Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Петрозаводский государственный университет»
Физико-технический институт
Кафедра информационно-измерительных систем и физической электроники

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ LORA В КОНЦЕПЦИИ IOT

ВЫПУСКНАЯ РАБОТА
на квалификацию бакалавра
по направлению подготовки
23.01.02 “Автоматизированные системы обработки информации и управления”

Автор работы:
студент группы 21412
_____ В. Б. Охотников
«____» _____ 2018 г.

Научный руководитель:
канд. физ.-мат. наук, доцент
_____ А. П. Мощевикин
«____» _____ 2018 г.

Петрозаводск 2018

РЕФЕРАТ

Отчет содержит 64 стр., 19 рис., 8 табл., 23 источников, 1 прил.

Ключевые слова: LoRa, Интернет вещей, M2M, беспроводные технологии передачи данных, открытое программное обеспечение, разработка ПО.

Данная работа ставит перед собой цель изучения технологии беспроводной передачи данных LoRa в контексте её использования в Интернете вещей.

В работе кратко отображено современное состояние Интернета вещей и беспроводных технологий передачи небольших по объёму данных на дальние расстояние и с низким энергопотреблением. Рассмотрен способ использования принципа открытого программного обеспечения для создания конечных устройств в сети LoRaWAN.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В настоящей выпускной работе применяют следующие термины с соответствующими определениями.

Интернет — глобальная вычислительная сеть. Является самой большой компьютерной сетью в мире. Построена на базе стека протоколов TCP/IP.

Трансивер — приёмопередатчик.

Сетевой протокол — соглашение о наборе правил, позволяющих проводить соединение и обмен данными между двумя и более устройствами, подключенным к сети.

Сеть последующих поколений (СПП) (Next generation network) — Сеть с пакетной коммутацией, пригодная для предоставления услуг электросвязи и для использования нескольких широкополосных технологий транспортировки с подключенной функцией QoS, в который связанные с обслуживанием функции не зависят от примененных технологий, обеспечивающих транспортировку.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

LPWAN — Low-power Wide-aread Network — энергоэффективная сеть дальнего радиуса действия.

LoRa™— Long Range — технология энергоэффективной беспроводной связи дальнего действия.

UL — Up-link восходящий канал связи.

DL — Down-link — нисходящий канал связи.

CSS — Chirp spread spectrum — линейная частотная модуляция.

ISM — industrial, scientific and medical radio bands— международные зарезервированные частотные диапазоны.

NGN — Next Generation Network — сеть последующих поколений.

IoT — с англ. *Internet of Things* в переводе — *“Интернет вещей”*.

ВВЕДЕНИЕ

С ростом информационных технологий и коммуникаций в XXI веке новые тенденции проявляются в виде интеллектуальной обработки больших массивов данных (Big Data) и Интернете вещей (IoT или IdC). Интернет не только позволил объединить людей, но также и устройства, датчики в сложные системы. Так появляется термин «Интернет вещей», который соответствует соединению различных устройств с Интернетом, генерирующие данные в реальном времени. Так называемые сети с низким энергопотреблением и большим покрытием (LPWA) являются мостом к Интернету Вещей, разработанные с целью заполнить брешь в технологиях с низким энергопотреблением, большим покрытием и низкой стоимостью. Одной из лидирующих технологии LPWAN является LoRaWAN™. Она и станет объектом рассмотрения данной работы.

Целью данной работы является исследование технологии LoRa™и оценка применимости данной технологии для инфраструктуры Интернета вещей.

Поставленные задачи:

— изучить возможные применения технологии в контексте развития Интернета вещей;

— сравнить открытый стандарт LoRaWAN™с другой технологией беспроводной передачи данных, такой как SigFox;

— оценить зону покрытия в городской среде для радиоприёмника LoRa™SX1278 в разных режимах работы;

— создать программное обеспечение для работы с SX1278 в связке с STM32L476, используя инструменты открытого программного обеспечения;

— задокументировать полученное программное обеспечение.

1 Аналитический обзор

1.1 Что такое Интернет вещей

Интернет вещей (англ. *IoT*, *Internet of Things*) — это методология вычислительной сети физических объектов (“*вещей*”), имеющих встроенную поддержку технологий передачи данных для их взаимодействия, а также для взаимодействия с внешней средой. Эта методология рассматривает Интернет вещей как явление, способное перестроить культурные и экономические процессы, всё больше исключая человека из них. Влияние существующего Интернета на сферы образования, коммуникации, бизнеса, науки и политики позволяет говорить о том, что Интернет является одним из важнейших и мощнейших изобретений в истории человечества [1]. Интернет вещей стоит рассматривать как новую ветвь эволюции Интернета, где каждый предмет в поле зрения человека может быть оснащён датчиками, сенсорами, устройством управления и модулем передачи данных для общения со всем миром.

Как известно, большинство великих изобретений человечества потребовали десятки и даже сотни лет на переход от простых по форме представлений до сложных систем. От создания предпосылок, до массового внедрения Интернета ушло почти четверть века, однако похоже что для Интернета вещей на то же самое потребуется существенно меньше времени [2]. Международный союз связи (МСЭ) и Европейский Союз определили Интернету вещей главенствующую роль в дальнейшем развитии информационных технологий. По расчетам консалтингового подразделения Cisco IBSG (см. рис. 1.1) в промежутке между 2008 и 2009 годами, количество устройств, подключенных к интернету, превысило количество людей, и к 2020 году количество подключенных устройств достигнет 50 миллиардов [1] (по другим данным [3] — 25 миллиардов). Таким образом, в настоящее время происходит переход от “Интернета людей” к “Интернету вещей”. Хотя данная концепция на международном уровне уже обретает черты сформировавшейся технологии, для неё ведутся активные работы в области стандартизации компонентов, архитектуры и приложений. Количество мнений о том как будет построен Интернет вещей очень велико. Это подтверждается большим разнообразием предлагаемых технологий для создания LPWAN сетей на рынке.

1.1.1 Базовые принципы Интернета вещей

Интернет вещей основывается на трёх базовых принципах [4].

а) повсеместно распространенная инфраструктура;

б) глобальная идентификация каждого объекта;

в) возможность каждого физического объекта отправлять и получать данные, посредством локальной сети или сети Интернет, к которой он подключен.

Наиболее важными отличиями Интернета вещей от интернета людей являются:

— фокус на считывание информации, а не на коммуникациях;

— на порядки большее число подключенных к сети объектов;

— потребность в создании новых стандартов;

— намного меньше размеры объектов и скорости передачи данных;

— фокус не на человеке, а на вещах;

Концепция сетей следующего поколения NGN предполагала возможность коммуникаций людей в любой точке пространства и времени. Концепция интернета вещей включает ещё одно направление — коммуникация любых вещей или устройств (рис. 1.2)

Согласно принятым в МСЭ-Т представлениям о отображении физических и виртуальных вещей, виртуальные вещи могут обходиться без их физического соответствия, в то время как каждой физической вещи соответствует минимум один объект в виртуальном пространстве (см. рис. 1.3).

Рекомендация Y.2060 от МСЭ-Т описывает различное сочетание способов соединений. МСЭ-Т рассматривает множество сетевых технологий, как потенциально пригодных для приложений Интернета вещей, а именно: глобальные сети, локальные сети, ячеистые (mesh) сети и беспроводные самоорганизующиеся (ad-hoc) сети.

1.1.2 Основные характеристики Интернета вещей

IoT, имеет следующие характеристики:

— возможность установления соединений. Любую вещь можно соединить к Интернету вещей;

— гетерогенность: устройства в концепции Интернета вещей являются гетерогенными и базируются на различных аппаратных платформах и сетях. Могут обмениваться информацией с другими устройствами, независимо от структуры сети и применяемых технологий транспортного уровня. Примечательно, что современное состояние сети Интернет может удовлетворить этому лишь отчасти: пул адресов IPv4 исчерпан и большая часть устройств скрывается в локальных сетях за устройствами NAT, что противоречит изначальной концепции однородного ин-

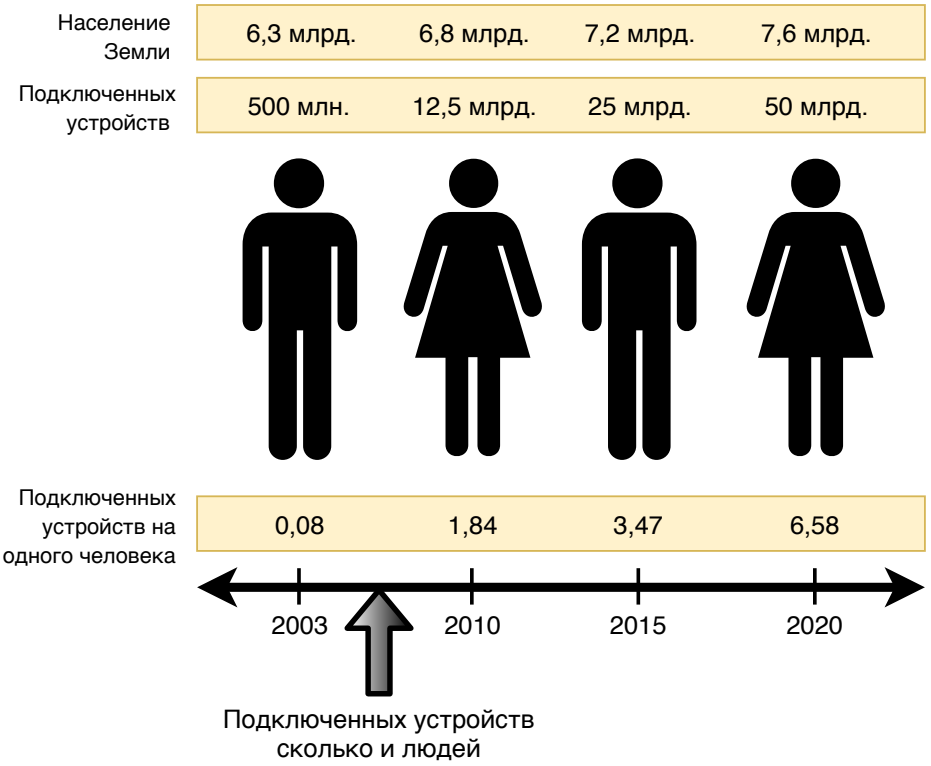

Рисунок 1.1 — Временная шкала изменения количества людей и предметов, подключенных к интернету

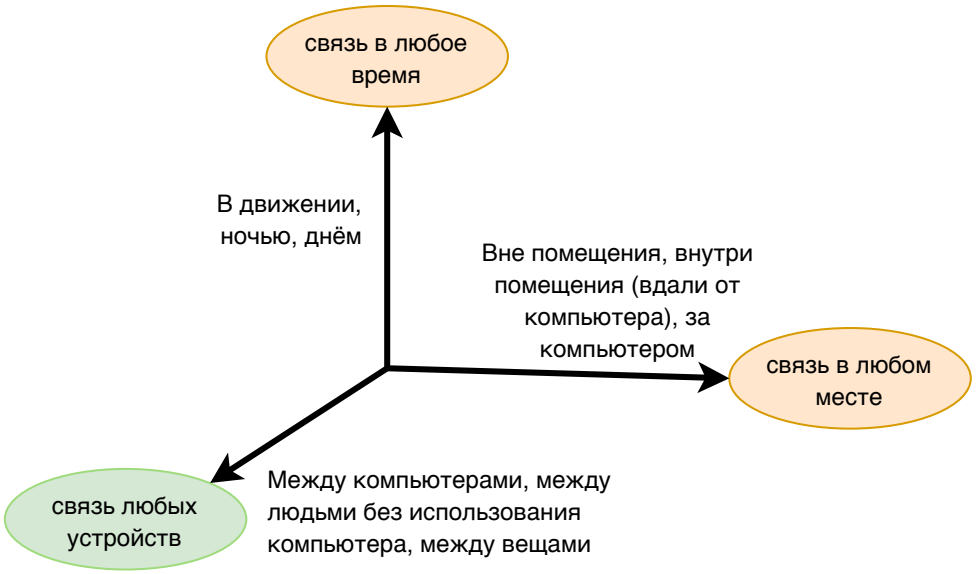

Рисунок 1.2 — Новое направление коммуникаций, реализуемой Интернетом вещей

тернета. Решением может стать повсеместное использование протокола IPv6 в качестве протокола сетевого уровня. Хотя внедрение этого протокола и затянулось, но уже к декабрю 2018 года ожидается, что 25% всех Интернет доменов будет доступно через этот протокол [5].;

— огромное количество одновременно подключенных устройств к сети, которыми необходимо управлять, обмениваться данными. Произойдёт существенное увеличение долей обмена данными, инициированными устройствами, по сравнению с долей информационного обмена, инициированного людьми;

— динамические изменения структуры сети. Устройства будут свободно подключаться к сетям, менять своё местоположение, отключаться от сети и подключаться к новым устройствам. Подразумевается, что количество устройств в одной сети - переменная величина с течением времени;

Важной частью рекомендации от МСЭ-Т являются требования предъявляемые к устройствам IoT. Любые технологии LPWAN, в том числе и LoRa™, должны соответствовать этим требованиям для предоставления возможности их включения в инфраструктуру Интернета вещей:

— предоставление автономных услуг: требуется, чтобы услуги могли предоставляться с помощью автоматической передачи, обработки и сбора данных вещей, основанных на правилах, задаваемых операторами или абонентами. Услуги могут зависеть от методов автоматизированной обработки и интеллектуального анализа данных;

— соединение на основе идентификатора: соединение с любой вещью в концепции Интернета вещей будет происходить на основе уникального идентификатора, которым обладает тот или иной объект. Отсюда выходит требование о создании универсального идентификатора (например адрес IPv6) для применения в гетерогенных сетях.

— функциональная совместимость: требуется обеспечение функциональной совместимости гетерогенных и распределенных систем в целях предоставления и потребления самый разных видов услуг;

— возможности определения местоположения: требуется, чтобы в Интернете вещей обеспечивались услуги, на основе информации о местоположении объекта. Требуется, чтобы информация о местоположении вещей отслеживалась автоматически. Связь и услуги на основе местоположения могут быть ограничены законами и нормативными актами и должны соответствовать требованиям безопасности;

— безопасность в Интернете вещей: каждая вещь имеет соединение с сетью, что приводит к серьёзной угрозе безопасности, таким как угроза аутентичности,

целостности и конфиденциальности как данных, так и услуг. Одним и важнейших требований к безопасности является необходимость объединения различных методов и принципов обеспечения безопасности множества устройств и сетей пользователей;

— защита неприкосновенности частной жизни: требуется, чтобы в IoT обеспечивалась неприкосновенность частной жизни. У многих вещей есть владельцы и эти вещи могут хранить личную информацию их владельцев. Необходимо обеспечить неприкосновенность частной жизни человека при сборе, обработке, анализе и передачи больших массивов информации вещами. Защита неприкосновенности частной жизни не должна служить препятствием для аутентификации источника данных;

— автоматическое конфигурирование: необходимо обеспечить возможность автоматического конфигурирования устройств, для возможности оперативной модификации программного обеспечения вещей, с целю повысить качество обслуживания клиентов, а также степень интеграции устройства с окружающим миром и сетью, не нарушая при этом, требования о безопасности и конфиденциальности.

