МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

ИНСТИТУТ МАШИНОСТРОЕНИЯ

(наименование института полностью)

Кафедра «Промышленная электроника»

(наименование)

11.04.04 «Электроника и наноэлектроника»

(код и наименование направления подготовки)

Промышленная электроника

(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Беспроводная система управления освещением на базе протокола ZigBee»

Студент О.И. Синичкин

(И.О. Фамилия) *(личная подпись)*

Руководитель К.т.н., доцент, Е.С. Глибин

(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Содержание

1. Актуальность. Состояние вопроса

В наше время с постоянно возрастающим темпом идёт информатизация общества. Технологии всё плотнее и плотнее входят в нашу жизнь. Однако в основном своём большинстве, устройства, использующиеся для передачи и обработки информации в быту, на заводах и предприятиях, на улицах и внутри отдельных зданий до сих пор требуют прокладки не только силовых питающих кабелей, но также и информационных кабелей управления.

Сейчас наиболее широко распространёнными методами беспроводной передачи информации является использование технологии Wi-Fi, Bluetooth и GSM/GPRS. Однако, по интегральному сочетанию стоимости, функциональных возможностей и габаритных размеров есть огромное количество применений, когда данные технологии крайне невыгодны.

В конечном исполнении альтернативное беспроводное устройство должно содержать высокоинтегрированный приёмопередатчик, к которому можно предъявить следующие требования:

- устойчивая двухсторонняя связь;
- дальность работы – 5 метров и более;
- малые габаритные размеры;
- низкое энергопотребление;
- невысокая цена для конечного пользователя;
- встроенный процессор для произведения локальных вычислений.

Таким условиям отвечает разработанный в 2001 году новый стандарт от ведущего института IEEE 802.15.4, который определяет функционирование беспроводных сетей, работающих на низкой скорости Low-Rate Wireless Personal Area Network (или LR-WPAN).

Стандарт IEEE 802.15.4 получил торговое название ZigBee (пчела, летающая зигзагом; связано с топологией сети). Всего за стандартом было

закреплено 27 каналов в трёх диапазонах частот. Глобальный ISM диапазон, т.е. 2,4 ГГц (2400–2483,6 МГц) содержит 16 каналов и один дополнительный диапазон в Соединённых Штатах Америки – 915 МГц (10 каналов). Отдельным звеном стоит европейский одноканальный поддиапазон на 868 МГц. Скорость передачи данных тут между устройствами напрямую зависит от числа каналов, которые заняты, и колеблется от 20 до 256 кбит/с.

В данной работе исследованы принципы работы технологии ZigBee, а также основанные на этой технологии приёмопередатчики фирмы XBee и Texas Instruments. Показана возможность интеграции модулей в различные устройства для удалённого сбора необходимых данных, обработки информации и управления ими. Среднее потребление модуля составляет примерно 3 мВт*ч при дальности передачи 30-50 м.

Данная технология оправдана и экономически эффективна для таких систем, требующих автономности и относительно невысокой скорости передачи данных, как автоматизированные системы контроля и учёта, систем управления освещением и отоплением и другие

2. Общие сведения о технологии ZigBee и модулях xBee

Сети беспроводной передачи данных в настоящее время пользуются огромной популярностью. Спрос есть во многих IT-зонах по всему миру, а также благодаря использованию беспроводных широкому использованию технологий в корпоративных и частных сетях.

Беспроводные сети могут использоваться в радиовещании, а также использоваться инфракрасной и оптической передачей и лазерами. В этом случае необходимо предоставить стандартную информацию для лицензирования передачи информации по коммерческому каналу.

Принцип работы Bluetooth основан на приёмо-передатчике радиоволн. Вся информация передаётся на мобильные телефоны, персональные компьютеры, клавиатуры, мыши и т.д. в радиусе 10 метров.

Стандартные функции безопасности регламентируются здесь стандартом IEEE 802.11 для компьютеров и для всех точек данных. В этой области (в радиусе 10 метров) есть рабочие станции и точки доступа.

ZigBee – это очень интересный сетевой протокол для данных верхнего уровня. ZigBee (рис. 2.1) фокусируется на таких приложениях, как пересылка малых данных, безопасная для узлов сети, и относительно качественные и многослойные данные, может использоваться для хранения автономных потоков и данных.

Рисунок 2.1 - Модуль ZigBee

Основное различие между технологией ZigBee и остальными заключается в том, что это устройства с низким энергопотреблением, они работают не только с простыми сетевыми топологиями, но также с самоорганизующимиси сетевыми топологиями, чьи функции – исполнять сценарии и маршрутизировать сообщения. Смысл спецификации ZigBee состоит в выборе маршрута в зависимости от требований и состояния сети.

Далее представлены характеристики модуля xBee фирмы Digi:

- Расстояние передачи в помещении, в городе: 29-31 м
- чувствительность приемника: -91 dBm
- расстояние передачи на открытом пространстве: 100м;
- передаваемая мощность: ~1 мВт (0 dBm).

