МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»
(БГТУ им. В.Г.Шухова)

(ТИТУЛЬНИК ВРЕМЕННЫЙ)Выпускная квалификационная работа

дисциплина «.»

«.»

Выполнил: студент группы ВТ-41 Макаров Д.С.

Проверил: Шамраев А.А.

Белгород 2021

Содержание

ВВЕДЕНИЕ

С развитием вычислительной техники и сетей, все большее распространение получает концепция Internet of Things (Интернета вещей) - оснащение вычислительными функциями датчиков, промышленного оборудования или повседневных предметов, с последующим подключением его к всемирной сети, для сбора и обработки генерируемой устройствами информации. Современные системы домашней автоматизации являются частным случаем IoT, каждый датчик или актуатор содержит вычислительное устройство и в том или ином виде отдает информацию в вычислительную сеть. Однако данный подход несет за собой следующие проблемы.

Так как такие вычислительные сети имеют доступ во всемирную сеть Интернет, становится важным аспект безопасности. Несанкционированный доступ к системам видеонаблюдения, климат-контроля или другим важным узлам домашней автоматизации, могут привести к серьезным повреждениям инфраструктуры дома и угрозе жизни проживающих в нем людей, а так же возможности утечки конфиденциальной информации.

Еще одна проблема современных систем домашней автоматизации - проблема совместимости или интероперабельности устройств. На данный момент существует множество реализаций компонентов “умного дома”, как в виде коммерческих решений от крупных компаний, защищенных патентами, так и в виде свободно-распространяемого программного и аппаратного обеспечения разрабатываемых энтузиастами. Разнообразие реализаций с различными протоколами и стандартами подключения между устройствами, затрудняет построение и расширение систем домашней автоматизации.

В данной ВКР будет предложено одно из решений проблемы совместимости вычислительных устройств и реализован межпротокольный шлюз CAN-Ethernet для интеграции устройств использующих CAN, в системы домашней автоматизации, при помощи прикладного протокола MQTT.

Пояснительная записка к выпускной квалификационной работе включает в себя следующие разделы.

Раздел “Описание предметной области, анализ и выбор методов решения задач”, описывающий основные принципы и понятия, используемые в домашней автоматизации, производится анализ существующих решений на рынке систем. Описываются проблемы существующих решений, и предлагаются варианты решений этих проблем.

В разделе “Проектирование аппаратного обеспечения” описывается, процесс проектирования схемотехники и печатной платы межпротокольного шлюза MQTT-CAN, а также обосновывается выбор тех или иных аппаратных компонентов.

В разделе “Проектирование программного обеспечения” описан процесс проектирования и спецификации сетевых и прикладных протоколов CAN шины, форматы кадров, алгоритмы их работы, а также алгоритмы работы межпротокольного шлюза MQTT-CAN.

Раздел “Программная реализация” описывает структуру данных программного обеспечения межпротокольного шлюза, спецификации подпрограмм, блок-схемы алгоритмов работы подпрограмм и описание используемых в ПО, сторонних библиотек и зависимостей, руководство пользователя, и описание процесса тестирования готового ПАК.

@todo РАМКИ

ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ, АНАЛИЗ И ВЫБОР МЕТОДОВ РЕШЕНИЯ ЗАДАЧ

2.1 Описание и анализ предметной области

Программно-аппаратный комплекс - набор аппаратных и программных средств, работающих для выполнения одной или нескольких связанных задач.

Аппаратная часть комплекса представляет собой: компьютеры и микроконтроллеры объединенные в одну вычислительную сеть, посредством гетерогенных сетей, а так же различные датчики и исполнительные устройства.

Программной частью комплекса являются встроенное программное обеспечение, реализующее функционал исполнительных устройств и датчиков, сетевое взаимодействие, различные программные модули сетевых протоколов.

Система домашней автоматизации решает ряд задач таких как:

- сбор и обработка данных полученных с датчиков системы, а так же с внешних источников.
- регулирование внутренних параметров системы, в зависимости от обработанных данных и предпочтений пользователей.
- обеспечение безопасности дома (пожарная безопасность, предупреждение аварийных ситуаций).
- упрощение однотипных процессов пользователей при помощи программируемых сценариев.
- экономия ресурсов домовладения, при помощи эффективного их использования.
- предоставление различных отчетов, по использованным ресурсам домовладения.