— управляемость: возможность вмешательства человека в работу вещей при необходимости.

1.1.3 Эталонная модель Интернета вещей

Также была разработана эталонная модель интернета вещей [6], она показана на рисунке 1.4. Она включает в себя четыре уровня, а также возможности обеспечения безопасности и управления, которые связаны с этими четырьмя уровнями:

— уровень приложения;

— уровень поддержки услуг и поддержки приложения;

— уровень сети;

— уровень устройства.

1.1.3.1 Уровень приложения

Содержит само приложение IoT.

1.1.3.2 Уровень поддержки услуг и поддержки приложений

Данный уровень состоит из следующих двух групп возможностей:

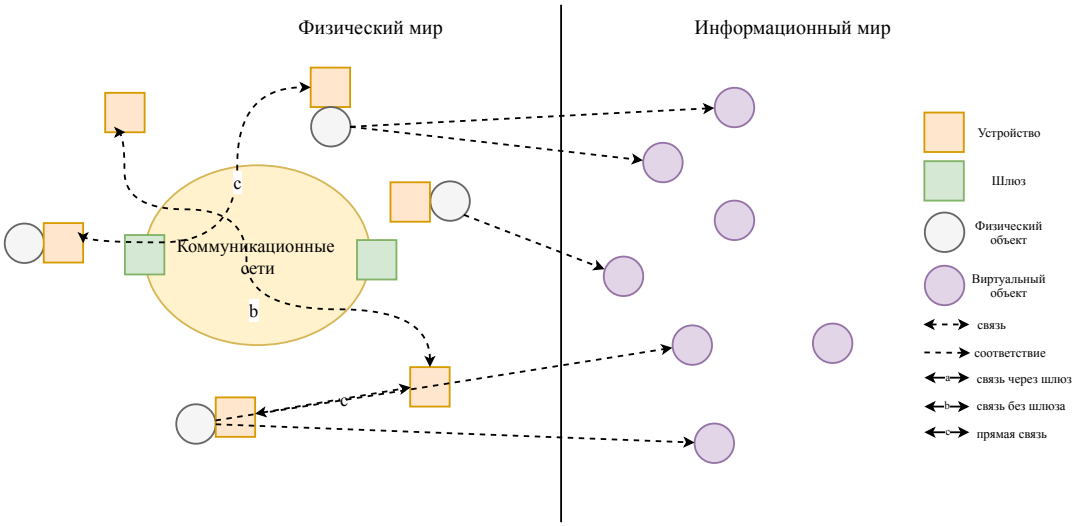

Рисунок 1.3 — Отображение физических и виртуальных вещей

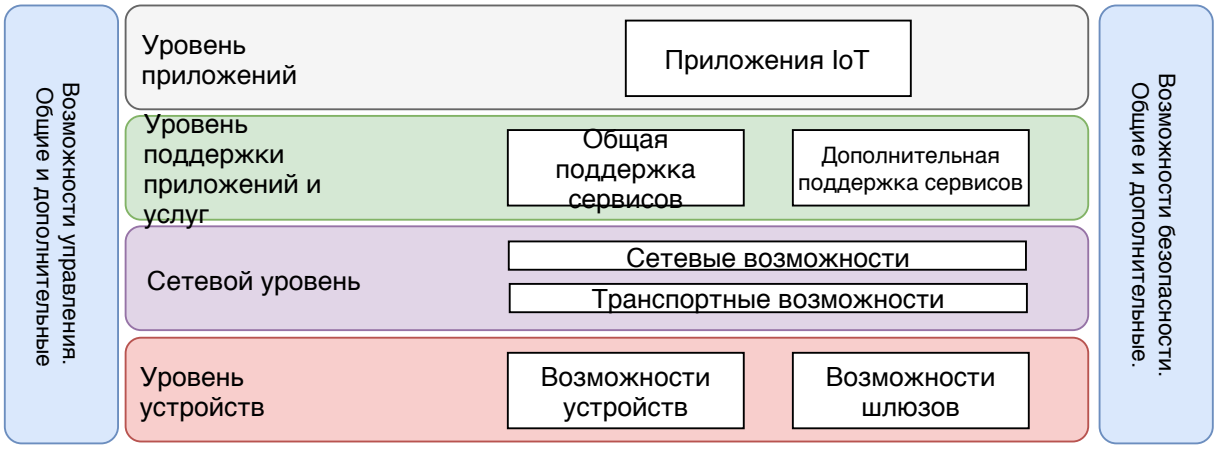

Рисунок 1.4 — Эталонная модель IoT

— общие возможности поддержки, или типовые возможности, которые могут использоваться приложениями Интернета вещей такими, как хранение или обработка данных.

— специализированные возможности поддержки или набор конкретных возможностей, предназначенных для удовлетворения требований разнообразных приложений.

1.1.3.3 Уровень сети

Существует два типа возможностей:

— возможности организации сетей: предоставляет функции управления сетевыми соединениями;

— возможности транспортировки: предназначены для предоставления соединений для транспортировки информации в виде данных, относящихся к услугам и приложениям IoT, а также транспортировки информации управления и контроля, относящейся к IoT.

1.1.3.4 Уровень устройства

Этот уровень можно логически разделить на два вида возможностей:

— возможности устройства. Это могут быть такие возможности, как: спящий режим и пробуждение, организация специальных сетей, прямое и непрямое взаимодействие устройства с сетью;

— возможности шлюза. Это возможность поддержки различных интерфейсов. Шлюза объединяют в себе различные сетевые интерфейсы, как проводные, так и беспроводные.

1.1.3.5 Возможности управления

Возможности управления IoT охватывают традиционные классы конфигурации, учета, безопасности и т.д.

Важнейшие возможности управления включают:

— управление устройствами, диагностика, обновление, прошивка, управление рабочим состоянием устройства;

— управление топологией локальной сети;

— управление трафиком и перегрузками.

1.1.3.6 Возможности обеспечения безопасности

Есть два вида возможностей обеспечения безопасности: общие и специализированные. Общие возможности не зависят от приложений и включают:

— на уровне приложений: авторизацию, конфиденциальность, аутентификацию, целостность данных приложения, защиту неприкосновенности частной жизни, аудит безопасности;

— на уровне сети: авторизацию, аутентификацию, защиту конфиденциальности и целостности;

— на уровне устройства: аутентификацию, авторизацию, проверку целостности устройства, управление доступом, защиту целостности и конфиденциальности.

Специализированные возможности зависят от вида приложений и могут налагать дополнительные специфичные требования по безопасности.

1.2 Обзор технологии LoRa™

LoRa™ представляет собой технологию энергоэффективной сети дальнего радиуса действия, разрабатываемый организацией LoRa™ Alliance. Данная технология нацелена на использование в устройствах с автономными источниками питания, где показатель энергопотребления является наиболее важным. В данном разделе будет дан обзор на данную технологию. Будут кратко рассмотрены физический и уровень управления доступом к сети (MAC) LoRaWAN™ сетей.

1.2.1 Стек протоколов LoRa™

На рисунке 1.5 изображён стек протоколов в сетях LoRaWAN™.

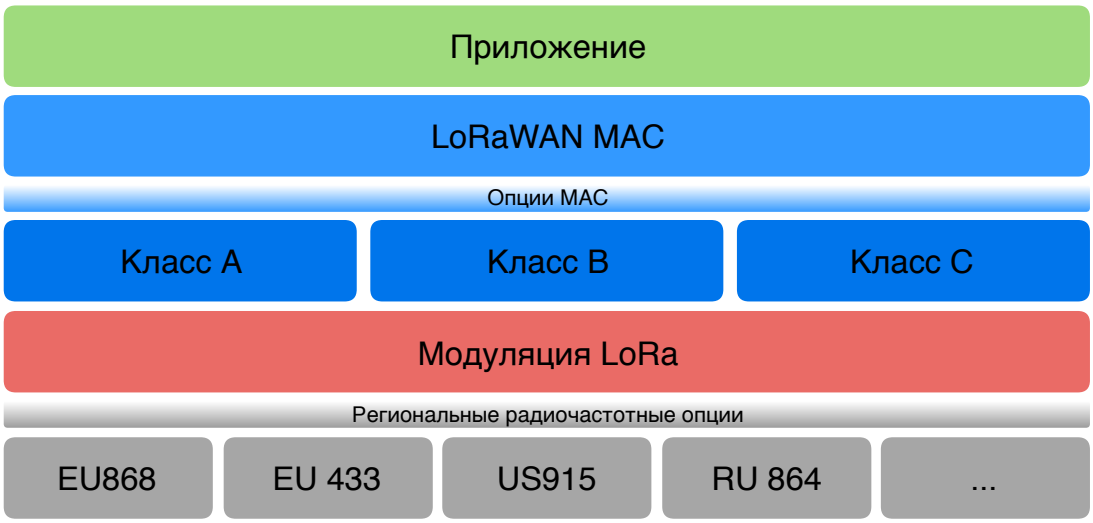

Рисунок 1.5 — Стек протоколов LoRaWAN™

Далее будут кратко рассмотрены все уровни данного стека протоколов.

1.2.2 Сетевая архитектура LoRa™

Стандартной топологией LoRa™ является “звезда из звёзд”, которая включает в себя различные типы устройств, как показано на рисунке 1.6.

1.2.3 Физический уровень

Технология LoRa™ описывает два независимых уровня протоколов: физический, с использование линейной частотной модуляцией (CSS); и протокол контроля доступа к среде (LoRaWAN™), хотя системы коммуникации LoRa™ также реализуют специфичные сетевые архитектуры [7].

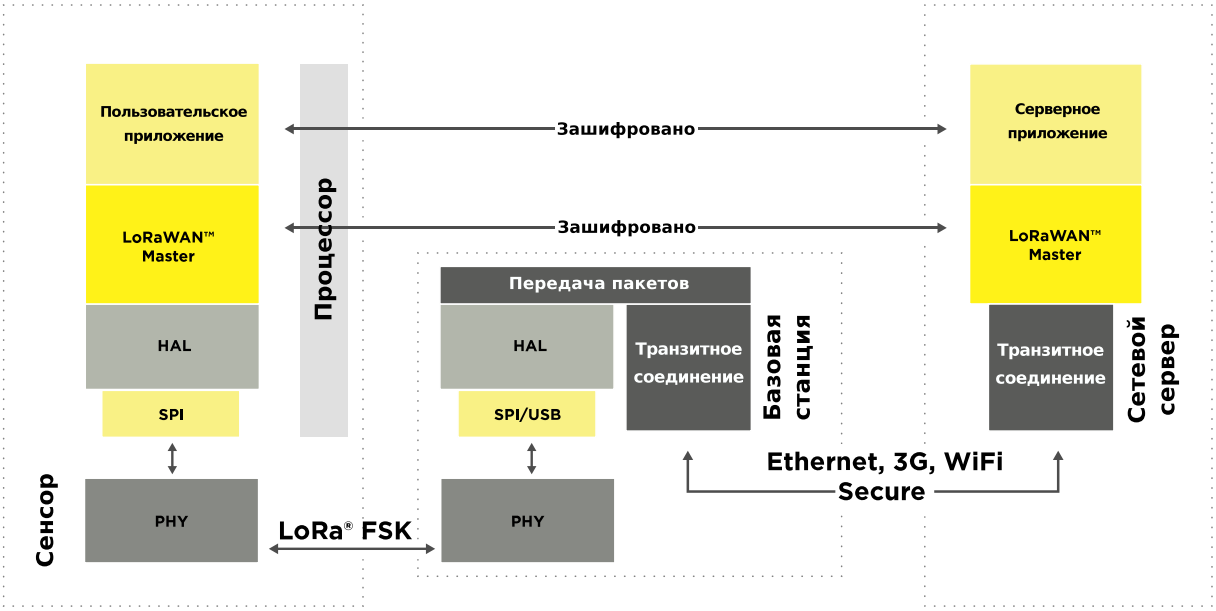

Рисунок 1.6 — Сетевая архитектура LoRa™

Физический уровень LoRa™ разработан компанией Semtech и он обеспечивает связь с низким энергопотреблением, низкой скоростью и большим радиусом действия. Размер полезных данных может изменятся в диапазоне от 2 до 255 октетов, а скорость передачи данных может достигать до 50 Кбит/с. Технология модуляции закрыта и является собственностью компании Semtech, поэтому здесь будут рассмотрены только известные принципы работы передатчиков и приёмников LoRa™.

LoRa™ использует линейную частотную модуляцию, которая использует линейное изменение частоты несущей во время передачи закодированной информации. Благодаря линейному возрастанию частоты сигнала смещение частоты между приёмником и передатчиком за промежуток времени передачи символа остаётся постоянным, что легко устраняется демодулятором [7]. Это делает данную модуляцию невосприимчивой к эффекту Доплера. Смещение частот между приёмником и передатчиком может достигать 20% ширины полосы частот без влияния на корректность демодуляции. Это помогает уменьшить стоимость приёмников и передатчиков LoRa™, поскольку смягчены требования на точность встроенных кварцевых резонаторов. Все эти особенности делают возможным для приёмников LoRa™ приём сигнала мощностью до -130 дБм.

1.2.3.1 Параметры физического уровня

Есть несколько параметров для настройки модуляции LoRa™:

а) полоса пропускания (BW);

б) коэффициент расширения спектра (SF);

в) кодовая скорость (CR).

Также могут быть изменены следующие настройки радиомодулей:

— длина преамбулы, значение синхронизирующего слова SyncWord;

— посылать ли явно заголовок с сообщением, он содержит информацию о параметрах приёма остальной части сообщения (длина полезных данных, параметр CR и наличие CRC);

— наличие поля CRC;

— бит LowDataRateOptimization (DE).

1.2.3.2 Модуляция и кодирование

Радиосигнал с линейной частотной модуляцией (ЛЧМ)

Физический радиоинтерфейс LoRa™ использует широкополосные сигналы с линейной частотной модуляцией [7]. ЛЧМ — давно известная технология, применявшаяся в радиолокации, но в качестве основы кодирования цифровых данных использовалась реже. При линейной модуляции частота сигнала испытывает линейную девиацию (возрастает или уменьшается) со временем. Частота изменяется в пределах ширины канала частот (с англ. *Bandwith, BW*), таким образом, она изменяется по закону:

$$\omega(t) = \omega_0 + \mu t \tag{1.1}$$

Здесь ω_0 - несущая частота; μ - параметр с размерностью $с^{-2}$, равный скорости изменения частоты во времени.

