|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ПРОЕКТИРОВАНИЕ И ТЕХНОЛОГИЯ ПРОИЗВОДСТВА ЭЛЕКТРОННОЙ АППАРАТУРЫ»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

*К КУРСОВОМУ ПРОЕКТУ*

*НА ТЕМУ:*

«Устройство визуальной индикации температуры в помещении»

Студент группы ИУ4-31М **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кутаев К.С.**

(Подпись, дата)

Руководитель       **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Орлов А.О.**

(Подпись, дата)

2023 г.

АННОТАЦИЯ

Работа посвящена разработке «Устройства визуальной индикации температуры в помещении» в ходе выполнения курсового проекта по дисциплине «Системотехника электронно-вычислительных средств, комплексы и сети». В работе представлено общетехническое обоснование разрабатываемого устройства, содержащее общую информацию об устройстве, анализ технического задания.

Курсовой проект подразделяется на несколько этапов: анализ существующих решений и аналогов, схемотехнический, программный, конструкторский, технологический.

На устройство разработана техническая документация и графические материалы. В заключении представлены выводы о проделанной работе и соответствии разработанного «Устройства визуальной индикации температуры в помещении» техническому заданию.

**ABSTRACT**

The work is devoted to the development of a “Device for visual indication of room temperature” during the course of a course project in the discipline “System engineering of electronic computers, complexes and networks.” The work presents a general technical justification for the device being developed, containing general information about the device and an analysis of the technical specifications.

The course project is divided into several stages: analysis of existing solutions and analogues, circuit design, software, design, and technology.

Technical documentation and graphic materials have been developed for the device. In conclusion, conclusions are presented about the work done and the compliance of the developed “Device for visual indication of room temperature” with the technical specifications.

СОДЕРЖАНИЕ

[АННОТАЦИЯ 2](#_Toc153503419)

[СОДЕРЖАНИЕ 3](#_Toc153503420)

[УСЛОВНЫЕ ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ И ТЕРМИНЫ 7](#_Toc153503421)

[ВВЕДЕНИЕ 9](#_Toc153503422)

[1 АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ И АНАЛОГОВ «УСТРОЙСТВА ВИЗУАЛЬНОЙ ИНДИКАЦИИ ТЕМПЕРАТУРЫ В ПОМЕЩЕНИИ» 14](#_Toc153503423)

[1.1 Обзор существующих систем визуальной индикации температуры в помещении 14](#_Toc153503424)

[1.2 Анализ задания на проектирование «Устройство визуальной индикации температуры в помещении» 18](#_Toc153503425)

[Выводы 20](#_Toc153503426)

[2 СХЕМОТЕХНИЧЕСКИЙ ЭТАП ПРОЕКТИРОВАНИЯ «УСТРОЙСТВА УПРАВЛЕНИЯ УЧЕТОМ ЭЛЕКТРОННЫХ КОМПОНЕНТОВ» 21](#_Toc153503427)

[2.1 Разработка и анализ схемы электрической структурной электронной ячейки «Устройства управления учетом электронных компонентов» 21](#_Toc153503428)

[2.2 Разработка и анализ схемы электрической структурной «Устройства управления учетом электронных компонентов» 22](#_Toc153503429)

[2.3 Разработка и анализ схемы электрической принципиальной «Устройства управления учетом электронных компонентов» 24](#_Toc153503430)

[Выводы 32](#_Toc153503431)

[3 ИССЛЕДОВАТЕЛЬСКИЙ ЭТАП ПРОЕКТИРОВАНИЯ «УСТРОЙСТВА УПРАВЛЕНИЯ УЧЕТОМ ЭЛЕКТРОННЫХ КОМПОНЕНТОВ» 33](#_Toc153503432)

[3.1 Анализ существующих разновидностей систем хранения электронных компонентов 33](#_Toc153503433)

[3.2 Анализ разработанного «Устройства управления учетом электронных компонентов» методами ТРИЗ 40](#_Toc153503434)

[3.3 Анализ протоколов передачи данных для «Устройства управления учетом электронных компонентов» 47](#_Toc153503435)

[Выводы 53](#_Toc153503436)

[4 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРОКОНТРОЛЛЕРА И СЕРВЕРА «УСТРОЙСТВА УПРАВЛЕНИЯ УЧЕТОМ ЭЛЕКТРОННЫХ КОМПОНЕНТОВ» 54](#_Toc153503437)

[4.1 Поставка условий работы алгоритма «Устройства управления учетом электронных компонентов» 54](#_Toc153503438)

[4.2 Реализация алгоритма «Устройства управления учетом электронных компонентов» 58](#_Toc153503439)

[4.3 Разработка алгоритма работы «Устройства управления учетом электронных компонентов» 62](#_Toc153503440)

[Выводы 66](#_Toc153503441)

[5 КОНСТРУКТОРСКИЙ ЭТАП ПРОЕКТИРОВАНИЯ «УСТРОЙСТВА УПРАВЛЕНИЯ УЧЕТОМ ЭЛЕКТРОННЫХ КОМПОНЕНТОВ» 67](#_Toc153503442)

[5.1 Анализ и выбор конструкции и материала печатной платы «Устройства управления учетом электронных компонентов» 67](#_Toc153503443)

[5.2 Анализ и выбор конструкции корпуса «Устройства управления учетом электронных компонентов» 68](#_Toc153503444)

[5.3 Анализ и выбор расположения и установки частей «Устройства управления учетом электронных компонентов» внутри корпуса 69](#_Toc153503445)

[5.4 Выбор электрических соединений в «Устройстве учета электронных компонентов» 71](#_Toc153503446)

[5.5 Расчёт надёжности «Устройства управления учетом электронных компонентов» 73](#_Toc153503447)

[5.6 Тепловой расчёт «Устройства управления учетом электронных компонентов» 80](#_Toc153503448)

[5.7 Расчёт «Устройства управления учетом электронных компонентов» на механические воздействия 88](#_Toc153503449)

[Выводы 93](#_Toc153503450)

[6 ТЕХНОЛОГИЧЕСКИЙ ЭТАП ПРОЕКТИРОВАНИЯ «УСТРОЙСТВА УПРАВЛЕНИЯ УЧЕТОМ ЭЛЕКТРОННЫХ КОМПОНЕНТОВ» 94](#_Toc153503451)

[6.1 Анализ конструкции «Устройства управления учетом электронных компонентов» 94](#_Toc153503452)

[6.2 Анализ корпуса «Устройства управления учетом электронных компонентов» 94](#_Toc153503453)

[6.3 Анализ электронной ячейки «Устройства управления учетом электронных компонентов» 95](#_Toc153503454)

[6.4 Анализ сборочного состава корпуса «Устройства управления учетом электронных компонентов» 96](#_Toc153503455)

[6.5 Анализ сборочного состава ячейки «Устройства управления учетом электронных компонентов» 97](#_Toc153503456)

[6.6 Анализ и оценка технологичности «Устройства управления учетом электронных компонентов» 101](#_Toc153503457)

[6.7 Разработка схемы сборки электронной ячейки «Устройства управления учетом электронных компонентов» 103](#_Toc153503458)

[6.8 Анализ и расчет такта выпуска «Устройства управления учетом электронных компонентов» 105](#_Toc153503459)

[Выводы 106](#_Toc153503460)

[7 ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ «УСТРОЙСТВА УПРАВЛЕНИЯ УЧЕТОМ ЭЛЕКТРОННЫХ КОМПОНЕНТОВ» 107](#_Toc153503461)

[7.1 Разработка схемы электрической структурной измерительного стенда «Устройства управления учетом электронных компонентов» 107](#_Toc153503462)

[7.2 Разработка методики проведения измерений характеристик «Устройства управления учетом электронных компонентов» 108](#_Toc153503463)

[7.3 Результаты экспериментального исследования ШИМ-сигнала на светодиодной ленте 108](#_Toc153503464)

[7.4 Результаты экспериментального исследования SPI-сигнала на RFID-считывателе 111](#_Toc153503465)

[Выводы 115](#_Toc153503466)

[ЗАКЛЮЧЕНИЕ 116](#_Toc153503467)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 119](#_Toc153503468)

[ПРИЛОЖЕНИЕ А 122](#_Toc153503469)

[ПРИЛОЖЕНИЕ Б 128](#_Toc153503470)

# УСЛОВНЫЕ ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ И ТЕРМИНЫ

|  |  |  |
| --- | --- | --- |
| ГОСТ | - | Государственный стандарт, |
| ДПП | - | Двусторонняя печатная плата, |
| ЕСКД | - | Единая система конструкторской документации, |
| ЕСТД | - | Единая система технологической документации, |
| ИП | - | Источник питания, |
| КД | - | Конструкторская документация, |
| КМО | - | Компоненты, монтируемые в отверстия, |
| КМП | - | Компоненты, монтируемые на поверхность, |
| КПД | - | Коэффициент полезного действия, |
| МК | - | Микроконтроллер, |
| МС | - | Микросхема, |
| ПК | - | Персональный компьютер, |
| ПО | - | Программное обеспечение, |
| ПП | - | Печатная плата, |
| РПЗ | - | Расчётно-пояснительная записка, |
| РТЗ | - | Расширенное техническое задание, |
| САПР | - | Система автоматизированного проектирования, |
| СТФ | - | Стеклотекстолит фольгированный, |
| ТЗ | - | Техническое задание, |
| УВИТП | - | «Устройство визуальной индикации температуры в помещении», |
| ШИМ | - | Широтно-импульсная модуляция, |
| ЭА | - | Электронная аппаратура, |
| ЭРЭ | - | Электрорадиоэлемент, |
| ЭС | - | Электронное средство, |
| DMA | - | Direct Memory Access (Прямой Доступ к Памяти), |
| GPIO | - | General Purpose Input-Output (Интерфейс Ввода-Вывода Общего назначения), |
| IoT | - | Internet of Thigs (Интернет Вещей), |
| RFID | - | Radio Frequency Identification (Радиочастотная Идентификация), |
| SPI | - | Serial Peripheral Interface (Последовательный Периферийный Интерфейс), |
| SWD | - | Serial Wire Debug (Последовательный Проводной Отладочный интерфейс), |
| UART | - | Universal Asynchronous Receiver-Transmitter (Универсальный Асинхронный Приёмопередатчик), |
| USB | - | Universal Serial Bus (Универсальная Последовательная Шина) |

# ВВЕДЕНИЕ

Работа посвящена исследованию и разработке цифрового изделия «Устройство визуальной индикации температуры в помещении», позволяющее производить визуальную индикацию температуры в помещении на температурной шкале, а также в цифровом виде на пользовательской веб странице. Устройство предназначено для эксплуатации в домашних и офисных помещениях, где необходимо иметь возможность контролировать температуру как визуально, так и иметь ее значение в цифровом виде для передачи каким-либо системам, способным ее регулировать. Устройство возможно использовать с системами умного дома, где датчиком будет УВИТП, а исполняющим устройством будет является, например регулятор потока горячей воды в радиаторе или же кондиционер.

Объектом исследования является изделие «Устройство визуальной индикации температуры в помещении», предназначенное для визуальной индикации температуры в помещении на температурной шкале, а также в цифровом виде на пользовательской веб странице.

Актуальность работы определяется необходимостью обеспечения рынка доступными устройствами, способными как получать значения температуры в цифровом виде, так и визуально отображать ее на шкале температур. Цифровые устройства измерения температуры в помещении, такие как термометры и термостаты, широко применяются в различных ситуациях, от домашнего использования до коммерческого и промышленного применения. Они предоставляют удобный способ контролировать и регулировать температуру внутри помещения для комфорта и энергоэффективности.

Визуальная индикация температуры поможет пользователям более эффективно контролировать и поддерживать комфортабельные условия внутри помещения, а также точнее настраивать системы отопления или кондиционирования. Это поможет снизить расход энергии, так как будет обеспечиваться оптимальный режим работы систем в соответствии с фактической температурой в помещении. Устройство может служить предупредительным средством, сигнализируя о повышенной или пониженной температуре в помещении. Это позволит оперативно реагировать на отклонения и принимать меры для предотвращения возможных проблем, таких как пожар или обморок от перегрева.

Устройство может быть использовано для сбора данных о температуре внутри помещения на протяжении длительного времени. Это может быть полезно для анализа и оптимизации системы отопления и кондиционирования воздуха, а также для исследований в области энергетики и здоровья. Таким образом, разработка устройства визуальной индикации температуры в помещении имеет практическую значимость и может принести пользу как для конечных пользователей, так и для производителей и исследователей в различных областях.

В итоге рассмотрения данного вопроса можно сказать, что необходимо недорогое решение, при этом способное выполнять задачи визуализации температуры на шкале, а также в цифровом виде. В связи с этим разрабатываемое устройство решено разрабатывать в соответствии с принципами IoT Индустрии 4.0, который представляет из себя концепцию сети передачи данных между физическими объектами, оснащёнными встроенными средствами и технологиями для взаимодействия друг с другом или с внешней средой. Типичными элементами для IoT являются сенсоры, актуаторы и гейты. Сенсоры считывают данные из окружающей среды и передают их на обработку. Актуаторы предназначаются для того, чтобы воздействовать на окружающую среду, или на определённый объект в ней. Гейты – устройства, на которые обычно возлагают логику поверхностного анализа информации, поступающей от подключенных к ним сенсоров. В определённых ситуациях, анализ данных может требовать малого количества вычислительных ресурсов, так что гейты вполне способны принимать некоторые решения самостоятельно. Принимая такие решения, они отправляют определённые команды управления на актуаторы, которые, в свою очередь, выполняют уже свои функции.

Важным принципов IoT является способ коммуникации элементов системы между собой. Дело в том, что идея интернета вещей заключается в создании среды устройств, коммуницирующих между собой без участия человека. Лучшим вариантов для решения данного вопроса является использование беспроводного соединения. В качестве беспроводного соединения было принято решение использовать технологию Wi-Fi, так как она является самой распространенной, не требует покупки отдельных хабов и проста в настройке.  
 Данная технология имеет широкий охват сигнала и высокую пропускную способность, что делает его идеальным для соединения большого количества устройств внутри помещений. Также Wi-Fi является универсальным стандартом беспроводной связи, что делает его широко поддерживаемым и совместимым с большинством устройств в рамках концепции IoT. Она может подключаться как к устройствам с операционными системами, такими как Windows, Linux, iOS и Android, так и к специализированным IoT-устройствам с WiFi-модулями. Стоит ещё отметить, что Wi-Fi обладает высоким уровнем надежности и безопасности, особенно если используются современные протоколы и шифрование данных. Это важно в IoT, где передаются и обрабатываются конфиденциальные информации. Wi-Fi обеспечивает приватность и безопасность передаваемых данных и предотвращает несанкционированный доступ к системам IoT.

**Целью работы** является разработка изделия «Устройство визуальной индикации температуры в помещении» с функциональностью визуальной индикации температуры в помещении на температурной шкале, с возможностью отображения температуры на веб странице и беспроводной передачи данных о температуре по сетевым протоколам, а также разработка комплекта конструкторской и технологической документации.

**Для достижения поставленных целей в процессе работы был решен следующий перечень задач:**

– разработка схемы электрической структурной УВИТП,

– разработка схемы электрической принципиальной электронной ячейки УВИТП,

– разработка электронной ячейки УВИТП,

– проведение расчетов конструкции УВИТП,

– анализ технологического процесса сборки УВИТП.

**Исходными данными** для работы являются:

– задание на выполнение курсового проекта,

– календарный план выполнения курсового проекта.

**Результатами работы** являются:

– схема электрическая структурная УВИТП (Э1),

– схема электрическая принципиальная УВИТП (Э3),

– схема электрических соединений УВИТП (Э4),

– перечень элементов электронной ячейки УВИТП (ПЭ),

– чертеж печатной платы УВИТП (ПП),

– сборочный чертеж электронной ячейки УВИТП (СБ),

– спецификация электронной ячейки УВИТП (СП),

– чертежи основания и крышки корпуса УВИТП,

– сборочный чертёж УВИТП,

– спецификация УВИТП (СП),

– алгоритм работы УВИТП,

– разработанный экспериментальный макет УВИТП,

– расчётно-пояснительная записка.

**Структура и объём работы.** Работа разделена на 5 глав. В первой главе проводится анализ задания и общетехническое обоснование разработки «Устройство визуальной индикации температуры в помещении». Результатом главы является формирование финальных требований к разрабатываемому изделию «Устройство визуальной индикации температуры в помещении» и приведено обоснование целесообразности разработки данного устройства. Во второй главе приведен схемотехнический этап проектирования «Устройство визуальной индикации температуры в помещении». В данной главе осуществляется разработка и анализ схемы электрической структурной устройства и схемы электрической принципиальной. Результатом является готовые структурные и принципиальные схемы. В третьей главе приведена разработка программной части устройства. В ней описывается процесс разработки и отладки внутреннего программного обеспечения устройства, а также разработка алгоритма работы устройства. В результате было получено работающее ПО для МК «Устройство визуальной индикации температуры в помещении», а также алгоритм, описывающий его работу. В четвертой главе осуществляется описание конструкторского проектирования разрабатываемого устройства. В ней приводится описание разработки конструкции, проводится ее расчет на надежность и тепло. Помимо этого, осуществляется анализ и выбор материалов печатной платы устройства, расположение установки частей устройства, а также выбор электрических соединений. В результате для устройства были подобраны необходимые конструкционные материалы и условия. В пятой главе приведены результаты технологического проектирования изделия, проведен анализ конструкции устройства, проведен анализ сборочного состава, оценка технологичности и разработан технологический процесс производства. В результате получен готовый технологический процесс для мелкосерийного производства «Устройство визуальной индикации температуры в помещении».

# 1 АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ И АНАЛОГОВ «УСТРОЙСТВА ВИЗУАЛЬНОЙ ИНДИКАЦИИ ТЕМПЕРАТУРЫ В ПОМЕЩЕНИИ»

## 1.1 Обзор существующих систем визуальной индикации температуры в помещении

Проведем обзор и анализ имеющихся аналогов разрабатываемого устройства. Системы визуальной индикации температуры в помещении имеют важное значение в различных отраслях, таких как промышленность, здравоохранение, розничная торговля, образование и другие. Они помогают обеспечить комфортные условия работы и безопасность людей.

На рынке доступно множество систем визуальной индикации температуры, и каждая из них имеет свои особенности и преимущества. Рассмотрим некоторые из них. ИК-термометры со световыми индикаторами используют инфракрасные технологии для измерения температуры тела. Они имеют интегрированные светодиодные индикаторы, которые меняют цвет в зависимости от измеренной температуры.



Рисунок 1.1 – Инфракрасная камера

Например, при нормальной температуре индикатор будет зеленым, а при повышении температуры он может стать красным. Это позволяет быстро и легко идентифицировать людей с повышенной температурой и принимать соответствующие меры безопасности.

Инфракрасные камеры используют для измерения температуры тела. Они могут сканировать несколько объектов одновременно и выявлять тела с повышенной температурой в режиме реального времени. Когда тело с повышенной температурой обнаруживается, система может предупреждать о возможных проблемах и предлагать меры для предотвращения распространения инфекции или заболевания.

Системы цветовой индикации используют различные цвета или светодиодные индикаторы для обозначения температурных зон.



Рисунок 1.2 – Система цветовой индикации температуры различных зон

Например, зеленый цвет может указывать на комфортную температуру, желтый - на повышенную, а красный - на опасную или недопустимую температуру. Такие системы часто используются в офисах, торговых центрах и других общественных местах для обеспечения комфортных условий и контроля за безопасностью.

Некоторые системы предлагают варианты индикации температуры через мобильные приложения. Пользователи могут устанавливать предпочитаемые границы температуры и получать уведомления при их нарушении. Такие системы обеспечивают удобство и мобильность, позволяя пользователю контролировать и реагировать на изменения температуры в любое время и в любом месте.

Домашние электронные термометры — это маленькие приборы, которые позволяют измерять температуру внутри помещения с помощью электронных датчиков. Они стали популярными инструментами для контроля и регулирования температуры в доме, офисе или других внутренних помещениях.



Рисунок 1.3 – Цифровой домашний термометр

Большинство домашних электронных термометров компактны и легкие, что делает их удобными в использовании. Большинство домашних электронных термометров компактны и легкие, что делает их удобными в использовании. Главным недостатком таких термометров является их ограниченность в совместимости с другими системами. Таким образом нет возможности получить данные о температуре по сети и автоматизировать какие-либо процессы на основе них, например реагирование на слишком высокую или низкую температуру или включение соответствующих приборов, призванных нормализировать уровень температуры в помещении.

