**СОДЕРЖАНИЕ**

[Введение 6](#_Toc105546139)

[1 Обзор аналогичных разработок 7](#_Toc105546140)

[2 Анализ технического задания 16](#_Toc105546141)

[3 Разработка структуры модуля 20](#_Toc105546142)

[**3.1 Основание для разработки** 20](#_Toc105546143)

[**3.2 Источники разработки** 22](#_Toc105546144)

[4 Аппаратно-программная реализация модуля 23](#_Toc105546145)

[**4.1 Составление функциональной схемы** 23](#_Toc105546146)

[**4.2 Описание работы составных частей принципиальной схемы** 25](#_Toc105546147)

[**4.3 Блок индикации** 26](#_Toc105546148)

[**4.4 Модуль Wi-Fi** 28](#_Toc105546149)

[**4.5 Выбор микроконтроллера** 29](#_Toc105546150)

[**4.6 Выбор элементов обвязки** 34](#_Toc105546151)

[**4.5 Расчет** **транзисторного ключа для управления реле**. 37](#_Toc105546152)

[**4.7 Расчет электромагнитной совместимости** 39](#_Toc105546153)

[5 Разработка алгоритма работы модуля 42](#_Toc105546154)

[**5.1 Проектирование алгоритма работы системы** 42](#_Toc105546155)

[**5.2 Разработка программного обеспечения** 44](#_Toc105546156)

[**5.3 Шифрование данных** 45](#_Toc105546157)

[**5.4 Тестирование пропускной способности** 47](#_Toc105546158)

[6 Разработка печатного узла модуля 48](#_Toc105546159)

[7 Технико-экономическое обоснование разработки и производства Коммуникационного контроллер с шифрованием данных для системы «умный дом» 56](#_Toc105546160)

[8 Анализ результатов проектирования 64](#_Toc105546161)

[Заключение 65](#_Toc105546162)

[Список использованных источников 66](#_Toc105546163)

[Приложение А 67](#_Toc105546164)

[Приложение Б 83](#_Toc105546165)

# Введение

На сегодняшний день микроэлектронику трудно представить без такой важной составляющей, как микроконтроллеры. На помощь человеку приходит всё больше и больше электронных помощников. Многие привыкли к ним и часто даже не подозревают, что во многих подобных устройствах присутствует микроконтроллер.

Микроконтроллерные технологии очень эффективны в решении некоторых задач. Одно и то же устройство, которое раньше собиралось на традиционных схемотехнических элементах, будучи собрано с применением микроконтроллеров, становится проще в проектировании и эксплуатации. Оно не требует регулировки и зачастую меньше в размере. Кроме того, с применением микроконтроллеров появляются практически безграничные возможности по добавлению новых потребительских функций и возможностей к уже существующим устройствам.

Телевизор, мобильный телефон, игровая приставка, стиральная машина, калькулятор – это те самые устройства, которые имеют как минимум один микроконтроллер. В случае данного дипломного проекта на микроконтроллере выполняется коммуникационный контроллер с шифрованием данных для системы «Умный дом».

Устройства систем типа «Умный дом» используются в быту для облегчения осуществления ежедневных рутинных действий в домашнем обиходе. Обычно это регулирование параметров микроклимата, управление «умной» бытовой техники дистанционно, охранные системы и системы санкционированного доступа в помещение. Для реализации подобных систем необходимо подключение множества различных устройств друг к другу и/или к общему командному серверу.

Одной из важнейших проблем таких сетей на данный момент является их незащищённость от несанкционированного доступа. Эту проблему и должно решать разрабатываемое в данном проекте устройство, выполняя шифрование передаваемых данных в сети устройств «Умного дома». Контроллер будет позволять осуществлять шифрование и передачу данных между устройствами в системе «Умный дом» по различным интерфейсам, включая беспроводную связь.

Задачи, которые были решены в рамках дипломного проектирования:

* анализ существующих систем и готовых решений;
* разработка базового прототипа;
* проектирование и разработка устройства.

# 1 Обзор аналогичных разработок

Большинство разработчиков систем умного дома не раскрывает подробностей про то, имеется ли в их продукте встроенная защита передачи данных. Делается это по нескольким причинам. Во-первых, основная масса потребителей не компетентна в достаточной мере для проведения сравнительного анализа, а так же не слишком заинтересована в мерах обеспечения повышенной безопасности сети. Во-вторых, производители не стремятся публиковать информацию об этих характеристиках собственных систем, чтобы избежать интереса злоумышленников к их разработкам в сфере информационной безопасности.

Рассмотрим некоторые стандарты сетевых протоколов, а также готовые решения на рынке и способы защиты информации в них.

ZigBee является одной из самых распространённых в использовании на данный момент спецификацией сетевых протоколов для интернета вещей (IoT) [1]. Основная особенность технологии ZigBee заключается в том, что она при малом энергопотреблении поддерживает не только простые топологии сети («точка-точка», «дерево» и «звезда»), но и самоорганизующуюся и самовосстанавливающуюся ячеистую (mesh) топологию с ретрансляцией и маршрутизацией сообщений. Кроме того, спецификация ZigBee содержит возможность выбора алгоритма маршрутизации в зависимости от требований приложения и состояния сети, механизм стандартизации приложений — профили приложений, библиотека стандартных кластеров, конечные точки, привязки, гибкий механизм безопасности (рисунок 1.1), а также обеспечивает простоту развертывания, обслуживания и модернизации.

Шифрование данных в ZigBee основано на протоколе 802.15.4 (рисунок 1.2). Алгоритм шифрования, используемый в ZigBee, — это AES (Advanced Encryption Standard) [2] с длиной ключа 128 бит (16 байт). Алгоритм AES используется не только для шифрования информации, но и для проверки отправляемых данных. Эта концепция называется проверкой целостности данных и достигается при помощи кода целостности сообщения (MIC), также называемого кодом аутентификации сообщения (MAC), который добавляется к сообщению. Этот код обеспечивает целостность заголовка MAC и прикрепленных данных полезной нагрузки.

Он создается путем шифрования частей кадра MAC-адреса IEEE с использованием ключа сети, поэтому, если мы получим сообщение от недоверенного узла, мы увидим, что MAC-адрес, сгенерированный для отправленного сообщения, не соответствует тому, который был бы сгенерирован с использованием текущего секретного ключа, поэтому мы можем отбросить это сообщение. MAC может иметь разный размер: 32, 64, 128 бит, однако всегда создается с использованием 128-битного алгоритма AES. Его размер - это просто длина битов, которая прикреплена к каждому кадру. Чем больше размер, тем безопаснее (хотя сообщение может принять меньшую полезную нагрузку). Защита данных осуществляется путем шифрования поля полезных данных с помощью 128-битного ключа.

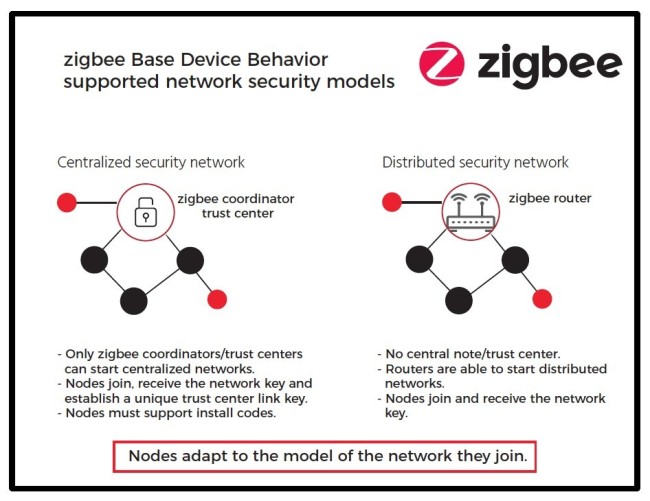


Рисунок 1.1 – Модели безопасности ZigBee

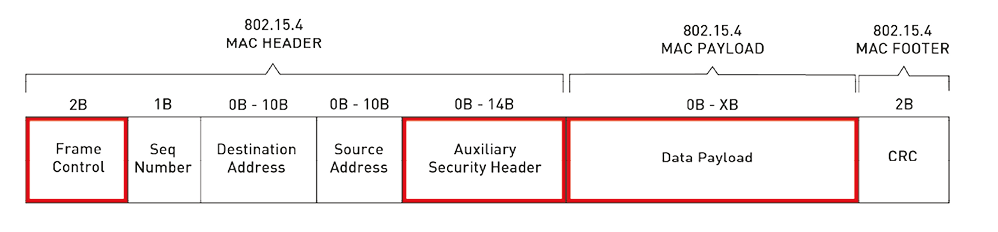


Рисунок 1.2 – Структура посылки в протоколе IEEE 802.15.4

Системы умного дома разработчика MiMiSmart (рисунок 1.3) [3] также используют передовой алгоритм шифрования Advanced Encryption Standard (AES), который широко используется в банковской сфере в том числе. Так что можно прийти к выводу, что данное шифрование довольно популярна на рынке IoT.



Рисунок 1.3 – Схема устройства систем умного дома MiMiSmart

KNX — коммуникационная шина, широко используемая для автоматизации зданий [4]. Стандарт шины KNX стал развитием более ранней разработки EIB (European Installation Bus). EIB — устаревшее обозначение, но оно продолжает использоваться, особенно в Европе. Иногда используется обозначение EIB/KNX. Продукция KNX (рисунок 1.4) распространялась под несколькими торговыми марками. Наиболее известны Instabus, ABB i-Bus, Tebis, Theben.



Рисунок 1.4 – KNX-Transceiver-Board фирмы Elmos

Автоматизация дома и зданий с KNX безопасна. Данная технология соответствует всем необходимым правилам безопасности. Технология KNX Secure стандартизирована в соответствии с EN 50090-3-4 [5], что означает, что KNX успешно блокирует хакерские атаки на цифровую инфраструктуру сетевых зданий. Таким образом, сводится к минимуму риск цифровых взломов.

Кроме того, KNX Secure соответствует самым высоким стандартам шифрования (согласно ISO 18033-3, таким как шифрование AES 128 CCM), чтобы эффективно предотвращать атаки на цифровую инфраструктуру зданий и достигать высочайшего уровня защиты данных.

KNX Secure гарантирует максимальную защиту. KNX IP Secure расширяет протокол IP таким образом, что все передаваемые телеграммы и данные полностью зашифрованы. KNX Data Secure эффективно защищает данные пользователя от несанкционированного доступа и манипуляций с помощью шифрования и аутентификации.

Производителем и разработчиком систем KNX является компания Xiot (рисунок 1.5).



Рисунок 1.5 – Xiot – разработчик KNX Secure

Control4 — поставщик систем автоматизации и сетевых систем для дома и бизнеса, предлагающий персонализированную и унифицированную систему умного дома для автоматизации и управления подключенными устройствами, включая освещение, аудио, видео, климат-контроль, внутреннюю связь и безопасность [6]. Control4 (рисунок 1.6) использует AES для шифрования данных, а также иные средства защиты персональной информации, которые не раскрываются компанией.

Control4 logo.png

Рисунок 1.6 – Компания Control4

Z-Wave — это распространённый радио протокол передачи данных, предназначенный для домашней автоматизации [7]. Характерной особенностью Z-Wave является стандартизация от физического уровня, до уровня приложения. Т.е. протокол покрывает все уровни OSI классификации, что позволяет обеспечивать совместимость устройств разных производителей при создании гетерогенных сетей.

Что позволяет делать технология Z-Wave:

* управление освещением (реле/диммеры), шторами, рольставнями и воротами;
* управление жалюзи и другими моторами (10-230 В);
* включение/выключение любых нагрузок до 3.5 кВт (модуль в розетку или встраиваемое реле);
* дистанционное управление с ПДУ;
* управление обогревом (электрические тёплые полы с защитой от перегрева, электро котлы и радиаторы, термостаты для водяных клапанов радиаторов);
* управление кондиционерами (через ИК интерфейс имитируя пульт);
* детектирование тревожных событий (датчики движения, открытия двери/окна, протечки, сухие контакты);
* мониторинг состояния (датчики температуры, влажности, освещённости);
* управление A/V аппаратурой (по протоколу Z-Wave или через ИК интерфейс имитируя пульт);
* связь с любым программным обеспечением через ПК контроллер;
* сбор данных со счётчиков.

В различных готовых решениях на основе данной технологии для шифрования обычно используется AES128. На пример в модуле ZM2102, изображенном на рисунке 1.7, на одном кристалле SD3402.

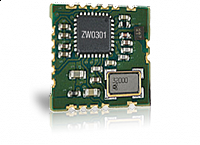


Рисунок 1.7 – Модуль ZM2102

В итоге можно прийти к следующим выводам:

* большинство производителей используют собственные реализации одного и того же международного стандарта, что повышает риски обнаружения злоумышленника уязвимости и использования ее сразу для всех устройств;
* полное или частичное сокрытие способов реализации криптозащиты является ошибкой, так как исследователи не смогут обнаружить из общедоступных источников уязвимости в алгоритмах или конкретной реализации раньше злоумышленников;
* не было обнаружено использование отечественных методов шифрования данных, что в том числе и повлияло на выбор алгоритма шифрования для разрабатываемого в дипломном проекте контроллера;
* практически полное отсутствие на рынке реализаций на отдельных микроконтроллерах, что могло бы повысить универсальность и переиспользуемость на различных системах «Умного дома» и не только, а также повысить быстродействие системы.

Из вышеизложенных тезисов и были сделаны выводы о необходимости разработки коммуникационного контроллера с шифрованием данных для системы «Умный дом».

# 2 Анализ технического задания

В данном дипломном проекте необходимо разработать и реализовать коммуникационный контроллер с шифрованием данных для системы «Умный дом».

Для удаленного управления через Интернет используется web - сервер на микроконтроллере и клиентская часть - браузер на компьютере, подключенные к сети интернет. Сервер иметь реальный ip - адрес, или выход на него через NAT - сервер (например, по технологии проброса портов). IP - может быть и динамическим, но в этом случае необходимо использование DynamicDNS(DDNS), позволяющей связать внешний динамический ip-адрес и постоянное доменное имя. Так же есть возможность управлять объктом через сервер в облаке. В этом случае данные загружаются на сервер, который способен исполнять скрипты на PHP. Данные храниться на сервере в базе данных MySQL.

Устройство подключается к серверу и проверяет данные. Если данные изменились с последнего посещения сервера, то микроконтроллер корректирует параметры работы и сохраняет новые данные в энергонезависимую память.

Wi-Fi модуль получает данные с сервера и передает из микроконтроллеру по шине RS-232. Микроконтроллер обрабатывает полученные данные и согласно управляющей программы включает либо отключает реле.

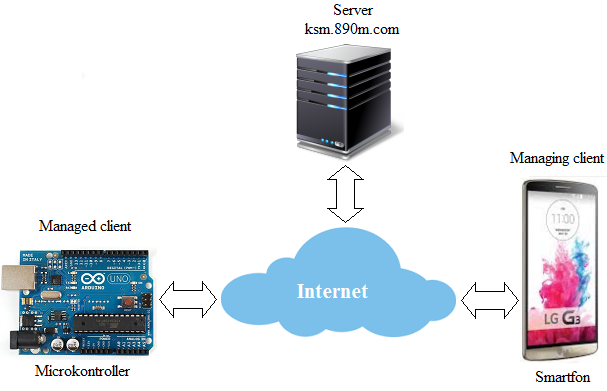


Рисунок 2.1 – Схема работы устройства дистанционного управления объектом

Клиент на микроконтроллере по командам от клиента на Андроид устройстве может управлять 2-мя устройствами (включать и выключать) и снимать показания с температурного датчика. Для поддержания стабильного питания устройства необходим стабилизатор напряжения – электромеханическое или электрическое (электронное) устройство, имеющее вход и выход по напряжению, предназначенное для поддержания выходного напряжения в узких пределах, при существенном изменении входного напряжения и выходного тока нагрузки [3].

Разработке подлежит аппаратная и программная части контроллера. Контроллер должен быть построен на базе микроконтроллера. Должен осуществляется контроль параметров измеряемого объекта. Система должна иметь достаточное быстродействие для обработки команд с передачей данных по сети интернет. Система должна иметь удобное управление для пользователя.

Требования к конструкции разрабатываемого устройства вытекают из его функционального назначения и условий его эксплуатации. Конструкция прибора должна обеспечивать ремонтопригодность, удобство в эксплуатации, иметь, по возможности малые габариты и вес, и высокую надежность в работе. Эстетические требования должны соответствовать ГОСТ 23852 – 79. Конструкция прибора должна отвечать требованиям к технологичности по ГОСТ18831 – 73 и ГОСТ 14205 – 83.

Технические характеристики:

* напряжение питания: 5 В;
* ток потребления: 500 мА;
* количество каналов управления: 1;
* разрядность АЦП: 10 бит;
* напряжение на входе АЦП: до 1 В;

Общие технические требования:

* класс точности 1,0 по ГОСТ 31819.21-2012
* температура окружающей среды от +10 до +35°С;
* относительная влажность окружающей среды до 85% при температуре 25°С;
* время непрерывной работы микропроцессорного блока без технического обслуживания – 720 часов;
* потребляемая мощность не более 10 Вт;
* масса не более 1 кг;
* средняя наработка на отказ прибора с учетом технического обслуживания – 30000 часов;
* установленная безотказная наработка 3000 часов при уровне доверия 0,8;
* полный средний срок службы устройства – 5 лет;
* средний ресурс до среднего ремонта – 10000 часов;
* установленный срок сохраняемости 1 год на период до ввода устройства в эксплуатацию;

Изделия должны сохранять свои параметры в пределах норм, установленных техническими заданиями, стандартами или техническими условиями в течение сроков службы и сроков сохраняемости, указанных в техническом задании после или в процессе воздействия климатических факторов, значения которых установлены ГОСТ 15150-69 [4].

Изделия предназначают для эксплуатации в одном или нескольких макроклиматических районах и изготавливают в различных климатических исполнениях.

Разрабатываемое устройство предназначено для эксплуатации в районах с умеренным и холодным климатами.

К макроклиматическому району с умеренным климатом относятся районы, где средняя из абсолютных максимумов температура воздуха равна или ниже + 40 °С.

К макроклиматическому району с холодным климатом относятся районы, в которых средняя из ежегодных абсолютных минимумов температура воздуха ниже - 45 °С.

Исходя из вышесказанного, устройство будет изготавливаться в климатическом исполнении УХЛ.