Саму система может включать в себя различные подсистемы разделенные по функциональным возможностям:

- подсистема управления освещением:
  - элементы управления освещением
  - модули управления естественным освещением (шторы, рольставни, электрохромные стекла)
  - датчики присутствия, освещения
  - контроллеры сцен освещения (RGB контроллеры)
- подсистема управления климатом:
  - внутренние и внешние метеорологические датчики (термометр, барометр, гигрометр, датчик уровня CO_2 и чистоты воздуха)
  - модули управления климатом
  - термостаты, рекуператоры, гигростаты, приточная и вытяжная вентиляция (с электрическим управлением)
- подсистема безопасности:
  - подсистема видеонаблюдения
  - модули детектирования утечек природного газа, утечек воды
  - умные замки
  - модули сигнализации и сирены
- управляющая подсистема

Рисунок 1 - Схема подсистем системы домашней автоматизации

Существует несколько механизмов коммуникации устройств системы между собой:

- устройство-устройство.
- устройство-облако.
- устройство-шлюз.

Подключение типа устройство-устройство, представляет собой прямое подключение по одному из стандартов связи (WiFi, Ethernet, Zigbee), без сторонних сервисов или устройств. Механизм прост в настройке и при использовании в сетях с малым количеством устройств, но используемые устройствами протоколы разнятся от устройства к устройству, тем самым создавая проблемы совместимости для конечного пользователя, ограничивая выбор устройств одним семейством. Так же устройства использующие одну и ту же среду передачи могут использовать различные протоколы и форматы сообщений, создавая проблемы совместимости.

Устройства использующие подключение вида устройство-облако, подключаются напрямую к серверам поставщика услуг, по стандартным протоколам всемирной сети, получить доступ к данным или функциям устройства, при этом можно только через интерфейсы предоставляемые облачными сервисами поставщика услуг. Таким образом данные передаваемые устройством могут обрабатываться на удаленных вычислительных мощностях, а потом выводится в веб-интерфейс или мобильное приложение пользователю. Это позволяет, в некоторых случаях, расширять функционал устройств, а так же не иметь пользователю никакой дополнительной инфраструктуры для поддержания системы домашней автоматизации, кроме локальной сети с подключением в сеть Интернет. Проблемы интероперабельности, при использовании данного типа устройств возникают в некоторых случаях: производители устройств могут ограничивать выбор поставщиков облачных услуг.

При подключении типа устройство-шлюз, некоторый набор функциональных устройств системы автоматизации работающих по единому

сетевому и прикладному протоколу, подключены к шлюзу, который уже в свою очередь агрегирует всю входящую и исходящую информацию и отправляет ее поставщику услуг.

2.2 Анализ существующих аналогов

В данные момент на рынке систем домашних автоматизаций преобладают 2 типа реализаций: готовые наборы устройств от крупных производителей и открытые реализации систем поддерживаемые и разрабатываемые сообществами энтузиастов.

Среди готовых продуктов зарубежных производителей можно отметить Xiaomi Smart Home, Philips Hue, Samsung SmartThings и другие. Отечественные компании так же предлагают к покупке продукты такие как “Умный дом Sber” и “Умный дом (Rostelecom)”.

Достоинства готовых систем заключаются в простоте настройки и расширения системы, гарантированной тех. поддержки от производителя. Недостатки таких продуктов связаны с привязкой всех устройств одного производителя к его инфраструктуре, производитель сам является поставщиком облачных услуг для устройств домашней автоматизации, а поэтому они могут контролировать почти все аспекты использования устройств. Все данные генерируемые устройствами, будут собираться и хранится производителем для дальнейшего использования, в контекстной рекламе или машинном обучении. Использование проприетарных, не задокументированных или запатентованных протоколов и интерфейсов, затрудняет или вовсе исключает возможность разработки устройств, сторонними производителями или энтузиастами. А прекращение поддержки или отключение облачных сервисов, со стороны поставщика услуг, может привести к значительному уменьшению функциональности устройства, вплоть до полного выхода из строя.

