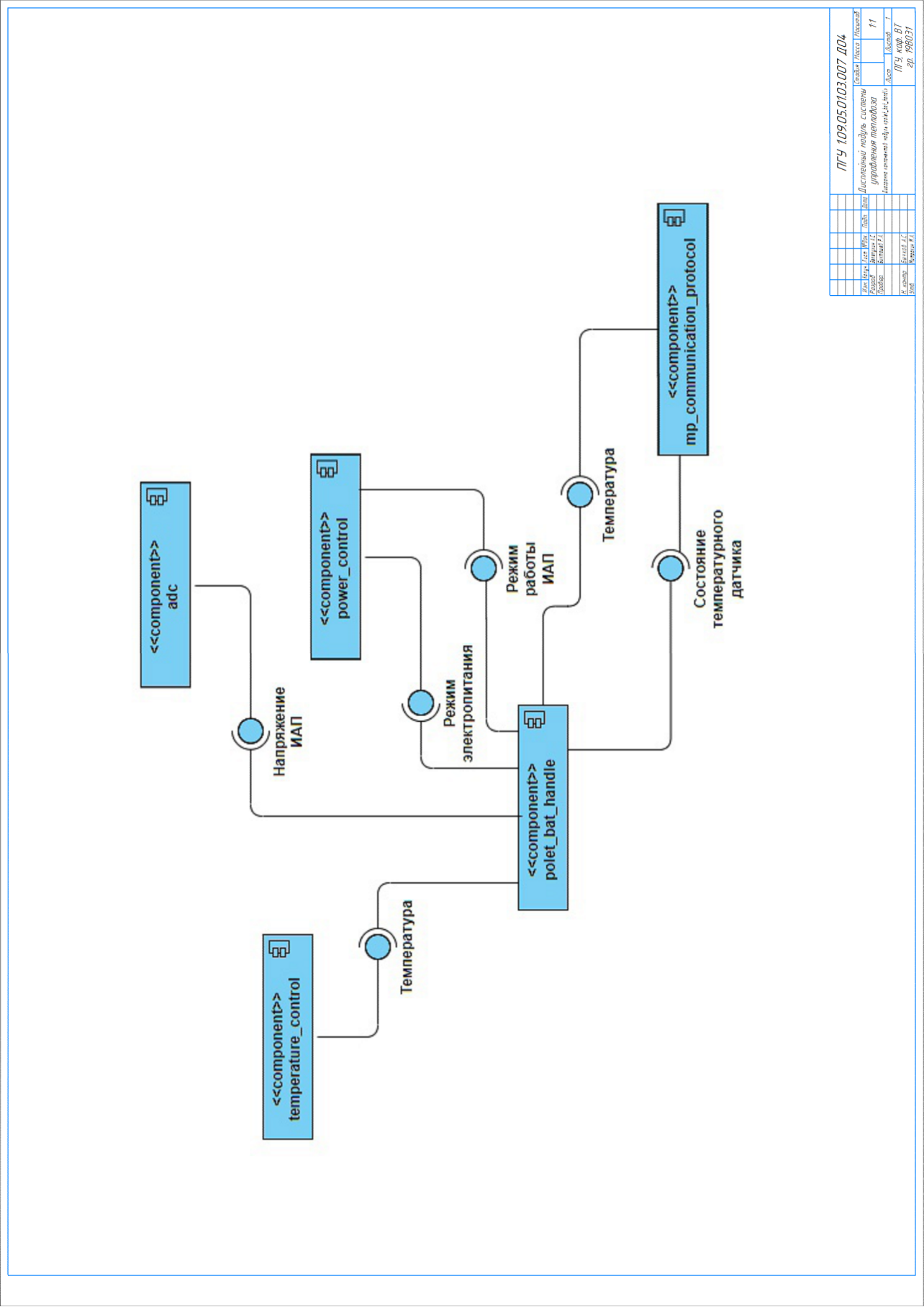
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.03.001 ПЗ

Приложение А.5 – Диаграмма компонентов: модуль «matrix_keyboard»



Приложение А.6 – Диаграмма компонентов: модуль «polet_bat_handle»



ПГУ 1090501.03.007 Л04									
Изм.	Изм.	Изм.	Изм.	Изм.	Изм.	Изм.	Изм.	Изм.	Изм.
Разраб.	Исполн.	Провер.	Исполн.	Провер.	Исполн.	Провер.	Исполн.	Провер.	Исполн.
Директор	Исполн.	Провер.	Исполн.	Провер.	Исполн.	Провер.	Исполн.	Провер.	Исполн.
И.о.дир.	И.о.исполн.	И.о.провер.	И.о.исполн.	И.о.провер.	И.о.исполн.	И.о.провер.	И.о.исполн.	И.о.провер.	И.о.исполн.
Подп.	Подп.	Подп.	Подп.	Подп.	Подп.	Подп.	Подп.	Подп.	Подп.
ПГУ, каф. ВТ									
зр. 090031									

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.03.001 ПЗ

Приложение Б

(обязательное)

Дисплейный модуль системы управления тепловоза

Листинг программы

					ПГУ 1.090501.03.001 ПЗ	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

```

main.c
/* USER CODE BEGIN Header */
/**

*****
*****
* @file           : main.c
* @brief          : Main program body

*****
*****
* @attention
*
* <h2><center>&copy; Copyright (c) 2022 STMicroelectronics.
* All rights reserved.</center></h2>
*
* This software component is licensed by ST under BSD 3-Clause
license,
* the "License"; You may not use this file except in compliance
with the
* License. You may obtain a copy of the License at:
*
*               opensource.org/licenses/BSD-3-Clause
*

*****
*****
*/
/* USER CODE END Header */
/* Includes -----
-----*/
#include "main.h"
#include "usb_device.h"

/* Private includes -----
-----*/
/* USER CODE BEGIN Includes */

#include "polet_bat_handle.h"
#include "matrix_keyboard.h"
#include "temperature_control.h"
#include "usbd_customhid.h"
#include "adc.h"
#include "brightness.h"
#include "power_control.h"
#include "USB_HID.h"
#include "date_operations.h"
#include "fram.h"
#include "ControlVar.h"
#include "warm_up.h"
#include "fan.h"
#include "buzzer.h"

```

```

#include "usbd_customhid.h"
/* USER CODE END Includes */

/* Private typedef -----
-----*/
/* USER CODE BEGIN PTD */

typedef enum
{
    LED_OK = 0, LED_NO_BAT = 1, LED_NO_COM
} LED_State;

/* USER CODE END PTD */

/* Private define -----
-----*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -----
-----*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables -----
-----*/
ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

IWDG_HandleTypeDef hiwdg;

RTC_HandleTypeDef hrtc;

SPI_HandleTypeDef hspi2;

TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim4;

UART_HandleTypeDef huart3;
DMA_HandleTypeDef hdma_usart3_tx;

/* USER CODE BEGIN PV */

uint8_t one_ms_is_elapsed_flag;

/* USER CODE END PV */

/* Private function prototypes -----
-----*/

```

```

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_ADC1_Init(void);
static void MX_RTC_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM4_Init(void);
static void MX_USART3_UART_Init(void);
static void MX_IWDG_Init(void);
static void MX_SPI2_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code -----
-----*/
/* USER CODE BEGIN 0 */

volatile const uint32_t FLASH_MINIMAL_DATE = 0x07E70516;
/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */
    /* USER CODE END 1 */

    /* MCU Configuration-----
    -----*/

    /* Reset of all peripherals, Initializes the Flash interface
    and the Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */
    HAL_Delay(3000);
    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.03.001 ПЗ

Лист

72

```

MX_DMA_Init();
MX_ADC1_Init();
MX_RTC_Init();
MX_TIM1_Init();
MX_TIM2_Init();
MX_TIM3_Init();
MX_TIM4_Init();
MX_USART3_UART_Init();
MX_IWDG_Init();
MX_USB_DEVICE_Init();
MX_SPI2_Init();
/* USER CODE BEGIN 2 */
//-----Variables-----
-----
-----//
// HAL_GPIO_WritePin(DOUT_LCD_EN_GPIO_Port, DOUT_LCD_EN_Pin,
GPIO_PIN_SET);
HAL_Delay(1000);

#ifndef IWDG_DISABLE
HAL_IWDG_Refresh(&hiwdg);
#endif

uint8_t timer_1ms = 0;
uint8_t timer_10ms = 0;
uint8_t timer_100ms = 0;
uint8_t timer_500ms = 0;
uint8_t timer_1s = 0;
uint8_t timer_5s = 0;

LED_State led_state = LED_NO_COM;

one_ms_is_elapsed_flag = 0;

Pin_Struct_Type tmp_pin_array[11];

//-----Modules Init-----
-----
-----//

#ifndef IWDG_DISABLE
HAL_IWDG_Refresh(&hiwdg);
#endif

tmp_pin_array[0].pin = ROW_1_Pin;
tmp_pin_array[0].port = ROW_1_GPIO_Port;
tmp_pin_array[1].pin = ROW_2_Pin;
tmp_pin_array[1].port = ROW_2_GPIO_Port;
tmp_pin_array[2].pin = ROW_3_Pin;
tmp_pin_array[2].port = ROW_3_GPIO_Port;
tmp_pin_array[3].pin = ROW_4_Pin;
tmp_pin_array[3].port = ROW_4_GPIO_Port;

```