Следует отметить, что изделия в исполнении УХЛ могут эксплуатироваться в теплом влажном, жарком сухом и очень жарком сухом климатических районах по ГОСТ 16350-80, в которых средняя из ежегодных абсолютных максимумов температура воздуха выше 35 °С, и сочетание температуры, равной или выше -10 °С, и относительной влажности, равной или выше 90%, наблюдается более 1 часов в сутки за непрерывный период более двух месяцев в году [5].

Изделия в различных климатических исполнениях в зависимости от места размещения при эксплуатации в воздушной среде на высотах до 4300 м изготавливают по категориям размещения изделий.

При разработке должны использоваться только такие объекты интеллектуальной собственности, права на которые приобретены (получены) и используются без нарушений прав на интеллектуальную собственность третьих лиц. Это требование должно обеспечивать соблюдение авторских, смежных, патентных и иных прав.

# 3 Разработка структуры модуля

## 3.1 Основание для разработки

Принцип действия системы сводится к следующему. Вся информация с датчиков, имеющая аналоговый вид, преобразуется в цифровую форму аналого-цифровым преобразователем (АЦП) (в случае использования цифровых датчиков такой преобразователь не нужен). Затем формируется полный цифровой телеметрический сигнал, который обеспечивает высокую помехоустойчивость и эффективность. Выход приемного устройства телеметрической системы подключается к системе обработки информации, представляющую собой ЭВМ. Таким образом, последовательная цифровая передача и обработка информации приводит к системе, обладающей такими свойствами, как хорошее качество, большая скорость передачи-приема сообщений, высокая степень автоматизации, надежность, гибкость и т.д.

Структурная схема системы изображена на рисунке 3.1

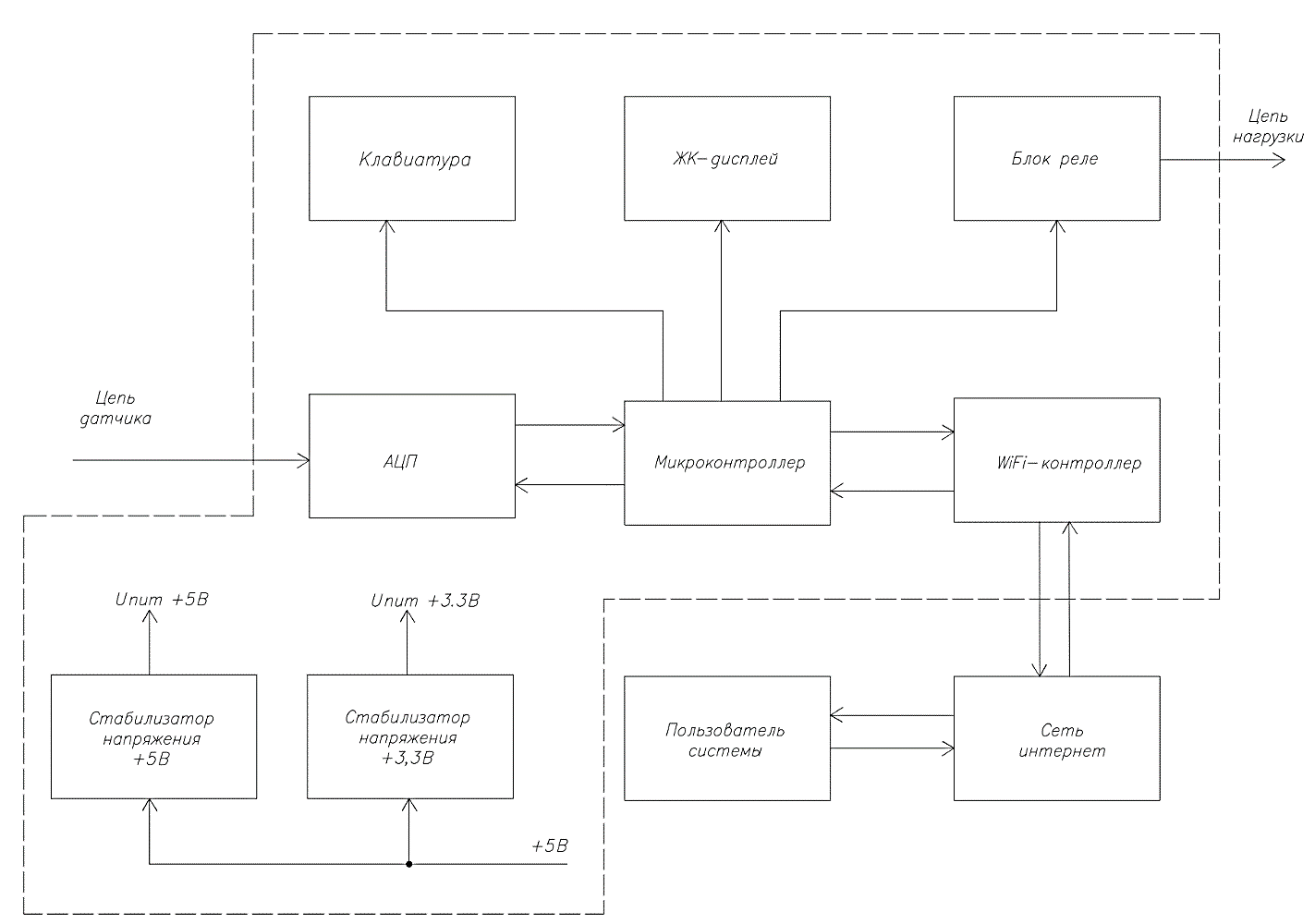


Рисунок 3.1 – Схема электрическая структурная системы

Датчики принято классифицировать[8]:

- по физическому параметру, преобразуемому в электрический сигнал, классификация датчиков весьма многообразна. Чаще всего наименование датчика согласуется с измеряемой физической величиной (например, датчик давления);

- по форме сигналов различают датчики функциональных и сигнальных параметров;

- по характеру электрических сигналов датчики подразделяются на датчики постоянного и переменного тока;

- по величине выходного электрического сигнала различают датчики сигнала высокого (0-6В) и низкого уровня (0-100 мВ). Наряду с однодиапозонными датчиками в ряде случаев используются многодиапазонные датчики, которые позволяют охватывать более широкие пределы изменения контролируемого параметра;

- по форме представления сигнала датчики делятся на аналоговые и цифровые;

- в зависимости от метода преобразования неэлектрических величин в электрические сигналы различают активные (генераторные) и пассивные (параметрические) датчики. Различие между активными и пассивными датчиками обусловлено их эквивалентными электрическими схемами, отражающими фундаментальные отличия в природе используемых в датчиках физических явлений. Активный датчик является источником непосредственно выдаваемого электрического сигнала, а измерение изменений параметров импеданса пассивного датчика производится косвенно, по изменению напряжения или тока в результате его обязательного включения в схему с внешним источником питания. Электрическая схема, непосредственно связанная с пассивным датчиком, формирует его сигнал, и, таким образом, совокупность датчика и этой электрической схемы является источником электрического сигнала.

Большинство современных и перспективных систем являются цифровыми, что обусловлено их высокими показателями. Цифровые системы обеспечивают ряд важных преимуществ по сравнению с аналоговыми, хотя и являются более сложными. К основным достоинствам цифровых методов передачи в телеметрии относятся:

- высокая точность передачи и отображения информации, практически недостижимая при современной технологии в аналоговых системах;

- высокая помехоустойчивость, возможность многократной ретрансляции и перезаписи информации;

- малый удельный расход полосы частот, то есть малые затраты полосы частот канала на передачу 1 бит/с;

- удобство использования временного разделения каналов, создания адаптивных и адресных телеметрических систем, их сопряжения с ЭВМ и интегральными сетями связи;

- широкое использование достижений современной микроэлектроники, обеспечивающее устранение противоречия между сложностью, надежностью и стоимостью цифровой телеметрической аппаратуры.

На основании схемы структурной разрабатывается схема электрическая принципиальная.

## 3.2 Источники разработки

При разработке ТЗ использовались следующие источники:

1. ГОСТ 34.003-90. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения [1].

2. ГОСТ 34.201-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем [2].

3. ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания [3].

4. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы [4].

5. ГОСТ Р МЭК 62657-2-2014. Сети промышленной коммуникации. Беспроволочные коммуникационные сети.

# 4 Аппаратно-программная реализация модуля

## 4.1 Составление функциональной схемы

На основе схемы структурной электрической разработана схема функциональная электрическая.

Блок управления согласует все блоки входящие в структурную схему. На блоке управления выполняется управляющая программа. Блок управления считывает информацию с цифровых датчиков, обрабатывает её, производит необходимые вычисления, генерирует сигнал управления на исполнительные устройства. С учетом функций блока управления для его реализации лучше использовать микроконтроллер.

Основной функциональный узел схемы – микроконтроллер DD1. С ним связаны все остальные функциональные узлы разрабатываемого устройства. Микроконтроллер – микросхема, предназначенная для управления электронными устройствами.

Типичный микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. По сути, это однокристальный компьютер, способный выполнять относительно простые задачи.

Кроме ОЗУ, микроконтроллер может иметь встроенную энергонезависимую память для хранения программы и данных. Многие модели контроллеров вообще не имеют шин для подключения внешней памяти.

Наиболее дешёвые типы памяти допускают лишь однократную запись, либо хранимая программа записывается в кристалл на этапе изготовления (конфигурацией набора технологических масок). Такие устройства подходят для массового производства в тех случаях, когда программа контроллера не будет обновляться. Другие модификации контроллеров обладают возможностью многократной перезаписи программы в энергонезависимой памяти.

Неполный список периферийных устройств, которые могут использоваться в микроконтроллерах, включает в себя:

* универсальные цифровые порты, которые можно настраивать как на ввод, так и на вывод;
* различные интерфейсы ввода-вывода, такие, как UART, I²C, SPI, CAN, USB, IEEE 1394, Ethernet;
* аналого-цифровые и цифро-аналоговые преобразователи;
* компараторы;
* широтно-импульсные модуляторы (ШИМ-контроллер);
* таймеры;
* контроллеры бесколлекторных двигателей, в том числе шаговых;
* контроллеры дисплеев и клавиатур;
* радиочастотные приемники и передатчики;
* массивы встроенной флеш-памяти;
* встроенный тактовый генератор и сторожевой таймер;
* Отличается от микропроцессора интегрированными в микросхему устройствами ввода-вывода, таймерами и другими периферийными устройствами.

Ограничения по цене и энергопотреблению ограничивает тактовую частоту контроллеров. Хотя производители стремятся обеспечить работу своих изделий на высоких частотах, они, в то же время, предоставляют заказчикам выбор, выпуская модификации, рассчитанные на разные частоты и напряжения питания. Во многих моделях микроконтроллеров используется статическая память для ОЗУ и внутренних регистров. Это даёт контроллеру возможность работать на меньших частотах и даже не терять данные при полной остановке тактового генератора. Часто предусмотрены различные режимы энергосбережения, в которых отключается часть периферийных устройств и вычислительный модуль.

Использование в современном микроконтроллере достаточного мощного вычислительного устройства с широкими возможностями, построенного на одной микросхеме вместо целого набора, значительно снижает размеры, энергопотребление и стоимость построенных на его базе устройств.

В то время как 8-разрядные микропроцессоры общего назначения полностью вытеснены более производительными моделями, 8-разрядные микроконтроллеры продолжают широко использоваться. Это объясняется тем, что существует большое количество применений, в которых не требуется высокая производительность, но важна низкая стоимость. В то же время, есть микроконтроллеры, обладающие большими вычислительными возможностями, например, цифровые сигнальные процессоры, применяющиеся для обработки большого потока данных в реальном времени (например, аудио-, видеопотоков).

Для питания устройства служит интегральный стабилизатор напряжения 5 В и 3,3В.

Для индикации информации служит модуль HG1. HG1 представляет собой жидкокристаллический индикатор на 2 строки по 16 символов.

В качестве WiFi-контроллера будет использоваться модуль WiFI с возможностью подключения к микроконтроллеру через интерфейс RS232.

Схема электрическая функциональная системы контроля и управления микроклиматом офисного помещения составляется согласно разработанной структурной схеме и имеет вид, приведённый на рисунке 4.1.

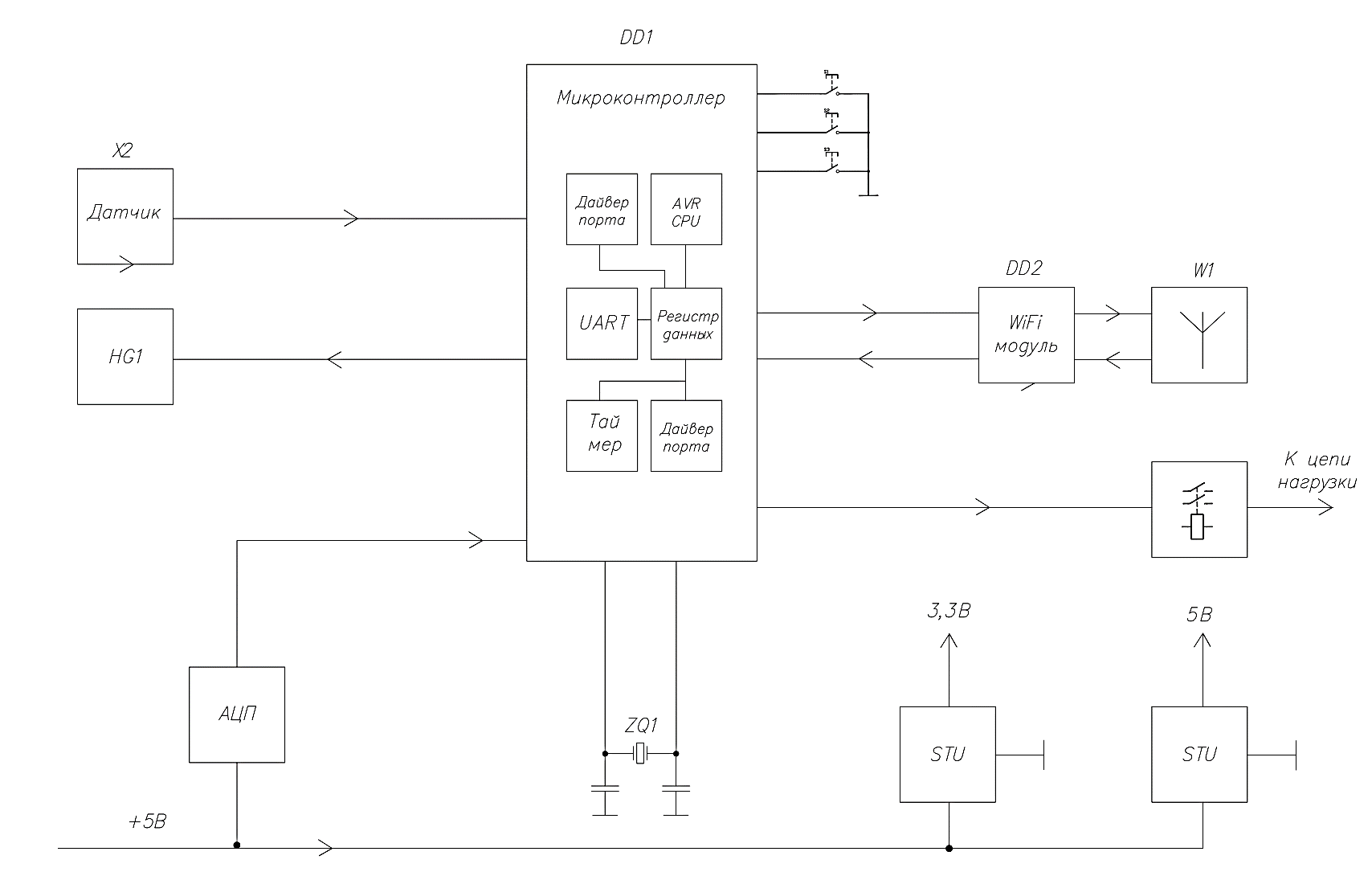


Рисунок 4.1 – Схема электрическая функциональная системы

## 4.2 Описание работы составных частей принципиальной схемы

После включения питания происходит инициализация портов микроконтроллера и его периферии, на дисплей выводится соответствующая информация об успешном включении.

По шине 1-ware происходит опрос датчиков, подключенных к разъему Х3. Микроконтроллер согласно программе производит необходимые преобразования и вычисление значений, производит сравнение полученных значений с установленными. В случае отклонения заданных параметров на порту PB3 микроконтроллера возникает высокий логический уровень, который в управляет реле. На ЖКИ-индикатор выводится соответствующая информация. Управление производится внешним исполнительным устройством.

Микроконтроллер DD2 представляет собой Wi-Fi модуль, который подключается к беспроводной сети с возможностью подключения к сети интернет. На микроконтроллере DD2 работает web-сервер, который доступен по протоколы http. Для доступа к web-серверу необходимо произвести соответствующие настройки на роутере, к которому будет подключено устройство.

По шине UART происходит обмен данными с внешним устройством, например персональным компьютером. Подключение используется для возможности изменения настроек Wi-Fi сети.

Светодиоды VD1-VD3 служат для индикации текущего режима работы.

Микросхема DA1, DA2 – стабилизаторы напряжения, конденсаторы С1, С2, С3 – сглаживают пульсации от импульсного источника питания.

Резистор R16 – регулирует контраст изображения ЖКИ-дисплея.

Резистор R17 –токоограничительный резистор для питания подсветки ЖКИ-дисплея.

С помощью кнопок SB1-SB3 пользователь устанавливает новые значения и режимы работы устройства.

## 4.3 Блок индикации

В качестве индикатора HG1 применён алфавитно-цифровой ЖК-модуль WH1602B фирмы Winstar c встроенным микроконтроллером. На рисунке 4.2 показана его структура.



Рисунок 4.2 - Структура индикатора DV1602A (HG1)

Интерфейс WH1602B − параллельный. Для соединения индикатора с микроконтроллером нужно использовать 11 линий − восемь для передачи данных (DB0-DB7), одну для информирования индикатора о направлении обмена (R/W; R/W = 1 – чтение, R/W = 0 – запись, одну для информирования о типе передаваемых данных (RS; RS = 1 – данные; RS = 0 – команда), и одну в качестве строб-сигнала, по перепаду которого из 1 в 0 осуществляется запись данных в индикатор или чтение из него. Для передачи данных будем использовать только DB4-DB7. Использование такого включения индикатора позволит сэкономить количество задействованных выводов микроконтроллера и получить при этом необходимую функциональность. Внешний вид ЖК-дисплея показан на рисунке 4.3.