В качестве альтернативы облачным поставщикам услуг, заложенными производителем устройства при проектировании, могут выступать решения разворачиваемые в локальной сети. Среди них можно выделить

системы домашней автоматизации HomeAssistant, Majordomo, OpenHUB. Данные система предоставляет возможности описания пользовательских сценариев, мониторинга данных, полученных с датчиков, имеет множество интеграций с устройствами крупных производителей или подключении устройств по открытым протоколам HTTP API, MQTT. В отличие от систем автоматизации, поставляемых производителями устройств, вышеперечисленные системы не передают данные собранные устройствами третьим лицам. Так же благодаря широкой поддержки устройств, решается проблема интероперабельности. Недостаток таких систем - необходимость арендовать вычислительные мощности или же иметь локальный сервер для развертывания экземпляра системы, а также для установки, конфигурации данных систем требуют некоторых компетенций, что затрудняет их использование рядовому пользователю.

2.3 Выбор методов решения задачи

Для описания системы взаимодействия устройств между собой, можно использовать модель OSI. Модель разделена на 7 уровней, каждый из которых коммуницирует только с соседними.

Уровни модели OSI, в порядке убывания:

- Прикладной - обеспечивает доступ и взаимодействие пользовательских приложений с сетью
- Представления - обеспечивает преобразования информации (например, кодирование или шифрование), между сеансовым/транспортным и прикладным протоколом
- Сеансовый - обеспечивает поддержание сеанса связи, может не использоваться в некоторых случаях.
- Транспортный - предназначен для обеспечения доставки сегментированных сообщений, гарантий доставки сообщений, а так же мультиплексировании соединений (в некоторых протоколах)
- Сетевой - обеспечивает маршрут передачи данных
- Канальный - на данном уровни происходит контроль ошибок и обеспечивается взаимодействие между сетями на физическом уровне

˘ Физический - регламентирует физическую среду передачи сообщения, типы сигналов, механизмы передачи информации.

Рисунок 2 - Модель OSI

В современных вычислительных сетях используется стек TCP/IP, использующий различные физические среды передачи информации: медный кабель (802.3), оптическое волокно (802.3), радиоволны (802.11), и IP в качестве сетевого протокола, а TCP/UDP в качестве протоколов транспортного уровня.

Рисунок 3 - Стек TCP/IP

Но использование стека TCP/IP во всех встраиваемых устройствам затруднено, из-за сложности протоколов физического и канального уровня, а следовательно относительно большой стоимости аппаратных компонентов и энергопотребления, реализующих это протоколы.

Для исправления данных недостатков можно использовать схему подключения “устройство-шлюз”, реализовав внутренее подключение между устройствами “умного дома”, по каналам связи с более примитивным протоколом канального и сетевого уровня, а подключение к провайдеру реализовать через широкораспространенный стек TCP/IP.

В качестве протокола внутри сети шлюза, предлагается выбрать протокол с возможностью построения сетей по топологии “шина” для проводных соединений или полнозвязную ячеистую сеть для беспроводных соединений. Топология “шина” позволит упростить и удешевить монтаж коммуникаций в доме. Среди подходящих протоколов канального уровня, для проводных соединений: можно отметить протокол CAN, широко используемый в промышленности и автомобильных коммуникациях, и протокол KNX, широко

распространенный в системах домашней автоматизации, европейских стран.

@todo (рассказываем про CAN)

@todo (предлагаем гетерогенные приколы)

@todo (рассказываем про MQTT)

@todo (рассказываем про Bluetooth LE Mesh)

@todo (рассказываем про шлюзы и что именно будет в реализовано в дипломе)

ПРОЕКТИРОВАНИЕ АППАРАТНОГО ОБЕСПЕЧЕНИЯ

3.1 Разработка принципиальной схемы

@todo (нарисовать схему шлюза и рассказать про ее модули)

3.2 Разработка печатной платы

@todo (нарисовать плату и рассказать про нее)

Печатная плата спроектирована для изготовления по следующим техничским процессам.

- Материал изготовления: стеклотекстолит FR4
- Минимальный зазор между проводниками: 0.1 мм
- Количество слоев: 2
- Переходные отверстия: 0.2 мм
- Минимальный размер отверстия: 0.2 мм
- Толщина печатной платы: 1.5 мм
- Толщина слоя металлизации: 18 мкм

@todo добавить ссылку на литературу на ГОСТ. Класс точности печатной платы: 5 (ГОСТ 23751-86)

4.1 Разработка методов решения задач

@todo (схема модели OSI но с CAN)

Так как стандарт CAN не описывает протоколы прикладного и канального уровня, а существующие протоколы заточены на использования в автомобильной индустрии или крупных промышленных установок.