За время, равное длительности импульса, девиация частоты равна

$$\Delta\omega = \mu\tau_и, \tag{1.2}$$

где $\tau_и$ — длительность сигнала, а полная фаза сигнала:

$$\psi(t) = \omega_0 + \mu t^2/2 \tag{1.3}$$

Сигнал ЛЧМ представляется следующей математической моделью [8]:

$$u_{ЛЧМ}(t) = \begin{cases} 0, & t < -\tau_и/2, \\ U_m cos(\omega_0 t + \mu t^2/2), & -\tau_и/2 \le t \le \tau_и/2, \\ 0, & t > \tau_и/2. \end{cases} \tag{1.4}$$

Анализ характера частотной зависимости модуля и фазы спектральной плотности прямоугольного ЛЧМ-импульса выявил [8] полную зависимость от безразмерного числа:

$$B = \Delta f \tau_{\text{и}} = \mu \tau_{\text{и}}^2 / (2\pi), \tag{1.5}$$

равного произведению девиации частоты на длительность импульса, называемого *базой* ЛЧМ-сигнала.

На практике обычно стараются выполнить условие $B \gg 1$. Спектр таких ЛЧМ-сигналов имеет ряд особенностей, и одной из них является то, что модуль спектральной плотности практически постоянен в пределах полосы частот шириной $\Delta\omega$ с центром в точке ω_0. В LoRa™ этим параметром косвенно манипулируют за счёт увеличения коэффициента расширения спектра (*Spreading Factor*, *SF*). SF — это логарифмический параметр, соответствующий продолжительности передачи одного символа:

$$B = \Delta f \tau_{\text{и}} = 2^{SF} \tag{1.6}$$

Значение SF может варьироваться от 6 до 12.

LoRa™ кодирует символы циклическим сдвигом ЛЧМ-сигнала относительно кадра времени. Скачок фазы и обозначает кодируемый символ. Поскольку 2^{SF} ЛЧМ-сигнала могут находится в символе, то и один символ может эффективно кодировать SF бит информации.

Пропускная способность ЛЧМ-сигналов зависит только от ширины полосы частот. Увеличение SF повлечет за собой деление продолжительности ЛЧМ-сигнала на два (поскольку 2^{SF} ЛЧМ-сигналов покрывают всю ширину полосы частот) и увеличением в два раза продолжительности передачи символа. Пропускная способность не уменьшиться в два раза, поскольку теперь каждый символ кодирует на один бит больше.

Также LoRa™ реализован механизм прямой коррекции ошибок (*Forward Error Correction*, *FEC*). Параметр CR может быть равен $4/(4+n)$, где $n \in \{1, 2, 3, 4\}$. Принимая это во внимание, можно вычислить полезную пропускную способность по формуле 1.7.

$$R_b = SF \frac{\Delta f}{2^{SF}} CR \tag{1.7}$$

Для примера, если имеем Δf (Он же BW) = 125 кГц, SF = 8, CR = 4/8, то получаем полезную пропускную способность равную 1,95 Кбит/с. Все вышеприведенные особенности позволяют добиться высокой помехозащищённости и, как следствие, большой зоны покрытия сети.

Пример сигнала, принятым анализатором спектра от передатчика LoRa™, изображен на рисунке 1.7.

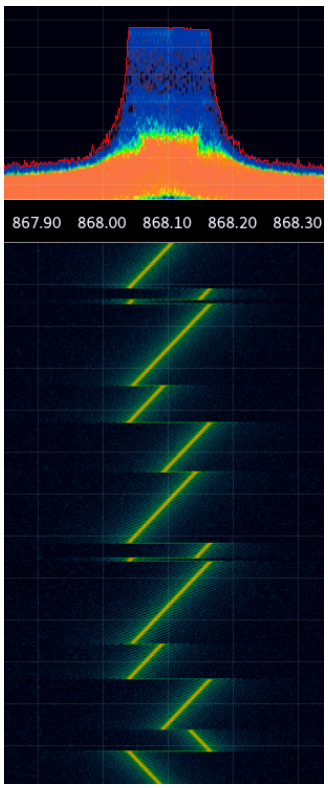

Рисунок 1.7 — Сигнал LoRa™ (внизу) и его спектр (вверху) [9]

1.2.3.3 Формат кадра физического уровня

Хотя модуляция LoRa™ позволяет отправлять произвольные кадры, формат кадра физического уровня был описан и реализован в передатчика и приёмниках от Semtech. Ширина полосы частот и коэффициент расширения спектра не изменяются в рамках кадра.

Кадр LoRa™ начинается с преамбулы (см. рис. 1.8): последовательности растущих ЛЧМ-сигналов, занимающих всю частотную полосу. Два последних ЛЧМ-сигнала кодируют синхронизирующее слово. Синхронизирующее слово это одно-байтовое значение, которое используется для опознания двух разных сетей, использующих один и тот же канал для связи. Устройство, сконфигурированное на приём заданного слова синхронизации остановит приём данных если принятое слово синхронизации не совпало с заданным. Слово синхронизации следует за двумя или четырьмя спадающими ЛЧМ-сигналами в преамбуле. Длина преамбулы может быть изменено между 10,25 и 65539,25 символами.

Когда после преамбулы задаётся необязательный заголовок, он кодируется и передаётся с CR = 4/8. Он указывает на размер полезной нагрузки (в байтах), CR используемое для кодирования и наличие необязательного 16-битного поля CRC. CRC (если оно есть) находится в конце кадра, сразу после полезной нагруз-

ки. Поле размера полезной нагрузки имеет размер в один байт, что ограничивает максимальный размер полезной нагрузки в 255 байт.

Рисунок 1.8 — Структура кадра физического уровня LoRa™

кГц.

1.2.3.4 Доступные частотные диапазоны

Физический уровень LoRa™ работает в различных субгигагерцовых частотных ISM диапазонах. В различных странах приняты различные диапазоны частот, поэтому устройствам с LoRa™ необходимо самостоятельно выбирать используемый радиодиапазон для физического уровня, в зависимости от принятых местных соглашениях. Таблица 1.1 описывает, принятые в различных странах, частотные диапазоны для строительства сетей связи.

Таблица 1.1 — частотные диапазоны, принятые в разных странах

Частотный диапазон	Страна
EU 863-870 МГц ISM	Европейский союз
EU 433 МГц ISM	Европейский союз
RU 864-870 МГц ISM	Российская Федерация
US 902-928 МГц ISM	США
CN 779-787 МГц ISM	КНР

В 2017 году был утвержден частотный план LoRa™ Alliance, в котором определены единые региональные параметры LoRaWAN™, используемые на территории РФ. Согласно этому плану, для РФ выделяется частотный диапазон шириной в 6 МГц, с максимальной шириной полосы частот в 125 кГц. При этом для конечных устройств **обязательна** поддержка двух каналов с несущими 868,9 и 869,1 МГц/ DR0 до DR5 (см. таблицу 1.2).

Радиопомехи

Поскольку в РФ LoRa™ использует нелицензируемый диапазон частот, то строящиеся сети будут работать в условиях внешних помех, создаваемыми прочи-

Таблица 1.2 — Стандартные каналы частотного диапазона RU864-870

Модуляция	Ширина полосы частот [кГц]	Несущая [МГц]	Пропускная способность FSK или LoRa™ DR	Кол-во каналов	Рабочий цикл
LoRa™	125	868,9 869,1	DR0 до DR5 / 0,3-5 Кбит/с	2	<1%

ми пользователями диапазона, включая коммерческие и частные сети Интернета вещей, работающих по технологиям LoRa™, NB-Fi, “СТРИЖ Телематика” и пр.

1.2.4 Протокол LoRaWAN™

LoRaWAN™ — это MAC протокол, созданный, преимущественно для сетей сенсоров [7, 10], которые обмениваются пакетами с сервером на небольшой скорости и на относительно больших интервалах времени (одна передача в час или даже в день). На данном уровне обеспечиваются:

— адаптация скорости передачи данных;

— шифрование полезной нагрузки на уровне сети, передаваемой между конечным устройством и приложением;

— управление выделением окон для нисходящего канала связи;

1.2.4.1 Компоненты сети LoRaWAN™

В спецификации LoRaWAN™ определены несколько компонентов для создания сети: конечные устройства (*end-devices*), шлюзы (или базовые станции) и сетевой сервер (*network server*).

— конечное устройство представляет собой, как правило, сенсор с небольшим энергопотреблением, которое обменивается данными с базовой станцией, использую LoRa™;

— шлюз: промежуточное устройства, перенаправляющее пакеты, приходящие от конечных устройств на сетевой сервер, который имеет соединение с сетью Интернет. В сети могут находится несколько шлюзов и один и тот же пакет с данными может быть получен (и перенаправлен) несколькими шлюзами одновременно;

— сетевой сервер: ответственен за устранение повторяющихся пакетов и декодирования пакетов, отправленных устройствами и отправки пакетов устройствам.

В отличии от традиционных сотовых сетей, конечные устройства не ассоциированы с конкретными шлюзами с целью получения доступа к сети. Шлюзы только предоставляют услуги транспорта пакетов от конечных устройств до сетевой сервер включают в пакет информацию о качестве связи. Таким образом, конечные устройства соединены с узсетевой сервер который ответственен за обнаружение дублирующихся пакетов, выбора подходящего шлюза для отправки ответа и т.д. Логически, шлюзы прозрачны для конечных устройств [7].

LoRaWAN™ определяет три класса конечных устройств для удовлетворения нужд различных приложений:

— Класс A, полудуплекс: устройства могут планировать передачу данных по восходящему каналу связи (UL) в соответствии со своими нуждами. Этот класс устройств получают пакеты только после отправки своего пакета в сеть. После отправки открываются два небольших окна приёма. Данные по нисходящему каналу (DL) должны поступить точно во время открытия окон приёма. Эти устройства имеют наименьший уровень энергопотребления, но предоставляют меньше гибкости для передачи им данных.

— Класс B, полудуплекс со запланированными слотами приёма: данный класс устройств открывают дополнительные окна приёма данных в назначенное время. Для временной синхронизации им требуется маячковый сигнал от шлюзов поблизости. С такой синхронизации сетевой сервер знает когда конечное устройство находится в состоянии ожидания приёма данных.

— Класс C, полудуплекс с постоянным прослушиванием канала: устройства данного класса имеют самое протяженное окно приёма и, соответственно, наибольшее среди остальных потребление энергии.

Следует отметить, что LoRaWAN™ не допускает коммуникации между конечными устройствами: пакеты только могут быть отправлены от конечного устройства на сетевой сервер или наоборот. Передача данных от одного устройства на другое, если она потребуется, может осуществляться только через сетевой сервер (и, соответственно, через все промежуточные шлюзы).

1.2.4.2 Формат сообщения LoRaWAN™

LoRaWAN™ использует на физическом уровне формат кадра, описанный в разделе 1.2.3.3. Заголовок и CRC в сообщениях восходящего канала являются обязательными, что делает невозможным использование SF равным шести с сетях LoRaWAN™. Сообщения нисходящего трафика содержат заголовки, но не имеют поля CRC.

Формат сообщения подробно описан на рисунке 1.9.

а) *MHDR* — заголовок пакет уровня MAC. Содержит:

1) поле *Major* (2 бита) — определяет major часть версии формата сообщений процедуры активации по воздуху (OTA - over-the-air);

2) поле *MType*, определяющее тип сообщения (3 бита). Существует 6 типов сообщений (см. таблицу 1.3);

б) *MACPayload* — фрейм данных. Данный фрейм состоит из следующих подполей:

1) *FHDR* — заголовок фрейма. Он включает в себя:

— *DevAddr* — адрес устройства;

— *FCtrl* — октет управляющей информацией фрейма. Состоит из:

— *ADR* (1 бит) — флаг режима адаптации скорости;

— *ADRAckReq* (1 бит) — флаг, устанавливающийся только в режиме адаптации скорости, указывает на запрос устройством подтверждения факта получения сообщений от данного устройства;

— *FPending* (1 бит, только DL канал) — флаг, обозначающий наличие запроса со стороны сети на передачу устройству дополнительных данных сверх объема ограничения на максимальный размер кадра;

— *CLASS-B* (1 бит, только UL канал) — флаг, обозначающий что конечное устройство переключилось в режим класса B;

— *FOptLen* (4 бита) — размер полня опций FOpt заголовка MAC уровня;

— *FCnt* (16 бит) — номер фрейма. После процедуры активации по воздуху (OTA), конечное устройство и сетевой сервер инициализируют два счётчика — счетчик количества принятых фреймов и количества переданных фреймов. При получении каждого нового сообщения принимающая сторона сравнивает значение поля *FCnt* со значением внутреннего счётчика принятых фреймов. Если разница превышает MAX_FCNT_GAP, принимается решение о большом количестве потерянных фреймов;

— *FOpt* (0..120 бит) — опциональные данные (до 15 октетов). Используется для передачи команд MAC. Команды MAC могут отправляться в поле *FOpt* (и тогда $FOptLen > 0$ и $FPort > 0$), так и в поле полезной нагрузки *FRMPayload* (тогда $FOptLen = 0$ и $FPort = 0$);

2) *FPort* — номер порта фрейма.

— если оно равно 0, это значит что полезная нагрузка содержит MAC команду. В этом случае поле FOptLen должно быть равно 0;

— значения от 1 до 223 определяются приложением для своих нужд (*application specific*);

— значения 224-225 зарезервированы.

3) *FRMPayload* — полезная нагрузка. Содержит пользовательские данные, которые передаются между целевым приложением и конечным устройством. Содержимое этого поля шифруется по стандарту AES либо на сетевом уровне (с использованием 128-битного ключа *NwkSKey*), либо на уровне приложения (128-битным ключом *AppSKey*).

в) *MIC* (*Message Integrity Code*) — код контроля целостности сообщения. Вычисляется алгоритмом AES128 с ключом *NwkSKey* по всем полям сообщения.