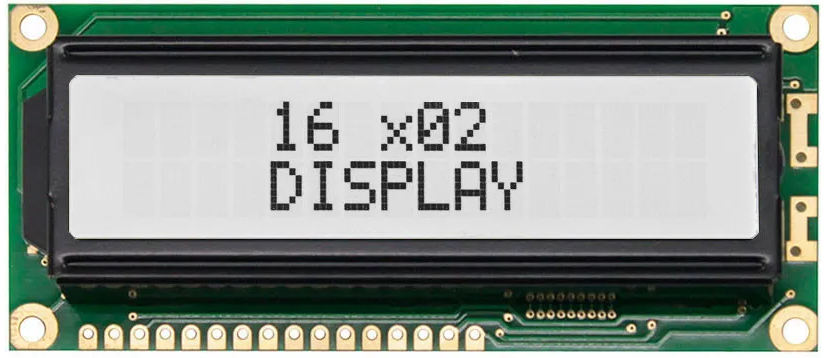


Рисунок 4.3 – Дисплей WH1602B

## 4.4 Модуль Wi-Fi

ESP32 может подключиться к Wi-Fi сети, создать собственную точку доступа, представляться сервером и клиентом, формировать GET и POST запросы. Также микроконтроллер имеет два АЦП и датчик Хола.



Рисунок 4.4 – Модуль Wi-Fi ESP32

ESP32 поддерживает весь стек протоколов стандартов Wi-Fi 802.11n и BT4.2, обеспечивая данный функционал через интерфейсы SPI/SDIO или I²C/UART.

Чип Espressif ESP 32 может работать в качестве центрального процессора (поддержка Open CPU) и как подчинённое устройство (slave device), управляемое микроконтроллером.

Отличительные особенности:

* CPU: Xtensa Dual-Core 32-bit LX6, 160 MHz или 240 MHz (до 600 DMIPS)
* Memory: 520 KByte SRAM, 448 KByte ROM
* Flash на модуле: 1, 2, 4… 64 Мб

Wireless:

* Wi-Fi: 802.11b/g/n/e/i, до 150 Mbps c HT40
* Bluetooth: v4.2 BR/EDR и BLE
* Peripheral interfaces:
* 12-bit SAR ADC до 18 каналов
* 2 × 8-bit DAC
* 10 × touch сенсоров
* Temperature сенсор
* 4 × SPI
* 2 × I²S
* 2 × I²C
* 3 × UART
* 1 host (SD/eMMC/SDIO)
* 1 slave (SDIO/SPI)
* Ethernet MAC с поддержкой DMA и IEEE 1588
* CAN 2.0
* IR (TX/RX)
* Motor PWM
* LED PWM до 16 каналов
* Hall sensor
* Ultra low power analog pre-amplifier

Security:

* IEEE 802.11 безопасность WFA, WPA/WPA2 и WAPI
* Secure boot

## 4.5 Выбор микроконтроллера

В качестве микроконтроллера DD1 выбран ATmega328 фирмы ATMEL. Он представляет собой 8-разрядный высокопроизводительный микроконтроллер с малым потреблением и имеет прогрессивную RISC архитектуру. Ниже перечислены его технические параметры [4]:

* 130 высокопроизводительных команд, большинство команд выполняется за один тактовый цикл;
* 32 8-разрядных рабочих регистра общего назначения;
* полностью статическая работа,
* производительность 16 MIPS при тактовой частоте 16 МГц;
* 8 Кбайт внутрисистемно программируемой Flash памяти (In-System Self-Programmable Flash);
* 512 байт EEPROM;
* 1 Кбайт встроенной SRAM;
* Два 8-разрядных таймера/счетчика с отдельным предварительным делителем;
* Один 16-разрядный таймер/счетчик с отдельным предварительным делителем и режимами захвата и сравнения;
* Три канала PWM;
* 8-канальный аналого-цифровой преобразователь;
* Программируемый последовательный USART;
* Последовательный интерфейс SPI (ведущий/ведомый).

Выводы I/O и корпуса:

* 23 программируемые линии ввода/вывода;
* 28-выводной корпус PDIP, 32-выводной корпус TQFP и 32-выводной корпус MLF;

    Рабочие напряжения:

* 2,7 - 5,5 В (ATmega328L);
* 4,5 - 5,5 В (ATmega328).

Рабочая частота:   
     − 0 - 8 МГц (ATmega328L);   
    − 0 - 16 МГц (ATmega328).

Расположение выводов ATMega328 в корпусе DIP28 и TQFP показан на рисунке 4.5-4.6.

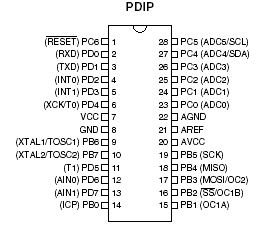


Рисунок 4.5 – Расположение выводов ATmega328 (корпус DIP28)

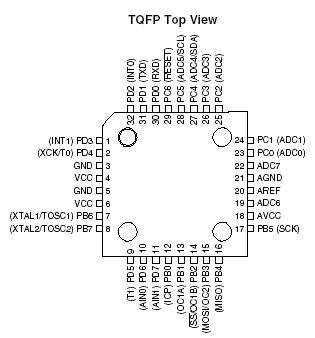


Рисунок 4.6 – Расположение выводов ATmega328 (корпус TQFP)

Таблица 4.1 – Описание выводов микроконтроллера ATmega328:

|  |  |  |  |
| --- | --- | --- | --- |
| Питание: | | | |
| № | Название | Тип | Описание |
| 7 | VCC | Вход | напряжение питания от +4.5 до +5.5 В |
| 8,22 | GND | Вход | Общий (земля) |
| 20 | AVcc | Вход | напряжение питания + 5 В для модуля АЦП |
| 21 | ARef | Вход | вход опорного напряжения для АЦП |
| Порт B: | | | |
| 14 | PB0 | Вход/Выход | цифровой порт РВ0 |
| 14 | ICP1 | Вход | захват входа 1 |
| 15 | PB1 | Вход/Выход | цифровой порт РВ1 |
| 15 | OC1A | Выход | выход сравнения/ШИМ 1А |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| 16 | PB2 | Вход/Выход | цифровой порт PB2 |
| 16 | OC1B | Выход | выход сравнения/ШИМ 1В |
| 16 | SS | Вход | вход Slave для SPI |
|  |  |  |  |
| 17 | PB3 | Вход/Выход | цифровой порт РВЗ |
| 17 | OC2 | Выход | выход сравнения/ШИМ 2 |
| 17 | MOSI | Вход/Выход | вход данных в режиме Slave для SPI и ISP / выход данных в режиме Master для SPI и ISP |
| 18 | PB4 | Вход/Выход | цифровой порт РВ4 |
| 18 | MISO | Вход/Выход | вход данных в режиме Master для SPI и ISP / выход данных в режиме Slave для SPI и ISP |
| 19 | PB5 | Вход/Выход | цифровой порт РВ5 |
| 19 | SCK | Вход/Выход | тактовый вход в режиме Slave для SPI и ISP / тактовый выход в режиме Master для SPI и ISP |
| 9 | PB6 | Вход/Выход | цифровой порт РВ6 при работе от встроенного генератора |
| 9 | XTAL1 | Вход | тактовый вход, кварцевый или керамический резонатор |
| 9 | TOSC1 | Вход | не используется при работе от внешнего генератора |
|  |  |  |  |
| 10 | PB7 | Вход/Выход | цифровой порт РВ7 при работе от встроенного генератора |
| 10 | XTAL2 | Вход | для подключения кварцевого или керамического резонатора |
| 10 | TOSC2 | Выход | тактовый выход при работе от встроенного генератора |
| Порт C: | | | |
| 23 | PC0 | Вход/Выход | цифровой порт РС0 |
| 23 | ADC0 | Вход | аналоговый вход канал 0 |
|  |  |  |  |
| 24 | PC1 | Вход/Выход | цифровой порт РС1 |
| 24 | ADC1 | Вход | аналоговый вход канал 1 |
|  |  |  |  |
| 25 | PC2 | Вход/Выход | цифровой порт PC2 |
| 25 | ADC2 | Вход | аналоговый вход канал 2 |
|  |  |  |  |
| 26 | PC3 | Вход/Выход | цифровой порт РСЗ |
| 26 | ADC3 | Вход | аналоговый вход канал 3 |
|  |  |  |  |
| 27 | PC4 | Вход/Выход | цифровой порт РС4 |
| 27 | ADC4 | Вход | аналоговый вход канал 4 |
| 27 | SDA | Вход/Выход | канал данных для 2-проводного последовательного интрефеиса |
|  |  |  |  |
| 28 | PC5 | Вход/Выход | цифровой порт РС5 |
| 28 | ADC5 | Вход | аналоговый вход канал 5 |
| 28 | SCL | Выход | тактовый выход для 2-проводного последовательного интерфейса |

Окончание таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | PC6 | Вход/Выход | | цифровой порт РС6 |
| 1 | RESET | Вход | | внешний сброс |
| Порт D: | | | | | |
| 2 | PD0 | Вход/Выход | цифровой порт PD0 | | |
| 2 | RxD | Вход | вход приемника USART | | |
|  |  |  |  | | |
| 3 | PD1 | Вход/Выход | цифровой порт PD1 | | |
| 3 | TxD | Выход | выход передатчика USART | | |
|  |  |  |  | | |
| 4 | PD2 | Вход/Выход | цифровой порт PD2 | | |
| 4 | INT0 | Вход | внешнее прерывание канал 0 | | |
|  |  |  |  | | |
| 5 | PD3 | Вход/Выход | цифровой порт PD3 | | |
| 5 | INT1 | Вход | внешнее прерывание канал 1 | | |
|  |  |  |  | | |
| 6 | PD4 | Вход/Выход | цифровой порт PD4 | | |
| 6 | XCK | Вход/Выход | внешний такт для USART | | |
| 6 | T0 | Вход | внешний вход Timer 0 | | |
|  |  |  |  | | |
| 11 | PD5 | Вход/Выход | цифровой порт PD5 | | |
| 11 | T1 | Вход | внешний вход Timer 1 | | |
|  |  |  |  | | |
| 12 | PD6 | Вход/Выход | цифровой порт PD6 | | |
| 12 | AIN0 | Вход | вход аналогового компаратора канал 0 | | |
|  |  |  |  | | |
| 13 | PD7 | Вход/Выход | цифровой порт PD7 | | |
| 13 | AIN1 | Вход | вход аналогового компаратора канал 1 | | |

## 4.6 Выбор элементов обвязки

Для питания Wi-Fi-модуля необходимо стабилизированное напряжение 3,3 В. Для этих целей служит стабилизатор напряжения на DA2 (LD1117). LD1117 - линейный регулятор напряжения 3,3 В.

В качестве микросхемы DA1 используем стабилизатор напряжения LM78L05. Максимальное входное напряжение стабилизатора – 15 В. Максимальный выходной ток – 2 А. Исходя из ТЗ и максимальному значению токов потребления, стабилизатор напряжения КР142ЕН5А подходит для использования в данной системе.

Выбор резисторов:

Чип резистор 1206

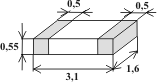


Рисунок 4.7 – резистор 1206

Технические характеристики чип резисторов 1206 1%,

Номинальная мощность чип резистора 1206 при 70°С..............0.25 Вт

Рабочее напряжение чип резистора 1206..................................200 В

Максимальное напряжение чип резистора 1206........................400 В

Диапазон рабочих температур чип резистора 1206..................-55° +125°С

Температурный коэффициент сопротивления...........................100 ppm/°С

Чип резисторы типоразмера 1206 5% поставляются со склада по ряду e24.

Этот типоразмер удобен при выборе низкоомных резисторов.

Резистор С2-33Н - резисторы постоянные непроволочные общего применения всеклиматического неизолированного варианта исполнения, предназначены для работы в электрических цепях постоянного, переменного токов и в импульсном режиме.

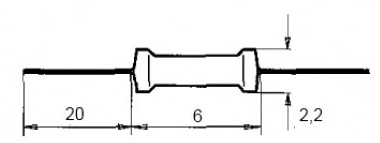


Рисунок 4.8 – резистор С2-33Н

Выбор конденсаторов:

Конденсатор танталовый SMD



Рисунок 4.9 – Конденсатор танталовый SMD

Танталовые электролитические конденсаторы – это малогабаритные конденсаторы высокой стабильности с низким током утечки, устойчивыми частотными и температурными характеристиками и длительным сроком службы.

Пригодны как для автоматического, так и для ручного монтажа.

Герметичная конструкция обеспечивает защиту от воздействия влаги, допускает промывку платы после монтажа.

На корпусе указываются номинальная емкость, рабочее напряжение, черная полоса со стороны положительного вывода.

Выбор оптопары

Оптопары являются электронными приборами, которые служат для гальванической развязки электрических цепей.

Таблица 4.2 – параметры потопары MOC5103

|  |  |
| --- | --- |
| Количество каналов | 1 |
| Тип выхода | фототранзистор |
| Напряжение изоляции,кВ | 5 |
| Максимальный прямой ток,мА | 50 |
| Максимальное выходное напряжение, В | 70 |
| Время включения/выключения, мкс | 2 |
| Тип корпуса | dip6 |

Механизмом преобразования электрического сигнала в оптопаре служат отдельные компоненты. Светодиод в качестве излучателя, и фотоприемник, в лице фотодиода, фототранзистора, фоторезистора, фототиристора или фотодинистора.

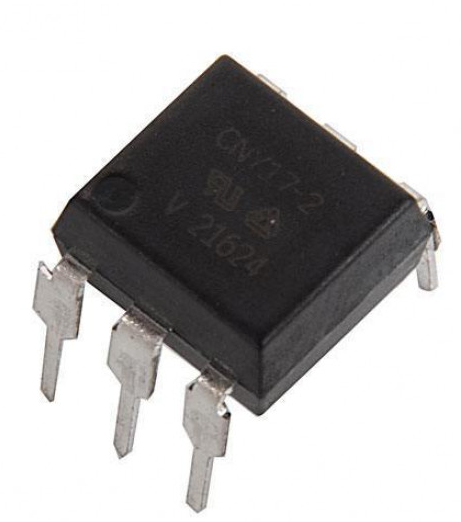


Рисунок 4.10 – Оптопара

Именно оптические внутриструктурные элементы оптрона определяют его класс и рабочие возможности.

## 4.5 Расчет транзисторного ключа для управления реле.

В качестве нагрузки выступает контактор типа КНЕ030 на напряжение 27В с катушкой сопротивлением 150 Ом. Индуктивным характером катушки в данном примере пренебрежем, считая, что реле будет включено раз и надолго.

Рассчитываем ток коллектора:

Ik=(Ucc-Uкэнас)/Rн    , где

Ik –ток коллектора

Ucc- напряжение питания (27В)

Uкэнас- напряжение насыщения биполярного транзистора (типично от 0.2 до 0.8В, хотя и может прилично различаться для разных транзисторов), в нашем случае примем 0.4В

Rн – сопротивление нагрузки (150 Ом)

Ik= (27-0.4)/150 = 0.18A = 180мА

На практике из соображений надежности элементы всегда необходимо выбирать с запасом. Возьмем коэффициент 1.5

Таким образом, нужен транзистор с допустимым током коллектора не менее 1.5\*0.18=0.27А и максимальным напряжением коллектор-эмиттер не менее 1.5\*27=40В.

В справочнике по биполярным транзисторам.  По заданным параметрам подходит [КТ815А](http://www.trzrus.ru/sourse/kt815.htm) (Ikмакс=1.5А Uкэ=40В)

Следующим этапом рассчитываем ток базы, который нужно создать, чтобы обеспечить ток коллектора 0.18А.

 Как известно, ток коллектора связан с током базы соотношением

Ik=Iб\*h21э,

где h21э – статический коэффициент передачи тока.

 При отсутствии дополнительных данных можно взять табличное гарантированное минимальное значение для КТ815А (40). Но для [КТ815](http://www.trzrus.ru/sourse/kt815.pdf)есть график зависимости h21э от тока эмиттера. В нашем случае ток эмиттера 180мА, этому значению соответствует h21э=60. Разница невелика, но для чистоты эксперимента возьмем графические данные.

Iб=180/60=3мА

Для расчета базового резистора R1 смотрим второй график, где приведена зависимость напряжения насыщения база-эмиттер (Uбэнас) от тока коллектора. При токе коллектора 180мА напряжение насыщения базы будет 0.78В (При отсутствии такого графика можно использовать допущение, что ВАХ перехода база-эмиттер подобна ВАХ диода и в диапазоне рабочих токов напряжение база-эмиттер находится в пределах 0.6-0.8 В)

Следовательно, сопротивление резистора R1 должно быть равно:

R1=(Uвх-Uбэнас)/Iб = (5-0.78)/0.003 = 1407 Ом = 1.407 кОм.

Из стандартного ряда сопротивлений выбираем ближайшее в меньшую сторону (1.3 кОм)

Если к базе подключен шунтирующий резистор (вводится для более быстрого выключения транзистора или для повышения помехоустойчивости) нужно учитывать, что часть входного тока уйдет в этот резистор, и тогда формула примет вид:

R1= (Uвх-Uбэнас)/(Iб+IR2) = (Uвх-Uбэнас)/(Iб+ Uбэнас/R2)

Так, если R2=1 кОм, то

R1= (5-0.78)/(0.003+0.78/1000) = 1116 Ом = 1.1 кОм

Рассчитываем потери мощности на транзисторе:

P=Ik\*Uкэнас

Uкэнас (из документации на транзистор): при 180мА оно составляет 0.07В

P= 0.07\*0.18= 0.013 Вт

Мощность не большая, радиатора не потребуется.

## 4.7 Расчет электромагнитной совместимости

В радиоэлектронных изделиях печатные проводники, электрически объединяющие те или иные элементы схемы, проходят на достаточно близком расстоянии друг от друга и имеют относительно малые размеры сечения. При большом времени переключения и малых тактовых частотах параметры печатных проводников, соединяющие вводы одних элементов со входами других, не оказывают существенного воздействия на быстродействие всей схемы в целом и на помехоустойчивость элементов.

С уменьшением времени переключения (в микроэлектронных изделиях оно составляет единицы наносекунд) большое значение имеют степени влияния линий связи (сопротивления, емкости, индуктивности и т.д.) друг на друга (паразитная емкость, взаимоиндуктивность и т.д.). Постоянный ток в печатных проводниках распределяется равномерно по его сечению при условии, что материал проводника однороден и не имеет локальных посторонних включений других веществ [19].