```

tmp_pin_array[4].pin = ROW_5_Pin;
tmp_pin_array[4].port = ROW_5_GPIO_Port;
tmp_pin_array[5].pin = COLUMN_1_Pin;
tmp_pin_array[5].port = COLUMN_1_GPIO_Port;
tmp_pin_array[6].pin = COLUMN_2_Pin;
tmp_pin_array[6].port = COLUMN_2_GPIO_Port;
tmp_pin_array[7].pin = COLUMN_3_Pin;
tmp_pin_array[7].port = COLUMN_3_GPIO_Port;
tmp_pin_array[8].pin = COLUMN_4_Pin;
tmp_pin_array[8].port = COLUMN_4_GPIO_Port;
tmp_pin_array[9].pin = COLUMN_5_Pin;
tmp_pin_array[9].port = COLUMN_5_GPIO_Port;
tmp_pin_array[10].pin = COLUMN_6_Pin;
tmp_pin_array[10].port = COLUMN_6_GPIO_Port;
Matrix_Keyboard_Init(tmp_pin_array);

ADC_Handle_Init(&hadc1, TRUE);

__HAL_SPI_ENABLE(&hspi2);
Temperature_Control_Init(&hspi2, DOUT_SPI1_CS_TEMP_GPIO_Port,
DOUT_SPI1_CS_TEMP_Pin);

FAN_Init(&htim3, TIM_CHANNEL_1, TIM_CHANNEL_2, 0);
FAN_Off(0);
FAN_Off(1);

WARM_UP_Init(DOUT_HEATING_GPIO_Port, DOUT_HEATING_Pin);

for (uint8_t counter = 0; counter < 50; counter++)
{
    HAL_Delay(10);
    ADC_Handle_Update_Value();
}
for (uint8_t counter = 0; counter < 20; counter++)
{
    HAL_Delay(250);

    ADC_Handle_Update_Value();
    Temperature_Control_Update_MC_Data();
    Temperature_Control_Update_Sensor_Data();

    WARM_UP_Analyze_1s(Temperature_Control_Get_Temp(),
Temperature_Control_Get_Sensor_Status());

#ifdef IWDG_DISABLE
    HAL_IWDG_Refresh(&hiwdg);
#endif
}
Temperature_Control_Set_Regular_K();

tmp_pin_array[0].pin = DIN_110V_Pin;
tmp_pin_array[0].port = DIN_110V_GPIO_Port;

```

```

tmp_pin_array[1].pin = DIN_MBOARD_IS_ON_Pin;
tmp_pin_array[1].port = DIN_MBOARD_IS_ON_GPIO_Port;
tmp_pin_array[2].pin = DOUT_MBOARD_POWER_EN_Pin;
tmp_pin_array[2].port = DOUT_MBOARD_POWER_EN_GPIO_Port;
tmp_pin_array[3].pin = DOUT_AUDIO_EN_Pin;
tmp_pin_array[3].port = DOUT_AUDIO_EN_GPIO_Port;
Power_Control_Init(Temperature_Control_Get_Temp(),
tmp_pin_array, Temperature_Control_Get_Sensor_Status());

tmp_pin_array[2].pin = DIN_BATTERY_CHARGE_STATUS_Pin;
tmp_pin_array[2].port = DIN_BATTERY_CHARGE_STATUS_GPIO_Port;
tmp_pin_array[3].pin = DOUT_BATTERY_CHARGE_EN_Pin;
tmp_pin_array[3].port = DOUT_BATTERY_CHARGE_EN_GPIO_Port;
tmp_pin_array[4].pin = DOUT_BATTERY_BOOST_EN_Pin;
tmp_pin_array[4].port = DOUT_BATTERY_BOOST_EN_GPIO_Port;
tmp_pin_array[5].pin = DIN_BATTERY_FAULT_Pin;
tmp_pin_array[5].port = DIN_BATTERY_FAULT_GPIO_Port;
tmp_pin_array[6].pin = GPIO_PIN_7;
tmp_pin_array[6].port = GPIOC;
Polet_BAT_Init(Temperature_Control_Get_Temp(),
Temperature_Control_Get_Sensor_Status(), tmp_pin_array);

#ifdef FLASH_DISABLE
    FRAM_Init(&hspi2, DOUT_SPI1_CS_FLASH_GPIO_Port,
DOUT_SPI1_CS_FLASH_Pin);
    CtrlVar_Init();
#endif

tmp_pin_array[0].pin = DOUT_LCD_EN_Pin;
tmp_pin_array[0].port = DOUT_LCD_EN_GPIO_Port;

Brightness_Init(tmp_pin_array[0], Polet_BAT_Get_Status(),
ADC_Handle_Get_Value(ADC_Channel_Bright), 3,
(CtrlVar_GetValue_u8(Prm_u8_ManualBright) &
CV_BRIGHT_MANUAL_EN_MASK_INV), Temperature_Control_Get_Temp());
if ((CtrlVar_GetValue_u8(Prm_u8_ManualBright) &
CV_BRIGHT_MANUAL_EN_MASK) != 0)
{
    Brightness_Set_Manual_Mode(Brightness_Switch_Mode_MP,
CtrlVar_GetValue_u8(Prm_u8_ManualBright) &
CV_BRIGHT_MANUAL_EN_MASK_INV);
    Brightness_SaveMode();
}

USB_HID_Init();

Date_Init(&hrtc, FLASH_MINIMAL_DATE);

Buzzer_Init(&htim1, TIM_CHANNEL_1, BUZZER_VOLUME);

FAN_Enable();

```

```

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET);
    HAL_Delay(1000);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_SET);

//-----Cycle start-----
//-----
//-----//
#ifndef IWDG_DISABLE
    HAL_IWDG_Refresh(&hiwdg);
#endif

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */

    __HAL_SPI_ENABLE(&hspi2);

    while (1)
    {
//-----1 ms-----
//-----
//-----//
        if (one_ms_is_elapsed_flag == 1)
        {
            one_ms_is_elapsed_flag = 0;

            Polet_BAT_Analyze_STAT();

Power_Control_Perform_Transition(Matrix_Keyboard_Get_FuncBTN_State(M
AXRIX_KEYBOARD_FUNC_POWER_SWITCH), Temperature_Control_Get_Temp(),
Polet_BAT_Get_Status(),
                                Temperature_Control_Get_Sensor_Status());

#ifdef DEBUG_BUZZER_VOLUME_CONTROL

Buzzer_Debug_Control_Volume(Matrix_Keyboard_Get_Key_State(BUTTON_IND
EX_SB14), Matrix_Keyboard_Get_Key_State(BUTTON_INDEX_SB13));
#endif
//-----10 ms-----
//-----
//-----//
            if (++timer_1ms >= 10)
            {
                timer_1ms = 0;

                Buzzer_Timer_10ms();

                Matrix_Keyboard_Check_All_Buttons();

                ADC_Handle_Update_Value();

```

```
Brightness_Update_Data(Matrix_Keyboard_Get_FuncBTN_State(MAXRIX_KEYBOARD_FUNC_BRIGHTNESS_INC),
```

```
Matrix_Keyboard_Get_FuncBTN_State(MAXRIX_KEYBOARD_FUNC_BRIGHTNESS_DEC), ADC_Handle_Get_Value(ADC_Channel_Bright),  
Polet_BAT_Get_Status(),
```

```
Temperature_Control_Get_Temp());
```

```
CtrlVar_Upd_Next_Val();
```

```
USB_HID_10ms();
```

```
//-----100 ms-----
```

```
-----
```

```
-----//
```

```
if (++timer_10ms >= 10)
```

```
{
```

```
timer_10ms = 0;
```

```
if (led_state == LED_NO_BAT)
```

```
{
```

```
HAL_GPIO_TogglePin(DOUT_LED_TEST_GPIO_Port, DOUT_LED_TEST_Pin);
```

```
}
```

```
//-----500 ms-----
```

```
-----
```

```
-----//
```

```
if (++timer_100ms >= 5)
```

```
{
```

```
timer_100ms = 0;
```

```
if (Polet_BAT_Is_Exist() == FALSE)
```

```
{
```

```
led_state = LED_NO_BAT;
```

```
}
```

```
else
```

```
{
```

```
if (USB_HID_CommunicationIsOK() ==
```

```
TRUE)
```

```
{
```

```
led_state = LED_OK;
```

```
}
```

```
else
```

```
{
```

```
led_state = LED_NO_COM;
```

```
}
```

```
}
```

```
if (led_state == LED_OK)
```

```
{
```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.03.001 ПЗ

Лист

77

```

HAL_GPIO_TogglePin(DOUT_LED_TEST_GPIO_Port, DOUT_LED_TEST_Pin);
    }
    else
    {
        if (led_state == LED_NO_COM)
        {

HAL_GPIO_WritePin(DOUT_LED_TEST_GPIO_Port, DOUT_LED_TEST_Pin,
GPIO_PIN_SET);