Скорость передачи данных: 250 кбит/с

Сетевые возможности и безопасность:

- Повторение и подтверждение передачи

- DSSS (Прямой Спектр Распространения Последовательности)
- Каждый канал имеет 65 000 уникальных доступных адресов сети
- Поддерживаются топологии «точка-точка», «точка-точкам» и «каждый с каждым».
- Поддержаются режимы Координатора/Конечного устройства
- 128-битовое шифрование (прошивка скоро будет)

Низкое энергопотребление XBee:

- ок TX: 45 мА (3.3 В)
- ток RX: 50 мА (3.3 В)
- ток отключения: < 10 мкА

Технологии ZigBee нашлось применение во многих областях жизни и быта, например, в автоматизации жилья, в промышленных системах управления, в электронике в быту и «периферии» компьютеров.

Самоорганизация и самовосстановление, хорошая устойчивость от помех, малое энергопотребление, топология ячеистой сети, защищённость данных, криптозащита и отсутствие необходимости получения частотного лицензирования приводят к тому, что сеть ZigBee является хорошей основой инфраструктуры беспроводных систем управления.

На рисунке 2.2 показаны топологии Wi-Fi сети, сети Bluetooth и ZigBee. В сетях Bluetooth и Wi-Fi всё взаимодействие между элементами происходит централизованно. И если центральный шлюз выходит из строя, данные не полетят. Также наличие препятствий на пути следования сигнала от устройств к шлюзу играет огромную роль для передачи данных.

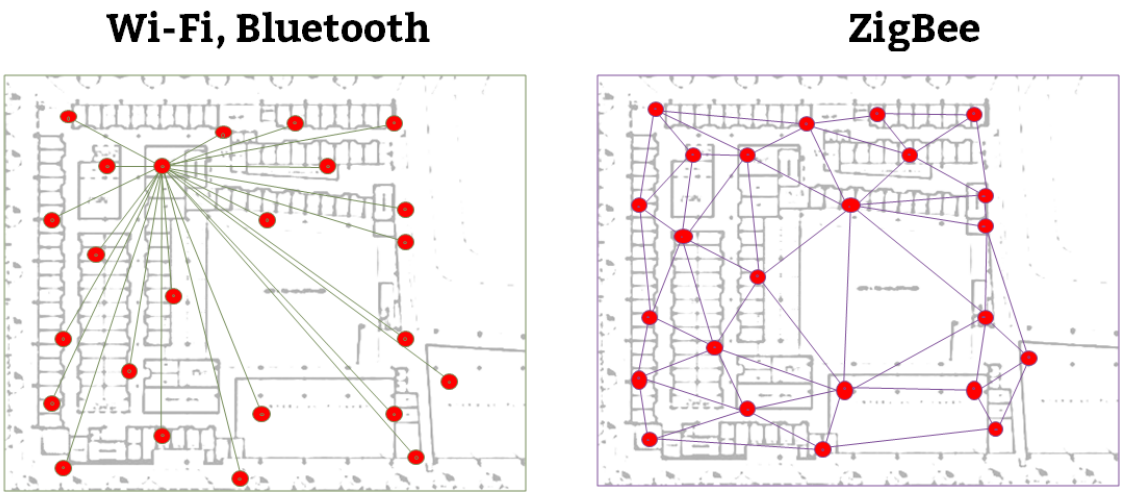

Рисунок 2.2 - Топологии сетей

В ZigBee сетях для построения более устойчивой топологии были созданы избыточные связи между узлами передачи данных. Все устройства, не уходящие в режим сна, здесь являются маршрутизаторами.

Так, при выходе из строя какого-либо узла этой сети, трафик попросту игнорирует этот узел и устремляется к цели через другой узел.

Наряду с этим, препятствия на пути следования сигнала так же не сильно повлияют на доставку данных, хотя и усложнят для внутреннего устройства сети его маршрутизацию.

3. Работа модуля

3.1.Последовательный интерфейс

3.1.1. Передача данных UART

Устройства, имеющие интерфейс UART, могут быть напрямую соединены с выводами модуля.

Модуль управляется CMOS на логических уровнях 2,8–3,4 В. Чтобы подключить модуль к COM-порту ПК, необходим преобразователь уровня COM-USB.

Все данные поступают в модуль дискретными сигналами в виде последовательного кода с уровнем бездействия (Idle), т.е. логическим нулём и логической единицей. Каждый передаваемый информационный байт дополняется двумя битами: стартовым и стоповым, (рисунок 3.1). Здесь младший бит передается первым, т.е. он идёт сразу за стартовым.