Рассчитаем сопротивление проводника по формуле (4.1):

R= (4.1)



где ρ - удельное объемное электрическое сопротивление проводника, ρ = 0,0175 мкОм/м – для медных проводников, полученных методом химиче­ского травления.

ln– длинна проводника, мм,

b – ширина проводника, мм,

tn – толщина проводника, мкм,



R= = 0,202 Ом. (4.2)

Рассчитаем допустимый ток в печатном проводнике:

Рисунок 4.11. Фрагмент печатной платы



Imax=10-3⋅γдоп⋅b⋅tn, (4.3)

где γдоп – допустимая плотность тока, γдоп=30 А/мм2для проводников, полученных методом химического травления.

Imax=10-3⋅30 ⋅0,25⋅35=0,26 мА.

Рассчитаем паразитную емкость в выбранном нами участке, где она наибольшая. Ёмкость между двумя выбранными проводящими элементами определяем по формуле:

 (4.4)

где:  - диэлектрическая проницаемость среды между проводниками, рас­положенных на наружных поверхностях платы, покрытой лаком.

 (4.5)

где ξп и ξл- диэлектрические проницаемости материала платы и лака (для стеклотекстолита ξП = 6, для лака ЭП ξЛ =4).

 (4.6)

ln - длина участка, на котором проводники параллельны друг другу, мм (ln =101 мм),

b - ширина проводника, мм (b=0,25 мм),

tn – толщина проводника, мм (tn =35мкм),

а - толщина диэлектрика, мм (а =0,25 мм),



(4.7)

Так как значение паразитной емкости достаточно мало, то никаких мер принимать не следует.

# 5 Разработка алгоритма работы модуля

## 5.1 Проектирование алгоритма работы системы

Процесс разработки программного обеспечения для микроконтроллера – это наиболее сложная часть конструирования, именно это вызывает наибольшие сложности. Совпадение целей и интересов у автора-разработчика и человека, решившего повторить конструкцию, происходит редко, и это порождает множество вопросов.

При написании программ для микроконтроллеров все большую популярность приобретает язык Си. При его использовании сокращается время на разработку, что особенно заметно при написании больших программ, обеспечивается их переносимость на другие платформы. Недостатком языка Си по сравнению с языком Ассемблер является больший объем кода и, как следствие, более низкая скорость работы. Однако благодаря тому, что в архитектуре AVR изначально заложено эффективное декодирование и исполнение инструкций, генерируемых компиляторами, после компиляции Си-программ получается высокопроизводительный код.

Разработка программного обеспечения для микроконтроллеров AVR мало отличается от разработки любых иных программ, разве что от программиста требуется чуть более глубокие знания электроники – хотя бы на уровне понимания действия логических элементов и триггеров. Разумеется, чем лучше программист владеет электроникой, тем более качественные программы он сможет создать. Тесная связь «софта и железа» по сути требует, чтобы программист был электронщиком или же электронщик был программистом. В настоящее время обе ситуации имеют место. Существует три более-менее устоявшихся подхода к разработке микроконтроллерных устройств:

- от схемы к программе

- от программы к схеме

- смешанный.

Далее описан процесс разработки и отладки программы на языке Си для дипломного проекта. В качестве среды программирования использована программа AVR Studio 5.

Алгоритм – точный набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за определенное время.

Вообще, написать программу можно в любом текстовом редакторе, так же как вы бы написали письмо другу, например. После этого, текст надо скомпилировать (иногда говорят - ассемблировать) т.е. перевести в форму, понятную процессору. Суть работы компилятора в переводе письменных символов понятных для человека в машинный код (в код нулей и единиц) и создание нового файла с расширением .hex

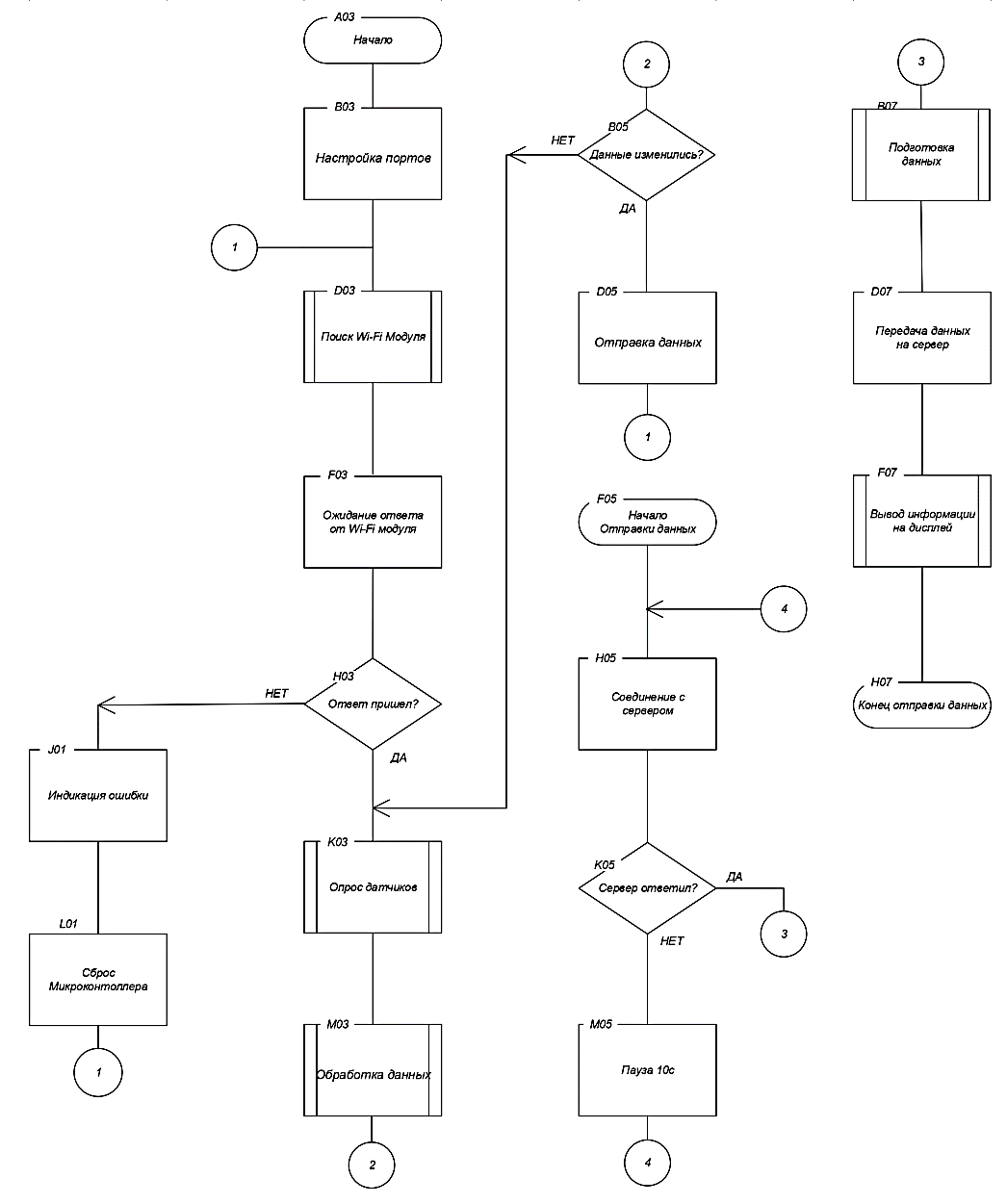


Рисунок 5.1 – Алгоритм работы устройства

Алгоритм функционирования устройства определяется управляющей программой МК.

Устройство функционирует следующим образом:

* включение;
* инициализация микроконтроллера;
* Инициализация датчиков
* определение режима работы;
* цикл обработки прерываний;
* индикация;
* передача данных Wi-Fi-модулю;
* функционирование по результатам обработанных данных от датчиков;
* условие выхода из цикла – отключение или перенастройка системы;

После инициализации устройство переходит в режим запроса о дальнейшем состоянии. Алгоритм главного цикла работы представлен на рисунке 5.1.

Инициализация – создание, активация, подготовка к работе, определение параметров. Приведение программы или устройства в состояние готовности к использованию.

Представленные уровневые структуры алгоритма легли в основу разработанной программы системы команд микроконтроллера ATmega328.

Назначение языка программирования – предоставить программисту средства для изложения алгоритма решения какой-либо задачи в форме, воспринимаемой компилятором (специальной программой, служащей для «перевода» с одного языка на другой, понятный конкретной аппаратной платформе, т.е. процессору или микроконтроллеру).

## 5.2 Разработка программного обеспечения

AVR Studio — основанная на Visual Studio бесплатная проприетарная интегрированная среда разработки (IDE) для разработки приложений для 8- и 32-битных микроконтроллеров семейства AVR и 32-битных микроконтроллеров семейства ARM от компании Atmel, работающая в операционных системах Windows. AVR Studio содержит компилятор GNU C/C++ и эмулятор, позволяющий отладить выполнение программы без загрузки в микроконтроллер.

Ранее среда разработки носила название AVR Studio, но начиная с версии 4.0, вышедшей в 2012 году, в неё была добавлена поддержка разработки для микроконтроллеров архитектуры ARM, также выпускаемых фирмой Atmel, и среда разработки получила новое название Atmel Studio.

Atmel Studio содержит в себе менеджер проектов, редактор исходного кода, инструменты виртуальной симуляции и внутрисхемной отладки, позволяет писать программы на ассемблере или на C/C++.

Внешний вид окна программы (IDE) AVR Studio 5 представлен на рисунке 5.2.

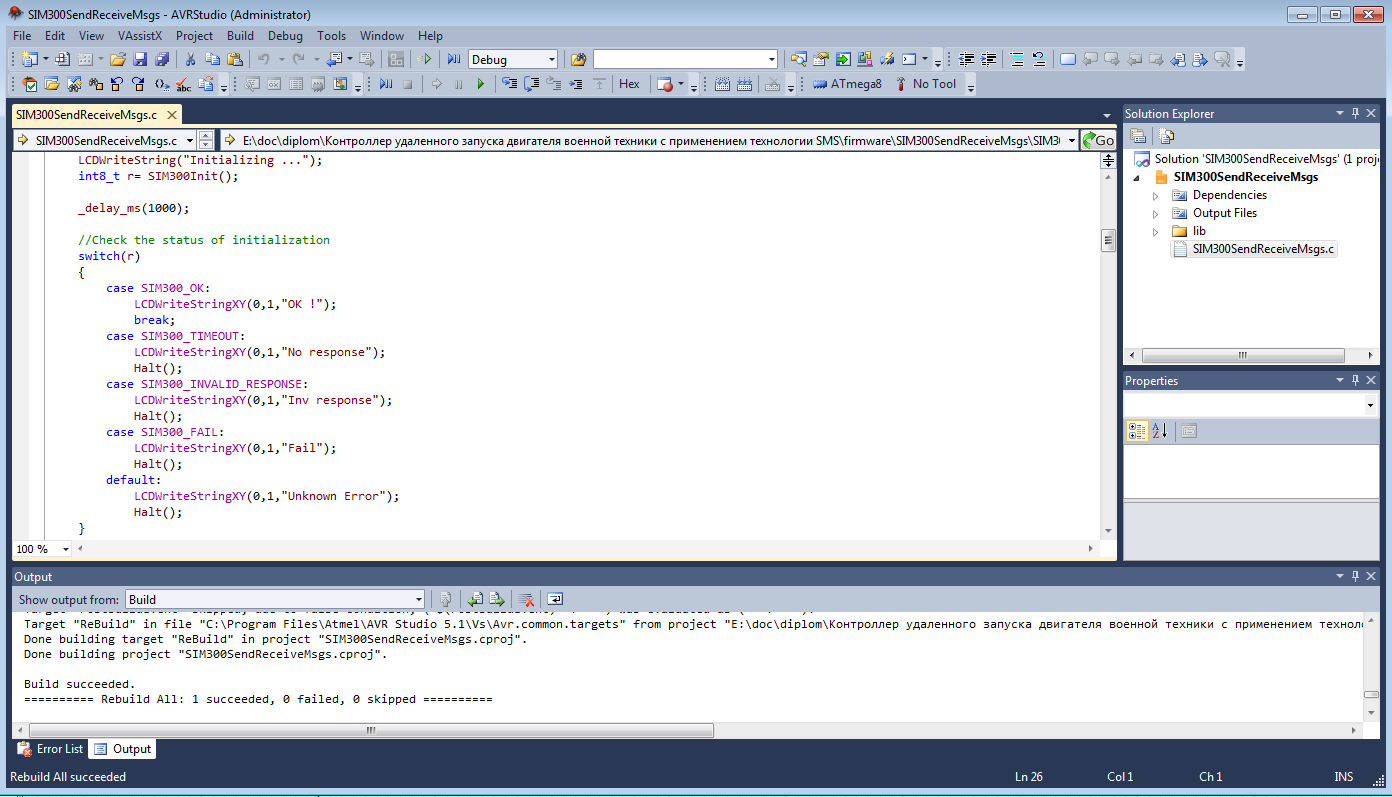


Рисунок 5.2 – Интерфейс ИСР AVR Studio 5

Листинг программы приведён в приложении А.

## 5.3 Шифрование данных

В некоторых случаях данные, собираемые датчиками системы умный дом могут содержать чувтсвительную информацию. Например, по ним можно определить, находятся ли дома люди. Поэтому передаваемая информация должна быть надежно защищена.

Для защиты данных, которыми обменивается устройство был выбран стандарт BelT. BelT — государственный стандарт симметричного шифрования и контроля целостности Республики Беларусь. Полное название стандарта — СТБ 34.101.31-2007 «Информационные технологии и безопасность. Криптографические алгоритмы шифрования и контроля целостности». Принят в качестве предварительного стандарта в 2007 году. Введен в действие в качестве окончательного стандарта в 2011 году.

BelT — блочный шифр с 256-битным ключом и 8 циклами криптопреобразований, оперирующий с 128-битными словами. Криптографические алгоритмы стандарта построены на основе базовых режимов шифрования блоков данных. Все алгоритмы стандарта делятся на 8 групп:

1. алгоритмы шифрования в режиме простой замены;
2. алгоритмы шифрования в режиме сцепления блоков;
3. алгоритмы шифрования в режиме гаммирования с обратной связью;
4. алгоритмы шифрования в режиме счётчика;
5. алгоритм выработки имитовставки;
6. алгоритмы одновременного шифрования и имитозащиты данных;
7. алгоритмы одновременного шифрования и имитозащиты ключей;
8. алгоритм хэширования;

В данной работе нас интересуют первые четыре группы, которые предназначены для обеспечения безопасного обмена сообщениями. Каждая группа включает алгоритм зашифрования и алгоритм расшифрования на секретном ключе. Стороны, располагающие общим ключом, могут организовать обмен сообщениями путём их зашифрования перед отправкой и расшифрования после получения. В режимах простой замены и сцепления блоков шифруются сообщения, которые содержат хотя бы один блок, а в режимах гаммирования с обратной связью и счётчика — сообщения произвольной длины.

Поскольку наше устройтво обменивается небольшими посылками данных, в качестве алгоритма шифрования был выбран BelT в режиме счетчика. Этот режим позволяет сократить объем передаваемых данных, т. к. работает с посылками любой длины, а также сократить объем кода, т. к. для расшифрования может быть использована та же функция, что и для шифрования данных.

При шифровании в режимах сцепления блоков, гаммирования с обратной связью и счетчика используется синхропосылка S ∈ {0, 1}128, которая обеспечивает уникальность криптографических преобразований на фиксированном ключе.

## 5.4 Тестирование пропускной способности

Алгоритмы шифрования как правильно требуют значительных вычислительных ресурсов. В связи с этим, важным вопросом является оценка скорости работы алгоритма шифрования BelT в режиме счетчика на микронтроллере ATmega328, а также пропускной способоности разрабатываемого коммуникационного контроллера.

Для оценки скорости работы шифрования и пропускной способности контроллера была разработана тестовая программа для микроконтроллера Atmega328, осуществляющая шифрование и расшифрование блока данных размером 100 000 байт, а также производящая замер интервала времени. Результаты выводятся в терминал подключенного компьютера через порт UART с использованием библиотеки Serial.

belt\_ctr\_st stateEncr = {0};

long time\_counter = 0;

unsigned long interval = 0;

// инициализация состояния шифратора

beltCTRStart(&stateEncr, key, len, iv);

unsigned long start\_time = millis();

for(unsigned long k=0;k<100000;k++) {

// шифрование данных

beltCTRStepE(&outData, 1, &stateDecr);

}

unsigned long end\_time = millis();

unsigned long elapsed = end\_time - start\_time;

Serial.print("Elapsed ms: ");

Serial.println(elapsed);

В результате запуска данной программы на микроконтроллере ATmega328 была получена оценка скорости шифрования при помощи алгоритма BelT в режиме счетчика, которая составила 176 килобит в секунду, что является достаточным для устройств системы умный дом.

# 6 Разработка печатного узла модуля

Рассмотрим технологию разработки печатной платы в САПР Altium Designer. Начальным этапом разработки любого радиоэлектронного устройства является описание его работы на некотором уровне абстракции, в данном случае – схемы электрической принципиальной [28].

Формирование новой электрической схемы в Altium Designer начинается с создания нового файла проекта и листа схемы командами File>New>Project>PCB Project и File>New>Schematic. Для сохранения проекта выполняется File>Save Project As, для схемы – File>Save.

Настройки во всех редакторах Altium Designer можно разделить на глобальные – относящиеся ко всем документам и локальные – относящиеся только к текущему документу. Настройки текущего документа устанавливаются на вкладке Design>Document Options. Она содержит три вкладки: Sheet Options, на которой задаются настройки листа (размер листа, шаги сеток, ориентация листа и т.д.), Parameters и Units, в которой задаются единицы измерения. Глобальные настройки находятся в меню: DXP>Preferences>Schematic.

Перед формированием схемы необходимо подключить библиотеки, в которых находятся компоненты схемы. Для поиска компонентов и подключения библиотек служит панель управления библиотеками Libraries, которая вызывается выбором вкладки в правой части окна Design Explorer. Если вкладка отсутствует, то панель можно вызвать через меню View> Workspace Panels>System>Library или через меню вызова панелей System>Libraries, расположенном в правом нижнем углу рабочего окна.