        }
    }

    USB_HID_Push_Errors();

    Brightness_Update_500ms();

//-----1000 ms-----
//-----
//-----//
        if (++timer_500ms >= 2)
        {
            timer_500ms = 0;

            Date_Update_1s();

Polet_BAT_1s_Update(Temperature_Control_Get_Temp(),
Temperature_Control_Get_Sensor_Status(),
Date_Get_Difference(Date_GetCURRENT(), Date_GetSET()));

WARM_UP_Analyze_1s(Temperature_Control_Get_Temp(),
Temperature_Control_Get_Sensor_Status());

FAN_Analyze_1s(Power_Control_Get_State(),
Temperature_Control_Get_Temp(), Power_Control_IsDisabledByUser());

Temperature_Control_Update_MC_Data();

Temperature_Control_Update_Sensor_Data();
            Temperature_Control_1s();

#ifdef DEBUG_SET_CONSTANT_BRIGHTNESS

Brightness_Set_Manual_Mode(Brightness_Switch_Mode_MP,
DEBUG_SET_CONSTANT_BRIGHTNESS);
#endif

#ifdef IWDG_DISABLE

```

```

                                HAL_IWDG_Refresh(&hiwdg);
#endifif

//-----5 s-----
//-----//
                                if (++timer_1s >= 5)
                                {
                                    timer_1s = 0;

#ifndef FLASH_DISABLE
                                    CtrlVar_Upd_Values();
#endifif
//-----5 m-----
//-----//

                                if (++timer_5s >= 60)
                                {
                                    timer_5s = 0;

#ifndef FLASH_DISABLE
                                    Date_Update_5m();
#endifif

                                    uint8_t FTEST = 0;

                                    if (FTEST)
                                    {

CtrlVar_SetValue_u8(Prm_u8_FLG_init, 0x55,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u8(Prm_u8_CNT_crg_miss, 0,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u8(Prm_u8_FLAGS, 0b00000010,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u8(Prm_u8_ManualBright, 20,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u16(Prm_u16_set_date, 0b0000011010100101,
setPrmTChange_ApplayAndSave, fram_WrToAll); //05.05.2023

CtrlVar_SetValue_u16(Prm_u16_cur_date, 0,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u16(Prm_u16_crg_date, 0b0000011010100101,
setPrmTChange_ApplayAndSave, fram_WrToAll); //05.05.2023

```

```

CtrlVar_SetValue_u16(Prm_u16_CNT_crg_end, 0,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u16(Prm_u16_CNT_err, 0,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u16(Prm_u16_CNT_highT, 0,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u16(Prm_u16_CNT_lowT, 0,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u16(Prm_u16_CNT_date_err, 0,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u16(Prm_u16_cur_time, 0,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u32(Prm_u32_serial, 1, setPrmTChange_ApplayAndSave,
fram_WrToAll);

CtrlVar_SetValue_u32(Prm_u32_work_time, 0,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValue_u32(Prm_u32_crg_time, 0,
setPrmTChange_ApplayAndSave, fram_WrToAll);

CtrlVar_SetValueU16_f16(Prm_f16_min_T,
Temperature_Control_Get_Temp(), setPrmTChange_ApplayAndSave,
fram_WrToAll);

CtrlVar_SetValueU16_f16(Prm_f16_min_T,
Temperature_Control_Get_Temp(), setPrmTChange_ApplayAndSave,
fram_WrToAll);

}

}

//-----5 m ends-----
-----
-----//

}

//-----5 s ends-----
-----
-----//

}

```

```

//-----1000 ms ends-----
//
}
//-----500 ms ends-----
//
}
//-----100 ms ends-----
//
}
//-----10 ms ends-----
//
}
//-----1 ms ends-----
//

/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

```

matrix_keyboard.c

```

/*
 * This module provide matrix keyboard work
 *
 * Module use:
 * 1)Execute Matrix_Keyboard_Init
 * 2)Execute Matrix_Keyboard_Check_All_Buttons every n
milliseconds. Recommended N value - 10 milliseconds
 * *)For get USB HID keyboard message with key data use
Matrix_Keyboard_Get_Key_Data
 *
 * Date 12.05.2023
 */

//-----Not exported includes-----
//
#include "matrix_keyboard.h"

```

```

//-----Not exported defines-----
//

//Pressed key state. Should be not equals
MAXRIX_KEYBOARD_NOT_PRESSED_STATE
#define MAXRIX_KEYBOARD_KEY_PRESSED
    (GPIO_PIN_RESET)

//Not pressed key state. Should be not equals
MAXRIX_KEYBOARD_PRESSED_STATE
#define MAXRIX_KEYBOARD_KEY_NOT_PRESSED
    (GPIO_PIN_SET)

//Get button data by button index
#define BUTTON(value)
    Keyboard.buttons[value / MAXRIX_KEYBOARD_COLLIS_AMOUNT][value %
MAXRIX_KEYBOARD_COLLIS_AMOUNT]

//Minimal timer value for "pressed" state
#define TIMER_PRESSED_VALUE
    (4)

//Timer decrement step value
#define TIMER_DEC_STEP
    (10)

//Maximal timer value for "unpressed" state
#define TIMER_MAX_UNPRESSED_VALUE (0)

//"Holded" state timer value
#define TIMER_HOLDED_VALUE
    (2000)

//
#define MAXRIX_KEYBOARD_MAX_PRESSED
    (2)

#define MAXRIX_KEYBOARD_REPEAT_DOWNCOUNTER_MAXVAL (0xFF)
//-----Not exported structs-----
//

/**
 * @brief Filter for antybounce
 */
typedef struct
{
    uint16_t timer; /*!< Timer for antybounce filtration */
    Button_State button_state; /*!< Current button state */
} AntyBounce_Filter_Handle;

```

```

/**
 * @brief One button handle
 */
typedef struct
{
    //bit 15 - right GUI
    //bit 14 - right ALT
    //bit 13 - right SHIFT
    //bit 12 - right CONTROL
    //bit 11 - left GUI
    //bit 10 - left ALT
    //bit 9 - left SHIFT
    //bit 8 - left CONTROL
    //bit 7..0 - key scancode
    uint16_t code; /*!< Key Code */
    Button_Mode mode; /*!< Mode of button */
    uint8_t event_flag; /*!< Button event flag */
    AntyBounce_Filter_Handle filter; /*!< AntyBounce filter
structure */
    uint8_t repeat_downcounter;
} Button_Handle;

/**
 * @brief Structure for matrix keyboard data
 */
typedef struct
{
    Pin_Struct_Type rows[MAXRIX_KEYBOARD_ROWS_AMOUNT]; /*!< Row
pins data */
    Pin_Struct_Type colls[MAXRIX_KEYBOARD_COLLIS_AMOUNT]; /*!< Coll
pins data */
    Button_Handle
buttons[MAXRIX_KEYBOARD_ROWS_AMOUNT][MAXRIX_KEYBOARD_COLLIS_AMOUNT];
/*!< Buttons data */
    uint8_t amount_of_pressed_buttons; /*!< Amount of pressed in
that moment buttons */
    uint8_t repeat_counter_zero; /*!< Amount of "No keys pressed"
signal repeats (sends before "Key pressed" signal) */
    uint8_t repeat_counter; /*!< Amount of "Key pressed" signal
repeats */
    uint8_t key_data[MAXRIX_KEYBOARD_MESSAGE_LENGTH]; /*!< Pressed
key code */

    uint8_t inEvent;
    uint8_t eventChanged;

/*

uint8_t prsdBtnExist;
uint8_t eventFlag;
uint8_t prsdBtnRowIndex;
uint8_t prsdBtnCollIndex;

```

```

void BtnExist;
void BtnChanged;

*/
} Matrix_Keyboard_Handle;

typedef struct
{
    uint8_t isPressed;
    uint8_t keyID;
} PressedKey_Handle;
//-----Not exported constantes-----
-----//

//Amount of key data transmit for "Button_Mode_At_Down" and
"Button_Mode_At_Up" button types
const uint8_t REPEAT_COUNTER_MAX = 3;

const uint8_t REPEAT_COUNTER_ZERO_MAX = 1;

//-----Not exported variables-----
-----//

Matrix_Keyboard_Handle Keyboard;

uint8_t funkBTN_array[MAXRIX_KEYBOARD_FUNC_PLAST];

PressedKey_Handle pressedKeys[MAXRIX_KEYBOARD_MAX_PRESSED];

//-----Not exported function
prototypes-----
-----//

void Matrix_Keyboard_Search_Pressed_Key();

void Matrix_Keyboard_Send_Signal_On_Row(uint8_t row_index);
void Matrix_Keyboard_Clear_Row(uint8_t row_index);
void Matrix_Keyboard_Send_Signal_On_Coll(uint8_t row_index);
void Matrix_Keyboard_Clear_Coll(uint8_t row_index);

void Matrix_Keyboard_Poll_Coll(uint8_t coll_index);
void Matrix_Keyboard_Poll_Row(uint8_t row_index);