Анализ продуктов от вышеперечисленных конкурентов делает экономически целесообразным разработку «Устройство визуальной индикации температуры в помещении» с РТЗ, поскольку разрабатываемое изделие обладает наглядностью, низкой стоимостью, универсальностью и масштабируемостью и в будущем всеми возможностями устройств IoT – концепции в рамках надвигающейся Индустрии 4.0, при соразмерной стоимости в случае серийного выпуска разрабатываемого «Устройство визуальной индикации температуры в помещении».

Четвертая индустриальная революция (Индустрия 4.0) - переход на полностью автоматизированное цифровое производство, управляемое интеллектуальными системами в режиме реального времени в постоянном взаимодействии с внешней средой, выходящее за границы одного предприятия, с перспективой объединения в глобальную промышленную сеть вещей и услуг.

Четвертая промышленная революция (Индустрия 4.0) предполагает новый подход к производству, основанный на массовом внедрении информационных технологий в промышленность, масштабной автоматизации бизнес-процессов и распространении искусственного интеллекта.

Преимущества Четвертой промышленной революции очевидны: повышение производительности, большая безопасность работников за счет сокращения рабочих мест в опасных условиях труда, повышение конкурентоспособности, принципиально новые продукты и многое другое.

Подобно всем предыдущим промышленным революциям, четвертая меняет не только производство, но и всю нашу жизнь — экономику, отношения между людьми, даже в какой-то степени само понимание того, что это значит — быть человеком. Искусственный интеллект и роботизация, интернет вещей (IoT) и 3D-печать, виртуальная и дополненная реальность, био- и нейротехнологии — эти новейшие методы на глазах становятся частью нашего повседневного существования.

## 1.2 Анализ задания на проектирование «Устройство визуальной индикации температуры в помещении»

Проанализируем расширенное техническое задание на курсовой проект подробнее. Основная цель разрабатываемого устройства согласно РТЗ — измерение температуры и ее визуализация на температурной шкале и электронном носителе. Таким образом, необходимо обеспечить точность измеряемой температуры и ее наглядную визуализацию за счет светодиодной ленты. Для реализации измерения и визуализации температуры требуется достаточные вычислительные мощности и объем память — значит, необходимо добавить в устройство вычислительную систему, способное выполнять данные задачи, такой системой является микроконтроллер STM32. Тогда следует добавить в УВИТП функциональность, дающую возможность взаимодействовать с пользователем по беспроводной сети. Такой функционал возможно обеспечить за счет установки в УВИТП беспроводного передатчика. В связи с простотой подключения, распространенности и масштабируемости был выбран интерфейс передачи по технологии Wi-Fi.

С ростом вычислительных возможностей современных программно-управляемых устройств, а также наличие множества различных сенсоров сделало возможным построение сетей обработки информации, обеспечивающих автоматизацию всевозможных процессов. В простейшем случае беспроводная сенсорная сеть представляет собой совокупность взаимосвязанных по беспроводной сети сенсоров, расположенных по контролируемой среде, где сенсор – это модуль, состоящий из считывающего элемента, блока обработки данных, приемопередатчика и элемента питания.

Подобные системы позволяют собирать при помощи считывающего элемента всевозможные данные из окружающей среды и затем соответствующим образом на них реагировать. При этом одна из систем может управлять поведением другой по заранее разработанным алгоритмам. Важным результатом объединения подсистем является синергетический эффект. На данный момент распределенные сети применяются в промышленности, сельском хозяйстве, медицине и других областях.

Функционирование сети заключается в сборе отдельными узлами информации об окружающей среде посредством датчиков, которая затем передается главному модулю по беспроводной сети. Главный модуль агрегирует полученные данные, после чего отображает обработанную информацию оператору, а также может принимать решения на основе заранее определенных алгоритмов. При этом оператору нет необходимости знать о показании всех сенсоров (их количество может доходить до сотен и тысяч единиц). Однако совокупная информация является наиболее важной, чтобы принимать корректные решения для последующих действий. Вследствие этого необходимо использовать алгоритмы, обеспечивающие наиболее надежное и достоверное вычисление.

## Выводы

# 2 СХЕМОТЕХНИЧЕСКИЙ ЭТАП ПРОЕКТИРОВАНИЯ «Устройство визуальной индикации температуры в помещении»

## 2.1 Разработка и анализ схемы электрической структурной электронной ячейки «Устройство визуальной индикации температуры в помещении»

## 2.2 Разработка и анализ схемы электрической структурной «Устройство визуальной индикации температуры в помещении»

## 2.3 Разработка и анализ схемы электрической принципиальной «Устройство визуальной индикации температуры в помещении»

## Выводы

## 3.3 Анализ протоколов передачи данных для «Устройство визуальной индикации температуры в помещении»

В настоящее время технологии обработки данных стремительно развиваются. В связи с увеличением трафика растет нагрузка на сервер, вследствие чего все чаще требуется применение высоко масштабируемых технологии, позволяющей добиться сокращения объема передаваемых данных. Существует множество различных способов повышения эффективности передачи данных, таких как:

- рациональное использование памяти в базе данных;

- разделение данных на шарды;

- правильное применение протоколов передачи данных;

- использование многопоточного асинхронного сервера.

Нельзя сказать, что не было предпринято никаких попыток разгрузить протокол TCP/IP, чтобы уменьшить нагрузку на сервер. Для уменьшения нагрузки впервые была использована технология TCP Offload Engine (TOE).

TOE — это технология, реализованная в некоторых сетевых адаптерах для разгрузки центрального процессора и переноса функций по обработке сетевых пакетов стека протоколов TCP/IP на контроллер сетевого адаптера. Как правило, применяется в высокоскоростных сетевых адаптерах, таких как Gigabit Ethernet и 10 Gigabit Ethernet, когда накладные расходы на обработку сетевых пакетов становятся существенными.

Технологии TOE реализованы в некоторых сетевых картах производства компаний Chelsio и Broadcom; поддерживаются в ОС Windows, FreeBSD. Первоначально TCP разрабатывался для ненадежных низкоскоростных сетей, но с развитием Интернета с точки зрения магистральная скорости передачи (с использованием каналов Optical Carrier, Gigabit Ethernet и 10 Gigabit Ethernet ), а также более быстрые и надежные механизмы доступа (например, DSL и кабельные модемы), он часто используется в центрах обработки данных и настольных ПК средах со скоростью более 1 гигабита в секунду. Реализация программного обеспечения TCP в хост-системах требует значительных вычислительных мощностей. В начале 2000-х годов полнодуплексная гигабитная TCP-связь могла потреблять более 80% процессора 2,4 ГГц Pentium 4, что приводило к небольшой обработке или ее отсутствию.

Вместо полной замены стека TCP существуют альтернативные методы для разгрузки некоторых операций во взаимодействии со стеком TCP операционной системы. Разгрузка контрольной суммы TCP и разгрузка большого сегмента поддерживаются большинством современных сетевых адаптеров Ethernet. Новые методы, такие как большая разгрузка приема и разгрузка подтверждения TCP, уже реализованы в некоторых высокопроизводительных аппаратных средствах Ethernet, но эффективны даже при реализации чисто в программном обеспечении.

На сегодняшний день существует множество протоколов передачи данных. Наиболее подробно остановимся на TCP и UDP.

UDP представляет собой транспортный протокол передачи данных, позволяющий отправлять информацию по сети без предварительного соединения виртуального канала.

Другими словами, UDP является протоколом, ориентированным на дейтаграммы без организации соединения, и предоставляет быстрое, но необязательно надежное обслуживание. UDP поддерживает взаимодействие «один ко многим» и применяется для групповой передачи информации. При отправке данных другому хосту в сети UDP дополняет данные восьмибитным заголовком, содержащим номера, как адресата, так и отправителя, а также общую длину данных и контрольную сумму. Поверх дейтаграммы UDP свой заголовок добавляет и IP, формируя, таким образом, собственную дейтаграмму IP. Контрольная сумма необходима для проверки целостности доставки данных в пункт назначения. В случае, если полученная сумма равна нулю, дейтаграмма отбрасывается. На следующем шаге IP добавляет свои 20 байт заголовка, который включает в себя TTL, а также IP-адрес получателя и источника. Такое действие получило название IP-инкапсуляция.

Существует, и так называемый, максимальный размер пакета, равный 65507 байтам. Если же пакет данных превышает допустимый размер, то, в таком случае, IP разбивает пакет на сегменты (фрагменты), т.е. происходит фрагментация. IP-заголовок содержит полную информацию о фрагментах. Когда отправитель посылает дейтаграмму по сети, она направляется по заданному IP-адресу, указанному в заголовке. А при проходе через маршрутизатор TTL в IP-заголовке уменьшается на единицу. После того, как получателю приходит дейтаграмма по заданному значению и порту, уровень IP позволяет по заголовку установить наличие фрагментации дейтаграммы. Если фрагментация была установлена, то дейтаграмма собирается в соответствии с информацией IP-заголовка, после чего, прикладной уровень извлекает отфильтрованные данные, удаляя при этом заголовок.

TCP является протоколом транспортного уровня, предоставляющий транспортировку потока данных, с необходимостью предварительного соединения, при этом гарантируя целостность получаемых данных.

При отправке данных с использованием TCP, информация перемещается вниз по стеку протоколов.

Данные проходят по всем уровням, после чего, передаются по сети, как поток битов. Каждый уровень в наборе протоколов TCP/IP добавляет к данным свою информацию в форме заголовков. По прибытии пакета на конечный узел в сети, он вновь проходит через все уровни.

Для установления соединения TCP использует процесс «трехфазное квитирование», представленное на рисунке 3.8.

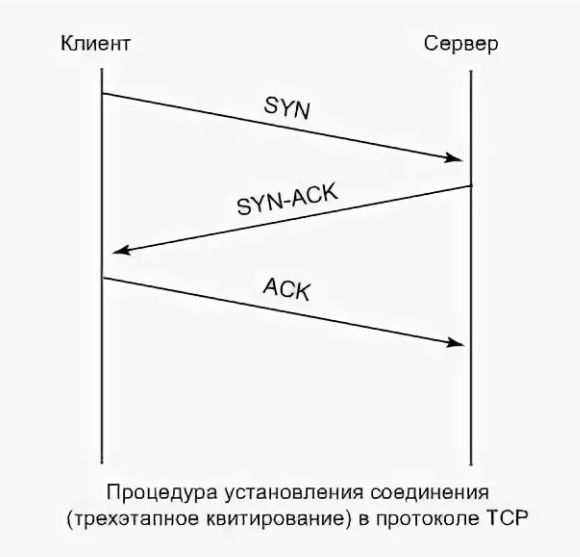


Рисунок 3.8 - Трехэтапное квитирование

Каждый уровень проверяет данные, отделяя от пакета свою информацию в заголовке.

Рассмотрим данный процесс:

1. Клиент взаимодействует с сервером, посылая сегмент с установленным битом SYN (используется для синхронизации порядковых номеров), содержащим начальный порядковый номер клиента.

2. Сервер отвечает отправкой сегмента с установленными битами SYN и ACK, который указывает, что сегмент содержит номер подтверждения. Этот сегмент содержит начальный порядковый номер сервера, не связанный с порядковым номером клиента, и номер подтверждения, на единицу больший порядкового номера клиента, т.е. равный следующему порядковому номеру, ожидаемому от клиента.

3. Клиент подтверждает данный сегмент обратной отправкой сегмента с битом ACK.

Для установления соединения TCP передает данные в виде сегментов для гарантии правильного получения сегментов, имеющих свой порядковый номер. В случае, если подтверждение не получено, то до истечения интервала - тайм-аута, данные отправляются повторно. TCP предоставляет возможность нескольким процессам на одной машине одновременно использовать сокет TCP, а TCP-соединение включает два сокета на разных концах сети.

Таблица 3.2 - Cравнение протоколов TCP и UDP

|  |  |
| --- | --- |
| UDP | TCP |
| Не гарантирует доставку данных | Гарантирует доставку данных |
| Менее защищен | Более защищен |
| Отсутствует управление потоком | Присутствует управление потоком |
| Более быстрый | Менее быстрый |
| Менее устойчивый | Более устойчивый |
| Топологическое разнообразие | Топологическое разнообразие |
| Использует меньше ресурсов системы | Использует больше ресурсов системы |
| Длина заголовка 8 байт | Длина заголовка 20 байт |
| Отсутствие сигналов квитирования | Присутствие сигналов квитирования |

Отсутствие сигналов квитирования - т.е. в UDP у отправителя не имеется возможности узнать, достигла ли дейтаграмма конечного адресата. т. е. UDP в отличии от TCP не гарантирует доставку данных. Протокол TCP же в свою очередь обеспечивает взаимодействие, использую пакеты. При помощи сигналов квитирования проверяется успешность транспортировки данных.

TCP использует идентификатор сеанса, позволяющий отслеживать соединения между двумя хостами, он более защищен, чем UDP. Во многих организациях брандмауэры и маршрутизаторы настроены так, чтобы не пропускать пакеты UDP, в связи с тем, что можно воспользоваться портами UDP, не устанавливая явных соединений.

В UDP отсутствует управление потоком, поэтому UDP-приложение может захватить значительную часть пропускной способности сети. Он отличается своим быстродействием в отличии от TCP. Средства, делающие TCP более устойчивым, замедляют его работу. UDP приобретает форму «топологического разнообразия», поддерживает взаимодействие «один с одним» и «один с многими», в то время как TCP-«один с одним». Он использует меньше ресурсов системы, поэтому при высокой нагрузке на сервера используют именно этот протокол. UDP потребляет меньше пропускной способности сети, так как длина заголовка лишь восемь байт, в то время как TCP использует 20-байтовые заголовки.

Опираясь на вышеприведенные рассуждения, было принято решение воспользоваться протоколом TCP для дальнейшей модификации, так как он наиболее устойчив и гарантирует защиту и передачу данных без каких-либо потерь.

Выводы

# 4 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРОКОНТРОЛЛЕРА «Устройство визуальной индикации температуры в помещении»

## 4.1 Поставка условий работы алгоритма «Устройство визуальной индикации температуры в помещении»

## 4.2 Реализация алгоритма «Устройство визуальной индикации температуры в помещении»

## 4.3 Разработка алгоритма работы «Устройство визуальной индикации температуры в помещении»

## Выводы

# 5 КОНСТРУКТОРСКИЙ ЭТАП ПРОЕКТИРОВАНИЯ «Устройство визуальной индикации температуры в помещении»

## 5.1 Анализ и выбор конструкции и материала печатной платы «Устройство визуальной индикации температуры в помещении»

## 5.2 Анализ и выбор конструкции корпуса «Устройство визуальной индикации температуры в помещении»

## 5.3 Анализ и выбор расположения и установки частей «Устройство визуальной индикации температуры в помещении» внутри корпуса

## 5.4 Выбор электрических соединений в «Устройство визуальной индикации температуры в помещении»

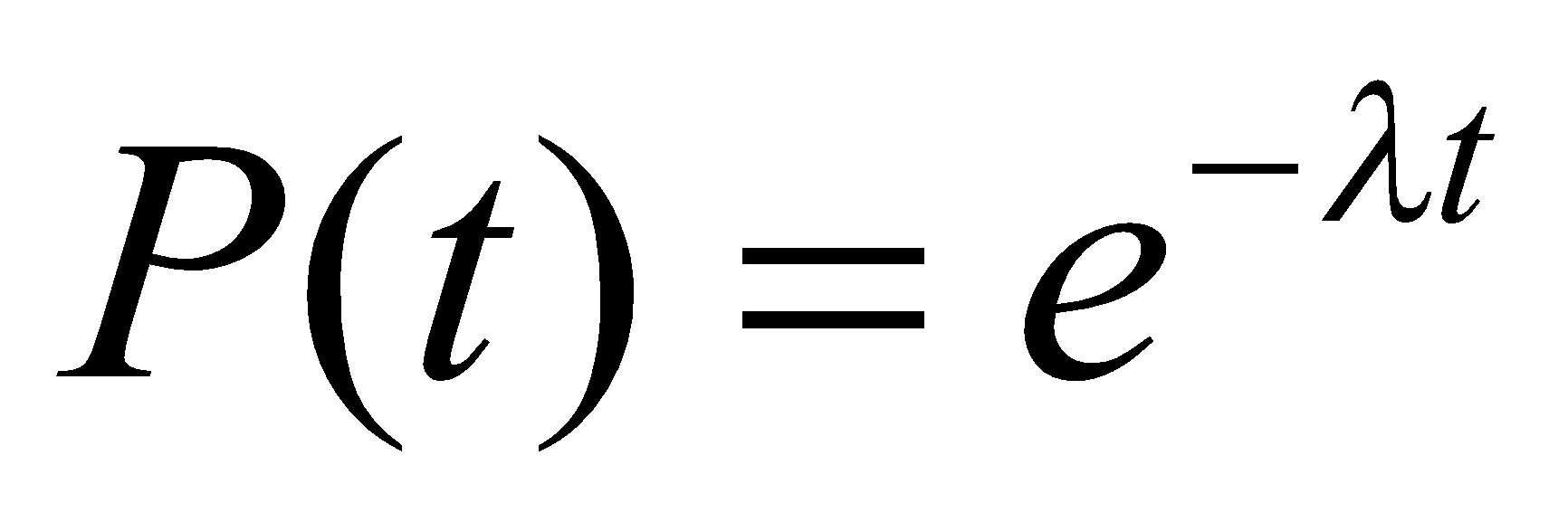
## 5.5 Расчёт надёжности «Устройство визуальной индикации температуры в помещении»

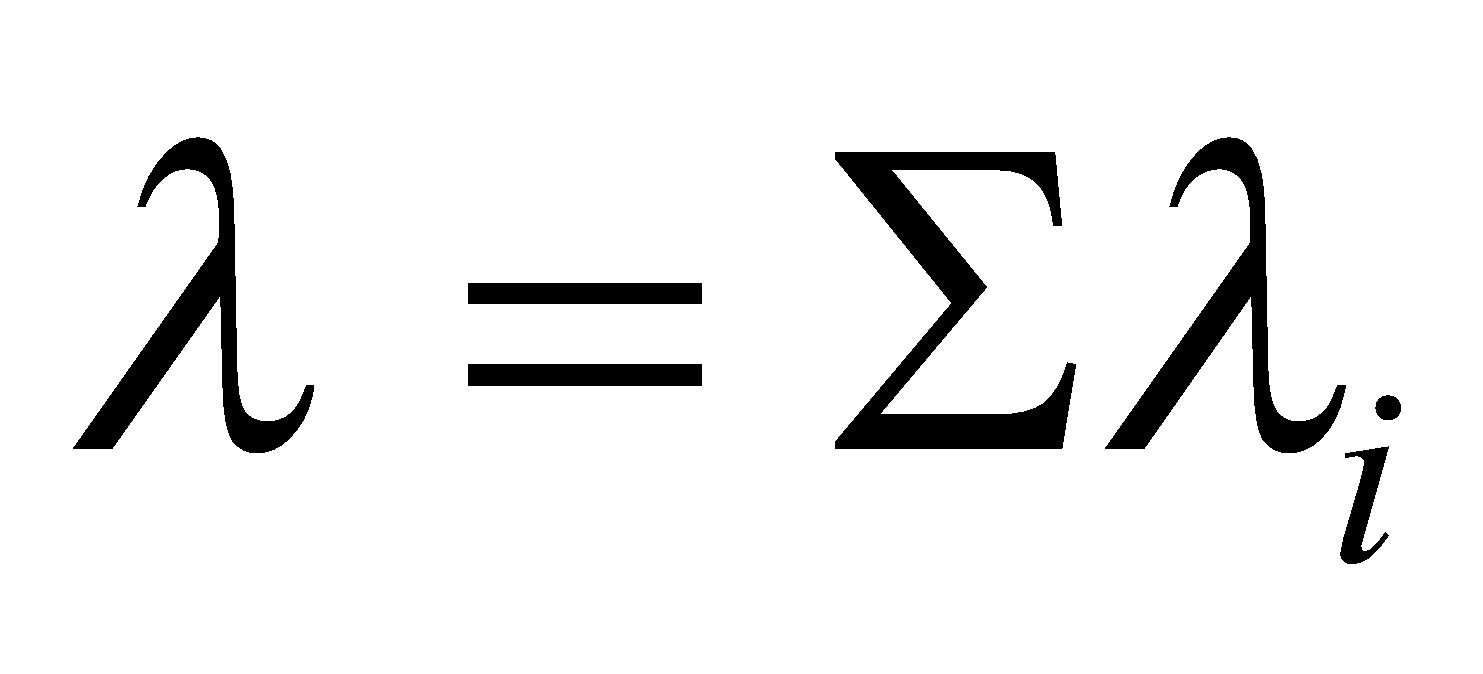
Одной из главных проблем, которую требуется решить в процессе проектирования ППВС, является вопрос надежности, так как предполагается, что эта система будет иметь в своем составе сотни тысяч, миллионы ядер. Под надежностью подразумевается обеспечение отказоустойчивости системы путем создания средств (как программных, так и аппаратных) восстановления работоспособности после сбоя или отказа. В потоковых вычислительных системах из-за сложной структуры параллельных вычислительных процессов возникают трудности с локализацией источника ошибки, ее обнаружения, изоляции отказавшего вычислительного ядра и восстановления нормального функционирования вычислительной системы после сбоя или отказа.

