ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННЫХ

СИСТЕМ И ТЕХНОЛОГИЙ

РАЗРАБОТКА УСТРОЙСТВА АНАЛИЗА ЧАСТОТНЫХ РЕСУРСОВ Wi-Fi СЕТИ

Выпускная квалификационная работа
обучающегося по направлению подготовки 11.03.02 Инфокоммуникационные технологии и системы связи
очной формы обучения, группы 07001308
Ангелюка Артема Юрьевича

Научный руководитель
канд. техн. наук,
доцент кафедры
Информационно-
телекоммуникационных
систем и технологий
НИУ «БелГУ» Урсол Д.В.

Рецензент
Генеральный директор
ООО «Региональные
ТелеСистемы»
Романенко Д.В.

БЕЛГОРОД 2017

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
(НИУ «БелГУ»)
ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХНАУК
КАФЕДРА ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ
Направление подготовки *11.03.02 Инфокоммуникационные технологии и системы связи*
Профиль *«Системы радиосвязи и радиодоступа»*

Утверждаю
Зав. кафедрой
«____» _____ 201_г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Ангелюка Артема Юрьевича

1. Тема ВКР «*разработка устройства анализа частотных ресурсов Wi-Fi сети*»

Утверждена приказом по университету от «____» _____ 201_ г. № _____

2. Срок сдачи студентом законченной работы _____

3. Исходные данные к работе:
Обьект разработки анализатор частотных ресурсов WiFi сети, способный полностью проанализировать все данные базовых беспроводных точек доступа (MAC адрес, RSSI, частота и канал сети)
4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов):
4.1 Описание стандартов IEEE 802.11
4.2 Реализация устройства анализа частотного ресурса Wi-Fi сети
4.3 Проведение экспериментов разработанного устройства
4.4 Экономическая оценка проекта

5. Перечень практического материала (с точным указанием обязательных чертежей)
5.1 Обзор и выбор платформ для реализации устройства
5.2 Платформа Arduino
5.3 Платформа Raspbery pi
5.4 Платформа STM32
5.5 Тестирование в лабораторных условиях
5.6 Экономическая оценка проекта
5.7 Планирование работ по разработке
5.8 Расчет расходов на оплату труда на разработку
5.9 Расчет продолжительности разработки
5.10 Расчет стоимости расходных материалов
5.11 Расчет сметы расходов на разработку

6. Консультанты по работе с указанием относящихся к ним разделов работы

Раздел	Консультант	Подпись, дата	
		Задание выдал	Задание принял
4.1 – 4.3	*канд. техн наук, доцент каф. ИТСиТ Урсол.Д.В.*		
4.4	*канд. техн наук, доцент каф. ИТСиТ Болдышев А.В.*		

7. Дата выдачи задания _____

Руководитель
Кандидат технических наук, доцент кафедры Информационно-телекоммуникационных систем и технологий»
НИУ «БелГУ» _____*Урсол Д.В.*
(подпись)

Задание принял к исполнению _____*Ангелюк А.Ю.*
(подпись)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Анализатор WiFi (Wireless Fidelity) сетей – это прибор, программное обеспечение или стационарный программно-аппаратный комплекс, который используется для анализа и устранения неполадок, возникающих в работе беспроводных Wi-Fi сетей стандартов 802.11 a/b/g/n/ac. WiFi анализатор сканирует радиоэфир, обнаруживая все доступные беспроводные сети, точки доступа, Wi-Fi клиентов и выдаёт по ним подробную информацию, включая: название сетей, типы шифрования, данные об уровнях сигнала, уровнях помех в каналах, определение несанкционированных подключений и другую полезную информацию.

Применение этих решений на этапе развертывания беспроводной сети позволит быстро оценить уровень радиопомех и выбрать оптимальное расположение точек доступа для обеспечения наилучшей зоны покрытия. При появлении сбоев в работе Wi-Fi сети анализатор радиочастотного спектра поможет быстро выявить и локализовать источники помех.

Устройство представляет собой программно-аппаратный комплекс реализованный на микроконтроллере Arduino связанным беспроводным модулем ESP8266 и предназначенный для получения подробной информации о диагностируемых беспроводных сетях с отображением информации на жидкокристаллическом экране.

Подавляющее большинство анализаторов WiFi сетей являются программным обеспечением для современных операционных систем, что накладывает ряд ограничений на их использование:

1-необходимость устройства с операционной системой, которая необходима для запуска программного обеспечения анализатора, что приводит к увеличению стоимости сканера.

2-у большинства персональных компьютеров и смартфонов отсутствует разъем для подключения внешней антенны, что является препятствием для точной настройки используемых для построения беспроводных сетей.

3-отсутствие возможности подключения специализированных микросхем для анализа частотных помех.

В настоящее время российской промышленностью не выпускается аппаратных анализаторов беспроводных сетей, в том числе сетей WiFi для гражданского использования, а зарубежные аналоги стоят очень дорого. На российском рынке наиболее популярным является анализатор беспроводных сетей Airchek (Airchek G2) американской компании Fluke Networks. Стоимость этого прибора составляет около двухсот тысяч рублей. Краткое описание возможностей Airchek:

1) Выявление проблем, связанных с покрытием WiFi сети

2) Определение перегруженности сетей или каналов

3) Идентификация помех в каналах, на частотах 2,4 ГГц и 5 ГГц

4) Определение нестабильности соединений

5) Поиск неисправных точек доступа

6) Поиск несанкционированных точек доступа

7) Выявление клиентских проблем

8) Синхронизация с облачной службой Link-Live

9) Автоматическая отправка отчетов на email пользователя

10) Возможность тестирования медной Ethernet линии

11) Возможность создания разных пользовательских профилей

12) Возможность подключения внешней направленной антенны.

На этапе развертывания беспроводной сети наиболее востребованными являются функции выявления связанных с покрытием WiFi сети и определения перегруженности сетей или каналов, а так же поиск несанкционированных точек доступа уже в процессе эксплуатации беспроводной сети или беспроводного моста. В связи с большим количеством подключений конечных

пользователей по технологии WiFi перед провайдерами использующими данную технологию встает задача обеспечения качественного развертывания и дальнейшего функционирования беспроводной сети, что требует большого количества однотипного диагностического оборудования. Закупка и использование оборудования компании Fluke Networks является затратным для российских провайдеров. Использование программных анализаторов не является оптимальным решением, так как: стоимость устройства с операционной системой, обычно не обладающим возможностью подключения внешней антенны, и программного обеспечения будет больше, чем стоимость программно-аппаратного комплекса, состоящего из недорогого микроконтроллера и не сложного программного обеспечения, обеспечивающего минимально необходимый уровень диагностики беспроводной сети.

Цель выпускной квалификационной работы разработка устройства поиска свободных частот для сети беспроводной связи стандарта 802.11 WiFi 2.4ГГц.

Задачи:

1. Анализ проблемы и существующий решений по оценке эфирной обстановки WiFi сети
2. Разработка на базе ARM микроконтроллера готового устройства для анализа частот беспроводных точек доступа стандарта 802.11
3. Проведение экспериментов, получение информации об уровне сигнала как для всех диапазонов 2.4ГГц, так и для любого из каналов, с получением информации о каждой точке доступа.

Полученная информация может быть использована специалистами по развертыванию беспроводной сети для настройки сети с максимальными показателями уровня сигнала.

Устройство должно соответствовать следующим требованиям:

- Возможностью сканирования и анализа каналов беспроводных сетей диапазона 2.4 ГГц.
- Возможностью отображения на собственном экране результатов сканирования как по всему диапазону, так по выбранному.
- Возможностью отображения списка точек доступа на выбранном канале с выводом детальной информации о точках доступа, максимальной, минимальной, средней и текущей мощности принимаемого сигнала, а так же данных о зашумленности канала.
- Иметь низкое энергопотребление и возможность длительной работы от собственной аккумуляторной батареи.
- Иметь простой интерфейс, не требующий длительного обучения.
- Иметь возможность подключения внешней антенны.

1 ОПИСАНИЕ СТАНДАРТОВ IEEE 802.11

Набор стандартов 802.11 определяет целый ряд технологий реализации физического уровня (PHY), которые могут быть использованы подуровнем 802.11 MAC. К этому набору относятся:

- Физический уровень стандарта 802.11 со скачкообразной перестройкой частоты (frequency hopping) в диапазоне 2,4 ГГц.
- Физический уровень стандарта 802.11 с расширением спектра методом прямой последовательности (direct sequence) в диапазоне 2,4 ГГц.
- Физический уровень стандарта 802.11b с расширением спектра методом прямой последовательности в диапазоне 2,4 ГГц.
- Физический уровень стандарта 802.11a с разделением по ортогональным частотам (orthogonal frequency division multiplexion, OFDM) в диапазоне 5 ГГц.
- Расширенный физический уровень (extended rate physical (ERP) layer) стандарта 802.11g в диапазоне 2,4 ГГц.
- Физический уровень стандарта 802.11n (как развитие стандарта 802.11a). Основными дополнениями, внесенными в стандарт 802.11n по сравнению со стандартом 802.11a являются поддержка частоты 2.4ГГц и поддержка MIMO-режима передачи данных в стандарте 802.11n.

Подуровень MAC отвечает за разделение совместно используемого радиоканала между пользователями (терминалами, станциями), а также за обнаружение и, если необходимо, повторную передачу ошибочно принятых пакетов информации (кадров или фреймов). Эта задача решается с помощью специальных алгоритмов доступа к каналу и обмена служебной информацией. Подуровень MAC одинаков для стандартов 802.11. Станции стандарта 802.11 не обладают способностью обнаруживать коллизии, как это делают Ethernet-станции стандарта 802.3, осуществляющие множественный доступ к сети с

контролем несущей и обнаружением коллизий. Вследствие этого для доступа к среде необходим более сложный и масштабируемый подуровень MAC при минимальных дополнительных издержках. Механизм доступа стандарта 802.11 должен предпринять все усилия для того, чтобы коллизии не возникали в принципе. Основывающиеся на стандарте 802.11 беспроводные сети используют механизм, который называется «множественный доступ с контролем несущей и предотвращением коллизий» (carrier sense multiple access with collision avoidance, CSMA/CA). CSMA/CA представляет собой механизм "прослушивание перед передачей" (listen before talk, LBT). Передающая станция проверяет, присутствует ли в среде сигнал несущей и, прежде чем начать передачу, ожидает ее освобождения. При использовании технологии CSMA/CA коллизия обнаруживается только при неполучении передающей станцией ожидаемого подтверждения.

Реализация технологии CSMA/CA стандартом 802.11 осуществляется при посредстве распределенной функции координации (distribted coordination function, DCF). Основные для технологии CSMA/CA 802.11 компоненты это:

- Контроль несущей (или контроль наличия в сети сигнала от работающей станции).
- Фрагментация фреймов.
- Распределенная функция координации.
- Резервирование среды с помощью механизма «готовность к передаче/готовность к приему» (RTS/CTS).

Станция, которая намеревается осуществить передачу в беспроводной среде, должна вначале проверить, используется ли несущая, то есть какой-либо сигнал. Наличие сигнала означает, что какая-либо станция осуществляет передачу информации. Если это так, станция должна отложить свою передачу до момента освобождения среды. Станция может проверить физический уровень и убедиться в том, что несущая свободна. Но в некоторых случаях среда может быть все еще занята другой станцией через вектор распределения

сети. Это таймер, значение которого обновляется данными фреймов, передаваемых через среду. Фреймы стандарта 802.11 содержат поле продолжительности (duration field). Значение продолжительности достаточно велико для того, чтобы осуществить передачу фрейма и получить подтверждение его приема. Таким образом, станция определяет состояние среды с помощью двух методов:

Проверка физического уровня PSY на предмет наличия несущей.

Использование виртуальной функции контроля несущей – вектора распределения сети (network allocation vector, NAV).

Любая станция обновляет значение вектора распределения сети (NAV) только тогда, когда полученное значение поля продолжительности превышает значение, хранимое в ее векторе распределения сети. Например, если значение вектора распределения сети станции составляет 10 мс, она не станет обновлять свой вектор распределения сети, если получит фрейм со значением поля продолжительности 5 мс. Однако она обновит свой вектор распределения сети, если получит фрейм со значением поля продолжительности 20 мс.

Фрагментация фрейма - это выполняемая на уровне MAC функция, назначение которой - повысить надежность передачи фреймов через беспроводную среду. Под фрагментацией понимается дробление фрейма на меньшие фрагменты и передача каждого из них отдельно (рисунок 1.1). Предполагается, что вероятность успешной передачи меньшего фрагмента через зашумленную беспроводную среду выше. Получение каждого фрагмента фрейма подтверждается отдельно; следовательно, если какой-нибудь фрагмент фрейма будет передан с ошибкой или вступит в коллизию, только его придется передавать повторно, а не весь фрейм. Это увеличивает пропускную способность среды.

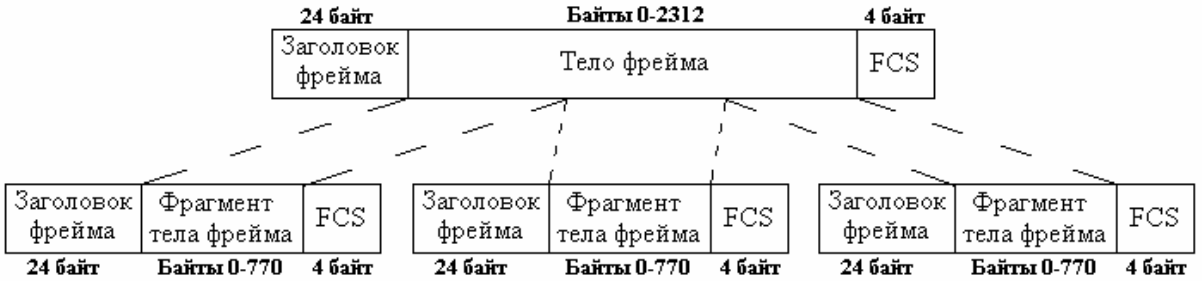

Рисунок 1.1 - Фрагментация фрейма

Размер фрагмента может задавать администратор сети. Фрагментации подвергаются только одноадресатные фреймы. Широковещательные, или многоадресатные, фреймы передаются целиком. Кроме того, фрагменты фрейма передаются пакетом, с использованием только одной итерации механизма доступа к среде DSF.

Хотя за счет фрагментации можно повысить надежность передачи фреймов в беспроводных локальных сетях, она приводит к увеличению «накладных расходов» MAC-протокола стандарта 802.11. Каждый фрагмент фрейма включает информацию, содержащуюся в заголовке 802.11 MAC, а также требует передачи соответствующего фрейма подтверждения. Это увеличивает число служебных сигналов MAC-протокола и снижает реальную производительность беспроводной станции. Фрагментация – это баланс между надежностью и непроизводительной загрузкой среды.

Станция, намеревающаяся передать фрейм, должна выждать определенное время после того, как среда освободится. По истечении интервала времени станция может принять участие в состязании за право доступа к среде. При этом существует большая вероятность того, что несколько станций одновременно попытаются начать передачу тотчас после освобождения среды, что приведет к возникновению коллизии. Основным методом доступа к среде передачи данных в IEEE 802.11 MAC является распределенная координационная функция (DCF – Distributed Coordination Function), которая реализует механизм множественного доступа с контролем

несущей и предотвращением коллизий (CSMA/CA – Carrier Sense Multiple Access with Collision Avoidance)

Стандарт IEEE 802.11a регламентирует несколько временных интервалов, которые используются распределенной координационной функцией DCF:

SlotTime = 9 мкс,

SIFSTime = 16 мкс,

DIFSTime = SIFSTime + 2*SlotTime = 34 мкс.

Станция, работающая под управлением DCF, должна следовать двум правилам:

1) станция может начинать передачу, если канал свободен в течение промежутка времени DIFS.

2) станция должна отложить передачу на случайный промежуток времени (откат), чтобы уменьшить вероятность коллизий.

Для подтверждения успешного приёма кадра (фрейма) принимающая станция передает специальный небольшой кадр, имеющий тип «подтверждение» (ACK - Acknowledgment). Передающая станция считает передачу успешной только при получении подтверждения на каждый отправленный кадр данных. Между кадром данных и его подтверждением должен использоваться наименьший из межкадровых интервалов – SIFS. Это даёт возможность завершить последовательность обмена кадрами без вмешательства третьей станции.

Если в процессе передачи фрагмента обнаруживается ошибка (передающая станция не получила подтверждения, либо проверочная последовательность CRC принятого ACK кадра не является верной), то станция должна выполнить процедуру отката перед попыткой повторной передачи фрагмента. На рисунке 1.2 продемонстрированы последовательности обмена кадрами между передающей и принимающей станцией в случае успешной передачи фрагмента и в случае ошибки. Если число попыток передачи фрагментов превысит значение MaxTransmitMSDULifetime, оставшиеся

фрагменты не передаются, и на вышележащий уровень отправляется сообщение об ошибке.

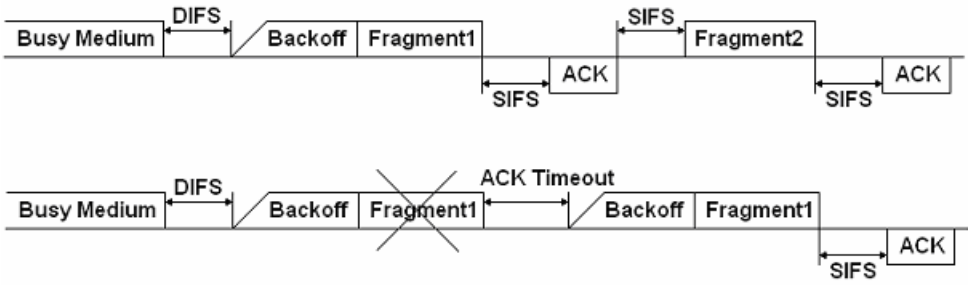

Рисунок 1.2 - Последовательность обмена при успешной передаче и потере фрагмента

Процедура отката заключается в следующем. В соответствии со стандартом IEEE 802.11 станция, осуществляющая процедуру отката, сначала должна сгенерировать случайное число – время отката и присвоить его значение таймеру отката. Однако если таймер отката (Backoff timer) уже содержит ненулевое значение, этого делать не требуется.

Backoff Time = Random*SlotTime,

где Random – псевдослучайное целое число из интервала [0, CW] (CW может принимать значения от CWmin до CWmax). Изначально величина CW принимает значение CWmin, и при неудачной попытке передачи кадра пересчитывается по формуле:

$CW = 2(CW + 1) – 1.$

При достижении CW своего максимального значения (CWmax) величина CW остаётся ему равной. В случае успешной передачи кадра CW сбрасывается на CWmin. В стандарте IEEE 802.11 CWmin и CWmax принимают значения 15 и 1023 соответственно.

Станция, осуществляющая процедуру отката, «прослушивает» канал. Если канал свободен в течение SlotTime, таймер отката уменьшается на SlotTime. Заметим, что если канал занят, станция не меняет значение таймера отката. Процедура отката будет продолжена после того, как канал будет свободен в течение промежутка времени DIFS.

Основное назначение физических уровней стандарта 802.11 - обеспечить механизмы беспроводной передачи данных, а также поддерживать выполнение вторичных функций, таких как оценка состояния беспроводной среды и сообщение о нем подуровню MAC. Стандарт 802.11 разработан таким образом, что обеспечивается независимость между подуровнем MAC и физическим (PHY) уровнем PHY. Именно независимость между MAC и PHY позволила использовать дополнительные высокоскоростные физические уровни, описанные в стандартах 802.11b, 802.11a, 802.11g и 802.11n.

Основное назначение физических уровней стандарта 802.11 - обеспечить механизмы беспроводной передачи для подуровня управления доступом к среде передачи (MAC - Medium Access Control), а также поддерживать выполнение вторичных функций, таких как оценка состояния беспроводной среды и сообщение о ней подуровню MAC. Разрабатывая эти механизмы передачи независимо от подуровня, стандарт 802.11 усовершенствовал как подуровень MAC, так и физический подуровень (PHY), а также интерфейс, поддерживаемый MAC. Именно независимость между подуровнем MAC и физическим подуровнем и позволила использовать дополнительные высокоскоростные физические уровни, описанные в стандартах 802.11b и 802.11a

Каждый из физических уровней стандарта 802.11 имеет два следующих подуровня:

Physical Layer Convergence Procedure (PLCP) - процедура определения состояния физического уровня;

Physical Medium Dependent (PMD) - подуровень физического уровня, зависящий от среды передачи.

На рисунок 1.3 показано, как эти подуровни соотносятся между собой и с вышестоящими уровнями, где подуровень LLC (logical link control) представляет собой подуровень управления логическим соединением.

Рисунок 1.3- Подуровни физического уровня модели взаимодействия открытых систем (Open System Interconnection, OSI)

Подуровень PLCP по существу является уровнем обеспечения взаимодействия, на котором осуществляется перемещение элементов данных протокола MAC (MAC protocol data units, MPDU) между MAC-станциями с использованием PMD подуровня, зависящего от среды передачи, на котором реализуется тот или иной метод передачи и приема данных через беспроводную среду. Можно считать, что подуровень PMD выполняет функцию службы беспроводной передачи; а взаимодействие этих служб осуществляется посредством процедуры определения состояния физического уровня (PLCP). Подуровни PLCP и PMD отличаются для разных вариантов стандарта 802.11.

Исходный стандарт 802.11 определяет два метода передачи данных на физическом уровне:

- технология расширения спектра путем скачкообразной перестройки частоты (FHSS) в диапазоне 2,4 ГГц.
- технология широкополосной модуляции с расширением спектра методом прямой последовательности (DSSS) в диапазоне 2,4 ГГц.
- Обе эти технологии работают в диапазоне 2,4 ГГц, в котором выделена спектральная полоса шириной 82 МГц.

Беспроводные локальные сети со скачкообразной перестройкой частоты (FHSS) поддерживают скорости передачи 1 и 2 Мбит/с. Как следует из

названия, устройства FHSS осуществляют скачкообразную перестройку частоты по заранее заданной схеме. Устройства FHSS делят предназначенную для их работы полосу частот от 2,402 до 2,480 ГГц на 79 неперекрывающихся каналов (это справедливо для Северной Америки и большей части Европы). Ширина каждого из 79 каналов составляет 1 МГц, поэтому беспроводные локальные сети FHSS используют относительно высокую скорость передачи символов, 1 МГц, и намного меньшую скорость перестройки с канала на канал. Последовательность перестройки частоты имеет следующие параметры: частота перескоков не менее 2,5 раз в секунду как минимум между 6-ю (6 МГц) каналами. Чтобы минимизировать число коллизий между перекрывающимися зонами покрытия, возможные последовательности перескоков разбиваются на три набора последовательно стей, длина которых для Северной Америки и большей части Европы составляет 26. Первый набор состоит из 0, 3, 6, 9, …, 75 частот, второй набор – из 1, 4, 7, 10, …, 76 частот, и третий набор – из 2, 5, 8, 11, …, 77 частот.

Схема обеспечивает медленный переход с одного возможного канала на другой таким образом, что после каждого скачка перекрывается полоса частот, равная как минимум 6 МГц. Благодаря этому в многосотовых сетях минимизируется возможность возникновения коллизий.

После того как уровень MAC пропускает MAC-фрейм, который в локальных беспроводных сетях FHSS называется также служебный элемент данных PLCP, или PSDU (сокращение от PLCP service data unit), подуровень PLCP добавляет два поля в начало фрейма, чтобы сформировать таким образом фрейм PPDU (PPDU - это элемент данных протокола PLCP). На рисунок 1.4 представлен формат фрейма FHSS подуровня PLCP.

Рисунок 1.4 - Формат фрейма FSSS подуровня PLCP стандарта 802.11

Преамбула фрейма PLCP состоит из двух подполей:

Подполе Sync размером 80 бит. Строка, состоящая из чередующихся 0 и 1, начинается с 0. Приемная станция использует это поле, чтобы принять решение о выборе антенны при наличии такой возможности, откорректировать уход частоты (frequency offset) и синхронизировать распределение пакетов (packet timing).

Подполе флага начала фрейма (start of frame delimiter, SFD) размером 16 бит. Состоит из строки (0000 1100 1011 1101, крайний слева бит первый). Обеспечивает синхронизацию фреймов (frame timing) для приемной станции.

Заголовок фрейма PLCP состоит из трех подполей.

Слово длины служебного элемента данных PLCP (PSDU), PSDU length word (PLW) размером 12 бит. Указывает размер фрейма MAC (PSDU) в октетах.

Сигнальное поле PLCP (signaling field PLCP, PSF) размером 4 бит, которое указывает скорость передачи данных конкретного фрейма.

Подполе CRC (циклический избыточный код) шириной 16 бит, обеспечивающее оценку правильности переданного фрейма.

Служебный элемент данных PLCP (PSDU) проходит через операцию скремблирования с целью рандомизации последовательности входных битов.

При использовании технологии FHSS применяется модуляция, основанная на гауссовом переключении частот (Gaussian frequency shift keying, GFSK). Для ее пояснения рассмотрим вначале частотную манипуляцию (frequency shift keying, FSK), которая осуществляется путем представления каждого символа соответствующей частотой. Например, двоичное значение 0 передается синусоидальным сигналом с частотой f_1, а двоичное значение 1 - сигналом с частотой f_2. Эти частоты обычно указывают относительно несущей частоты f_c. Тогда имеем, формула (1.1):

$$f_1=f_c-f_d \text{ и } f_2=f_c+f_d \qquad (1.1)$$

Преамбула фрейма PLCP состоит из двух подполей:

Подполе Sync размером 80 бит. Строка, состоящая из чередующихся 0 и 1, начинается с 0. Приемная станция использует это поле, чтобы принять решение о выборе антенны при наличии такой возможности, откорректировать уход частоты (frequency offset) и синхронизировать распределение пакетов (packet timing).

Подполе флага начала фрейма (start of frame delimiter, SFD) размером 16 бит. Со- стоит из строки (0000 1100 1011 1101, крайний слева бит первый). Обеспечивает синхронизацию фреймов (frame timing) для приемной станции.

Заголовок фрейма PLCP состоит из трех подполей:

Слово длины служебного элемента данных PLCP (PSDU), PSDU length word (PLW) размером 12 бит. Указывает размер фрейма MAC (PSDU) в октетах.

Сигнальное поле PLCP (signaling field PLCP, PSF) размером 4 бит. Указывает скорость передачи данных конкретного фрейма.

Подполе CRC (циклический избыточный код) шириной 16 бит, обеспечивающее оценку правильности переданного фрейма.

Служебный элемент данных PLCP (PSDU) проходит через операцию скремблирования с целью рандомизации последовательности входных битов.