Таблица 1.3 — Допустимые значения поля MType

MType	Описание
000	Запрос процедуры активации по воздуху (OTA) — join request
001	Подтверждение процедуры активации по воздуху (OTA) — join accept
010	Передача данных “вверх” без подтверждения (unconfirmed data up)
011	Передача данных “вниз” без подтверждения (unconfirmed data down
100	Передача данных “вверх” с подтверждением (confirmed data up)
101	Передача данных “вниз” с подтверждением (confirmed data down)
110	RFU
111	Для пользовательских решений

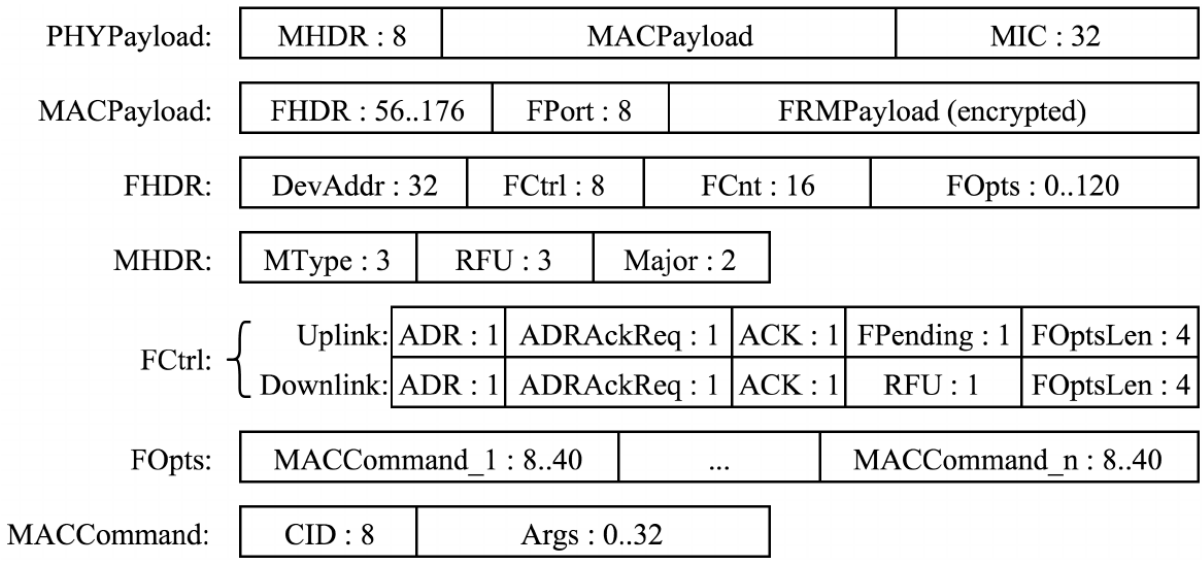

Рисунок 1.9 — Формат кадра LoRaWAN™. Размеры полей указаны в битах [7]

1.2.4.3 Возможные расширения протокола LoRaWAN™

Промышленная адаптация

Для работы протокола LoRaWAN™ в концепции индустрии 4.0 исследователи Mattia Rizzi, Paolo Ferrari и др. в своей работе [11] предложили адаптировать протокол LoRaWAN™, добавив к нему поддержку быстрого переключения каналов на каждый временной слот(*Time Slotted Channel Hopping, TSCH*). Исследователями был проведён анализ, в результате которого удалось выяснить, что, используя планирование параметрами физического уровня LoRa™, возможно достичь безошибочного опроса конечных устройств интенсивностью до 6000 раз в минуту. Подобная адаптация протокола LoRaWAN™ позволит составить ему конкуренцию с беспроводными технологиями на базе HART (*Highway Addressable Remote Transducer Protocol*).

Адаптация к работе в сетях IPv6

Поскольку часть протоколов LPWAN и LoRaWAN™, в частности, не предполагают прямого использования в собственном стеке протокола IP, то процесс интеграции систем в инфраструктуру Интернета вещей с использованием LPWAN сетей становится более сложным, чем должен быть. Нельзя не упомянуть также о дефиците адресов протокола IPv4, что вновь делает актуальным вопрос о массовой интеграции устройств, поддерживающих протокол IPv6.

В исследовании Patrick Weber, Axel Sikora и др. было предложено решение по адаптации LoRaWAN™ к работе в IPv6-сетях, добавлением поддержки протокола IPv6 в стек протоколов LoRaWAN™ [12], подобно тому как было предложена адаптация IPv6 для стандарта IEEE 802.15.4. Данное решение было названо 6LoRaWAN, а также было реализовано и протестировано авторами работы.

Особенностью данной адаптации является то, что помимо топологии "звезда из звёзд", будет возможно применение других топологий на базе LoRaWAN™. Устройства, поддерживающие IPv6, смогут обмениваться сообщениями с любыми устройствами в Интернете, что делает их интеграцию в инфраструктуру Интернета вещей более лёгкой.

Данная адаптация также поддерживает обратную совместимость (на уровне шлюзов и конечных устройств) с конечными устройствами не поддерживающих расширение IPv6. Более того, подразумевается поддержка 6LoRaWAN существующими маршрутизаторами, которые могут образовать сети с любыми конечными IP-устройствами (не только на базе LoRaWAN™).

В таблице 1.4 представлена адаптация 6LoRaWAN как изменение стека LoRaWAN на сетевом, транспортном уровне и на уровне приложения по модели OSI/RM.

Таблица 1.4 — Классификация протоколов по модели OSI/RM

	LoRaWAN™	6LoRaWAN
Уровень приложений	пользовательское приложение	например COAP
Уровень представления		
Сеансовый уровень		
Транспортный уровень		например UDP
Сетевой уровень		IPv6
		6LoRaWAN адаптация
Канальный уровень	LoRaMAC	LoRaMAC
Физический уровень	LoRaPHY	LoRaPHY

1.3 Примеры применения технологии LoRa™ в рамках концепции Интернета вещей

Существует большое множество областей, где применима концепция Интернета вещей, а значит имеет место быть применение технологиям LPWAN и LoRaWAN™ в частности. Перечислим некоторые возможные приложения энергоэффективной технологии беспроводной передачи данных дальнего радиуса действия:

— “умное” городское и сельское освещение;

— полевые испытания со съёмом показании в режиме реального времени;

— транспорт: управление, мониторинг, логистика;

— “умный” дом, оснащённый различными сенсорами;

— сельское хозяйство: автономные комбайны, сенсоры и т.п;

— управление энергией;

— здравоохранение: легковесные переносные датчики слежения жизненных показаний пациента для слежения в реальном времени;

— промышленность и производство;

— измерение и учёт потребления электроэнергии в системах АСКУЭ, а также расхода воды;

— слежение за местоположением в реальном времени. Например для отслеживания перемещения животных;

— системы безопасности;

1.3.1 Умный светильник

Умные светильники — это то направление, которое уже активно развивается в современной индустрии как за рубежом, так и в России. В Петрозаводске уже сейчас существует как минимум одна компания, специализирующаяся на производстве умных светильников, а также всей необходимой сетевой инфраструктурой к ним. Речь идёт о компании ООО ПК “Энергосбережение”, которая с 2012 года разрабатывает и производит светодиодное осветительное оборудование — уличные и промышленные светильники. В 2017 году произошёл качественный скачок в системе мониторинга и управления. В светильники начали интегрировать модули (см. рис. 1.10) с поддержкой LPWAN сетей, а именно LoRaWAN™, и, тем самым,

смогли дать дополнительный толчок развитию инфраструктуре Интернета вещей на Северо-Западе РФ.

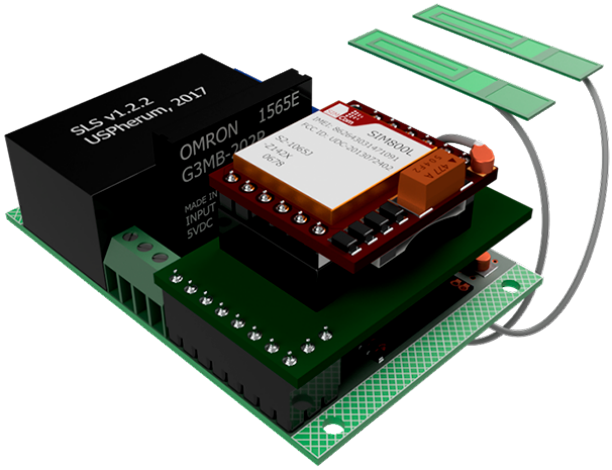

Рисунок 1.10 — Модуль управления светильником с поддержкой сети LoRa™, разработанный в компании ООО ПК “Энергосбережение” [13]

На рисунке 1.11 отображен используемый в компании принцип управления “умными” светильниками. Как можно видеть, не только конечные устройства управляют светильниками, но и базовые станции наделены такими функциями, что позволяет гибко разместить сеть LoRa™ на существующих опорах уличного освещения без выделения отдельных от опор мест под базовые станции. Сами светильники могут включаться, выключаться и диммироваться (изменять освещённость) посредством чат-ботов. Чат-бот — это приложение, построенное на базе существующих мессенджеров и не требующих от пользователей дополнительной установки сторонних приложений для оказания услуг. Всё, что необходимо сделать пользователю для управления имеющимися у него светильниками, это:

а) установить мессенджер, который поддерживает чат-ботов (например Telegram или Viber);

б) найти чат компании, обслуживающей инфраструктуру умных светильников

в) пройти быструю авторизацию (этот пункт может быть опущен, если он авторизовался ранее);

г) выбрать светильник или группу светильников в чате;

д) нажать кнопку “включить/выключить” или “диммирование”;

Команда будет отправлена на соответствующий светильник или на группу светильников.

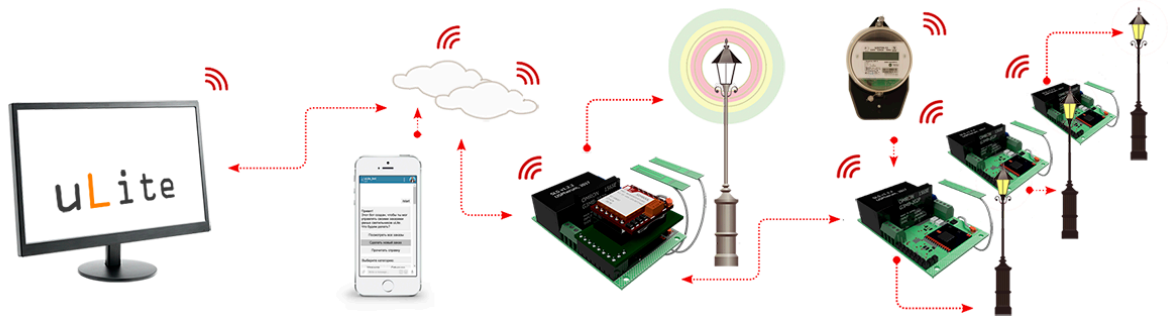

Рисунок 1.11 — Схема управления “умными” светильниками [13]

1.3.2 Умный счётчик

На современном рынке представлено большое разнообразие счётчиков электрической энергии, имеющие различные интерфейсы для снятия показании внешними устройствами. Наиболее популярные интерфейсы — импульсный выход, RS-232 и RS-485. Пример такого счётчика приведён на рисунке 1.12

Любое вычислительное устройство (микроконтроллер, ПЛК, FGPA) с поддержкой вышеприведенных интерфейсов и с поддержкой сетей LoRaWAN™ может быть подключено к счётчику, а следом, и к системе автоматизированного сбора и учета электроэнергии (АСКУЭ). Всё вышесказанное справедливо не только для счётчиков электроэнергии, но и для измерения потребления воды, газа.

Всё это также входит в концепцию Интернета вещей и даже в концепцию индустрии 4.0.

1.4 Преимущества и недостатки в сравнении с другими технологиями

На рынке беспроводных технологии представлены десятки различных технологии, удовлетворяющих требованиям LPWAN сетей. Это и не удивительно, поскольку такое большое разнообразие объясняется текущим уровнем развития инфраструктуры Интернета вещей: единые стандарты установлены не были и на рынке наблюдается ожесточённая конкуренция за право стать де-факто стандартом, как это было например с USB и FireWire.

Забегая вперёд, стоит отметить, что все перечисленные технологии различаются достаточно существенно и их можно выделить в две группы: широкополосные и узкополосные. К широкополосной (*Ultra Wide Band*) относят LoRa™. К узкополосным (*Ultra Narrow Band*) относят Sigfox, “Стриж”, NB-Fi и др. Рассмотрим другие технологии энергоэффективной передачи данных на большие расстояния, претендующие на использование в Интернете вещей.

Рисунок 1.12 — Счётчик Меркурий 201.5 с импульсным выходом

1.4.1 Sigfox

Sigfox — французская компания, основанная в 2009 году и специализирующаяся на создании сетей с низким энергопотреблением, которым необходимо продолжительное время передавать небольшие по объёму данные.

Sigfox использует проприетарную технологию беспроводной связи, работающую в ISM диапазоне частот: 868 МГц в Европе и 902 МГц в США. Их беспроводная технология использует узкополосные каналы шириной в 100 Гц.

В России на нелицензируемый диапазон около 868 МГц в свободном распоряжении можно использовать лишь 500 кГц.

Как факт преймущества UNB-систем над UWB-системами можно отнести количество каналов, которые могут сосуществовать одновременно. При ширине каналов LoRa™ в 125 кГц, нельзя уместить больше трёх каналов на выделенный частотный диапазон. Каналов Sigfox можно выделить около 5 тысяч.

Однако системы на базе UNB крайне чувствительны к установке частоты. Если брать в расчёт очень качественные кварцевые резонаторы, доступные на рынке, то можно получить погрешность в 5 ppm от несущей частоты, что в случае с 868 МГц даёт погрешность в ±4340 Гц! При этом не учитывалась погрешность резонатора с колебанием температуры внешней среды.

Утранять данный недостаток были призваны базовые станции Sigfox, которые могли видеть сигнал в более широком частотном диапазоне, но, к сожалению, реализовать такие же алгоритмы на дешёвых и экономичных конечных устройствах было проблематично, поэтому связь Sigfox долгое время была строго однонаправленной.

Системы с UWB обеспечивают симметричный канал связи, благодаря полосе шириной в сотни килогерц. LoRa™ допускает уход от частоты на 25% от ширины канала, что в диапазоне 868 МГц означает допустимую погрешность кварцевого резонатора в 35 ppm. Этой погрешности хватает для того чтобы конечному устройству без проблем передавать данные в полном диапазоне температур от -40 до +85 °C.

Также UWB-системы не сильно чувствительны к допплеровскому эффекту, что нельзя сказать для UNB-систем. Sigfox может потерять стабильность работы уже на скоростях в районе 5-10 км/ч.

Скорость передачи данных в UNB-системах фиксирована. И фиксирована она шириной полосы частот. У Sigfox максимальная скорость передачи данных равна 100 бит/с. UWB-системы могут работать на адаптивной скорости. Конечные устройства LoRa™ могут перейти в режим ADR (*Adaptive Data Rate*), позволяя работать на скоростях от 30 бит/с до 50 кбит/с. Режим ADR меняет скорость передачи автоматически. Это неизбежно делает UWB-системы более гибкими для применения в разных сферах Интернета вещей.

В Sigfox также есть жёсткое ограничение по объёму пользовательских данных — один пакет не может передать больше 12 байт, а максимальное количество сообщений, которое устройство может передать в сеть, составляет 140, что ставит крест на возможном применении данной технологии во многих приложениях Интернета вещей.