Все перечисленные требования предполагают сложных математических вычислений, которые под силу только мощным микропроцессорам с внутренней кэш-памятью и наличием аппаратных возможностей вычисления математических функций. Требования повышения надёжности и в том числе радиационной стойкости микропроцессоров остро встают и при создании больших вычислительных комплектов. Переход к технологическим нормам 20-28 нм привёл к существенному снижению радиационной стойкости микросхем, сбои микропроцессоров становятся регулярными не только во время солнечных вспышек. Соответственно становится актуальным применение технологий проектирования надежных микросхем для создания высокопроизводительных микропроцессоров.

Заявленная в РТЗ вероятность безотказной работы в течение 5000 ч должна быть не менее 0,95 в соответствии с РД 50-690-89 [19].

Рассматриваемое устройство содержит большое число неремонтируемых элементов, интенсивность отказов которых является постоянной во времени. При этом, вероятность безотказной работы *P* в течение требуемого времени *t* выражается следующей формулой:

 (1)

где — суммарная интенсивность отказов всех модулей изделия.

Расчет надежности заключается в определении показателей надежности изделия по известным характеристикам надежности составляющих элементов и условиям эксплуатации.

1. Индикатор светодиодный

Отказов / 106 час:

|  |  |
| --- | --- |
|  | (2) |

где:

λb – стандартная интенсивность отказов;

πt - температурный фактор;

πQ – фактор качества;

πE – фактор окружающей среды.

Следовательно:

1. Катушка индуктивности

Отказов / 106 час:

|  |  |
| --- | --- |
|  | (3) |

где:

– стандартная интенсивность отказов;

– фактор индуктивности;

πE – фактор окружающей среды;

πQ – фактор качества;

Следовательно:

1. Кварцевый резонатор

Отказов / 106 час:

|  |  |
| --- | --- |
|  | (4) |

где:

– стандартная интенсивность отказов;

πE – фактор окружающей среды;

Следовательно:

1. Конденсатор

Отказов / 106 час:

|  |  |
| --- | --- |
|  | (5) |

где:

– стандартная интенсивность отказов;

– фактор емкости;

πE – фактор окружающей среды;

πQ – фактор качества;

Следовательно:

1. Переключатель

Отказов / 106 час:

|  |  |
| --- | --- |
|  | (6) |

где:

– стандартная интенсивность отказов;

– фактор цикличности;

πL – фактор загрузки нагрузкой;

πC – фактор формы и количества контактов;

πE – фактор окружающей среды;

Следовательно:

1. Микросхема аналоговая

Отказов / 106 час:

|  |  |
| --- | --- |
|  | (7) |

где:

C1 – комплексная интенсивность отказа кристалла;

C2 – интенсивность отказа корпуса;

πt - температурный фактор;

πE – фактор окружающей среды;

πQ – фактор качества;

πL– обучающий фактор.

Следовательно:

1. Микроконтроллер

Отказов / 106 час:

|  |  |
| --- | --- |
|  | (8) |

где:

C1 – комплексная интенсивность отказа кристалла;

C2 – интенсивность отказа корпуса;

πt - температурный фактор;

πE – фактор окружающей среды;

πQ – фактор качества;

πL– обучающий фактор.

Следовательно:

1. Соединительный элемент

Отказов / 106 час:

|  |  |
| --- | --- |
|  | (9) |

где:

λb – стандартная интенсивность отказов;

πK – фактор соединений/разъединений;

πP – фактор активных контактов;

πE – фактор окружающей среды;

Следовательно:

1. Резисторы

Отказов / 106 час:

|  |  |
| --- | --- |
|  | (10) |

где:

λb – стандартная интенсивность отказов;

πR – фактор сопротивления;

πE – фактор окружающей среды;

πQ – фактор качества;

Следовательно:

1. Светодиодный индикатор

Отказов / 106 час:

|  |  |
| --- | --- |
|  | (11) |

где:

λb – стандартная интенсивность отказов;

πt - температурный фактор;

πQ – фактор качества;

πE – фактор окружающей среды.

Следовательно:

1. Транзистор биполярный

Отказов / 106 час:

|  |  |
| --- | --- |
|  | (12) |

где:

λb – стандартная интенсивность отказов;

πT – температурный показатель;

πА – фактор применения;

πR – фактор установленной мощности;

πS –фактор нагрузки по напряжению;

πА – фактор применения;

πQ – фактор качества

Следовательно:

В таблице 5.1 приведены имеющиеся в устройстве компоненты, их интенсивности отказов, а также их количество. Ещё в таблице учтены паяльные соединения, которые имеют особую интенсивность отказов.

Таблица 5.1 – Интенсивности отказов компонентов узлов изделия

|  |  |  |
| --- | --- | --- |
| **Компонент** | **Интенсивность отказа, λi×10-6** | **Количество, шт** |
| Индикатор светодиодный | 0,00798 | 1 |
| Катушка индуктивности | 0,0008 | 1 |
| Кварцевый резонатор | 0,2 | 1 |
| Конденсатор | 0,0016 | 14 |
| Выключатель | 0,0865 | 1 |
| Микросхема аналоговая | 0,004 | 1 |
| Микроконтроллер | 0,043 | 1 |
| Соединительный элемент | 0,0046 | 2 |
| Резистор  постоянный | 0,00258 | 7 |
| Светодиод | 0,000996 | 1 |
| Транзистор | 0,01 | 1 |
| Ручная пайка | 0,15 | 4 |

Интенсивность отказа всего устройства: Λ = 1,002936×10-6 1/ч

Расчёт вероятности безотказной работы в течении времени заданной наработки (5000 ч):

|  |  |
| --- | --- |
|  |  |

Исходя из полученных данных, рассчитанная вероятность безотказной работы устройства в течение заданной наработки (5000 ч.) удовлетворяет условиям расширенного технического задания:

## 5.6 Тепловой расчёт «Устройство визуальной индикации температуры в помещении»

Ярко выраженные тенденции к повышению плотности элементов, увеличению рабочих частот, активному применению смешанных аналого-цифровых элементов РЭА и в то же время предъявляемые к РЭА требования к снижению энергопотребления, веса и габаритных размеров неизбежно приводят к постоянному росту рабочих температур РЭА, а также к совмещению в одной конструкции мощных и теплочувствительных элементов. Многие устройства современной РЭА работают в предельном для них температурном режиме. В такой ситуации тепловое проектирование РЭА без учета взаимного влияния теплового и электрического режимов работы схемы является некорректным. В маршруте проектирования современной аппаратуры тепловое проектирование должно быть не отдельным этапом, проводимым на завершающих этапах разработки, а оно должно проводиться совместно с электрическим и топологическим этапами проектирования. Таким образом, требуются средства электротеплового моделирования, интегрированные в современные САПР РЭА промышленного уровня.

При проектировании РЭА выделяют следующие иерархические уровни компоновки:

1. микросхемы, микросборки и дискретные электрорадиоэлементы, являющиеся элементной базой РЭА;

2. функциональные ячейки, обычно представляющие собой печатные платы, на которых компонуются элементы первого уровня;

3. блоки, объединяющие в одной несущей конструкции пакет функциональных ячеек;

4. многоблочные конструкции, в которых блоки компонуются в общем несущем основании (например, шкафы, стойки, пульты, стеллажи и монтажные рамы).

Целью данного расчета является определение температур нагретой зоны и среды вблизи поверхности электронных компонентов. Расчет проводится для наиболее критичного элемента.

Размеры корпуса:

- Длина L = 0,135 м;

- Ширина B = 0,075 м;

- Высота H = 0,051 м.

Предельная температура среды: t0 = 40 °С (по ГОСТ 15150-69).

Этапы расчёта температуры корпуса:

1. Определяем удельную поверхностную мощность корпуса блока.

Площадь поверхности блока:

|  |  |
| --- | --- |
|  |  |

Мощность, рассеиваемая блоком в виде тепла, P0 = 2.7 Вт.

Удельную поверхностную мощность рассчитаем по формуле:

|  |  |
| --- | --- |
|  |  |

По графику, изображенному на рисунке 5.8 зададимся перегревом корпуса блока в первом приближении Δtk.

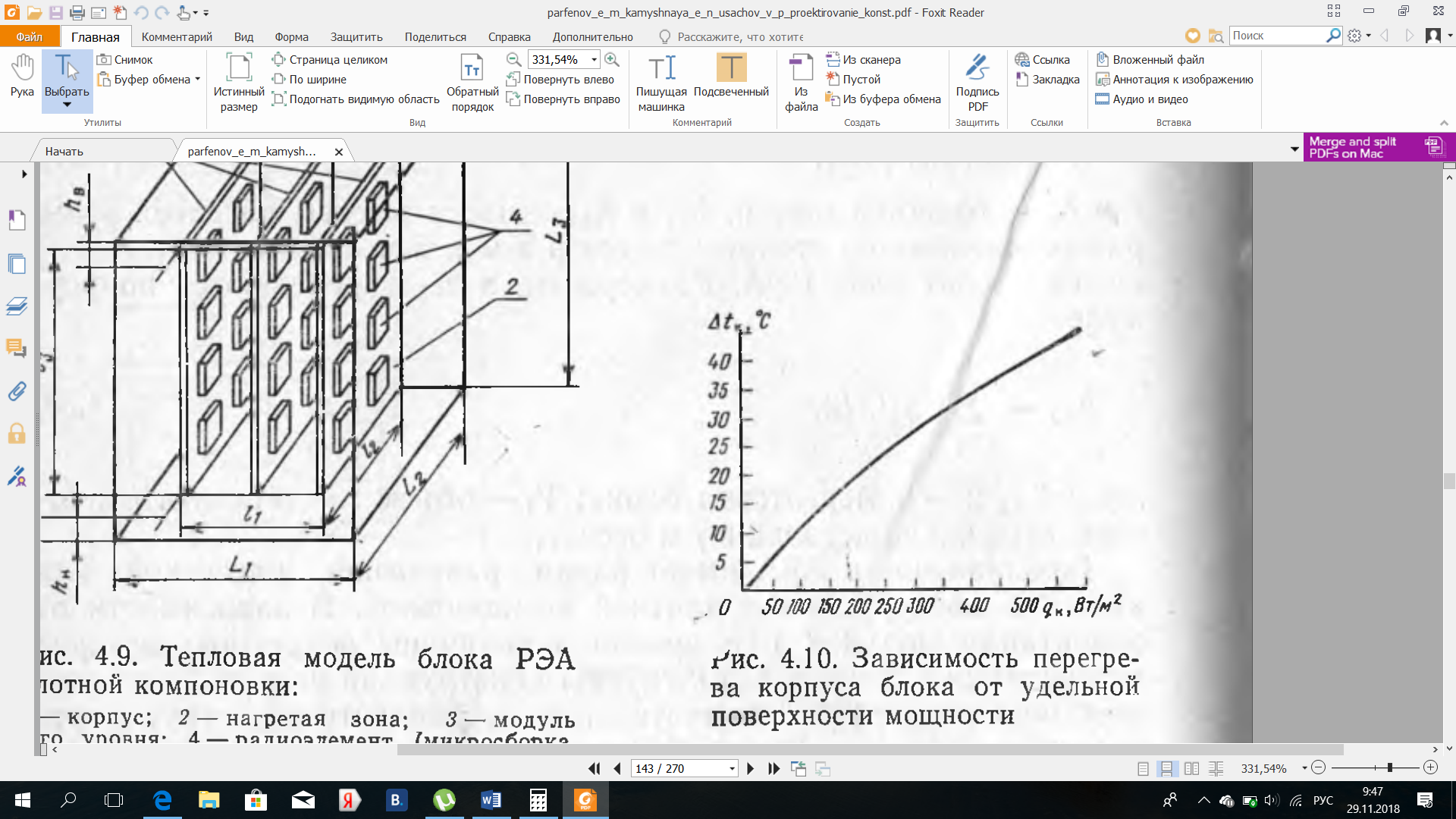


Рисунок 5.8 Зависимость перегрева корпуса блока от удельной поверхности мощности

Определяем по графику и предыдущим расчетам температуру перегрева корпуса и *Δtk = 17 °С.* При дальнейших расчётах значение корректируется до достижения приемлемой ошибки расчёта. Получившееся значение: *Δtk = 16,5 °С.*

Теплообмен между размещёнными в одной конструкции элементами может быть трех видов: кондуктивный, конвективный и радиационый.

Кондуктивный теплообмен происходит между телами, находящимися в непосредственном контакте, за счет колебаний их молекулярных решеток. Этот вид теплообмена один из самых эффективных: в большинстве случаев таким образом передается больше всего тепловой энергии.

Конвективный теплообмен осуществляется между телами, не находящимися в непосредственном контакте; теплообмен осуществляется за счет движения газа (обычно воздуха, в особых случаях жидкости). Причем движение газа может быть или вызвано естественным перемещением нагретых масс газа, или принудительно создано за счет специальных вентиляторов обдува элементов РЭА. Конвективный теплообмен также играет большую роль в процессе рассеивания тепла, выделяемого в элементах РЭА.

Радиационный теплообмен осуществляется также между телами, не находящимися в непосредственном контакте, за счет энергии, излучаемой с поверхности одного тела и поглощаемой поверхностью другого. В большинстве случаев данный вид теплообмена не вносит существенный вклад в общий теплообмен и им можно пренебречь, если излучающее тело не имеет большой площади поверхности близкой по своим свойствам к поверхности абсолютно чёрного тела. Если это условие не выполняется (например, для большинства радиаторов), то пренебречь этим видом теплообмена невозможно без внесения серьёзной ошибки в результат моделирования.

Затем определим коэффициент лучистого теплообмена для верхней, боковой и нижней граней корпуса. Все части корпуса выполнены из пластика, с коэффициентом черноты ε = 0,91. Коэффициент лучеиспускания верхней, нижней и боковых граней корпуса рассчитаем по формуле:

|  |  |
| --- | --- |
|  |  |

Для определяющей критической температуры:

|  |  |
| --- | --- |
|  |  |

Найдем для каждой поверхности блока число Грасгофа:

|  |  |
| --- | --- |
|  | (13) |

где:

βm = (tопр+273)-1 = 3,1×10-3 – коэффициент объемного расширения воздуха (газов);

g = 9,8 м/с2 – ускорение свободного падения;

νm = 17,96×10-6 м2/с – кинематическая вязкость воздуха;

Lопр1 = H = 0,05 м – определяющий размер для боковой поверхности корпуса блока;

Lопр2 = B = 0,022 м – определяющий размер для верхней и нижней поверхностей корпуса блока (выбран наихудший вариант);

Для определяющей температуры *tопр = 48,25 °С* по таблице теплофизических характеристик определяем число Прандтля Pr = 0,699, который является критерием подобия тепловых процессов в жидкостях и газах, учитывающий влияние физических свойств теплоносителя на теплоотдачу.

Рассчитаем конвективный коэффициент теплообмена для каждой поверхности

|  |  |
| --- | --- |
|  | (14) |

λm = 2,83×10-2 [Вт/(м×К)] – теплопроводность воздуха;

- для верхней поверхности корпуса:

где:

NB = 1,3 – коэффициент, учитывающий ориентацию блок (верхняя поверхность).

- для нижней поверхности корпуса:

где:

NB = 0,7 – коэффициент, учитывающий ориентацию блок (нижняя поверхность).

- для боковых, передней и задней поверхностей корпуса:

где:

NB = 1 – коэффициент, учитывающий ориентацию блок (боковая поверхность).

Определим тепловую проводимость между поверхностью корпуса и окружающей средой

|  |  |
| --- | --- |
|  | (15) |

где:

SН, SБ, SВ – площади нижней, боковой и верхней поверхностей корпуса соответственно.

Рассчитаем температуру корпуса устройства:

1-й этап расчета теплового режиме устройства окончен.

Проведем вычисления условной удельной поверхностной мощности нагретой зоны блока:

|  |  |
| --- | --- |
|  | (16) |

где:

P3 – мощность рассеиваемая в нагретой зоне.

l1 = 0,062 м – длина нагретой зоны;

l2 = 0,04 м – ширина нагретой зоны;

l3 = 0,012 м – высота нагретой зоны.

где: Pк – мощность тепловыделяющих элементов, расположенных на корпусе блока. Её значением можно пренебречь.

По графику, представленному на рисунке 5.9, найдем в первом приближении перегрев нагретой зоны относительно температуры – *Δt3 = 23,4 °С*. Итоговое значение *Δt3 = 23,75 °С.*

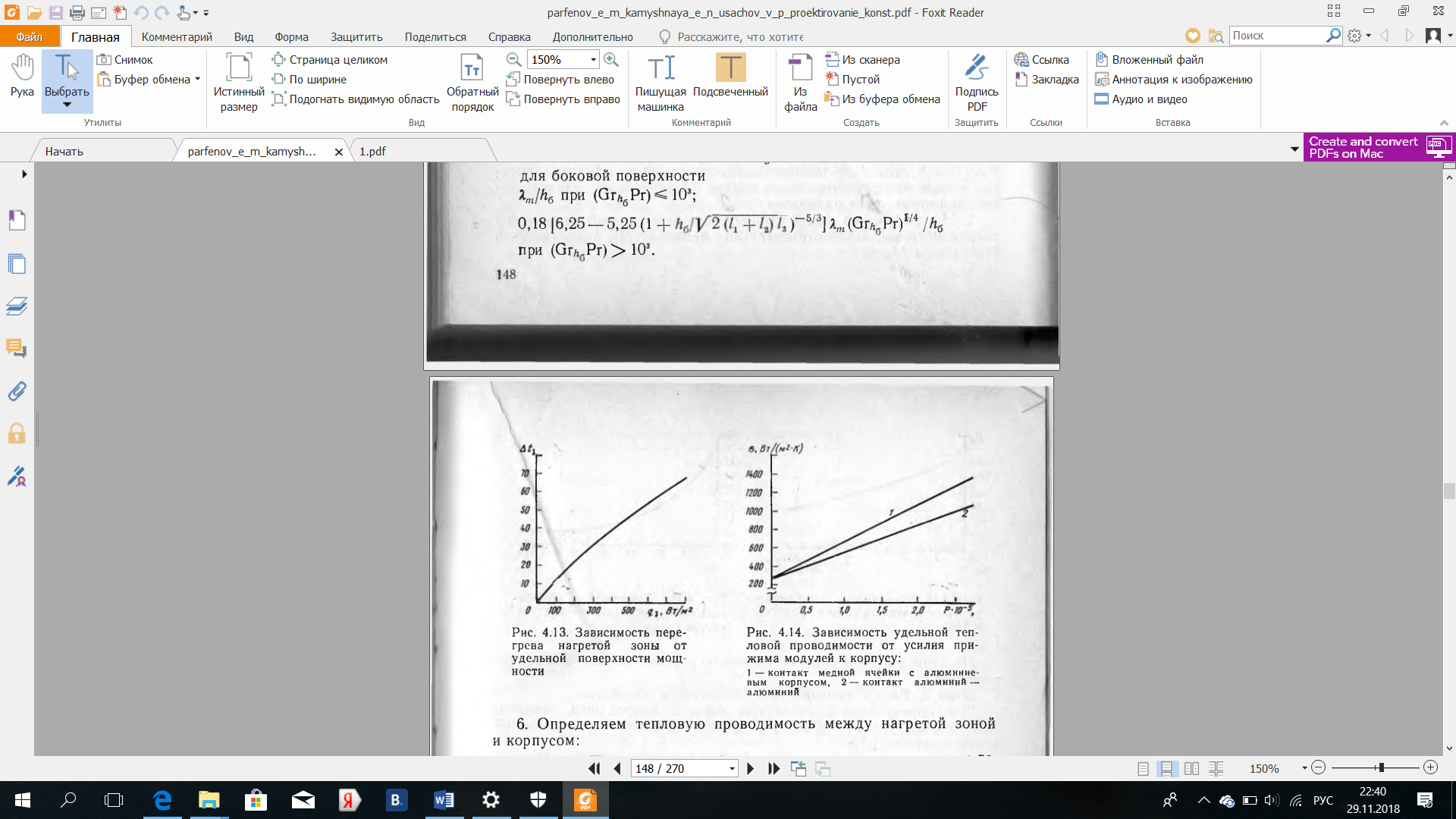


Рисунок 5.9 - Зависимость перегрева нагретой зоны от удельной поверхностной мощности

Найдем коэффициент теплообмена между нижними, верхними и боковыми поверхностями нагретой зоны:

|  |  |
| --- | --- |
|  | (17) |

где:

εп.i – приведенная степень черноты i-поверхности нагретой зоны.

|  |  |
| --- | --- |
|  | (18) |

εк.i = 0,91 – для пластика.