Одна из серьезных проблем при использовании FSK, заключается в следующем. Если рассмотреть процесс передачи 0, следующего сразу же после 1, то при этом требуется, чтобы частота сигнала мгновенно изменилась со значения fc-fd на значение fc+fd. Это приводит к прерывному изменению выходного сигнала, во время которого выделяется много энергии на частотах, выходящих за частотный диапазон. Чтобы решить эту проблему, сигнал, поступающий на частотный модулятор, фильтруется для сглаживания переходов с частоты fc-fd на частоту fc+fd. В случае использования модуляции GFSK используется гауссов фильтр. В соответствии со стандартом 802.11 девиация частоты fd составляет не менее 110 кГц.

При работе на скорости 2 Мбит/с используется четверичная модуляция 4GFSK; в этом случае два бита модулируют сигнал одновременно с использованием двух девиаций частоты. При этом используется следующее преобразование пар бит в частоту: биты 10 передаются синусоидальным сигналом с частотой fc+fd2; биты 11 - сигналом с частотой fc+fd1; биты 01 - сигналом с частотой fc-fd1; биты 00 - сигналом с частотой fc-fd2. Первая девиация частоты fd1, примерно в 3 раза больше, чем вторая (fd2).

В спецификации стандарта 802.11 оговорено использование и другого физического уровня - на основе технологии широкополосной модуляции с расширением спектра методом прямой последовательности (DSSS). В соответствии со стандартом 802.11 от 1997 года, технология DSSS поддерживает скорости передачи 1 и 2 Мбит/с. Беспроводные локальные сети DSSS используют каналы шириной 22 МГц, благодаря чему многие WLAN могут работать в одной и той же зоне покрытия. В Северной Америке и большей части Европы каналы шириной 22 МГц позволяют создать в диапазоне 2,4-2,483 ГГц три неперекрывающихся канала передачи.

Аналогично подуровню PLCP, используемому в технологии FHSS, подуровень PLCP технологии DSSS стандарта 802.11 добавляет два поля во фрейм MAC, чтобы сформировать PPDU: преамбулу PLCP и заголовок PLCP. Формат фрейма представлен на рисунок 1.5

Рисунок 1.5 - Формат фрейма DSSS PPDU подуровня PLCP стандарта 802.11

Преамбула PLCP состоит из двух подполей:

Подполе Sync шириной 128 бит, представляющее собой строку, состоящую из единиц. Задача этого подполя - обеспечить синхронизацию для приемной станции.

Подполе SFD шириной 16 бит. Его задача - обеспечить синхронизацию (timing) для приемной станции.

Заголовок PLCP состоит из четырех подполей:

Подполе Signal шириной 8 бит, указывающее тип модуляции и скорость передачи для данного фрейма.

Подполе Service шириной 8 бит, зарезервировано для будущих модификаций стандарта.

Подполе Length шириной 16 бит, указывающее количество микросекунд (из диапазона 16÷216-1), необходимое для передачи части MAC фрейма.

Подполе CRC (циклический избыточный код) шириной 16 бит, обеспечивающее оценку правильности переданного фрейма

Подуровень PLCP преобразует фрейм в поток битов и передает данные на подуровень физического уровня, зависящий от среды передачи (PMD - Physical Medium Dependent). Весь PPDU проходит через процесс скремблирования с целью рандомизации данных. Скремблированная преамбула PLCP всегда передается со скоростью 1 Мбит/с, в то время как скремблированный фрейм MPDU передается со скоростью, указанной в подполе Signal. Подуровень PMD модулирует рандомизированный поток битов, используя следующие методы модуляции:

- Двоичная относительная фазовая манипуляция (differential binary phase shift keying, DBPSK) для скорости передачи 1 Мбит/с.
- Квадратурная фазовая манипуляция (quadrature phase shift key, QPSK) для скорости передачи 2 Мбит/с.

Рассмотрим процесс модуляции при использовании технологии расширения спектра методом прямой последовательности (DSSS) на подуровне физического уровня, зависящий от среды передачи (PMD - Physical Medium Dependent)

Импульсные сигналы принято характеризовать произведением длительности импульса T на ширину его спектра W. Этот параметр называют базой сигнала формула (1.2)

$$B=WT \tag{1.2}$$

Для простых импульсов база сигнала равна примерно единице. В стандарте 802.11 с технологией DSSS используются сигналы с базой, которая много больше единицы. Такие сигналы называются широкополосными. Расширение полосы сигнала достигается за счет дополнительной модуляции, которая используется как способ кодирования сигнала. Отметим, что полоса частот, занимаемая таким сигналом, все же остается малой по сравнению с несущей частотой fc, формула (1.3):

$$W<<fc \tag{1.3}$$

Можно сказать, что каждый бит заменяется или расширяется кодом, расширяющим полосу частот. Во многом благодаря такому кодированию, когда информационный бит заменяется намного большим числом бит, эта технология позволяет передавать информацию при малом соотношении сигнал/шум. При использовании DSSS переданный сигнал, по сути, усиливается за счет применения расширяющей последовательности, совместно используемой передатчиком и приемником

Беспроводные локальные сети типа DSSS особым образом кодируют данные, получая поток данных со скоростью 1 Мбит/с с канального уровня и преобразуя его в 11-мегагерцевый поток элементарных сигналов, или чипов (chip). Расширяющая спектр последовательность (ее еще называют расщепляющей (chipping) последовательностью или последовательностью Баркера) преобразует биты данных в элементарные сигналы, и имеет длину 11

бит. В случае работы на скорости 1 и 2 Мбит/с один бит данных «расширяется» до 11 бит (двоичная 1 расширяется до значения 11111111111, а двоичный 0 - до значения 00000000000). «Расширенные» биты данных затем подаются на схему "ИЛИ" либо "исключающее ИЛИ" (exclusive OR - HOR) одновременно с расширяющей последовательностью. Получившиеся в результате чипы преобразуются в символы и модулируются. Этот процесс представлен схематически на рисунке 1.6.

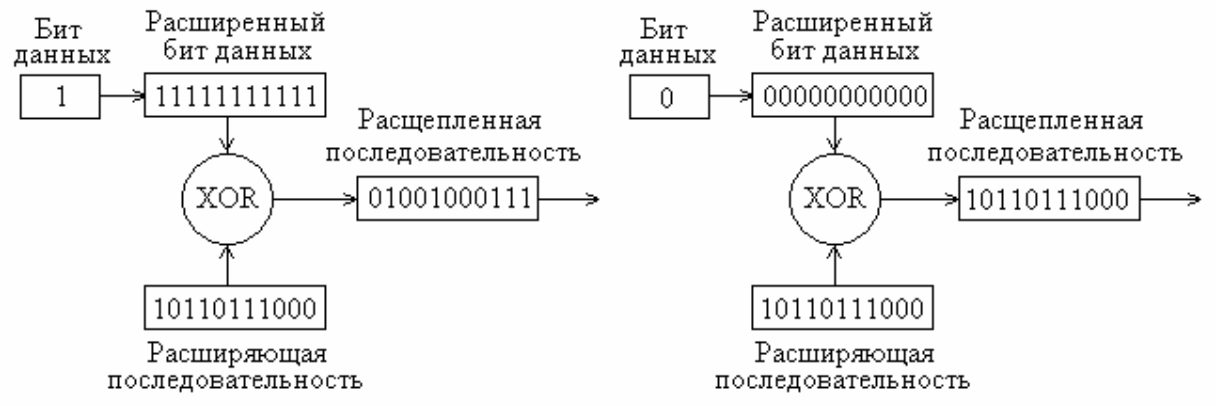

Рисунок 1.6 - Расширение битов данных для 1 и 0

Чтобы приемнику не приходилось удалять фазовую составляющую, возникающую при уходе частоты, в стандарте 802.11 используется относительная (иногда ее называют дифференциальной) двоичная фазовая модуляция (DBPSK). Относительная модуляция осуществляется следующим образом. Каждый чип преобразуется в один символ. При поступлении 0 преобразователь символов (symbol mapper) передает тот же символ, который передавался в предыдущий период передачи символов. При поступлении 1 преобразователь символов изменяет фазу на 180 градусов, или на Pi радиан.

Для достижения скорости передачи 2 Мбит/с в созвездии QPSK отображаются два чипа на символ. При этом снова используется относительная модуляция, когда символы этих двух чипов преобразуются в поворот фазы с целью реализации модуляции DQPSK. Четырем возможным вариантам

входных данных (00, 01, 10 и 11) соответствует следующие изменения фазы (в градусах): 0, 90, 180 и 270.

Передача как с использованием DBPSK, так и DQPSK приводит к необходимости передавать символы с частотой 11 МГц, но, поскольку при DQPSK каждый символ содержит два чипа, результирующая скорость передачи чипов составляет 22 МГц, что соответствует скорости передачи 2 Мбит/с

Стандарт 802.11b появился в 1999 году. Он регламентирует правила использования высокоскоростной (High Rate) технологии расширения спектра методом прямой последовательности (HR-DSSS). Такие системы обеспечивают скорость передачи в локальных беспроводных сетях диапазона 2,4 ГГц до 5,5 и 11 Мбит/с. При этом используется кодирование с использованием комплементарных кодов (complementary code keying, CCK) или технология двоичного пакетного сверточного кодирования (packet binary convolutional coding, PBCC). В технологии HR-DSSS используется та же схема организации каналов, что и в технологии DSSS: полоса частот шириной 22 МГц, 11 каналов, 3 неперекрывающихся, диапазон 2,4 ГГц. Рассмотрим, как достигаются эти повышенные скорости передачи.

Процедура PLCP (определения состояния физического уровня) технологии расширения спектра методом прямой последовательности технологии (HR-DSSS) использует фреймы PPDU двух типов: длинный и короткий. Преамбула и заголовок длинного фрейма подуровня PLCP технологии HR-DSSS всегда передаются со скоростью 1 Мбит/с для обеспечения совместимости с обычной технологией DSSS. Действительно, длинный фрейм подуровня PLCP технологии HR-DSSS почти такой же, как фрейм подуровня PLCP в технологии DSSS, но с небольшими расширениями, обеспечивающими повышенную скорость передачи данных. Эти расширения следующие:

- в подполе Signal могут быть указаны дополнительные скорости передачи данных

- (5.5 Мбит/с и 11 Мбит/с)
- подполе Service определяет ранее зарезервированные биты;
- подполе Length по-прежнему указывает количество микросекунд, необходимых для передачи PSDU.

Короткий фрейм PLCP PPDU обеспечивает минимизацию числа служебных сигналов, позволяющих передатчику и приемнику связываться друг с другом. Короткий фрейм, используемый в высокоскоростной технологией стандарта 802.11b, показан на рисунке 1.7 Он использует те же преамбулу, заголовок и формат PSDU, но заголовок PLCP передается на скорости 2 Мбит/с, в то время как PSDU передается со скоростью 2, 5.5 или 11 Мбит/с. Кроме того, его подполя модифицированы следующим образом:

- ширина поля Sync сокращена со 128 до 56 бит. Оно представляет собой строку, состоящую из одних нулей;
- поле SFD имеет ширину 16 бит и выполняет ту же функцию указания на начало фрейма, но также указывает на использование длинных или коротких заголовков. В случае коротких заголовков 16 бит передается в порядке, обратном по отношению к длинным заголовкам

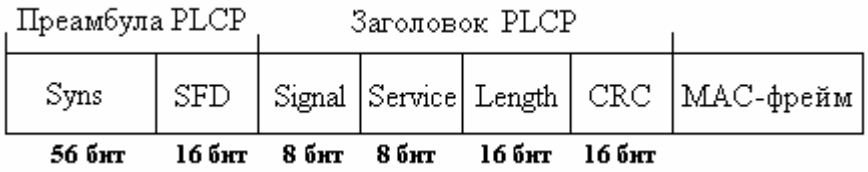

Рисунок 1.7 - Короткий PPDU фрейм технологии HR-DSSS стандарта 802.11b

Так же как и физический уровень стандарта 802.11, PLCP преобразует весь PPDU посредством аналогичной операции скремблирования, которая применяется в стандарте 802.11, на подуровне PMD.

Рассмотрим механизм расширения спектра, используемый для получения скоростей 5,5 и 11 Мбит/с при использовании модуляции комплементарных кодов (complementary code keying, CCK). Теперь расширяющий код состоит из 8 комплексных чипов (complex chip) и определяется 4 или 8 битами - в зависимости от скорости передачи данных. Скорость передачи чипов

составляет 11 Мчип/с. Таким образом, при 8 комплексных чипах на символ и 4 или 8 битах на символ достигается скорость передачи данных 5.5 и 11 Мбит/с.

Чтобы передавать данные со скоростью 5.5 Мбит/с скремблированный поток битов группируется в символы по 4 бита (b0, b1, b2 и b3). Последние два бита (b2 и b3) определяют 8 последовательностей комплексных чипов, как показано в табл. 1, где {c1, c2, c3, c4, c5, c6, c7, c8} - чипы последовательности, j – мнимая единица

Таблица 1.1 - Последовательность чипов CCK

(b2, b3)	c1	c2	c3	c4	c5	c6	c7	c8
00	j	1	j	-1	j	1	-j	1
01	-j	-1	-j	1	j	1	-j	1
10	-j	1	-j	-1	-j	1	j	1
11	j	-1	j	1	-j	1	j	1

Имея последовательность чипов, определенную битами (b2 и b3), можно использовать первые два бита (b0, b1) для определения поворота фазы, осуществляемого при дифференциальной четверичной фазовой модуляции (DQPSK), который будет применен к модулируемой последовательности (см. таблицу 1.2). Необходимо также пронумеровать каждый 4-битовый символ PSDU, начиная с 0, чтобы можно было определить, преобразуется четный либо нечетный символ в соответствии с этой таблицей. В стандарте используется дифференциальная DQPSK модуляция, а не обычная QPSK модуляция. Поэтому представленные в таблице изменения фазы отсчитываются по отношению к предыдущему символу или, в случае первого символа подполя PSDU, по отношению к последнему символу предыдущего DQPSK символа, передаваемого со скоростью 2 Мбит/с. Этот поворот фазы применяется по отношению к 8 комплексным чипам символа. Затем осуществляется модуляция на несущей частоте. Напомним, что PSDU (сокращение от PLCP service data unit) – это MAC-фрейм, который во WLAN- сетях называется также служебным элементом данных подуровня PLCP

Таблица 1.2 - Поворот фазы при модуляции CCK

(b0, b1)	Изменение фазы четных	Изменение фазы нечетных
00	0	18
01	90	-90
10	18	0
11	-90	90

Для передачи данных со скоростью 11 Мбит/с, скремблированная последовательность битов PSDU разбивается на группы по 8 символов. Последние 6 битов выбирают одну последовательность, состоящую из 8 комплексных чипов, из числа 64 возможных последовательностей, так же, как использовались биты (b2 и b3) для выбора одной из четырех возможных последовательностей. Биты (b0, b1) используются таким же образом, как при модуляции CCK на скорости 5,5 Мбит/с для поворота фазы последовательности и дальнейшей модуляции на несущей частоте.

Высокоскоростной стандарт HR-DSSS определяет также опциональный механизм модуляции для передачи данных со скоростью 5.5 и 11 Мбит/с. Такая модуляция отличается как от CCK, так и от DSSS стандарта 802.11. Вначале скремблированные биты PSDU передаются на двоичный сверточный кодер со скоростью кодирования ½, который имеет шесть линий задержки (или запоминающих ячеек), и выдает 2 бита на каждый входной бит. Поскольку стандарт 802.11 рассчитан на использование фреймов и сверточные кодеры имеют память, все элементы задержки обнуляются с началом фрейма, а в его конец добавляется 8 нулей.

Кодированный поток битов пропускается через преобразователь символов (symbol mapper) BPSK, чтобы достичь скорости передачи данных 5.5 Мбит/с, или через преобразователь символов QPSK, чтобы реализовать передачу со скоростью 11 Мбит/с. Преобразование символов, используемое в данном случае, зависит от двоичного значения, s, поступающего от 256-битовой псевдослучайной последовательности. Как преобразуются два символа QPSK,

показано на рисунке 1.8, а как преобразуются два символа BPSK - на рисунке 1.8. Для PSDU размером более 256 бит псевдослучайная последовательность просто повторяется

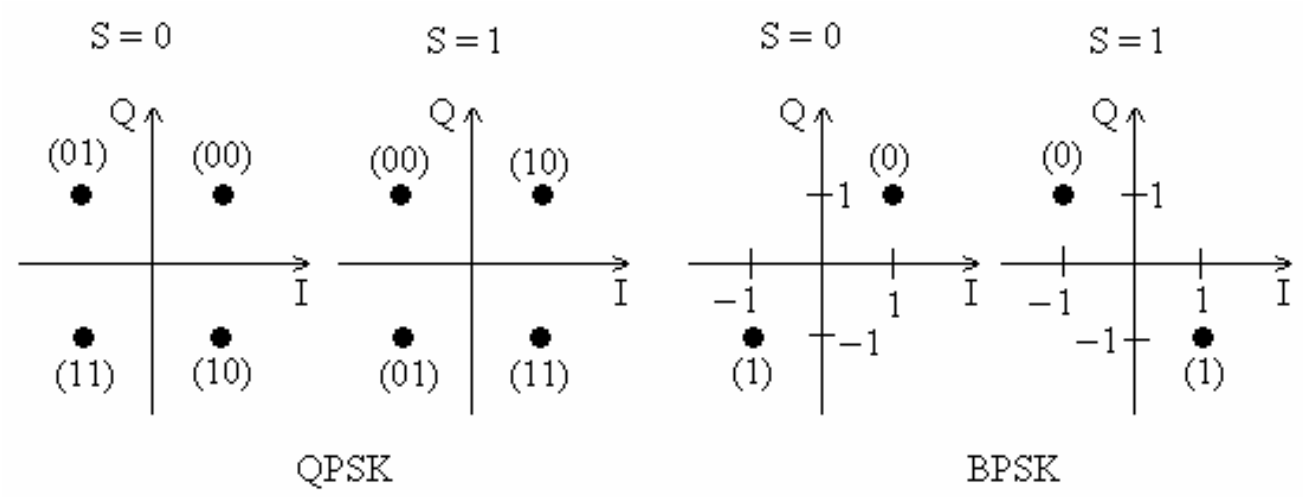

Рисунок 1.8- Преобразование символов PBCC BPSK (слева) и QPSK (справа) высокоскоростного стандарта 802.11b

Уровень излучаемой мощности не должен превышать 1000 мВт (США), 100 мВт при условии изотропно-излучаемой мощности (Европа), 10 мВт/МГц (Япония).

Спектральная плотность мощности передаваемого сигнала должна быть в пределах спектральной маски, в соответствии с которой спектр может быть прямоугольным в диапазоне частот шириной до 22 МГц. В диапазонах частот (fc–22 МГц < f < fc–11 МГц); (fc+11 МГц < f < fc+22 МГц) спектральные составляющие не должны превышать уровня – 30 дБ относительно максимума (fc – несущая частота). Вне этих частот уровень спектра не должен быть выше уровня -50 дБ. Отклонение несущей частоты не должно превышать ±25*10-6. Спектральная маска стандарта 802.11b показана на рисунке 1.9.

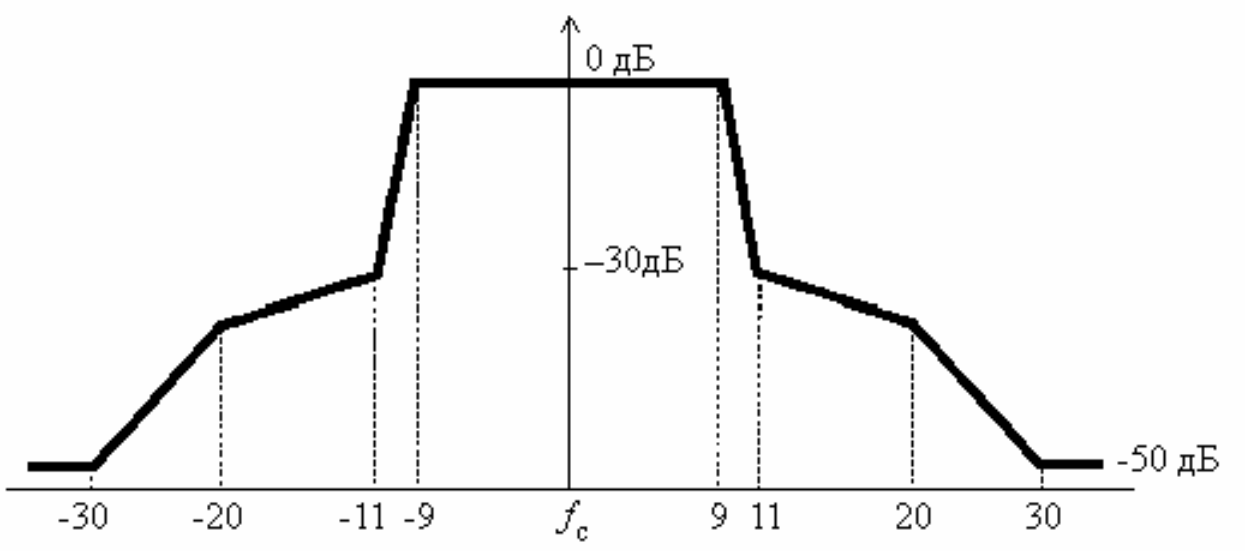

Рисунок 1.9- Маска стандарта 802.11b

На рисунке 1.9 показаны спектральные маски излучения для 11 каналов, используемых в США. Видно, несмотря на наличие 11 каналов, на самом деле только три из них (1, 6 и 11) не перекрывают друг друга

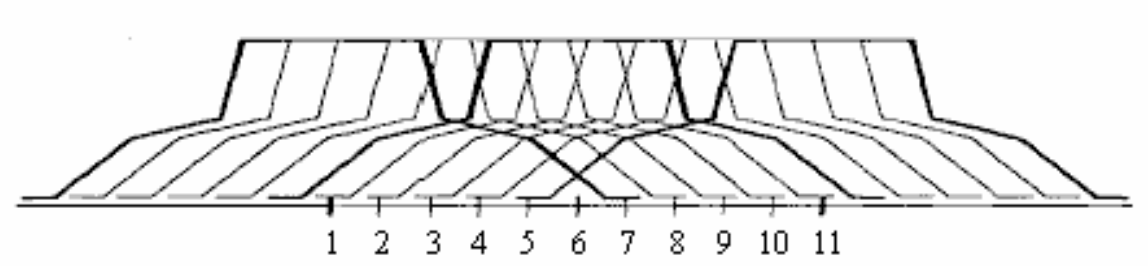

Рисунок 1.10- Неперекрывающиеся каналы стандарта 802.11b

Стандарт IEEE 802.11g, предложенный в июне 2003 года, определяет технологию физического уровня с увеличенной скоростью передачи (ERP extended rate PHY) как средство обеспечения скоростей передачи до 54 Мбит/с в диапазоне 2,4 ГГц. Этот стандарт основан на методе OFDM, используемом в стандарте 802.11a. Однако в отличие от стандарта 802.11a он обеспечивает совместимость со стандартом 802.11b. Данная совместимость обеспечивается тем, что устройства, соответствующие стандарту 802.11g, могут изменять скорость передачи данных до значений, меньших, чем регламентированы стандартом 802.11b.

В стандарте IEEE 802.11g определены три схемы модуляции:

- ERP-OFDM (физический уровень с расширенным набором скоростей и с использованием OFDM модуляции).
- ERP-CCK (физический уровень с расширенным набором скоростей и с использованием CCK модуляции).
- DSSS-OFDM (физический уровень с расширением спектра методом прямой последовательности и с использованием OFDM модуляции).

При использовании модуляции ERP-OFDM применяются специально разработанные для этой модуляции механизмы, обеспечивающие скорость передачи 6, 9, 12, 18, 24, 36, 48, и 54 Мбит/с. Обязательными являются скорости 6, 12 и 24 Мбит/с в дополнение к скоростям передачи данных 1; 2; 5,5 и 11 Мбит/с. Стандарт также позволяет опционально использовать двоичное сверточное кодирование (packet binary convolutional coding – PBCC) со скоростями 22 и 33 Мбит/с, и также опционально технологию расширения спектра методом прямой последовательности с передачей на ортогональных частотных поднесущих (DSSS-OFDM) со скоростями 6, 9, 12, 18, 24, 36, 48 и 54 Мбит/с.

Стандарт 802.11g определяет пять PLCP:

1) с длинной преамбулой;

2) с короткой преамбулой

3) преамбулой ERP-OFDM

4) длинной преамбулой DSSS-OFDM

5) короткой преамбулой DSSS-OFDM. Поддержка первых трех подуровней обязательна, а двух последних - опциональна. В табл. 2.3 приведены различные преамбулы и схемы модуляции, а также скорости передачи данных, которые они поддерживают или с которыми взаимодействуют.

Таблица 1.3 - Преамбулы стандарта 802.11g

Тип преамбулы	Скорости передачи данных; поддерживают или
Длинная	1; 2;, 5,5 и 11 Мбит/с; DSSS-OFDM на всех скоростях OFDM; ERP-PBCC на всех скоростях
Короткая	2;, 5,5 и 11 Мбит/с; DSSS-OFDM на всех скоростях OFDM; ERP-PBCC на всех скоростях
ERP-OFDM	ERP-OFDM на всех скоростях
Длинная DSSS-OFDM	DSSS-OFDM на всех скоростях
Короткая DSSS-	DSSS-OFDM на всех скоростях

Длинная преамбула использует ту же самую длинную преамбулу, что определенна для HR-DSSS, но ее поле Service модифицировано так, как показано в таблице 1.4

Таблица 1.4- Определения полей для ERP Service

Бит	Наименование	Назначение
b0	Зарезервирован	0
b1	Зарезервирован	0
b2	Блокировка	0 - не заблокированы, 1 -
b3	Выбор модуляции	0 - не ERP-PBCC, 1 - ERP-PBCC
b4	Зарезервирован	0
b5	Расширение длины	Для ERP-PBCC
b6	Расширение длины	Для ERP-PBCC
b7	Расширение длины	Для PBCC

Биты расширения длины определяют число октетов, когда используются режимы PBCC на скорости 11 Мбит/с и ERP-PBCC на скоростях 22 и 33 Мбит/с.

Короткая преамбула так же модифицируется по отношению к аналогичной преамбуле при HR-DSSS, как указано в таблице 1.3.

Преамбула ERP-OFDM использует таковую стандарта 802.11a и расширяет сигнал на дополнительные 6 мкс, в течение которых не происходит никакая передача данных, чтобы сделать пакет длиннее для согласования его с более длинным 16- микросекундным синхронизатором SIFS стандарта 802.11a по сравнению с 10- микросекундным синхронизатором SIFS стандарта 802.11b.