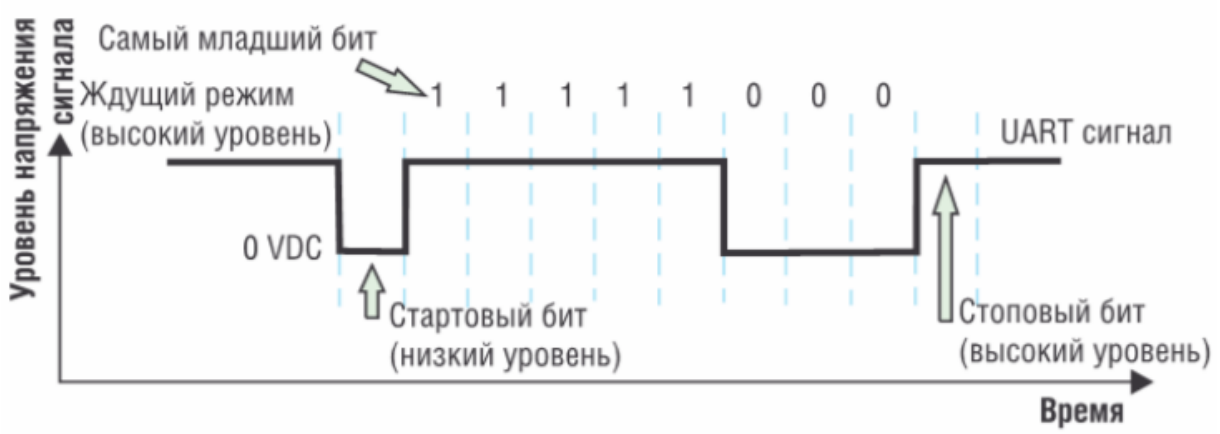

Рисунок 3.1 – Диаграмма передачи данных. Байт 0x1F в модуле XBee

Для успешной передачи данных с помощью последовательного интерфейса оба модуля UART приемника и передатчика должны быть сконфигурированы с одинаковыми параметрами.

3.1.2. Управление передачей данных

3.1.2.1. Буфер обмена

При поступлении последовательных данных в модуль по линии дискретных сигналов, они сохраняются во внутреннем буфере до момента их передачи во внешний эфир. Такая передача откладывается, если в данный момент происходит прием данных.

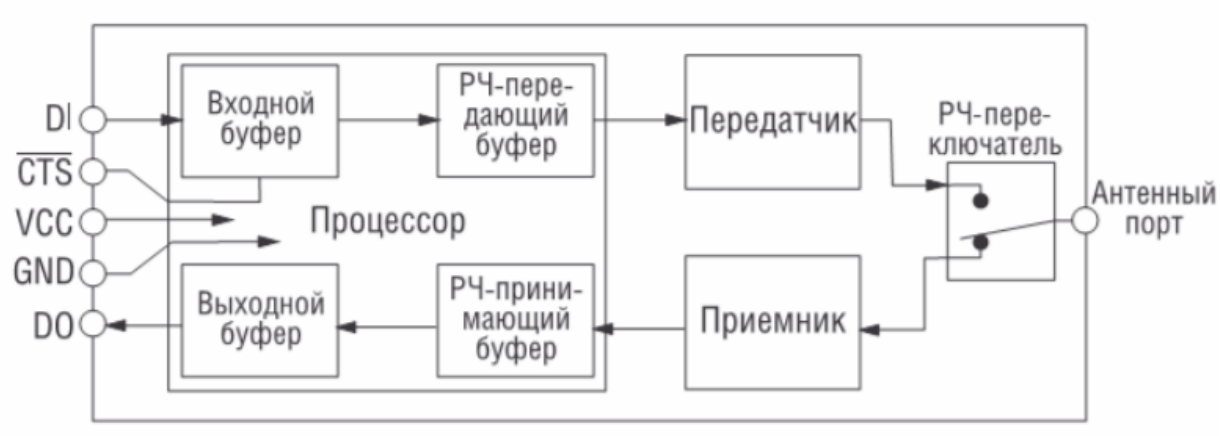

Рисунок 3.2 – Внутреннее устройство модуля XBee

3.1.2.2. Прозрачный режим

Этот режим применяется по умолчанию. В нём все данные, пришедшие на дискретные выводы, стоят в очереди на передачу. Все принятые радиоданные отправляются на вывод дискретных выходов.

3.1.2.3. Режим API

Альтернатива прозрачному режиму. Этот режим расширяет уровень использования модульных сетевых возможностей приложением хоста. В данном режиме все данные, приходящие и уходящие с модуля, содержатся в пакетах, которые описывают операции или события внутри модуля.

Пакеты передачи (принятые на вывод 3) содержат:

• РЧ пакет передачи;

• командный пакет (эквивалент АТ команды);

Принятые пакеты (отсылаются на вывод 2) содержат:

• принятый РЧ пакет;

• события, такие как сброс, ассоциация, выход из ассоциации;

• ответ на команду.

Приложение хоста способно посылать пакеты данных со служебной информацией

3.1.3. Работа в сети 802.15.4

Следующий режимы сети IEEE 802.15.4 доступны для модуля:

• NonBeacon;

Ассоциация (Association) – только в режиме NonBeacon (с координатором). Конечные устройства и координаторы.