//-----Exported function
definitions-----
-----//

/*

```

```

*   @brief      Initialize module
*   @param      Pins data array
*   @note       Pins data array should include:
MAXRIX_KEYBOARD_ROWS_AMOUNT rows pins and
MAXRIX_KEYBOARD_COLLIS_AMOUNT colls pins
*   @retval     None
*/
void Matrix_Keyboard_Init(Pin_Struct_Type *pins)
{
    for (uint8_t index = 0; index < MAXRIX_KEYBOARD_ROWS_AMOUNT;
index++)
    {
        Keyboard.rows[index].pin = pins->pin;
        Keyboard.rows[index].port = pins->port;
        pins++;
    }

    for (uint8_t index = 0; index < MAXRIX_KEYBOARD_COLLIS_AMOUNT;
index++)
    {
        Keyboard.colls[index].pin = pins->pin;
        Keyboard.colls[index].port = pins->port;
        pins++;
    }

    #if (PROJECT_VARIETY == 0)

        //SB1 - "1" button - send "1"
        BUTTON(BUTTON_INDEX_SB1).code = 0x001E;
        BUTTON(BUTTON_INDEX_SB1).mode = Button_Mode_Enabled;

        //SB2 - "2" button - send "2"
        BUTTON(BUTTON_INDEX_SB2).code = 0x001F;
        BUTTON(BUTTON_INDEX_SB2).mode = Button_Mode_Enabled;

        //SB3 - "3" button - send "3"
        BUTTON(BUTTON_INDEX_SB3).code = 0x0020;
        BUTTON(BUTTON_INDEX_SB3).mode = Button_Mode_Enabled;

        //SB4 - "4" button - send "4"
        BUTTON(BUTTON_INDEX_SB4).code = 0x0021;
        BUTTON(BUTTON_INDEX_SB4).mode = Button_Mode_Enabled;

        //SB5 - "5" button - send "5"
        BUTTON(BUTTON_INDEX_SB5).code = 0x0022;
        BUTTON(BUTTON_INDEX_SB5).mode = Button_Mode_Enabled;

        //SB6 - "6" button - send "6"
        BUTTON(BUTTON_INDEX_SB6).code = 0x0023;
        BUTTON(BUTTON_INDEX_SB6).mode = Button_Mode_Enabled;

        //SB7 - "7" button - send "7"

```

```

BUTTON(BUTTON_INDEX_SB7).code = 0x0024;
BUTTON(BUTTON_INDEX_SB7).mode = Button_Mode_Enabled;

//SB8 - "8" button - send "8"
BUTTON(BUTTON_INDEX_SB8).code = 0x0025;
BUTTON(BUTTON_INDEX_SB8).mode = Button_Mode_Enabled;

//SB9 - "9" button - send "9"
BUTTON(BUTTON_INDEX_SB9).code = 0x0026;
BUTTON(BUTTON_INDEX_SB9).mode = Button_Mode_Enabled;

//SB10 - "0" button - send "0"
BUTTON(BUTTON_INDEX_SB10).code = 0x0027;
BUTTON(BUTTON_INDEX_SB10).mode = Button_Mode_Enabled;

//SB11 - "E" button - send "Enter"
BUTTON(BUTTON_INDEX_SB11).code = 0x0058;
BUTTON(BUTTON_INDEX_SB11).mode = Button_Mode_Enabled;

//SB12 - "A <-> 1" button - send "Alt + Ctrl + H"
BUTTON(BUTTON_INDEX_SB12).code = 0x050B;
BUTTON(BUTTON_INDEX_SB12).mode = Button_Mode_At_Down;

//SB13 - "\|/" button - send "Cursor_Down"
BUTTON(BUTTON_INDEX_SB13).code = 0x0051;
BUTTON(BUTTON_INDEX_SB13).mode = Button_Mode_Enabled;

//SB14 - "/|\\" button - send "Cursor_Up"
BUTTON(BUTTON_INDEX_SB14).code = 0x0052;
BUTTON(BUTTON_INDEX_SB14).mode = Button_Mode_Enabled;

//SB15 - "<-" button - send "Cursor_Left"
BUTTON(BUTTON_INDEX_SB15).code = 0x0050;
BUTTON(BUTTON_INDEX_SB15).mode = Button_Mode_Enabled;

//SB16 - "->" button - send "Cursor_Right"
BUTTON(BUTTON_INDEX_SB16).code = 0x004F;
BUTTON(BUTTON_INDEX_SB16).mode = Button_Mode_Enabled;

//SB17 - "C" button - send "Backspace"
BUTTON(BUTTON_INDEX_SB17).code = 0x002A;
BUTTON(BUTTON_INDEX_SB17).mode = Button_Mode_Enabled;

//SB18 - "Settings" button - send "Alt + Ctrl + G"
BUTTON(BUTTON_INDEX_SB18).code = 0x050A;
BUTTON(BUTTON_INDEX_SB18).mode = Button_Mode_At_Down;

//SB19 - "Decrement brightness" button - send "Ctrl + F5"
BUTTON(BUTTON_INDEX_SB19).code = 0x013E;
BUTTON(BUTTON_INDEX_SB19).mode = Button_Mode_At_Down;

//SB20 - "Increment brightness" button - send "Ctrl + F6"

```

```

BUTTON(BUTTON_INDEX_SB20).code = 0x013F;
BUTTON(BUTTON_INDEX_SB20).mode = Button_Mode_At_Down;

//SB21 - "V = 0" button - send "Alt + Ctrl + F"
BUTTON(BUTTON_INDEX_SB21).code = 0x0509;
BUTTON(BUTTON_INDEX_SB21).mode = Button_Mode_At_Down;

//SB22 - "V > 0" button - send "Alt + Ctrl + E"
BUTTON(BUTTON_INDEX_SB22).code = 0x0508;
BUTTON(BUTTON_INDEX_SB22).mode = Button_Mode_At_Down;

//SB23 - "Language" button - send "Alt + Ctrl + D"
BUTTON(BUTTON_INDEX_SB23).code = 0x0507;
BUTTON(BUTTON_INDEX_SB23).mode = Button_Mode_At_Down;

//SB24 - "i" button - send "Alt + Ctrl + C"
BUTTON(BUTTON_INDEX_SB24).code = 0x0506;
BUTTON(BUTTON_INDEX_SB24).mode = Button_Mode_At_Down;

//SB25 - "Select of section" button (right part) - send "Alt +
Ctrl + I"
BUTTON(BUTTON_INDEX_SB25).code = 0x050C;
BUTTON(BUTTON_INDEX_SB25).mode = Button_Mode_At_Down;

//SB26 - "Select of section" button (left part) - send "Alt +
Ctrl + B"
BUTTON(BUTTON_INDEX_SB26).code = 0x0505;
BUTTON(BUTTON_INDEX_SB26).mode = Button_Mode_At_Down;

//SB27" - "Security" - send "Alt + Ctrl + A"
BUTTON(BUTTON_INDEX_SB27).code = 0x0504;
BUTTON(BUTTON_INDEX_SB27).mode = Button_Mode_At_Down;

BUTTON(BUTTON_INDEX_NONE1).code = 0;
BUTTON(BUTTON_INDEX_NONE1).mode = Button_Mode_Disabled;

BUTTON(BUTTON_INDEX_NONE2).code = 0;
BUTTON(BUTTON_INDEX_NONE2).mode = Button_Mode_Disabled;

BUTTON(BUTTON_INDEX_NONE3).code = 0;
BUTTON(BUTTON_INDEX_NONE3).mode = Button_Mode_Disabled;

funkBTN_array[MAXRIX_KEYBOARD_FUNC_BRIGHTNESS_INC] =
BUTTON_INDEX_SB20;
funkBTN_array[MAXRIX_KEYBOARD_FUNC_BRIGHTNESS_DEC] =
BUTTON_INDEX_SB19;
funkBTN_array[MAXRIX_KEYBOARD_FUNC_POWER_SWITCH] =
BUTTON_INDEX_SB18;

#elif (PROJECT_VARIETY == 1)

//SB1 - "1" button - send "1"

```

```

BUTTON(BUTTON_INDEX_SB1).code = 0x001E;
BUTTON(BUTTON_INDEX_SB1).mode = Button_Mode_At_Down;

//SB2 - "2" button - send "2"
BUTTON(BUTTON_INDEX_SB2).code = 0x001F;
BUTTON(BUTTON_INDEX_SB2).mode = Button_Mode_At_Down;

//SB3 - "3" button - send "3"
BUTTON(BUTTON_INDEX_SB3).code = 0x0020;
BUTTON(BUTTON_INDEX_SB3).mode = Button_Mode_At_Down;

//SB4 - "4" button - send "4"
BUTTON(BUTTON_INDEX_SB4).code = 0x0021;
BUTTON(BUTTON_INDEX_SB4).mode = Button_Mode_At_Down;

//SB5 - "5" button - send "5"
BUTTON(BUTTON_INDEX_SB5).code = 0x0022;
BUTTON(BUTTON_INDEX_SB5).mode = Button_Mode_At_Down;

//SB6 - "6" button - send "6"
BUTTON(BUTTON_INDEX_SB6).code = 0x0023;
BUTTON(BUTTON_INDEX_SB6).mode = Button_Mode_At_Down;