В верхней части окна панели имеются 3 вкладки: Project – библиотеки проекта, Installed – установленные библиотеки, Search Path – поисковая система. Для поиска компонентов ко всем доступным библиотекам независимо от того, подключены они или нет, служит кнопка Search панели Libraries.

Инструменты формирования электрической схемы сгруппированы в панели инструментов Wiring. Величину шага сетки можно изменить командой View>Grid>Set Snap Grid. После выбора необходимого компонента в библиотеке необходимо воспользоваться кнопкой Place.

Прежде чем зафиксировать компонент на плате, нажатием клавиши Tab следует открыть окно Component Properties, в котором информация о компоненте разбита на группы: Properties – основные свойства компонента, Library Link – ссылка на соответствующий библиотечный элемент, Graphical – параметры графического изображения, Parameters – атрибуты компонента, Models – модели компонента, Edit Pins – таблица выводов компонента.

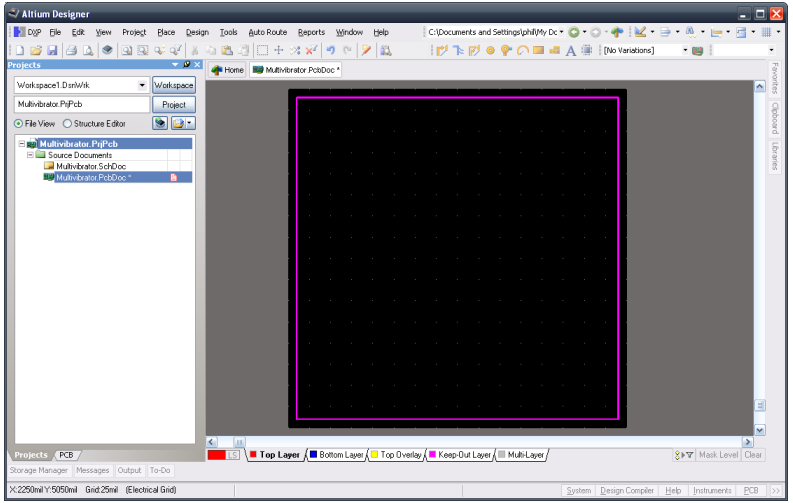


Рисунок 6.1 –Редактор печатных плат в Altium Designer

Соединение элементов электрическими цепями осуществляется командой Place>Wire. В процессе рисования цепей можно клавишами Shift + Space выбирать один из 4 режимов рисования: 90º, 45º, произвольный угол и режим Auto Wire (соединение двух выбранных точек по оптимальному маршруту).

Имя (метка) электрической цепи присваивается командой Place>Net Label.

Расстановка позиционных обозначений компонентов схемы выполняется командой Tools>Annotate Schematic. Выполнение команды приводит к появлению окна Annotate, в котором можно выбрать требуемые параметры расстановки.

В процессе компиляции обнаруживаются нарушения, ошибки в проекте (верификация проекта), создается отчет о корректности проекта, найденные нарушения помечаются на схеме и сопровождаются комментариями об их природе. Результат компиляции – отлаженный файл проекта, готовый к проектированию печатной платы. Процесс компиляции состоит из следующих этапов:

* настройка параметров проекта (схемы), которая заключается в задании правил проверки схемы (ERC – Electrical Rule Check). Командой Project >Project Options создается окно Options for Project, во вкладках которого и задаются правила проверки;
* выполнение компиляцию проекта командой Project>Compile PCB Project. Результаты компиляции будут показаны на панели Compiled, где будут описаны компоненты, цепи, выводы и др. Обнаруженные нарушения будут указаны на панели Messages. Двойным щелчком мыши на строке с ошибкой компиляции можно вызвать панель Compile Errors с подробным описанием ошибки. Двойной щелчок на значке приведет к появлению наглядного изображения этого элемента на схеме, остальная часть схемы будет маскирована;
* отладка схемы. Необходимо добиться, чтобы в списке нарушений в окне Messages не осталось ни одной ошибки (Fatal Error и Error). Рекомендуется отладку выполнять постепенно: исправить ошибку и снова провести компиляцию.

Редактор печатных плат предназначен для создания, редактирования и тестирования печатных плат, генерации файлов для изготовления фотошаблонов.

Для перемещения по чертежу можно использовать браузер Mini Viewer (прямоугольник из пунктирных линий Zoom Box показывает поле просмотра) путем его перетаскивания с помощью левой клавиши мыши (ЛКМ). Щелчком ЛКМ по кнопке Magnifier возможно осуществить перемещение курсора в вид лупы с изменением степени увеличения клавишей Space Bar (три возможных значения).

Панорамирование чертежа возможно выполнять с помощью четырех стрелок клавиатуры. Другой способ выполнять панорамирование – применить инструмент Slider Hand, который активизируется, если нажать и не отпускать правую кнопку мыши (ПКМ), после чего изображение в окне можно передвинуть. Просмотреть чертеж можно с помощью команд меню View.

Объекты, которые может обрабатывать редактор плат, делятся на примитивы и составные объекты. Под обработкой понимаются такие операции, как размещение объектов, выделение, копирование, перемещение, изменение, удаление и др. Разновидности объектов редактора плат: примитивы-графические объекты (линии, дуги, текстовые строки), примитивы-электрические объекты (проводники, контактные площадки, переходные отверстия, области металлизации), составные объекты, создаваемые пользователем (компоненты, полигоны), составные объекты, создаваемые системой (размеры, координатные метки).

Для идентификации объекта в окне редактора плат достаточно навести на него указатель мыши. Информация появится в виде текстовой строки, а также в строке состояния.

Для выделения объектов применяют два подхода: выделение фокусом и комплексное выделение. Выделение фокусом осуществляется щелчком ЛКМ по объекту, в результате объект становиться активным. После щелчка ЛКМ по прямоугольнику в его углах появятся 5 меток манипуляторов: в углах – метки для изменения размеров объекта, в центре – метка вращения. Выделенный объект можно удалить, нажав на клавиатуре клавишу Del. Удаление можно выполнить и с помощью меню командой Edit>Delete, затем указать удаляемые объекты. Если объекты накладываются друг на друга или расположены близко, то следует выполнить двойной щелчок ЛКМ, после чего на экране появится табличка со списком объектов. Щелчком ЛКМ выбирается нужный объект.

В редакторе плат системы Altium Designer проектируемая конструкция представляется в виде совокупности слоев. Все слои разбиты на следующие группы: Signal Layers, Internal, Mechanical, Mask, Silkscreen и Other Layers.

Signal Layers – сигнальные слои. Их может быть до 32 в многослойной печатной плате. Из них: Top – верхний слой, Mid – внутренние сигнальные слои, Bottom – нижний слой.

Internal Layers – экранные слои. Это могут быть внутренние слои питания и земли, металлизированные полигоны. Отображение форм на экранных слоях инверсное.

Mechanical Layers – механические слои, предназначенные для прорисовки вспомогательных элементов чертежа печатной платы, которые не должны быть на самой плате.

Mask Layers – слои паяных паст и защитных масок. Слои Top Solder и Bottom Solder предназначены для прорисовки масок, используемых при нанесении припоя на верхнюю и нижнюю стороны печатной платы. Слои Top Paste и Bottom Paste предназначены для прорисовки масок, используемых при нанесении паяльной пасты на верхнюю и нижнюю стороны печатной платы.

Silkscreen Layers – слои шелкографии. Слои Top Overlay и Bottom Overlay предназначены для нанесения рисунков и надписей, выполненных методом шелкографии, на верхнюю и нижнюю стороны печатной платы.

Other Layers – дополнительные слои, к которым относятся: Drill Guide – слой сверления. Здесь создается чертеж расположения центров всех отверстий на печатной плате, Multi-Layers – слой для размещения контактных площадок и переходных отверстий многослойных печатных плат, Keep Out – слой для задания областей, где разрешено размещение компонентов и проводников. Области Keep Out могут быть заданы для отдельных слоев. Для этого нужный контур необходимо разместить командой Place>Keep Out.

Активизация слоев осуществляется командой Design>Options> Document Options>Layers>включить переключатели отображения нужных слоев. В результате вкладка с именем слоя появится в нижней части окна редактора плат.

Первый шаг конструкторского проектирования печатной платы –создание заготовки чертежа платы, в которой заданы её границы и набор слоёв. Возможны следующие варианты получения заготовки чертежа

платы: вручную с использованием инструментов редактора плат; с помощью специализированного мастера PCB Wizard; с помощью мастера PCB Wizard можно в качестве заготовки использовать один из готовых шаблонов плат промышленных стандартов или ранее подготовленных разработчиком шаблонов; импорт из другой САПР.

Предварительно следует установить начало системы координат командой Edit>Origin >Set, по которой начало координат устанавливается в текущую позицию курсора.

Шаг сетки можно изменить в любой момент проектирования командой

Design>Options > Document Options или комбинацией горячих клавиш Ctrl + G. Задать величину шага сетки Electrical Grid можно командой Design>Options Document Options> Options. Включить или выключить электрическую сетку в процессе проектирования можно комбинацией горячих клавиш Shift + E. Сетка Component Grid помогает разработчику ориентироваться при размещении компонентов и ее действие аналогично сетке SnapGrid. Сетки Visible Grid облегчают ориентацию в чертеже.

Контур платы можно задать командой меню Design>Board Shape>Redefine Board, Define Board Cutout.

Крепежные отверстия устанавливаются как обычные контактные площадки командой Place>Pad, затем в свойствах указываются нулевые значения в параметрах формы контактной площадки (Size and Shape) и отключается металлизацию внутри отверстия (Plated). Проектные данные, которые передаются из редактора схем в редактор плат – это список электрических цепей. При этом извлекается информация о каждом компоненте и параметрах связанности из схемы электрической принципиальной, отыскивается соответствующее компоненту топологическое посадочное место (Footprint) в библиотеке элементов. Посадочное место размещается на чертеже платы с добавлением линий соединений.

Правила проектирования должны быть согласованы с ограничениями используемой технологии производства печатных плат и конструктивными ограничениями компонентов, применяемых в проекте. Настройка и редактирование правил проектирования может производиться вручную (команды меню Design>Rules) или с помощью мастера Rule Wizard. Каждое задаваемое правило имеет область действия (Scope) от всей платы до отдельного объекта.

Чтобы использовать разработанные правила в других последующих проектах, их можно записать в отдельный файл. Для этого в окне PCB Rules and Constraints нажать ПКМ в списке правил и выбрать Export Rules. Далее выбрать нужное правило и нажать кнопку OK.

САПР Altium Dеsigner содержит две программы автоматического размещения компонентов. Для интерактивного размещения используются в основном инструменты меню Tools>Component Placement.

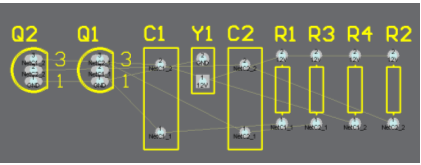


Рисунок 6.2 – компоненты и связи в Altium Designer

Предварительно следует настроить основные опции размещения (Tools > Preference > Options).

Некоторые компоненты, бывает необходимо зафиксировать на нужном месте. Для этого следует выполнить двойной щелчок мыши по компоненту и в появившемся диалоговом окне Component на вкладке Properties установить флажок Locked.

Основные параметры настройки интерактивной трассировки находятся в окне DXP > Preferences > PCB Editor > Interactive Routing и соответствуют установленным ранее правилам проектирования. Эти установки могут быть изменены в процессе прокладки трассы при нажатии клавиши Tab.

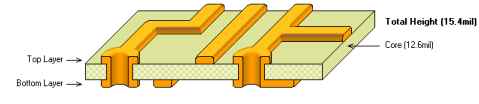


Рисунок 6.3 – проектирование металлизации отверстий в Altium Designer

В процессе прокладки трасс редактор печатных плат непрерывно контролирует выполнение правил проектирования (on-line DRC) и препятствует их нарушению.

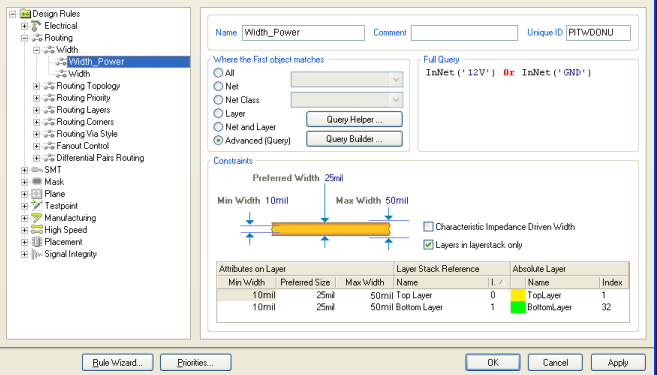


Рисунок 6.4 – настройка параметров печатных проводников в Altium Designer

Для увеличения числа доступных каналов трассировки количество контактных площадок компонентов, попадающих в узлы сетки Snap Grid, должно быть как можно большим. Проверить, находятся ли компоненты в сетке, можно с помощью команды меню Edit >Select>Off Grid Pads. Привязку всех компонентов к узлам сетки размещения можно выполнить с помощью команды меню Tools>Interactive Placement>Move to Grid, после чего на экране появится диалоговое окно, позволяющее пользователю установить параметры сетки.

Все неразведенные проводники представляются в слое Connection в виде тонких виртуальных линий связи (маршрутов From-To). Режим интерактивной трассировки включается командой Place>Interactive Routine либо соответствующей пиктограммой на панели инструментов. В результате указатель мыши примет вид креста. После выбора начального контакта трассы указатель примет вид восьмиугольника. Это признак того, что сработала электрическая сетка – произошел захват электрического объекта. После выбора ЛКМ очередной точки трассы система пунктиром прорисовывает предполагаемый следующий сегмент трассы. Прокладка трассы завершается нажатием ПКМ.

Полигонами называют области металлизации неправильной формы, которые могут состоять из одной или нескольких частей, соединенных с заданной цепью.

Область металлизации может быть создана на любом сигнальном слое. Сначала прямыми линиями или дугами задаются границы полигона, который впоследствии автоматически будет залит медью в соответствии с заранее определенными правилами проектирования. Границы заливки можно редактировать в любой момент работы над проектом. После изменения положения отдельных компонентов и проводников на плате можно выполнить повторную заливку полигона. Размещение полигонов выполняется с помощью команды меню Place>Polygon Pour, после чего появится диалоговое окно Polygon Pour, которое позволяет установить нужные параметры полигона. Изменения границ полигона выполняется командой Edit>Change>Polygon Vertices и последующим перемещением его вершин. В диалоговом окне Place > Polygon Pour выбирается вариант заливки полигона, характер соединение полигона с электрической цепью и прочие опции.

Верификация печатной платы заключается в проверке выполнения правил проектирования. Для верификации предназначен программный модуль Design Rule Checker (DRC). Запуск верификации печатной платы осуществляется нажатием кнопки Run Design Rule Clock в окне Design Rule Checker. В результате объекты, которые содержат обнаруженную ошибку, будут подсвечены соответствующим цветом.

После завершения разработки печатной платы её чертёж можно экспортировать для оформления по ЕСКД в САПР AutoCAD командой меню File>Export (формат dxf или dwg).

# 7 Технико-экономическое обоснование разработки и производства Коммуникационного контроллер с шифрованием данных для системы «умный дом»

**7.1 Характеристика изделия**

Разрабатываемый в дипломном проекте коммуникационный контроллер с шифрованием данных для системы «Умный дом» — это устройство управления, которое предназначено для приема данных с устройства системы, их шифрования и передачи на серверную часть системы. Внедрение нового контроллера позволит сократить расходы на производство.

Разрабатываемое устройство должно иметь меньшую стоимость чем аналогичные устройства и не уступать им по функционалу.

В главе ТЭО рассчитывается прогнозируемый экономический эффект от разработки и внедрения в производство устройства за 5 лет.

**7.2 Расчет затрат на производство системы**

Расчет затрат на материалы представлен в таблице 7.1.

Таблица 7.1 – Расчет затрат на материалы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование материала | Единица измерения | Норма | Цена за единицу, р. | Сумма, р. |
| расхода |
| Стеклотекстолит | кг | 0,10 | 3,00 | 0,3 |
| Спирт | л | 0,02 | 4,40 | 0,09 |
| Припой | кг | 0,05 | 70,00 | 3,5 |
| Лак | л | 0,02 | 40,00 | 0,8 |
| Клей | кг | 0,06 | 12,00 | 0,72 |
| Флюс | л | 0,06 | 20,00 | 1,2 |
| Всего | | | | 6,61 |
| Всего с учетом транспортных расходов (20%) | | | | 7,93 |

Расчет затрат по статье «Покупные комплектующие изделия, полуфабрикаты и услуги производственного характера» представлен в таблице 7.2.

Таблица 7.2 – Расчет затрат на комплектующие изделия и полуфабрикаты

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование комплектующего изделия или полуфабриката | Количество на единицу, шт. | Цена за единицу, р. | Сумма, р. |
| Микроконтроллер Atmega328P-AU | 1 | 12 | 12 |
| Резонатор кварцевый 16 МГц HC-49S | 1 | 0,66 | 0,66 |
| Конденсатор 0,22 пФ GRM1555C1ER22BA01D | 3 | 0,7 | 2,1 |
| Конденсатор 0,1 мкФ GRM21BR71H104K\*\* | 2 | 0,77 | 1,54 |
| Разъём 2pin 15EDGRC-3.5-02 | 1 | 1 | 1 |
| Разъём 4pin 15EDGRC-3.81-04 | 1 | 1 | 1 |
| Кнопка тактовая KLS7-TS6601 | 3 | 0,5 | 1,5 |
| Резистор 330 Ом 0,062 Вт SMD0402 1% | 10 | 0,1 | 1 |
| Резистор 4,7 кОм 0,125 Вт SMD0805 5% | 5 | 0,11 | 0,55 |
| Всего | | | 21,35 |
| Всего с учётом расходов на транспортировку и установку (10%) | | | 23,49 |

Произведем расчет затрат по статье «Основная заработная плата производственных рабочих». Расчёт основной заработной платы основных производственных рабочих (Зо) представлен в таблице 7.3.

По данным предприятия среднемесячная зарплата первого разряда составляет 380 руб. Часовая тарифная ставка первого разряда, при фонде рабочего времени 168 часов составит 2,26 рубля.