Координатор (Coordinator) – центральный модуль устанавливается для синхронизации передачи данных.

Конечное устройство (End Device) – если уже есть координатор. Модули используют координатор для синхронизации и могут переходить в спящий режим.

Персональная сеть (PAN) – содержит одно или более конечных устройств и возможно, координатор.

3.1.3.1. Режим NonBeacon

По умолчанию, XBee/XBee-PRO модули установлены в режим NonBeacon (без координатора). Модули работают в сети «каждый с каждым» (Peer-to-Peer). Это означает, что модули остаются синхронизированными без использования конфигураций владельца/сервера, и каждый модуль в сети выступает роли и ведущего и ведомого. Эта конфигурация по умолчанию подходит для широкого диапазона приложений.

Сеть «каждый с каждым» может быть сконфигурирована путем установки следующих параметров:

• CE=0 – установка модуля как конечного устройства;

• A1=0 – запрет ассоциаций для всех модулей;

• ID и CH – одинаковые во всей сети.

3.1.3.2. Ассоциация

Ассоциация – это объединение координатора и конечных устройств. Ассоциация возможна исключительно в режиме NonBeacon, о котором говорилось ранее. Объединение полезно в сценариях, требующих от центрального устройства передачи сообщений нескольким отдаленным устройствам, назначения им каналов или назначения идентификаторов сети (PAN ID).

Сети передачи данных, состоящие из одного координатора и одного или более конечных устройств, формирует персональную сеть (PAN). Каждое устройство в такой сети имеет идентификатор сети (параметр ID (PAN ID)). Идентификаторы сети должны быть уникальными. Идентификатор сети координатора устанавливается с помощью параметров ID и A2.

Конечное устройство может связаться с координатором, не зная адрес, идентификатор сети или канал координатора. Биты параметра A1 (ассоциация конечного устройства) определяют гибкость конечного устройства в ассоциации. Параметр A1 может использоваться для динамической установки адреса назначения, идентификатора сети и/или канала конечного устройства.

Например, если идентификатор сети координатора известен, но операционный канал нет; в параметре A1 на конечном устройстве должны быть установлены биты

«Auto_Associate» и «Reassign_Channel». Дополнительно, идентификатор сети должен соответствовать идентификатору сети координатора ассоциации.

3.1.4. Режимы функционирования модулей

Модули XBee способны функционировать в пяти различных режимах (рисунок 3.3).

Рисунок 3.3 – Режимы функционирования модуля xBee

3.1.4.1. Режим ожидания

Если с данными ничего не происходит, модуль переходит в режим ожидания. Если данные пересылаются, он переходит в следующие режимы:

• режим передачи;

• режим приема;

• режим сна;

• командный режим.

3.1.4.2. Режимы приема и передачи

Пакеты данных

Каждый пакет радиоданных содержит поля адресации источника и адресата.

Поле адрес назначения складывается из значений параметров DH и DL (Destination Address High и Low). Поля <адрес источника> и <адрес назначения> содержат или короткий 16-битовый адрес или длинный на 64 бита.

Структура пакета данных соответствует спецификациям IEEE 802.15.4.

3.1.5. Прямая и косвенная передача данных

3.1.5.1. Прямая передача

Координатор в режиме NonBeacon может конфигурироваться на использование только прямой передачи, установкой параметра SP (циклический период сна) в «0». Кроме того, координатор в режиме NonBeacon, использующий косвенную передачу вернется в режим прямой передачи, если обнаружит, что модуль предназначения вышел из спящего режима.

Чтобы это обеспечить, параметр ST координатора должен соответствовать значению ST у конечного устройства. Как только оно начнет передавать радиоданные координатору или отсылать запрос на передачу данных по радиоканалу, координатор будет использовать прямую передачу для всех следующих передач по этому адресу модуля, в то время как период ST протекает без активности. Это означает отсутствие приема или передачи сообщений с определенным адресом. Сообщения глобального рода не будут перезагружать таймер бездействия.

3.1.5.2. Косвенная передача

Для настройки косвенных передач в персональной сети (PAN), значение параметра SP координатора должно соответствовать самому большому значению параметра SP любого конечного устройства. Значение параметра SP координатора определяет, как долго координатор сможет сохранить сообщение до того, как удалить его.

В сетях конечные устройства сообщают координатору свой выход из спящего режима, чтобы определить, имеет ли он для них косвенное сообщение. Для спящего режима это делается автоматом после пробуждения модуля. Для режима PinSleep, значение параметра ассоциации конечного устройства должно быть установлено на разрешение пересылать координатору запрос по пробуждении.

3.1.6. Алгоритм передачи данных

До передачи пакета на нужном канале выполняется оценка чистоты канала, чтобы определить, доступен ли этот канал для передачи. Обнаруженная энергия на выделенном канале сравнивается со значением параметра CA. Если обнаруженная энергия превышает значение параметра CA, пакет не передается.