Площадь нагретой зоны боковых поверхностей корпуса

Площадь нагретой зоны верхней и нижней поверхностей корпуса

- для нижней и верхней поверхности нагретой зоны:

- для боковых, передней и задней поверхностей нагретой зоны:

Рассчитаем коэффициенты конвективного теплообмена между нагретой зоной и корпусом для каждой поверхности:

- для нижней поверхности

- для верхней поверхности

- для боковой поверхности:

Рассчитаем нагрев нагретой зоны Δtз.о. во втором приближении:

|  |  |
| --- | --- |
|  | (19) |

Определим ошибку расчета:

Так как величина ошибки δ=0,014 ≤ 0,1, то расчет можно считать законченным.

Рассчитаем температуру нагретой зоны:

На этом 2-й этап расчета теплового режима устройства окончен.

Определим температуру корпуса МК устройства.

Определим эквивалентный коэффициент теплопроводности устройства, в котором расположен МК. При отсутствии теплопроводных шин λэкв = λп, где λп – теплопроводность материала основания платы (стеклотекстолит):

Определим эквивалентный радиус корпуса микросхемы:

где:

S0 ИС =0,0093 м2 – площадь основания микросхемы.

Рассчитаем коэффициент распространения теплового потока:

|  |  |
| --- | --- |
|  |  |

где:

α1+ α2 = 17 Вт/(м2×К) – коэффициенты теплообмена с 1-й и 2-й сторон ПП;

δП = 1,5 мм – толщина ПП модуля;

λэкв = 0,24…0,34 – справочная величина.

Подставляя значения в формулу, получаем ΔtИС = 6,67

Температура поверхности корпуса ИМС:

tИС = ΔtИС + t3 = 86,64

## 5.7 Расчёт «Устройство визуальной индикации температуры в помещении» на механические воздействия

Целью расчета является определение перегрузок при действии вибрации и ударов, действующих на элементы устройства.

При изготовлении контрольно-измерительных чувствительных элементов повышаются требования к качеству кремния по всей толщине пластины. В результате удаления дефектного слоя на поверхности образуются микродефекты, которые распространятся вглубь объёма материала. Причём поверхностные атомы пластин, по сравнению с объёмными, имеют большее количество ненасыщенных химических связей, чем объясняется высокая адсорбционная способность, приводящая к загрязнениям рабочей поверхности.

В зависимости от вида взаимодействия загрязнений с поверхностью различают физическую и химическую адсорбцию. Физическая адсорбция – результат межмолекулярного электростатического кулоновского взаимодействия. Частицы, совершая тепловые движения, обладают достаточным запасом энергии, чтобы преодолеть силы связи и перемещаться, удаляясь с поверхности, или, наоборот, по механическим микродефектам попадают вглубь материала.

К физически адсорбированным загрязнениям относятся все виды механических частиц, а также все виды органических материалов, связанные с поверхностью силами физической адсорбции. К химически адсорбированным загрязнениям относятся различные виды оксидных и сульфидных пленок на поверхности пластин, катионы и анионы химических веществ, применяемых в процессе очистки поверхности пластины. Примеси или локальные нарушения кристаллической решётки образуют в материале неустранимые очаги механических напряжений, так называемых дислокаций. Дислокации оказывают существенное влияние на электрофизические свойства монокристаллов кремния и работу приборов, изготавливаемых на его основе.

Проведем расчёты на вибрацию. Исходные данные:

a = 0,116 м – длина ПП;

b = 0,056 м – ширина ПП;

H = 0,0015 м – толщина ПП;

М = 0,0174 кг – масса ПП с установленными электронными компонентами;

ν = 0,22 – коэффициент Пуассона;

E = 3,02×1010 Н/м2

Определим частоту собственных колебаний

Цилиндрическая жесткость:

|  |  |
| --- | --- |
|  |  |

Частоту собственных колебаний равномерно нагруженной пластины, закрепленной в 4-х точках, которую представляет из себя ПП, вычислим по формуле:

|  |  |
| --- | --- |
|  |  |

Определим коэффициент динамичности

|  |  |
| --- | --- |
|  | (20) |

где:

ε – коэффициент затухания;

Коэффициент динамичности рассчитываем во всем диапазоне вибрации. Согласно РТЗ, частота возбуждения находится в диапазоне от 10 до 100 Гц. Результаты расчета представлены в таблице 5.2.

Таблица 5.2 – Результаты расчет коэффициента динамичности

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **f, Гц** | **10** | **30** | **60** | **80** | **100** |
| **η** | 0,0085 | 0,0255 | 0,0509 | 0,0679 | 0,0848 |
| **Кдин** | 1,0001 | 1,0006 | 1,0026 | 1,0046 | 1,0073 |

Определим коэффициент передачи по ускорению на первой гармонике в центре платы.

Относительные координаты центра платы:

Согласно графику, представленному на рисунке 5.10, коэффициенты формы колебаний при защемлении обоих краёв платы (кривая 2): Кх = Ку = 1,2.

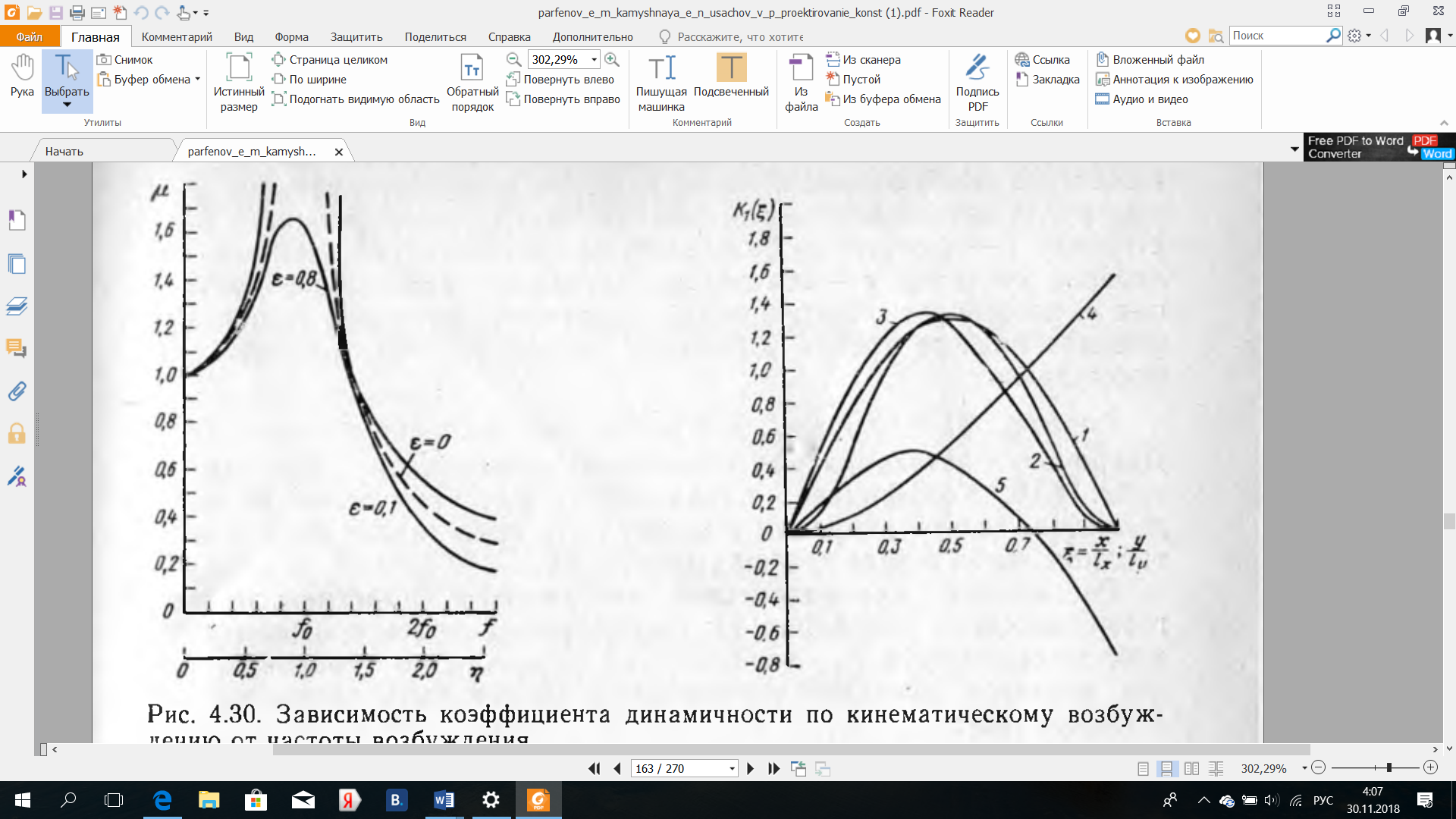


Рисунок 5.10 Зависимость коэффициента формы колебаний от относительной координаты

Коэффициент передачи:

|  |  |
| --- | --- |
|  |  |

Определим виброускорение для пластины в случае кинематического возбуждения:

Определим амплитуду виброперемещения основания:

|  |  |
| --- | --- |
|  |  |

Определим виброперемещение:

Согласно конструкторским расчетам, максимальное значение виброускорения составляет:

Исходя из этих данных, следует, что условие вибропрочности выполняется.

Для ПП с установленными на ней электронными компонентами максимально допустимый прогиб платы составляет *δВ<0,003b,*

где b – размер стороны печатной платы.

Максимальный прогиб ПП меньше допустимого, следовательно, условие вибропрочности выполняется.

Виброускорение и максимальное относительное перемещение меньше допустимого для ячейки, и таким образом, удовлетворяются требованиями ТЗ на вибропрочность ячейки. Дополнительных конструкционных мер по повышению жесткости ПП на данном этапе расчета не требуется.

## Выводы

# 6 ТЕХНОЛОГИЧЕСКИЙ ЭТАП ПРОЕКТИРОВАНИЯ «Устройство визуальной индикации температуры в помещении»

## 6.1 Анализ конструкции «Устройство визуальной индикации температуры в помещении»

## 6.2 Анализ корпуса «Устройство визуальной индикации температуры в помещении»

## 6.3 Анализ электронной ячейки «Устройство визуальной индикации температуры в помещении»

## 6.4 Анализ сборочного состава корпуса «Устройство визуальной индикации температуры в помещении»

## 6.5 Анализ сборочного состава ячейки «Устройство визуальной индикации температуры в помещении»

Конструкция электронной ячейки включает два типа компонентов — КМО и КМП. Монтаж двусторонний. КМП компоненты группируются в зависимости от стороны установки. КМО компоненты устанавливаются и паяются вручную паяльником. Установка и формовка ИЭТ осуществляется согласно ГОСТ 29137-91. Обозначения электронных компонентов согласно спецификации, количество компонентов на плате, варианты установки компонентов для единичного производства представлены в таблице 6.2.

Таблица 6.2 – Варианты установки ИЭТ на ПП «Устройство визуальной индикации температуры в помещении»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование** | **Вариант установки** | **Характеристика варианта установки** | **Форма, и число**  **выводов** | **Примечания** | | |
| **1** | **2** | **3** | **4** | **5** | **6** | | |
| 1 | Микросхема DD1 |  | Установка с зазором, фиксация на паяльную пасту | Планарные выводы – 64 шт. |  | | |
| 2 | Резисторы R1-R14, Конденсаторы С1-С74, Предохранитель F1, Светодиод HL1, Типоразмерные SMD 0803 |  | Установка без зазора, фиксация на паяльную пасту | Плоские выводы с торцов корпуса – 2 шт. |  | | |
| 3 | Катушка индуктивности L1 |  | Установка без зазора, фиксация на паяльную пасту | Плоские выводы с торцов корпуса – 2 шт. | |  |

Продолжение таблицы 6.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | Микросхема DA1 |  | Установка без зазора, фиксация на паяльную пасту | Плоские выводы с торцов корпуса – 2 шт. |  |
| 5 | Переключатель SA1 |  | Установка без зазора, фиксация на паяльную пасту | Плоские планарные выводы – 6 шт |  |
| 6 | Кнопка тактовая SB1 |  | Установка без зазора, фиксация на паяльную пасту | Плоские планарные выводы – 4 шт. |  |
| 7 | Кварцевый резонатор BQ1 |  | Установка с зазором, фиксация на паяльную пасту | Плоские планарные выводы на днище корпуса – 4 шт. |  |
| 8 | Биполярный транзистор VT1 |  | Установка с зазором, фиксация на паяльную пасту | Планарные выводы – 3 шт. |  |
| 9 | Разъемы штыревые XP1-XP2, XS2-XS3 |  | Установка без зазора, пайка паяльником | Штыревые выводы – 5 шт |  |
| 11 | Micro USB коннектор XS1 |  | Установка без зазора, фиксация на паяльную пасту | Плоские планарные выводы 11 шт. |  |

Пайка выводов КМО компонентов производится припоем ПОС-61 ГОСТ 21930-76 согласно ГОСТ 23592-96. После сборки изделие промывают в ультразвуковой очистительной ванне со специально очистительной смесью ГОСТ 18300-87.

Контроль паяных соединений производится визуально-оптическим методом по ГОСТ 24715-81. В случае серийного производства рекомендуется производить монтаж КМО за счет пайки волной.

## 6.6 Анализ и оценка технологичности «Устройство визуальной индикации температуры в помещении»

Коэффициенты для расчета и анализа технологичности устройства для мелкосерийного производства представлены в таблице 6.3.

Таблица 6.3 – Коэффициенты для расчета и анализа технологичности СУУ

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Обозначение** | **Значение** |
| Количество ИМС | HИМС | 2 |
| Количество других ИЭТ | HИЭТ | 40 |
| Количество механизированных соединений | Hмк | 52 |
| Общее количество соединений | Hм | 52 |
| Количество элементов, подготавливаемых механизированным путем | Hмп | 0 |
| Количество операций механизированного контроля | Hмкн. | 0 |
| Общее количество операций контроля | Hкн | 1 |
| Количество типов номиналов ИЭТ | HтИЭТ | 18 |
| Количество типов номиналов оригинальных ИЭТ | HорИЭТ | 0 |

Базовые показатели технологичности «Устройство визуальной индикации температуры в помещении» представлены в таблице 6.4.

Таблица 6.4 – Базовые показатели технологичности «Устройство визуальной индикации температуры в помещении»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Наименование показателя** | **Формула** | **Значение** | **Коэффициент значимости** ϕ***i*** |
| 1 | Коэффициент использования микросхем |  | 0,04 | 1 |
| 2 | Коэффициент механизации и автоматизации  монтажа |  | 1 | 1 |
| 3 | Коэффициент механизации подготовки к  монтажу |  | 0 | 0,8 |
| 4 | Коэффициент механизации контроля и  настройки |  | 0 | 0,5 |
| 5 | Коэффициент  повторяемости ИЭТ |  | 0,5 | 0,3 |
| 6 | Коэффициент применяемости  ИЭТ |  | 1 | 0,2 |

Рассчитаем показатель технологичности изделия:

.

Полученное значение комплексного показателя технологичности устройства соответствует показателю для мелкосерийного производства.

## 6.7 Разработка схемы сборки электронной ячейки «Устройство визуальной индикации температуры в помещении»

## 6.8 Анализ и расчет такта выпуска «Устройство визуальной индикации температуры в помещении»

Анализ объема выпуска устройства проводится с целью определения возможности выпуска изделий по данному ТП в заданном объеме в установленные сроки путем сравнения штучного времени сборки изделия с заданным тактом выпуска.

Заданный объём выпуска *N*вып= 1000 шт./год. По заданному объёму выпуска определяется такт выпуска:

 (21)

где  – такт выпуска, Ф – годовой запас рабочего времени (Ф ≈ 2070 ч);

*N*вып – программа выпуска. Тогда, по формуле (21):

 мин/шт.

Следовательно, производительность:

шт/ч.

## Выводы

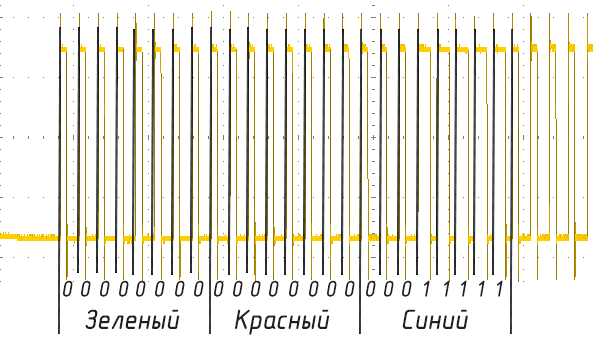
# 7 ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ «Устройство визуальной индикации температуры в помещении»

## 7.1 Разработка схемы электрической структурной измерительного стенда «Устройство визуальной индикации температуры в помещении»

## 7.2 Разработка методики проведения измерений характеристик «Устройство визуальной индикации температуры в помещении»

## 7.3 Результаты экспериментального исследования ШИМ-сигнала на светодиодной ленте

Одна из проверок работоспособности изделия заключалась в измерении напряжений на линиях ШИМ – сигнала светодиодной ленты во временной зависимости при помощи осциллографа. Сравнение теоретически рассчитанной модели измерений линии ШИМ – сигнала показано на рисунке 7.2.



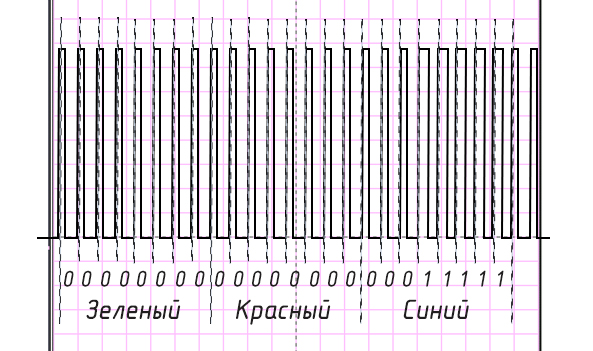


Рисунок 7.2 – Смоделированный ШИМ – сигнал, идущий от МК к светодиодной ленте

При проведении сравнения полученных значений обнаружилось, что предполагаемые теоретически модели поведения линий электрических связей имеют большое количество совпадений поведения, что говорит об удовлетворительном уровне выполнения поставленной задачи устройством. Небольшие помехи на экспериментальном изображении вызваны погрешностями в номиналах элементов и неточностью измерительной аппаратуры.

На графике можно наблюдать пакет данных, который кодирует цвет определенного светодиода. Он состоит из 24 бит, по 8 бит на каждый цвет в порядке зеленый, красный и синий [22]. Максимальная концентрация каждого цвета в результирующем цвете светодиода определяется величиной числа для каждого цвета. Максимальное значение 255. В данном пакете закодирован синий цвет, так как биты зеленого и красного равны нулю, и биты синего представляют собой число 31.

Количество тока (ампер), которое потребляет каждый светодиод, зависит от их цвета и яркости. Если горит только один из трех красных, зеленых или синих светодиодов, он будет потреблять меньше энергии. Если для пикселя установлен белый цвет (все три светодиода горят), он будет потреблять максимальное количество энергии.

Предполагается, что каждая из ламп RGB будет потреблять максимум 20 мА. Это дает нам 20 мА + 20 мА + 20 мА = 60 мА на пиксель. Поэтому для всей светодиодной ленты из 150 пикселей мы будем использовать максимальную мощность 9 А.

Установка для светодиодов максимальной яркости 255 может сделать их совершенно трудными для просмотра без какой-либо диффузии. Кроме того, когда мы думаем о требованиях к питанию, мы должны рассматривать только наихудший сценарий. Это означает, что, если мы готовы пожертвовать некоторой яркостью, мы можем безопасно подключить больше светодиодов на ампер тока.

Вместо того, чтобы просто принять 60 мА на пиксель по умолчанию, мы можем более точно согласовать наши светодиоды и источник питания. Например, если у нас есть источник питания на 5 ампер, и мы хотим подключить 150 светодиодов, это дает нам бюджет мощности 33,3 мА на пиксель (5000 мА / 150).

Важно использовать качественный источник питания на 5 вольт. Многие силовые блоки, особенно те, которые более чем на 5 ампер, используют более дешевые компоненты и фактически не могут поддерживать такой уровень тока.. Попытка получить большое количество тока может привести к падению напряжения, снижению яркости светодиодов или, что еще хуже, к повреждению их или блока питания.