4.2 Разработка структур данных

Ниже описаны структуры используемые в протоколе канального уровня CAN шины.

@todo CAN_ID, DHCP_CAN_IP: структуры данных

@todo Структуры данных, спецификации и блок схемы DHCP.

Следующие структуры используются в протоколе получения адреса DHCP_LIKE.

Адресная таблица, используемая для управления адресами на CAN шине

Для хранения информации о выданных адресах мастер на CAN шине (в данном случае шлюз), создает у себя в памяти таблицу соответствия адреса устройства на CAN шине и уникального аппаратного идентификатора.

Таблица представляет из себя массив размером до 255 элементов, каждый элемент содержит структуру запись таблицы. Она состоит из 2 полей:

- can_addr - 1 байтовое беззнаковое число, содержащее адрес устройства на шине.
- hwid - массив из 4-х 4 байтовых чисел, содержащее уникальный аппаратный идентификатор.

4.3 Разработка и описание алгоритмов

Алгоритмы работы с адресная таблицей

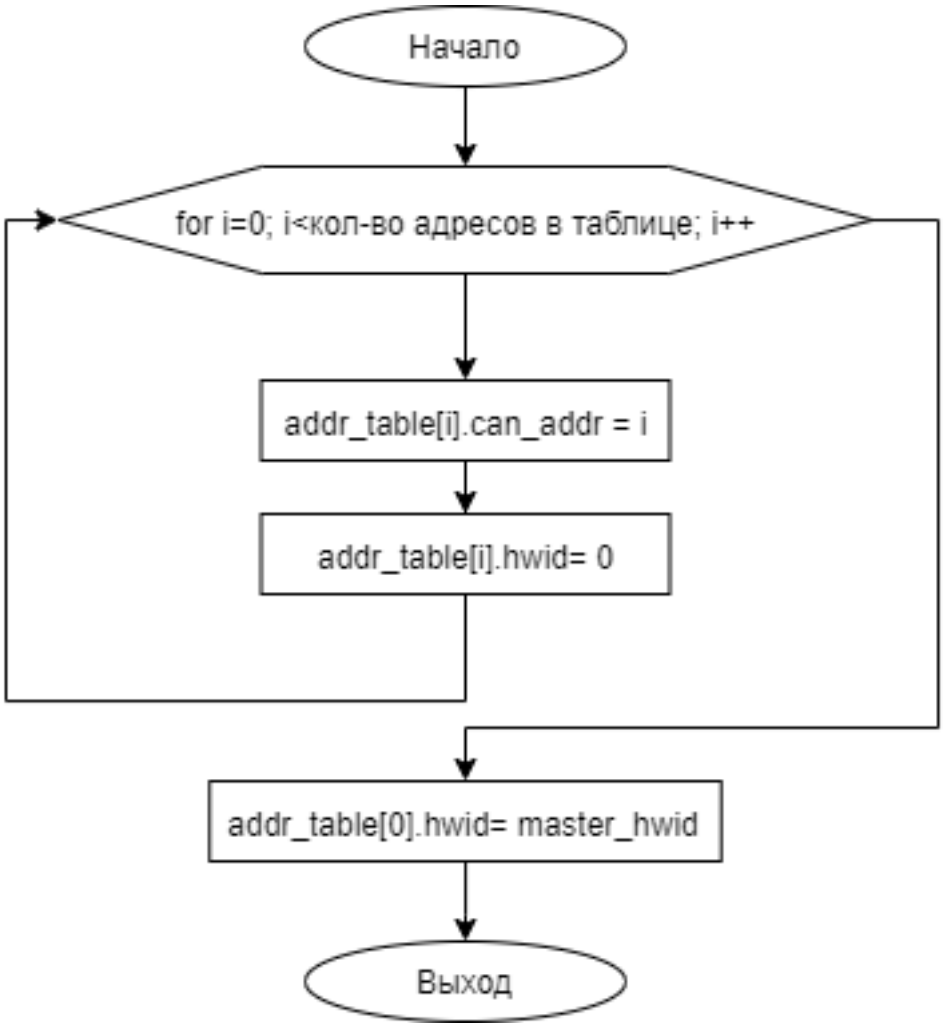

Рисунок 4 - Блок схема алгоритма инициализации адресной таблицы

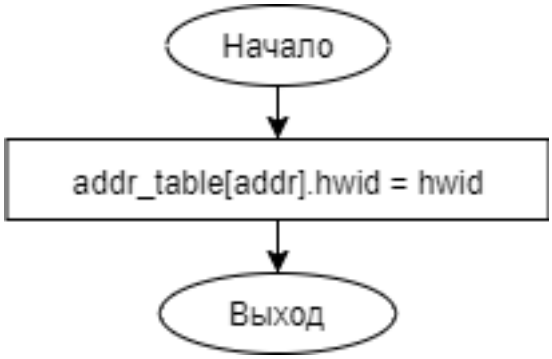

Рисунок 5 - Блок схема алгоритма добавления адреса в адресную таблицу

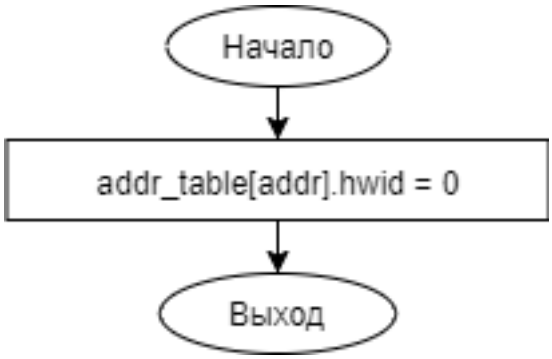