//SB7 - "7" button - send "7"
BUTTON(BUTTON_INDEX_SB7).code = 0x0024;
BUTTON(BUTTON_INDEX_SB7).mode = Button_Mode_At_Down;

//SB8 - "8" button - send "8"
BUTTON(BUTTON_INDEX_SB8).code = 0x0025;
BUTTON(BUTTON_INDEX_SB8).mode = Button_Mode_At_Down;

//SB9 - "9" button - send "9"
BUTTON(BUTTON_INDEX_SB9).code = 0x0026;
BUTTON(BUTTON_INDEX_SB9).mode = Button_Mode_At_Down;

//SB10 - "0" button - send "0"
BUTTON(BUTTON_INDEX_SB10).code = 0x0027;
BUTTON(BUTTON_INDEX_SB10).mode = Button_Mode_At_Down;

//SB11 - "E" button - send "Enter"
BUTTON(BUTTON_INDEX_SB11).code = 0x0058;
BUTTON(BUTTON_INDEX_SB11).mode = Button_Mode_At_Down;

//SB12 - "\|" button - send "Cursor_Down"
BUTTON(BUTTON_INDEX_SB12).code = 0x0051;
BUTTON(BUTTON_INDEX_SB12).mode = Button_Mode_At_Down;

//SB13 - "/|\\" button - send "Cursor_Up"
BUTTON(BUTTON_INDEX_SB13).code = 0x0052;
BUTTON(BUTTON_INDEX_SB13).mode = Button_Mode_At_Down;

//SB14 - "->" button - send "Cursor_Right"

```

```

BUTTON(BUTTON_INDEX_SB14).code = 0x004F;
BUTTON(BUTTON_INDEX_SB14).mode = Button_Mode_At_Down;

//SB15 - "<-" button - send "Cursor_Left"
BUTTON(BUTTON_INDEX_SB15).code = 0x0050;
BUTTON(BUTTON_INDEX_SB15).mode = Button_Mode_At_Down;

//SB16 - "C" button - send "Backspace"
BUTTON(BUTTON_INDEX_SB16).code = 0x002A;
BUTTON(BUTTON_INDEX_SB16).mode = Button_Mode_At_Down;

//SB17 - "UD" button - send "Alt + Ctrl + I"
BUTTON(BUTTON_INDEX_SB17).code = 0x050C;
BUTTON(BUTTON_INDEX_SB17).mode = Button_Mode_At_Down;

//SB18 - "Decrement brightness" button - send "Alt + Ctrl + H"
BUTTON(BUTTON_INDEX_SB18).code = 0x050B;
BUTTON(BUTTON_INDEX_SB18).mode = Button_Mode_At_Down;

//SB19 - "Increment brightness" button - send "Alt + Ctrl + G"
BUTTON(BUTTON_INDEX_SB19).code = 0x050A;
BUTTON(BUTTON_INDEX_SB19).mode = Button_Mode_At_Down;

//SB20 - "V = 0" button - send "Alt + Ctrl + F"
BUTTON(BUTTON_INDEX_SB20).code = 0x0509;
BUTTON(BUTTON_INDEX_SB20).mode = Button_Mode_At_Down;

//SB21 - "V > 0" button - send "Alt + Ctrl + E"
BUTTON(BUTTON_INDEX_SB21).code = 0x0508;
BUTTON(BUTTON_INDEX_SB21).mode = Button_Mode_At_Down;

//SB22 - "Select of section" button (right part) - send "Alt +
Ctrl + J"
BUTTON(BUTTON_INDEX_SB22).code = 0x050D;
BUTTON(BUTTON_INDEX_SB22).mode = Button_Mode_At_Down;

//SB23 - "Select of section" button (left part) - send "Alt +
Ctrl + D"
BUTTON(BUTTON_INDEX_SB23).code = 0x0507;
BUTTON(BUTTON_INDEX_SB23).mode = Button_Mode_At_Down;

//SB24 - "i" button - send "Alt + Ctrl + C"
BUTTON(BUTTON_INDEX_SB24).code = 0x0506;
BUTTON(BUTTON_INDEX_SB24).mode = Button_Mode_At_Down;

//SB25" - "S" - send "Alt + Ctrl + B"
BUTTON(BUTTON_INDEX_SB25).code = 0x0505;
BUTTON(BUTTON_INDEX_SB25).mode = Button_Mode_At_Down;

//SB26" - "aus" - send "Alt + Ctrl + A"
BUTTON(BUTTON_INDEX_SB26).code = 0x0504;
BUTTON(BUTTON_INDEX_SB26).mode = Button_Mode_At_Down;

```

```

    BUTTON(BUTTON_INDEX_NONE1).code = 0;
    BUTTON(BUTTON_INDEX_NONE1).mode = Button_Mode_Disabled;

    BUTTON(BUTTON_INDEX_NONE2).code = 0;
    BUTTON(BUTTON_INDEX_NONE2).mode = Button_Mode_Disabled;

    BUTTON(BUTTON_INDEX_NONE3).code = 0;
    BUTTON(BUTTON_INDEX_NONE3).mode = Button_Mode_Disabled;

    BUTTON(BUTTON_INDEX_NONE4).code = 0;
    BUTTON(BUTTON_INDEX_NONE4).mode = Button_Mode_Disabled;

    funkBTN_array[MAXRIX_KEYBOARD_FUNC_BRIGHTNESS_INC] =
    BUTTON_INDEX_SB19;
    funkBTN_array[MAXRIX_KEYBOARD_FUNC_BRIGHTNESS_DEC] =
    BUTTON_INDEX_SB18;
    funkBTN_array[MAXRIX_KEYBOARD_FUNC_POWER_SWITCH] =
    BUTTON_INDEX_SB17;

#else

#error "Project variety not include keyboard settings"

#endif

//Buttons initialize
for (uint8_t row_index = 0; row_index <
MAXRIX_KEYBOARD_ROWS_AMOUNT; row_index++)
{
    for (uint8_t coll_index = 0; coll_index <
MAXRIX_KEYBOARD_COLS_AMOUNT; coll_index++)
    {
        Keyboard.buttons[row_index][coll_index].filter.button_state =
        Button_State_Holded;
        Keyboard.buttons[row_index][coll_index].filter.timer =
        0;
        Keyboard.buttons[row_index][coll_index].event_flag =
        FALSE;
    }
}

Keyboard.amount_of_pressed_buttons = 0;
Keyboard.repeat_counter = 0;

Keyboard.key_data[1] = 0;
Keyboard.key_data[3] = 0;
Keyboard.key_data[4] = 0;
Keyboard.key_data[5] = 0;
Keyboard.key_data[6] = 0;
Keyboard.key_data[7] = 0;

```

```

    for (uint8_t i = 0; i < MAXRIX_KEYBOARD_MAX_PRESSED; i++)
    {
        pressedKeys[i].isPressed = 0;
        pressedKeys[i].keyID = 0;
    }
}

uint8_t Matrix_Keyboard_Get_KeyData(uint8_t index)
{
    if (index >= 8)
    {
        return 0;
    }
    else
    {
        return Keyboard.key_data[index];
    }
}

void Matrix_Keyboard_Update_KeyData()
{
    for (uint8_t i = 0; i < 8; i++)
    {
        Keyboard.key_data[i] = 0;
    }

    uint8_t amount = pressedKeys[0].isPressed +
pressedKeys[1].isPressed;
    uint8_t ptr = 2;

    for (uint8_t i = 0; i < amount; i++)
    {
        if (BUTTON(pressedKeys[i].keyID).repeat_downcounter > 0)
        {
            if (BUTTON(pressedKeys[i].keyID).repeat_downcounter !=
MAXRIX_KEYBOARD_REPEAT_DOWNCOUNTER_MAXVAL)
            {
                BUTTON(pressedKeys[i].keyID).repeat_downcounter-
-;
            }
            Keyboard.key_data[0] =
BUTTON(pressedKeys[i].keyID).code >> 8;
            Keyboard.key_data[ptr] =
BUTTON(pressedKeys[i].keyID).code & 0xFF;
            ptr++;
        }
    }
}

uint8_t keyboard_Event_Changed()
{

```

```

        uint8_t res = Keyboard.eventChanged;
        Keyboard.eventChanged = 0;
        return res;
    }

    uint8_t keyboard_In_Event_Stage()
    {
        return Keyboard.inEvent;
    }

    /*
     * @brief    Update all button status and search pressed key
     * @retval   None
     */
    void Matrix_Keyboard_Check_All_Buttons()
    {
        for (uint8_t row_index = 0; row_index <
MAXRIX_KEYBOARD_ROWS_AMOUNT; row_index++)
        {
            Matrix_Keyboard_Send_Signal_On_Row(row_index);

            Matrix_Keyboard_Poll_Row(row_index);

            Matrix_Keyboard_Clear_Row(row_index);
        }

        for (uint8_t coll_index = 0; coll_index <
MAXRIX_KEYBOARD_COLLIS_AMOUNT; coll_index++)
        {
            Matrix_Keyboard_Send_Signal_On_Coll(coll_index);