2 Практическая часть

2.1 Постановка задачи

В рамках практической части работы, было выделены следующие задачи:

— изучить средства разработки программного обеспечения для микрокотроллеров семейства ARM;

— адаптировать существующее открытое программное обеспечение к другой аппаратной платформе;

— создать работающие прототипы конечных устройств с трансиверами LoRa™;

2.2 Выбор аппаратной платформы

2.2.1 STM32L476G-Discovery

Для разработки конечных устройств была выбрана отладочная плата STM32L476G-Discovery на базе 32-битного микроконтроллера STM32L476VGT6 с ядром ARM-Cortex M4. Данный микроконтроллер является представителем семейства микроконтроллеров с низким энергопотреблением STM32L4 фирмы ARM. Микроконтроллер имеет:

— 3 устройства I2C;

— 3 устройства SPI;

— поддержка шины CAN;

— SWPMI;

— 2 x SAI;

— 12-битное ЦАП;

— драйвер LCD;

— 128 Кбайт SRAM;

— 1 МБайт Flash;

— Quad-SPI;

— touch sensing;

— USB OTG FS;

— поддержка JTAG отладки;

Удобство данной отладочной платы заключается в том, что вся необходимая вспомогательная периферия уже находится на плате, и подключена ко входам и выходам микроконтроллера. В качестве вспомогательной периферии служат:

— программатор/отладчик ST-LINK/V2-1;

— LCD дисплей;

— кнопка RESET;

— джойстик;

— встроенный амперметр, для измерения тока потребления микроконтроллера в режиме low power;

— USB OTG FS;

— аудио ЦАП;

— MEMS (микрофон, 3-осевой гироскоп, 6-осевой компас);

— Quad-SPI Flash память;

— светодиоды.

Вид на плату сверху отображён на рисунке 2.1.

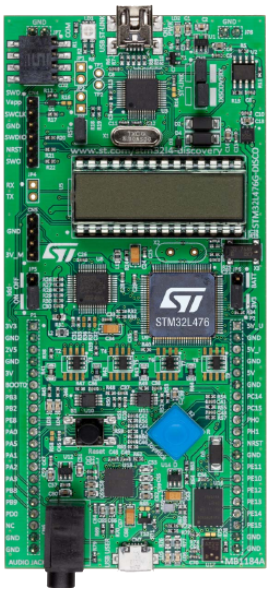

Рисунок 2.1 — Отладочная плата STM32L476G-Discovery

2.2.2 Трансивер LoRa™

Модули Semtech LoRa™ представляют собой ВЧ-трансиверы, с возможностью создания топологии M2M (машина-машина) и “звезда из здвёзд” (сети LoRa™). Эти устройства оптимизируют потребление энергии, увеличивая срок службы ба-

тарей конечных устройств. Они подробно задокументированы, что делает процесс подключения к интерфейсу доступным для широкого круга разработчиков программно-аппаратного обеспечения инфраструктуры Интернета вещей.

Серия SX1272/78 имеют бюджет канала в -148 dBm. Высокая чувствительность в сочетании с усилителем LNA (малошумящий усилитель) +20 дБм обеспечивают надёжную связь для применения в промышленности [11].

Таблица 2.1 — Рабочие частоты для SX1272 и SX1278

Устройство	Минимальная частота (МГц)	Максимальная частота (МГц)
SX1272	860	1020
SX1278	137	525

Чувствительность приёмника определяет минимальное значение мощности, которое требуется ему для демодуляции и декодирования с целью достижения определенной скорости передачи данных. Чувствительность обычно выражается в дБм и чем ниже значение, тем лучшую чувствительность имеет приёмник, поэтому, исходя из таблицы 2.2 можно сделать вывод, что SX1278 имеет большую чувствительность приёмника перед SX1272.

Таблица 2.2 — Основные параметры приёмопередатчиков LoRa™

Приёмо-передатчик	Параметры LoRa™			
	SF диапазон	Ширина полосы частот (КГц)	Эффективная скорость передачи данных (кбит/с)	Чувствительность (дБм)
SX1272	6..12	от 125 до 500	0,24..37,5	-117..-137
SX1278	6..12	от 7,8 до 500	0,018..37,5	-111..-148

Был выбран приёмопередатчик Semtech SX1278. Выбор был обусловлен тем, что он уже имелся в наличии и его внутренняя структура схожа с трансивером SX1272 (они имеют общее руководство по эксплуатации). Данный трансивер будет пригоден для исследовательских работ, но следует иметь в виду что для коммерческого использования в России следует выбрать трансивер SX1272, поскольку частоты около 868 МГц находятся в пределах поддерживаемых частот данного трансивера.

Внутренние регистры трансиверов доступны через интерфейс связи SPI. SPI — это синхронный последовательный полнодуплексный протокол передачи

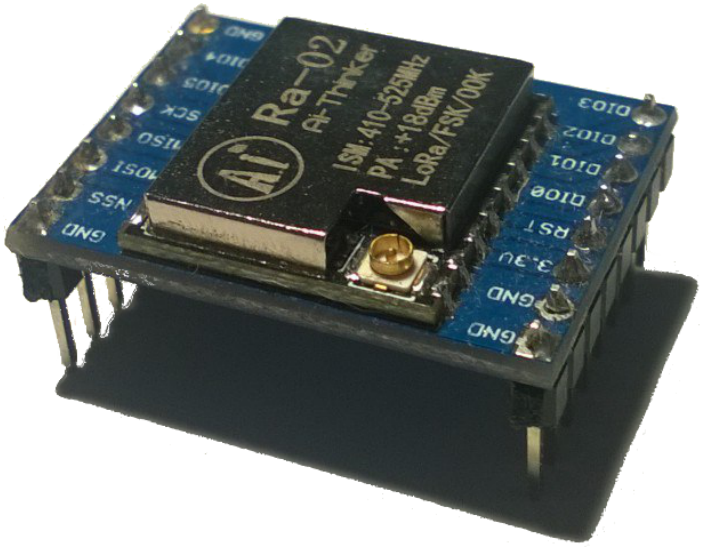

Рисунок 2.2 — Трансивер SX1278

данных. Обмен по протоколу SPI осуществляют ведущее (*Master*) и подчинённое (*Slave*) устройство. Приём и передачу данных инициирует только ведущее устройство.

Этот протокол использует 4 линии для связи, которые описываются как:

— **SCLK**. Соответствует тактовому сигналу, генерируется ведущим и синхронизирует передачу данных;

— **MOSI** (*Master Out Slave In*). Передача основных данных от ведущего к подчиненному устройству;

— **MISO** (*Master In Slave Out*). Передача основных данных от подчиненного к ведущему устройству устройству;

— $\overline{\textbf{NSS}}$ (*Slave Select*). Выборка подчиненного устройства. Используется для связи нескольких подчиненных устройств к ведущему.

Радиомодули LoRa™ работают как подчиненные устройства, а микроконтроллер встраиваемой системы будет ведущим в интерфейсе SPI.

На логическом уровне синхронизации и передачи данных для связи SPI требуется конфигурация полярности тактирующего сигнала (CPOL) и бит фазы синхронизации (CPHA).

Приёмопередатчики LoRa™ SX1272/78 используют параметры CPOL = 0 и CPHA = 0. Самый старший бит (MSB) отправленного байта должен быть первым, а скорость SCLK не должна превышать 10 МГц.

2.3 Описание используемого программного обеспечения

2.3.1 Важность свободного программного обеспечения

Свободное программное обеспечение играет важную роль для сотрудничества и развития поскольку оно обеспечивает технологический суверенитет, способствует национальным инновациям, оптимизирует расходы на создание собственного программного обеспечение, ускоряет местное развитие и способствует цифровой интеграции.

Использование открытого программного обеспечения позволит инфраструктуре Интернета вещей:

— приобрести технологическую автономию: доступ к исходному коду позволит многим пользователям перейти от потребителей к разработчикам программного обеспечения;

— провести стандартизацию и интеграцию: свободное программное обеспечение создается с использованием спецификаций и бесплатных общедоступных технологических стандартов, также называемыми “открытыми стандартами”. Это приносит пользу интеграции систем и обмену информацией, гарантирует доступ без ограничений для всех пользователей;

— обрести безопасность. Публикация исходных текстов программ или приложений способствует их безопасности. Используя открытое программное обеспечение, можно узнать и проанализировать, что фактически выполняется программой, тип информации, который она обрабатывает и как ей управлять. Хорошая безопасность должна основываться на прозрачности. Проприетарное программное обеспечение скрывает эти аспекты, и часто неизвестно, сохраняется ли конфиденциальность отправляемой информации;

— приобрести независимых поставщиков программно-аппаратного обеспечения: использование проприетарного программного обеспечения создает зависимость от производителя. Как только такое программное обеспечение будет установлено, оно будет зависеть от получения обновлений. Во многих случая производитель будет принуждать к обновлению до новых версии, даже если это нежелательно.

— добиться демократизации информации: информационные технологии заняли существенное положение в обществе. Хотя все больше и больше пользователей обращаются к указанным технологиям, “технологический разрыв” по-прежнему велик и является ещё одним фактором социальной изоляции;

— добиться экономичности: покупка проприетарного ПО, особенно когда производитель имеет монополию на данный вид программного продукта и используемых в нём алгоритмов, стоит несравненно больше, чем приобретение и использование программного продукта на основе отрытого программного обеспечения;

2.3.2 Основа проекта

За основу для разработки ПО для микроконтроллера STM32L476VGT6 в связке с трансивером SX1278 был взят общедоступный проект от разработчиков компании Semtech, находящийся по адресу `https://github.com/Lora-net/LoRaMac-node` (дата обращения 05.06.2018). Данное свободное программное обеспечение распространяется под лицензией BSD (*BSD license, Berkeley Software Distribution license — Программная лицензия университета Беркли*).

Данная лицензия допускает повторное распространение и использование как в виде исходного кода, так и в двоичном виде, с изменениями или без, при соблюдении следующих условий [14]:

— при повторном распространении исходного кода должно оставаться уведомление об авторском праве, этот список условий и последующий отказ от гарантий;

— при повторном распространении двоичного кода должна сохраняться указанная выше информация об авторском праве, этот список условий и последующий отказ от гарантий в документации и/или в других материалах, поставляемых при распространении;

— ни организация, ни имена её сотрудников не могут быть использованы в качестве поддержки или продвижения продуктов, основанных на этом ПО без предварительного письменного разрешения.

Основным используемым языком выбранного проекта является язык программирования Си и это не случайно. Дело в том, что для обеспечения компромисса между производительностью, качеством, скоростью, кроссплатформенностью легкостью разработки необходим высокоуровневый язык с возможностью кросскомпиляции на различные аппаратные платформы и, при этом, он должен иметь наилучшие показатели производительности среди всех языков по сравнению с аналогичной программой, написанной на ассемблере. Язык программирования Си удовлетворяет большей части этих требований.

Однако, для дальнейшего развития инфраструктуры Интернета вещей следует исключить из поставщиков услуг разработчиков ПО и предоставить пользователям системы возможность самостоятельно, с минимальными знаниями о информационных технологиях, изменять программное обеспечение вещей для удо-

влетворения собственных нужд. Потребуется создание продвинутых визуальных языков программирования и предметно-ориентированных языков использующих графическое представление данных и алгоритмов.

2.3.3 Используемое программное обеспечение

Для разработки использовалось следующее программное обеспечение:

— операционная система Linux Kubuntu 17.04;

— система контроля версии Git;

— текстовый редактор Vim;

— система автоматизации сборки CMake;

— утилита автоматизации преобразования файлов make;

— утилита прямой отладки программ на микроконтроллере OpenOCD;

— отладчик gdb для процессоров ARM;

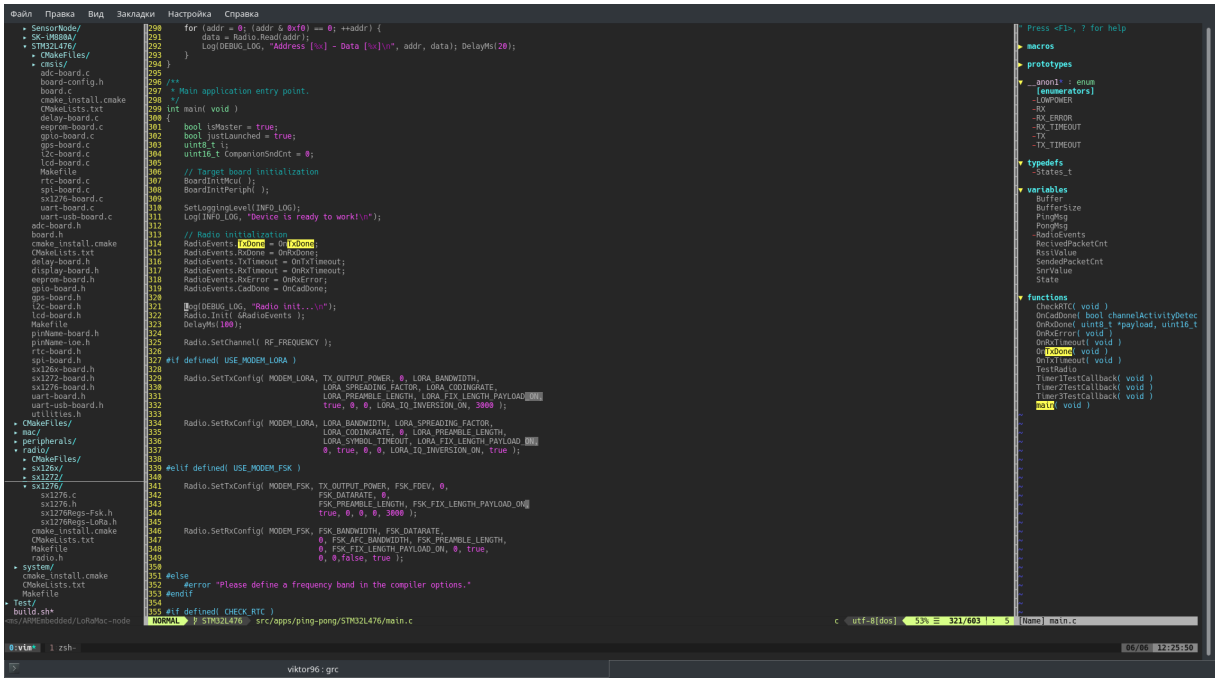

Рисунок 2.3 — Текстовый редактор Vim

2.3.3.1 CMake

CMake — это система сборки кроссплатформенного программного обеспечения из исходного кода. Она не занимается непосредственной сборкой, а лишь генерирует файлы управления сборкой исходя из инструкции в файле CMakeLists.txt. Такими файлами управления сборкой могут служить файлы Makefile в системах

Unix для сборки с помощью make, а также проекты XCode для Mac OS X и файлы project/solutions (*.sln/.vcxproj/.vcproj*) в *Windows* для сборки с помощью Visual C++.