Таблица 7.3 ‒ Расчет основной заработной платы производственных рабочих

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вид работы  (операция) | Разряд работ | Часовая тарифная ставка руб./ч | Норма времени по операции,  норма/час | Прямая зарплата (расценка), руб. |
| 1 Заготовительные | 3 | 3,05 | 1,1 | 3,36 |
| 2 Монтажная | 6 | 4,30 | 2,5 | 10,74 |
| 3 Сборочная | 8 | 4,91 | 2,1 | 10,31 |
| Итого | | | | 24,41 |
| Премия, (40%) | | | | 9,76 |
| Основная заработная плата | | | | 34,17 |

Результаты расчета остальных статей затрат, себестоимости и отпускной цены представлены в таблице 7.4.

Таблица 7.4 – Расчет себестоимости и отпускной цены единицы продукции

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование статьи затрат | Условное обозначение | Значение, руб. | Примечание |
| 1 | 2 | 3 | 4 |
| Сырье и материалы | Рм | 7,93 | См. табл.7.1 |
| Покупные комплектующие изделия | Рк | 23,49 | См. табл.7.2 |
| Основная заработная плата производственных рабочих |  | 34,17 | См. табл.7.3 |
| Дополнительная заработная плата производственных рабочих |  | 3,42 | , =10% |
| Отчисления на социальные нужды (отчисления в фонд социальной защиты населения и обязательное страхование) |  | 13,01 | ,= 34,6% |
| Накладные расходы | Рн | 51,26 | Нн = 150% |
| Производственная себестоимость | Спр | 133,28 |  |
| Продолжение таблицы 7.4 | | | |
| 1 | 2 | 3 | 4 |
| Коммерческие расходы | Рком | 2,67 | Нком = 2% |
| Полная себестоимость | Сп | 135,94 | Сп = Спр+Рком |
| Плановая прибыль на единицу продукции | Пед | 54,38 | Нре = 40% |
| Отпускная цена | Цотп | 190,32 | Цотп = Сп+Пед |

**7.3 Расчёт чистой прибыли**

Чистая прибыль рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (7.1) |

В первый год будет произведено 500 изделий:

Пч1 = 500 ˖ 54,38 ˖(1-18/100) = 22294,34 р.

Так как объём производства по годам не изменяется, чистая прибыль по годам имеет одинаковое значение.

Пч = 1000 ˖ 54,38 ˖(1-18/100) = 44588,69 р.

**7.3 Расчёт инвестиций в производство нового изделия**

Инвестиции в производство нового изделия включают:

* 1. Инвестиции на разработку нового изделия (Иразр).
  2. Инвестиции в основной и оборотный капитал.

Годовая потребность в материалах определяется по формуле

|  |  |
| --- | --- |
| Пм = Рм ˖ N | (7.2) |

Пм = 1000 ˖ 7,93 = 7929,6 р.

Годовая потребность в комплектующих изделиях определяется по формуле

|  |  |
| --- | --- |
| Пк = Рк ˖ N | (7.3) |

Пк = 1000 ˖ 23,49 = 23485 р.

Инвестиции на разработку нового изделия согласно  
смете разработчика составляют 40000 р.

Иразр = 40000 р.

Инвестиции в прирост собственного оборотного капитала составляют  
30-40% от стоимости годовой потребности в материалах и комплектующих изделиях по формуле

|  |  |
| --- | --- |
| Иоб = (Пк+Пм) ˖ 0,5 | (7.4) |

Иоб = (7929,6+ 23485) ˖ 0,4 = 12565,84 р.

Таким образом, инвестиции в производство нового изделия составят

|  |  |
| --- | --- |
| И = Иразр + Иоб | (7.5) |

И = 12565,84+40000 = 54565,84 р.

**7.5 Расчет показателей эффективности системы**

При оценке эффективности инвестиционных проектов необходимо осуществить приведение затрат и результатов, полученных в разные периоды времени, к расчетному году, путем умножения затрат и результатов на коэффициент дисконтирования , который определяется следующим образом:

|  |  |
| --- | --- |
| , | (7.6) |

где‒требуемая норма дисконта, 0,12;

‒ порядковый номер года, затраты и результаты которого приводятся к расчетному году;

‒ расчетный год, в качестве расчетного года принимается год вложения инвестиций, = 1.

Таким образом, коэффициенты дисконтирования составят:

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

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

|  |  |
| --- | --- |
| , |  |

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

Расчет чистого дисконтированного дохода и срока окупаемости представлен в таблице 7.4.

Таблица 7.4 – Расчет чистого дисконтированного дохода и срока окупаемости инвестиций в производство программно-аппаратного комплекса, р.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование показателя | Усл. обоз. | По годам расчётного периода | | | |
| 2022 | 2023 | 2024 | 2025 |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Результат |  |  |  |  |  |
| 1 Прирост чистой прибыли | Пч | 22294,34 | 44588,69 | 44588,69 | 44588,69 |
| 2 Прирост результата | Рt | 22294,34 | 44588,69 | 44588,69 | 44588,69 |
| 3 Коэффициент дисконтирования | αt | 1,00 | 0,89 | 0,80 | 0,71 |
| 4 Результат с учётом фактора времени | Рtαt | 22294,34 | 39811,33 | 35545,83 | 31737,35 |
| Затраты (инвестиции) |  |  |  |  |  |
| 5 Инвестиции в разработку нового изделия |  | 45000,00 |  |  |  |
| Продолжение таблицы 7.4 | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 6 Инвестиции в собственный оборотный капитал |  | 12565,84 |  |  |  |
| 7 Инвестиции | И | 57565,84 | - | - | - |
| 8 Инвестиции с учётом фактора времени | Иtαt | 57565,84 | - | - | - |
| 9 Чистый дисконтированный доход по годам (п.4− п.6) | ЧДДt | -35271,50 | 39811,33 | 35545,83 | 31737,35 |
| 10 ЧДД нарастающим итогом | ЧДД | -35271,50 | 4539,83 | 40085,66 | 71823,01 |

Как видно, инвестиции на проектные работы и монтаж системы окупятся на второй год.

Рентабельность инвестиций (Ри) определяется по формуле

|  |  |
| --- | --- |
| , | (7.7) |

где  ‒среднегодовая величина чистой прибыли за расчетный период, руб., которая определяется по формуле

|  |  |
| --- | --- |
| , | (7.8) |

где ‒ чистая прибыль, полученная в году *t*, руб.

Пчср = (22294,34+39811,33+35545,83+31737,35)/4 = 32347,21 р.

Рентабельность инвестиций составит:

Ри = (32347,21/57565,84)∙100 = 56,19 %

В процессе технико-экономического обоснования эффективности внедрения коммуникационного контроллера с шифрованием данных для системы умный дом получены следующие результаты:

1. Интегральный экономический эффект от внедрения в производство изделия за четыре года составил 71823,01 руб.;

2. Инвестиции окупятся на второй год с учетом фактора времени;

3. Рентабельность проекта составит 56,19 %.

Таким образом, разработка и внедрение коммуникационного контроллера с шифрованием данных для системы умный дом является эффективными для предприятия.

# 8 Анализ результатов проектирования

В дипломном проекте разработана аппаратная и программная части коммуникационного контроллера с шифрованием данных для системы умный дом. Предлагаемая система представляет собой совокупность устройств, предназначенных для обеспечения комфорта, безопасности, а также ресурсосбережения дома. Система позволяет легко наращивать функционал аппаратной части, а также быстро модифицировать программную составляющую, что является, несомненно, положительными аспектами, при построении систем такого рода. Кроме того, к плюсам системы относится ее относительно малая стоимость по сравнению с профессиональными коммерческими аналогами. Также в будущем планируется реализовать автоматическое проветривание помещений на основе датчиков присутствия газов MQ-9 и сервоприводов для открытия/закрытия окон, если в помещении поднимается концентрация углекислого газа, то сервопривод открывает окно, когда уровень концентрации углекислого газа приходит в норму, тогда сервопривод закрывает окно. В планах также реализация функции оповещения при протечке воды, что в многоэтажных домах будет эффективно работать, при затоплениях.

# Заключение

В ходе проектирования конструкции устройства, учитывая последние разработки аналогичных приборов, были разработаны структурная схема устройства, функциональная схема и схема электрическая принципиальная.

При выполнении данного дипломного проекта широко использованы возможности вычислительной техники и пакеты систем автоматизированного проектирования современного уровня. К таким пакетам относится Altium Designer 21, пакет Microsoft Office и другие.

Спроектированное микропроцессорного устройства дистанционного управления объектом удовлетворяет техническим требованиям задания на дипломное проектирование, пояснительная записка содержит все основные разделы.

К достоинствам устройства можно отнести простоту и компактность, которая обусловлена использованием в устройстве минимального количества электронных компонентов. Конструкция устройства может служить базовой для создания микропроцессорного устройства дистанционного управления объектом.

# Список использованных источников

[1] Micromax [Электронный ресурс] - Режим доступа: https://www.micromax.ru/solution/theory-practice/articles/2160/

[2] Wi-Fi модуль ESP-01 [Электронный ресурс] - Режим доступа: https://3d-diy.ru/wiki/arduino-moduli/wi-fi-modul-esp-01/

[3] Электронные патентно-информационные ресурсы и базы данных: портал [Электронный ресурс]. - Режим доступа: [http://www.fips.ru](http://www.fips.ru/ensite/dbs/ipc.htm).

[4] Проектирование и производство РЭС. Дипломное проектирование: Учеб. Пособие/ А.П. Достанко, В.М. Бондарик, С.В. Бордусов [и др.]; Под общ. ред. А.П. Достанко. – М.:БГУИР, 2002. – 204c.

[5] Иди, Ф. Сетевой и межсетевой обмен данными с микроконтроллерами / Ф. Иди. - М.: Додэка XXI, 2007. – 376 c.

[6] Сбо­рочно-монтажные процессы: учебно-методическое пособие / В.Л. Ланин, А. А. Костюкович, А. П. Достанко, А. А. Хмыль. – Мн.: БГУИР, 2008. – 133c.

[7] Технология радиоэлектронных устройств и автоматизация производства: Учебник/ А.П. Достанко, В.Л. Ланин, А.А. Хмыль, Л.П. Ануфриев. – Минск: Высш. школа, 2002.–513 с.

[8] Обеспечение тепловых режимов при конструировании РЭА/ Л.Л.Роткоп, Ю.Е Спокойный; - М: Сов. радио, 1976 – 233c.

[9] ГОСТ 30631-99 Общие требования к машинам, приборам и другим техническим Изделиям в части стойкости к механическим внешним воздействующим факторам.

[10] Справочник конструктора - приборостроителя. Проектирование. Основные нормы / В.Л.Соломахо, Р.И.Томилин,Б.В. Цитович– Мн.: Выcш.шк., 1988.

[11] Хофманн, М. Микроконтроллеры для начинающих / М. Хофманн. - СПб.: BHV, 2013. - 304 c.

[12] ГОСТ 2.413-72 ЕСКД. Правила выполнения конструкторской до­кументации изделий, изготовляемых с применением электрического мон­тажа; Введен 01.07.1973.

[13]ГОСТ 2.701-84 ЕСКД. Схемы. Виды и типы. Общие требования к выполнению; Введен 01.07.1985.

[14] ГОСТ 2.417-91 ЕСКД Платы печатные. Правила выполнения чертежей.- Взамен ГОСТ 2.417-78; Введен 01.01.2002.

[15] ГОСТ 2.104-2006ЕСКДОсновные надписи.- Взамен ГОСТ 2.104-68; Введен 01.08.2007.

[16]СТБ 1014-95 Изделия машиностроения. Детали. Общие техниче­ские условия; Введен 01.01.1998.

# Приложение А

**(обязательное)**

**Листинг программы микроконтроллера на языке С**

//

//file belt.h

//

/\*

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

Шифрование в режиме счетчика

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

\*/

#define B\_PER\_W 16

#define O\_PER\_W (B\_PER\_W / 8)

typedef unsigned short WORD;

typedef unsigned long u32;

typedef signed long i32;

typedef unsigned char u8;

typedef signed char i8;

typedef u8 octet;

typedef unsigned int size\_t;

typedef struct

{

u32 key[8]; //форматированный ключ

u32 ctr[4]; //счетчик

octet block[16]; //блок гаммы

size\_t reserved; //резерв октетов гаммы

} belt\_ctr\_st;

/\*

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

Ускорители

Реализованы быстрые операции над блоками и полублоками belt. Блок

представляется либо как [16]octet, либо как [4]u32,

либо как [W\_OF\_B(128)]word.

Суффикс U32 в именах макросов и функций означает, что данные интерпретируются

как массив u32. Суффикс W означает, что данные интерпретируются как

массив word.

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

\*/

#define beltBlockIncU32(block)\

if ((((u32\*)(block))[0] += 1) == 0 &&\

(((u32\*)(block))[1] += 1) == 0 &&\

(((u32\*)(block))[2] += 1) == 0)\

((u32\*)(block))[3] += 1\

#define beltBlockCopy(dest, src)\

((WORD\*)(dest))[0] = ((const WORD\*)(src))[0],\

((WORD\*)(dest))[1] = ((const WORD\*)(src))[1],\

((WORD\*)(dest))[2] = ((const WORD\*)(src))[2],\

((WORD\*)(dest))[3] = ((const WORD\*)(src))[3]\

#define beltBlockXor2(dest, src)\

((WORD\*)(dest))[0] ^= ((const WORD\*)(src))[0],\

((WORD\*)(dest))[1] ^= ((const WORD\*)(src))[1],\

((WORD\*)(dest))[2] ^= ((const WORD\*)(src))[2],\

((WORD\*)(dest))[3] ^= ((const WORD\*)(src))[3]\

/\*!

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

Блоб -- объект в памяти определенного размера. В функциях работы с блобами

используются их дескрипторы -- "умные" указатели. С дескрипторами можно

работать как с обычными указателями, т.е. использовать их в функциях типа

memcpy, memset. Дополнительно по указателю можно определить размер блоба.

Реализация работы с блобами может быть платформенно-зависимой.

Реализация должна гарантировать защиту содержимого блобов от утечек,

например, через файл подкачки. Поэтому в блобах рекомендуется размещать

ключи и другие критические объекты.

В функциях работы с блобами дескрипторы входных блобов корректны.

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

\*/

// память для блобов выделяется страницами

#define BLOB\_PAGE\_SIZE 1024

// требуется страниц

#define blobPageCount(size)\

(((size) + sizeof(size\_t) + BLOB\_PAGE\_SIZE - 1) / BLOB\_PAGE\_SIZE)

// требуется памяти на страницах

#define blobActualSize(size)\

(blobPageCount(size) \* BLOB\_PAGE\_SIZE)

// heap-указатель для блоба

#define blobPtrOf(blob) ((size\_t\*)blob - 1)

// размер блоба

#define blobSizeOf(blob) (\*blobPtrOf(blob))

// страничный размер блоба

#define blobActualSizeOf(blob) (blobActualSize(blobSizeOf(blob)))

// блоб для heap-указателя

#define blobValueOf(ptr) ((blob\_t)((size\_t\*)ptr + 1))

// дескриптор блоба

typedef void\* blob\_t;

/\* Инициализация шифрования в режиме CTR

По ключу [len]key и синхропосылке iv в state формируются

структуры данных, необходимые для шифрования в режиме CTR.

len == 16 || len == 24 || len == 32.

По адресу state зарезервировано beltCTR\_keep() октетов.

Буферы key и state могут пересекаться.

\*/

void beltCTRStart(

void\* state, //[out] состояние

const octet key[], //[in] ключ

size\_t len, //[in] длина ключа в октетах

const octet iv[16] //[in] синхропосылка

);

/\* Зашифрование фрагмента в режиме CTR

Буфер [count]buf зашифровывается в режиме CTR на ключе, размещенном

в state.

beltCTRStart() < beltCTRStepE()\*.

\*/

void beltCTRStepE(

void\* buf, //[in/out] открытый текст / шифртекст

size\_t count, //[in] число октетов текста

void\* state //[in/out] состояние

);

/\* Расшифрование фрагмента в режиме CTR

Зашифрование в режиме CTR не отличается от расшифрования.

\*/

#define beltCTRStepD beltCTRStepE

/\* Шифрование в режиме CTR

Буфер [count]src зашифровывается или расшифровывается на ключе

[len]key с использованием синхропосылки iv. Результат шифрования

размещается в буфере [count]dest.

{ERR\_BAD\_INPUT} len == 16 || len == 24 || len == 32.

ERR\_OK, если шифрование завершено успешно, и код ошибки

в противном случае.

Буферы могут пересекаться.