Кроме того, перед передачей вставляется задержка. Эта задержка устанавливается с помощью параметра RN. Если RN установлен в «0», то пред выполнением оценки чистоты канала (CCA) нет задержки. Значение параметра RN - эквивалент параметра minBE в спецификациях IEEE 802.15.4. Последовательность передачи соответствует спецификациями

3.2.IEEE 802.15.4.

3.2.1. Подтверждение передачи

Если передача не производится широковещательным сообщением, модуль будет ожидать подтверждения. Если подтверждение не получено, пакет будет посылаться далее до 3 раз. Если подтверждение не получено после трех передач, будет получен сбой подтверждения передачи.

3.2.2. Конфигурирование модуля

Конфигурирование модуля (изменение параметров) может производиться тремя способами: 1) путем подачи командных последовательностей на соответствующий вывод; 2) путем изменения параметров в программе X-CTU (вкладка Modem Configuration); 3) путем ввода АТ-команд в программе X-CTU (вкладка Terminal).

Каждый модуль должен иметь одинаковую версию встроенного ПО (firmware).

4. Общие сведения о технологии ZigBee и модулях cc2530.

4.1.CC2530 - ZigBee-трансивер.

4.1.1. Основные характеристики

Основное различие между технологией ZigBee и остальными заключается в том, что это устройства с низким энергопотреблением, они работают не только с простыми сетевыми топологиями, но также с самоорганизующимиси сетевыми топологиями, чьи функции – исполнять сценарии и маршрутизировать сообщения. Смысл спецификации ZigBee состоит в выборе маршрута в зависимости от требований и состояния сети.

Характеристики модулей cc2430 и cc2530 приведены на рисунке ниже.

Параметр	CC2430	CC2530
Функции		
Микроконтроллер	8051-совместимый	8051-совместимый
Флэш-память, кбайт	до 128	до 256
Оперативная память, кбайт	8 (<4 в режимах PM2/3)	8 во всех режимах
Контроль сигнала таймера	нет	есть
Каналов таймера	3	5
MAC, размер таймера	16-битный, 20-битное переполнение	16-битный, 24-битное переполнение
Частота ядра, МГц	32	32
Корпус, мм	7x7, 48 выводов	6x6, 40 выводов
Рабочий температурный диапазон, °C	-40. +85	-40. +125
Характеристики радиотракта		
Чувствительность, дБм	-92	-97
Максимальная Tx энергия, дБм	0	+4,5
Чувствительность, дБ	92	101,5
Подавление соседнего канала частотой - 5 МГц	30	49
Подавление соседнего канала частотой +5 МГц	41	49
Подавление соседнего канала частотой -10 МГц	53	57
Подавление соседнего канала частотой +10 МГц	55	57
Питание		
Рабочее напряжение, В	2-3,6	2-3,6
Rx ток, мА	27	24
Tx ток (0 дБм), мА	27	29
Tx ток (+4,5 дБм), мА	-	34
CPU активный ток (32 МГц), мА	10,5	6,5
PM1 ток, мкА	190	200
PM2 ток, мкА	0,5	1
PM3 ток, мкА	0,3	0,4
PM1 -> активный режим, мкс	4	4
PM2/3 -> активный режим, мс	0,1	0,1
Xtal, время запуска, мс	0,5	0,3

Рисунок 4.1 – Характеристики модулей

Модуль cc2530 для того, чтобы выбор топологии сетевой структуры был наиоптимальнейшим, поддерживает следующее бесплатное ПО:

- ПО Z-Stack TM для ZigBee приложений;
- протокол RemoTITM для удалённого управления ZigBee RF4CE;
- протокол SimpliciTITM для сетевых приложений нативных решений.

Особенности cc2530:

- Модуль аппаратного кодирования AES-64/128,
- 48 дБ подавление помех на соседнем канале,

- До 256 кб флеш-памяти/8кб оперативной памяти,
- Отличный энергетический потенциал радиоканала (101.5 дБ),
- 21 порт GPIOs, 2 порта USARTs, большое количество устройств,
- Расширенный температурный диапазон -40...125 °C,
- Полная совместимость с расширителями диапазонов CC259x.

ZigBee сеть, основанная на этих устройствах, может состоять из трех видов ролей:

Coordinator — это тот, кто управляет сетью. Именно это устройство самое главное. Но его наличие в ZigBee сети необязательно.

Router — именно роутеры и организуют ZigBee сеть. Они берут на себя самую сложную часть работы. Вот они и являются самой сложной частью в разработке. Наличие их в сети обязательно. Собственно вся сеть может состоять только из одних роутеров. Роутером может быть только CC2530F256.

End point — это конечное устройство. То, чем и предполагается управлять или получать данные. Наличие их необязательно. Т.к. с этой ролью запросто справится любой роутер.

4.1.2. Частота и полосы пропускания

Стандартом ZigBee выделен специальный диапазон частот для радиопередачи. В диапазоне частот от 2,4 до 2,48 ГГц здесь достигается лучшая помехоустойчивость и самая высокая скорость В нём предусмотрено 16 каналов по 5 МГц.