Рисунок 6 - Блок схема алгоритма удаление адреса из адресной таблицы

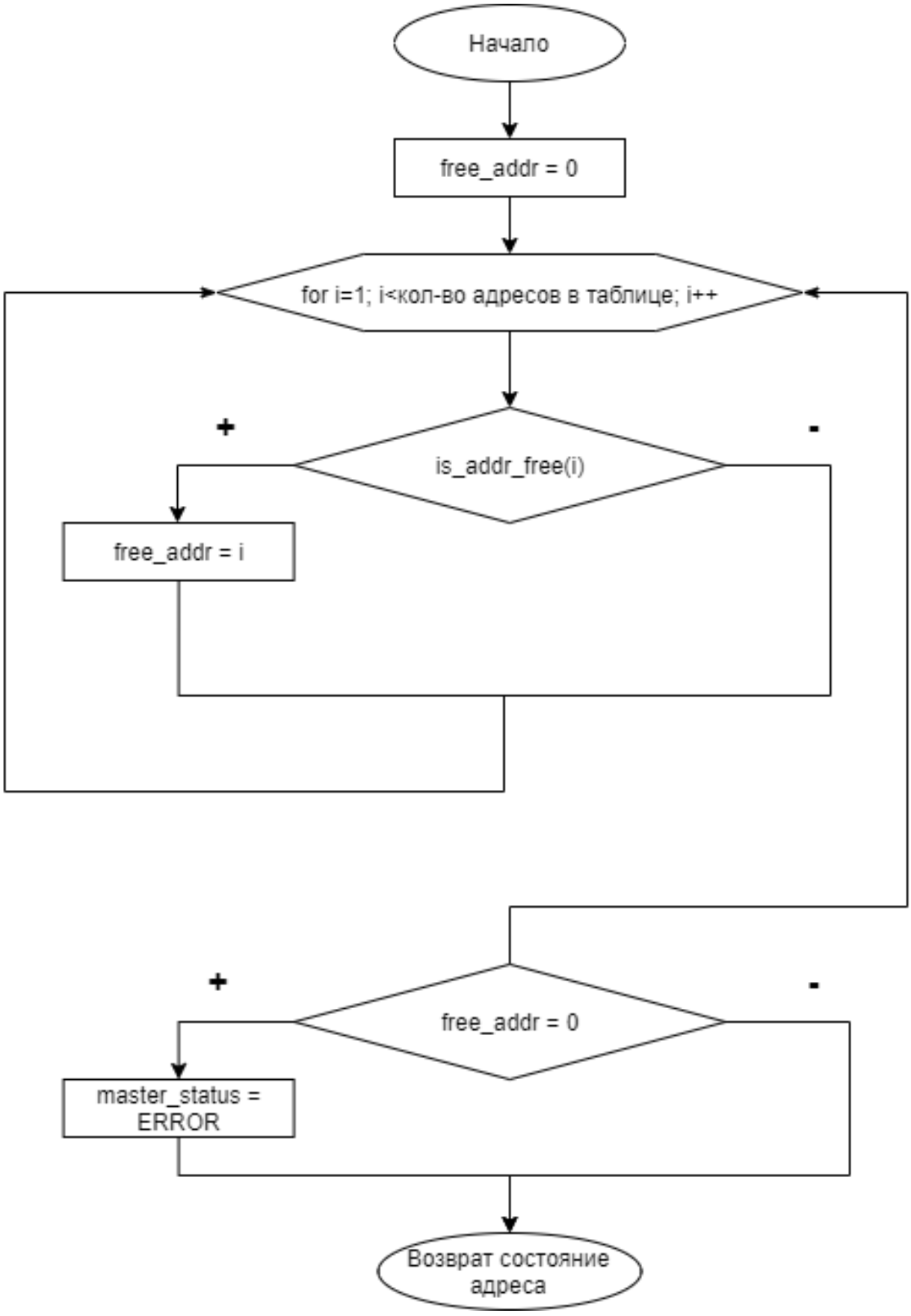

Рисунок 7 - Блок схема алгоритма поиска свободного адреса

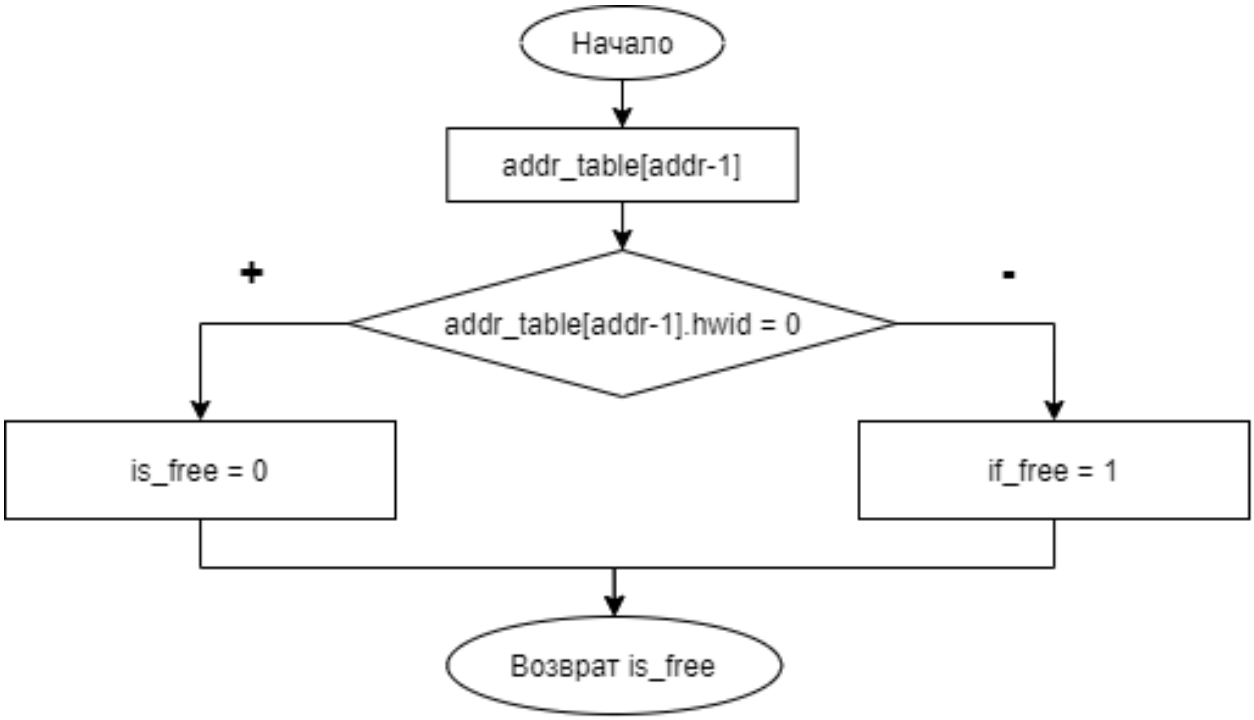

Рисунок 8 - Блок схема алгоритма проверка доступности адреса

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

5.1 Описание модульной структуры программы

5.2 Спецификации подпрограмм (методов)

Функции модуля can_bus

Функция `can_bus_init`

Назначение: Функция инициализации переферийного модуля CAN, так же в этой функции инициализируется адрес устройства на шине.