CMake используется в выбранном проекте и пример файла CMakeLists.txt для сборки проектов под STM32L4 можно посмотреть в листинге 1.

```
1  ## Authors: ObKo (https://github.com/ObKo/)
2
3  if(NOT DEFINED LINKER_SCRIPT)
4  message(FATAL_ERROR "No linker script defined")
5  endif(NOT DEFINED LINKER_SCRIPT)
6  message("Linker script: ${LINKER_SCRIPT}")
7
8  SET(OBJECT_GEN_FLAGS "-Og -g -mthumb -fno-builtin -mcpu=cortex-m4 -Wall
   ↪  -specs=nano.specs -specs=nosys.specs -mfpu=fpv4-sp-d16
   ↪  -mfloat-abi=softfp -ffunction-sections -fdata-sections
   ↪  -fomit-frame-pointer -mabi=aapcs -fno-unroll-loops -ffast-math
   ↪  -ftree-vectorize")
9  SET(CMAKE_C_FLAGS "${OBJECT_GEN_FLAGS} -std=gnu99" CACHE INTERNAL "C
   ↪  compiler flags")
10 SET(CMAKE_CXX_FLAGS "${OBJECT_GEN_FLAGS} -std=c++11" CACHE INTERNAL "cxx
   ↪  compiler flags")
11 SET(CMAKE_ASM_FLAGS "${OBJECT_GEN_FLAGS} -x assembler-with-cpp" CACHE
   ↪  INTERNAL "asm compiler flags")
```

Листинг 1 — Инструкции для кросс-компиляции кода на платформу STM32L4 (часть файла)

2.3.3.2 HAL

Для разработки программного обеспечения к аппаратной платформе STM32L476VGT6 решено было использовать слой аппаратных абстракций (HAL), разработанный производителями микроконтроллеров семейства STM32 и распространяющийся на свободной основе.

HAL предоставляет возможность создания кода, не зависящего от аппаратных особенностей выбранной платформы. Хотя данный слой написан на языке Си, он реализует в себе множество идей объектно-ориентированного программирования (ООП). Самой важной идеей разработки такого программного обеспечения, как слой аппаратных абстракций, является инкапсуляция. Инкапсуляция позволяет абстрагироваться от сложности разработки на низком уровне, предоставляя удобный интерфейс для написания программ работающих с различными устройствами, протоколами передачи данных и т.д.

2.3.3.3 ST-LINK/V2 и Openocd

ST-LINK/V2 является внутрисхемным отладчиком и программатором для отладки микроконтроллеров семейства STM32 и STM8. Данный отладчик спроектирован на базе микроконтроллера STM32F103C8, который включает в себя высокопроизводительное ядро ARM-Cortex M3. Для внутрисхемной отладки он использует JTAG/SWD/SWIM интерфейсы отлаживаемого микроконтроллера.

ST-LINK/V2 уже входит в состав отладочной платы STM32L4-Discovery.

OpenOCD — программное обеспечение для ОС Linux, Windows и Mac OS, реализующий интерфейс отлаживаемого микроконтроллера через внутрисхемный отладчик, такой как ST-LINK/V2 и другие. При запуске OpenOCD находит подключенный к компьютеру внутрисхемный отладчик, устанавливает с ним связь и открывает локальный TCP сервер, для последующего подключения к нему программного отладчика. TCP сервер OpenOCD принимает команды от программного отладчика и отсылает их внутрисхемному отладчику. Инструкции подключения OpenOCD к внутрисхемному отладчику отображены в листинге 2.

```
# This is an STM32L476G-DISCO board with a single STM32L476VGTx chip
#
# Generated by System Workbench for STM32
# Take care that such file, as generated, may be overridden without any
↪  early notice. Please have a look to debug launch configuration setup(s)

source [find interface/stlink-v2-1.cfg]

set WORKAREASIZE 0x8000

transport select "hla_swd"

set CHIPNAME STM32L476VGTx

# Enable debug when in low power modes
set ENABLE_LOW_POWER 1

# Stop Watchdog counters when halt
set STOP_WATCHDOG 1

# STlink Debug clock frequency
set CLOCK_FREQ 4000000

# use hardware reset, connect under reset
# connect_assert_srst needed if low power mode application running (WFI...)
reset_config srst_only srst_nogate connect_assert_srst
set CONNECT_UNDER_RESET 1

source [find target/stm32l4x.cfg]
```

Листинг 2 — Инструкции подключения OpenOCD к внутрисхемному отладчику

В качестве программного отладчика была использована утилита arm-none-eabi-gdb, конфигурации для подключения можно увидеть в листинге 3.

```
file ping-pong
target extended-remote localhost:3333
monitor reset halt
load
thbreak main
```

Листинг 3 — Инструкции подключения программного отладчика gdb к серверу OpenOCD

2.3.4 Структура проекта LoRaMac

Код проекта LoRaMAC разбит на несколько пакетов, каждый из которых реализует самостоятельный компонент программного обеспечения устройства:

— *system* — слой аппаратных абстракций “система”, использующийся пакетами поддержки радиомодулей и алгоритмов MAC. Является интерфейсом к используемому аппаратному обеспечению.

— *boards* — содержит реализации интерфейса абстракции “система” для каждого вида микроконтроллера;

— *radio* — пакет, содержащий интерфейс и реализацию алгоритмов и данных для работы с приёмопередатчиками LoRa™ такими, как SX1272, SX1276, SX126x и другими. Для своей работы этот пакет использует слой аппаратных абстракции “система”.

— *mac* — пакет, реализующий данные и алгоритмы работы протокола LoRaWAN™, включая шифрование данных и регионально-зависимые настройки несущих частот и прочих настроек физического уровня LoRa™;

— *apps* — примеры готовых приложений;

— *peripherals* — различные драйверы для периферии, доступной на некоторых аппаратных платформах.

Упрощенную структуру проекта можно посмотреть на рисунке 2.4.

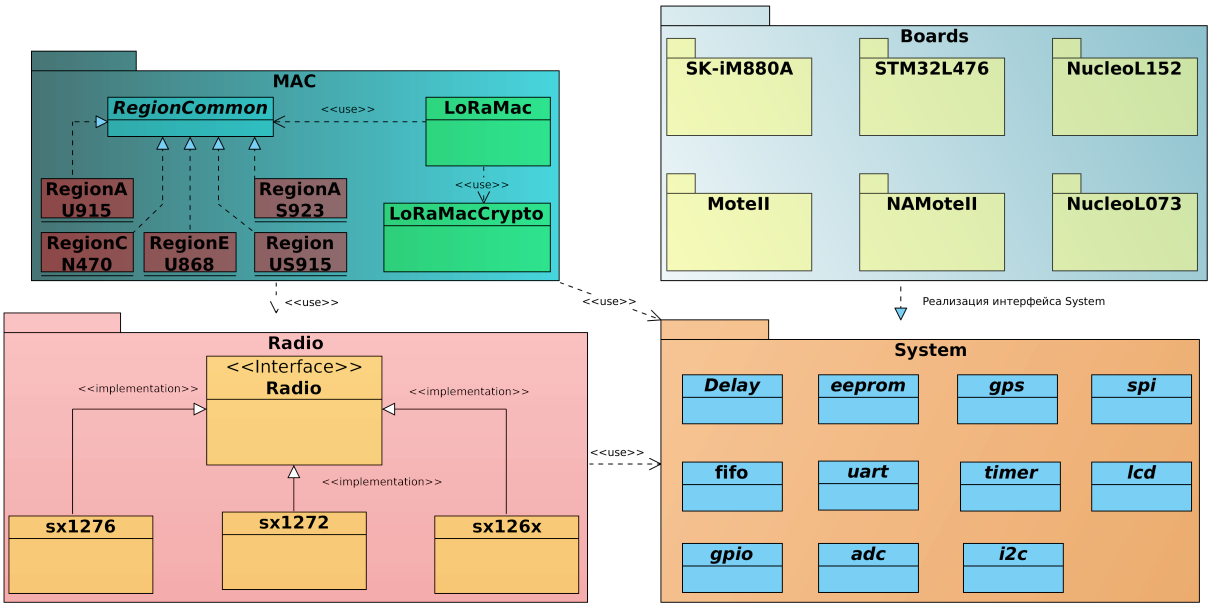

Рисунок 2.4 — Упрощенная структура проекта LoRaMAC в нотации UML

2.4 Реализация проекта

Проект LoRaMAC не содержал в себе исходного кода для реализации связки микроконтроллера STM32L476VGT6 на базе ядра ARM Cortex-M4, таким образом следовало сперва создать правила компиляции исходного кода для данной архитектуры. Эти правила описаны в файле *stm32l4.cmake* и сам файл размещён в папке cmake в корне проекта. Часть содержимого этого файла отображено на листинге 1.

Как можно видеть из листинга, требуется также файл с инструкциями для компоновщика под выбранную аппаратную платформу. Код для компоновщика поставляется разработчиком аппаратных абстракции HAL.

Далее необходимо:

а) добавить новую директорию в пакет *boards* с именем используемой аппаратной платформы;

б) реализовать интерфейс “системы” описав следующие модули под используемую аппаратную платформу:

1) board.c/.h;

2) gpio-board.c/.h;

3) piName-board.c/.h;

4) piName-ioe.c/.h;

5) rtc-board.c/.h;

6) spi-board.c/.h.

в) перенести в boards/mcu проект с HAL для STM32L4;

г) занести конфигурацию оборудования в файл board-config.h.

2.4.1 Подключение STM32 к трансиверам LoRa™

Отладочная плата была соединена к интерфейсу SPI пирёмопередатчика SX1278. Соответствие ножек и их функции приведено в таблице 2.3.

Собранный прототип отображен на рисунке 2.5.

2.4.2 Сложности переноса

Во время переноса кода на новую аппаратную платформу пришлось разрешить некоторые проблемы. Во-первых, слой аппаратных абстракции не рассматри-

Таблица 2.3 — Соответствие ножек отладочной платы и выполняемой функции

Ножка отладочной платы STM32L4-Discovery	Ножка радиомодуля SX1278
PE12	NSS
PE13	SCK
PE14	MISO
PE15	MOSI
PA5	DIO0
PA1	DIO1
PA2	DIO2
PA3	DIO3
PE11	DIO4
PE10	DIO5
PB7	RESET

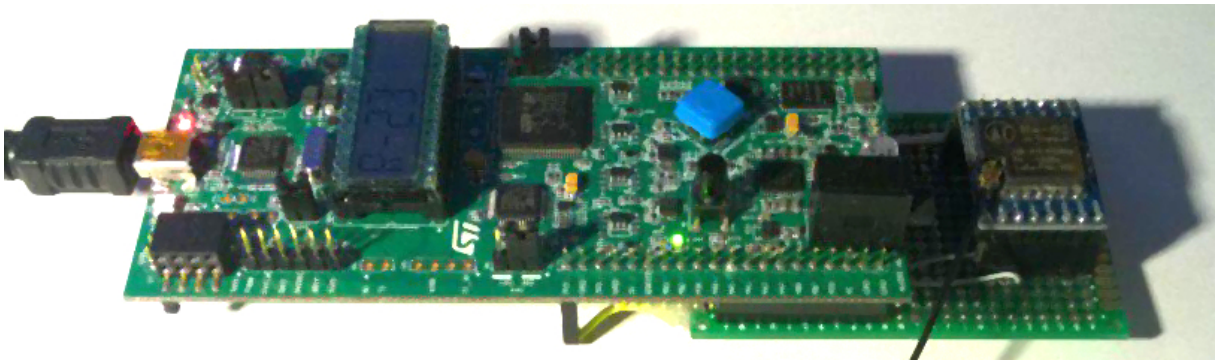

Рисунок 2.5 — Собранный прототип конечного устройства

вает использование новой архитектуры ARM Cortex-M4, приходилось подбирать верные ключи для кросс-компиляции. Во-вторых, также не предполагалось использование SX1278, поэтому пришлось перенастроить используемые частотные диапазоны в исходных файлах. В-третьих, сам слой аппаратных абстракции содержит ошибки (неточности) в выбранных типах данных для интерфейсов. Поскольку эти типы данных основаны на HAL, а HAL развивается отдельно, то в слое аппаратных абстракций LoRaMAC возникают множества неприятных ловушек, связанных с использование старой версии HAL.

Приведу конкретный пример: во время инициализации интерфейса SPI вызывалась функция выбора формата сигнала SPI — `SpiFormat`:

```
1 if( nss == NC )
2 {
3     SpiHandle[spiId].Init.NSS = SPI_NSS_SOFT;
```

```
    SpiFormat( obj, SPI_DATASIZE_8BIT, SPI_POLARITY_LOW, SPI_PHASE_1EDGE, 0 );
}
else
{
    SpiFormat( obj, SPI_DATASIZE_8BIT, SPI_POLARITY_LOW, SPI_PHASE_1EDGE, 1 );
}
```

Второй параметр определяет сколько бит данных должен передавать один кадр SPI (в момент времени когда на линии *NSS* логическая единица). Готовый интерфейс аппаратных абстракций содержит следующее определение:

```
/*!
 * \brief Configures the SPI peripheral
 *
 * \remark Slave mode isn't currently handled
 *
 * \param [IN] obj   SPI object
 * \param [IN] bits  Number of bits to be used. [8 or 16]
 * \param [IN] cpol  Clock polarity
 * \param [IN] cpha  Clock phase
 * \param [IN] slave When set the peripheral acts in slave mode
 */
void SpiFormat( Spi_t *obj, int8_t bits, int8_t cpol, int8_t cpha, int8_t slave
);
```

Здесь второй параметр определен как 8-битное целое число. Видимо первоначально предполагалось что туда будет передаваться всего два возможных значения — `SPI_DATASIZE_8BIT` и `SPI_DATASIZE_16BIT`. Так оно и было в старой версии HAL, однако в новой версии HAL для STM32L4 добавились новые значения:

```
/** @defgroup SPI_Data_Size SPI Data Size
  * @{
  */
#define SPI_DATASIZE_4BIT               (0x00000300U)
#define SPI_DATASIZE_5BIT               (0x00000400U)
#define SPI_DATASIZE_6BIT               (0x00000500U)
#define SPI_DATASIZE_7BIT               (0x00000600U)
#define SPI_DATASIZE_8BIT               (0x00000700U)
```

```
#define SPI_DATASIZE_9BIT                (0x00000800U)
#define SPI_DATASIZE_10BIT               (0x00000900U)
#define SPI_DATASIZE_11BIT               (0x00000A00U)
#define SPI_DATASIZE_12BIT               (0x00000B00U)
#define SPI_DATASIZE_13BIT               (0x00000C00U)
#define SPI_DATASIZE_14BIT               (0x00000D00U)
#define SPI_DATASIZE_15BIT               (0x00000E00U)
#define SPI_DATASIZE_16BIT               (0x00000F00U)
```

Как можно видеть теперь младший байт для передачи значения не используется, что приводило к неоднозначной ошибке исполнения программы: программа продолжала выполнятся, а SPI передавал 16 бит вместо 8. Ушло примерно два часа на анализ пока ошибка не была найдена и теперь определение функции выглядит так:

```
void SpiFormat( Spi_t *obj, int32_t bits, int8_t cpol, int8_t cpha, int8_t
slave
);
```

Ошибка была исправлена. Безусловно подобные ошибки происходят не только по вине человека, но и по несовершенству средств программирования. Язык Си является языком с, так называемой, мягкой типизацией, которая позволяет таким ошибкам происходить совершенно незаметно для разработчика.

3 Проверка работы устройства

3.1 Ping-Pong

Было собрано два идентичных устройства и портирован код приложения Ping-Pong, являющийся в беспроводных приложениях подобием доказательства концепции (*proof of concept*).