В каждую полосу встроены маленькие провода для подключения одного светодиода к другому. Если мы попытаемся установить для всех светодиодов максимальную белую яркость, то будет потребляться слишком много тока. К тому моменту, как ток доходит до конца, напряжение просело, а яркость и цветопередача светодиодов ухудшились. Чтобы исправить это, нам просто нужно подать питание и землю на обе стороны полосы. Многие полосы даже имеют дополнительную пару для проводов 5V и GND на каждом конце специально для «впрыскивания» большей мощности.

## Выводы

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Карпунин А.А., Власов А.И. Обработка данных с распределенным реестром в концепции "Индустрия 4.0" // Сборник «Энергосбережение и эффективность в технических системах». Материалы V Международной научно-технической конференции студентов, молодых учёных и специалистов. 2018. С. 120-121.

2. Автоматизация хранения SMD компонентов. Электронный ресурс. Адрес доступа: <https://www.iss-group.ru/electronic-components>. Дата обращения 02.05.2022.

3. Шахнов В.А., Резчикова Е.В. ТРИЗ в техническом университете // Saarbrucken, 2015.

4. Обзор 98 одноплатных компьютеров. Электронный ресурс. Адрес доступа: https://habr.com/ru/post/405023/. Дата обращения 23.04.2022.

5. Автоматизированная система хранения и учета электронных компонентов. Электронный ресурс. Адрес доступа: https://usamodelkina.ru/s/19920-avtomatizirovannaja-sistema-hranenija-i-ucheta-jelektronnyh-komponentov.html. Дата обращения 05.05.2022.

6. Официальный сайт компании Odoo. Электронный ресурс. Адрес доступа: https://www.odoo.com/ru\_RU. Дата обращения 14.03.2022.

7. ГОСТ 2.701-84 «Схемы. Виды и типы. Общие требования к выполнению».

8. ГОСТ 2.702-2011 «Правила выполнения электрических схем».

9. Микроконтроллер STM32F103RDT6. Официальный сайт «STMicroelectronics». Электронный ресурс. Адрес доступа: https://www.st.com/en/microcontrollers-microprocessors/stm32f103rd.html. Дата обращения 24.05.2022.

10. Документация на транзистор BC182LB. Электронный ресурс. Адрес доступа: https://static.chipdip.ru/lib/318/DOC000318843.pdf. Дата обращения 18.05.2022.

11. Кутаев К.С., Рыжов Ф.С. Визуализированная цифровая система хранения и автоматизированной выдачи электронных компонентов на базе собственного сервера с удаленным доступом // KPO-Science. 2021.

12. Кутаев К.С., Латыпова В.А., Кольцова В.С., Кольцов И.Р. Обеспечение безопасности клиент-серверных приложений от инъекций // KPO-Science. 2022.

13. ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения».

14. Кутаев К.С., Латыпова В.А. Модернизация распределения нагрузки на клиент-серверной архитектуре поверх сетевого протокола передачи данных // KPO-Science. 2022.

15. ГОСТ Р 53429-2009 «Платы печатные. Основные параметры конструкции».

16. ГОСТ 10316-78 «Гетинакс и стеклотекстолит фольгированные. Технические условия».

17. ГОСТ 33366.1-2015 «Пластмассы. Условные обозначения и сокращения. Основные полимеры и их специальные характеристики».

18. ГОСТ 21931-76 «Припои оловянно-свинцовые в изделиях. Технические условия».

19. РД 50-690-89 «Методы оценки показателей надежности»

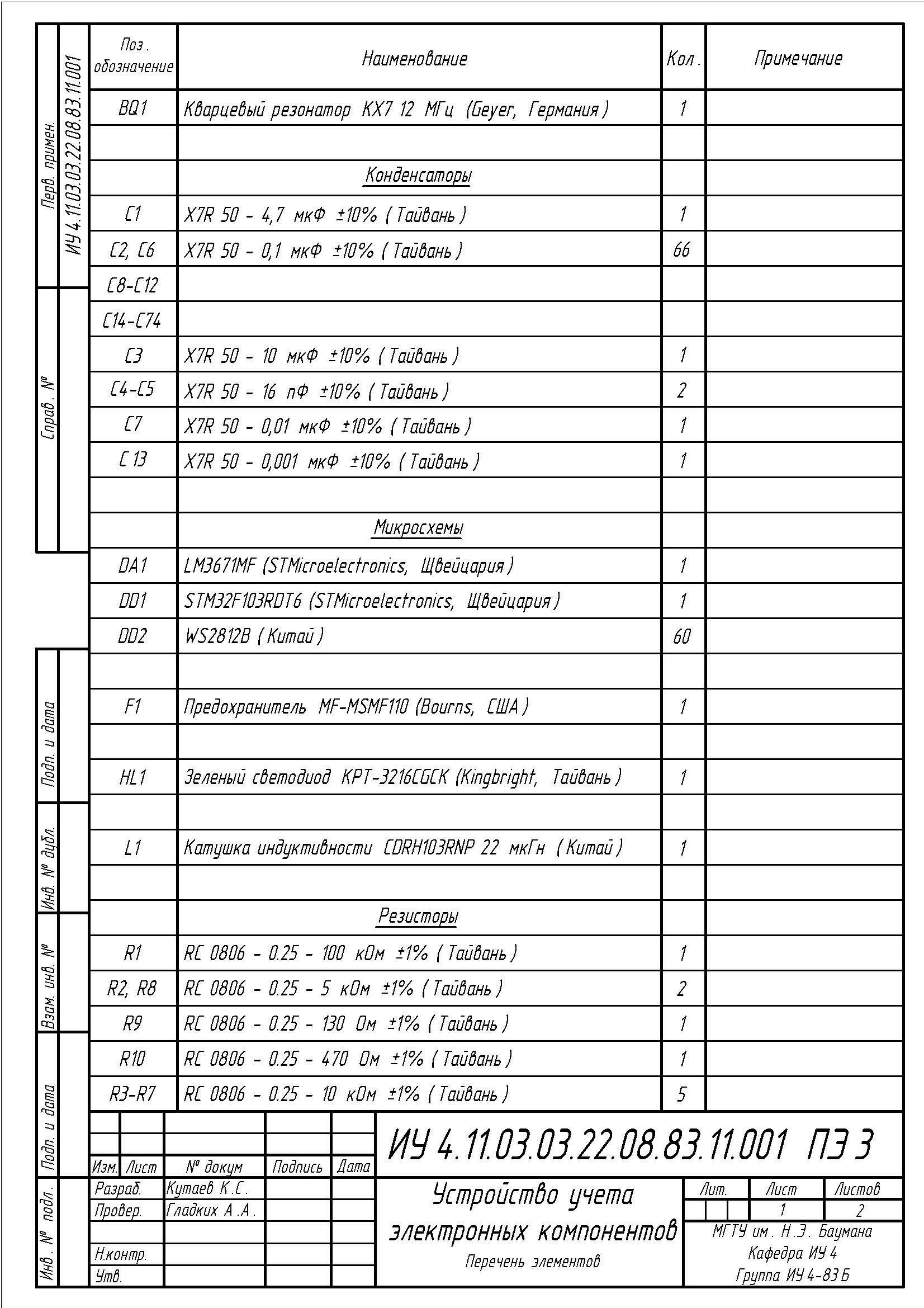
20. Конструкторско-технологические проектирование электронной аппаратуры: Учебник для вузов / К.И. Билибин, А.И, Власов, Л.В. Журавлева и др. Под общ.ред. В.А. Шахнова – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002.

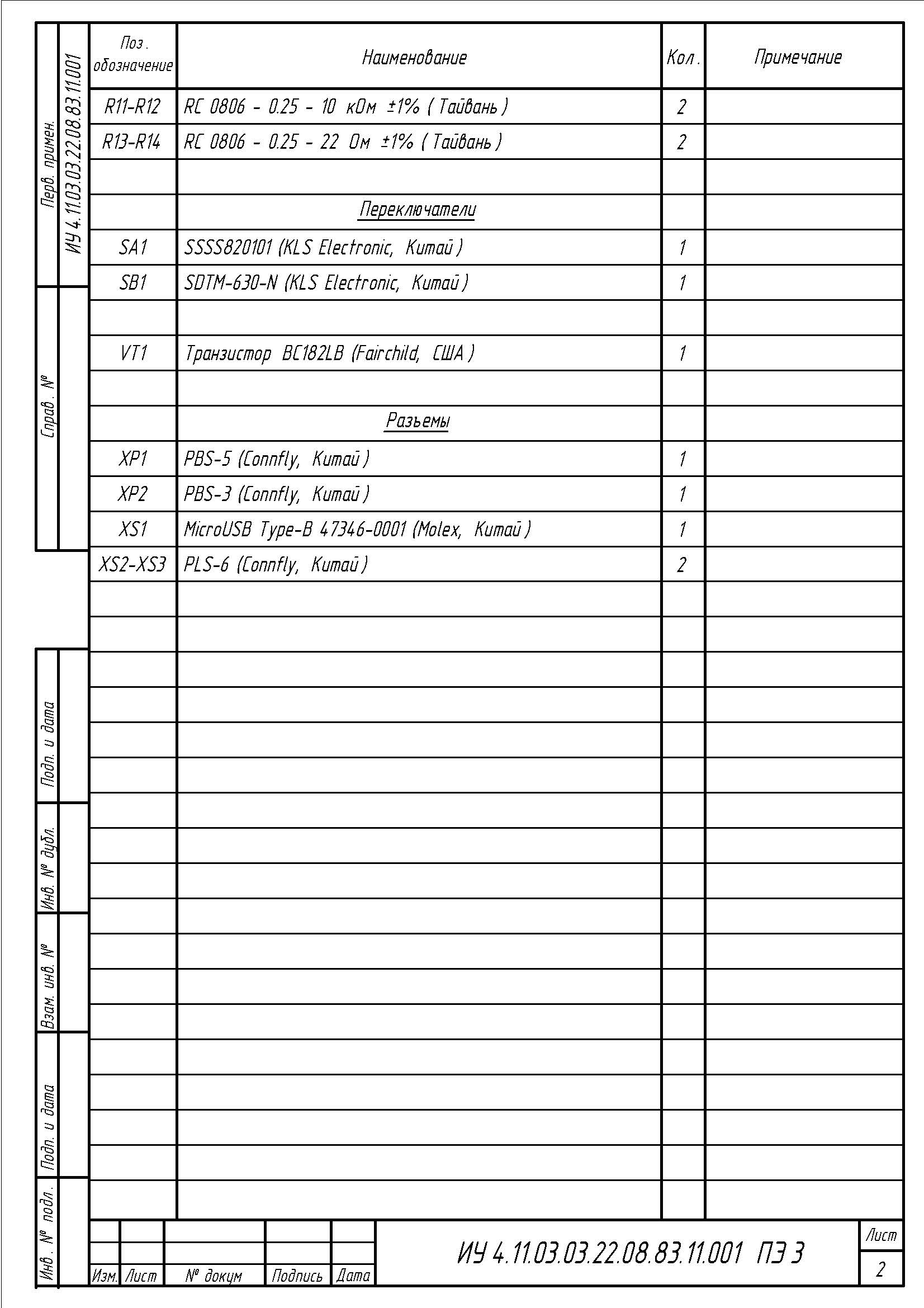
21. ГОСТ Р ИСО 7045-2013 «Винты со скругленной головкой и крестообразным шлицем типа Н или типа Z»

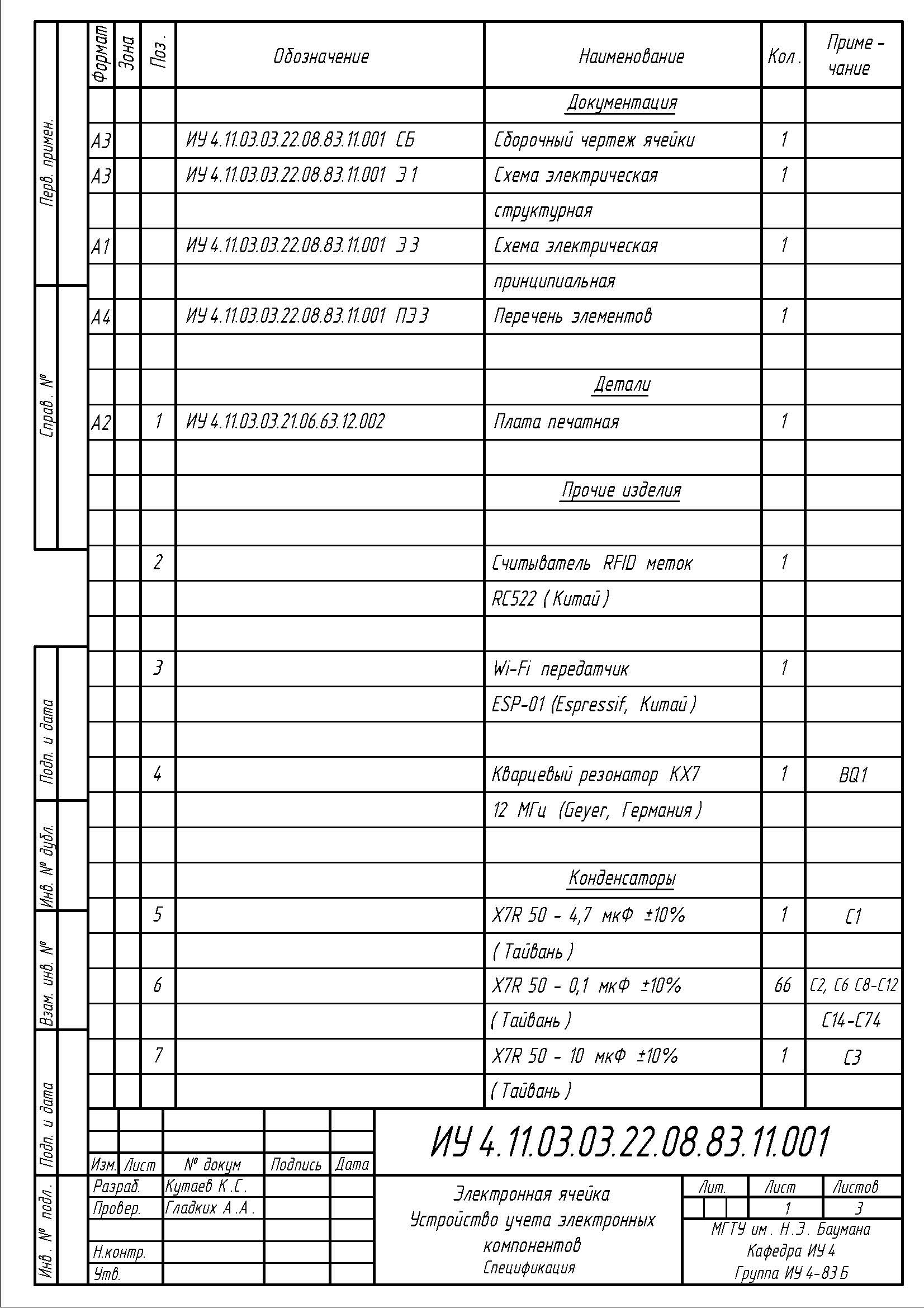
22. Документация на светодиодную ленту WS2812B. Электронный ресурс. Адрес доступа: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf. Дата обращения 13.05.2022.

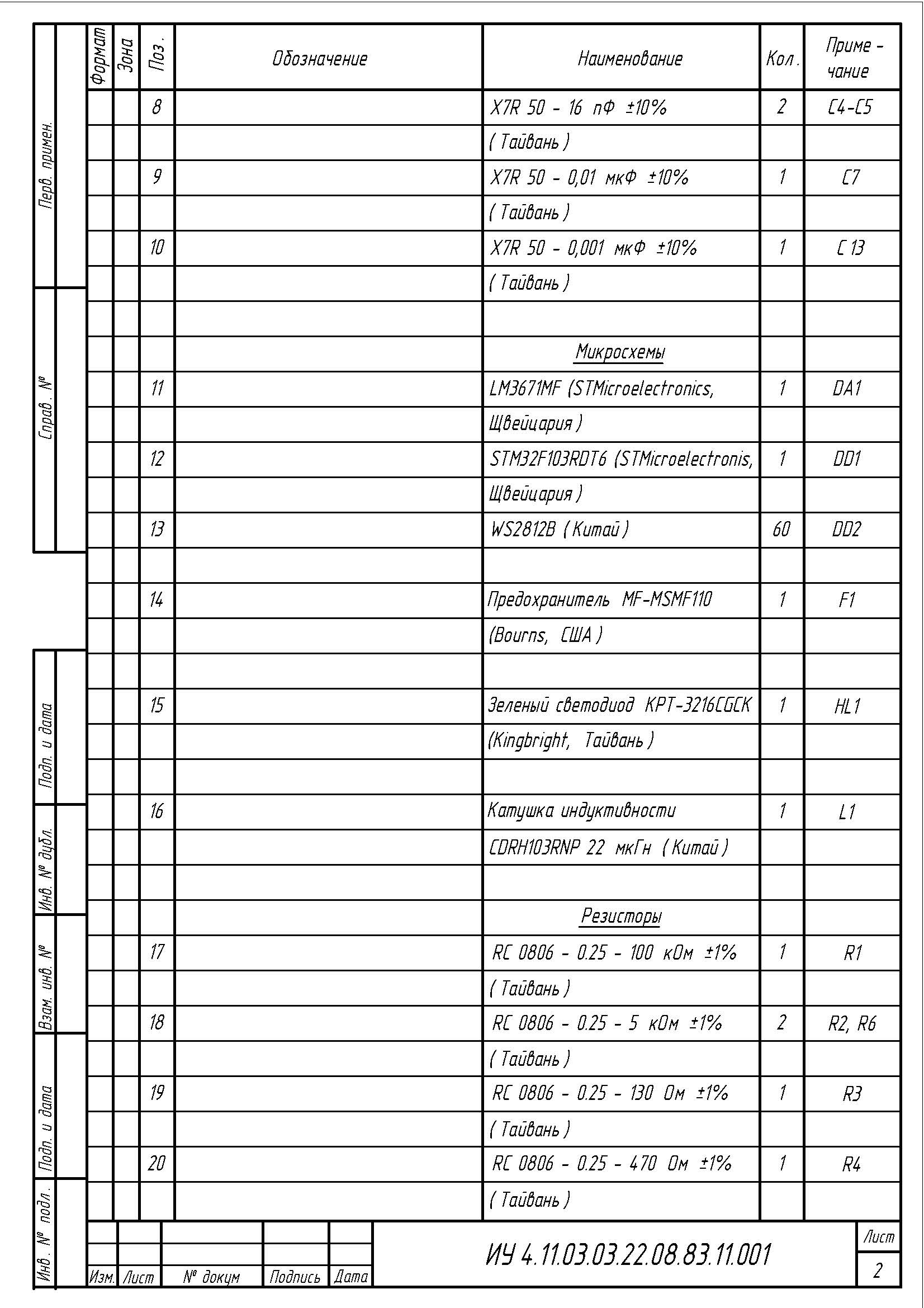
23. Документация на RFID-считыватель MFRC522. Электронный ресурс. Адрес доступа: https://www.elecrow.com/download/MFRC522%Datasheet.pdf. Дата обращения 13.05.2022.