Входные параметры: нет

Возвращаемое значения: нет

Функция `can_send_packet`

Назначение: Функция передачи пакета по CAN шине.

Входные параметры:

- `uint8_t dest_addr` - Адрес назначения отправленного пакета
- `uint8_t frame_data` - Поле данных зависимых от типа пакета
- `uint8_t frame_type` - Тип пакета
- `uint8_t prior` - Приоритет пакета
- `uint8_t size` - Размер данных в основном теле пакета
- `uint8_t* data_ptr` - Указатель на данные.

Возвращаемое значения: нет.

Функция `can_recv_packet`

Назначение: Функция настраивает программные фильтры входящих пакетов, и ожидает окончания приема первого подходяшего под фильтр пакета.

Входные параметры:

- int16_t src_addr - Адрес отправителя, (-1 если любой)
- int16_t frame_type - Тип принимаемого пакета (-1 если любой)
- can_id* id_ptr - Указатель на структуру-идентификатор CAN пакета
- uint8_t* data_ptr - Указатель на основное поле данных CAN пакета.

Возвращаемое значения: размер полученных данных в байтах.

Функции модуля dhcp_like

Функция `dhcp_master_init`

Назначение: функция инициализация dhcp_like протокола.

Входные параметры: нет

Возвращаемое значения: нет

Функция `dhcp_slave_recv_addr`

Назначение: функция запроса и получения адреса с master узла по протоколу dhcp_like.

Входные параметры: нет

Возвращаемое значения: адрес полученный в результате запроса, при ошибке или отсутствии свободных адресов -1.

Функции работы с адресной таблицей (модуль dhcp_like_master)

Функция `is_addr_free`

Назначение: проверка на доступность к выдаче адреса addr

Входные параметры: `uint8_t addr` - проверяемый адрес

Возвращаемое значения: 0 - адрес занят, иначе адрес свободен

Функция `find_free_addr`

Назначение: поиск свободного адрес в таблице адресов

Входные параметры: нет

Возвращаемое значения: 0 - свободных адресов нет, иначе свободный готовый к выдаче адрес

Функция `add_addr`

Назначение: выдача адреса addr, устройству с аппаратным идентификатором hwid

Входные параметры:

- `uint8_t addr` - выдаваемый адрес
- `uint8_t* hwid` - указатель на массив, содержащий аппаратный идентификатор получателя адреса

Возвращаемое значения: нет

Функция `rm_addr`

Назначение: освобождение адреса addr

Входные параметры: `uint8_t addr` - освобождаемый адрес

Возвращаемое значения: нет

Функция `init_addr_table`

Назначение: инициализация адресной таблицы

Входные параметры: нет

Возвращаемое значения: нет

5.3 Описание использованных внешних компонент и библиотек

@todo добавить ссылку https://github.com/ARM-software/CMSIS_5

Библиотека CMSIS

Так как разработка ПО ведется для процессоров архитектуры ARM, была использована библиотека абстрагирования аппаратного уровня CMSIS. CMSIS - Cortex Microcontroller Software Interface Standart (Стандарт программных интерфейсов микроконтроллеров архитектуры Cortex). Данная библиотека предоставляет независимые от аппаратной реализации программные абстракции для работы с процессором построенным на архитектуре ARM Cortex. Библиотека предназначена для уменьшения порога вхождения в разработку встраиваемых систем на основе процессоров ARM и уменьшении времени разработки новых устройств. В состав библиотеки входят:

- Core(M) - Стандартизированное API для процессоров и переферии семейства Cortex-M.
- Core(A) - Стандартизированное API для процессоров Cortex(A) и базовая среда выполнения для них.
- Driver - Интерфейсы для работы с базовой переферией и примеры работы с графическими интерфейсами, файловыми системами и тд.
- DSP - Модуль для работы с аппаратным ускорителем обработки чисел с фиксированной запятой (q7,q15,q31), и 32 битных чисел с плавающей запятой одинарной точности.
- NN - Модуль содержащий среды выполнений нейронных сетей различных архитектур, оптимизированные для работы с процессорами ARM.
- RTOSv1/v2 - Программный интерфейс с базовой реализацией компонентов операционной системы реального времени, независимых от самой реализации ОС.
- SVD - Модуль для работы с различными отладчиками
- Zone - Модуль описывающий методы разделения сред исполнения програмного кода.