Приложение было модернизировано, добавление системы логирования и счётчиками отправленных и принятых пакетов. Логирование осуществлялась посредством виртуального COM интерфейса STM32L4. Во время подключения к компьютеру плата отправляет через COM интерфейс отладочные данные. Пример отладочных данных дан ниже:

```
6  src/apps/ping-pong/STM32L476/main.c:563:L0 OnRxDone: Payload size=64
   ↪  RSSI=-118 New state = RX
7  src/apps/ping-pong/STM32L476/main.c:439:L0 CompanionSndCnt=123
   ↪  RecivedCnt=112
8  src/apps/ping-pong/STM32L476/main.c:461:L0 PONG sended...
9  src/apps/ping-pong/STM32L476/main.c:563:L0 OnRxDone: Payload size=64
   ↪  RSSI=-112 New state = RX
10 src/apps/ping-pong/STM32L476/main.c:439:L0 CompanionSndCnt=124
   ↪  RecivedCnt=113
11 src/apps/ping-pong/STM32L476/main.c:461:L0 PONG sended...
```

Полный исходный код приложения Ping-Pong будет дан в приложении А.

3.2 Проверка работы приёмопередатчиков

Для проверки работы приложения и оценки качества сигнала одно из устройств было оставлено на высоте 18 метров (5 этаж дома) с автономным источником питания (см. рис. 3.1. Второе устройство было взято с собой для измерения сигнала. В результате была установлена связь с между модулями и оценено качество связи по параметру Packet Error Rate (**PER**). Это отношение пропущенных (не принятых) пакетов к общему количеству отправленных.

Результаты измерений отображены на рисунке 3.2 и в таблице 3.1.

Используемые параметры физического уровня: модуляция LoRa, SF = 7, BW = 125 кГц, несущая частота равна 433 МГц, CR = 4/5, длина преамбулы равна 8, усиление передатчика - 14 дБ. Использовалась антенна с усилением 4 дБ.

Рисунок 3.1 — Автономное устройство, передающее кадры Ping в приложении Ping-Pong

Таблица 3.1 — Результаты замеров

Описание места	Расстояние до БС, м	PER, %	RSSI, дБм
За домом. Несколько стен.	40	0,8	-94
Улица. Прямая видимость БС	293	1	-88
В магазине Видимости нет. Глубоко внутри помещения	280	100	Нет сигнала
Улица. Прямой видимости нет. Рельеф закрывает БС	687	14,9	-102
Улица. Прямой видимости нет. Плотная застройка.	524	24,4	-112

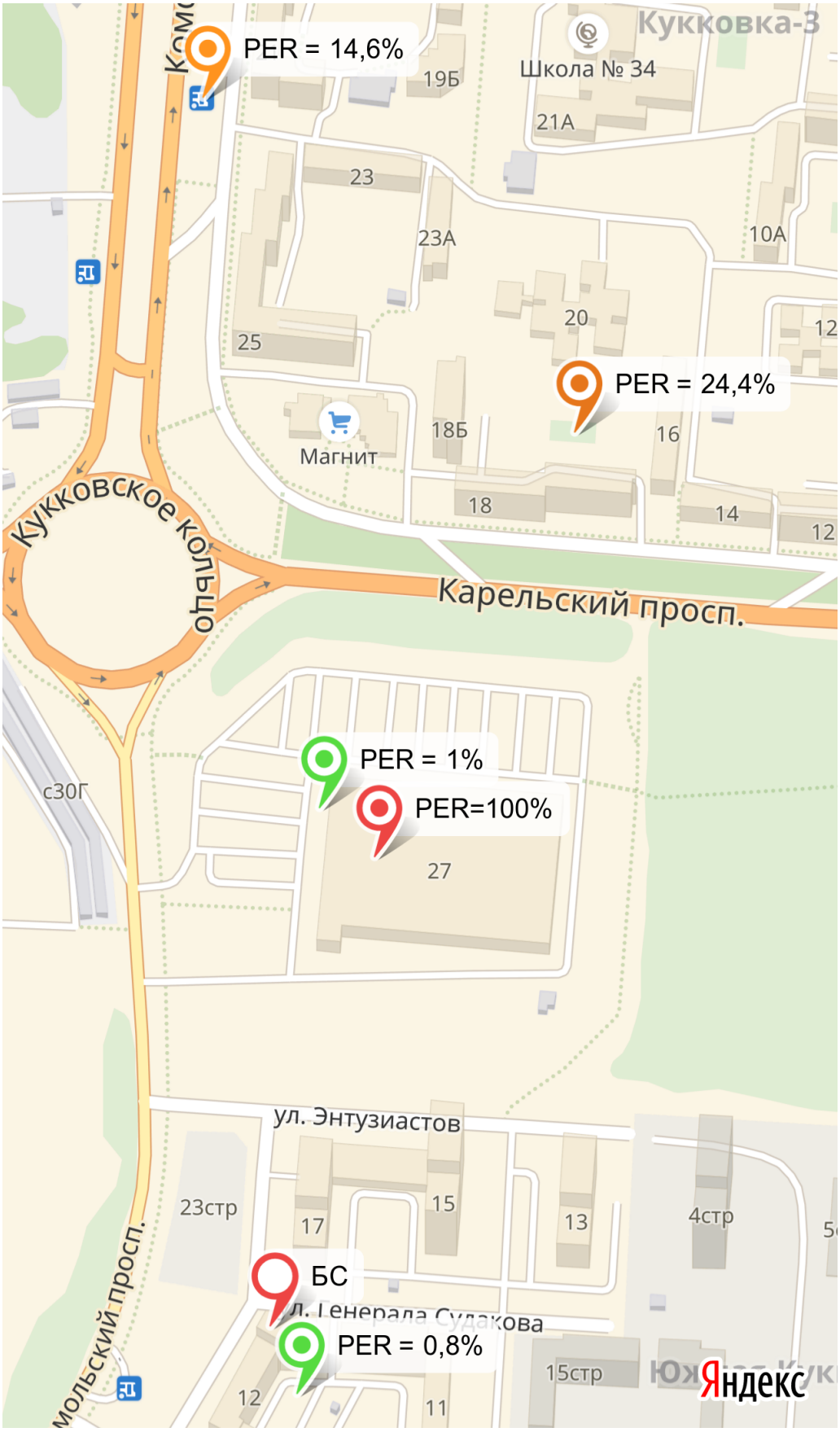

Рисунок 3.2 — Результаты замеров качества приёма пакетов

ЗАКЛЮЧЕНИЕ

В ходе исследования LoRa в контексте её использования для построения сети Интернета вещей, были выполнены следующие задачи:

— оценены достоинства и недостатки применения данной технологии для построения LPWAN сетей;

— рассмотрен и предложен ряд вариантов использования данной технологии с поправкой на её особенности в концепции Интернета вещей.

— было разработано программное обеспечение для сопряжения STM32L4 с трансивервами LoRa;

— разработанное ПО было задокументировано и выложено в открытый доступ;

Был приобретён навык работы с исходными кодами программного обеспечения, распространяющегося на свободной основе. Также приобретён навык переноса исходного кода на новую аппаратную платформу, что немаловажно для создания кросс-платформенного ПО, которое станет основой для инфраструктуры Интернета вещей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Evans Dave. The internet of things: How the next evolution of the internet is changing everything // CISCO white paper. — 2011. — Т. 1, № 2011. — С. 1–11.

2. Л. Черняк. Интернет вещей: новые вызовы и новые технологии // Открытые системы. СУБД. — 2013. — № 4. — С. 14–18.

3. Denise Carrie Vernon, Mario. Worldwide and Regional Internet of Things (IoT) 2014–2020 Forecast: A Virtuous Circle of Proven Value and Demand // IDC Anal. Futur. — 2014.

4. В. Росляков А. Интернет вещей. — Самара: ПГУТИ, ООО «Издательство Ас Гард, 2014.

5. Pickard Southworth Drummond. The IPv6 Internet: An Assessment of Adoption and Quality of Services // Journal of International Technology and Information Management. — 2017. — Т. 26.

6. МСЭ-Т. Рекомендация Y.2060. Обзор интернета вещей. — 2012. — Июнь.

7. A study of LoRa: Long range & low power networks for the internet of things / Aloÿs Augustin, Jiazi Yi, Thomas Clausen, William Mark Townsley // Sensors. — 2016. — Т. 16, № 9. — С. 1466.

8. Баскаков СИ. Радиотехнические сигналы и цепи. М. — Высшая школа, 2003.

9. Sikken Bertrik. DecodingLora [Электронный ресурс]. — Режим доступа: `https://revspace.nl/DecodingLora` (дата обращения: 03.06.2018).

10. Lavric Alexandru, Popa Valentin. Internet of Things and LoRa™ Low-Power Wide-Area Networks: A survey // Signals, Circuits and Systems (ISSCS), 2017 International Symposium on / IEEE. — 2017. — С. 1–5.

11. Using LoRa for industrial wireless networks / Mattia Rizzi, Paolo Ferrari, Alessandra Flammini и др. // IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS. — 2017.

12. IPv6 over LoRaWAN™ / Patrick Weber, Daniel Jäckle, David Rahusen, Axel Sikora // Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), 2016 3rd International Symposium on / IEEE. — 2016. — С. 75–79.

13. Светильники пятого поколения [Электронный ресурс]. — Режим доступа: `https://i-sberg.com/index.php/m-sol/esl` (дата обращения: 04.06.2018).

14. Текст лицензии BSD [Электронный ресурс]. — Режим доступа: https://opensource.org/licenses/BSD-3-Clause (дата обращения: 07.06.2018).

15. Georgiou Orestis, Raza Usman. Low Power Wide Area Network Analysis: Can LoRa Scale? // IEEE Wireless Communications Letters. — 2017. — Т. 6, № 2. — С. 162–165. — 1610.04793.

16. Varsier Nadege, Schwoerer Jean. Capacity limits of LoRaWAN technology for smart metering applications // IEEE International Conference on Communications. — 2017.

17. Stan Valentin Alexandru, Timnea Radu Serban, Gheorghiu Razvan Andrei. Overview of high reliable radio data infrastructures for public automation applications: LoRa networks // Proceedings of the 8th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2016. — 2017.

18. Tanenbaum Andrew S. Computer Networks. — 2011. — Т. 52. — С. 349–351. — ISBN: 0130661023. — 1011.1529.

19. Vangelista L. Frequency Shift Chirp Modulation: The LoRa Modulation // IEEE Signal Processing Letters. — 2017. — Т. 24, № 12. — С. 1818–1821.

20. A LoRa enabled building automation architecture based on MQTT / Susanna Spinsante, Gianluca Ciattaglia, Antonio Del Campo и др. // AEIT International Annual Conference, 2017 / IEEE. — 2017. — С. 1–5.

21. Sanchez-Gomez Jesus, Sanchez-Iborra Ramon, Skarmeta Antonio. Transmission Technologies Comparison for IoT Communications in Smart-Cities. — 2017.

22. TELIT. Why IIoT Projects Fail: 3 Secrets to solving the chokepoints // TELIT WHITEPAPER. — 2017. — Нояб.

23. Kafle V. P., Fukushima Y., Harai H. Internet of things standardization in ITU and prospective networking technologies // IEEE Communications Magazine. — 2016. — September. — Т. 54, № 9. — С. 43–49.

ПРИЛОЖЕНИЕ А

ЛИСТИНГИ ПРОГРАММ

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "board.h"
#include "gpio.h"
#include "uart.h"
#include "delay.h"
#include "timer.h"
#include "lcd.h"
#include "radio.h"
#include "logging.h"

#include "rtc-board.h"

#define REGION_EU433

#if defined( REGION_AS923 )

#define RF_FREQUENCY                                923000000 // Hz

#elif defined( REGION_AU915 )

#define RF_FREQUENCY                                915000000 // Hz

#elif defined( REGION_CN470 )

#define RF_FREQUENCY                                470000000 // Hz

#elif defined( REGION_CN779 )

#define RF_FREQUENCY                                779000000 // Hz

#elif defined( REGION_EU433 )

#define RF_FREQUENCY                                433000000 // Hz

#elif defined( REGION_EU868 )

#define RF_FREQUENCY                                868000000 // Hz

#elif defined( REGION_KR920 )

#define RF_FREQUENCY                                920000000 // Hz
```

```
44
45  #elif defined( REGION_IN865 )
46
47  #define RF_FREQUENCY                                865000000 // Hz
48
49  #elif defined( REGION_US915 )
50
51  #define RF_FREQUENCY                                915000000 // Hz
52
53  #elif defined( REGION_US915_HYBRID )
54
55  #define RF_FREQUENCY                                915000000 // Hz
56
57  #else
58      #error "Please define a frequency band in the compiler options."
59  #endif
60
61  #define TX_OUTPUT_POWER                             14        // dBm
62
63  #if defined( USE_MODEM_LORA )
64
65  #define LORA_BANDWIDTH                              0         // [0: 125 kHz,
66                                                                //  1: 250 kHz,
67                                                                //  2: 500 kHz,
68                                                                //  3: Reserved]
69  #define LORA_SPREADING_FACTOR                       7         // [SF7..SF12]
70  #define LORA_CODINGRATE                             1         // [1: 4/5,
71                                                                //  2: 4/6,
72                                                                //  3: 4/7,
73                                                                //  4: 4/8]
74  #define LORA_PREAMBLE_LENGTH                        8         // Same for Tx
    ↪  and Rx
75  #define LORA_SYMBOL_TIMEOUT                         5         // Symbols
76  #define LORA_FIX_LENGTH_PAYLOAD_ON                  false
77  #define LORA_IQ_INVERSION_ON                        false
78
79  #elif defined( USE_MODEM_FSK )
80
81  #define FSK_FDEV                                    25000     // Hz
82  #define FSK_DATARATE                                50000     // bps
83  #define FSK_BANDWIDTH                               50000     // Hz
84  #define FSK_AFC_BANDWIDTH                           83333     // Hz
85  #define FSK_PREAMBLE_LENGTH                         5         // Same for Tx
    ↪  and Rx
86  #define FSK_FIX_LENGTH_PAYLOAD_ON                   false
87
88  #else
```

```
    #error "Please define a modem in the compiler options."
#endif

#define INFO_LOG 0
#define DEBUG_LOG 1

typedef enum
{
    LOWPOWER,
    RX,
    RX_TIMEOUT,
    RX_ERROR,
    TX,
    TX_TIMEOUT,
}States_t;

#define RX_TIMEOUT_VALUE                            2000
#define BUFFER_SIZE                                 64 // Define the payload
 ↪  size here
#define NEW_SESSION_CMD                 0x01
#define IS_NEW_SESSION(b)               ((b & NEW_SESSION_CMD) ==
 ↪  NEW_SESSION_CMD)
#define HWORD(w)                    ((uint8_t) (w >> 8) & 0xFF)
#define LWORD(w)                    ((uint8_t) (w) & 0xFF)
#define GETW(h, l)                    ((uint16_t) (((h & 0xFF) << 8) | (l &
 ↪  0xFF)))

const uint8_t PingMsg[] = "PING";
const uint8_t PongMsg[] = "PONG";

uint16_t BufferSize = BUFFER_SIZE;
uint8_t Buffer[BUFFER_SIZE];