\*/

//

//file belt.cpp

//

/\*

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

STB 34.101.31 (belt): CTR encryption

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

\*/

#include "belt.h"

#include <string.h>

#include <avr/pgmspace.h>

/\*

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

Загрузка

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

\*/

void u32From(u32 dest[], const void\* src, size\_t count)

{

memmove(dest, src, count);

if (count % 4)

memset((octet\*)dest + count, 0, 4 - count % 4);

}

/\*

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

Расширение ключа

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

\*/

void beltKeyExpand(u32 key\_[8], const octet key[], size\_t len)

{

u32From(key\_, key, len);

if (len == 16)

{

key\_[4] = key\_[0];

key\_[5] = key\_[1];

key\_[6] = key\_[2];

key\_[7] = key\_[3];

}

else if (len == 24)

{

key\_[6] = key\_[0] ^ key\_[1] ^ key\_[2];

key\_[7] = key\_[3] ^ key\_[4] ^ key\_[5];

}

}

/\*

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

H-блок

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

\*/

/\*

static const octet H[256] = {

0xB1,0x94,0xBA,0xC8,0x0A,0x08,0xF5,0x3B,0x36,0x6D,0x00,0x8E,0x58,0x4A,0x5D,0xE4,

0x85,0x04,0xFA,0x9D,0x1B,0xB6,0xC7,0xAC,0x25,0x2E,0x72,0xC2,0x02,0xFD,0xCE,0x0D,

0x5B,0xE3,0xD6,0x12,0x17,0xB9,0x61,0x81,0xFE,0x67,0x86,0xAD,0x71,0x6B,0x89,0x0B,

0x5C,0xB0,0xC0,0xFF,0x33,0xC3,0x56,0xB8,0x35,0xC4,0x05,0xAE,0xD8,0xE0,0x7F,0x99,

0xE1,0x2B,0xDC,0x1A,0xE2,0x82,0x57,0xEC,0x70,0x3F,0xCC,0xF0,0x95,0xEE,0x8D,0xF1,

0xC1,0xAB,0x76,0x38,0x9F,0xE6,0x78,0xCA,0xF7,0xC6,0xF8,0x60,0xD5,0xBB,0x9C,0x4F,

0xF3,0x3C,0x65,0x7B,0x63,0x7C,0x30,0x6A,0xDD,0x4E,0xA7,0x79,0x9E,0xB2,0x3D,0x31,

0x3E,0x98,0xB5,0x6E,0x27,0xD3,0xBC,0xCF,0x59,0x1E,0x18,0x1F,0x4C,0x5A,0xB7,0x93,

0xE9,0xDE,0xE7,0x2C,0x8F,0x0C,0x0F,0xA6,0x2D,0xDB,0x49,0xF4,0x6F,0x73,0x96,0x47,

0x06,0x07,0x53,0x16,0xED,0x24,0x7A,0x37,0x39,0xCB,0xA3,0x83,0x03,0xA9,0x8B,0xF6,

0x92,0xBD,0x9B,0x1C,0xE5,0xD1,0x41,0x01,0x54,0x45,0xFB,0xC9,0x5E,0x4D,0x0E,0xF2,

0x68,0x20,0x80,0xAA,0x22,0x7D,0x64,0x2F,0x26,0x87,0xF9,0x34,0x90,0x40,0x55,0x11,

0xBE,0x32,0x97,0x13,0x43,0xFC,0x9A,0x48,0xA0,0x2A,0x88,0x5F,0x19,0x4B,0x09,0xA1,

0x7E,0xCD,0xA4,0xD0,0x15,0x44,0xAF,0x8C,0xA5,0x84,0x50,0xBF,0x66,0xD2,0xE8,0x8A,

0xA2,0xD7,0x46,0x52,0x42,0xA8,0xDF,0xB3,0x69,0x74,0xC5,0x51,0xEB,0x23,0x29,0x21,

0xD4,0xEF,0xD9,0xB4,0x3A,0x62,0x28,0x75,0x91,0x14,0x10,0xEA,0x77,0x6C,0xDA,0x1D,

};

const octet\* beltH()

{

return H;

}

\*/

/\*

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

Расширенные H-блоки

Описание построено с помощью функции:

void beltExtendBoxes()

{

unsigned r, x;

u32 y;

for (r = 5; r < 32; r += 8)

{

printf("static const u32 H%u[256] = {", r);

for (x = 0; x < 256; x++)

y = H[x],

y = y << r | y >> (32 - r),

printf(x % 8 ? "0x%08X," : "\n\t0x%08X,", y);

printf("\n};\n");

}

}

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

\*/

static const u32 H5[256] PROGMEM = {

0x00001620,0x00001280,0x00001740,0x00001900,0x00000140,0x00000100,0x00001EA0,0x00000760,

0x000006C0,0x00000DA0,0x00000000,0x000011C0,0x00000B00,0x00000940,0x00000BA0,0x00001C80,

0x000010A0,0x00000080,0x00001F40,0x000013A0,0x00000360,0x000016C0,0x000018E0,0x00001580,

0x000004A0,0x000005C0,0x00000E40,0x00001840,0x00000040,0x00001FA0,0x000019C0,0x000001A0,

0x00000B60,0x00001C60,0x00001AC0,0x00000240,0x000002E0,0x00001720,0x00000C20,0x00001020,

0x00001FC0,0x00000CE0,0x000010C0,0x000015A0,0x00000E20,0x00000D60,0x00001120,0x00000160,

0x00000B80,0x00001600,0x00001800,0x00001FE0,0x00000660,0x00001860,0x00000AC0,0x00001700,

0x000006A0,0x00001880,0x000000A0,0x000015C0,0x00001B00,0x00001C00,0x00000FE0,0x00001320,

0x00001C20,0x00000560,0x00001B80,0x00000340,0x00001C40,0x00001040,0x00000AE0,0x00001D80,

0x00000E00,0x000007E0,0x00001980,0x00001E00,0x000012A0,0x00001DC0,0x000011A0,0x00001E20,

0x00001820,0x00001560,0x00000EC0,0x00000700,0x000013E0,0x00001CC0,0x00000F00,0x00001940,

0x00001EE0,0x000018C0,0x00001F00,0x00000C00,0x00001AA0,0x00001760,0x00001380,0x000009E0,

0x00001E60,0x00000780,0x00000CA0,0x00000F60,0x00000C60,0x00000F80,0x00000600,0x00000D40,

0x00001BA0,0x000009C0,0x000014E0,0x00000F20,0x000013C0,0x00001640,0x000007A0,0x00000620,

0x000007C0,0x00001300,0x000016A0,0x00000DC0,0x000004E0,0x00001A60,0x00001780,0x000019E0,

0x00000B20,0x000003C0,0x00000300,0x000003E0,0x00000980,0x00000B40,0x000016E0,0x00001260,

0x00001D20,0x00001BC0,0x00001CE0,0x00000580,0x000011E0,0x00000180,0x000001E0,0x000014C0,

0x000005A0,0x00001B60,0x00000920,0x00001E80,0x00000DE0,0x00000E60,0x000012C0,0x000008E0,

0x000000C0,0x000000E0,0x00000A60,0x000002C0,0x00001DA0,0x00000480,0x00000F40,0x000006E0,

0x00000720,0x00001960,0x00001460,0x00001060,0x00000060,0x00001520,0x00001160,0x00001EC0,

0x00001240,0x000017A0,0x00001360,0x00000380,0x00001CA0,0x00001A20,0x00000820,0x00000020,

0x00000A80,0x000008A0,0x00001F60,0x00001920,0x00000BC0,0x000009A0,0x000001C0,0x00001E40,

0x00000D00,0x00000400,0x00001000,0x00001540,0x00000440,0x00000FA0,0x00000C80,0x000005E0,

0x000004C0,0x000010E0,0x00001F20,0x00000680,0x00001200,0x00000800,0x00000AA0,0x00000220,

0x000017C0,0x00000640,0x000012E0,0x00000260,0x00000860,0x00001F80,0x00001340,0x00000900,

0x00001400,0x00000540,0x00001100,0x00000BE0,0x00000320,0x00000960,0x00000120,0x00001420,

0x00000FC0,0x000019A0,0x00001480,0x00001A00,0x000002A0,0x00000880,0x000015E0,0x00001180,

0x000014A0,0x00001080,0x00000A00,0x000017E0,0x00000CC0,0x00001A40,0x00001D00,0x00001140,

0x00001440,0x00001AE0,0x000008C0,0x00000A40,0x00000840,0x00001500,0x00001BE0,0x00001660,

0x00000D20,0x00000E80,0x000018A0,0x00000A20,0x00001D60,0x00000460,0x00000520,0x00000420,

0x00001A80,0x00001DE0,0x00001B20,0x00001680,0x00000740,0x00000C40,0x00000500,0x00000EA0,

0x00001220,0x00000280,0x00000200,0x00001D40,0x00000EE0,0x00000D80,0x00001B40,0x000003A0,

};

static const u32 H13[256] PROGMEM = {

0x00162000,0x00128000,0x00174000,0x00190000,0x00014000,0x00010000,0x001EA000,0x00076000,

0x0006C000,0x000DA000,0x00000000,0x0011C000,0x000B0000,0x00094000,0x000BA000,0x001C8000,

0x0010A000,0x00008000,0x001F4000,0x0013A000,0x00036000,0x0016C000,0x0018E000,0x00158000,

0x0004A000,0x0005C000,0x000E4000,0x00184000,0x00004000,0x001FA000,0x0019C000,0x0001A000,

0x000B6000,0x001C6000,0x001AC000,0x00024000,0x0002E000,0x00172000,0x000C2000,0x00102000,

0x001FC000,0x000CE000,0x0010C000,0x0015A000,0x000E2000,0x000D6000,0x00112000,0x00016000,

0x000B8000,0x00160000,0x00180000,0x001FE000,0x00066000,0x00186000,0x000AC000,0x00170000,

0x0006A000,0x00188000,0x0000A000,0x0015C000,0x001B0000,0x001C0000,0x000FE000,0x00132000,

0x001C2000,0x00056000,0x001B8000,0x00034000,0x001C4000,0x00104000,0x000AE000,0x001D8000,

0x000E0000,0x0007E000,0x00198000,0x001E0000,0x0012A000,0x001DC000,0x0011A000,0x001E2000,

0x00182000,0x00156000,0x000EC000,0x00070000,0x0013E000,0x001CC000,0x000F0000,0x00194000,

0x001EE000,0x0018C000,0x001F0000,0x000C0000,0x001AA000,0x00176000,0x00138000,0x0009E000,

0x001E6000,0x00078000,0x000CA000,0x000F6000,0x000C6000,0x000F8000,0x00060000,0x000D4000,

0x001BA000,0x0009C000,0x0014E000,0x000F2000,0x0013C000,0x00164000,0x0007A000,0x00062000,

0x0007C000,0x00130000,0x0016A000,0x000DC000,0x0004E000,0x001A6000,0x00178000,0x0019E000,

0x000B2000,0x0003C000,0x00030000,0x0003E000,0x00098000,0x000B4000,0x0016E000,0x00126000,

0x001D2000,0x001BC000,0x001CE000,0x00058000,0x0011E000,0x00018000,0x0001E000,0x0014C000,

0x0005A000,0x001B6000,0x00092000,0x001E8000,0x000DE000,0x000E6000,0x0012C000,0x0008E000,

0x0000C000,0x0000E000,0x000A6000,0x0002C000,0x001DA000,0x00048000,0x000F4000,0x0006E000,

0x00072000,0x00196000,0x00146000,0x00106000,0x00006000,0x00152000,0x00116000,0x001EC000,

0x00124000,0x0017A000,0x00136000,0x00038000,0x001CA000,0x001A2000,0x00082000,0x00002000,

0x000A8000,0x0008A000,0x001F6000,0x00192000,0x000BC000,0x0009A000,0x0001C000,0x001E4000,

0x000D0000,0x00040000,0x00100000,0x00154000,0x00044000,0x000FA000,0x000C8000,0x0005E000,

0x0004C000,0x0010E000,0x001F2000,0x00068000,0x00120000,0x00080000,0x000AA000,0x00022000,

0x0017C000,0x00064000,0x0012E000,0x00026000,0x00086000,0x001F8000,0x00134000,0x00090000,

0x00140000,0x00054000,0x00110000,0x000BE000,0x00032000,0x00096000,0x00012000,0x00142000,

0x000FC000,0x0019A000,0x00148000,0x001A0000,0x0002A000,0x00088000,0x0015E000,0x00118000,

0x0014A000,0x00108000,0x000A0000,0x0017E000,0x000CC000,0x001A4000,0x001D0000,0x00114000,

0x00144000,0x001AE000,0x0008C000,0x000A4000,0x00084000,0x00150000,0x001BE000,0x00166000,

0x000D2000,0x000E8000,0x0018A000,0x000A2000,0x001D6000,0x00046000,0x00052000,0x00042000,

0x001A8000,0x001DE000,0x001B2000,0x00168000,0x00074000,0x000C4000,0x00050000,0x000EA000,

0x00122000,0x00028000,0x00020000,0x001D4000,0x000EE000,0x000D8000,0x001B4000,0x0003A000,

};

static const u32 H21[256] PROGMEM = {

0x16200000,0x12800000,0x17400000,0x19000000,0x01400000,0x01000000,0x1EA00000,0x07600000,

0x06C00000,0x0DA00000,0x00000000,0x11C00000,0x0B000000,0x09400000,0x0BA00000,0x1C800000,

0x10A00000,0x00800000,0x1F400000,0x13A00000,0x03600000,0x16C00000,0x18E00000,0x15800000,

0x04A00000,0x05C00000,0x0E400000,0x18400000,0x00400000,0x1FA00000,0x19C00000,0x01A00000,

0x0B600000,0x1C600000,0x1AC00000,0x02400000,0x02E00000,0x17200000,0x0C200000,0x10200000,

0x1FC00000,0x0CE00000,0x10C00000,0x15A00000,0x0E200000,0x0D600000,0x11200000,0x01600000,

0x0B800000,0x16000000,0x18000000,0x1FE00000,0x06600000,0x18600000,0x0AC00000,0x17000000,

0x06A00000,0x18800000,0x00A00000,0x15C00000,0x1B000000,0x1C000000,0x0FE00000,0x13200000,

0x1C200000,0x05600000,0x1B800000,0x03400000,0x1C400000,0x10400000,0x0AE00000,0x1D800000,

0x0E000000,0x07E00000,0x19800000,0x1E000000,0x12A00000,0x1DC00000,0x11A00000,0x1E200000,

0x18200000,0x15600000,0x0EC00000,0x07000000,0x13E00000,0x1CC00000,0x0F000000,0x19400000,

0x1EE00000,0x18C00000,0x1F000000,0x0C000000,0x1AA00000,0x17600000,0x13800000,0x09E00000,

0x1E600000,0x07800000,0x0CA00000,0x0F600000,0x0C600000,0x0F800000,0x06000000,0x0D400000,

0x1BA00000,0x09C00000,0x14E00000,0x0F200000,0x13C00000,0x16400000,0x07A00000,0x06200000,

0x07C00000,0x13000000,0x16A00000,0x0DC00000,0x04E00000,0x1A600000,0x17800000,0x19E00000,

0x0B200000,0x03C00000,0x03000000,0x03E00000,0x09800000,0x0B400000,0x16E00000,0x12600000,

0x1D200000,0x1BC00000,0x1CE00000,0x05800000,0x11E00000,0x01800000,0x01E00000,0x14C00000,

0x05A00000,0x1B600000,0x09200000,0x1E800000,0x0DE00000,0x0E600000,0x12C00000,0x08E00000,

0x00C00000,0x00E00000,0x0A600000,0x02C00000,0x1DA00000,0x04800000,0x0F400000,0x06E00000,

0x07200000,0x19600000,0x14600000,0x10600000,0x00600000,0x15200000,0x11600000,0x1EC00000,

0x12400000,0x17A00000,0x13600000,0x03800000,0x1CA00000,0x1A200000,0x08200000,0x00200000,

0x0A800000,0x08A00000,0x1F600000,0x19200000,0x0BC00000,0x09A00000,0x01C00000,0x1E400000,

0x0D000000,0x04000000,0x10000000,0x15400000,0x04400000,0x0FA00000,0x0C800000,0x05E00000,

0x04C00000,0x10E00000,0x1F200000,0x06800000,0x12000000,0x08000000,0x0AA00000,0x02200000,

0x17C00000,0x06400000,0x12E00000,0x02600000,0x08600000,0x1F800000,0x13400000,0x09000000,

0x14000000,0x05400000,0x11000000,0x0BE00000,0x03200000,0x09600000,0x01200000,0x14200000,

0x0FC00000,0x19A00000,0x14800000,0x1A000000,0x02A00000,0x08800000,0x15E00000,0x11800000,

0x14A00000,0x10800000,0x0A000000,0x17E00000,0x0CC00000,0x1A400000,0x1D000000,0x11400000,

0x14400000,0x1AE00000,0x08C00000,0x0A400000,0x08400000,0x15000000,0x1BE00000,0x16600000,

0x0D200000,0x0E800000,0x18A00000,0x0A200000,0x1D600000,0x04600000,0x05200000,0x04200000,

0x1A800000,0x1DE00000,0x1B200000,0x16800000,0x07400000,0x0C400000,0x05000000,0x0EA00000,

0x12200000,0x02800000,0x02000000,0x1D400000,0x0EE00000,0x0D800000,0x1B400000,0x03A00000,

};

static const u32 H29[256] PROGMEM = {

0x20000016,0x80000012,0x40000017,0x00000019,0x40000001,0x00000001,0xA000001E,0x60000007,

0xC0000006,0xA000000D,0x00000000,0xC0000011,0x0000000B,0x40000009,0xA000000B,0x8000001C,

0xA0000010,0x80000000,0x4000001F,0xA0000013,0x60000003,0xC0000016,0xE0000018,0x80000015,

0xA0000004,0xC0000005,0x4000000E,0x40000018,0x40000000,0xA000001F,0xC0000019,0xA0000001,

0x6000000B,0x6000001C,0xC000001A,0x40000002,0xE0000002,0x20000017,0x2000000C,0x20000010,

0xC000001F,0xE000000C,0xC0000010,0xA0000015,0x2000000E,0x6000000D,0x20000011,0x60000001,

0x8000000B,0x00000016,0x00000018,0xE000001F,0x60000006,0x60000018,0xC000000A,0x00000017,

0xA0000006,0x80000018,0xA0000000,0xC0000015,0x0000001B,0x0000001C,0xE000000F,0x20000013,

0x2000001C,0x60000005,0x8000001B,0x40000003,0x4000001C,0x40000010,0xE000000A,0x8000001D,

0x0000000E,0xE0000007,0x80000019,0x0000001E,0xA0000012,0xC000001D,0xA0000011,0x2000001E,

0x20000018,0x60000015,0xC000000E,0x00000007,0xE0000013,0xC000001C,0x0000000F,0x40000019,

0xE000001E,0xC0000018,0x0000001F,0x0000000C,0xA000001A,0x60000017,0x80000013,0xE0000009,

0x6000001E,0x80000007,0xA000000C,0x6000000F,0x6000000C,0x8000000F,0x00000006,0x4000000D,

0xA000001B,0xC0000009,0xE0000014,0x2000000F,0xC0000013,0x40000016,0xA0000007,0x20000006,

0xC0000007,0x00000013,0xA0000016,0xC000000D,0xE0000004,0x6000001A,0x80000017,0xE0000019,

0x2000000B,0xC0000003,0x00000003,0xE0000003,0x80000009,0x4000000B,0xE0000016,0x60000012,

0x2000001D,0xC000001B,0xE000001C,0x80000005,0xE0000011,0x80000001,0xE0000001,0xC0000014,

0xA0000005,0x6000001B,0x20000009,0x8000001E,0xE000000D,0x6000000E,0xC0000012,0xE0000008,

0xC0000000,0xE0000000,0x6000000A,0xC0000002,0xA000001D,0x80000004,0x4000000F,0xE0000006,

0x20000007,0x60000019,0x60000014,0x60000010,0x60000000,0x20000015,0x60000011,0xC000001E,

0x40000012,0xA0000017,0x60000013,0x80000003,0xA000001C,0x2000001A,0x20000008,0x20000000,

0x8000000A,0xA0000008,0x6000001F,0x20000019,0xC000000B,0xA0000009,0xC0000001,0x4000001E,

0x0000000D,0x00000004,0x00000010,0x40000015,0x40000004,0xA000000F,0x8000000C,0xE0000005,

0xC0000004,0xE0000010,0x2000001F,0x80000006,0x00000012,0x00000008,0xA000000A,0x20000002,

0xC0000017,0x40000006,0xE0000012,0x60000002,0x60000008,0x8000001F,0x40000013,0x00000009,

0x00000014,0x40000005,0x00000011,0xE000000B,0x20000003,0x60000009,0x20000001,0x20000014,

0xC000000F,0xA0000019,0x80000014,0x0000001A,0xA0000002,0x80000008,0xE0000015,0x80000011,

0xA0000014,0x80000010,0x0000000A,0xE0000017,0xC000000C,0x4000001A,0x0000001D,0x40000011,

0x40000014,0xE000001A,0xC0000008,0x4000000A,0x40000008,0x00000015,0xE000001B,0x60000016,

0x2000000D,0x8000000E,0xA0000018,0x2000000A,0x6000001D,0x60000004,0x20000005,0x20000004,

0x8000001A,0xE000001D,0x2000001B,0x80000016,0x40000007,0x4000000C,0x00000005,0xA000000E,

0x20000012,0x80000002,0x00000002,0x4000001D,0xE000000E,0x8000000D,0x4000001B,0xA0000003,

};

/\*

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

G-блоки

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

\*/

#define G5(x)\

pgm\_read\_dword\_near(H5 + ((x) & 255)) ^ pgm\_read\_dword\_near(H13 + ((x) >> 8 & 255)) ^ pgm\_read\_dword\_near(H21 + ((x) >> 16 & 255)) ^ pgm\_read\_dword\_near(H29 + ((x) >> 24))

#define G13(x)\

pgm\_read\_dword\_near(H13 + ((x) & 255)) ^ pgm\_read\_dword\_near(H21 + ((x) >> 8 & 255)) ^ pgm\_read\_dword\_near(H29 + ((x) >> 16 & 255)) ^ pgm\_read\_dword\_near(H5 + ((x) >> 24))

#define G21(x)\

pgm\_read\_dword\_near(H21 + ((x) & 255)) ^ pgm\_read\_dword\_near(H29 + ((x) >> 8 & 255)) ^ pgm\_read\_dword\_near(H5 + ((x) >> 16 & 255)) ^ pgm\_read\_dword\_near(H13 + ((x) >> 24))

/\*

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

Тактовая подстановка

Макрос R реализует шаги 2.1-2.9 алгоритмов зашифрования и расшифрования.