4.1.3. Протоколы и стандарты

Протоколы, продиктованные стандартами ZigBee и IEEE 802.15.4, обеспечивают надёжность беспроводной сети.

Стандарт IEEE 802.15.4 определяет уровень железа, в то время как спецификация и лицензирование ZigBee выделяет определения уровней приложений и сетевого окружения. На рисунке приведён стек протоколов ZigBee.

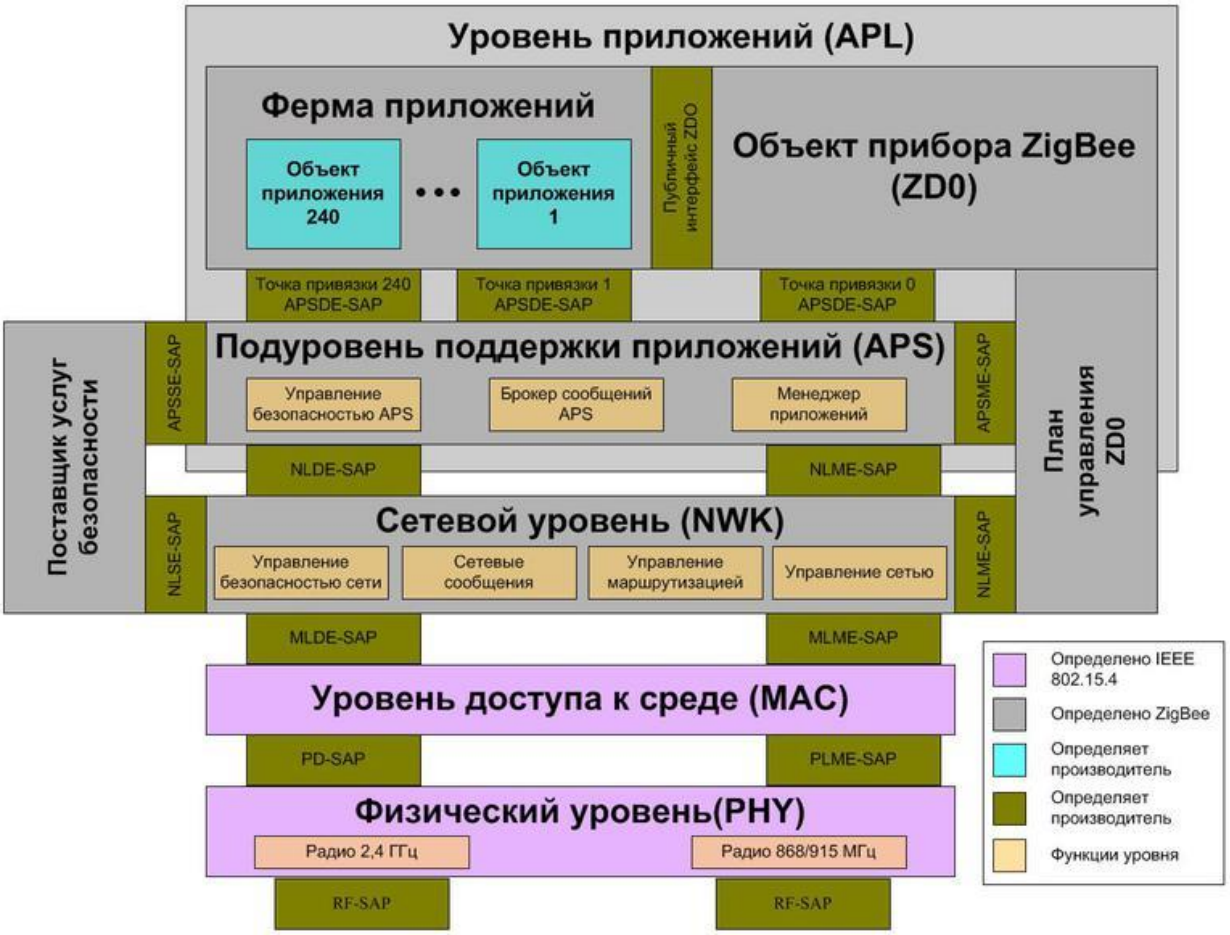

Рисунок 4.2 – Протоколы и стандарты ZigBee

4.1.4. Необходимые компоненты и программное обеспечение для работы

Для работы с этим модулем понадобится несколько устройств и программ:

- CC-DEBUGGER – устройство для отладки и программирования CC2530 (рисунок 4.3).