States_t State = LOWPOWER;

int8_t RssiValue = 0;
int8_t SnrValue = 0;

int16_t RecivedPacketCnt = 0;
int16_t SendedPacketCnt = 0;

/*!
 * Radio events function pointer
 */
static RadioEvents_t RadioEvents;

/*!
```

```
133  * LED GPIO pins objects
134  */
135 extern Gpio_t Led4;
136 extern Gpio_t Led5;
137
138 /*!
139  * Uart2 Handle
140  */
141 extern Uart_t Uart2;
142
143 /*!
144  * \brief Function to be executed on Radio Tx Done event
145  */
146 void OnTxDone( void );
147
148 /*!
149  * \brief Function to be executed on Radio Rx Done event
150  */
151 void OnRxDone( uint8_t *payload, uint16_t size, int16_t rssi, int8_t snr );
152
153 /*!
154  * \brief Function executed on Radio Tx Timeout event
155  */
156 void OnTxTimeout( void );
157
158 /*!
159  * \brief Function executed on Radio Rx Timeout event
160  */
161 void OnRxTimeout( void );
162
163 /*!
164  * \brief Function executed on Radio Rx Error event
165  */
166 void OnRxError( void );
167
168 /* Прослушка на предмет активности канала */
169 void OnCadDone( bool channelActivityDetected );
170
171 /**
172  * Main application entry point.
173  */
174 int main( void )
175 {
176     bool isMaster = true;
177   bool justLaunched = true;
178     uint8_t i;
179   uint16_t CompanionSndCnt = 0;
```

```
180
181     // Target board initialization
182     BoardInitMcu( );
183     BoardInitPeriph( );
184
185   SetLoggingLevel(INFO_LOG);
186   Log(INFO_LOG, "Device is ready to work!\n");
187
188     // Radio initialization
189     RadioEvents.TxDone = OnTxDone;
190     RadioEvents.RxDone = OnRxDone;
191     RadioEvents.TxTimeout = OnTxTimeout;
192     RadioEvents.RxTimeout = OnRxTimeout;
193     RadioEvents.RxError = OnRxError;
194   RadioEvents.CadDone = OnCadDone;
195
196   Log(DEBUG_LOG, "Radio init...\n");
197     Radio.Init( &RadioEvents );
198   DelayMs(100);
199
200     Radio.SetChannel( RF_FREQUENCY );
201
202 #if defined( USE_MODEM_LORA )
203
204     Radio.SetTxConfig( MODEM_LORA, TX_OUTPUT_POWER, 0, LORA_BANDWIDTH,
205                                    LORA_SPREADING_FACTOR, LORA_CODINGRATE,
206                                    LORA_PREAMBLE_LENGTH,
                                       ↪  LORA_FIX_LENGTH_PAYLOAD_ON,
207                                   true, 0, 0, LORA_IQ_INVERSION_ON, 3000 );
208
209     Radio.SetRxConfig( MODEM_LORA, LORA_BANDWIDTH, LORA_SPREADING_FACTOR,
210                                  LORA_CODINGRATE, 0, LORA_PREAMBLE_LENGTH,
211                                    LORA_SYMBOL_TIMEOUT,
                                       ↪  LORA_FIX_LENGTH_PAYLOAD_ON,
212                                0, true, 0, 0, LORA_IQ_INVERSION_ON, true );
213
214 #elif defined( USE_MODEM_FSK )
215
216     Radio.SetTxConfig( MODEM_FSK, TX_OUTPUT_POWER, FSK_FDEV, 0,
217                                    FSK_DATARATE, 0,
218                                    FSK_PREAMBLE_LENGTH,
                                       ↪  FSK_FIX_LENGTH_PAYLOAD_ON,
219                                    true, 0, 0, 0, 3000 );
220
221     Radio.SetRxConfig( MODEM_FSK, FSK_BANDWIDTH, FSK_DATARATE,
222                                  0, FSK_AFC_BANDWIDTH, FSK_PREAMBLE_LENGTH,
223                                    0, FSK_FIX_LENGTH_PAYLOAD_ON, 0, true,
```

```
                                    0, 0,false, true );

#else
    #error "Please define a frequency band in the compiler options."
#endif

    Radio.Rx( RX_TIMEOUT_VALUE );

    while( 1 )
    {
        switch( State )
        {
        case RX:
      RecivedPacketCnt++;

            if( isMaster == true )
            {
                if( BufferSize > 0 )
                {
                    if( strncmp( ( const char* )Buffer, ( const char*
                    ↪  )PongMsg, 4 ) == 0 )
                    {
                        // Indicates on a LED that the received frame is a
                        ↪  PONG
                        GpioWrite( &Led5, GpioRead( &Led5 ) ^ 1 );

            // Read command from packet
            if( IS_NEW_SESSION( Buffer[4] ) )
              SendedPacketCnt = RecivedPacketCnt = 0;

            CompanionSndCnt =  GETW(Buffer[5], Buffer[6]);
            Log(INFO_LOG, "CompanionSndCnt=%i\tRecivedCnt=%i\n", \
                CompanionSndCnt, RecivedPacketCnt); DelayMs( 5 );
                        // Send the next PING frame
                        Buffer[0] = 'P';
                        Buffer[1] = 'I';
                        Buffer[2] = 'N';
                        Buffer[3] = 'G';
                        Buffer[4] = justLaunched ? NEW_SESSION_CMD : 0;

            // Число отправленных пакетов увеличивается на 1
            SendedPacketCnt = justLaunched ? 0 : SendedPacketCnt + 1;
            justLaunched = false;

            Buffer[5] = HWORD(SendedPacketCnt);
            Buffer[6] = LWORD(SendedPacketCnt);

```

```
269             for (i = 7; i < BufferSize; ++i)
270               Buffer[i] = 0;
271 
272                         DelayMs( 1 );
273                         Radio.Send( Buffer, BufferSize );
274 
275             Log(INFO_LOG, "PING sended...\n"); DelayMs( 100 );
276                     }
277                     else if( strncmp( ( const char* )Buffer, ( const char*
                        )PingMsg, 4 ) == 0 )
278                     { // A master already exists then become a slave
279                         isMaster = false;
280                         GpioWrite( &Led4, 1 ); // Set LED off
281             Log(DEBUG_LOG, "Waiting RX...\n"); DelayMs(20);
282                         Radio.Rx( RX_TIMEOUT_VALUE );
283                     }
284                     else // valid reception but neither a PING or a PONG
                        message
285                     {    // Set device as master ans start again
286                         isMaster = true;
287             Log(DEBUG_LOG, "Waiting RX...\n"); DelayMs(20);
288                         Radio.Rx( RX_TIMEOUT_VALUE );
289                     }
290                 }
291             }
292             else
293             {
294                 if( BufferSize > 0 )
295                 {
296                     if( strncmp( ( const char* )Buffer, ( const char*
                        )PingMsg, 4 ) == 0 )
297                     {
298                         // Indicates on a LED that the received frame is a
                            PING
299                         GpioWrite( &Led5, GpioRead( &Led5 ) ^ 1 );
300 
301             // Read command from packet
302             if( IS_NEW_SESSION( Buffer[4] ) )
303               SendedPacketCnt = RecivedPacketCnt = 0;
304 
305             CompanionSndCnt =  GETW(Buffer[5], Buffer[6]);
306             Log(INFO_LOG, "CompanionSndCnt=%i\tRecivedCnt=%i\n", \
307                 CompanionSndCnt, RecivedPacketCnt); DelayMs( 5 );
308 
309                         // Send the reply to the PONG string
310                         Buffer[0] = 'P';
311                         Buffer[1] = 'O';
```

```
312                     Buffer[2] = 'N';
313                     Buffer[3] = 'G';
314                     Buffer[4] = justLaunched ? NEW_SESSION_CMD : 0;
315 
316             // Число отправленных пакетов увеличивается на 1
317             SendedPacketCnt = justLaunched ? 0 : SendedPacketCnt + 1;
318             justLaunched = false;
319 
320             Buffer[5] = HWORD(SendedPacketCnt);
321             Buffer[6] = LWORD(SendedPacketCnt);
322 
323             for (i = 7; i < BufferSize; ++i)
324               Buffer[i] = 0;
325 
326                     DelayMs( 1 );
327                     Radio.Send( Buffer, BufferSize );
328 
329             Log(INFO_LOG, "PONG sended...\n"); DelayMs( 100 );
330                     }
331                     else // valid reception but not a PING as expected
332                     {    // Set device as master and start again
333                         isMaster = true;
334             Log(DEBUG_LOG, "Waiting RX...\n"); DelayMs(10);
335                         Radio.Rx( RX_TIMEOUT_VALUE );
336                     }
337                 }
338             }
339             State = LOWPOWER;
340             break;
341           case TX:
342             State = LOWPOWER;
343             break;
344           case RX_TIMEOUT:
345           case RX_ERROR:
346 
347         Log(DEBUG_LOG, "RX_TIMEOUT or RX_ERROR...\n"); DelayMs( 10 );
348 
349             if( isMaster == true )
350             {
351           // Ожидаем произвольное время от времени ожидания пакета
352           DelayMs(rand( ) % RX_TIMEOUT_VALUE);
353 
354                 // Send the next PING frame
355                 Buffer[0] = 'P';
356                 Buffer[1] = 'I';
357                 Buffer[2] = 'N';
358                 Buffer[3] = 'G';
```

```
359
360         Buffer[4] = justLaunched ? NEW_SESSION_CMD : 0;
361
362         // Число отправленных пакетов увеличивается на 1
363         SendedPacketCnt = justLaunched ? 0 : SendedPacketCnt + 1;
364
365         Buffer[5] = HWORD(SendedPacketCnt);
366         Buffer[6] = LWORD(SendedPacketCnt);
367
368         for (i = 7; i < BufferSize; ++i)
369           Buffer[i] = 0;
370
371                 DelayMs( 1 );
372                 Radio.Send( Buffer, BufferSize );
373
374         Log(INFO_LOG, "PING sended...\n"); DelayMs( 100 );
375             }
376             else
377             {
378         Log(DEBUG_LOG, "Waiting RX...\n"); DelayMs(20);
379                 Radio.Rx( RX_TIMEOUT_VALUE );
380             }
381             State = LOWPOWER;
382             break;
383         case TX_TIMEOUT:
384       Log(DEBUG_LOG, "Waiting RX...\n"); DelayMs(20);
385             Radio.Rx( RX_TIMEOUT_VALUE );
386             State = LOWPOWER;
387             break;
388         case LOWPOWER:
389         default:
390             // Set low power
391             break;
392         }
393
394         //TimerLowPowerHandler( );
395     }
396 }
397
398 void OnTxDone( void )
399 {
400     Radio.Sleep( );
401     State = TX;
402
403   Log(DEBUG_LOG, "OnTxDone. New state = TX\n"); DelayMs( 100 );
404   GpioWrite( &Led4, GpioRead( &Led4 ) ^ 1 );
405   Log(DEBUG_LOG, "Waiting RX...\n"); DelayMs(20);
```

```
406   Radio.Rx( RX_TIMEOUT_VALUE );
407 }
408
409 void OnRxDone( uint8_t *payload, uint16_t size, int16_t rssi, int8_t snr )
410 {
411   char buff[64] = {};
412
413     Radio.Sleep( );
414     BufferSize = size;
415     memcpy( Buffer, payload, BufferSize );
416     RssiValue = rssi;
417     SnrValue = snr;
418     State = RX;
419
420   LcdClear();
421   snprintf(buff, 64, "R%i", rssi);
422   LcdWriteString( (uint8_t *) buff);
423
424   Log(INFO_LOG, "OnRxDone: Payload size=%i RSSI=%i New state = RX\n", size,
      ↪  rssi);
425   DelayMs( 100 );
426 }
427
428 void OnTxTimeout( void )
429 {
430     Radio.Sleep( );
431     State = TX_TIMEOUT;
432
433   Log(DEBUG_LOG, "OnTxTimeout... New state = TX_TIMEOUT\n"); DelayMs( 100
      ↪  );
434 }
435
436 void OnRxTimeout( void )
437 {
438     Radio.Sleep( );
439     State = RX_TIMEOUT;
440
441   Log(DEBUG_LOG, "OnRxTimeout... New state = RX_TIMEOUT\n"); DelayMs( 100
      ↪  );
442 }
443
444 void OnRxError( void )
445 {
446     Radio.Sleep( );
447     State = RX_ERROR;
448
449   Log(DEBUG_LOG, "OnRxError! New state = RX_ERROR\n"); DelayMs( 100 );
```

```
}

void OnCadDone( bool channelActivityDetected )
{
  if (channelActivityDetected)
  {
    Log(DEBUG_LOG, "OnCadDone. Channel ACTIVITY DETECTED!!!\n"); DelayMs( 100 );
  }
  else
  {
    Log(DEBUG_LOG, "OnCadDone. Channel activity not detected...\n"); DelayMs( 100 );
    Radio.StartCad();
  }
}
```

Листинг 4 — Главная программа приложения Ping-Pong

```
/*
 * logging.h
 *
 *  Logging module that uses USART2 for Virtual Com
 *
 *  Created on: 18 мая 2018 г.
 *      Author: viktor96
 */

#ifndef LOGGING_H_
#define LOGGING_H_

#if defined( __FILENAME__ )
 #define Log(level, format, ...)  __log(level, __FILENAME__, __LINE__, format, ##__VA_ARGS__)
#else
 #define Log(level, format, ...)  __log(level, __FILE__, __LINE__, format, ##__VA_ARGS__)
#endif


void __log(int logLevel, char *filename, int lineno, char *fmtstr, ...);
void SetLoggingLevel(int loggingLevel);

#endif /* LOGGING_H_ */
```

Листинг 5 — Интерфейс модуля logging

```c
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include "logging.h"
#include "uart.h"

extern Uart_t Uart2;

// Default value is 0 (max level)
static int CurrentLogLevel = 0;

void SetLoggingLevel(int loggingLevel)
{
  CurrentLogLevel = loggingLevel;
}

void __log(int logLevel, char *filename, int lineno, char *fmtstr, ...)
{
        va_list ap;
        uint8_t logbuff[192] = {};
        uint8_t buff[128] = {};
    char logfmt[] = "%s:%i:L%i\t";

    if (logLevel > CurrentLogLevel)
      return;

    snprintf((char *) logbuff, 192, logfmt, filename, lineno, logLevel);

        va_start(ap, fmtstr);
        vsnprintf((char *) buff, 128, fmtstr, ap);
        va_end(ap);

    strcat((char *) logbuff, (char *) buff);

    UartPutBuffer(&Uart2, (uint8_t *) logbuff, strlen((char *) logbuff));
}
```

Листинг 6 — Реализация модуля logging