На шагах 2.4-2.6 дополнительный регистр е не используется.

Нужные данные сохраняются в регистрах b и c.

Параметр-макрос subkey задает порядок использования тактовых ключей:

порядок subkey = subkey\_e используется при зашифровании и расшифровании

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

\*/

#define R(a, b, c, d, K, i, subkey)\

\*b ^= G5(\*a + subkey(K, i, 0));\

\*c ^= G21(\*d + subkey(K, i, 1));\

\*a -= G13(\*b + subkey(K, i, 2));\

\*c += \*b;\

\*b += G21(\*c + subkey(K, i, 3)) ^ i;\

\*c -= \*b;\

\*d += G13(\*c + subkey(K, i, 4));\

\*b ^= G21(\*a + subkey(K, i, 5));\

\*c ^= G5(\*d + subkey(K, i, 6));\

#define subkey\_e(K, i, j) K[(7 \* i - 7 + j) % 8]

/\*

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

Такты зашифрования

Перестановка содержимого регистров a, b, c, d реализуется перестановкой

параметров макроса R. После выполнения последнего макроса R и шагов 2.10-2.12

алгоритма зашифрования в регистрах a, b, c, d будут находиться значения,

соответствующие спецификации belt.

Окончательная перестановка abcd -> bdac реализуется инверсиями:

a <-> b, c <-> d, b <-> c.

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

\*/

#define E(a, b, c, d, K)\

R(a, b, c, d, K, 1, subkey\_e);\

R(b, d, a, c, K, 2, subkey\_e);\

R(d, c, b, a, K, 3, subkey\_e);\

R(c, a, d, b, K, 4, subkey\_e);\

R(a, b, c, d, K, 5, subkey\_e);\

R(b, d, a, c, K, 6, subkey\_e);\

R(d, c, b, a, K, 7, subkey\_e);\

R(c, a, d, b, K, 8, subkey\_e);\

\*a ^= \*b, \*b ^= \*a, \*a ^= \*b;\

\*c ^= \*d, \*d ^= \*c, \*c ^= \*d;\

\*b ^= \*c, \*c ^= \*b, \*b ^= \*c;\

/\*

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

Зашифрование блока

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

\*/

void beltBlockEncr(u32 block[4], const u32 key[8])

{

E((block + 0), (block + 1), (block + 2), (block + 3), key);

}

/\*

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

Расшифрование блока

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

\*/

void memXor2(void\* dest, const void\* src, size\_t count)

{

for (; count >= O\_PER\_W; count -= O\_PER\_W)

{

\*(WORD\*)dest ^= \*(const WORD\*)src;

src = (const WORD\*)src + 1;

dest = (WORD\*)dest + 1;

}

while (count--)

{

\*(octet\*)dest ^= \*(const octet\*)src;

src = (const octet\*)src + 1;

dest = (octet\*)dest + 1;

}

}

/\*

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

Шифрование в режиме CTR

Для ускорения работы счетчик ctr хранится в виде [4]u32. Это позволяет

зашифровывать счетчик с помощью функции beltBlockEncr(), в которой

не используется реверс октетов даже на платформах BIG\_ENDIAN.

Реверс применяется только перед использованием зашифрованного счетчика

в качестве гаммы.

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

\*/

void beltCTRStart(void\* state, const octet key[], size\_t len,

const octet iv[16])

{

belt\_ctr\_st\* st = (belt\_ctr\_st\*)state;

beltKeyExpand(st->key, key, len);

u32From(st->ctr, iv, 16);

beltBlockEncr(st->ctr, st->key);

st->reserved = 0;

}

void beltCTRStepE(void\* buf, size\_t count, void\* state)

{

belt\_ctr\_st\* st = (belt\_ctr\_st\*)state;

// есть резерв гаммы?

if (st->reserved)

{

if (st->reserved >= count)

{

memXor2(buf, st->block + 16 - st->reserved, count);

st->reserved -= count;

return;

}

memXor2(buf, st->block + 16 - st->reserved, st->reserved);

count -= st->reserved;

buf = (octet\*)buf + st->reserved;

st->reserved = 0;

}

// цикл по полным блокам

while (count >= 16)

{

beltBlockIncU32(st->ctr);

beltBlockCopy(st->block, st->ctr);

beltBlockEncr((u32\*)st->block, st->key);

beltBlockXor2(buf, st->block);

buf = (octet\*)buf + 16;

count -= 16;

}

// неполный блок?

if (count)

{

beltBlockIncU32(st->ctr);

beltBlockCopy(st->block, st->ctr);

beltBlockEncr((u32\*)st->block, st->key);

memXor2(buf, st->block, count);

st->reserved = 16 - count;

}

}

//

//file dip\_proj.ino

//

#include <OneWire.h>

#include <DallasTemperature.h>

#include <SPI.h>

#include <Ethernet.h>

#include <EEPROM.h>

#include <belt.h>

#define ONE\_WIRE\_BUS 6

OneWire oneWire(ONE\_WIRE\_BUS); //Настройка 1wire для работы с 6-м выводом Ардуино

DallasTemperature sensors(&oneWire); //Подключаем датчик температуры

bool SPI\_is\_receiving = false; //индикация получения данных по SPI

size\_t len = 16; //длина ключа

octet key[16] = {0}; //объявление глобального массива для хранения ключа

const octet iv[16] = { 'z', 'j', 'l', 'b', 'y', ':', 'b', 'd', '0', 'q', 'f', 'l', 'h', 'e', 'u', '7' }; //синхропосылка

char outData[256] = {0}; //выходные данные, нуждающиеся в шифровании

char inData[256] = {0}; //входные данные, нуждающиеся в расшифровке

belt\_ctr\_st stateEncr = {0};

belt\_ctr\_st stateDecr = {0};

int ii=1;

int ij=0;

byte r=0,g=0,b=0;

char buf[100];

char bb[40];

// задание MAC-адреса устройства

byte mac[] = {0x0E, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

// Параметры сети по умолчанию при отсуствии DHCP-свервера

// Настройка IP-адреса, шлюза, DNS-сервера, и маски сети

IPAddress ip(192,168,1,30);

IPAddress myDns(8,8,8,8);

IPAddress gateway(192,168,1,1);

IPAddress subnet(255,255,255,0);

EthernetClient client;

int addr\_len=0;

int addr = 0;

char server[100];

int buttonState = 1;

unsigned long lastConnectionTime = 0; // время последнего подключения к серверу (в миллисекундах)

boolean lastConnected = false; // последнее состояние подключения

const unsigned long postingInterval = 10\*1000; // задержка между обновлениями данных (в миллисекундах)

void setup() {

int i;

pinMode(3, OUTPUT); // LED 1

pinMode(4, INPUT); // Кнопка PD4

pinMode(5, OUTPUT); // LED 2

pinMode(7, OUTPUT); // LED 3

// инициализация SPI:

SPI.begin();

SPI.setBitOrder(MSBFIRST); // MSBFIRST – приоритет старшего бита, LSBFIRST – приоритет младшего бита

SPI.setClockDivider(SPI\_CLOCK\_DIV4); //установка делителя частоты SPI (16Mhz/4 = 4Mhz)

digitalWrite(SS, HIGH);

// инициализация UART:

Serial.begin(9600); //установка скорости передачи данных UART

// Ожидание загрузки Ethernet-модуля

delay(1000);

buttonState = digitalRead(4);

// Проверка, нажата ли кнопка PD4 сразу после включения питания

if (buttonState == 0){ // Если нажата, необходимо ввести имя сервера и ключ шифрования

for(ia=0;ia<100;ia++){

addr=ia; EEPROM.write(addr, 0); // Обнуляем EEPROM

}

Serial.print("Input server name: ");

while(Serial.available() <= 0){ ;} // Ожидание ввода

delay(1000); // Задержка для передачи в буфер

for(i = 0; i < len; i++) //считывание ключа с последовательного порта при запуске

{

while(Serial.available() == 0){} // ожидание начала передачи данных

key[i] = Serial.read(); // считывание ключа шифрования посимвольно

}

i=0;

while(Serial.available() > 0 && addr\_len < 100) {

server[i] = Serial.read(); addr\_len=i; i++; // Чтение адреса сервера

}

for(i = 0; i < 16; i++)

EEPROM.write(i, key[i]); // Запись ключа в EEPROM

for(i = 0; i < addr\_len; i++)

EEPROM.write(i + 16, server[i]); // Запись адреса в EEPROM

Serial.println();

}

else {

for(i = 0; i < 16; i++)

key[i] = EEPROM.read(i); // чтениею ключа с EEPROM

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

server[i]=EEPROM.read(i + 16); // Чтение имени сервера с EEPROM

}

// инициализация состояния шифратора

beltCTRStart(&stateEncr, key, len, iv);

// инициализация состояния дешифратора

beltCTRStart(&stateDecr, key, len, iv);

Serial.print("Server name: ");

Serial.println(server);

// Определение IP параметров с помощью DHCP

if (Ethernet.begin(mac) == 0) {

Serial.println("Failed to configure Ethernet using DHCP");

// initialize the ethernet device not using DHCP:

Ethernet.begin(mac, ip, myDns, gateway, subnet);

}

// start the Ethernet connection using a fixed IP address and DNS server:

// print the Ethernet board/shield's IP address:

Serial.print("My IP address: ");

Serial.println(Ethernet.localIP());

Serial.print("My DNS address: ");

Serial.println(Ethernet.dnsServerIP());

Serial.print("Gateway address: ");

Serial.println(Ethernet.gatewayIP());

Serial.print("SubnetMask: ");

Serial.println(Ethernet.subnetMask());

}

void loop() {

// Получены ли данные по сети

if (client.available()) {

char c = client.read(); // чтение данных

beltCTRStepE(&c, 1, &stateDecr); //расшифровка байта данных

Serial.print(c); // отладочный вывод

buf[ii-1]=c; ij=ii;

if(c=='\n')

ii=0;

ii++;

}

// обработка данных от сервера

if (!client.connected() && lastConnected) {

Serial.println();

Serial.print("disconnecting");

if(ij<=1) goto lab;

for(ii=0;ii<ij;ii++)

Serial.write(buf[ii]);

Serial.println();

if( buf[1]=='1') {digitalWrite(3,HIGH);}

if( buf[1]=='0') {digitalWrite(3,LOW);}

if( buf[2]=='1') {digitalWrite(5,HIGH);}

if( buf[2]=='0') {digitalWrite(5,LOW);}

if( buf[3]=='1') {digitalWrite(7,HIGH);}

if( buf[3]=='0') {digitalWrite(7,LOW);}

lab:

client.stop();

}

// Если клиент не подключен и прошло 10 секунд с последнего подключения

// Тогда подключаемся снова и отправляем данные

if(!client.connected() && (millis() - lastConnectionTime > postingInterval)) {

sensors.requestTemperatures();

int temp=sensors.getTempCByIndex(0);

httpPost(&temp, 2); // шифрование и отправка данных от сенсора

}

if( !client.connected() && (millis() - lastConnectionTime > postingInterval/2)) {

httpGet(); // опрашиваем сервер

}

// сохранить последнее состояние (подключен/не подключен)

lastConnected = client.connected();

}

// Шифрование и отправка данных на сервер

void httpPost(char \* data, size\_t len) {

if (client.connect(server, 80)) {

Serial.println("connecting (POST)...");

// send the HTTP PUT request:

client.println("POST /command.php HTTP/1.1");

client.println("User-Agent: arduino-ethernet");

client.println("Connection: close");

client.println();

for(int i = 0; i < len; i++) {

// шифрование

beltCTRStepE(enc\_data + i, 1, &stateEncr);

// отправка данных

client.print(data[i]);

}

// note the time that the connection was made:

lastConnectionTime = millis();

}

else {

// if you couldn't make a connection:

Serial.println("connection failed");

Serial.println("disconnecting.");

client.stop();

}

}

// опрос сервера

void httpGet() {

if (client.connect(server, 80)) {

Serial.println("connecting (GET)...");

client.println("GET /command.php HTTP/1.1");

client.println("Connection: close");

client.println();

}

}

# Приложение Б

**(обязательное)**

**Отчёт о проверке на заимствование**

Отчёт о проверке на заимствование материалов пояснительной записки представлен на рисунке Б.1.

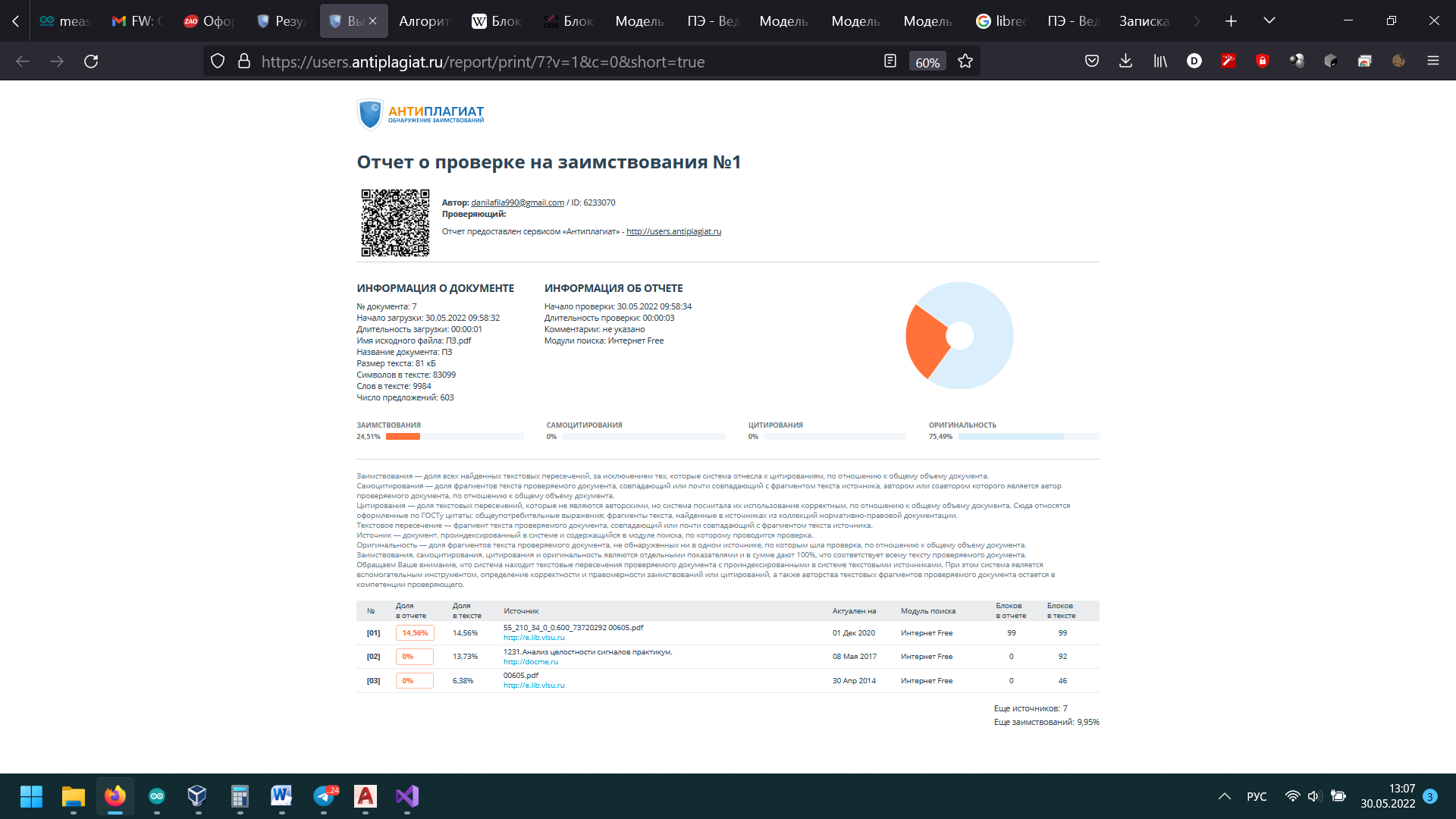


Рисунок Б.1 – Отчёт о проверке на заимствование

|  |  |
| --- | --- |
|  | Филипцов Д. А. |