Библиотека STM32CUBE для работы с переферией микроконтроллера

STM32Cube - набор инструментов для быстрой разработки встраиваемых систем, на базе процессоров STM32.

Он состоит из нескольких независимых друг от друга инструментов, документации к ним и примеров использования.

STM32CubeMX - генератор начального кода инициализации и пользовательского интерфейса позволяющий:

- Генерировать код инциализации процессора, тактовых генераторов, переферии и других сторонних библиотек, на языке программирования Си.
- Генерировать готовые проекты под разные системы сборки и IDE.
- Расчитывать потребляемую устройством мощность, в зависимости от выбранных пользователем тактовых частот и включенной переферии.
- Встроенная утилита для поддержания библиотек в актуальном состоянии.

Существуют пакеты STM32Cube для каждого семейства микроконтроллеров STM32, каждый такой пакет включает в себя:

- HAL - абстракцию между различными микроконтроллерами STM32, предоставляющую единый стандартизированный программный интерфейс.
- LL - легковесный оптимизированный программный интерфейс, жестко привязанный к конкретному процессору, но разработанный с расчетом на энергоэффективность и быстродействие.
- Набор сторонних библиотек, таких как операционные системы реального времени (FreeRTOS, Azure RTOS), библиотеки работы с USB устройствами, TCP-IP стек, библиотеки работы с сенсорными устройствами, а так же инструменты для создания графических пользовательских интерфейсов.

Библиотека ioLibrary для работы с чипами Wiznet и протоколами Internet

Библиотека ioLibrary (“Internet Offload Library”) предназначена для работы с контроллерами фирмы Wiznet. Данная библиотека предоставляет программный интерфейс для взаимодействия с чипами, совместимый с Berkley Socket.

Так же в поставке библиотеки присутствуют реализации таких протоколов прикладных как FTP, DHCP, DNS, MQTT, HTTP.

Реализация семейства функции printf без зависимостей от stdlib.h

Для генерации символьных сообщений протокола MQTT удобно использовать функцию *sprintf*, из семейства функций *print formatted*, позволяющая выводить строки содержащие значения различных типов. В случае функции sprintf, она позволяет выводить в отформатированную функцию в переменную строки а не потоки ввода-вывода, в отличии от других функций семейства. Но функции входящие в стандартную библиотеку языка Си, не подходят для использования в встраиваемых система из-за ряда особенностей.

Во первых стандартная реализация функций *printf* имеет ряд зависимостей из стандартной библиотеки языка Си, которые после компиляции занимают значительный объем в флеш памяти микроконтроллера (около 32 Кб).

Во вторых стандартная реализация *printf* использует динамическое выделение памяти, что не рекомендовано стандартами и руководствами разработки программно-аппаратных комплексом высокой надежности и ответственности (IEC 61508, MISRA C).

Для решения этой проблемы была использована сторонняя реализация функций семейства printf (ссылка на репозиторий) для языка Си, разработанная специально для использования в встраиваемых системах.

Данная реализация имеет совместимость с реализацией из стандартной библиотеки, но занимает значительно меньший объем в флеш памяти микроконтроллера (14 Кб против 32 Кб).

Так же реализация не использует динамическое выделение памяти для работы функций, что соответствует мировым стандартам разработки встраиваемых систем, а так же исправляет некоторые особенности стандартной реализации функций семейств *printf*, связанных с потоко-безопасностью.

Библиотека со сторонней реализацией распространяется с открытой лицензией MIT, что позволяет использовать данную реализацию в любых коммерческих и не коммерческих разработках.

5.4 Руководство пользователя

5.5 Тестирование и экспериментальная проверка программно-аппаратного комплекса

5.6 Оценка качества разработанного комплекса

ЗАКЛЮЧЕНИЕ

Список литературы

https://www.internetsociety.org/wp-content/uploads/2015/10/report-InternetOfThings-20151221-ru.pdf