            Matrix_Keyboard_Poll_Coll(coll_index);

            Matrix_Keyboard_Clear_Coll(coll_index);
        }

        Matrix_Keyboard_Search_Pressed_Key();
    }

    Button_State Matrix_Keyboard_Get_FuncBTN_State(Keyboard_Functtions
func_index)
    {
        return BUTTON(funkBTN_array[func_index]).filter.button_state;
    }

    //-----Not exported function
    definitions-----
    -----//

    /*
     * @brief    Set row in output mode and switch it in reset state
     * @param    Selected row index

```

```

*   @retval   None
*/
void Matrix_Keyboard_Send_Signal_On_Row(uint8_t row_index)
{
    if (row_index < MAXRIX_KEYBOARD_ROWS_AMOUNT)
    {
        GPIO_InitTypeDef GPIO_InitStructure = { 0 };

        GPIO_InitStructure.Pin = Keyboard.rows[row_index].pin;
        GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_OD;
        GPIO_InitStructure.Pull = GPIO_PULLUP;
        GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_LOW;
        HAL_GPIO_Init(Keyboard.rows[row_index].port,
&GPIO_InitStructure);

        HAL_GPIO_WritePin(Keyboard.rows[row_index].port,
Keyboard.rows[row_index].pin, MAXRIX_KEYBOARD_KEY_PRESSED);
    }
}

/*
*   @brief     Set row in input mode
*   @param     Selected row index
*   @retval    None
*/
void Matrix_Keyboard_Clear_Row(uint8_t row_index)
{
    if (row_index < MAXRIX_KEYBOARD_ROWS_AMOUNT)
    {
        HAL_GPIO_WritePin(Keyboard.rows[row_index].port,
Keyboard.rows[row_index].pin, MAXRIX_KEYBOARD_KEY_NOT_PRESSED);

        GPIO_InitTypeDef GPIO_InitStructure = { 0 };

        GPIO_InitStructure.Pin = Keyboard.rows[row_index].pin;
        GPIO_InitStructure.Mode = GPIO_MODE_INPUT;
        GPIO_InitStructure.Pull = GPIO_PULLUP;
        HAL_GPIO_Init(Keyboard.rows[row_index].port,
&GPIO_InitStructure);
    }
}

/*
*   @brief     Set coll in output mode and switch it in reset state
*   @param     Selected coll index
*   @retval    None
*/
void Matrix_Keyboard_Send_Signal_On_Coll(uint8_t coll_index)
{
    if (coll_index < MAXRIX_KEYBOARD_COLLIS_AMOUNT)
    {
        GPIO_InitTypeDef GPIO_InitStructure = { 0 };

```

```

        GPIO_InitStruct.Pin = Keyboard.colls[coll_index].pin;
        GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_OD;
        GPIO_InitStruct.Pull = GPIO_PULLUP;
        GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
        HAL_GPIO_Init(Keyboard.colls[coll_index].port,
&GPIO_InitStruct);

        HAL_GPIO_WritePin(Keyboard.colls[coll_index].port,
Keyboard.colls[coll_index].pin, MAXRIX_KEYBOARD_KEY_PRESSED);
    }
}

/*
 * @brief    Set coll in input mode
 * @param    Selected coll index
 * @retval    None
 */
void Matrix_Keyboard_Clear_Coll(uint8_t coll_index)
{
    if (coll_index < MAXRIX_KEYBOARD_COLLIS_AMOUNT)
    {
        HAL_GPIO_WritePin(Keyboard.colls[coll_index].port,
Keyboard.colls[coll_index].pin, MAXRIX_KEYBOARD_KEY_NOT_PRESSED);

        GPIO_InitTypeDef GPIO_InitStruct = { 0 };

        GPIO_InitStruct.Pin = Keyboard.colls[coll_index].pin;
        GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
        GPIO_InitStruct.Pull = GPIO_PULLUP;
        HAL_GPIO_Init(Keyboard.colls[coll_index].port,
&GPIO_InitStruct);
    }
}

/*
 * @brief    Update pressed key indexes
 * @retval    None
 */
void Matrix_Keyboard_Search_Pressed_Key()
{
    Keyboard.amount_of_pressed_buttons = 0;
    for (uint8_t coll_index = 0; coll_index <
MAXRIX_KEYBOARD_COLLIS_AMOUNT; coll_index++)
    {
        for (uint8_t row_index = 0; row_index <
MAXRIX_KEYBOARD_ROWS_AMOUNT; row_index++)
        {
            switch (Keyboard.buttons[row_index][coll_index].mode)
            {
                case Button_Mode_Enabled:
                case Button_Mode_At_Down:

```

```

        case Button_Mode_No_Send:
        {
            if
(Keyboard.buttons[row_index][coll_index].filter.button_state ==
Button_State_Pressed)
            {
                Keyboard.amount_of_pressed_buttons++;
            }
            break;
        }
        default:
        {
            break;
        }
    }
}

if ((Keyboard.amount_of_pressed_buttons > 2) ||
(Keyboard.amount_of_pressed_buttons == 0))
{
    pressedKeys[0].isPressed = 0;
    pressedKeys[1].isPressed = 0;
    if (Keyboard.inEvent == TRUE)
    {
        Keyboard.inEvent = FALSE;
        Keyboard.eventChanged = TRUE;
    }
    if (Keyboard.amount_of_pressed_buttons > 2)
    {
        for (uint8_t coll_index = 0; coll_index <
MAXRIX_KEYBOARD_COLLIS_AMOUNT; coll_index++)
        {
            for (uint8_t row_index = 0; row_index <
MAXRIX_KEYBOARD_ROWS_AMOUNT; row_index++)
            {

Keyboard.buttons[row_index][coll_index].filter.button_state =
Button_State_Holded;

Keyboard.buttons[row_index][coll_index].repeat_downcounter = 0;

Keyboard.buttons[row_index][coll_index].filter.timer =
TIMER_HOLDED_VALUE;
            }
        }
    }
}
else
{
    for (uint8_t i = 0; i < 2; i++)
    {

```

```

        if ((pressedKeys[i].isPressed == TRUE) &&
            (BUTTON(pressedKeys[i].keyID).filter.button_state !=
            Button_State_Pressed) &&
            ((BUTTON(pressedKeys[i].keyID).repeat_downcounter ==
            MAXRIX_KEYBOARD_REPEAT_DOWNCOUNTER_MAXVAL) ||
            (BUTTON(pressedKeys[i].keyID).repeat_downcounter == 0)))
        {
            pressedKeys[i].isPressed = FALSE;
        }
    }

    uint8_t ptr = 0;
    if ((pressedKeys[0].isPressed == FALSE))
    {
        if ((pressedKeys[1].isPressed == TRUE))
        {
            pressedKeys[0].isPressed = TRUE;
            pressedKeys[1].isPressed = FALSE;
            pressedKeys[0].keyID = pressedKeys[1].keyID;
            ptr++;
        }
    }
    else
    {
        ptr++;
    }

    for (uint8_t coll_index = 0; coll_index <
    MAXRIX_KEYBOARD_COLLIS_AMOUNT; coll_index++)
    {
        for (uint8_t row_index = 0; row_index <
        MAXRIX_KEYBOARD_ROWS_AMOUNT; row_index++)
        {
            switch
            (Keyboard.buttons[row_index][coll_index].mode)
            {
                case Button_Mode_Enabled:
                case Button_Mode_At_Down:
                {
                    if
                    (Keyboard.buttons[row_index][coll_index].filter.button_state ==
                    Button_State_Pressed)
                    {
                        if (((pressedKeys[0].isPressed ==
                        TRUE) && (pressedKeys[0].keyID == (row_index *
                        MAXRIX_KEYBOARD_COLLIS_AMOUNT + coll_index)))
                        || ((pressedKeys[1].isPressed ==
                        TRUE) && (pressedKeys[1].keyID == (row_index *
                        MAXRIX_KEYBOARD_COLLIS_AMOUNT + coll_index))))
                        {
                        }
                    }
                }
            }
        }
    }

```

```

else
{
    if (ptr < 2)
    {
pressedKeys[ptr].isPressed = TRUE;
                                pressedKeys[ptr].keyID =
(row_index * MAXRIX_KEYBOARD_COLLIS_AMOUNT + coll_index);
                                ptr++;
                                }
    }
    }
    break;
}
default:
{
    break;
}
}
}

if (Keyboard.inEvent == FALSE)
{
    if (((pressedKeys[0].isPressed == TRUE) &&
(BUTTON(pressedKeys[0].keyID).repeat_downcounter > 0)) ||
((pressedKeys[1].isPressed == TRUE) &&
(BUTTON(pressedKeys[1].keyID).repeat_downcounter > 0)))
    {
        Keyboard.inEvent = TRUE;
        Keyboard.eventChanged = TRUE;
    }
}
else
{
    if (((pressedKeys[0].isPressed == FALSE) ||
(BUTTON(pressedKeys[0].keyID).repeat_downcounter == 0)) &&
((pressedKeys[1].isPressed == FALSE) ||
(BUTTON(pressedKeys[1].keyID).repeat_downcounter == 0)))
    {
        Keyboard.inEvent = FALSE;
        Keyboard.eventChanged = TRUE;
    }
}
}