Рисунок 4.3 - CC-DEBUGGER

- Специальный комплект для работы с CC2530, который, возможно, облегчит работу с модулями (рисунок 4.4)

Рисунок 4.4 – Специальный комплект

- Сам модуль CC2530 (рисунок 4.5);

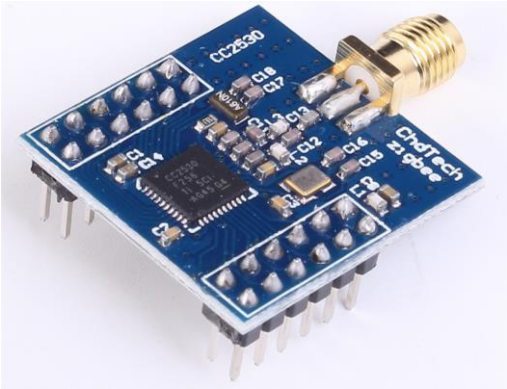

Рисунок 4.5 – Внешний вид беспроводного модуля на базе CC2530

- IAR Embedded Workbench –приложение для редактирования кода программы, прошиваемое в модуль (*30 дней пробный период*)
- Z-Stack 3.0 – совместимое с ZigBee приложение
- SmartRF™ Studio – программа для начальной настройки микросхемы CC2530
- SmartRF Flash Programmer – программа для прошивки модуля CC2530 через CC-DEBUGGER

4.2.Порядок исследования

4.2.1. Прошивка модуля на версию ZNP CC2530

Для прошивки модуля используем специальную плату и CC-Debugger и подключаем их к компьютеру, используя USB-порты. При первом подключении CC-Debugger начнет установку драйверов, что займет несколько минут и затем затребует перезагрузку компьютера. После этого светодиод на CC-Debugger должен загореться зеленым, что означает нормальную работу модуля, в противном случае в соответствии с цветом светодиода потребуется решить проблему (в основном просто перезагрузить устройство).

Далее с помощью программы SmartRF Flash Programmer прошиваем модуль на специальную прошивку с ZNP, находящуюся обычно на месте:

C:\ Texas Instruments\ Z-Stack 3.0.1\ Projects\ zstack\ ZNP\ CC253x\ dev\ CC2530ZNP-with-SBL.hex. Выбираем пункт System-on-Chip, ставим отметку “Erase, program and verify”, после этого ждем некоторое время.

4.2.2. Настройка модуля на модифицированную версию ZNP CC2530, способную посылать данные по UART.

Для того, чтобы модуль мог принимать информацию по UART, используем следующее подключение: P0_2 – RX, P0_3 – TX, P2_0 – CFG1. Также мы использовали модуль прошивки Xbee для создания связи с компьютером. Подключение разъемов: P0_2 (CC2530) к TX (Xbee), P0_3 (CC2530) к RX (Xbee), P2_0 через резистор номиналом 4,7 кОм к земле.

С помощью программы IAR Embedded Workbench открываем файл C:\Texas Instruments\Z-Stack 3.0.1\Projects\zstack\ZNP\CC253x\znp.eww. Затем выбираем пункты CC2530 и ZNP-with-SBL.

Открываем пункт Tools\znp.cfg, в него добавляем некоторые параметры:

-DBDB_FINDING_BINDING_CAPABILITY_ENABLED=0
-DTC_LINKKEY_JOIN
-DNWK_MAX_DEVICE_LIST=10
-DZDSECMGR_TC_DEVICE_MAX=30
-DDISABLE_GREENPOWER_BASIC_PROXY
-DPOWER_SAVING
-DFEATURE_SYSTEM_STATS
-DMT_GP_CB_FUNC
-DMT_SYS_FUNC
-DMT_UTIL_FUNC
-DMT_ZDO_FUNC
-DASSERT_RESET
-DFAKE_CRC_SHDW

-DZNP_ENABLED

-DHAL_SPI=FALSE

-DHAL_UART=TRUE

-DNV_RESTORE

-DZTOOL_P1

В пункте App\znp_app.c ставим в параметре uartConfig.flowControl = FALSE для отключения контроля потоком в модуле.

В пункте ZMain\OnBoard.c выставляем значение znpCfg1 = ZNP_CFG1_UART для того, чтобы модуль использовал эту конфигурацию при работе.

После этого используем функцию программы «Rebuild All» при нажатии правой кнопки мыши на поле CC2530-ZNP-with-SBL, ждем, пока пройдет процесс перекомпиляции прошивки (*необходима хотя бы пробная лицензия*). Затем, аналогично пункту 3.1, прошиваем модуль на модифицированную прошивку. Для связи с CC2530 используем какую-либо программу для просмотра данных, приходящих на USB-порт, например RealTerm. В ней выбираем номер COM-порта, baudrate 9600, data size 8, 1 stop bit, no flow control, отображение данных в формате hex. После всего этого при нажатии на специальной плате физической кнопки перезагрузки “RES”, через некоторое время отображается сообщение в формате hex: 00 FE 06 41 80 01 02 00 02 07 01 C0. К сожалению, понять, что означает это сообщение, нам пока не удалось. На англоязычном форуме есть одно сообщение о том, что это означает то, что модуль cc2530 готов к работе.

Либо же мы можем прошить модуль стандартными утилитами компании IAR. Об этом ниже.

4.3.Прошивка модуля средствами IAR

Прошивка модуля зависит от задачи устройства и его вида (координатор, роутер или конечное устройство).