Формат длинной преамбулы PPDU технологии CCK-OFDM представлен на рисунок 1.11 В подполе Rate поля Signal устанавливается скорость 3 Мбит/с. Благодаря такой установке обеспечивается совместимость со станциями, не поддерживающими EPR, потому что они по-прежнему считывают значение поля Length и откладывают передачу, несмотря на то, что не способны демодулировать полезную нагрузку. Заголовок PLCP соответствует заголовку PLCP предварительно определенной длинной преамбулы, но эта преамбула точно такая же, как для режима HR-DSSS. И заголовок, и преамбула передаются со скоростью 1 Мбит/с с использованием DBPSK, а PSDU передается с использованием подходящей скорости передачи данных OFDM. Заголовок скремблируется посредством скремблера HR-DSSS, а символы данных скремблируются с помощью скремблера стандарта 802.11a.

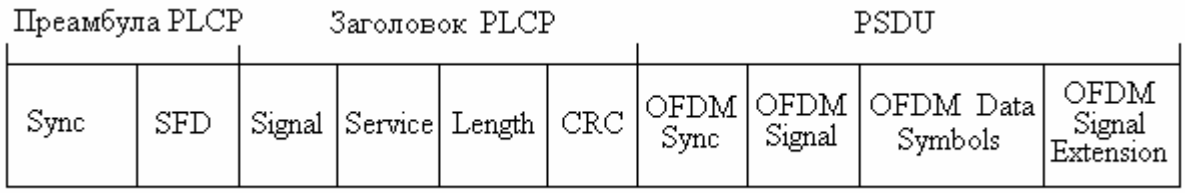

Рисунок 1.11-Формат длинной преамбулы PPDU технологии CCK-OFDM

В формате короткой преамбулы PPDU технологии DSSS-OFDM, аналогично длинной преамбуле DSSS-OFDM, используются короткая преамбула HR-DSSS и заголовок при скорости передачи данных 2 Мбит/с. Посредством скремблера HR-DSSS и символов данных короткая преамбула и заголовок передаются с использованием технологии OFDM и используют скремблер стандарта 802.11a.

ERP-OFDM обеспечивает механизм для использования скоростей передачи данных стандарта 802.11a в диапазоне 2,4 ГГц таким образом, что обеспечивается совместимость с технологиями DSSS и HR-DSSS. В дополнение к использованию модуляции OFDM стандарта 802.11a по схеме распределения частот диапазона ERP-OFDM также устанавливает, что центральная частота передачи и тактовая частота символов определяются тем же генератором, который был опциональным для DSSS. Он использует канальный интервал длительностью 20 мкс, но она может быть уменьшена до 9 мкс, если выяснится, что в BSS находятся только устройства ERP.

Для передачи данных с более высокими скоростями, 22 и 33 Мбит/с, технология двоичного пакетного сверточного кодирования (PBCC) использует тот же механизм, что и на меньших скоростях 5,5 и 11 Мбит/с PBCC, но с использованием восьмеричной фазовой модуляции (8-PCK) вместо QPCK и BPCK для достижения скорости 22 Мбит/с. Скорость 33 Мбит/с достигается за счет применения генератора с частотой 16,5 МГц вместо генератора с частотой 11 МГц. Схема отображения символов для 8-PCK показана на рисунок 1.12.

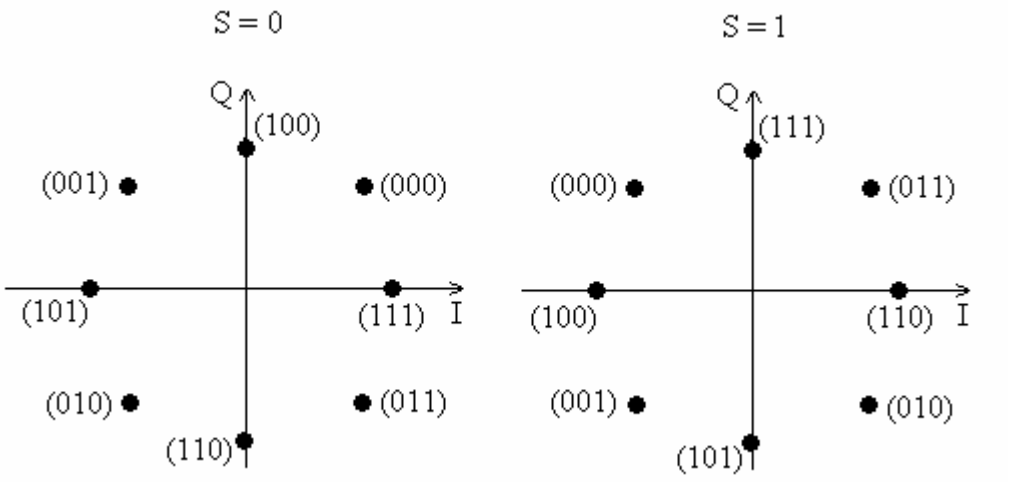

Рисунок 1.12- Сигнальное созвездие ERP-PBCC стандарта 802.11g

Таким образом, что следует иметь в виду относительно стандарта 802.11g, состоит в следующем. Этот стандарт увеличивает поддерживаемые скорости передачи данных в диапазоне 2,4 ГГц до 54 Мбит/с способом, обеспечивающим совместимость со «старыми» устройствами,

соответствующими стандарту 802.11b. Если в локальной сети используются только устройства стандарта 802.11g, передача осуществляется с наиболее возможной скоростью. Однако, если в нее вводятся устройства стандарта 802.11b, информация заголовков должна передаваться со скоростями стандарта 802.11b, чтобы их могли понимать эти «старые» устройства. Такое снижение скорости должно выполняться при всех передачах, независимо от того, происходят они между устройствами стандарта 802.11g или стандарта 802.11b.

Различные стандарты семейства 802.11 определяют пять режимов оценки незанятости канала (CCA).

- Решение о незанятости основывается на выявлении в канале энергии, превосходящей некоторое пороговое значение
- Решение о незанятости основывается на обнаружении сигнала несущей, соответствующей стандарту 802.11.
- Обнаружение несущей и выявление энергии (комбинация способов 1 и 2).
- Обнаружение несущей с сообщениями о том, что среда не занята, если никакой сигнал не обнаружен в течение 3,65 мс.
- Выявление энергии, соответствующей повышенным скоростям передачи на физическом уровне, и обнаружение несущей по способу 3, но применительно к ERP.

В стандарте указано, что процесс CCA должен применять, по крайней мере, один из названных методов.

В таблице 1.5 даны основные параметры различных технологий, применяемых на физическом уровне стандартов 802.11. Несмотря на то, что технология FHSS распространялась очень быстро, ее догоняют технологии DSSS и HR-DSSS. В настоящее время пользователи, в основном, переходят к использованию устройств на основе стандартов 802.11a и 802.11g

Таблица 1.5- Физический уровень стандартов 802.11

	Физический уровень стандартов 802.11					
Параметр	802.11 FHSS	802.11 DSSS	802.11b HR-DSSS	802.11a OFDM	802.11g FHSS	802.11n OFDM-MIMO
Частотный диапазон, ГГц	2,4	2,4	2,4	5	5	2,4-5
Максимальная скорость передачи данных, Мбит/с	2	2	11	54	54	600
Тип модуляции	QPSK	GFSK	CCK	OFDM	OFDM	OFDM

Стандарт 802.11n для сетей Wi-Fi был утвержден организацией IEEE 11 сентября 2009 года и включает в себя множество усовершенствований по сравнению с устройствами стандарта 802.11g. Устройства 802.11n могут работать в одном из двух диапазонов 2.4 или 5.0 ГГц.

На физическом уровне (PHY) реализована усовершенствованная обработка сигнала и модуляции, а так же добавлена возможность одновременной передачи сигнала через четыре антенны.

На канальном подуровне управления доступом к среде (MAC) реализовано более эффективное использование доступной пропускной способности. Вместе эти усовершенствования позволяют увеличить максимальную теоретическую скорость передачи данных до 600 Мбит/с – увеличение более чем в десять раз, по сравнению с 54 Мбит/с стандарта 802.11a/g.

В реальности, производительность беспроводной локальной сети зависит от многочисленных факторов, таких как среда передачи данных, частота радиоволн, размещение устройств и их конфигурация.

Одним из основных моментов стандарта 802.11n является поддержка технологии MIMO (Multiple-Input Multiple-Output, Многоканальный вход/выход).

С помощью технологии MIMO реализована способность одновременного приема/передачи нескольких потоков данных через несколько антенн, вместо одной.

Стандарт 802.11n определяет различные антенные конфигурации "MxN", начиная с "1x1" до "4x4" (самые распространенные на сегодняшний день это конфигурации "3x3" или "2x3"). Первое число (M) определяет количество передающих антенн (T), а второе число (N) определяет количество приемных антенн (R). Например, точка доступа с двумя передающими и тремя приемными антеннами является "2x3" (или 2T3R) MIMO-устройством.

Чем больше устройство 802.11n использует антенн для одновременной работы передачи/приема, тем будет выше максимальная скорость передачи данных. Однако, само по себе использование нескольких антенн не увеличивает скорость передачи данных или расширение диапазона. Основным в устройствах стандарта 802.11n является то, что в них реализован усовершенствованный метод обработки сигнала, который и определяет алгоритм работы MIMO-устройства при использовании нескольких антенн.

Конфигурация "4x4" при использовании модуляции 64-QAM обеспечивает скорость до 600 Мбит/с, конфигурация "3x3" при использовании модуляции 64-QAM обеспечивает скорость до 450 Мбит/с, в то время как конфигурации "2x3" и "1x2" обеспечат скорость до 300 Мбит/с.

Еще одной особенностью стандарта 802.11n является увеличение ширины канала с 20 до 40 МГц. Большинство беспроводных локальных сетей 802.11n используют каналы 40 МГц только в диапазоне частот 5 ГГц. В сетях, использующих полосу частот 5 ГГц (802.11n), проблемы пересекающихся каналов не существует.

Устройства стандарта 802.11n могут использовать ширину канала 20 или 40 МГц в любом частотном диапазоне (2.4 или 5 ГГц). При использовании ширины канала 40 МГц (устройства 802.11n) происходит двойное увеличение

пропускной способности по сравнению с шириной канала 20 МГц (устройства 802.11b/g).

В полосе частот 5 ГГц доступно 19 непересекающихся каналов, которые более пригодны для применения в устройствах стандарта 802.11n, обеспечивающих максимально возможную скорость передачи данных. Сигналы распределяются без взаимного перекрытия каналов с шириной полосы 40 МГц, но при использовании полосы 40 МГц устройствами 802.11n, их работе могут мешать существующие 802.11b/g точки доступа, что приведет к снижению производительности всего сегмента сети.

Существуют три режима работы 802.11n: HT, Non-HT и HT Mixed.

Режим High Throughput (HT), известный также как "чистый" режим (Greenfield-режим), который предполагает отсутствие поблизости (в зоне покрытия) работающих устройств 802.11b/g, использующих ту же полосу частот. Если же такие устройства существуют в зоне покрытия, то они не смогут общаться с точкой доступа 802.11n. Таким образом, в этом режиме разрешены к использованию только клиенты 802.11n, что позволит воспользоваться преимуществами повышенной скорости и увеличенной дальностью передачи данных, обеспечиваемыми стандартом 802.11n.

Режим Non-HT, отправляет все кадры в формате 802.11b/g, чтобы устаревшие станции смогли понять их. В этом режиме точка доступа должна использовать ширину каналов 20 МГц и при этом не будет использовать преимущества стандарта 802.11n. Для обеспечения обратной совместимости все устройства должны поддерживать этот режим. Нужно учитывать, что точка доступа 802.11n с использованием режима Non-HT не будет обеспечивать высокую производительность. При использовании этого режима передача данных осуществляется со скоростью, поддерживаемой самым медленным устройством.

Режим HT Mixed будет наиболее распространенным режимом для точек доступа 802.11n в ближайшие несколько лет. В этом режиме,

усовершенствования стандарта 802.11n могут быть использованы одновременно с существующими станциями 802.11b/g. Режим HT Mixed обеспечит обратную совместимость устройств, но устройства 802.11n получат уменьшение пропускной способности. В этом режиме точка доступа 802.11n распознает наличие старых клиентов и будет использовать более низкую скорость передачи данных, пока старое устройство осуществляет прием-передачу данных.

Стандарт 802.11n определяет Индекс модуляции и схемы кодирования MCS (Modulation and Coding Scheme). MCS - простое целое число, присваиваемое каждому варианту модуляции (всего возможно 77 вариантов). Каждый вариант определяет тип модуляции радиочастоты (Type), скорость кодирования (Coding Rate), защитный интервал (Short Guard Interval – SGI или GI) и значения скорости передачи данных. Сочетание всех этих факторов определяет реальную физическую (PHY) или канальную скорость передачи данных, начиная от 6,5 Мбит/с до 600 Мбит/с (данная скорость может быть достигнута за счет использования всех возможных опций стандарта 802.11n).

Некоторые значения индекса MCS определенны и показаны в таблице 1.6.

Новые методы модуляции (64-QAM) и кодирования, как правило, более эффективные и поддерживают более высокие скорости передачи данных, но устаревшие методы и скорости все еще поддерживаются для обратной совместимости.

Для достижения максимальной скорости соединения 300 Мбит/с необходимо, чтобы и точка доступа и беспроводной адаптер поддерживали два пространственных потока (Spatial Streams) и удвоенную ширину канала 40 МГц. Исходя из полученной скорости соединения по приведенной выше таблице можно точно определить сколько потоков и какая ширина канала были задействованы. Так скорости соединения 65 или 130 Мбит/с говорят от том, что одно из устройств точка доступа или адаптер используют одинарную ширину канала 20 МГц.

Таблица 1.6- Индекс модуляции и схемы кодирования стандарта 802.11n

Индекс MSC	Тип модуляции	Темп кодирования	Количество пространственных потоков	Максимальная скорость передачи (Мбит/с) при ширине канал 20 МГц		Максимальная скорость передачи (Мбит/с) при ширине канал 40 МГц	
				800 нс (GI)	400 нс (SGI)	800 нс (GI)	400 нс (SGI)
0	BPSK	1/2	1	6.50	7.20	13.50	15.00
1	QPSK	1/2	1	13.00	14.40	27.00	30.00
2	QPSK	3/4	1	19.50	21.70	40.50	45.00
3	16-QAM	1/2	1	26.00	28.90	54.00	60.00
4	16-QAM	3/4	1	39.00	43.30	81.00	90.00
5	64-QAM	2/3	1	52.00	57.80	108.00	120.00
6	64-QAM	3/4	1	58.50	65.00	121.00	135.00
7	64-QAM	5/6	1	65.00	72.20	135.00	150.00
8	BPSK	1/2	2	13.00	14.40	27.00	30.00
9	QPSK	1/2	2	26.00	28.90	54.00	60.00
10	QPSK	3/4	2	39.00	43.30	81.00	90.00
11	16-QAM	1/2	2	52.00	57.80	108.00	120.00
12	16-QAM	3/4	2	78.00	86.70	162.00	180.00
13	64-QAM	2/3	2	104.00	115.60	216.00	240.00
14	64-QAM	3/4	2	117.00	130.00	243.00	270.00
15	64-QAM	5/6	2	130.00	144.40	270.00	300.00
16	BPSK	1/2	3	19.50	21.70	40.50	45.00
17	QPSK	1/2	3	39.00	43.30	81.00	90.00
18	QPSK	3/4	3	58.50	65.00	121.50	135.00
19	16-QAM	1/2	3	78.00	86.70	162.00	180.00
20	16-QAM	3/4	3	117.00	130.00	243.00	270.00
21	64-QAM	2/3	3	156.00	173.30	324.00	360.00
22	64-QAM	3/4	3	175.50	195.00	364.50	405.00
23	64-QAM	5/6	3	195.00	216.70	405.00	450.00
24	BPSK	1/2	4	26.00	28.90	54.00	60.00
25	QPSK	1/2	4	52.00	57.80	108.00	120.00
26	QPSK	3/4	4	78.00	86.70	162.00	180.00
27	16-QAM	1/2	4	104.00	115.60	216.00	240.00
28	16-QAM	3/4	4	156.00	173.30	324.00	360.00
29	64-QAM	2/3	4	208.00	231.10	432.00	480.00
30	64-QAM	3/4	4	234.00	260.00	486.00	540.00
31	64-QAM	5/6	4	260.00	288.90	540.00	600.00

MCS значения от 0 до 31 определяют тип модуляции и схемы кодирования, которые будут использоваться для всех потоков. MCS значения с 32 по 77 описывают смешанные комбинации, которые могут быть использованы для модуляций от двух до четырех пространственных потоков.

Точки доступа 802.11n должны поддерживать MCS значения от 0 до 15, в то время как 802.11n станции должны поддерживать MCS значения от 0 до 7. Все другие значения MCS, в том числе связанные с каналами шириной 40 МГц, коротким защитным интервалом (SGI), являются опциональными. Определение значения MCS и SGI для всех используемых устройств 802.11n, является хорошим способом для определения набора скоростей передачи данных, которые могут быть использованы беспроводной сетью.

2 РЕАЛИЗАЦИЯ УСТРОЙСТВА АНАЛИЗА ЧАСТОТНОГО РЕСУРСА Wi-Fi СЕТИ

2.1 Реализация алгоритмов для устройства анализа частотного ресурса

SDL диаграммы для разработанных алгоритмов представлены на рисунках 2.1 -2.16 инициализация основных параметров