/*
 * @brief    Returns key state
 * @param    Selected key index
 * @retval   Button_State status
 */

```

```

Button_State Matrix_Keyboard_Get_Key_State(uint8_t index)
{
    return BUTTON(index).filter.button_state;
}

/*
 * @brief    Poll button on selected row
 * @param    Selected row index
 * @retval    None
 */
void Matrix_Keyboard_Poll_Coll(uint8_t coll_index)
{
    for (uint8_t row_index = 0; row_index <
MAXRIX_KEYBOARD_ROWS_AMOUNT; row_index++)
    {
        switch
(Keyboard.buttons[row_index][coll_index].filter.button_state)
        {
            case (Button_State_Unpressed):
            {
                if
(HAL_GPIO_ReadPin(Keyboard.rows[row_index].port,
Keyboard.rows[row_index].pin) == MAXRIX_KEYBOARD_KEY_PRESSED)
                {

Keyboard.buttons[row_index][coll_index].filter.timer++;

                if
(Keyboard.buttons[row_index][coll_index].filter.timer >=
TIMER_PRESSED_VALUE)
                {

Keyboard.buttons[row_index][coll_index].filter.button_state =
Button_State_Pressed;

                switch
(Keyboard.buttons[row_index][coll_index].mode)
                {
                    case Button_Mode_Enabled:
                    {

Keyboard.buttons[row_index][coll_index].repeat_downcounter =
MAXRIX_KEYBOARD_REPEAT_DOWNCOUNTER_MAXVAL;

                    break;

                    }
                    case Button_Mode_At_Down:
                    {

Keyboard.buttons[row_index][coll_index].repeat_downcounter =
REPEAT_COUNTER_MAX;

                    break;

                    }
                    default:
                    {

```

```

//Keyboard.buttons[row_index][coll_index].repeat_downcounter = 0;
                                break;
                                }
                                }
                                }
                                else
                                {

Keyboard.buttons[row_index][coll_index].filter.timer = 0;
                                }
                                break;
                                }
                                case (Button_State_Pressed):
                                {
                                        if
                                        (HAL_GPIO_ReadPin(Keyboard.rows[row_index].port,
Keyboard.rows[row_index].pin) == MAXRIX_KEYBOARD_KEY_PRESSED)
                                        {

Keyboard.buttons[row_index][coll_index].filter.timer++;
                                                if
                                                (Keyboard.buttons[row_index][coll_index].filter.timer >=
TIMER_HOLDED_VALUE)
                                                {

Keyboard.buttons[row_index][coll_index].filter.button_state =
Button_State_Holded;

Keyboard.buttons[row_index][coll_index].repeat_downcounter = 0;
                                                }
                                                }
                                                else
                                                {
                                                        if
                                                        (Keyboard.buttons[row_index][coll_index].filter.timer >
TIMER_MAX_UNPRESSED_VALUE)
                                                        {

Keyboard.buttons[row_index][coll_index].filter.timer =
TIMER_MAX_UNPRESSED_VALUE;
                                                        }

                                                        if
                                                        (Keyboard.buttons[row_index][coll_index].filter.timer <=
TIMER_DEC_STEP)
                                                        {

Keyboard.buttons[row_index][coll_index].filter.timer = 0;
                                                        }
                                                        else

```

```

        {

Keyboard.buttons[row_index][coll_index].filter.timer -=
TIMER_DEC_STEP;

        }

        if
(Keyboard.buttons[row_index][coll_index].filter.timer <
TIMER_PRESSED_VALUE)
        {

Keyboard.buttons[row_index][coll_index].filter.button_state =
Button_State_Unpressed;

        }
        break;
    }
    case (Button_State_Holded):
    {
        if
(HAL_GPIO_ReadPin(Keyboard.rows[row_index].port,
Keyboard.rows[row_index].pin) == MAXRIX_KEYBOARD_KEY_PRESSED)
        {

Keyboard.buttons[row_index][coll_index].filter.timer =
TIMER_HOLDED_VALUE;
        }
        else
        {
            if
(Keyboard.buttons[row_index][coll_index].filter.timer <=
TIMER_DEC_STEP)
            {

Keyboard.buttons[row_index][coll_index].filter.timer = 0;
            }
            else
            {

Keyboard.buttons[row_index][coll_index].filter.timer -=
TIMER_DEC_STEP;

            }

            if
(Keyboard.buttons[row_index][coll_index].filter.timer <
TIMER_PRESSED_VALUE)
            {

Keyboard.buttons[row_index][coll_index].filter.button_state =
Button_State_Unpressed;

            }
        }
    }
}

```

```

        break;
    }
    default:
    {

Keyboard.buttons[row_index][coll_index].filter.button_state =
Button_State_Holded;

Keyboard.buttons[row_index][coll_index].filter.timer =
TIMER_HOLDED_VALUE;
        break;
    }
    }
}

/*
 * @brief    Poll button on selected row
 * @param    Selected row index
 * @retval   None
 */
void Matrix_Keyboard_Poll_Row(uint8_t row_index)
{
    for (uint8_t coll_index = 0; coll_index <
MAXRIX_KEYBOARD_COLLIS_AMOUNT; coll_index++)
    {
        switch
(Keyboard.buttons[row_index][coll_index].filter.button_state)
        {
            case (Button_State_Unpressed):
            {
                if
(HAL_GPIO_ReadPin(Keyboard.colls[coll_index].port,
Keyboard.colls[coll_index].pin) == MAXRIX_KEYBOARD_KEY_PRESSED)
                {

Keyboard.buttons[row_index][coll_index].filter.timer++;
                    if
(Keyboard.buttons[row_index][coll_index].filter.timer >=
TIMER_PRESSED_VALUE)
                    {

Keyboard.buttons[row_index][coll_index].filter.button_state =
Button_State_Pressed;

                        switch
(Keyboard.buttons[row_index][coll_index].mode)
                        {
                            case Button_Mode_Enabled:
                            {

Keyboard.buttons[row_index][coll_index].repeat_downcounter =
MAXRIX_KEYBOARD_REPEAT_DOWNCOUNTER_MAXVAL;

```

```

        break;
    }
    case Button_Mode_At_Down:
    {

Keyboard.buttons[row_index][coll_index].repeat_downcounter =
REPEAT_COUNTER_MAX;

        break;
    }
    default:
    {

//Keyboard.buttons[row_index][coll_index].repeat_downcounter = 0;
        break;
    }
    }
}
else
{

Keyboard.buttons[row_index][coll_index].filter.timer = 0;
}
break;
}
case (Button_State_Pressed):
{
    if
(HAL_GPIO_ReadPin(Keyboard.colls[coll_index].port,
Keyboard.colls[coll_index].pin) == MAXRIX_KEYBOARD_KEY_PRESSED)
    {

Keyboard.buttons[row_index][coll_index].filter.timer++;
        if
(Keyboard.buttons[row_index][coll_index].filter.timer >=
TIMER_HOLDED_VALUE)
        {

Keyboard.buttons[row_index][coll_index].filter.button_state =
Button_State_Holded;

Keyboard.buttons[row_index][coll_index].repeat_downcounter = 0;
        }
    }
    else
    {
        if
(Keyboard.buttons[row_index][coll_index].filter.timer >
TIMER_MAX_UNPRESSED_VALUE)
        {

```

```

Keyboard.buttons[row_index][coll_index].filter.timer =
TIMER_MAX_UNPRESSED_VALUE;
    }
    if
(Keyboard.buttons[row_index][coll_index].filter.timer <
TIMER_PRESSED_VALUE)
    {

Keyboard.buttons[row_index][coll_index].filter.button_state =
Button_State_Unpressed;
    }
    }
    break;
}
case (Button_State_Holded):
{
    if
(HAL_GPIO_ReadPin(Keyboard.colls[coll_index].port,
Keyboard.colls[coll_index].pin) == MAXRIX_KEYBOARD_KEY_PRESSED)
    {

Keyboard.buttons[row_index][coll_index].filter.timer =
TIMER_HOLDED_VALUE;
    }
    else
    {
        if
(Keyboard.buttons[row_index][coll_index].filter.timer <=
TIMER_DEC_STEP)
        {

Keyboard.buttons[row_index][coll_index].filter.timer = 0;
        }
        else
        {

Keyboard.buttons[row_index][coll_index].filter.timer -=
TIMER_DEC_STEP;
        }

        if
(Keyboard.buttons[row_index][coll_index].filter.timer <
TIMER_PRESSED_VALUE)
        {

Keyboard.buttons[row_index][coll_index].filter.button_state =
Button_State_Unpressed;
        }
    }
    break;
}
}

```

```

        default:
        {

Keyboard.buttons[row_index][coll_index].filter.button_state =
Button_State_Holded;