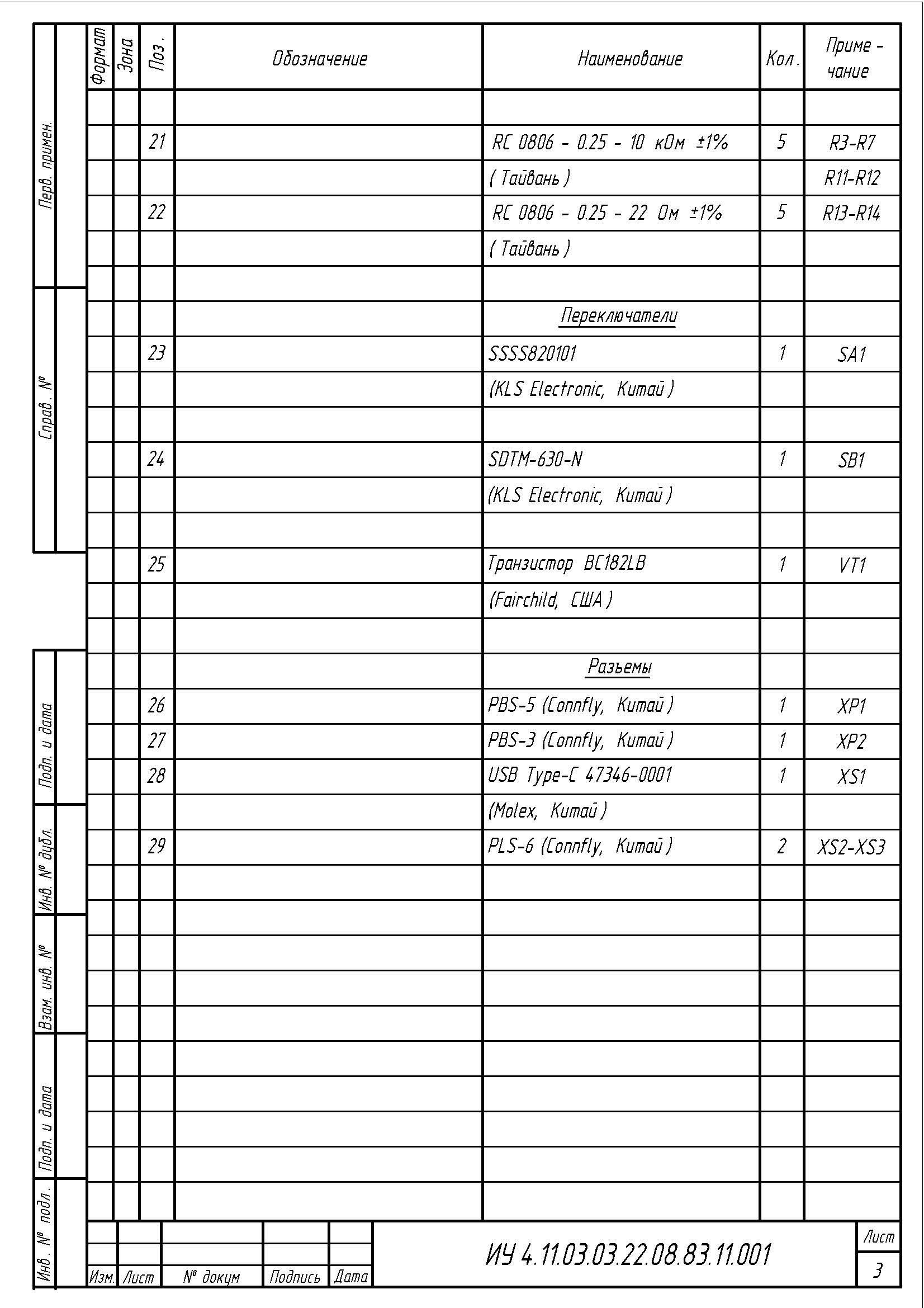
ПРИЛОЖЕНИЕ А

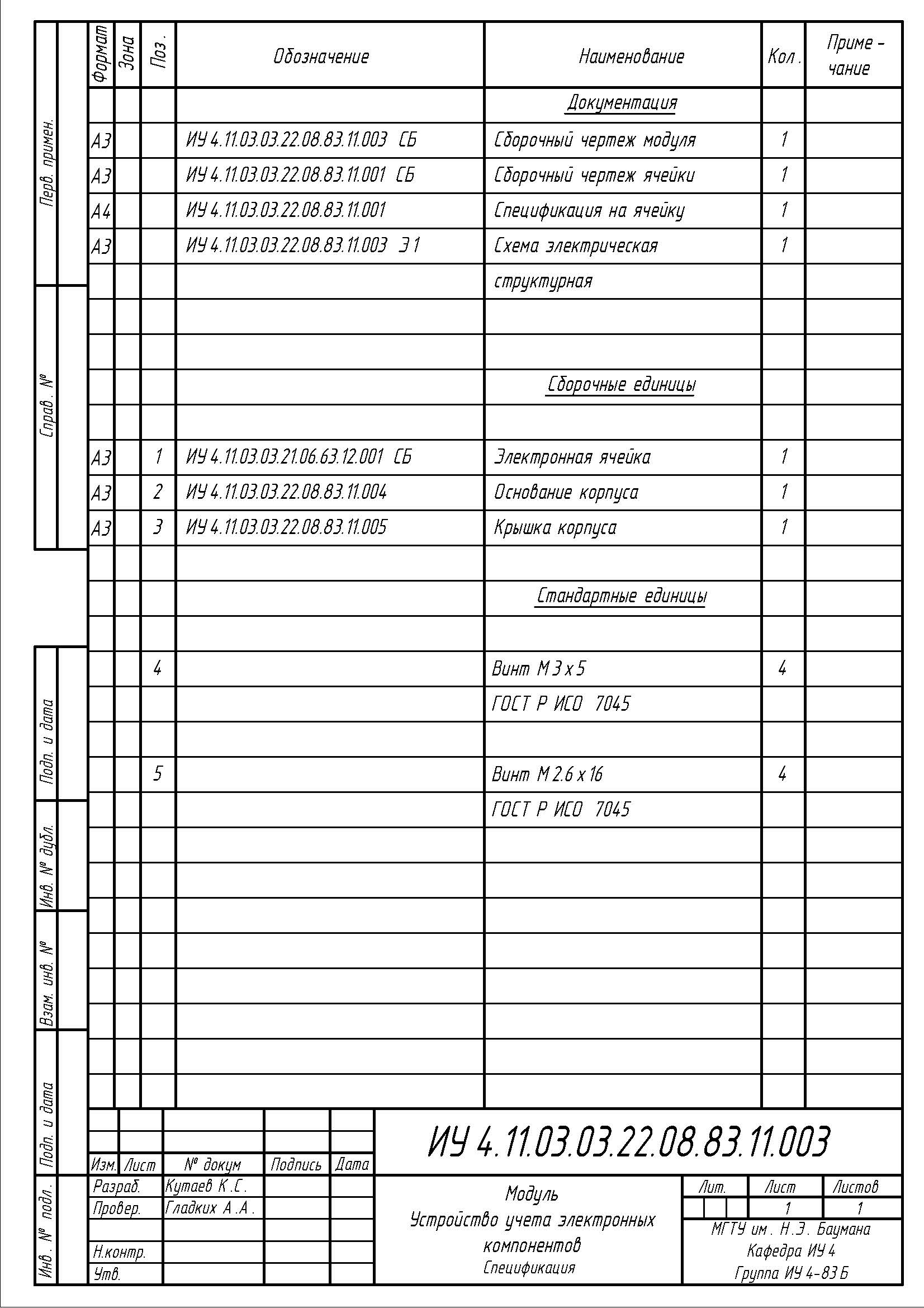












ПРИЛОЖЕНИЕ Б

**Листинг ВПО МК. Файл 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 "dma.h"

#include "spi.h"

#include "tim.h"

#include "usart.h"

#include "gpio.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* 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 ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

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

void SystemClock\_Config(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* 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 \*/

/\* 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\_SPI1\_Init();

MX\_USART2\_UART\_Init();

MX\_DMA\_Init();

MX\_USART1\_UART\_Init();

MX\_TIM1\_Init();

/\* USER CODE BEGIN 2 \*/

rfidStatus\_t status = MI\_ERR;

RetargetInit(&huart2);

MFRC522\_Init();

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "[DEBUG] LedStripStorage started!\r\n");

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

uint8\_t \*pRxData;

pRxData = (uint8\_t \*) malloc(sizeof(\*pRxData)\*48);

stream\_t local\_stream;

char\* dma\_buf = (char \*) malloc(sizeof(char)\*(ESP\_RX\_BUFFER\_SIZE + 1));

stream\_t\* stream = stream\_init(&local\_stream, dma\_buf, ESP\_RX\_BUFFER\_SIZE);

HAL\_UART\_Receive\_DMA(&huart1, (uint8\_t\*) stream->buf, stream->size);

volatile int cur\_remain = 0;

volatile int prv\_remain = stream->size;

while (1)

{

status = RFID\_ReadSectorData(1, pRxData);

if (status == MI\_OK)

{

ESP\_SendMessage(pRxData);

}

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

ESP\_MessageHandler(stream, &cur\_remain, &prv\_remain);

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

RCC\_PeriphCLKInitTypeDef PeriphClkInit = {0};

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSI;

RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL6;

RCC\_OscInitStruct.PLL.PREDIV = RCC\_PREDIV\_DIV1;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_1) != *HAL\_OK*)

{

Error\_Handler();

}

PeriphClkInit.PeriphClockSelection = RCC\_PERIPHCLK\_USART1;

PeriphClkInit.Usart1ClockSelection = RCC\_USART1CLKSOURCE\_PCLK1;

if (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInit) != *HAL\_OK*)

{

Error\_Handler();

}

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

DEBUG\_PRINT(DEBUG\_PRINT\_ERROR, "[ERROR] Error\_Handler\r\n");

\_\_disable\_irq();

while (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/

**Листинг ВПО МК. Файл main.h**

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.h

\* @brief : Header for main.c file.

\* This file contains the common defines of the application.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @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 \*/

/\* Define to prevent recursive inclusion -------------------------------------\*/

#ifndef \_\_MAIN\_H

#define \_\_MAIN\_H

/\* Includes ------------------------------------------------------------------\*/

#include "stm32f0xx\_hal.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

#include "rc522.h"

#include "stdio.h"

#include "retarget.h"

#include "string.h"

#include "stdlib.h"

#include "rfid.h"

#include "wifi.h"

#include "LedController.h"

/\* USER CODE END Includes \*/

/\* Exported types ------------------------------------------------------------\*/

/\* USER CODE BEGIN ET \*/

/\* USER CODE END ET \*/

/\* Exported constants --------------------------------------------------------\*/

/\* USER CODE BEGIN EC \*/

/\* USER CODE END EC \*/

/\* Exported macro ------------------------------------------------------------\*/

/\* USER CODE BEGIN EM \*/

/\* USER CODE END EM \*/

/\* Exported functions prototypes ---------------------------------------------\*/

void Error\_Handler(void);

/\* USER CODE BEGIN EFP \*/

void UART1\_RxCpltCallback(void);

void ESP\_RxMessageHandler(void);

/\* USER CODE END EFP \*/

/\* Private defines -----------------------------------------------------------\*/

#define SPI1\_CS\_Pin GPIO\_PIN\_4

#define SPI1\_CS\_GPIO\_Port GPIOA

#define LED\_Pin GPIO\_PIN\_3

#define LED\_GPIO\_Port GPIOB

/\* USER CODE BEGIN Private defines \*/

/\* DEBUG \*/

#define DEBUG\_PRINT\_TRACE 3

#define DEBUG\_PRINT\_INFO 2

#define DEBUG\_PRINT\_ERROR 1

#define DEBUG\_PRINT\_NONE 0

/\* Current debug lvl \*/

#define DEBUG\_PRINT\_LVL DEBUG\_PRINT\_INFO

/\* Debug output \*/

#define DEBUG\_PRINT(lvl,fmt, ...) \

do { if(DEBUG\_PRINT\_LVL >= lvl) printf(fmt, ##\_\_VA\_ARGS\_\_); } while(0)

/\* USER CODE END Private defines \*/

#endif /\* \_\_MAIN\_H \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/

**Листинг ВПО МК. Файл LedController.c**

#include "LedController.h"

uint8\_t ledData[MAX\_LED][AMOUNT\_OF\_DATA] = {0};

//bool dataSentFlag = false;

uint16\_t pwmData[(WS2812\_DATA\_BIT\_WIDTH \* MAX\_LED) + RESERVE] = {0};

void HAL\_TIM\_PWM\_PulseFinishedCallback(TIM\_HandleTypeDef \*htim)

{

HAL\_TIM\_PWM\_Stop\_DMA(&htim1, TIM\_CHANNEL\_1);

// dataSentFlag = true;

}

void LedController\_SetLED (int ledNum, int red, int green, int blue)

{

ledData[ledNum][NUMBER\_OF\_LED] = ledNum;

ledData[ledNum][GREEN\_INTENSITY] = green;

ledData[ledNum][RED\_INTENSITY] = red;

ledData[ledNum][BLUE\_INTENSITY] = blue;

}

void LedController\_WS2812Send (void)

{

uint32\_t indx = 0;

uint32\_t color;

for (int i = 0; i < MAX\_LED; i++)

{

color = ((ledData[i][1]<<SHIFT\_FOR\_GEEN\_LED) | (ledData[i][2]<<SHIFT\_FOR\_RED\_LED) | (ledData[i][3]));

for (int i = WS2812\_DATA\_BIT\_WIDTH - 1; i >= 0; i--)

{

if (color&(1<<i))

{

pwmData[indx] = 40; // 2/3 of 90

}

else

{

pwmData[indx] = 20; // 1/3 of 90

}

indx++;

}

}

for (int i = 0; i < RESERVE; i++)

{

pwmData[indx] = 0;

indx++;

}

HAL\_TIM\_PWM\_Start\_DMA(&htim1, TIM\_CHANNEL\_1, (uint32\_t \*)pwmData, indx);

// while (!dataSentFlag){

//

// };

// dataSentFlag = false;

}

void LedController\_OnLed (int ledNumber){

if (ledNumber <= MAX\_LED)

{

LedController\_SetLED(ledNumber, MIN\_COLOUR\_INTENSITY, MIN\_COLOUR\_INTENSITY, BLUE\_COLOR\_INTENSITY);

}

LedController\_WS2812Send();

HAL\_Delay(TIME\_FOR\_SENDING\_DATA);

HAL\_TIM\_PWM\_Stop\_DMA(&htim1, TIM\_CHANNEL\_1);

}

void LedController\_OffAllLeds(void){

for (int m = 0; m < MAX\_LED; m++)

{

LedController\_SetLED(m, MIN\_COLOUR\_INTENSITY, MIN\_COLOUR\_INTENSITY, MIN\_COLOUR\_INTENSITY);

}

LedController\_WS2812Send();

HAL\_Delay(TIME\_FOR\_SENDING\_DATA);

}

void LedController\_OnXY (int xLedCoordinate, int yLedCoordinate)

{

if(xLedCoordinate % 2 == UNEVEN\_CELL)

{

switch(xLedCoordinate)

{

case(FIRST\_CELL):

LedController\_OnLed(21);

LedController\_OnLed(20);

LedController\_OnLed(19);

LedController\_OnLed(18);

break;

case(FIRST\_CELL+2):

LedController\_OnLed(13);

LedController\_OnLed(12);

LedController\_OnLed(11);

break;

case(FIRST\_CELL+4):

LedController\_OnLed(6);

LedController\_OnLed(5);

LedController\_OnLed(4);

break;

}

}

else

{ //even cell

switch(xLedCoordinate)

{

case(FIRST\_CELL+1):

LedController\_OnLed(FIRST\_LED+17);

LedController\_OnLed(FIRST\_LED+16);

LedController\_OnLed(FIRST\_LED+15);

LedController\_OnLed(FIRST\_LED+14);

break;

case(FIRST\_CELL+3):

LedController\_OnLed(FIRST\_LED+10);

LedController\_OnLed(FIRST\_LED+9);

LedController\_OnLed(FIRST\_LED+8);

LedController\_OnLed(FIRST\_LED+7);

break;

case(FIRST\_CELL+5):

LedController\_OnLed(FIRST\_LED+3);

LedController\_OnLed(FIRST\_LED+2);

LedController\_OnLed(FIRST\_LED+1);

LedController\_OnLed(FIRST\_LED);

break;

}

}

if(yLedCoordinate % 2 == UNEVEN\_CELL){

switch(yLedCoordinate){

case(FIRST\_CELL):

LedController\_OnLed(FIRST\_LED+22);

LedController\_OnLed(FIRST\_LED+23);

LedController\_OnLed(FIRST\_LED+24);

break;

case(FIRST\_CELL+2):

LedController\_OnLed(FIRST\_LED+27);

LedController\_OnLed(FIRST\_LED+28);

LedController\_OnLed(FIRST\_LED+29);

break;

case(FIRST\_CELL+4):

LedController\_OnLed(FIRST\_LED+32);

LedController\_OnLed(FIRST\_LED+33);

LedController\_OnLed(FIRST\_LED+34);

break;

}

}else{ //even cell

switch(yLedCoordinate){

case(FIRST\_CELL+1):

LedController\_OnLed(FIRST\_LED+25);

LedController\_OnLed(FIRST\_LED+26);

break;

case(FIRST\_CELL+3):

LedController\_OnLed(FIRST\_LED+30);

LedController\_OnLed(FIRST\_LED+31);

break;

case(FIRST\_CELL+5):

LedController\_OnLed(FIRST\_LED+35);

LedController\_OnLed(FIRST\_LED+36);

LedController\_OnLed(FIRST\_LED+37);

break;

}

}

}

**Листинг ВПО МК. Файл LedController.h**

#ifndef INC\_LEDCONTROLLER\_H\_

#define INC\_LEDCONTROLLER\_H\_

#include "stdio.h"

#include "retarget.h"

#include "string.h"

#include "stdlib.h"

#include "usart.h"

#include "stm32f0xx\_hal.h"

#include "tim.h"

#include "dma.h"

//Strip configuration

#define MAX\_LED 53

#define USE\_BRIGHTNESS 0

#define RESERVE 50

#define WS2812\_DATA\_BIT\_WIDTH 24

#define PI 3.14159265

#define MAX\_BRIGHTNESS 45

#define MIN\_BRIGHTNESS 0

#define RIGHT\_ANGLE 90

#define UNFOLDED\_CORNER 180

#define SHIFT\_FOR\_GEEN\_LED 16

#define SHIFT\_FOR\_RED\_LED 8

#define TIME\_FOR\_SENDING\_DATA 50

#define MAX\_COLOUR\_INTENSITY 255

#define BLUE\_COLOR\_INTENSITY 31

#define MIN\_COLOUR\_INTENSITY 0

#define AMOUNT\_OF\_DATA 4

#define NUMBER\_OF\_LED 0

#define GREEN\_INTENSITY 1

#define RED\_INTENSITY 2

#define BLUE\_INTENSITY 3

#define FIRST\_LED 0

#define UNIT\_BIT 1

#define UNEVEN\_CELL 1

#define FIRST\_CELL 1

void LedController\_SetLED (int LEDnum, int Red, int Green, int Blue);

void LedController\_SetBrightness (int brightness);

void LedController\_WS2812Send (void);

void LedController\_OnLed (int ledNumber);

void LedController\_OffAllLeds (void);

void LedController\_OnXY (int xLedCoordinate, int yLedCoordinate);

#endif /\* INC\_LEDCONTROLLER\_H\_ \*/

**Листинг ВПО МК. Файл pb.c**

#include "pb.h"

#include "LedStripData.pb.h"

#include "pb\_common.h"

#include "pb\_decode.h"

#include "pb\_encode.h"

LedStripData \*PB\_Decode(char \*msg)

{

Proto ProtoReceive = {0, 0, {0}};

LedStripData ReceivedMessage = LedStripData\_init\_default;

for (int i = 0; i < 100; i++)

{

if (ProtoReceive.Data[i] == '\0')

{

ProtoReceive.messageLength = (size\_t)i;

break;

}

}

pb\_istream\_t stream = pb\_istream\_from\_buffer((uint8\_t\*)ProtoReceive.Data, ProtoReceive.messageLength);

ProtoReceive.status = pb\_decode(&stream, LedStripData\_fields, &ReceivedMessage);

if (!ProtoReceive.status)

{

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "Decoding failed: %s\n", PB\_GET\_ERROR(&stream));

return ERR\_ARG;

}

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Strip number - %d\r\n", (int)ReceivedMessage.stripNumber);

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Board number - %d\r\n", (int)ReceivedMessage.boardNumber);

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "X coordinate - %d\r\n", (int)ReceivedMessage.LedX);

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Y coordinate - %d\r\n", (int)ReceivedMessage.LedY);

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Red - %d\r\n", (int)ReceivedMessage.Colour.red);

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Green - %d\r\n", (int)ReceivedMessage.Colour.green);

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Blue - %d\r\n", (int)ReceivedMessage.Colour.blue);

}

**Листинг ВПО МК. Файл rfid.c**

#include "rfid.h"

uint8\_t BLOCK\_KEY[6] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

rfidStatus\_t RFID\_PrepareConnect(uint8\_t \*pSerialNum)

{

rfidStatus\_t status = MI\_ERR;

if (pSerialNum == NULL)

{

return status;

}

MFRC522\_Init();

HAL\_Delay(20);

uint8\_t \*pSakBuff;

pSakBuff = (uint8\_t \*) malloc(sizeof(\*pSakBuff)\*16);

status = MFRC522\_Request(PICC\_REQIDL, pSakBuff);

if (status == MI\_OK)

{

switch(pSakBuff[0])

{

case 0x44:

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Card type: MF1S500yX\r\n");

break;

case 0x04:

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Card type: MF1S503yX\r\n");

break;

case 0x42:

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Card type: MF1S700yX\r\n");

break;

case 0x02:

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Card type: MF1S703yX\r\n");

break;

case 0x08:

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Card type: MF1S50yyX/V1\r\n");

break;

default:

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "Unknown card type\r\n");

}

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "SAK: 0x%02X 0x%02X\r\n", pSakBuff[1], pSakBuff[0]);

}

free(pSakBuff);

// Anti-collision, return the card's 4-byte serial number

status = MFRC522\_Anticoll(pSerialNum);

if (status == MI\_OK)

{

DEBUG\_PRINT

(

DEBUG\_PRINT\_TRACE,

"SN: 0x%02X 0x%02X 0x%02X 0x%02X\r\n",

pSerialNum[0],

pSerialNum[1],

pSerialNum[2],

pSerialNum[3]

);

}

// Election card, return capacity

uint8\_t sizeRC;

sizeRC = MFRC522\_SelectTag(pSerialNum);

if (sizeRC != 0)

{

DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "CS: %d\r\n", sizeRC);

}

return status;

}

void RFID\_CloseConnect(void)

{

MFRC522\_Halt();

MFRC522\_AntennaOff();

}

rfidStatus\_t RFID\_ReadFullMem(void)

{

rfidStatus\_t status = MI\_ERR;

uint8\_t \*pSerialNum;

pSerialNum = (uint8\_t \*) malloc(sizeof(\*pSerialNum)\*16);

status = RFID\_PrepareConnect(pSerialNum);

if (status == MI\_OK)

{

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "[DEBUG] Card memory dump\r\n");

// Card reader

uint8\_t \*pCardBuff;

pCardBuff = (uint8\_t \*) malloc(sizeof(\*pCardBuff)\*16);

for (int sectorNum = 1;sectorNum<=16;sectorNum++)

{

int blockNum = sectorNum \* 4 - 1;

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "Sector number 0x%02X\r\n", sectorNum - 1);

for (int k = 4;k>0;k--)

{

status = MFRC522\_Auth(PICC\_AUTHENT1A, blockNum, BLOCK\_KEY, pSerialNum);

if (status == MI\_OK)

{

status = MFRC522\_Read(blockNum, pCardBuff);

if (status == MI\_OK)

{

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "0x%02X ", blockNum);

for (int j = 0;j<16;j++)

{

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "%02X ", pCardBuff[j]);

}

DEBUG\_PRINT(DEBUG\_PRINT\_INFO,"\r\n");

}

}

blockNum--;

}

}

free(pCardBuff);

}

free(pSerialNum);

RFID\_CloseConnect();

return status;

}

rfidStatus\_t RFID\_ReadSectorData(uint8\_t sectorNum, uint8\_t \*pData)

{

rfidStatus\_t status = MI\_ERR;

if (sectorNum < 1 || sectorNum > 15)

{

DEBUG\_PRINT(DEBUG\_PRINT\_ERROR, "[ERROR] Card sector must be between [1..15]\r\n");

return status;

}

if (pData == NULL)

{

DEBUG\_PRINT(DEBUG\_PRINT\_ERROR, "[ERROR] Empty data\r\n");

return status;

}

memset(pData, 0, 48);

uint8\_t \*pSerialNum;

pSerialNum = (uint8\_t \*) malloc(sizeof(\*pSerialNum)\*16);

status = RFID\_PrepareConnect(pSerialNum);

if (status == MI\_OK)

{

uint8\_t blockNum = (sectorNum + 1) \* 4 - 2;

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "Sector number 0x%02X\r\n", sectorNum);

for(uint8\_t idx = 3; idx>0 ;idx--)

{

status = MFRC522\_Auth(PICC\_AUTHENT1A, blockNum, BLOCK\_KEY, pSerialNum);

if (status == MI\_OK)

{

status = MFRC522\_Read(blockNum, pData);

if (status == MI\_OK)

{

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "0x%02X ", blockNum);

for (int idx = 0;idx<16;idx++)

{

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "%02X ", pData[idx]);

}

DEBUG\_PRINT(DEBUG\_PRINT\_INFO,"\r\n");

}

}

blockNum--;

pData += 16;

}

}

free(pSerialNum);

RFID\_CloseConnect();

return status;

}

rfidStatus\_t RFID\_WriteSectorData(uint8\_t sectorNum, uint8\_t \*pData)

{

rfidStatus\_t status = MI\_ERR;

if (pData == NULL)

{

DEBUG\_PRINT(DEBUG\_PRINT\_ERROR, "[ERROR] Empty data\r\n");

return status;

}

if (sectorNum < 1 || sectorNum > 15)

{

DEBUG\_PRINT(DEBUG\_PRINT\_ERROR, "[ERROR] Card sector must be between [1..15]\r\n");

return status;

}

uint8\_t \*pSerialNum;

pSerialNum = (uint8\_t \*) malloc(sizeof(\*pSerialNum)\*16);

status = RFID\_PrepareConnect(pSerialNum);

if (status == MI\_OK)

{

uint8\_t blockNum = (sectorNum + 1) \* 4 - 2;

for (uint8\_t idx = 3; idx>0 ;idx--)

{

status = MFRC522\_Auth(PICC\_AUTHENT1A, blockNum, BLOCK\_KEY, pSerialNum);

if (status == MI\_OK)

{

status = MFRC522\_Write(blockNum, pData);

if (status != MI\_OK)

{

DEBUG\_PRINT(DEBUG\_PRINT\_ERROR, "[ERROR] Error with writing sector %02X\r\n", sectorNum);

return status;

}

}

blockNum--;

pData += 16;

}

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "[DEBUG] Sector %02X sucessfully written\r\n", sectorNum);

}

free(pSerialNum);

RFID\_CloseConnect();

return status;

}

rfidStatus\_t RFID\_ResetAllSectorsData(void)

{

rfidStatus\_t status = MI\_ERR;

uint8\_t resetData[16] = {0};

uint8\_t \*pSerialNum;

pSerialNum = (uint8\_t \*) malloc(sizeof(\*pSerialNum)\*16);

status = RFID\_PrepareConnect(pSerialNum);

if (status == MI\_OK)

{

for (int sectorNum = 2;sectorNum<=16;sectorNum++)

{

int blockNum = sectorNum \* 4 - 2;

for (int k = 3;k>0;k--)

{

status = MFRC522\_Auth(PICC\_AUTHENT1A, blockNum, BLOCK\_KEY, pSerialNum);

if (status == MI\_OK)

{

status = MFRC522\_Write(blockNum, &resetData);

if (status != MI\_OK)

{

DEBUG\_PRINT(DEBUG\_PRINT\_ERROR, "[ERROR] Error with resetting all sectors data at %02X sector\r\n", sectorNum - 1);

return status;

}

}

blockNum--;

}

}

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "All data sectors reseted\r\n");

}

free(pSerialNum);

RFID\_CloseConnect();

return status;

}

rfidStatus\_t RFID\_WriteBasicData(void)

{

rfidStatus\_t status = MI\_ERR;

Proto TxProto = {0, 0, {0}};

Item\_t TxComponent = Item\_t\_init\_default;

Resistor\_t TxResistor = Resistor\_t\_init\_default;

TxComponent.type = *RESISTOR*;

TxResistor.partNumber = 2;

TxResistor.amount = 3;

TxResistor.maxVoltage = 4;

TxResistor.package = 5;

TxResistor.power = 6;

TxResistor.resistance = 7;

TxResistor.tolerance = 8;

pb\_ostream\_t TxStream = pb\_ostream\_from\_buffer(TxProto.Data, sizeof(TxProto.Data));

TxProto.status = pb\_encode(&TxStream, Item\_t\_fields, &TxComponent);

TxProto.status = pb\_encode(&TxStream, Resistor\_t\_fields, &TxResistor);

TxProto.messageLength = TxStream.bytes\_written;

status = RFID\_WriteSectorData(1, TxProto.Data);

return status;

}

**Листинг ВПО МК. Файл rfid.h**

#ifndef INC\_RFID\_H\_

#define INC\_RFID\_H\_

#include "rc522.h"

#include "stdio.h"

#include "retarget.h"

#include "string.h"

#include "stdlib.h"

#include "pb\_encode.h"

#include "pb\_decode.h"

#include "pb\_common.h"

#include "LedStripData.pb.h"

#define MAX\_DATA\_SIZE 48

typedef uint8\_t rfidStatus\_t;

rfidStatus\_t RFID\_ReadFullMem(void);

rfidStatus\_t RFID\_ReadSectorData(uint8\_t sectorNum, uint8\_t \*pData);

void RFID\_CloseConnect(void);

rfidStatus\_t RFID\_PrepareConnect(uint8\_t \*pSerialNum);

rfidStatus\_t RFID\_WriteSectorData(uint8\_t sectorNum, uint8\_t \*pData);

rfidStatus\_t RFID\_ResetAllSectorsData(void);

rfidStatus\_t RFID\_WriteBasicData(void);

typedef enum

{

*RESISTOR* = 1,

*CAPACITOR*,

*BJT*

} eComponenets;

typedef struct

{

size\_t messageLength;

uint8\_t status;

uint8\_t Data[MAX\_DATA\_SIZE];

} Proto;

#endif /\* INC\_RFID\_H\_ \*/

**Листинг ВПО МК. Файл wifi.c**

#include "wifi.h"

stream\_t\* stream\_init(stream\_t\* stream, char\* buf, int size) {

stream->buf = buf;

stream->size = size;

stream->pos = 0;

stream->remain = 0;

return stream;

}

void stream\_increase\_remain(stream\_t\* stream, int size) {

stream->remain += size;

}

char circular\_stream\_peek\_char(stream\_t\* stream) {

if (stream->remain == 0)

return 0;

uint16\_t pos = stream->pos % stream->size;

return stream->buf[pos];

}

char circular\_stream\_read\_char(stream\_t\* stream) {

char ch = circular\_stream\_peek\_char(stream);

if (ch != 0) {

stream->pos += 1;

stream->remain -= 1;

}

return ch;

}

int circular\_stream\_skip(stream\_t\* stream, int count) {

if (stream->remain < count)

return -1;

stream->pos += count;

stream->remain -= count;

return count;

}

int circular\_stream\_read\_string(stream\_t\* stream, char\* buf, int size) {

int k = 0;

for (k = 0; k < size; k++) {

buf[k] = circular\_stream\_read\_char(stream);

if (buf[k] == 0)

return -1;

}

return 0;

}

char\* circular\_stream\_read\_astring(stream\_t\* stream, int size) {

char\* buf = (char\*) calloc(size + 1, sizeof(char));

if (circular\_stream\_read\_string(stream, buf, size) < 0) {

free(buf);

return NULL;

}

return buf;

}

int circular\_stream\_lookup\_char(stream\_t\* stream, char ch) {

int k = 0;

for (k = 0; k < stream->remain; k++) {

int offset = stream->pos + k;

uint16\_t pos = offset % stream->size;

char sch = stream->buf[pos];

if (sch == ch)

return offset;

}

return -1;

}

char\* circular\_stream\_read\_astring\_until(stream\_t\* stream, char ch) {

int offset = circular\_stream\_lookup\_char(stream, ch);

if (offset == -1)

return NULL;

int size = offset - stream->pos;

return circular\_stream\_read\_astring(stream, size);

}

void ESP\_MessageHandler(stream\_t\* stream, volatile int \*cur\_remain, volatile int \*prv\_remain)

{

HAL\_Delay(100);

// get current remain from DMA buffer

\*cur\_remain = huart1.hdmarx->Instance->CNDTR;

// check if something incomming

int addend = 0;

if (\*cur\_remain > \*prv\_remain)

{

int size1 = \*prv\_remain;

int size2 = stream->size - \*cur\_remain;

addend = size1 + size2;

}

else if (\*cur\_remain < \*prv\_remain)

{

addend = \*prv\_remain - \*cur\_remain;

}

else

{

return;

}

stream\_increase\_remain(stream, addend);

// processing messages

while (1)

{

// filter out all bad characters

while (circular\_stream\_peek\_char(stream) == '\0')

{

if (circular\_stream\_skip(stream, 1) == -1)

{

break;

}

}

char\* msg = circular\_stream\_read\_astring\_until(stream, '\r');

if (msg == NULL)

{

break;

}

DEBUG\_PRINT(DEBUG\_PRINT\_INFO, "[ESP] %s\r\n", msg);

LedStripData \*ReceivedMessage = PB\_Decode(msg);

LedController\_OnXY(ReceivedMessage->LedX, ReceivedMessage->LedY);

free(msg);

// get rid of '\r' char

circular\_stream\_read\_char(stream);

// drop next char if it is '\n'

if ('\n' == circular\_stream\_peek\_char(stream))

{

circular\_stream\_read\_char(stream);

}

}

\*prv\_remain = \*cur\_remain;

}

**Листинг ВПО МК. Файл wifi.h**

#ifndef INC\_WIFI\_H\_

#define INC\_WIFI\_H\_

#include "stdio.h"

#include "retarget.h"

#include "string.h"

#include "stdlib.h"

#include "usart.h"

#include "stm32f0xx\_hal.h"

typedef struct {

char\* buf;

int pos;

int remain;

int size;

} stream\_t;

stream\_t\* stream\_init(stream\_t\* stream, char\* buf, int size);

void stream\_increase\_remain(stream\_t\* stream, int size);

char circular\_stream\_peek\_char(stream\_t\* stream);

char circular\_stream\_read\_char(stream\_t\* stream);

int circular\_stream\_skip(stream\_t\* stream, int count);

int circular\_stream\_read\_string(stream\_t\* stream, char\* buf, int size);

char\* circular\_stream\_read\_astring(stream\_t\* stream, int size);

int circular\_stream\_lookup\_char(stream\_t\* stream, char ch);

char\* circular\_stream\_read\_astring\_until(stream\_t\* stream, char ch);

void ESP\_MessageHandler(stream\_t\* stream, volatile int \*cur\_remain, volatile int \*prv\_remain);

#define ESP\_RX\_BUFFER\_SIZE 256

#endif /\* INC\_WIFI\_H\_ \*/

**Листинг ВПО МК. Файл rc522.с**

// Mifare RC522 RFID Card reader 13.56 MHz

#include "stm32f0xx\_hal.h"

#include "rc522.h"

#include "stdlib.h"

extern SPI\_HandleTypeDef hspi1;

// RC522

extern uint8\_t MFRC522\_Check(uint8\_t\* id);

extern uint8\_t MFRC522\_Compare(uint8\_t\* CardID, uint8\_t\* CompareID);

extern void MFRC522\_WriteRegister(uint8\_t addr, uint8\_t val);

extern uint8\_t MFRC522\_ReadRegister(uint8\_t addr);

extern void MFRC522\_SetBitMask(uint8\_t reg, uint8\_t mask);

extern void MFRC522\_ClearBitMask(uint8\_t reg, uint8\_t mask);

extern uint8\_t MFRC522\_Request(uint8\_t reqMode, uint8\_t\* TagType);

extern uint8\_t MFRC522\_ToCard(uint8\_t command, uint8\_t\* sendData, uint8\_t sendLen, uint8\_t\* backData, uint16\_t\* backLen);

extern uint8\_t MFRC522\_Anticoll(uint8\_t\* serNum);

extern void MFRC522\_CalulateCRC(uint8\_t\* pIndata, uint8\_t len, uint8\_t\* pOutData);

extern uint8\_t MFRC522\_SelectTag(uint8\_t\* serNum);

extern uint8\_t MFRC522\_Auth(uint8\_t authMode, uint8\_t BlockAddr, uint8\_t\* Sectorkey, uint8\_t\* serNum);

extern uint8\_t MFRC522\_Read(uint8\_t blockAddr, uint8\_t\* recvData);

extern uint8\_t MFRC522\_Write(uint8\_t blockAddr, uint8\_t\* writeData);

extern void MFRC522\_Init(void);

extern void MFRC522\_Reset(void);

extern void MFRC522\_AntennaOn(void);

extern void MFRC522\_AntennaOff(void);

extern void MFRC522\_Halt(void);

uint8\_t SPI1SendByte(uint8\_t data) {

unsigned char writeCommand[1];

unsigned char readValue[1];

writeCommand[0] = data;

HAL\_SPI\_TransmitReceive(&hspi1, (uint8\_t\*)&writeCommand, (uint8\_t\*)&readValue, 1, 10);

return readValue[0];

}

void SPI1\_WriteReg(uint8\_t address, uint8\_t value) {

cs\_reset();

SPI1SendByte(address);

SPI1SendByte(value);

cs\_set();

}

uint8\_t SPI1\_ReadReg(uint8\_t address) {

uint8\_t val;

cs\_reset();

SPI1SendByte(address);

val = SPI1SendByte(0x00);

cs\_set();

return val;

}

void MFRC522\_WriteRegister(uint8\_t addr, uint8\_t val) {

addr = (addr << 1) & 0x7E; // Address format: 0XXXXXX0

SPI1\_WriteReg(addr, val);

}

uint8\_t MFRC522\_ReadRegister(uint8\_t addr) {

uint8\_t val;

addr = ((addr << 1) & 0x7E) | 0x80;

val = SPI1\_ReadReg(addr);

return val;

}

uint8\_t MFRC522\_Check(uint8\_t\* id) {

uint8\_t status;

status = MFRC522\_Request(PICC\_REQIDL, id); // Find cards, return card type

if (status == MI\_OK) status = MFRC522\_Anticoll(id); // Card detected. Anti-collision, return card serial number 4 bytes

MFRC522\_Halt(); // Command card into hibernation

return status;

}

uint8\_t MFRC522\_Compare(uint8\_t\* CardID, uint8\_t\* CompareID) {

uint8\_t i;

for (i = 0; i < 5; i++) {

if (CardID[i] != CompareID[i]) return MI\_ERR;

}

return MI\_OK;

}

void MFRC522\_SetBitMask(uint8\_t reg, uint8\_t mask) {

MFRC522\_WriteRegister(reg, MFRC522\_ReadRegister(reg) | mask);

}

void MFRC522\_ClearBitMask(uint8\_t reg, uint8\_t mask){

MFRC522\_WriteRegister(reg, MFRC522\_ReadRegister(reg) & (~mask));

}

uint8\_t MFRC522\_Request(uint8\_t reqMode, uint8\_t\* respData) {

uint8\_t status;

uint16\_t backBits; // The received data bits

uint8\_t\* pSendData;

pSendData = (uint8\_t \*) malloc(sizeof(uint8\_t)\*16);

MFRC522\_WriteRegister(MFRC522\_REG\_BIT\_FRAMING, 0x07); // TxLastBists = BitFramingReg[2..0]

pSendData[0] = reqMode;

status = MFRC522\_ToCard(PCD\_TRANSCEIVE, pSendData, 1, respData, &backBits);

if ((status != MI\_OK) || (backBits != 0x10)) status = MI\_ERR;

free(pSendData);

return status;

}

uint8\_t MFRC522\_ToCard(uint8\_t command, uint8\_t\* sendData, uint8\_t sendLen, uint8\_t\* backData, uint16\_t\* backLen) {

uint8\_t status = MI\_ERR;

uint8\_t irqEn = 0x00;

uint8\_t waitIRq = 0x00;

uint8\_t lastBits;

uint8\_t FIFOLevel;

uint8\_t IRQAnsw;

uint16\_t i;

switch (command) {

case PCD\_AUTHENT: {

irqEn = 0x12;

waitIRq = 0x10;

break;

}

case PCD\_TRANSCEIVE: {

irqEn = 0x77;

waitIRq = 0x30;

break;

}

default:

break;

}

MFRC522\_WriteRegister(MFRC522\_REG\_COMM\_IE\_N, irqEn | 0x80);

MFRC522\_ClearBitMask(MFRC522\_REG\_COMM\_IRQ, 0x80);

MFRC522\_SetBitMask(MFRC522\_REG\_FIFO\_LEVEL, 0x80);

MFRC522\_WriteRegister(MFRC522\_REG\_COMMAND, PCD\_IDLE);

// Writing data to the FIFO

for (i = 0; i < sendLen; i++)

MFRC522\_WriteRegister(MFRC522\_REG\_FIFO\_DATA, sendData[i]);

//DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "[DEBUG] cmdReq -> 0x%X\r\n", MFRC522\_ReadRegister(MFRC522\_REG\_FIFO\_DATA));

// Execute the command

MFRC522\_WriteRegister(MFRC522\_REG\_COMMAND, command);

if (command == PCD\_TRANSCEIVE)

MFRC522\_SetBitMask(MFRC522\_REG\_BIT\_FRAMING, 0x80); // StartSend=1,transmission of data starts

// Waiting to receive data to complete

i = 2000; // i according to the clock frequency adjustment, the operator M1 card maximum waiting time 25ms

do {

// CommIrqReg[7..0]

// Set1 TxIRq RxIRq IdleIRq HiAlerIRq LoAlertIRq ErrIRq TimerIRq

IRQAnsw = MFRC522\_ReadRegister(MFRC522\_REG\_COMM\_IRQ);

i--;