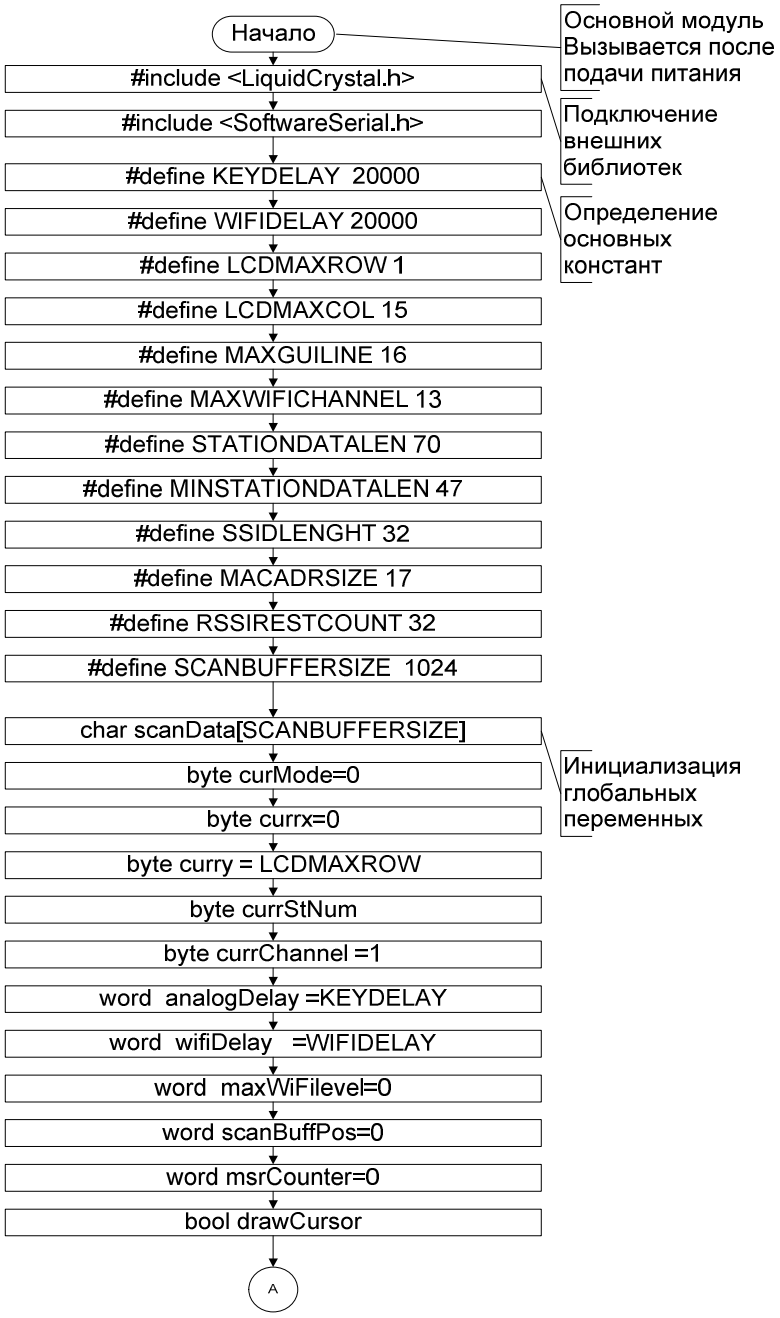

Рисунок 2.1- SDL диаграмма инициализации устройства часть 1

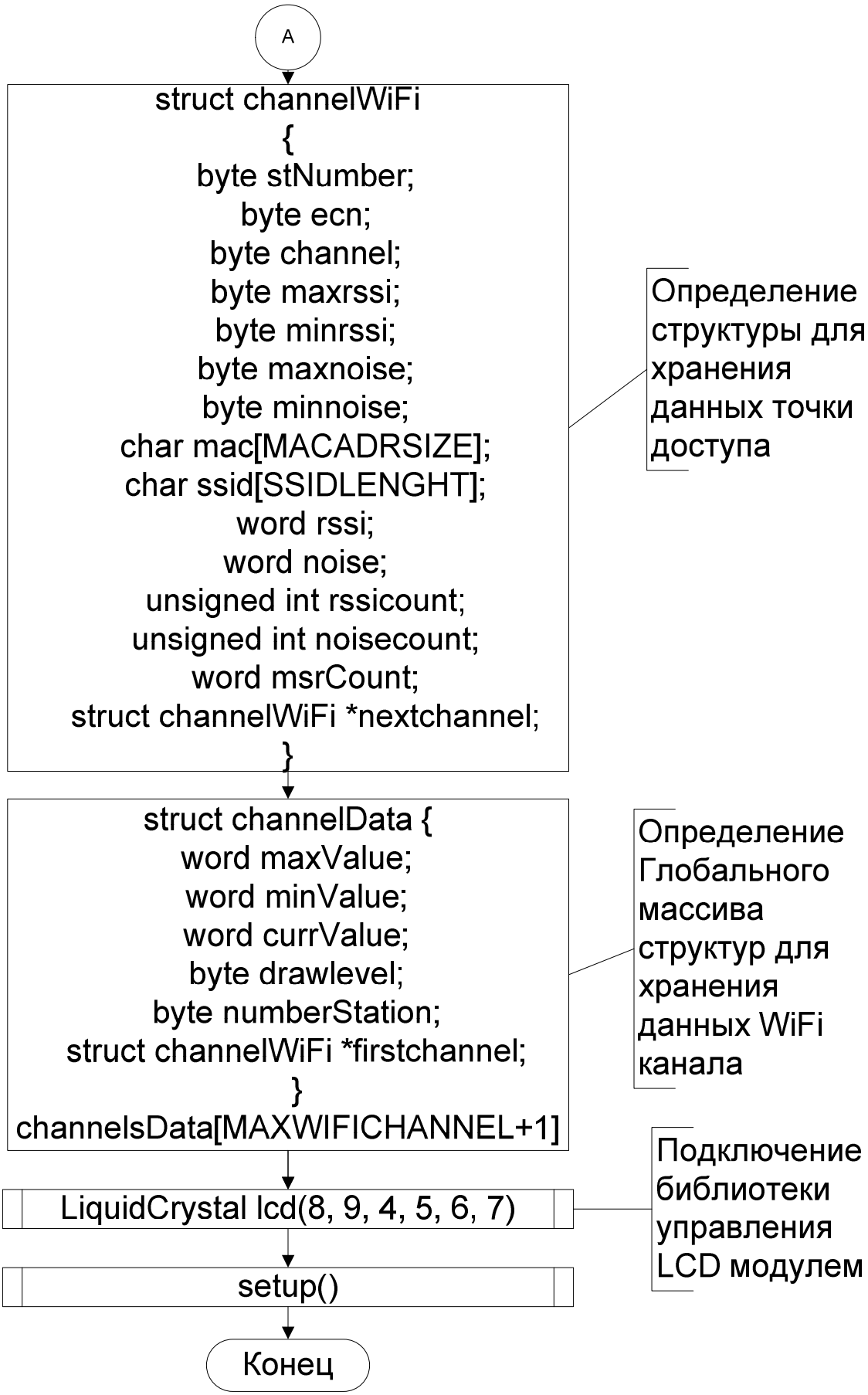

Рисунок 2.2- SDL диаграмма инициализации устройства часть 2

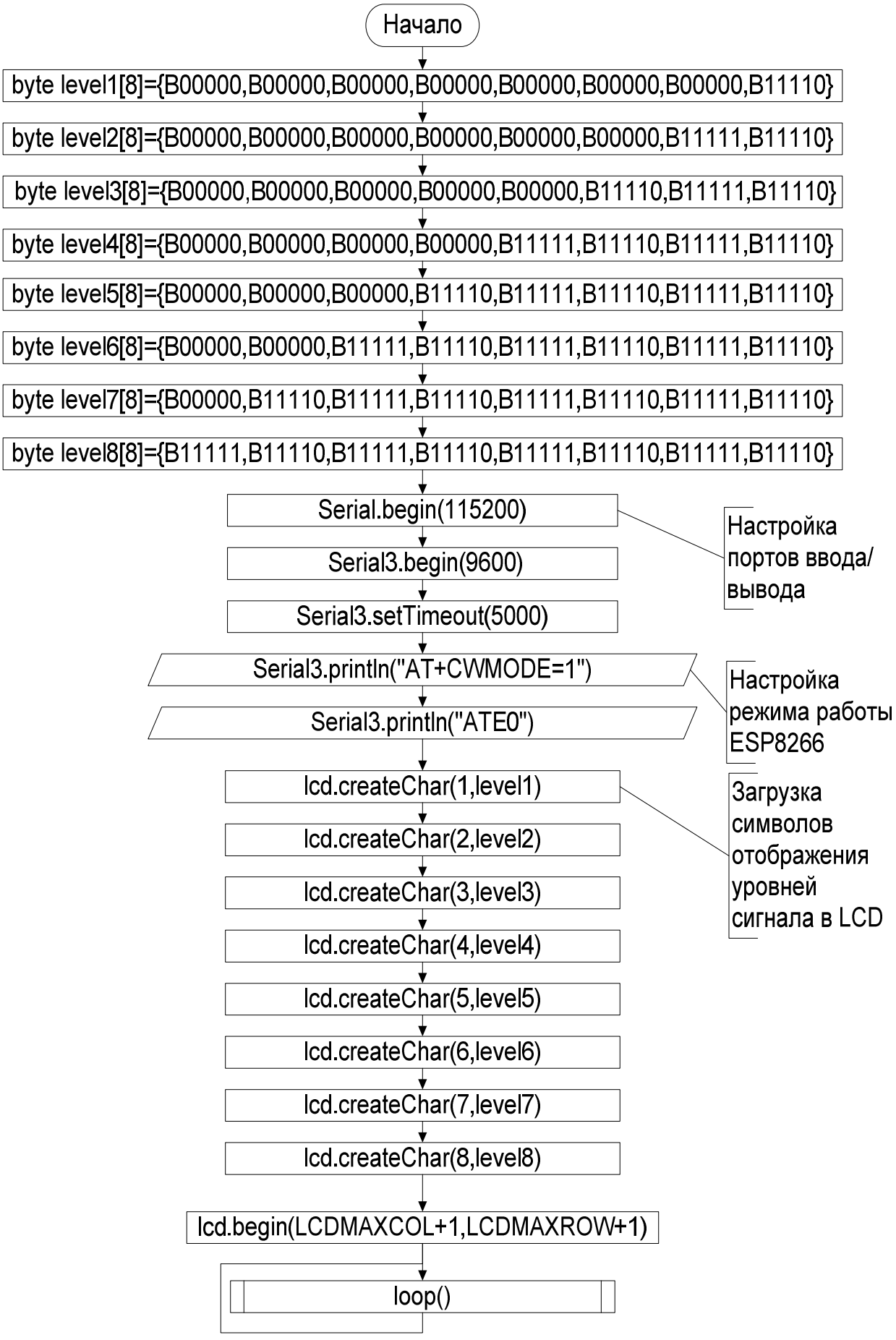

Рисунок 2.3- SDL диаграмма инициализации подпрограммы setup

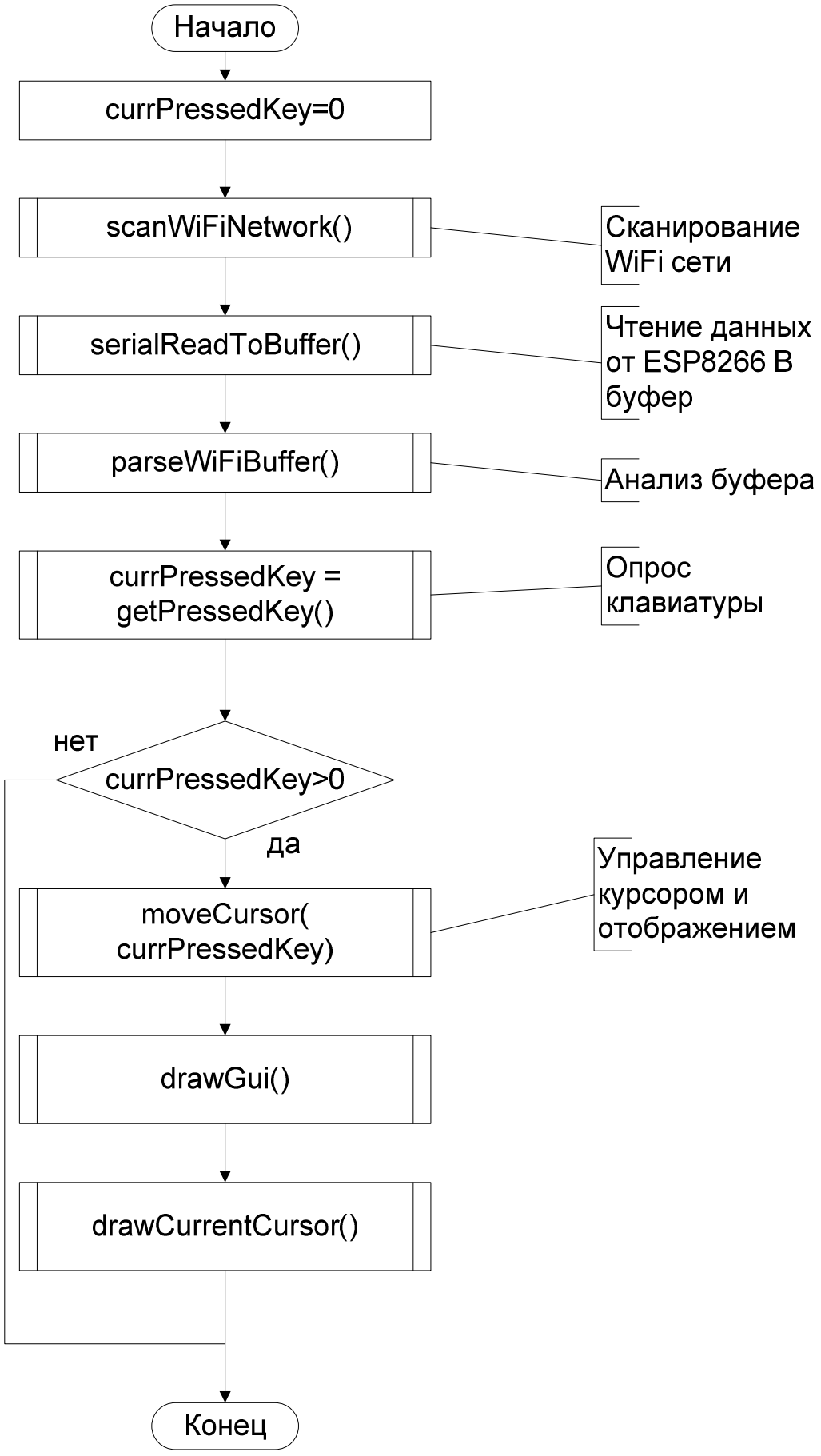

Рисунок 2.4 - SDL диаграмма инициализации подпрограммы loop

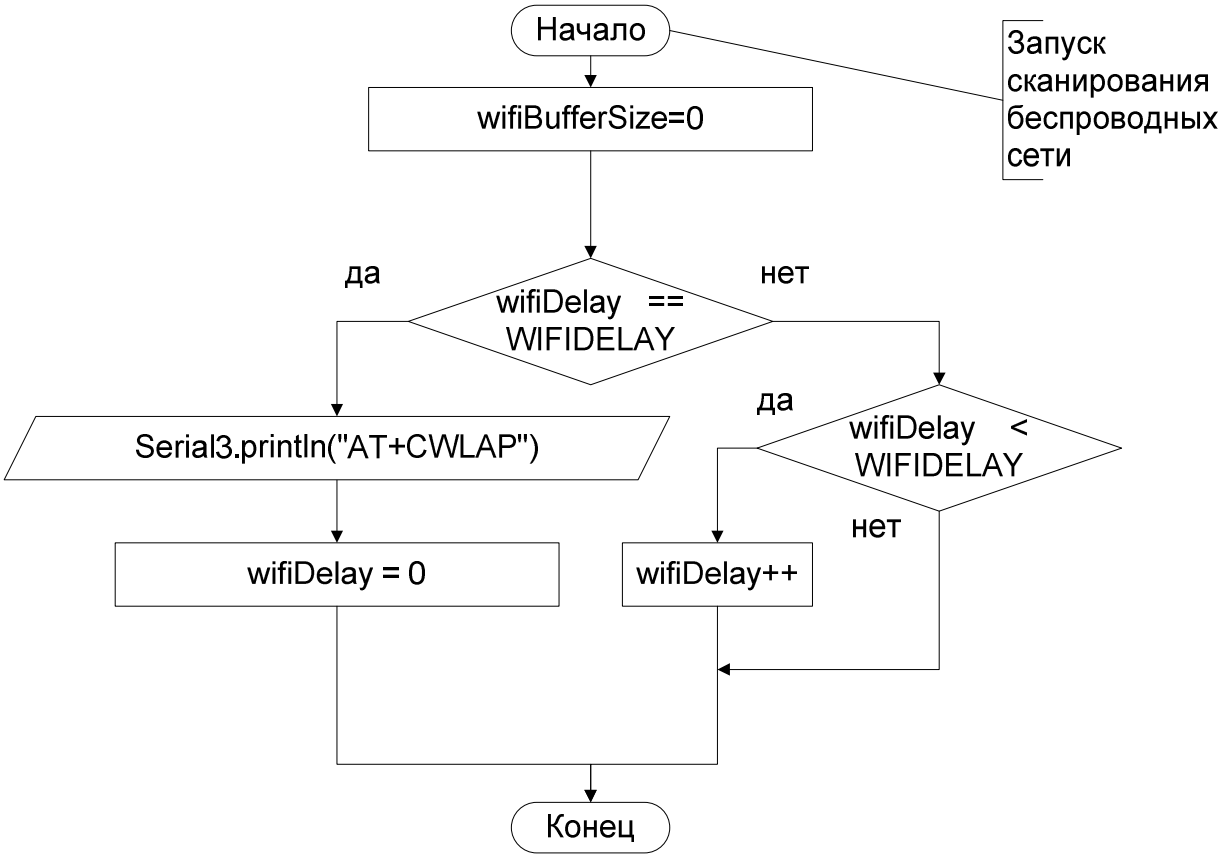

Рисунок 2.5 - SDL диаграмма инициализации подпрограммы scanWiFiNetwork

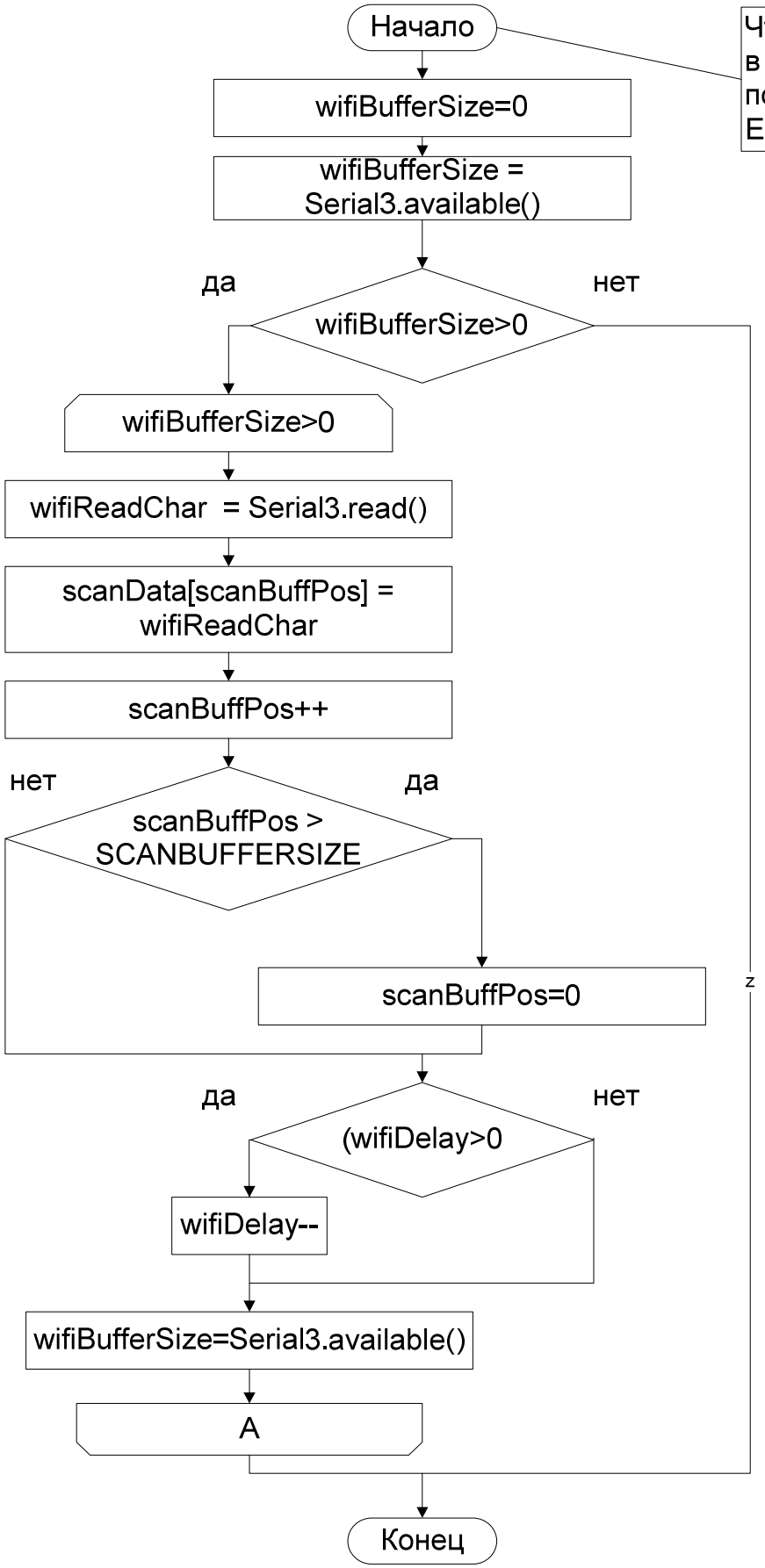

Рисунок 2.6 - SDL диаграмма инициализации подпрограммы serialReadToBuffer

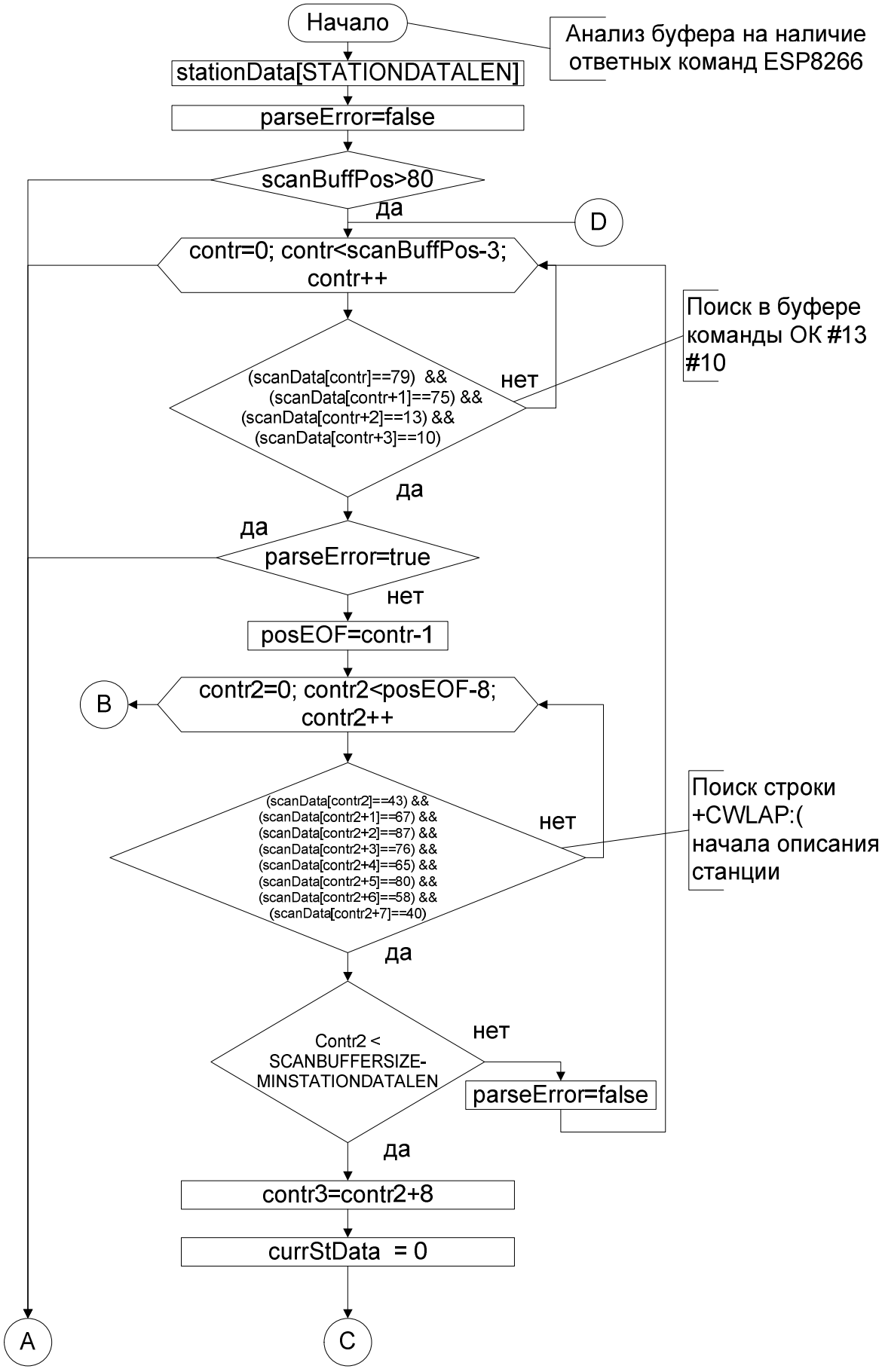

Рисунок 2.7 - SDL диаграмма инициализации подпрограммы parseWiFiBuffer 1

Рисунок 2.8 - SDL диаграмма инициализации подпрограммы parseWiFiBuffer 2

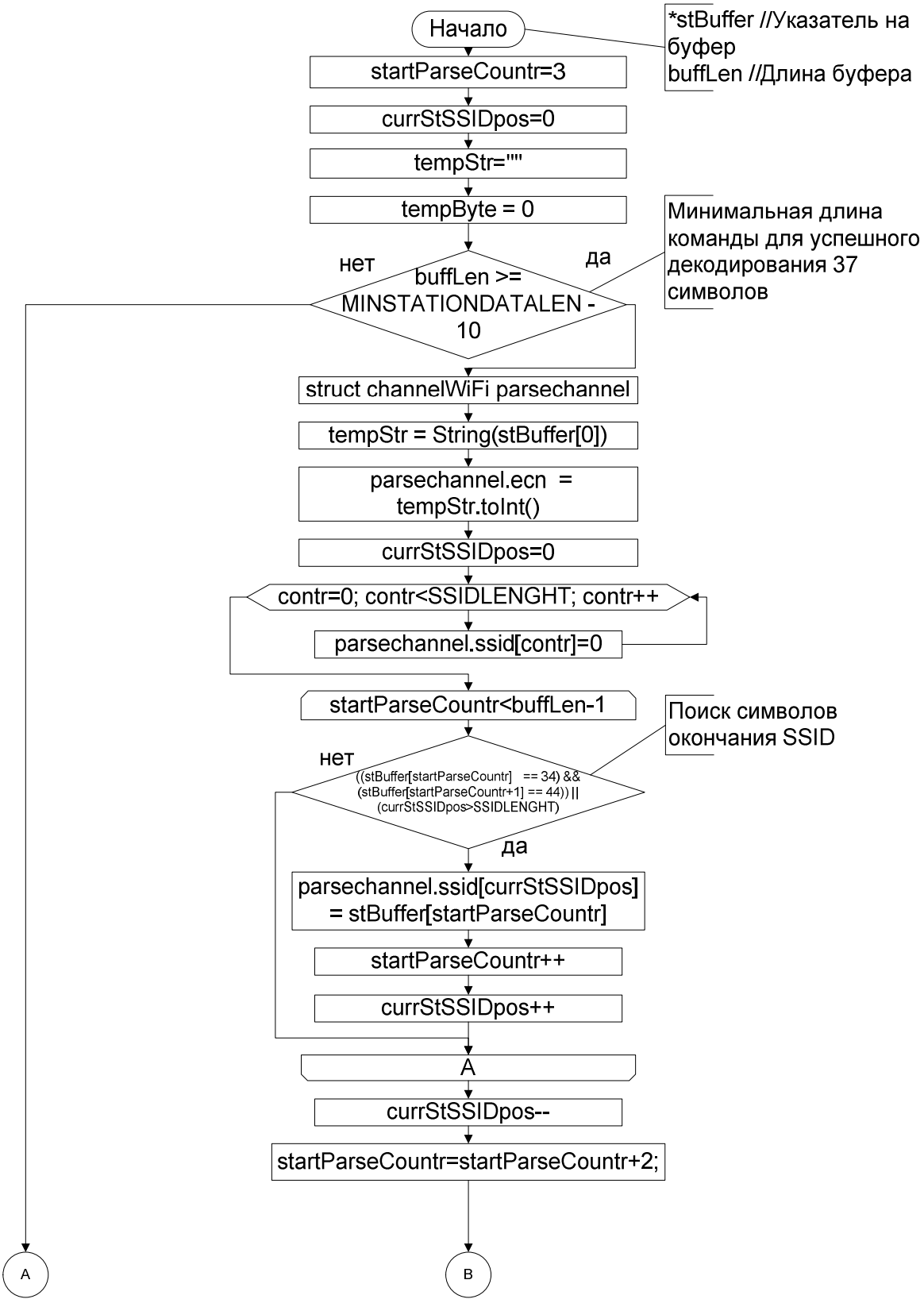

Рисунок 2.9 - SDL диаграмма инициализации подпрограммы parseWiFiStation(stationData,currStData) часть 1

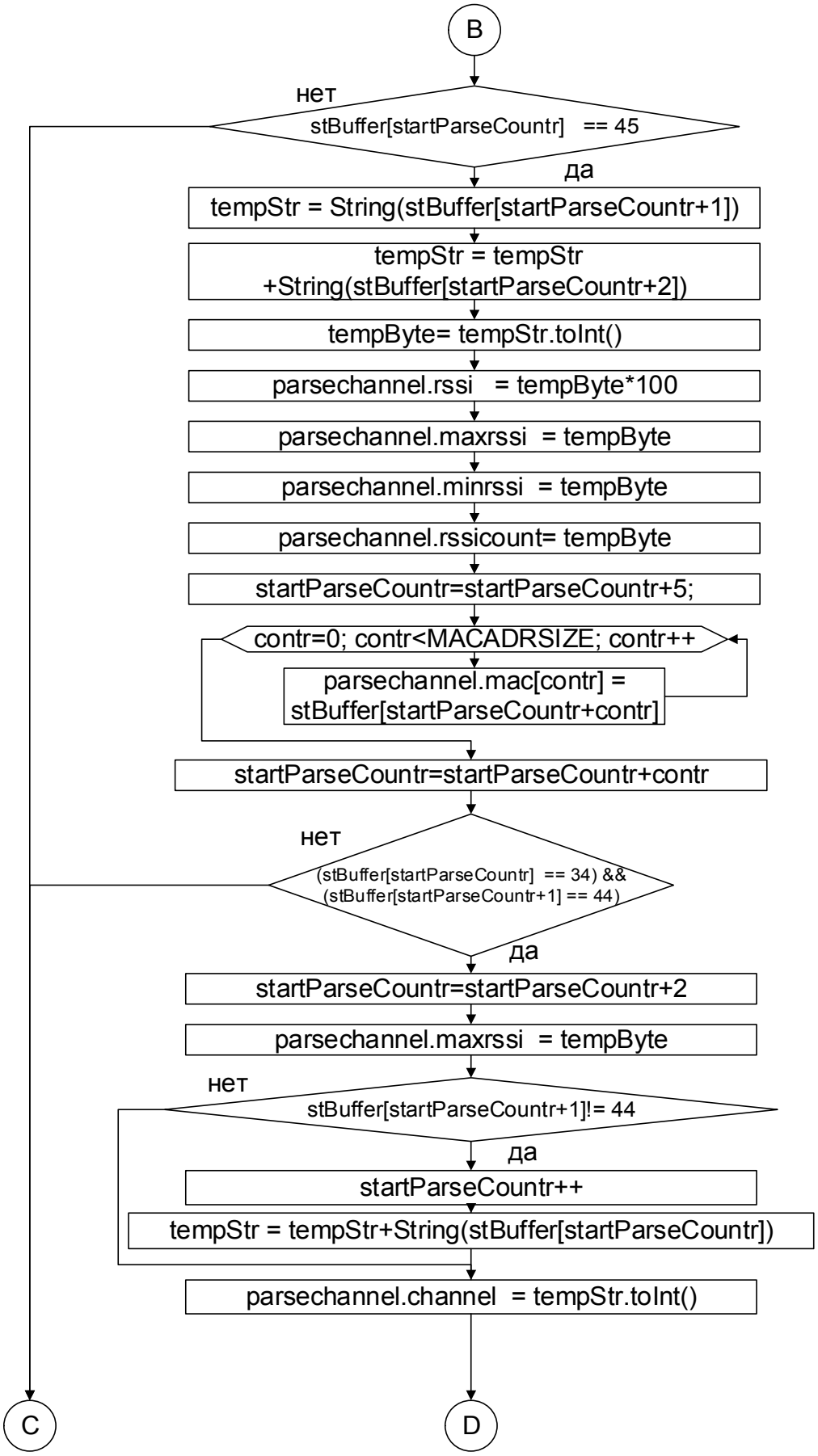

Рисунок 2.10 - SDL диаграмма инициализации подпрограммы parseWiFiStation(stationData,currStData) часть 2

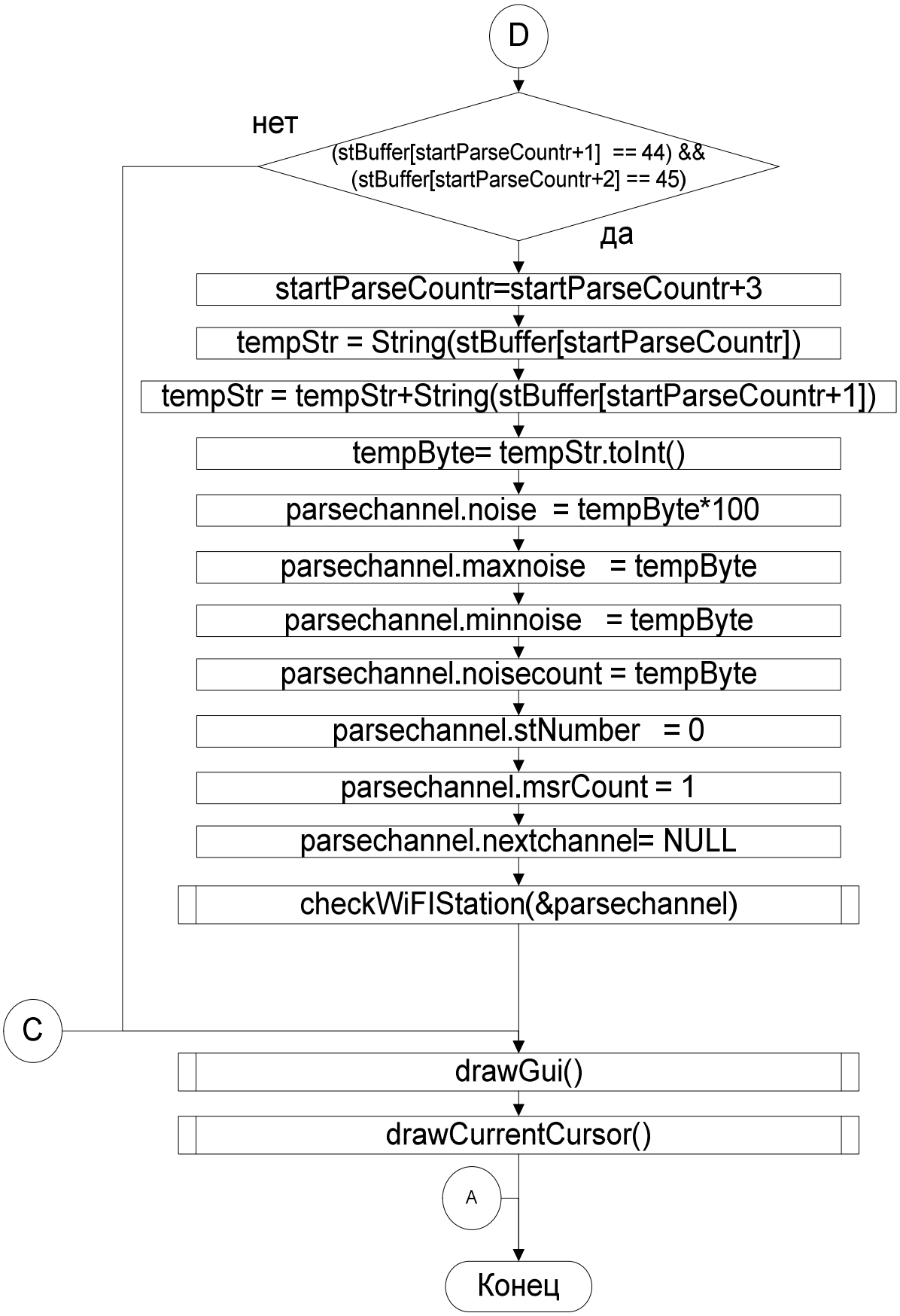

Рисунок 2.11 - SDL диаграмма инициализации подпрограммы parseWiFiStation(stationData,currStData) часть 3