Keyboard.buttons[row_index][coll_index].filter.timer =
TIMER_HOLDED_VALUE;
                break;
        }
    }
}

```

Приложение В

(обязательное)

Дисплейный модуль системы управления тепловоза

Презентация

					ПГУ 1.090501.03.001 ПЗ	Лист
						105
Изм.	Лист	№ докум.	Подпись	Дата		

ДИСПЛЕЙНЫЙ МОДУЛЬ СИСТЕМЫ УПРАВЛЕНИЯ ТЕПЛОВОЗА

Автор работы: студент гр. 19ВОЭ1 Зелепухин Андрей Сергеевич
Руководитель: к.т.н., доцент каф. ВТ Бикташев Равиль Айнулович

Пенза, 2024

Цель работы

Целью данной работы является разработка программы для микроконтроллера, входящего в состав дисплейного модуля системы управления тепловоза

Постановка задач

Программа, загружаемая в микроконтроллер, должна выполнять следующие задачи:

- 1) сбор и фильтрация сигналов, поступающих на АЦП;
- 2) управление уровнем яркости подсветки экрана и клавиатуры на основе показаний датчика, пользовательских настройках и текущих режимах электропитания и температуры;
- 3) управление охлаждающими вентиляторами;
- 4) сбор и анализ информации о нажатых клавишах клавиатуры;
- 5) осуществление приема и передачи данных между микроконтроллером и материнской платой;
- 6) управление работой источника автономного питания;
- 7) управление режимом электропитания устройства;
- 8) сбор и анализ показаний температурного датчика;
- 9) управление устройством подогрева.

Инструменты разработки и технологии

Среда разработки - STM32CubeIDE

Библиотека для взаимодействия с периферией - HAL

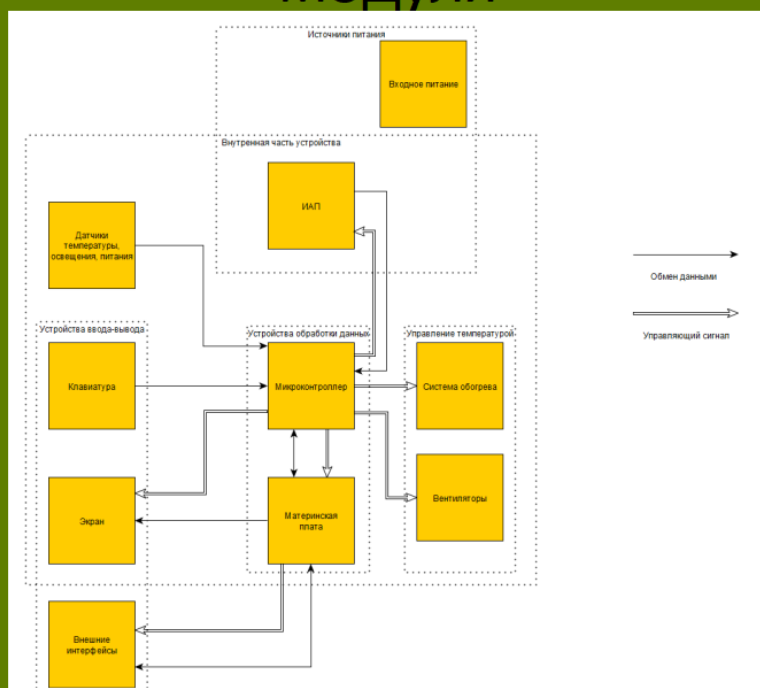
Язык разработки - C

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.03.001 ПЗ

Лист
107

Взаимодействие частей дисплейного модуля



Сравнение с аналогом

Наименование характеристики	Характеристики серии ВС7	Характеристики дисплейного модуля
Процессор	E3940 4 ядра, 1,8 ГГц	Intel Celeron J1900 4 ядра, 2.42 ГГц
Клавиатура	Клавиатура (3 клавиши) Сенсорный экран	Клавиатура (27 клавиш)
Дисплей	10,4" TFT дисплей, VGA 640x480 Автоматическая система регулирования яркости подсветки	10,4" TFT LCD дисплей, 640x480 Автоматическая система регулирования яркости подсветки
Диапазон температуры окружающего воздуха	От -25 до +70 (версия Т3) От -40 до +70 (версия Тх)	От -40 до +70
Интерфейсы	Ethernet, USB, RS232, RS422, MVB, CAN, Audio	RS422/RS485, RS232, Ethernet, CAN, USB, Audio
Входное питание	24 В - 110 В (+/- 40%)	36-160 В Источник автономного питания
Габариты	310x214x45 мм	310x214x101

Цена устройства

Статья затрат	Сумма, рублей для серии	Примечание
Материал	4741,96	
ПКИ	178279,00	
Работы и услуги сторонних организаций производственного характера	3882,84	
Затраты на оплату труда основных производственных рабочих — всего, включая НДФЛ	15297,79	
Заработная плата основных производственных рабочих сборочное производство	4313,92	
Заработная плата основных производственных рабочих в механическом цехе	8988,51	При расчете применен коэффициент многостаночности
Дополнительная заработная плата	1995,36	15% от основной заработной платы
Социальные отчисления и налоги	3334,92	21,8% от затрат на оплату труда
Общепроизводственные расходы	26072,75	196% от основной заработной платы
Общехозяйственные расходы	19421,54	146% от основной заработной платы
Себестоимость	251030,81	
Прибыль	75309,24	30% от себестоимости
Цена	326340,05	

Цена аналогов

Блок индикации Gersys BC3741 (3454130031)

Категории товаров

Автоматика

Аппараты

Вентиляция

Запчасти

Блок индикации Gersys BC3741 (3454130031)

от 1 571 560 р. - В наличии 10000 шт.

1

В корзину

Категория: Запчасти

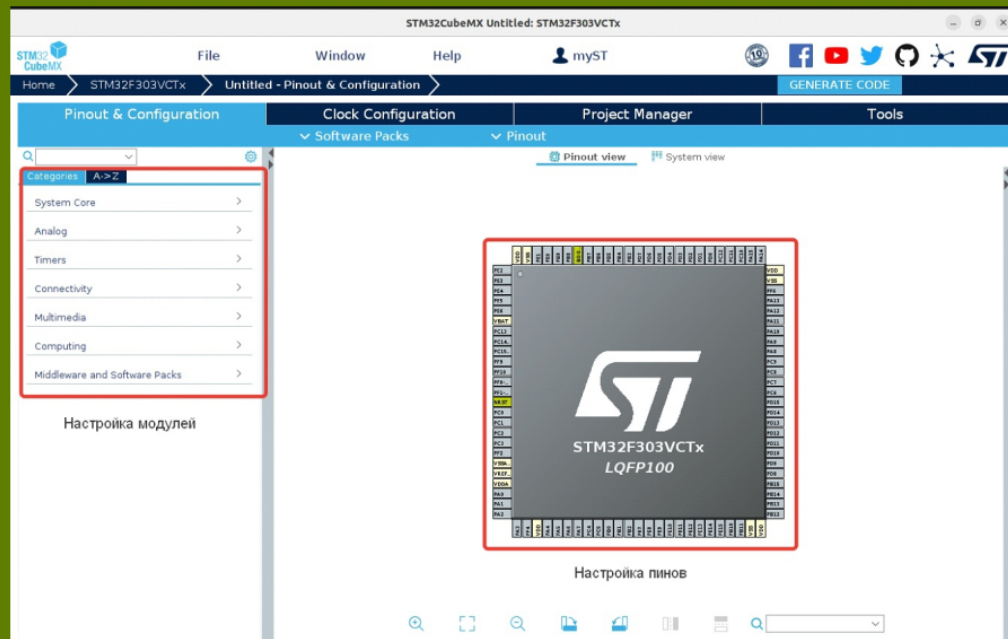
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.03.001 ПЗ

Лист

109

Обеспечение удобства сопровождения проекта



```
#if (PROJECT_VARIETY == 0)
    *
    *
    *
    funkBTN_array[MAXRIX_KEYBOARD_FUNC_BRIGHTNESS_INC]
= BUTTON_INDEX_SB20;
    funkBTN_array[MAXRIX_KEYBOARD_FUNC_BRIGHTNESS_DEC]
= BUTTON_INDEX_SB19;
    funkBTN_array[MAXRIX_KEYBOARD_FUNC_POWER_SWITCH] =
BUTTON_INDEX_SB18;

#elif (PROJECT_VARIETY == 1)
    *
    *
    *
    funkBTN_array[MAXRIX_KEYBOARD_FUNC_BRIGHTNESS_INC]
= BUTTON_INDEX_SB19;
    funkBTN_array[MAXRIX_KEYBOARD_FUNC_BRIGHTNESS_DEC]
= BUTTON_INDEX_SB18;
    funkBTN_array[MAXRIX_KEYBOARD_FUNC_POWER_SWITCH] =
BUTTON_INDEX_SB17;
#else
```

Заключение

В результате выполнения работы было разработано программное обеспечение для микроконтроллера, входящего в состав дисплейного модуля системы управления тепловоза, которое обеспечивает выполнение всех поставленных в техническом задании функций.

Разработанная программа планируется к внедрению в продукцию, производимую компанией ООО ПКФ «Полет».

Спасибо за внимание!

					ПГУ 1.090501.03.001 ПЗ	Лист
						111
Изм.	Лист	№ докум.	Подпись	Дата		