} while ((i!=0) && !(IRQAnsw&0x01) && !(IRQAnsw&waitIRq));

MFRC522\_ClearBitMask(MFRC522\_REG\_BIT\_FRAMING, 0x80); // StartSend=0

if (i != 0) {

if (!(MFRC522\_ReadRegister(MFRC522\_REG\_ERROR) & 0x1B)) {

status = MI\_OK;

if (IRQAnsw & irqEn & 0x01)

status = MI\_NOTAGERR;

if (command == PCD\_TRANSCEIVE) {

FIFOLevel = MFRC522\_ReadRegister(MFRC522\_REG\_FIFO\_LEVEL);

lastBits = MFRC522\_ReadRegister(MFRC522\_REG\_CONTROL) & 0x07;

if (lastBits)

\*backLen = (FIFOLevel-1)\*8+lastBits;

else

\*backLen = FIFOLevel\*8;

if (FIFOLevel == 0)

FIFOLevel = 1;

// DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "[DEBUG] queueSize -> %d\r\n", FIFOLevel);

if (FIFOLevel > MFRC522\_MAX\_LEN)

FIFOLevel = MFRC522\_MAX\_LEN;

for (i = 0; i < FIFOLevel; i++)

{

backData[i] = MFRC522\_ReadRegister(MFRC522\_REG\_FIFO\_DATA); // Reading the received data in FIFO

}

// DEBUG\_PRINT(DEBUG\_PRINT\_TRACE, "[DEBUG] IRQ reg: 0x%X\r\n", IRQAnsw);

}

} else status = MI\_ERR;

}

return status;

}

uint8\_t MFRC522\_Anticoll(uint8\_t\* serNum) {

uint8\_t status;

uint8\_t i;

uint8\_t serNumCheck = 0;

uint16\_t unLen;

uint8\_t\* pSendData;

pSendData = (uint8\_t \*) malloc(sizeof(pSendData)\*16);

MFRC522\_WriteRegister(MFRC522\_REG\_BIT\_FRAMING, 0x00); // TxLastBists = BitFramingReg[2..0]

pSendData[0] = PICC\_ANTICOLL;

pSendData[1] = 0x20;

status = MFRC522\_ToCard(PCD\_TRANSCEIVE, pSendData, 2, serNum, &unLen);

if (status == MI\_OK) {

// Check card serial number

for (i = 0; i < 4; i++) serNumCheck ^= serNum[i];

if (serNumCheck != serNum[i]) status = MI\_ERR;

}

free(pSendData);

return status;

}

void MFRC522\_CalculateCRC(uint8\_t\* pIndata, uint8\_t len, uint8\_t\* pOutData) {

uint8\_t i, n;

MFRC522\_ClearBitMask(MFRC522\_REG\_DIV\_IRQ, 0x04); // CRCIrq = 0

MFRC522\_SetBitMask(MFRC522\_REG\_FIFO\_LEVEL, 0x80); // Clear the FIFO pointer

// Write\_MFRC522(CommandReg, PCD\_IDLE);

// Writing data to the FIFO

for (i = 0; i < len; i++) MFRC522\_WriteRegister(MFRC522\_REG\_FIFO\_DATA, \*(pIndata+i));

MFRC522\_WriteRegister(MFRC522\_REG\_COMMAND, PCD\_CALCCRC);

// Wait CRC calculation is complete

i = 0xFF;

do {

n = MFRC522\_ReadRegister(MFRC522\_REG\_DIV\_IRQ);

i--;

} while ((i!=0) && !(n&0x04)); // CRCIrq = 1

// Read CRC calculation result

pOutData[0] = MFRC522\_ReadRegister(MFRC522\_REG\_CRC\_RESULT\_L);

pOutData[1] = MFRC522\_ReadRegister(MFRC522\_REG\_CRC\_RESULT\_M);

}

uint8\_t MFRC522\_SelectTag(uint8\_t\* serNum) {

uint8\_t i;

uint8\_t status;

uint8\_t size;

uint16\_t recvBits;

uint8\_t \*pBackData;

uint8\_t\* pSendData;

pSendData = (uint8\_t \*) malloc(sizeof(pSendData)\*16);

pBackData = (uint8\_t \*) malloc(sizeof(pBackData)\*16);

pSendData[0] = PICC\_SElECTTAG;

pSendData[1] = 0x70;

for (i = 0; i < 5; i++) pSendData[i+2] = \*(serNum+i);

MFRC522\_CalculateCRC(pSendData, 7, &pSendData[7]); //??

status = MFRC522\_ToCard(PCD\_TRANSCEIVE, pSendData, 9, pBackData, &recvBits);

if ((status == MI\_OK) && (recvBits == 0x18)) size = pBackData[0]; else size = 0;

free(pSendData);

free(pBackData);

return size;

}

uint8\_t MFRC522\_Auth(uint8\_t authMode, uint8\_t BlockAddr, uint8\_t\* Sectorkey, uint8\_t\* serNum) {

uint8\_t status;

uint16\_t recvBits;

uint8\_t i;

uint8\_t \*pBackData;

uint8\_t\* pSendData;

pSendData = (uint8\_t \*) malloc(sizeof(pSendData)\*16);

pBackData = (uint8\_t \*) malloc(sizeof(pBackData)\*16);

// Verify the command block address + sector + password + card serial number

pSendData[0] = authMode;

pSendData[1] = BlockAddr;

for (i = 0; i < 6; i++)

pSendData[i+2] = \*(Sectorkey+i);

for (i=0; i<4; i++)

pSendData[i+8] = \*(serNum+i);

status = MFRC522\_ToCard(PCD\_AUTHENT, pSendData, 12, pBackData, &recvBits);

if ((status != MI\_OK) || (!(MFRC522\_ReadRegister(MFRC522\_REG\_STATUS2) & 0x08)))

status = MI\_ERR;

free(pSendData);

free(pBackData);

return status;

}

uint8\_t MFRC522\_Read(uint8\_t blockAddr, uint8\_t\* recvData) {

uint8\_t status;

uint16\_t unLen;

recvData[0] = PICC\_READ;

recvData[1] = blockAddr;

MFRC522\_CalculateCRC(recvData,2, &recvData[2]);

status = MFRC522\_ToCard(PCD\_TRANSCEIVE, recvData, 4, recvData, &unLen);

if ((status != MI\_OK) || (unLen != 0x90)) status = MI\_ERR;

return status;

}

uint8\_t MFRC522\_Write(uint8\_t blockAddr, uint8\_t\* writeData) {

uint8\_t status;

uint16\_t recvBits;

uint8\_t i;

uint8\_t buff[18];

buff[0] = PICC\_WRITE;

buff[1] = blockAddr;

MFRC522\_CalculateCRC(buff, 2, &buff[2]);

status = MFRC522\_ToCard(PCD\_TRANSCEIVE, buff, 4, buff, &recvBits);

if ((status != MI\_OK) || (recvBits != 4) || ((buff[0] & 0x0F) != 0x0A)) status = MI\_ERR;

if (status == MI\_OK) {

// Data to the FIFO write 16Byte

for (i = 0; i < 16; i++) buff[i] = \*(writeData+i);

MFRC522\_CalculateCRC(buff, 16, &buff[16]);

status = MFRC522\_ToCard(PCD\_TRANSCEIVE, buff, 18, buff, &recvBits);

if ((status != MI\_OK) || (recvBits != 4) || ((buff[0] & 0x0F) != 0x0A)) status = MI\_ERR;

}

return status;

}

void MFRC522\_Init(void) {

MFRC522\_Reset();

MFRC522\_WriteRegister(MFRC522\_REG\_T\_MODE, 0x8D);

MFRC522\_WriteRegister(MFRC522\_REG\_T\_PRESCALER, 0x3E);

MFRC522\_WriteRegister(MFRC522\_REG\_T\_RELOAD\_L, 30);

MFRC522\_WriteRegister(MFRC522\_REG\_T\_RELOAD\_H, 0);

MFRC522\_WriteRegister(MFRC522\_REG\_RF\_CFG, 0x70); // 48dB gain

MFRC522\_WriteRegister(MFRC522\_REG\_TX\_AUTO, 0x40);

MFRC522\_WriteRegister(MFRC522\_REG\_MODE, 0x3D);

MFRC522\_AntennaOn(); // Open the antenna

}

void MFRC522\_Reset(void) {

MFRC522\_WriteRegister(MFRC522\_REG\_COMMAND, PCD\_RESETPHASE);

}

void MFRC522\_AntennaOn(void) {

uint8\_t temp;

temp = MFRC522\_ReadRegister(MFRC522\_REG\_TX\_CONTROL);

if (!(temp & 0x03)) MFRC522\_SetBitMask(MFRC522\_REG\_TX\_CONTROL, 0x03);

}

void MFRC522\_AntennaOff(void) {

MFRC522\_ClearBitMask(MFRC522\_REG\_TX\_CONTROL, 0x03);

}

void MFRC522\_Halt(void) {

uint16\_t unLen;

uint8\_t buff[16];

buff[0] = PICC\_HALT;

buff[1] = 0;

MFRC522\_CalculateCRC(buff, 2, &buff[2]);

MFRC522\_ToCard(PCD\_TRANSCEIVE, buff, 4, buff, &unLen);

}

**Листинг ВПО МК. Файл rc522.h**

// Mifare RC522 RFID Card reader 13.56 MHz

#include "stm32f0xx\_hal.h"

#include "retarget.h"

#include "main.h"

#include "stdio.h"

// SPI CS define

//#define SPI\_I2S\_FLAG\_BSY ((uint16\_t)0x0080)

#define cs\_reset() HAL\_GPIO\_WritePin(SPI1\_CS\_GPIO\_Port, SPI1\_CS\_Pin, GPIO\_PIN\_RESET)

#define cs\_set() HAL\_GPIO\_WritePin(SPI1\_CS\_GPIO\_Port, SPI1\_CS\_Pin, GPIO\_PIN\_SET)

// Status enumeration, Used with most functions

#define MI\_OK 0

#define MI\_NOTAGERR 1

#define MI\_ERR 2

// MFRC522 Commands

#define PCD\_IDLE 0x00 // NO action; Cancel the current command

#define PCD\_AUTHENT 0x0E // Authentication Key

#define PCD\_RECEIVE 0x08 // Receive Data

#define PCD\_TRANSMIT 0x04 // Transmit data

#define PCD\_TRANSCEIVE 0x0C // Transmit and receive data,

#define PCD\_RESETPHASE 0x0F // Reset

#define PCD\_CALCCRC 0x03 // CRC Calculate

// Mifare\_One card command word

#define PICC\_REQIDL 0x26 // find the antenna area does not enter hibernation

#define PICC\_REQALL 0x52 // find all the cards antenna area

#define PICC\_ANTICOLL 0x93 // anti-collision

#define PICC\_SElECTTAG 0x93 // election card

#define PICC\_AUTHENT1A 0x60 // authentication key A

#define PICC\_AUTHENT1B 0x61 // authentication key B

#define PICC\_READ 0x30 // Read Block

#define PICC\_WRITE 0xA0 // write block

#define PICC\_DECREMENT 0xC0 // debit

#define PICC\_INCREMENT 0xC1 // recharge

#define PICC\_RESTORE 0xC2 // transfer block data to the buffer

#define PICC\_TRANSFER 0xB0 // save the data in the buffer

#define PICC\_HALT 0x50 // Sleep

// MFRC522 Registers

// Page 0: Command and Status

#define MFRC522\_REG\_RESERVED00 0x00

#define MFRC522\_REG\_COMMAND 0x01

#define MFRC522\_REG\_COMM\_IE\_N 0x02

#define MFRC522\_REG\_DIV1\_EN 0x03

#define MFRC522\_REG\_COMM\_IRQ 0x04

#define MFRC522\_REG\_DIV\_IRQ 0x05

#define MFRC522\_REG\_ERROR 0x06

#define MFRC522\_REG\_STATUS1 0x07

#define MFRC522\_REG\_STATUS2 0x08

#define MFRC522\_REG\_FIFO\_DATA 0x09

#define MFRC522\_REG\_FIFO\_LEVEL 0x0A

#define MFRC522\_REG\_WATER\_LEVEL 0x0B

#define MFRC522\_REG\_CONTROL 0x0C

#define MFRC522\_REG\_BIT\_FRAMING 0x0D

#define MFRC522\_REG\_COLL 0x0E

#define MFRC522\_REG\_RESERVED01 0x0F

// Page 1: Command

#define MFRC522\_REG\_RESERVED10 0x10

#define MFRC522\_REG\_MODE 0x11

#define MFRC522\_REG\_TX\_MODE 0x12

#define MFRC522\_REG\_RX\_MODE 0x13

#define MFRC522\_REG\_TX\_CONTROL 0x14

#define MFRC522\_REG\_TX\_AUTO 0x15

#define MFRC522\_REG\_TX\_SELL 0x16

#define MFRC522\_REG\_RX\_SELL 0x17

#define MFRC522\_REG\_RX\_THRESHOLD 0x18

#define MFRC522\_REG\_DEMOD 0x19

#define MFRC522\_REG\_RESERVED11 0x1A

#define MFRC522\_REG\_RESERVED12 0x1B

#define MFRC522\_REG\_MIFARE 0x1C

#define MFRC522\_REG\_RESERVED13 0x1D

#define MFRC522\_REG\_RESERVED14 0x1E

#define MFRC522\_REG\_SERIALSPEED 0x1F

// Page 2: CFG

#define MFRC522\_REG\_RESERVED20 0x20

#define MFRC522\_REG\_CRC\_RESULT\_M 0x21

#define MFRC522\_REG\_CRC\_RESULT\_L 0x22

#define MFRC522\_REG\_RESERVED21 0x23

#define MFRC522\_REG\_MOD\_WIDTH 0x24

#define MFRC522\_REG\_RESERVED22 0x25

#define MFRC522\_REG\_RF\_CFG 0x26

#define MFRC522\_REG\_GS\_N 0x27

#define MFRC522\_REG\_CWGS\_PREG 0x28

#define MFRC522\_REG\_\_MODGS\_PREG 0x29

#define MFRC522\_REG\_T\_MODE 0x2A

#define MFRC522\_REG\_T\_PRESCALER 0x2B

#define MFRC522\_REG\_T\_RELOAD\_H 0x2C

#define MFRC522\_REG\_T\_RELOAD\_L 0x2D

#define MFRC522\_REG\_T\_COUNTER\_VALUE\_H 0x2E

#define MFRC522\_REG\_T\_COUNTER\_VALUE\_L 0x2F

// Page 3:TestRegister

#define MFRC522\_REG\_RESERVED30 0x30

#define MFRC522\_REG\_TEST\_SEL1 0x31

#define MFRC522\_REG\_TEST\_SEL2 0x32

#define MFRC522\_REG\_TEST\_PIN\_EN 0x33

#define MFRC522\_REG\_TEST\_PIN\_VALUE 0x34

#define MFRC522\_REG\_TEST\_BUS 0x35

#define MFRC522\_REG\_AUTO\_TEST 0x36

#define MFRC522\_REG\_VERSION 0x37

#define MFRC522\_REG\_ANALOG\_TEST 0x38

#define MFRC522\_REG\_TEST\_ADC1 0x39

#define MFRC522\_REG\_TEST\_ADC2 0x3A

#define MFRC522\_REG\_TEST\_ADC0 0x3B

#define MFRC522\_REG\_RESERVED31 0x3C

#define MFRC522\_REG\_RESERVED32 0x3D

#define MFRC522\_REG\_RESERVED33 0x3E

#define MFRC522\_REG\_RESERVED34 0x3F

#define MFRC522\_DUMMY 0x00 // Dummy byte

#define MFRC522\_MAX\_LEN 16 // Buf len byte

// RC522

uint8\_t MFRC522\_Check(uint8\_t\* id);

uint8\_t MFRC522\_Compare(uint8\_t\* CardID, uint8\_t\* CompareID);

void MFRC522\_WriteRegister(uint8\_t addr, uint8\_t val);

uint8\_t MFRC522\_ReadRegister(uint8\_t addr);

void MFRC522\_SetBitMask(uint8\_t reg, uint8\_t mask);

void MFRC522\_ClearBitMask(uint8\_t reg, uint8\_t mask);

uint8\_t MFRC522\_Request(uint8\_t reqMode, uint8\_t\* TagType);

uint8\_t MFRC522\_ToCard(uint8\_t command, uint8\_t\* sendData, uint8\_t sendLen, uint8\_t\* backData, uint16\_t\* backLen);

uint8\_t MFRC522\_Anticoll(uint8\_t\* serNum);

void MFRC522\_CalulateCRC(uint8\_t\* pIndata, uint8\_t len, uint8\_t\* pOutData);

uint8\_t MFRC522\_SelectTag(uint8\_t\* serNum);

uint8\_t MFRC522\_Auth(uint8\_t authMode, uint8\_t BlockAddr, uint8\_t\* Sectorkey, uint8\_t\* serNum);

uint8\_t MFRC522\_Read(uint8\_t blockAddr, uint8\_t\* recvData);

uint8\_t MFRC522\_Write(uint8\_t blockAddr, uint8\_t\* writeData);

void MFRC522\_Init(void);

void MFRC522\_Reset(void);

void MFRC522\_AntennaOn(void);

void MFRC522\_AntennaOff(void);

void MFRC522\_Halt(void);

**Листинг ВПО МК. Файл retarget.с**

// All credit to Carmine Noviello for this code

// https://github.com/cnoviello/mastering-stm32/blob/master/nucleo-f030R8/system/src/retarget/retarget.c

#include <\_ansi.h>

#include <\_syslist.h>

#include <errno.h>

#include <sys/time.h>

#include <sys/times.h>

#include <limits.h>

#include <signal.h>

#include <../Inc/retarget.h>

#include <stdint.h>

#include <stdio.h>

#if !defined(OS\_USE\_SEMIHOSTING)

#define STDIN\_FILENO 0

#define STDOUT\_FILENO 1

#define STDERR\_FILENO 2

UART\_HandleTypeDef \*gHuart;

void RetargetInit(UART\_HandleTypeDef \*huart) {

gHuart = huart;

/\* Disable I/O buffering for STDOUT stream, so that

chars are sent out as soon as they are printed. \*/

setvbuf(stdout, NULL, \_IONBF, 0);

}

int \_isatty(int fd) {

if (fd >= STDIN\_FILENO && fd <= STDERR\_FILENO)

return 1;

errno = EBADF;

return 0;

}

int \_write(int fd, char\* ptr, int len) {

HAL\_StatusTypeDef hstatus;

if (fd == STDOUT\_FILENO || fd == STDERR\_FILENO) {

hstatus = HAL\_UART\_Transmit(gHuart, (uint8\_t \*) ptr, len, HAL\_MAX\_DELAY);

if (hstatus == HAL\_OK)

return len;

else

return EIO;

}

errno = EBADF;

return -1;

}

int \_close(int fd) {

if (fd >= STDIN\_FILENO && fd <= STDERR\_FILENO)

return 0;

errno = EBADF;

return -1;

}

int \_lseek(int fd, int ptr, int dir) {

(void) fd;

(void) ptr;

(void) dir;

errno = EBADF;

return -1;

}

int \_read(int fd, char\* ptr, int len) {

HAL\_StatusTypeDef hstatus;

if (fd == STDIN\_FILENO) {

hstatus = HAL\_UART\_Receive(gHuart, (uint8\_t \*) ptr, 1, HAL\_MAX\_DELAY);

if (hstatus == HAL\_OK)

return 1;

else

return EIO;

}

errno = EBADF;

return -1;

}

int \_fstat(int fd, struct stat\* st) {

if (fd >= STDIN\_FILENO && fd <= STDERR\_FILENO) {

st->st\_mode = S\_IFCHR;

return 0;

}

errno = EBADF;

return 0;

}

#endif //#if !defined(OS\_USE\_SEMIHOSTING)

**Листинг ВПО МК. Файл retarget.h**

// All credit to Carmine Noviello for this code

// https://github.com/cnoviello/mastering-stm32/blob/master/nucleo-f030R8/system/include/retarget/retarget.h

#ifndef \_RETARGET\_H\_\_

#define \_RETARGET\_H\_\_

#include "stm32f0xx\_hal.h"

#include <sys/stat.h>

void RetargetInit(UART\_HandleTypeDef \*huart);

int \_isatty(int fd);

int \_write(int fd, char\* ptr, int len);

int \_close(int fd);

int \_lseek(int fd, int ptr, int dir);

int \_read(int fd, char\* ptr, int len);

int \_fstat(int fd, struct stat\* st);

#endif //#ifndef \_RETARGET\_H\_\_