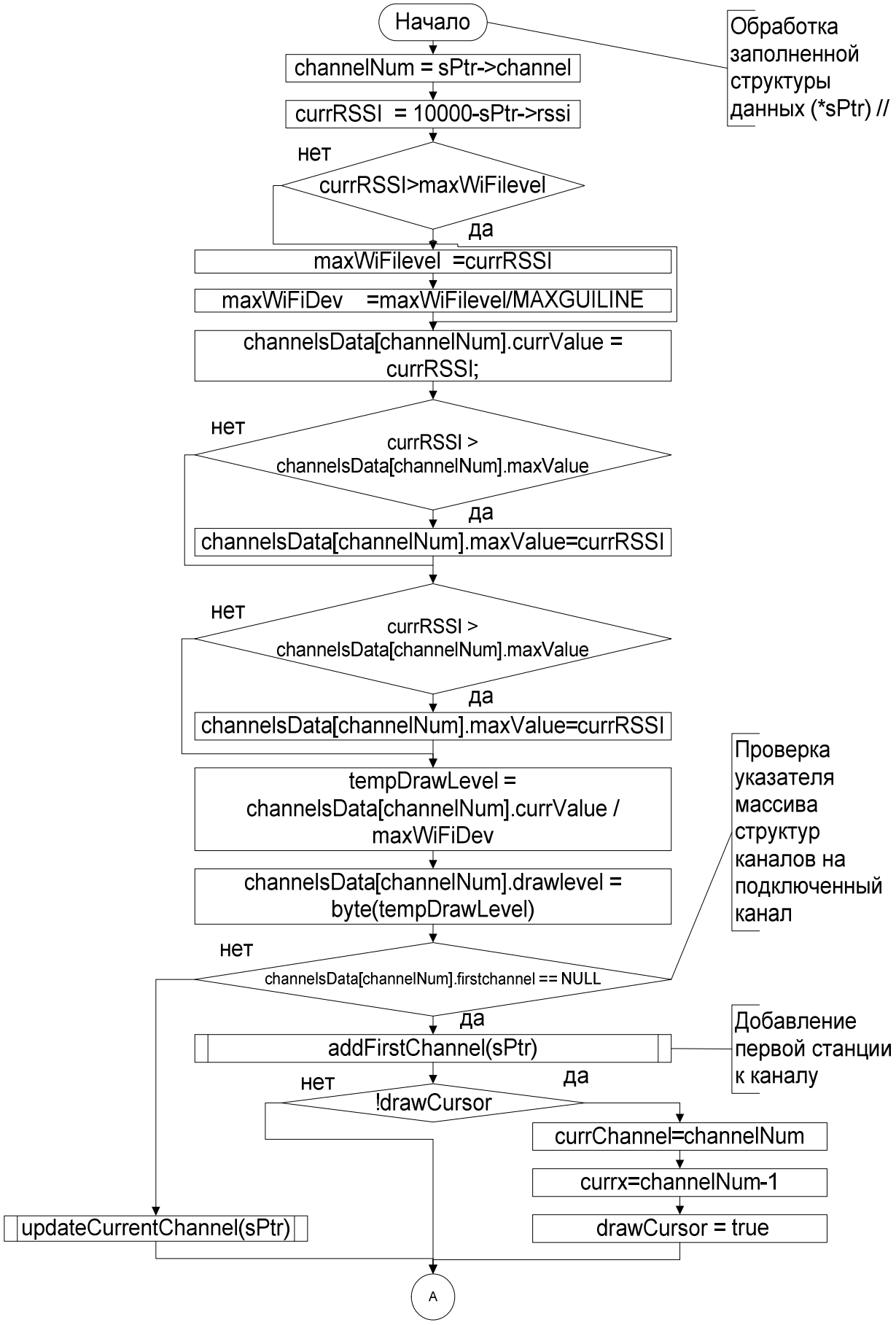

Рисунок 2.12 - SDL диаграмма инициализации подпрограммы checkWiFIStation часть 1

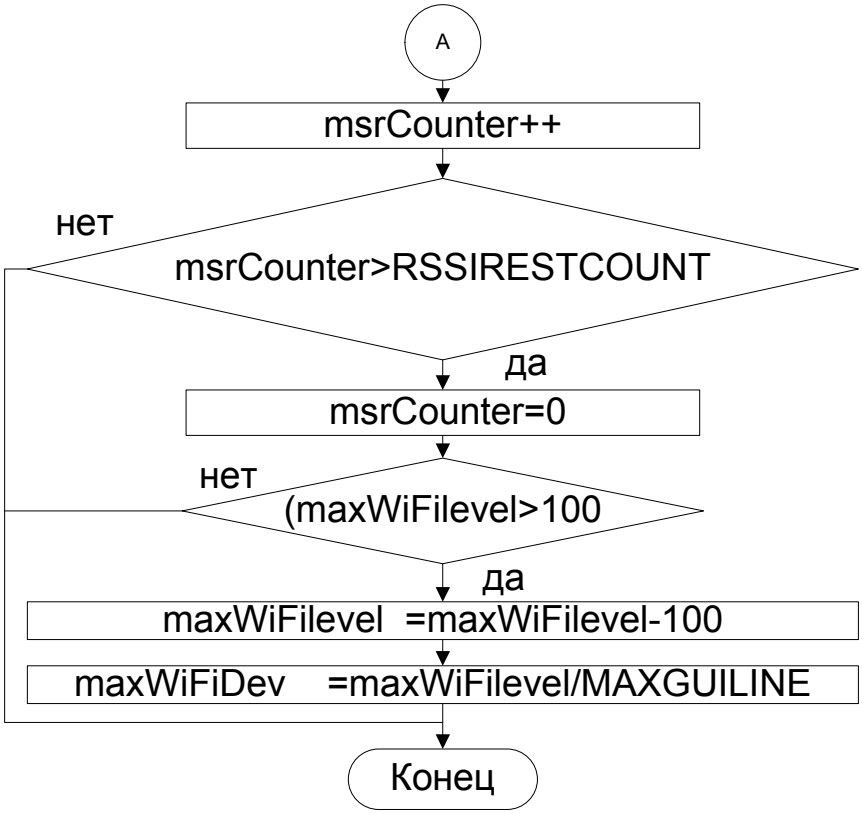

Рисунок 2.13 - SDL диаграмма инициализации подпрограммы checkWiFIStation часть 2

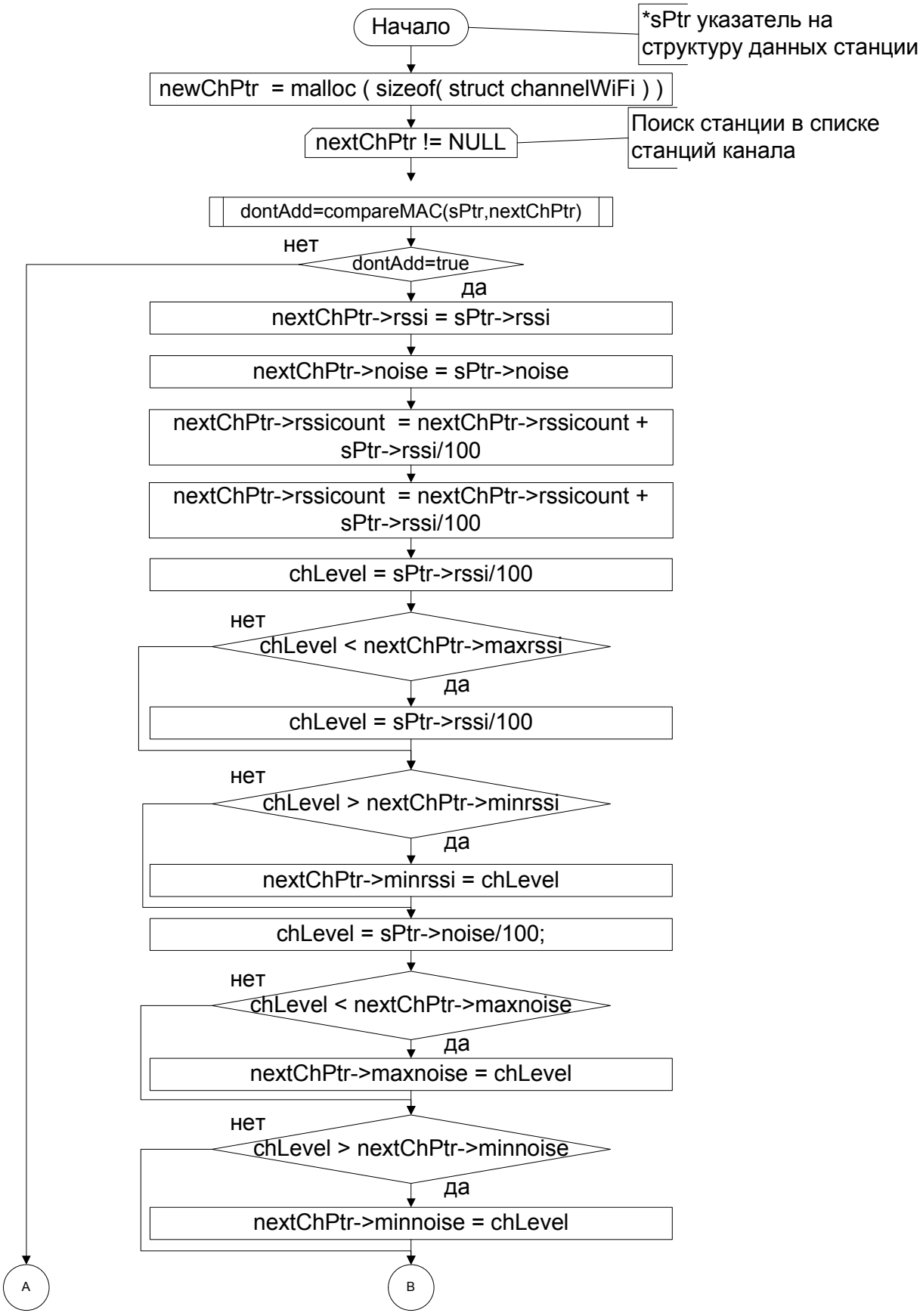

Рисунок 2.14 - SDL диаграмма инициализации подпрограммы updateCurrentChannel часть 1

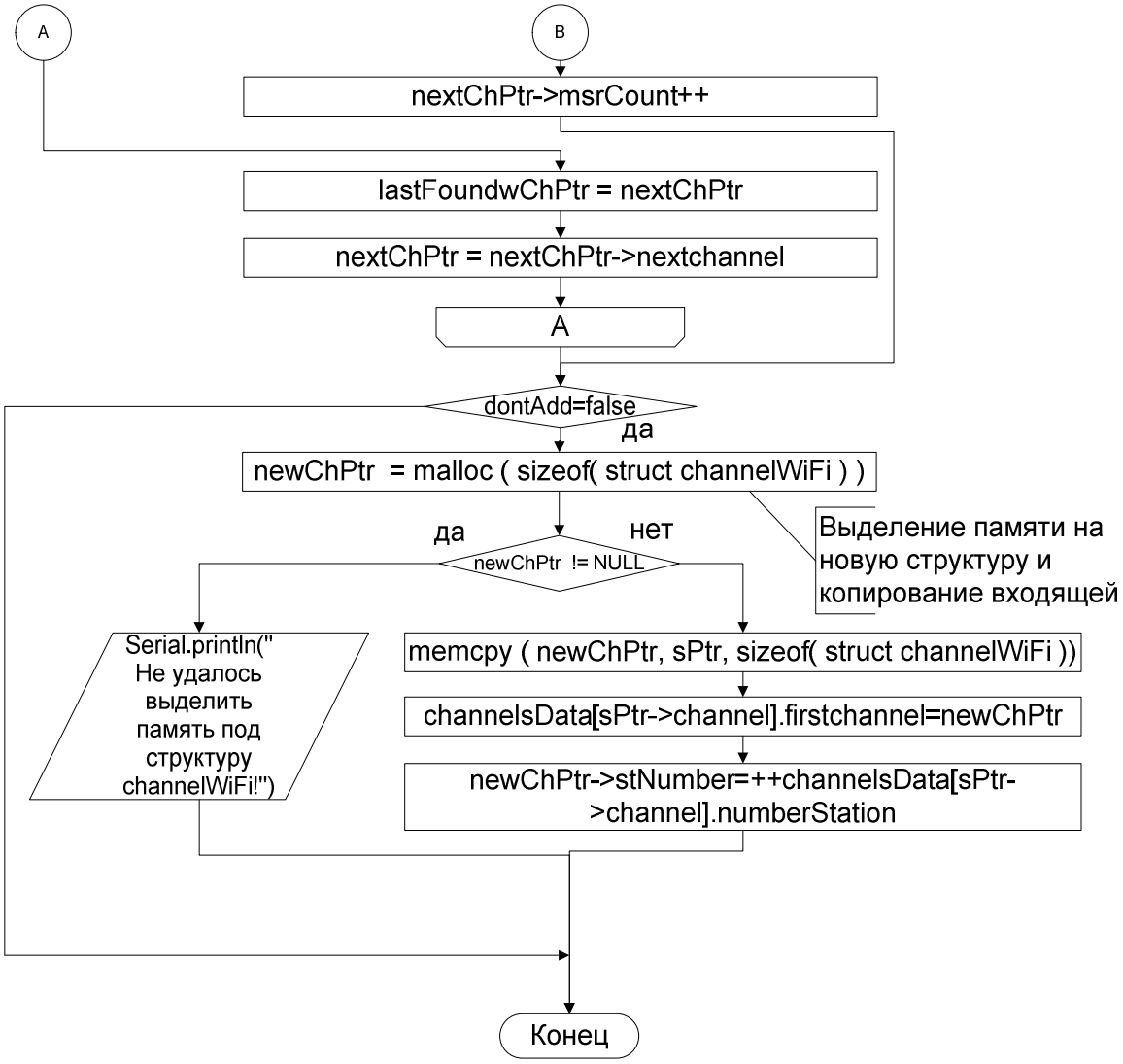

Рисунок 2.15- SDL диаграмма инициализации подпрограммы updateCurrentChannel часть 2

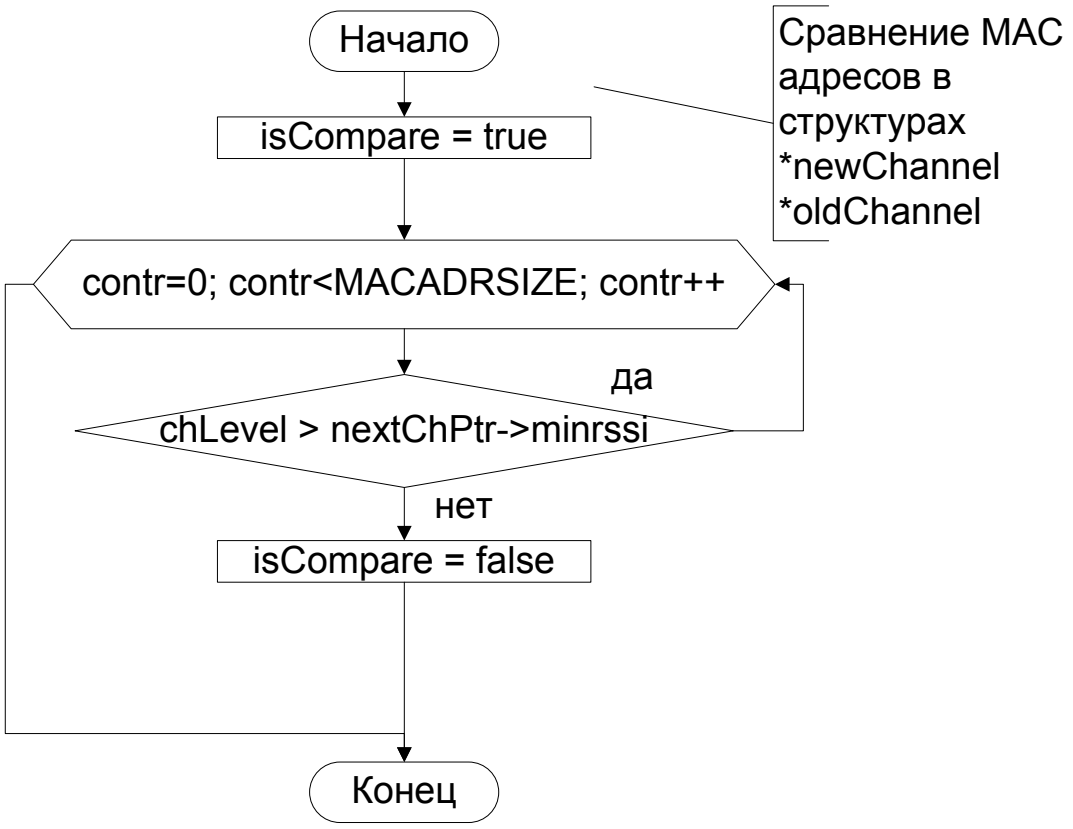

Рисунок 2.16- SDL диаграмма инициализации подпрограммы compareMAC

Описание основного алгоритма программы

1. Инициализация структур глобальных данных.
2. Инициализация контроллера ESP8266
3. Инициализация дисплея.
4. Вывод начального изображения на дисплей
5. Запрос к микроконтроллеру ESP8266 на сканирование доступных беспроводных сетей.
6. Декодирование полученных данных и заполнение структур хранения информации.
7. Вывод полученной информации в соответствии с текущим режим работы.
8. Опрос пользовательского ввода.
9. Обработка пользовательского ввода и изменение режима отображения данных в случае необходимости.
10. Переход к пункту 5.

2.2 Конфигурация оборудования (программы, печатные платы схемы)

Arduino Mega 2560 построена на микроконтроллере ATmega2560. Платформа содержит 54 цифровых входа/выходов (14 из которых могут использоваться как выходы ШИМ), 16 аналоговых входов, 4 последовательных порта UART, кварцевый генератор 16 МГц, разъем USB, силовой разъем, разъем ICSP и кнопку перезагрузки.

Спецификация:

- Микроконтроллер ATmega2560
- Напряжение питания 5В
- Входное напряжение (рекомендуемое) 7-12В
- Входное напряжение (предельное) 6-20В
- Цифровой ввод-вывод 54 линии (14 из них = ШИМ)
- Аналоговый ввод 16 линий
- Постоянный ток на линиях ввода-вывода 40мА
- Постоянный ток на линии 3.3В 50мА
- Flash-память 256КВ, 4 КВ из них использованы для загрузчика
- SRAM-память 8КВ
- EEPROM-память 4КВ
- Тактовая частота 16МГц

Схема 2.17 микроконтроллера atmega 2560.

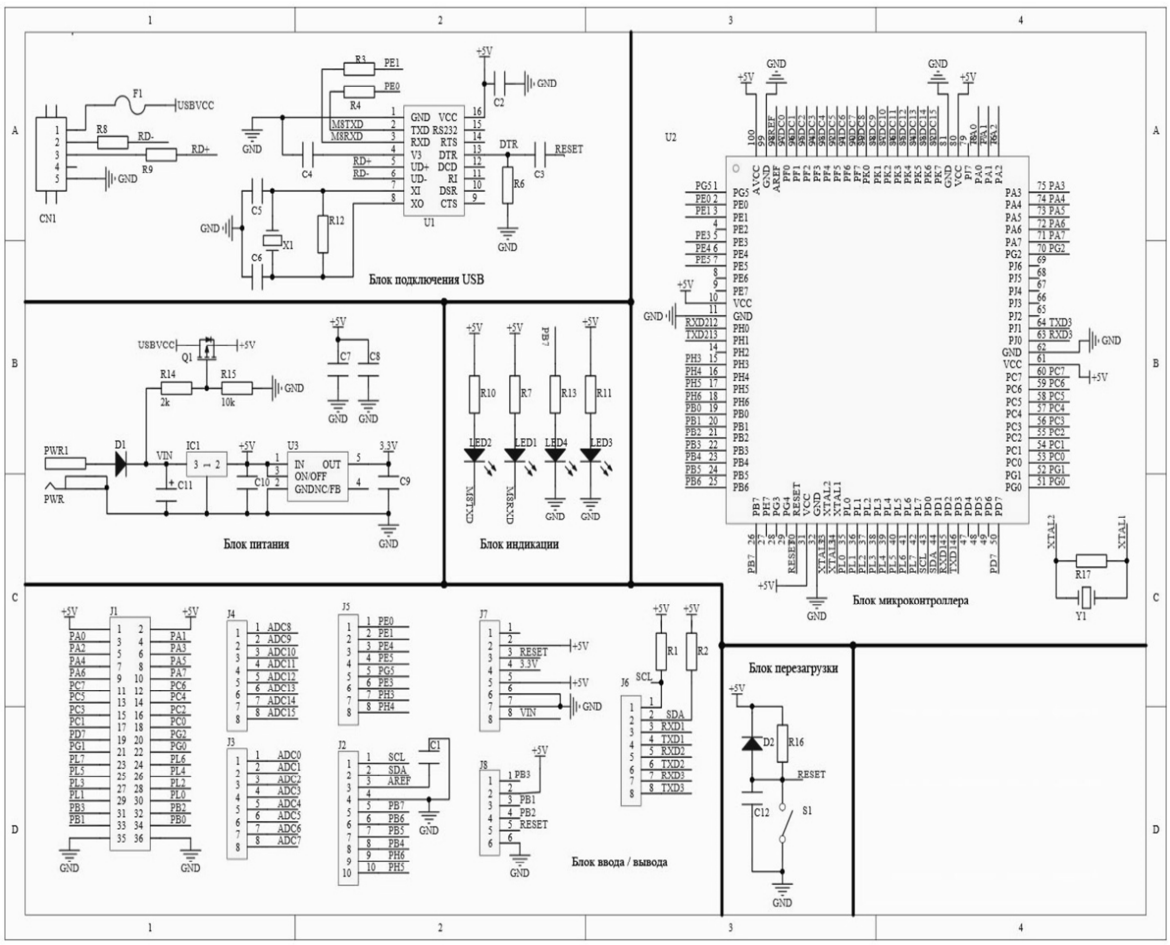

Рисунок 2.17 - Микроконтроллера atmega 2560

Радиомодуль ESP8266

Технические характеристики:

- WI-FI: 802.11 b/g/n с WEP, WPA, WPA2.
- Режимы работы: Клиент (STA), Точка доступа (AP), Клиент+Точка доступа (STA+AP).
- Напряжение питания 1.7..3.6 В.
- Потребляемый ток: до 215мА в зависимости от режима работы.
- Количество GPIO: 16.
- Flash память размером 512кб.
- RAM данных 80 кб
- RAM инструкций — 32 кб.

Недорогой радиомодуль ESP8266, работающий на частоте 2.4 ГГц и поддерживающий скорость передачи данных до 20 MBps (20 мегабита в

секунду). ESP8266 - отличный вариант для объединения отдаленных модулей в системе умного дома внутри помещений без прокладки проводов.

Характеристики

- Частота передачи/приема 2.4ггц.
- Дальность до 100 м. В помещении - до 30 м.
- Скорость до 2 Мб. Возможные варианты: 250kbps, 1Mbps и 2Mbps.
- Интерфейс взаимодействия с микроконтроллером - SPI
- Напряжение: 3-3.6V (рекомендуется 3,3V) В. При попытке подключения 5V вывод из строя маловероятен, но работать устройство отказывается..
- Максимальная выходная мощность: 0 dBm
- Коэффициент усиления антенны (пиковая): 2dBm
- Количество каналов: 126. Нулевой канал начинается с 2400мгц и далее с шагом 1 мгц, например 70 канал находится соответственно на 2470мгц. При установке скорости 2мгц занимается ширина двух каналов.

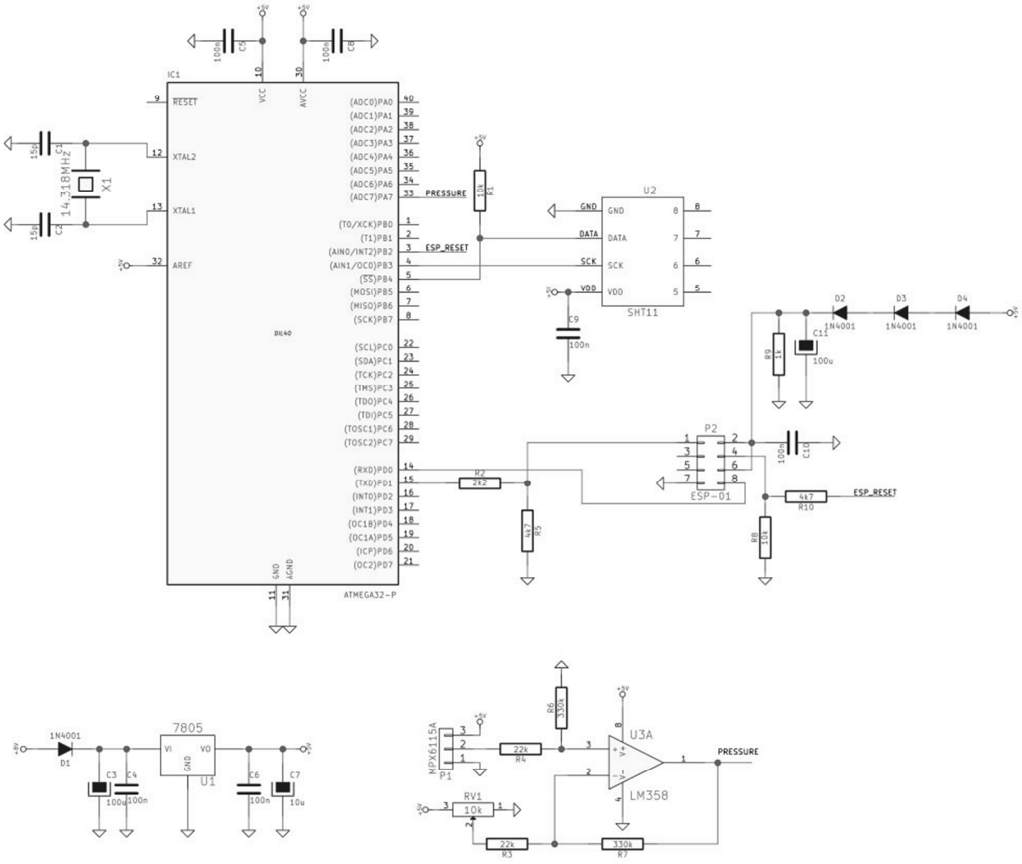

Рисунок 2.18 – Схема радиомодуля ESP8266

2.3 Платформа Arduino

Проект Arduino представляют собой набор, состоящий из готового электронного блока и программного обеспечения. Электронный блок— это печатная плата с установленным микроконтроллером и минимумом элементов, необходимых для его работы. На плате имеются разъемы для подключения внешних устройств, а также разъем для связи с компьютером, по которому и осуществляется программирование микроконтроллера. Особенности используемых микроконтроллеров ATmega фирмы Atmel позволяют производить программирование без применения специальных программаторов. Для разработки нового электронного устройства достаточно платы Arduino, USB кабеля связи и компьютер. Второй частью проекта Arduino является бесплатное программное обеспечение для создания управляющих программ, которое объединяет в себе простейшую среду разработки и язык программирования, представляющий собой вариант языка C/C++ для микроконтроллеров. В него добавлены элементы, позволяющие создавать программное обеспечение без глубокого изучения аппаратной части. Для разработки приложения для Arduino достаточно знания только основ программирования на C/C++, а большое количество библиотек, содержащих код, значительно облегчает низкоуровневое программирование различных устройств.

Преимущество Arduino:

Arduino позволяет разработчику сосредоточиться на разработке проектов, не тратя много времени на изучении устройства и принципов функционирования отдельных элементов. Разработчик может использовать готовые платы расширения или просто напрямую подключить к Arduino необходимые элементы. Все остальные усилия будут направлены на разработку и отладку управляющей программы на языке высокого уровня. В итоге доступ к разработке микропроцессорных устройств получили не только

профессиональные разработчики программного обеспечения для микроконтроллеров. Наличие готовых модулей и библиотек программ позволяет непрофессионалам в электронике и программировании создавать готовые работающие устройства для решения своих задач. А варианты использования Arduino ограничены только возможностями микроконтроллера и имеющегося варианта платы.

2.4 Платформа Raspberry Pi

Raspberry Pi изначально разрабатывался как недорогой одноплатный компьютер для образовательных целей, с возможностью установки многозадачной операционной системы. Отличается от большинства микроконтроллеров наличием производительного процессора, большим объемом оперативной памяти и обладает видеовыходом. Старшие модели оснащаются портами Ethernet и контролером беспроводной сети WiFi. Для разработки программного обеспечения требуется установка операционной системы Linux или Windows 10 IoT Core, от выбора которых зависит выбор среды разработки приложений. Разработка приложений ведется на языках C/C++, python или C#, требует от разработчика знание основных принципов работы установленной операционной системы и разработки программного обеспечения для этой системы.

2.5 Платформа STM32

Компания STMicroelectronics одной из первых вывела на рынок семейство микроконтроллеров на ядре ARM и на сегодняшний день по праву занимает лидирующее место среди производителей микроконтроллеров на этом ядре . Многие клиенты используют STM32 так как он наиболее дешевый своих предшественников таких как Arduino и Raspbery pi. Для микроконтроллеров STM32 существует широкий выбор оценочных плат. При чём не только от

«STMicroelectronics», но и от прочих производителей. Например, модули «Махаон» и «Барракуда» от компании «Терраэлектроника». Поэтому, для большинства разработчиков коммерческих компаний и радиолюбителей нет необходимости самостоятельно изготавливать отладочные платы и программаторы. STM32F746G-DISCO - отладочная плата из серии бюджетных отладочных плат Discovery для оценки функциональных возможностей новейшего микроконтроллера семейства STM32F7 на базе ядра ARM® Cortex®-M7. Плата позволит создавать большое количество разнообразных приложений с поддержкой аудио, графического интерфейса, мультитача, видео и высокоскоростного соединения. Плата поддерживает подключение большого количество плат расширения Arduino (shields), которые позволят существенно расширить функциональные возможности.

Сравнительный анализ Arduino , Raspberry , STM32 в таблице 2.1.

Таблица 2.1- Сравнение микроконтроллеров

платформа	Микроконтроллер Arduino	Одноплатный компьютер Raspberry Pi	Микроконтроллер STM32
Производительность	1 ядро, 10-100 Мгц, 10+ КБ оперативной памяти, 10-100+ КБ постоянной памяти	1 или более ядер, 100-1000+МГц, 100+МБ оперативной памяти, ГБ постоянной памяти	1 или более ядер 200+МГц, 1-100+МБ оперативной памяти, 128МБ(64) постоянной памяти
Многозадачность	Нет, но можно эмулировать	Управляется ОС	Управляется ОС
Удобство работы в сети	Нужны доп. модули и знание протоколов	Подключение из корпуса, сетевой модуль уже встроен	Сетевой модуль встроен
Длительность работы от батареи	Потребляет 1-10+мА, возможна длительная работа	Потребляет 100-1000+мА, малая емкость батареи хватает на 10+ часов	Питается через встроенный V2-1, через аудиоразъем, подключение к разъему 5V
Скорость реакции в проектах критичных к времени	100% контроль над временем и длительностью подачи сигналов	Из-за многозадачности критические процессы могут исполняться с запозданием	100% контроль над временем хорошая многозадачность.
Выбор языков программирования	Ограничен , чаще C/C++	Любые доступные в ОС	Любые доступные в ОС

Окончание таблицы 2.1

Возможность для работы с видео, экраном монитора	Не возможно, не хватает мощности	Присутствуют аппаратные видеокодеки, HDMI-выход	Присутствуют видео рамки а также видеокодеки, есть HDMI разъем
Возможность работать со звуком	На мощных микроконтроллерах возможен синтез звука, для работы с MP3/OGG/WAV нужны доп.модули	Поддержка MP3/OGG/WAV на уровне ОС, аудиовыход HDMI и разъем 3.5 мм	Поддерживает 2 микрофона ST MEMS, SAI аудикодек, аудиовыход и линейный выход разъем, Поддержка MP3/OGG/WAV
Экономическая составляющая и среда разработки	Небольшая цена, бесплатная среда разработки, легкость программирования	Высокая цена, сложность программирования	высокая цена, сложность программирования, платная среда разработки
Цена за модуль	Средняя стоимость	Высокая стоимость	Высокая стоимость более дороже чем Raspbery
Сложность разработки программного обеспечения	низкая	высокая	высокая
Сложность загрузки прошивки	Не высокая - поддерживается платформой	Не высокая – поддерживается платформой	Высокая требуются доп.компоненты

Заключение по выбору платформы для разрабатываемого устройства

Для устройства была выбрана платформа Arduino, потому что

2.6 Оценка характеристик и работоспособности

Устройство выполняет заданные функции по сканированию беспроводных сетей в диапазоне 2.4 ГГц, тестирование было произведено в нескольких местах(лаб.,условиях, жилом комплексе, торговом центре). Все функции работают правильно и без сбоев в работе.

3 ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТОВ РАЗРАБОТАНОГО УСТРОЙТСТВА

3.1 Сборка разработанного устройства

Общая схема. Фотографии основной платы представлены рисунок 3.1:

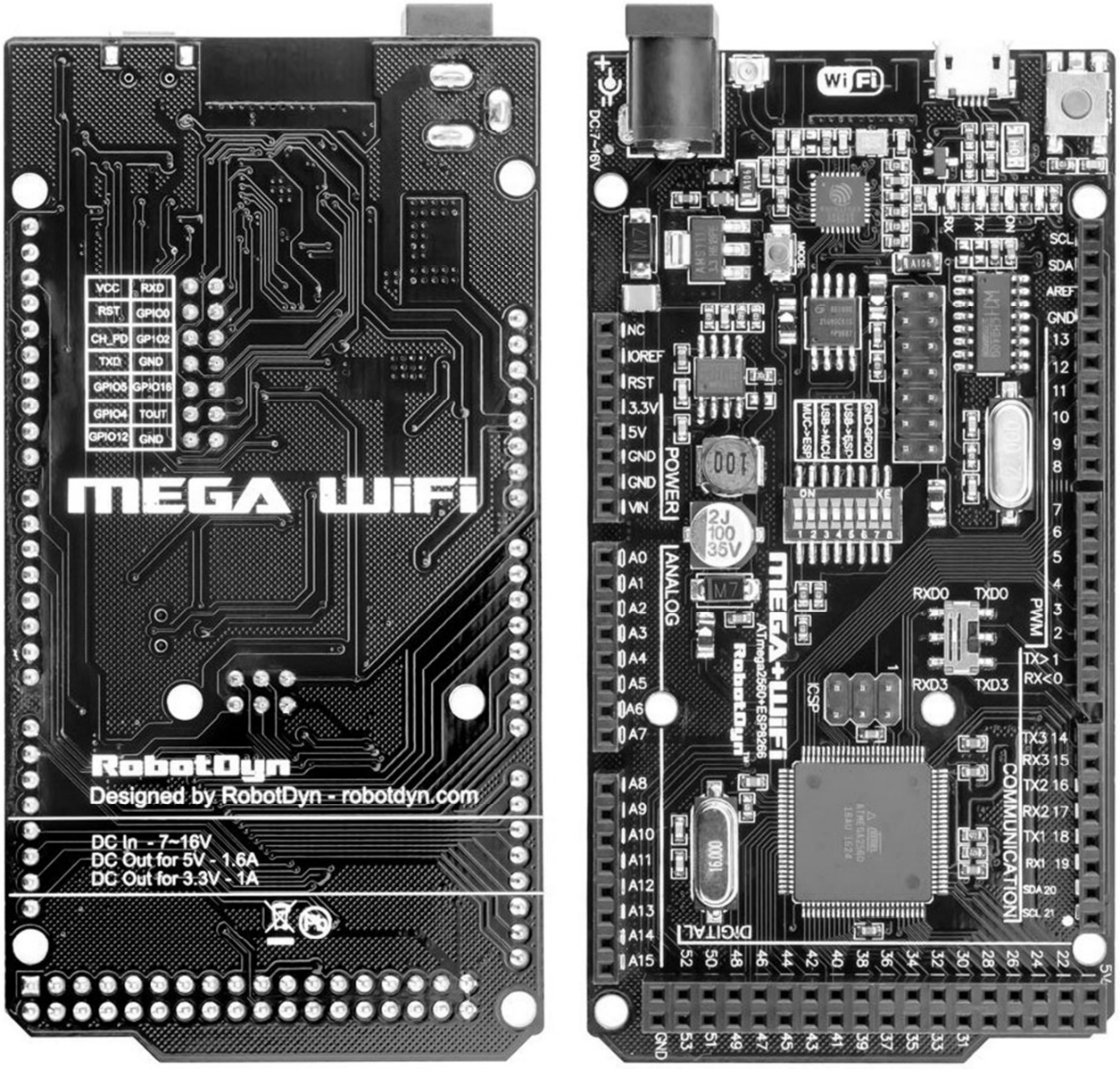

Рисунок 3.1 – Фотография основной платы разрабатываемого устройства

Модуль индикации и управления. Сочетание в одном модуле жидкокристаллического индикатора принимающего отображаемую информацию по шине 4 бит и небольшой клавиатуры упрощает

разрабатываемый прибор. Применение модуля для сборки передней панели упрощает конструкцию. ЖКИ дисплей имеет регулировку подсветки. Предусмотрены отверстия для установки на переднюю панель прибора. Схема модуля управления и отображения представлена на рисунке 3.2.

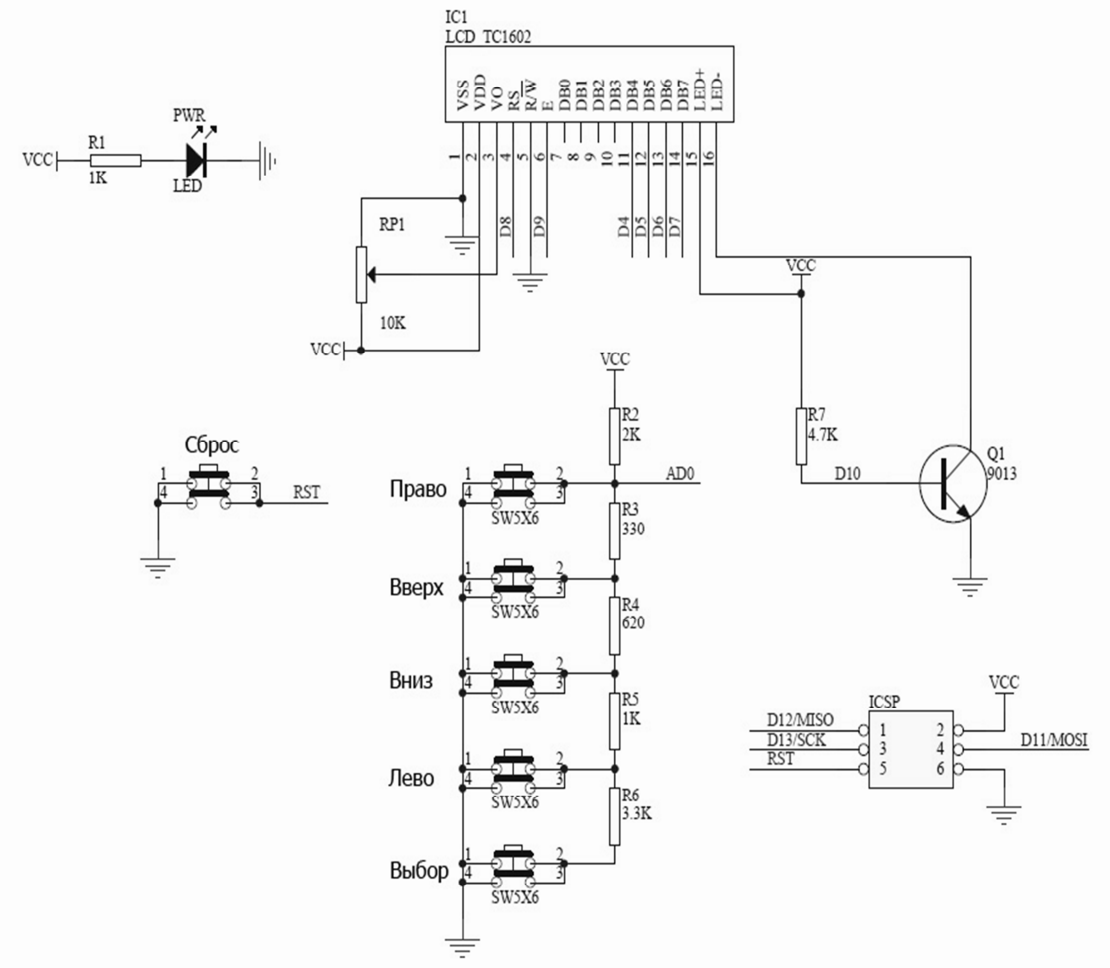

Рисунок 3.2 – Схема модуля управления и отображения

Таблица с расшифровкой обозначений – таблица 3.1.

Таблица 3.1- Назначение и расшифровка аббревиатур

Аббревиатура	Значение
VSS	Общий
VDD	Питание
V0	Контрастность
RS	Трактовка принятых данных
R/W	Общий
E	Активирование

Окончание таблицы 3.1

Аббревиатура	Значение
DB0	Не подключены
DB1	
DB2	
DB3	
DB4	Данные
DB5	
DB6	
DB7	
LED+	Питание подсветки
LED-	Управление подсветкой

На рисунке 3.3 и 3.4 представлены фотографии панели управления и отображения разработанного устройства.

Рисунок 3.3 – Схема модуля ЖКИ монитора отображения

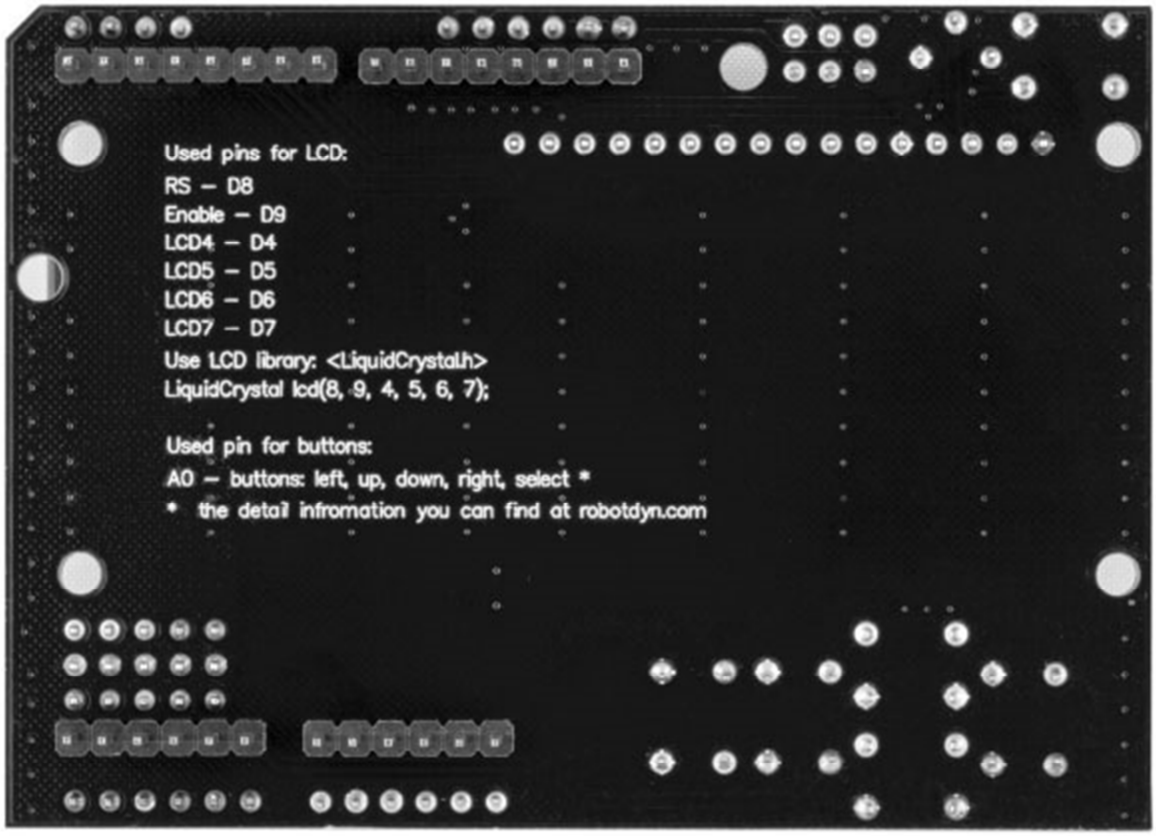

Рисунок 3.4 – Схема платы панели управления

Назначение контактов индикатора IC1 LCD_TC1602:

Vdd и Vss — питание индикатора.

V0 — используется для задания контраста изображения, обычно к нему подключают средний вывод от переменного резистора. Хотя можно один раз подобрать контраст индикатора и вместо переменника впаять два постоянных резистора делителя.

RS — определяет что записывается в индикатор — данные (1) или команда (0).

R/W — определяет направление данных — чтение (1) или запись (0). Обычно в индикатор только записывают данные, поэтому на этот вывод можно дать 0 на постоянку.

E — используется как строб для записи/чтения.

DB0-DB7 — отвечают за входящие/исходящие данные. Контроллер индикатора предусматривает работу с 4-х битным интерфейсом. Тогда для

отправки данных используется только выводы DB4-DB7, а данные передаются в две команды. Выводы при DB0-DB3 при этом не используются.

На рисунке 3.5 можно увидеть ЖКИ и контроллер соединенные вместе

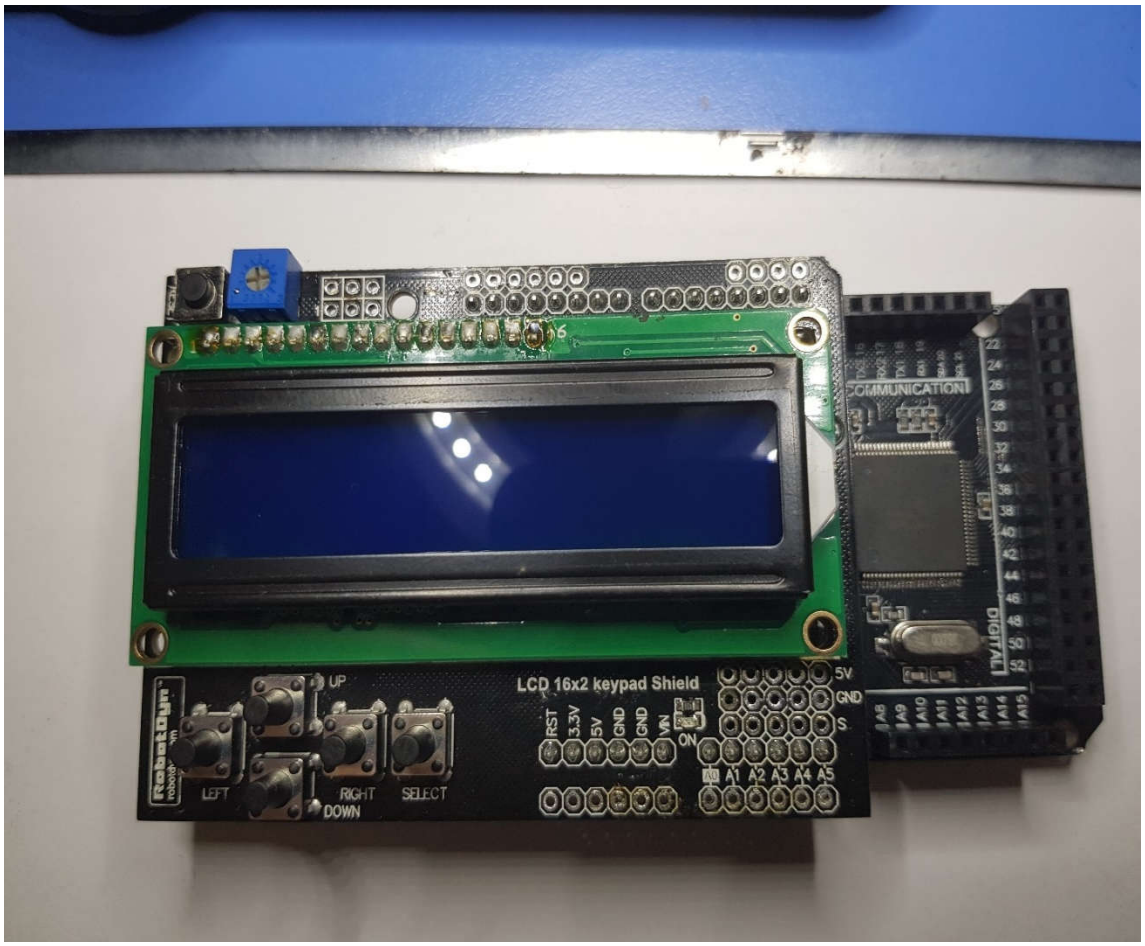

Рисунок 3.5 – Устройства в сборе без корпуса и АКБ

Среда разработки Arduino

Среда разработки Arduino (рисунок 3.6) состоит из встроенного текстового редактора программного кода, области сообщений, окна вывода текста(консоли), панели инструментов с кнопками часто используемых команд и основного меню. Для загрузки программ и связи среда разработки должна быть подключена к аппаратной части Arduino.

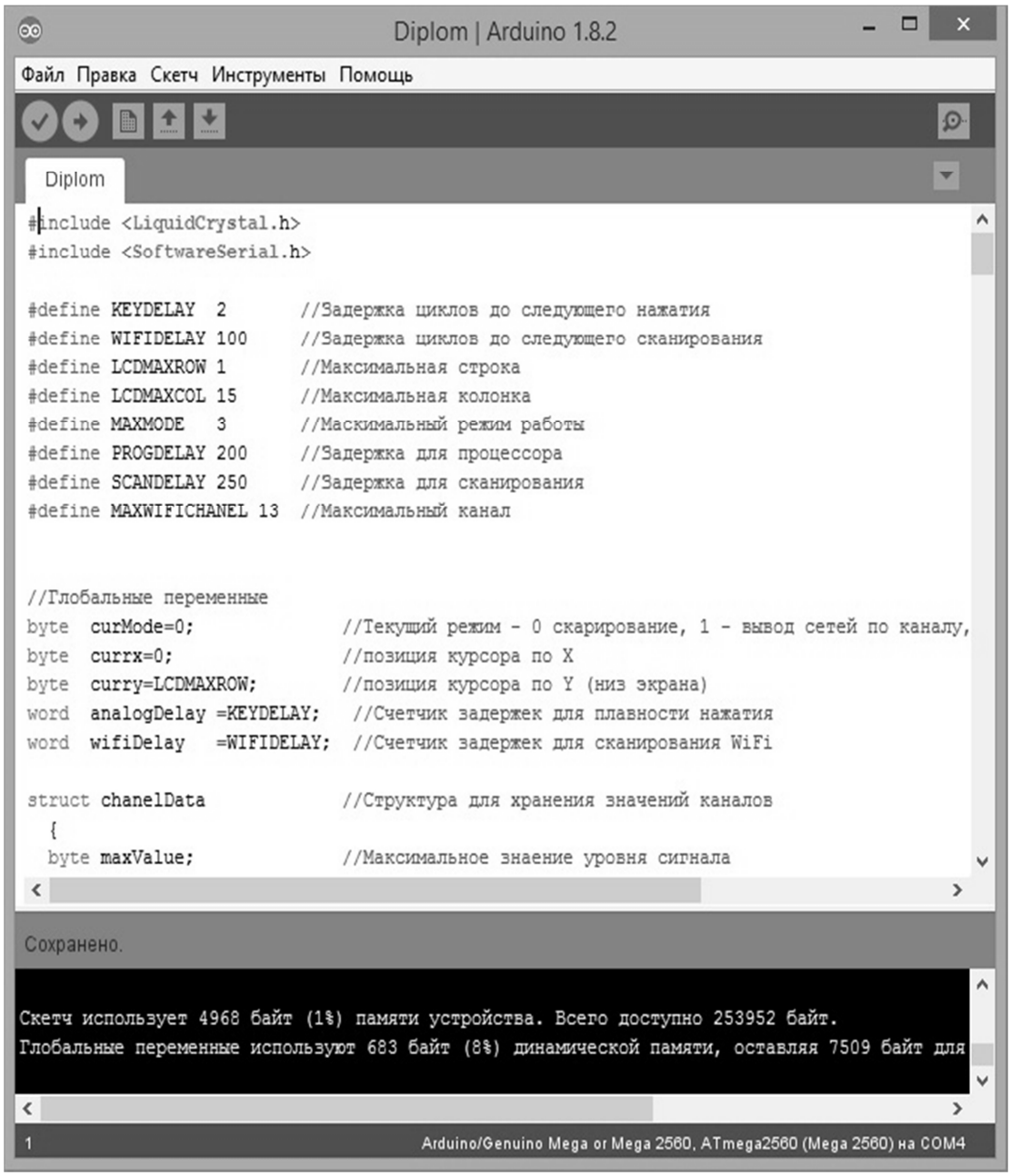

Рисунок 3.6- Среда разработки ПО ArduinoIDE

Программа, написанная в среде Arduino, называется скетч. Скетч пишется в текстовом редакторе (рис.3.6), имеющем инструменты вырезки/вставки, поиска/замены текста. Во время сохранения и компиляции проекта в области сообщений появляются пояснения, также могут отображаться возникшие ошибки. Окно вывода текста(консоль) показывает сообщения Arduino, включающие полные отчеты об ошибках и другую информацию. Кнопки панели инструментов позволяют проверить и записать программу, создать,

открыть и сохранить скетч, открыть мониторинг последовательной шины (Serial Monitor).

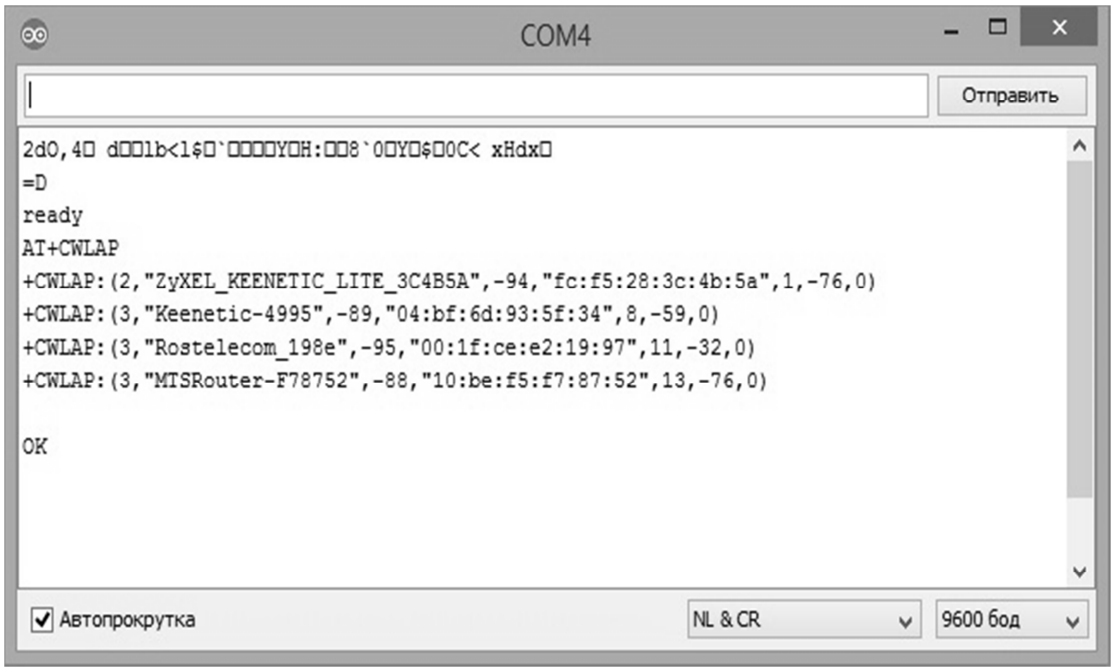

Рисунок 3.7 - Монитор последовательной шины

Монитор последовательной отображает данные посылаемые в платформу Arduino (плата USB или плата последовательной шины). Для отправки данных необходимо ввести текст и нажать кнопку Send или Enter. Затем выбирается скорость передачи из выпадающего списка, соответствующая значению Serial.begin в скетче.

В среде разработки ПО Arduino IDEиспользуется язык программирования C++. C++ - компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции.

3.2 Тестирование в лабораторных условиях

Таблица 3.2 - Эксперимент произведен в лабораторных условиях

Имя сети	MAC адрес	канал	Частота ГГц	RSSI
My router TP-LINK-FC1572	14-cc-20-fc-15-72	1	2.412	-30
My router	14-cc-20-fc-15-72	2	2.417	-32
My router	14-cc-20-fc-15-72	3	2.422	-37
My router	14-cc-20-fc-15-72	4	2.427	-50
My router	14-cc-20-fc-15-72	5	2.437	-56
My router	14-cc-20-fc-15-72	6	2.442	-56
My router	14-cc-20-fc-15-72	7	2.412	-60
My router	14-cc-20-fc-15-72	8	2.437	-62
My router	14-cc-20-fc-15-72	9	2.462	-65
My router	14-cc-20-fc-15-72	10	2.442	-80
My router	14-cc-20-fc-15-72	11	2.462	-81
My router	14-cc-20-fc-15-72	12	2.472	-85
My router	14-cc-20-fc-15-72	13	2.472	-90

Эксперимент в лабораторных условиях показал, что при изменении канала сети частота при этом тоже изменялась и с увеличением расстояния от беспроводной точки доступа RSSI изменялся в худшую сторону, показывая на сколько качество сигнала зависит от расстояния.

Таблица 3.3 - Эксперимент произведен в условиях жилого комплекса

Имя сети	MAC адресс	канал	Частота GHz	RSSI
G2D	F8-d1-11-a9-11-62	1	2.412	-10
roscha	64-70-02-B9-D0-82	1	2.412	-87
Ruslan	C8-D3-A3-28-84-48	1	2.412	-90
TP-LINK_E2300E	C0-4A-00-E2-30-0E	1	2.412	-95
DIR-320	00-90-4c-c1-00-00	1	2.412-	-92
ROSTELECOM_426C	8C-10-D4-5F-42-6C	2	2.417	-87
ROSTELECOM_5799	44-E9-DD-DB-57-99	2	2.417	-86
opel6052	C0-4A-00-64-53-20	6	2.437	-83
TP-LINK_530666	30-B5-C2-53-06-66	6	2.442	-90
TP-LINK_C072	F4-F2-6D-D3-C0-72	7	2.442	-91
Mts6	B0-B2-DC-F7-98-34	11	2.462	-89
MTSRouter-076875	80-26-89-07-68-75	13	2.472	-90
MTSRouter-81B226	10-62-EB-81-B2-26	13	2.472	-87

Эксперимент в жилом комплексе показал, что множество точек беспроводного доступа перекрывают друг друга, что и наблюдается в таблице 3.2, но зная что 1.6 и 11 каналы не пересекающиеся можно сделать вывод, что наилучшим вариантом будет перейти на эти каналы сети, но в нашем случаи наименее забиты или полностью свободны 3,4,5,8,9,10 и 12 каналы. RSSI у каждой точки доступа наблюдается высокий, причиной послужило большее расстояние от точки доступа

Таблица 3.3 - Эксперимент произведен в условиях торгового центра

Имя сети	MAC адресс	канал	частота	RSSI
MonPlezir	2C-56-DC-89-0C-F1	1	2.412	-82
G2D	F8-D1-11-A9-11-62	1	2.412	-10
FitnessCity	2C-56-DC-89-0C-F0	1	2.412	-81
DIR-320	00-90-4C-C1-00-00	1	2.412	-92
AndroidAP	C8-38-70-46-4D-24	1	2.412	-71
Aks Finance	6C-3B-6B-29-B1-AD	1	2.412	-91
MikroTik-173E39	6C-3B-6B-17-3E-39	3	2.422	-65
Elesir31	6A-28-5D-79-24-38	4	2.427	-88
FreeWiFi	0E-18-D6-A9-D9-24	6	2.437	-52
FreeWiFi	04-18-D6-7E-1F-61	6	2.437	-60
FreeWiFi	C6-9F-DB-F1-BF-25	6	2.437	-61
eugene	E4-8D-8C-BF-4D-C3	6	2.437	-95
Milavitsa	00-23-B1-73-C9-C4	6	2.437	-82
MGP	98-DE-D0-89-F9-00	10	2.457	-58
mts6	B0-B2-DC-F7-98-34	11	2.462	-90
Keenetic-9042	1C-74-0D-8B-B5-B8	11	2.462	-95
FreeWiFi	0A-27-22-0B-5A-67	11	2.462	-92
MARMELAD	64-70-02-B5-97-0A	13	2.472	-48

Эксперимент в торговом центре показал что множество сетей находятся в хаотичном порядке RSSI и многие из них отдаленны на значительно большое расстояние от программного аппарата, так же наблюдается значительное

забитость каналов что препятствует нормальной работы беспроводных точек доступа и создается значительная помеха сигнал-шум. Из таблице можно наблюдать, что каналы 2,5,7,8,9 и 12 свободны и можно сделать вывод что для улучшения качества сети необходимо распределить на свободные каналы.

4 ЭКОНОМИЧЕСКАЯ ОЦЕНКА ПРОЕКТА

Основной целью научно-исследовательской или опытно-конструкторской работы является проведение работ, направленных на анализ, проектирование или разработку каких-либо устройств. Результатом таких работ могут являться разработанный прототип прибора или программный продукт, выполняющий определенные функции, рекомендации по эксплуатации прибора или технологии, нормативные акты и так далее.

4.1 Планирование работ по разаработке

В проведении исследования задействованы следующие специалисты:

- главный инженер или старший научный сотрудник (заведующий лабораторией), осуществляющий общее руководство исследованием;

- инженер I категории или младший научный сотрудник, проводящий разработку, исследование, необходимые расчеты, составляющий техническую документацию на исследование;

- экономист, дающий экономическую оценку исследования.

Расчет сроков проведения и трудоемкости представлен в таблице 4.1.

Таблица 4.1 - Планирование работ по исследованию

Наименование этапов работ	Исполнитель	Трудоемкость, час	Продолжительность, дней
1	2	3	4
1.Подготовительный			
1.1.Сбор информации	Младший научный сотрудник	48	6
1.2.Выработка идеи	Старший научный сотрудник	48 48	6 6
1.3.Определение объема исследовательских работ	Младший научный сотрудник	16	2
1.4.Формирование исследовательской работы	Младший научный сотрудник	16	2

Окончание таблицы 4.1

1	2	3	4
1.5.Обработка и анализ Информации	Младший научный сотрудник	80	10
Итого:		256	32
2.Основной (экономический анализ)			
2.1.Обоснование целесообразности работы	Старший научный сотрудник	32	4
2.2.Выполнение работы	Младший научный сотрудник	96	12
Итого:		128	16
3.Заключительный			
3.1.Технико-экономическое Обоснование	Экономист	48	6
3.2.Оформление и утверждение документации	Младший научный сотрудник	48	6
Итого:		96	12

Результат планирования представляет собой расчет трудоемкости исследования по часам и по количеству дней. Была определена численность штата производственного персонала, который необходим для проведения исследования.

4.2 Расчет расходов на оплату труда на разработку

Расчет расходов на оплату труда разработки исследования представлен в таблице 4.2.

Таблица 4.2 - Расчет расходов на оплату труда

Должность исполнителей	**Трудоемкость, час**	**Оклад, Руб**
1	**2**	**3**
Младший научный сотрудник	304	13000
Старший научный сотрудник	128	15000
Экономист	48	12000
Итого:	480	

Часовая тарифная ставка ($Ч_{ТС}$) рассчитывается следующим образом:

$$Ч_{ТС} = \frac{Р}{F_{мес}}, \quad (14)$$

где $F_{мес}$ – фонд рабочего времени месяца, составляет 176 часов (22 рабочих дня по 8 часов в день); Р – оклад сотрудника.

Расход на оплату труда ($Р_{ОТ}$) можно найти по следующей формуле:

$$Р_{ОТ} = Ч_{ТС} * Т_{сум}, \quad (15)$$

где $Т_{сум}$ – суммарная трудоемкость каждого из исполнителей.

Результаты расчетов представлены в таблице 4.3.

Таблица 4.3 - Расчет расходов на оплату труда

Должность исполнителей	**Трудоемкость, час**	**Оклад, Руб**	**$Ч_{ТС}$, руб/час**	**$Р_{ОТ}$, руб**
Младший научный сотрудник	304	13000	73,86	22453,44
Старший научный сотрудник	128	15000	85,23	10909,44
Экономист	48	12000	68,18	3272,64
Итого:	480			36635,52

4.3 Расчет продолжительности разработки

Согласно данным таблицы 7 трудоемкость исследования составила 480 часов.

Продолжительность исследования составит:

$$Т_{иссл} = Т_{сум} / Т_{РД}, \quad (16)$$

где $Т_{СУМ}$ = 480 часов суммарная трудоемкость исследования; $Т_{РД}$ = 8 часов – продолжительность рабочего дня.

$$Т_{иссл} = 480/8 = 60 \text{ дней.}$$

Продолжительность исследования составляет 60 дней, расчет производится без учета выходных и праздничных дней.

4.4 Расчет стоимости расходных материалов

В данном разделе учитываются расходы на приобретение основных материалов, необходимых для проведения исследования, оформления соответствующей документации, а также учитывается стоимость картриджа. Расчет стоимости расходных материалов представлен в таблице 9.

Таблица 4.4 - Стоимость расходных материалов

Наименование расходных материалов	Цена за единицу, руб.	Количество, шт.	Сумма, руб.
1	**2**	**3**	**4**
Ноутбук	23000	1	23000
ПО «Arduino»	6000	1	6000
ПО «Audacity»	-	1	-
Бумага	170	2	340
Канцтовары	150	-	150
Расходные материалы для принтера (картридж)	3200	-	3200
Итого:			32690

Было определено, что для проведения исследования затраты на приобретение расходных материалов составят 32690 рублей.

4.5 Расчет сметы расходов на разработку

С учетом часовой тарифной ставки были рассчитаны общие расходы на разработку и проведение исследования. В данную статью расходов включаются премиальные выплаты, районный коэффициент и страховые взносы. Для оценки затрат на исследование была составлена смета на разработку и проведение исследования.

Был произведен расчет расходов:

Премиальные выплаты рассчитываются по формуле:

$$ПВ = P_{ОТ} K_{ПВ}, \quad (17)$$

где $К_{ПВ}$ - коэффициент премиальных выплат, составляет 20 %, в случае если

премии не предусмотрены $К_{ПВ}$=1.

$$ПВ = 36635,52\ 0,2=7327,10 \text{ руб.}$$

Дополнительные затраты на проведение исследования определяются по формуле:

$$З_{ДОП} = Р_{ОТ} К, \quad (18)$$

где К - коэффициент дополнительных затрат (К=14%).

$$З_{ДОП} = Р_{ОТ} \cdot 14\ \%.$$

$$З_{ДОП} = 36635,52 \cdot 0,14=5128,97$$

В заработной плате может быть предусмотрен районный коэффициент, характеризующий доплату при работе в трудных условиях. Величина коэффициента определяется в зависимости от характера производства.

$$РК = Р_{ОТ} К_{РВ} \quad (19)$$

где $К_{РВ}$ – коэффициент районных выплат, для примера составляет 15 % от суммы.

$$РК = (36635,52) \cdot 0,15 =5495,33 \text{ руб.}$$

Общие расходы на оплату труда вычисляются по формуле:

$$Р_{общ} = Р_{ОТ} + ПВ + РК + З_{ДОП} \quad (20)$$

где $Р_{ОТ}$ - основная заработная плата; ПВ - премиальные выплаты; $З_{ДОП}$ - дополнительные затраты; РК - районный коэффициент.

$$\Sigma Р_{ОТ} =36635,52+7327,10+5495,33+5128,97$$

$$\Sigma Р_{ОТ} =54586,92 \text{ руб.}$$

Из таблицы 9 берется итоговая сумма стоимости расходных материалов по статье расходных материалов.

$$\Sigma Р_{РМ} =3690 \text{ руб.}$$

Страховые взносы рассчитываются по формуле:

$$СВ = Р_{ОТ} 0,3, \quad (21)$$

$$СВ =36635,52 \cdot 0,30=10990,66$$

Амортизационные исчисления на использование компьютера составляют

25% от стоимости компьютера и вычисляются по формуле.

$$AO = C_{ПК} 0{,}25, \quad (22)$$

$$АО = 28000 \cdot 0{,}25 = 7000 \text{ руб.}$$

Расходы на использование Интернета берутся из расчета месячной абонентской платы для предприятия. Пусть:

$$Р_{ИНТ} = 1250 \text{ руб.}$$

Административно-хозяйственные расходы составляют 50% от основной заработной платы ($Р_{ОТ}$).

$$P_{AX} = P_{OT} 0{,}5, \quad (23)$$

$$Р_{АХ} = 36635{,}52 \cdot 0{,}5 = 18317{,}76 \text{ руб.}$$

Результаты расчета расходов были сведены в таблицу. Смета расходов на разработку и проведение исследования представлена в таблице 4.5.

Таблица 4.5 - Смета расходов на разработку и проведение исследования

Наименование статей расходов	Сумма, руб.	Удельный вес статей, %
1	**2**	**3**
1.Стоимость расходных материалов	32690	6,92
2. Расходы на оплату труда 2.1. Основная заработная плата	54586,92 36635,52	 33,36
2.2. Дополнительные затраты	5128,97	4,67
2.3. Премиальные выплаты	7327,10	13,35
2.4 Районный коэффициент	5495,33	5,0
3. Единый социальный налог	10990,66	11,66
4. Амортизационные исчисления на использование компьютера	7000	7,36
5. Расходы на использование Интернет	1250	0,99
6.Административно-хозяйственные расходы	18317,76	16,68
Итого:	95835,34	100

Результатом экономической оценки исследования является определение затрат на разработку и реализацию исследования:

- продолжительность исследовательских работ составила 60 дней;

- сметы расходов на исследование – 95 835 рублей.

ЗАКЛЮЧЕНИЕ

В ходе разработки российского аналога анализатора WiFi сети было собранно устройство и разработано собственное программное обеспечение с функциональностью соответствующей целям и задачам выпускной квалификационной работы, а также были произведены измерения в лабораторных условиях и местах, с развернутой беспроводной сетью. В процессе измерений устройство показало стабильную работу и малое энергопотребление составляющее в среднем 170 мА при напряжение питания 5 В, что намного меньше чем у современных смартфонов, потребляющих не менее 300 мА при напряжении 5 В. В процессе демонстрации специалистами по развертыванию беспроводных сетей было отмечено простота использования устройства не требующая длительного обучения, что позволяет использовать устройство специалистами любой квалификации.

Основными преимуществами разработанного устройства являются:

1. Является возможность быстрой диагностики занятости каналов беспроводной сети WiFi для развертывания сети с максимальным уровнем сигнала.
2. Возможность подключения внешней антенны как для измерения параметров сети с антенной заказчика, так и для точной настройки радиомостов.
3. Длительное время работы от аккумулятора.
4. Полностью российское программное обеспечение, написанное на языке С, что позволяет модернизировать программную и аппаратную часть.
5. Простота в эксплуатации

Устройство было продемонстрировано специалистам компании ООО «Белгородские Региональные ТелеСистемы», которые проявили большой интерес к разработанному устройству и заинтересованы в использовании подобного оборудования в работе

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гольдштейн Б.С. Беспроводные сети доступа [Текст] // Б.С. Гольдштейн, - М.: Радио и связь, 200.-317с.
2. Одом У. Официальное руководство по подготовке к сертификационным экзаменам CCNA Маршрутизация и коммутация, академическое издание [Текст] // У. Одом - М.: Вильямс, 2015. -761с.
3. Петин В.А. Проекты с использованием контроллера Arduino-СПБ: БХВ-Петербург, 2014 год, 400с
4. Официальный сайт по документации проекта Arduino. [Электронный ресурс] Arduino платформа для проектов Режим доступа: http://www.arduino.cc Дата обращения (17.05.2017)
5. Э.Таненбаум., Д.Уэзеролл. компьютерные сети. 5-е издание.- СПБ-Питер, 2012 год-960с.
6. Д.Гейер. Беспроводные сети. Первый шаг. Издательский дом «Вильяме», 2005 год.-192с
7. СН 512-78 Инструкция по проектированию зданий и помещений для электронно-вычислительных машин, редакция №2 [Электронный ресурс] // Каталог ГОСТ Е.: URL: http://docs.cntd.ru/document/901707386/ (Дата обращения 15.04.17)
8. Руководящий технический материал «Принципы обеспечения безопасности на объектах связи» [Текст]– ФГУП ЦНИИС, 2010.- 145 с.

ПРИЛОЖЕНИЕ А

Листинг 1.

```
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>

#define KEYDELAY  20000      //Задержка циклов до следующего нажатия
#define WIFIDELAY 20000      //Задержка циклов до следующего сканирования модулем ESP8266
#define LCDMAXROW 1          //Максимальное кол-во строк на экране в символах
#define LCDMAXCOL 15         //Максимальное кол-во столбцов на экране в символах
#define MAXGUILINE 16        //Максимальное кол-во отображаемых экраном строк в пикселях
#define MAXWIFICHANNEL 13    //Максимальный канал
#define STATIONDATALEN 70    //Минимальная длина буфера команды описания станции
#define MINSTATIONDATALEN 47 //Минимальная длина буфера строки описания станции
#define SSIDLENGHT 32        //Длина SSID станции
#define MACADRSIZE 17        //Длина MAC адреса
#define RSSIRESTCOUNT 32     //Количество измерений для увеличения максимального RSSI
#define MSRRESETCOUNT 60000  //Количество измерений для увеличения максимального RSSI
#define SCANBUFFERSIZE  1024 //Размер буфера приема


//Глобальные переменные
char scanData[SCANBUFFERSIZE];  //Буфер для приема
byte curMode=0;                 //Текущий режим - 0 скарирование, 1 - вывод сетей по каналу, 2 - инормация по конкретной сети
byte currx=0;                   //позиция курсора по X
byte curry      =LCDMAXROW;     //позиция курсора по Y (низ экрана)
byte currStNum;                 //Текущая станция для режимов 1 и 2
byte currChannel =1;            //Текущий канал для режимов 1 и 2

word  analogDelay =KEYDELAY;    //Счетчик задержек для плавности нажатия
word  wifiDelay   =WIFIDELAY;   //Счетчик задержек для сканирования WiFi
word  maxWiFilevel=0;           //Максимальный уровень сигнала
word  maxWiFiDev=1;             //Кол-во еденицы максиального уровня сигнала
word scanBuffPos=0;             //Текущий индекс буфера
word msrCounter=0;              //Глобальный счетчик измерений для коррекции максимального RSSI

bool drawCursor;                //Отображать курсор - не отрисовывать до получения первой станции

struct channelWiFi
  {
    byte stNumber;  //Номер станции в списке
    byte ecn;       //Тип шифрования
    byte channel;   //Канал
    byte maxrssi;   //максимальный RSSI
```

```
    byte minrssi;   //минимальный RSSI
    byte maxnoise;  //максимальный noise
    byte minnoise;  //минимальный noise
    char mac[MACADRSIZE];
    char ssid[SSIDLENGHT]; //SSID точки
    word rssi;     //RSSI * 100 (для точности)
    word noise; //* 100
    unsigned int rssicount; //Счетчик значений RSSI
    unsigned int noisecount; //Счетчик значений noise
    word msrCount;  //Кол-во измерений
    struct channelWiFi *nextchannel;
  }; //Стуктура для хранения данных информация о станциях

struct channelData            //Структура для хранения значений каналов
  {
  word maxValue;              //Максимальное знаение уровня сигнала * 100
  word minValue;              //Минимальное значение уровня сигнала * 100
  word currValue;             //Текущие значение  * 100
  byte drawlevel;             //Отображаемы уровень
  byte numberStation;         //Количество станций
  struct channelWiFi *firstchannel;      //Указатель на первую станцию
  } channelsData[MAXWIFICHANNEL+1]; //Нулевой канал не используется

  LiquidCrystal lcd(8, 9, 4, 5, 6, 7);//Подключение дисплея

void setup() {
  //Символы уровня сигнала
  byte level1[8]={B00000,B00000,B00000,B00000,B00000,B00000,B00000,B11110}; //Уровень 1
  byte level2[8]={B00000,B00000,B00000,B00000,B00000,B00000,B11111,B11110}; //Уровень 2
  byte level3[8]={B00000,B00000,B00000,B00000,B00000,B11110,B11111,B11110}; //Уровень 3
  byte level4[8]={B00000,B00000,B00000,B00000,B11111,B11110,B11111,B11110}; //Уровень 4
  byte level5[8]={B00000,B00000,B00000,B11110,B11111,B11110,B11111,B11110}; //Уровень 5
  byte level6[8]={B00000,B00000,B11111,B11110,B11111,B11110,B11111,B11110}; //Уровень 6
  byte level7[8]={B00000,B11110,B11111,B11110,B11111,B11110,B11111,B11110}; //Уровень 7
  byte level8[8]={B11111,B11110,B11111,B11110,B11111,B11110,B11111,B11110}; //Уровень 8
  Serial.begin(115200);             //Установка скорости порта свзяи с ПК
  Serial3.begin(9600);              //Установка скорости порта связи с ESP8266
  Serial3.setTimeout(5000);         //Время отжидания порта
  Serial3.println("AT+CWMODE=1");   //Переключение в режим точки доступа
  Serial3.println("ATE0");          //Отключение эха

  //Загрузка символов в дисплей
  lcd.createChar(1,level1);
  lcd.createChar(2,level2);
  lcd.createChar(3,level3);
  lcd.createChar(4,level4);
  lcd.createChar(5,level5);
  lcd.createChar(6,level6);
  lcd.createChar(7,level7);
  lcd.createChar(8,level8);
```

```
  //Инициализация дисплея
  lcd.begin(LCDMAXCOL+1,LCDMAXROW+1);    //Устанавливается размер экрана 16x2

}

void loop() {
  byte currPressedKey=0; //Нажатая кнопка неопределена
  scanWiFiNetwork();       //Запуск сканирования сети
  //debugControlSerial();
  serialReadToBuffer();

  parseWiFiBuffer();
  ////////////////////////////////////////////////////////////////////////////////////////////
  //Управление клавиатурой
  currPressedKey=getPressedKey();

  if (currPressedKey>0) {//Была нажата клавиша
     //debugShowPressedKey(currPressedKey);  //отладочный вывод нажатой кнопки

     moveCursor(currPressedKey);          //Обрабатываем клавиатурный ввод

     //Принудительная отрисовка интефрейса
     drawGui();                           //Отрисовываем интерфейс

     //Отрисовка курсора
     drawCurrentCursor();                 //Перерисовываем курсор
  }//Окончание функции определения нажатой кнопки

}//Окончание основного цикла программы


void(* resetFunc) (void) = 0;


//Сканирование сети
void scanWiFiNetwork()
{
    byte wifiBufferSize=0; //Размер буфера

    if (wifiDelay   == WIFIDELAY) {//Сканирование разрешено
      //Запуск сканирования WiFi сети
      Serial3.println("AT+CWLAP");
      wifiDelay = 0;
    }

   else  //Задержка для сканирования
   {
      if (wifiDelay<WIFIDELAY) {
         wifiDelay++;
      }
   }
```

```
}

//Отображение курсора
void drawCurrentCursor(void){

  if (drawCursor) {
    lcd.setCursor(currx,curry);
    //lcd.cursor();
    lcd.blink();
  }

}

//Управление курсором
void moveCursor(byte pressedKey){

  if (curMode==0) {
    //Обработка управления при режиме работы 0
     if (pressedKey==4) {//Кнопка лево

       if (currx>0) {
        moveCursorLeft();
       }
       currChannel=currx+1; //Запоминаем текущий канал
      }
      else if (pressedKey==1) {//Кнопка вправо

       moveCursorRight();
       if ((currx>MAXWIFICHANNEL-1)&&(currx!=15))
       {
        currx=MAXWIFICHANNEL-1;

       }
       if (currx<MAXWIFICHANNEL) {
        currChannel=currx+1; //Запоминаем текущий канал
       }
      }
      else if (pressedKey==5) {//Кнопка выбор
         //Проверка возможности перейти в режим просмотра станций
         if (currx<MAXWIFICHANNEL) {
          if (channelsData[currx+1].numberStation>0) {//Есть станции на текущем канале
           //Смена режима
           currStNum = 1; //Отрисовка начинается с первой станции
           currChannel=currx+1; //Запоминаем текущий канал
           clearScreen(); //Очистка экрана
           curMode=1;     //Переход в режим 1
           currx  =0;
          }
         }
         else
         {
```

```
            resetFunc();
          }
        }

    }

    else if (curMode==1) {
       //Обработка управления при режиме работы 1
       if (pressedKey==4 || pressedKey==1) {//Кнопки лево/право возвращают в режим 0
          clearScreen(); //Очистка экрана
          curMode=0;     //Переход в режим 0
          currx=currChannel-1; //Восстанавливаем текущий канал

       }
       else if (pressedKey==2) {//Кнопка вверх
         if (currStNum>1) {
            currStNum--;
         }
         else
         {//Переход в конец списка
          currStNum=channelsData[currChannel].numberStation;
         }

       }
       else if (pressedKey==3) {//Кнопка вниз
          currStNum++;
          if (currStNum>channelsData[currChannel].numberStation) {
             currStNum=1;
          }

       }
       else if (pressedKey==5) {//Кнопка выбор
          //Смена режима
          clearScreen(); //Очистка экрана
          curMode=2;     //Переход в режим 2
       }

    }
    else if (curMode==2) {
       //Обработка управления при режиме работы 2
       //if (pressedKey==5) {//Кнопка выбор
          clearScreen(); //Очистка экрана
           //Смена режима
           curMode=0;     //Переход в режим 0
          currx=currChannel-1; //Восстанавливаем текущий канал

       //}
    }

}
```

```
void clearScreen(void){//Очищает экран при смене режима
  byte countX;
  byte countY;
  for (countY=0; countY<LCDMAXROW+1; countY++) {//Цикл по строкам
     for (countX=0; countX<LCDMAXCOL+1; countX++) {//Цикл по строкам
        lcd.setCursor(countX,countY);
        lcd.print(" ");
     }
  }
}

void moveCursorLeft(){//Перемещает курсор на ближайшую левую станцию станцию
   byte contr;

   if (currx>MAXWIFICHANNEL) {
      currx=MAXWIFICHANNEL;
   }
   for ( contr=currx; contr>0;contr--)
   {
    if (channelsData[contr].numberStation>0) {//Найдена станция слева

      currx=contr-1;
      break;
    }
   }

}

void moveCursorRight(){//Перемещает курсор на ближайшую правую станцию станцию
   byte contr;
   boolean currMoved=false;

   if (currx<MAXWIFICHANNEL-1) {
    for ( contr=currx+2; contr<MAXWIFICHANNEL+1;contr++)
    {
      if (channelsData[contr].numberStation>0) {//Найдена станция справа
      currMoved=true;
      currx=contr-1;
      break;
      }
    }
   }

   if (!currMoved) {
      currx=15;
   }
}


//Отрисовываем пользовательский интерфейс
void drawGui(void){
```

```
    byte contr;         //Счетчик цикла
    byte stCount;
    byte tempByte;      //Байт для временных расчетов
    char tempChar;      //Временный буфер
    struct channelWiFi *CurrStPtr;
    if (curMode==0) {
     //Обработка управления при режиме работы 0
     byte counterOver10;  //10,11,12,13 канал отображаются как 0,1,2,3

     printBytetoLCD(currChannel,LCDMAXCOL-1,0);
;

     for ( contr=1; contr<MAXWIFICHANNEL+1;contr++)
     {
      if (channelsData[contr].drawlevel>16) {

        channelsData[contr].drawlevel=16;
      }
      if (contr ==  currx+1) {//Выводим кол-во точек текущего канала
       stCount =   channelsData[contr].numberStation;
       lcd.setCursor(LCDMAXCOL-1,1);
       lcd.print(stCount);
       if  (stCount<10){//Затираем лишний символ
          lcd.setCursor(LCDMAXCOL,1);
          lcd.print(" ");
        }
      }

     //В любом случае верхний ряд пустой
      if (channelsData[contr].drawlevel>8) {//Первый ряд
        lcd.setCursor(contr-1,0);
        lcd.write(channelsData[contr].drawlevel-8);
        lcd.setCursor(contr-1,1);
        counterOver10=contr;
        if (counterOver10>9)
        {
         counterOver10=counterOver10-10;
        }
        lcd.write(48+counterOver10); //Отрисовываем номер канала
       }
       else //Нижний ряд
       {
        lcd.setCursor(contr-1,0);
        lcd.print(" ");

        if (channelsData[contr].drawlevel>0) {//Первый ряд
          lcd.setCursor(contr-1,1);
          lcd.write(channelsData[contr].drawlevel);

        }
        else //Пусто
```

```
      {
       lcd.setCursor(contr-1,1);
       lcd.print(" ");
      }
     }

   }
}//Окончание отрисовки режима №1
else if (curMode==1) {
  //Отрисовка при режиме работы 2

  CurrStPtr=getStationData(currStNum); //Получаем указатель на станцию по номеру
  for (contr=0; contr<LCDMAXCOL+1;contr++){
    //Выводим название станции
    lcd.setCursor(contr,0);
    tempChar=CurrStPtr->ssid[contr];
    if (tempChar==0) tempChar=32; //Пробелы
    lcd.print(tempChar);

  }
  //Выводи номер точки и общее кол-во точек
  printOneBytetoLCD(currStNum,0,1);
  lcd.setCursor(1,1);
  lcd.print("/");
  printOneBytetoLCD(channelsData[currChannel].numberStation,2,1);

  lcd.setCursor(6,1);
  lcd.print("/-");
  printBytetoLCD(CurrStPtr->rssi/100,8,1);
  lcd.setCursor(3,1);
  lcd.print("-");
  printBytetoLCD(CurrStPtr->noise/100,4,1);
  lcd.setCursor(11,1);
  lcd.print("     ");//Очистка знакомест
  lcd.setCursor(11,1);
  lcd.print(CurrStPtr->msrCount); //Вывод кол-ва измерений

}
else if (curMode==2) {
  //Отрисовка при режиме работы 3
  byte contr;         //Счетчик цикла
  byte charContr    = 0;//Счетчик символов
  char printChar;     //Выводимый символ
  byte startLCDPos  = 0;//Стартовая позиция для вывода
  String ChName     = "";
  CurrStPtr=getStationData(currStNum); //Получаем указатель на станцию по номеру
  for (contr=0; contr<MACADRSIZE;contr++){
     printChar=CurrStPtr->mac[contr];
     if (printChar==58) {//Первые 2 символа : пропускаются
       if (charContr<1) {
         charContr++;
```

```
            continue;//Продолжаем цикл без вывода символа :
          }

        }
        lcd.setCursor(startLCDPos,0);
        lcd.write(printChar);
        startLCDPos++;

    }
    //Отображения типа шифрования
    ChName=getChannelEnc(CurrStPtr->ecn,true);
    for (contr=0; contr<3;contr++){
        lcd.setCursor(contr,1);
        lcd.print(ChName[contr]);

    }
    //Выводим минимальный средний и макс шум
    lcd.setCursor(5,1);
    lcd.print("-");
    printBytetoLCD(CurrStPtr->minrssi,6,1);
    lcd.setCursor(9,1);
    lcd.print("-");
    printBytetoLCD(CurrStPtr->rssicount/CurrStPtr->msrCount,10,1);
    lcd.setCursor(13,1);
    lcd.print("-");
    printBytetoLCD(CurrStPtr->maxrssi,14,1);



  }
}
void printBytetoLCD(byte printByte, byte xPos, byte yPos){//Вывод числа на LCD в формате NN
    byte tempByte = printByte;
    //Убираем "лишние" символы
    if (tempByte>200) {
      tempByte=tempByte-200;
    }
    if (tempByte>100) {
        tempByte=tempByte-100;
    }


    if (tempByte<10) {
        lcd.setCursor(xPos,yPos);
        lcd.print("0");
        lcd.setCursor(xPos+1,yPos);
        lcd.print(tempByte);
    }
    else
    {
        lcd.setCursor(xPos,yPos);
```

```
        lcd.print(tempByte);
    }
}

void printOneBytetoLCD(byte printByte, byte xPos, byte yPos){//Вывод числа на LCD в формате
N
    byte tempByte = printByte;
    bool byteCorr = false; //Истина если была коррекция
    //Убираем "лишние" символы
    if (tempByte>200) {
      tempByte=tempByte-200;
      byteCorr=true;
    }
    if (tempByte>100) {
       tempByte=tempByte-100;
       byteCorr=true;
    }

    if  (tempByte>10) {
       byteCorr=true;
       while (tempByte>10) {
          tempByte=tempByte-10;
        }
    }

    lcd.setCursor(xPos,yPos);
    if (byteCorr) {
     lcd.print("*");
    }
    else
    {
    lcd.print(tempByte);
    }
}
struct channelWiFi *getStationData(byte stNum){//Возвращает ссылку на структуру данных по
номеру станции

    struct channelWiFi *nextChPtr; //Указатель на проверяемую стуктуру

    nextChPtr = channelsData[currChannel].firstchannel; //Получение указателя на первый канал
массива
    while (nextChPtr != NULL){

      if (nextChPtr->stNumber == stNum) {//Ищем станцию по номеру
          break; //Прекращаем поиск
      }
      nextChPtr = nextChPtr->nextchannel; //Следующая станция в списке
    }
    return nextChPtr;
}
```

```
//Получение значение нажатой кнопки
//Возвращает:
//0 - Нет нажатия
//1 - Право
//2 - Вверх
//3 - Вниз
//4 - Лево
//5 - Выбор
byte getPressedKey(void)
{
   byte pressedKey=0; //Нажатая кнопка (0 - ничего не нажато)
   int analogInp=0;   //Переменная для чтения уровня аналогового порта A0

   if (analogDelay==KEYDELAY) {
    analogInp   = analogRead(0);

    if (analogInp < 100)      {// Если x меньше 100 перейти на следующею строк
      pressedKey=1;
      analogDelay = 0;
    }
    else if (analogInp < 200) {// Если x меньше 200 перейти на следующею строк
      pressedKey=2;
       analogDelay = 0;
    }
    else if (analogInp < 400) {// Если x меньше 400 перейти на следующею строк
      pressedKey=3;
      analogDelay = 0;
    }
    else if (analogInp < 600) {// Если x меньше 600 перейти на следующею строк
      pressedKey=4;
      analogDelay = 0;
    }
    else if (analogInp < 800) {// Если x меньше 800 перейти на следующею строк
      pressedKey=5;
      analogDelay = 0;
    }

   }
  else  //Задержка для плавности нажатия кнопок
  {
    if (analogDelay<KEYDELAY) {
       analogDelay++;
    }
  }
  return pressedKey;
}

void serialReadToBuffer(void){
  byte wifiBufferSize=0; //Размер буфера
  char wifiReadChar;
  wifiBufferSize=Serial3.available();
```

```
  if (wifiBufferSize>0)
  {
    while(wifiBufferSize>0){
     //Чтение поступившего байта
     wifiReadChar  = Serial3.read();
     scanData[scanBuffPos] = wifiReadChar; //Сохранение в буфер
     scanBuffPos++;                    //Перемещаем буфер на следующую позицию
     if (scanBuffPos>SCANBUFFERSIZE) {     //Выход за границы буфера
        scanBuffPos=0;
     }

     if (wifiDelay>0) {
        wifiDelay--;
      }
     wifiBufferSize=Serial3.available();
    }

  }
}

void parseWiFiBuffer(void){//Анализ буфера ответа от радиомодуля
   word contr;         //Счетчик цикла
   word contr2;        //Счетчик цикла 2
   word contr3;        //Счетчик цикла 3
   word posEOF;        //Позиция окончания данных
   char stationData[STATIONDATALEN]; //Буфер строки описания станции
   byte currStData;                  //Размер буфера
   bool parseError=false;            //Ошибка при разбор

    if (scanBuffPos>80) {
     //Поиск символов окончания сканирования
     for ( contr=0; contr<scanBuffPos-3;contr++) {
      if ((scanData[contr]   == 79) &&
         (scanData[contr+1] == 75) &&
         (scanData[contr+2] == 13) &&
         (scanData[contr+3] == 10))
         {
           //Получены символы окончания сканирования
            if (parseError) {
              break;
            }


            posEOF=contr-1;
            for ( contr2=0; contr2<posEOF-8; contr2++) {


              //Поиск строки +CWLAP:(
              if ((scanData[contr2]   == 43) &&
                 (scanData[contr2+1] == 67) &&
```

```
                (scanData[contr2+2] == 87) &&
                (scanData[contr2+3] == 76) &&
                (scanData[contr2+4] == 65) &&
                (scanData[contr2+5] == 80) &&
                (scanData[contr2+6] == 58) &&
                (scanData[contr2+7] == 40))
                {
                  if  (contr2<SCANBUFFERSIZE-MINSTATIONDATALEN){  //Определение целесообразности сканирования,
                                          //минимальная порция данных 47 символов
                    //Начало строки выдачи
                    contr3=contr2+8;
                    currStData  = 0;  //Начало буфера
                    while (contr3<posEOF-2) {

                      //Проверка окончания строки данных
                      if (((scanData[contr3]     == 41) && //Получено окончание строки описания буфера или буфер заполнен
                          (scanData[contr3+1] == 13) &&
                          (scanData[contr3+2] == 10)) || (currStData == STATIONDATALEN)){
                           //Разбор строки буфера
                           parseWiFiStation(stationData,currStData);
                           break;
                       }
                       //Первый символ в строке номер канала
                      stationData[currStData]=scanData[contr3];
                      currStData++;
                      contr3++;

                    }

                  }
                   else {
                     parseError= true;
                     break; //Команда не может быть декодирована из-за недостаточной длинны
                    }
                }

            }

          //Очистка буфера и выход
          for (contr2=0; contr2<SCANBUFFERSIZE;contr2++) {
             scanData[contr2]=0;
           }
          scanBuffPos =0;
          break;//Выход из цикла
        }
    }
  }
```

```
}

void parseWiFiStation(char *stBuffer, byte buffLen){
   byte contr;              //Счетчик цикла
   byte startParseCountr=3;  //Счетчик начала позиции SSID станции
   byte currStSSIDpos=0;     //Текущая позиция
   String tempStr="";        //Строка для преобразования строк в числа
   byte tempByte = 0;        //Байт для хранения временных величин

   if (buffLen>=MINSTATIONDATALEN-10) {//Проверка минимальной длины строки - за вычетом символов
      //Первый символ тип шифрования

      struct channelWiFi parsechannel;

      tempStr = String(stBuffer[0]);
      parsechannel.ecn  = tempStr.toInt();

      //Получение SSID
      currStSSIDpos=0;
      //Очистка буфера
      for (contr=0; contr<SSIDLENGHT; contr++){
         parsechannel.ssid[contr]=0;
      }

      while (startParseCountr<buffLen-1){
          if (((stBuffer[startParseCountr]   == 34) && //Получены символы окончания SSID
             (stBuffer[startParseCountr+1] == 44)) || (currStSSIDpos>SSIDLENGHT))  {
             break;
         }
         parsechannel.ssid[currStSSIDpos] = stBuffer[startParseCountr];
         startParseCountr++; //Следующий символ
         currStSSIDpos++;    //Следующая позиция в буфере

      }
      currStSSIDpos--; //Уменьшаем размер буфера

      //Получение RSSI
      startParseCountr=startParseCountr+2;
       if (stBuffer[startParseCountr]   == 45) {

         //Получение RSSI
         tempStr = String(stBuffer[startParseCountr+1]);
         tempStr = tempStr+String(stBuffer[startParseCountr+2]);
         tempByte= tempStr.toInt();
         parsechannel.rssi   = tempByte*100;
         //Заполнение максимальных/минимальных величин
         parsechannel.maxrssi  = tempByte;
         parsechannel.minrssi  = tempByte;
         parsechannel.rssicount= tempByte;
```

```
          startParseCountr=startParseCountr+5;
          //Получение MAC
            for (contr=0; contr<MACADRSIZE; contr++){
               parsechannel.mac[contr]=stBuffer[startParseCountr+contr];
            }
             startParseCountr=startParseCountr+contr;

            //Получаем номер канала
             if ((stBuffer[startParseCountr]  == 34) && //Получены символы окончания MAC
               (stBuffer[startParseCountr+1] == 44)){
               startParseCountr=startParseCountr+2;
               tempStr = String(stBuffer[startParseCountr]);
               if (stBuffer[startParseCountr+1]!=44) {//Номер канала может состоять из 1 или 2 символов
                  startParseCountr++;
                  tempStr = tempStr+String(stBuffer[startParseCountr]);
               }
               parsechannel.channel  = tempStr.toInt();

               //Получение шума в канале
               if  ((stBuffer[startParseCountr+1] == 44) && //Получены символы окончания номера канала
                   (stBuffer[startParseCountr+2] == 45)){
                     startParseCountr=startParseCountr+3;
                     tempStr = String(stBuffer[startParseCountr]);
                     tempStr = tempStr+String(stBuffer[startParseCountr+1]);
                     tempByte= tempStr.toInt();
                     parsechannel.noise  = tempByte*100;
                      //Заполнение максимальных/минимальных величин
                     parsechannel.maxnoise   = tempByte;
                     parsechannel.minnoise   = tempByte;
                     parsechannel.noisecount = tempByte;
                     //Заполнение неопределенных значений
                     parsechannel.stNumber   = 0;  //Номер станции в списке
                     parsechannel.msrCount   = 1;  //Для новых станций
                     parsechannel.nextchannel= NULL;//Следующая станция в списке

                     //Обработка полученной структуры
                     checkWiFIStation(&parsechannel);

                  }
               }
            }
       drawGui();     //Отрисовываем интерфейс
       drawCurrentCursor();  //Перерисовываем курсор


     }

}
```

```
void checkWiFIStation(struct channelWiFi *sPtr){
  byte channelNum = sPtr->channel; //Получаем номер канала
  word currRSSI  = 10000-sPtr->rssi;//-sPtr->noise;
  word tempDrawLevel;
  byte contr;

  if (currRSSI>maxWiFilevel) {//Если полученный уровень сингнала больше максимального
    maxWiFilevel  =currRSSI;              //- увеличиваем максимальный
    maxWiFiDev    =maxWiFilevel/MAXGUILINE; //- изменяем уровень для GUI
  }

 //Обновляем общую статистику канала

 channelsData[channelNum].currValue = currRSSI;
 if (currRSSI>channelsData[channelNum].maxValue) {
    channelsData[channelNum].maxValue=currRSSI; //Увеличиваем текущий RSSI
 }
 if (currRSSI<channelsData[channelNum].minValue) {
    channelsData[channelNum].minValue=currRSSI; //Увеличиваем текущий RSSI
 }
 tempDrawLevel = channelsData[channelNum].currValue/maxWiFiDev;
 channelsData[channelNum].drawlevel = byte(tempDrawLevel);

 if (channelsData[channelNum].firstchannel == NULL) {//В базе каналов нет записей

    addFirstChannel(sPtr);
    if (!drawCursor) {
      currChannel=channelNum;
      currx=channelNum-1;
      drawCursor = true;
    }
 }
 else //В базе каналов есть записи
 {
    updateCurrentChannel(sPtr);
 }

 msrCounter++; //Увеличение счетчика измерений
 if (msrCounter>RSSIRESTCOUNT) {
    msrCounter=0;
  //Кореекция максиального RSSI
   if (maxWiFilevel>100) {

      maxWiFilevel  =maxWiFilevel-100;          //- увеличиваем максимальный
      maxWiFiDev    =maxWiFilevel/MAXGUILINE;   //- изменяем уровень для GUI

   }
 }
}

void updateCurrentChannel(struct channelWiFi *sPtr){//Добавление следующего канала в
```

```
список
    struct channelWiFi *nextChPtr; //Указатель на проверяемую стуктуру
    struct channelWiFi *newChPtr;  //Указатель на новую стуктуру
    struct channelWiFi *lastFoundwChPtr;  //Указатель на предпоследнюю стуктуру
    byte chLevel;
    bool dontAdd;       //Станция не найдена в списке - нужно добавить


    nextChPtr = channelsData[sPtr->channel].firstchannel; //Получение указателя на первый
канал массива
    while (nextChPtr != NULL){

        //Проверям канал по mac
        dontAdd=compareMAC(sPtr,nextChPtr);
        if (dontAdd) {//Эта станция уже есть в списке
            //Обновляем информацию
            nextChPtr->rssi = sPtr->rssi;
            nextChPtr->noise= sPtr->noise;
            nextChPtr->rssicount  = nextChPtr->rssicount + sPtr->rssi/100;
            nextChPtr->noisecount = nextChPtr->noisecount+ sPtr->noise/100;
            //Обнолвение максимальных/минимальных параметров канала
             chLevel     =sPtr->rssi/100;

             if (chLevel < nextChPtr->maxrssi) {//max rssi
                nextChPtr->maxrssi = chLevel;
             }

            if (chLevel > nextChPtr->minrssi) {
              nextChPtr->minrssi = chLevel;
            }
            chLevel     =sPtr->noise/100;

            if (chLevel < nextChPtr->maxnoise) {//max rssi
                nextChPtr->maxnoise = chLevel;
            }

            if (chLevel > nextChPtr->minnoise) {
               nextChPtr->minnoise = chLevel;
            }

            nextChPtr->msrCount++;//Обновляем кол-во измерений
            break;
        }
        lastFoundwChPtr = nextChPtr;
        nextChPtr = nextChPtr->nextchannel;
    }
    if (!dontAdd) {

       newChPtr  = malloc ( sizeof( struct channelWiFi ) );
       if ( newChPtr  != NULL ) {//Память успешно выделена
         memcpy ( newChPtr, sPtr, sizeof( struct channelWiFi ));
```

```
        //Привязка точки к массиву каналов
        lastFoundwChPtr->nextchannel=newChPtr;
        //Увеличение кол-ва станций
        newChPtr->stNumber=++channelsData[sPtr->channel].numberStation;
       }
       else
       {
        Serial.println("Не удалось выделить память под структуру channelWiFi!");
       }
      }

}

void addFirstChannel(struct channelWiFi *sPtr){//Добавление первого канала
     struct channelWiFi *newChPtr; //Указатель на новую стуктуру

     //Выделение памяти под стуктуру
     newChPtr  = malloc ( sizeof( struct channelWiFi ) );
     if ( newChPtr  != NULL ) {//Память успешно выделена
        //Копирование данных в новую стуктуру
        memcpy ( newChPtr, sPtr, sizeof( struct channelWiFi ));
        //Привязка точки к массиву каналов
        channelsData[sPtr->channel].firstchannel=newChPtr;
        //Увеличение кол-ва станций
        newChPtr->stNumber=++channelsData[sPtr->channel].numberStation;
     }
     else
     {
      Serial.println("Не удалось выделить память под структуру channelWiFi!");
     }
}

boolean compareMAC(struct channelWiFi *newChannel, struct channelWiFi *oldChannel){
   boolean isCompare = true; //Результат сравнения
   byte contr;               //Счетчик цикла

   for (contr=0; contr<MACADRSIZE; contr++){

      if (newChannel->mac[contr] != oldChannel->mac[contr]){
         isCompare = false;
         break;
      }
   }
   return isCompare;
}

String getChannelEnc(byte enc,bool shorName){
   String ChName;
   //Возвращает тип шифрования в полном или сокращенном виде
   //0 OPEN (доступ без пароля, доступ не защищен).
   //1 WEP
```

```
    //2 WPA_PSK
    //3 WPA2_PSK
    //4 WPA_WPA2_PSK

    if (shorName) {//Выводим сокращенный набор
      if (enc==0) {
        ChName="OPN";
      }else if (enc==1) {
        ChName="WEP";
      }else if (enc==2) {
        ChName="WPA";
      }else if (enc==3) {
        ChName="WP2";
      }else if (enc==4) {
        ChName="W2P";
      }
    }
    else
    {//Выводим полный набор
      if (enc==0) {
        ChName="OPEN";
      }else if (enc==1) {
        ChName="WEP";
      }else if (enc==2) {
        ChName="WPA_PSK";
      }else if (enc==3) {
        ChName="WPA2_PSK";
      }else if (enc==4) {
        ChName="WPA_WPA2_PSK";
      }
    }

    return ChName;
}
```