Прошивка модуля осуществляется с помощью программы SmartRF Flash Programmer и отличается для каждого вида устройства. Прошивка определяет конфигурационные данные работы модуля и имеет шестнадцатеричную кодировку, и, соответственно, расширение .hex.

Генерируется прошивка в программе IAR EW8051, после чего загружается в модуль, занимая всю его флэш-память.

В сети на разных сайтах гуляет большое количество написанных “на коленке” прошивок для всех типов устройств, стоит только поискать.

При помощи координатора Zigbee-сети на базе Texas Instruments SoC cc2530 (и другими) есть возможность создать собственную сеть, в которую подключаются zigbee-устройства. Взаимодействуя напрямую с координатором сети, драйвер позволяет управлять устройствами без дополнительных шлюзов/бриджей от производителей устройств (Xiaomi/TRADFRI/Hue).

5. Системы-аналоги

Над объединением этих устройств бьются многие производители, однако самыми успешными в этом плане являются проекты:

- OpenHab
- MajorDomo
- IOBroker
- Home Assistant
- Domoticz

5.1.MajorDoMo

Первые модули системы были написаны в 2012 году Сергеем Джейгало из Беларуси.

Система является одной из самых популярных в СНГ (и в целом среди русскоязычных пользователей) - во многом благодаря русскоязычному создателю.

Небольшая часть пользователей зарегистрирована в системе MajorDoMo Connect.

Система написана преимущественно на php

5.2.OpenHab

Одна из самых известных в Германии (и в целом в Европе - да и очень известная во всем мире) систем.

Система начала разрабатываться в 2010 году Kai Kreuzer из Германии, который являлся и является сотрудником департамента Умный дом компании Deutche Telekom.

Основным языком разработки является Java

5.3.ioBroker

Система является одной из самых молодых - запущена в 2015 году.

Но тем не менее у платформы достаточно много поклонников (особенно в Германии - откуда система родом) - во многом благодаря тому что ioBroker является преемником популярной в Германии платформы для домашней автоматизации CCU.io

5.4.Domoticz

Разработка системы ведется с конца 2012 года

Основным языком разработки является C++

Система в 1-ю очередь популярна в Испании - благодаря испаноговорящим основателям.

5.5.Home Assistant

Основным языком разработки является Python.

Описание систем будет постепенно расширяться.

Составление рейтинга систем «Умный дом» ввиду огромного количества критериев – невероятно объёмная задача, поэтому в данной работе представлено сравнение с наиболее схожими из них, а ниже приведена часть всех существующих на настоящий момент фирм, оборотный капитал которых в год свыше $1.000.000:

- Ago Control;
- Calaos;
- Domoticz;
- DomotiGa;
- FHEM;
- Freedomotic;
- Home Assistant;
- HomeGenie;
- Homeseer;
- Homey;
- HoMIDoM;
- Indigo Domotics;

- ioBroker;
- Jeedom;
- MajorDoMo;
- Misterhouse;
- MyController;
- myHouse;
- MyNodesNET;
- OpenHAB;
- openLuup;
- openMotics;
- PiDome;
- Pimatic;
- Vera;
- XTension и другие.

5.6.Особенности работы платформ «Умный дом»

Вариантов работы модульной платформы существует несколько:

- автономная работа в изолированной системе, состоящей из нескольких модулей, сопряжённых между собой. Для этого пункта необходима программная прошивка, которая разработана для модулей и содержит исключительно внутренние функции, необходимые для функционирования системы;
- работа в составе программной системы с выходом на внешний сетевой протокол. Для этого необходимо предусмотреть вывод информации и

обмен данными между системами, не входящими в число модулей и вне формата ZigBee;

- работа с сетевыми платформами автоматизации (в т.ч. платформами "Умного дома"). Для этого пункта код программ должен включать в себя типовые структуры, включённые в платформы автоматизации.

Каждый из них стремится выделиться из общей массы, и все кричат о своей уникальности. Все проекты являются OpenSource проектами, т.е. с открытым исходным кодом и присоединиться может любой желающий.

Все они по-своему хороши, однако остановиться поначалу стоит на одном единственном, ~~а на каком, вы узнаете в следующей серии~~, ioBroker.

ioBroker – это OpenSource проект, который разрабатывается сообществом информатиков. Каждый, кто интересуется темой Smart Home, может присоединиться к проекту и под лицензией MIT на Github начать разрабатывать приложения. Дополнительно имеется обширный форум для конечных пользователей, в котором активно обсуждаются новые идеи, проблемы и пожелания клиентов. Опытные разработчики, некоторые из которых имеют 17 лет опыта работы в промышленной автоматизации на ведущих немецких фирмах консультируют по вопросам автоматизации дома, и её внедрения. Проверенные и отработанные идеи размещаются на так называемом Trello-Whiteboard, это доска с текущими, актуальными заданиями, таким образом любой может на неё заглянуть и быть в курсе происходящего.

На рисунке ниже представлен график прироста комьюнити данного проекта.

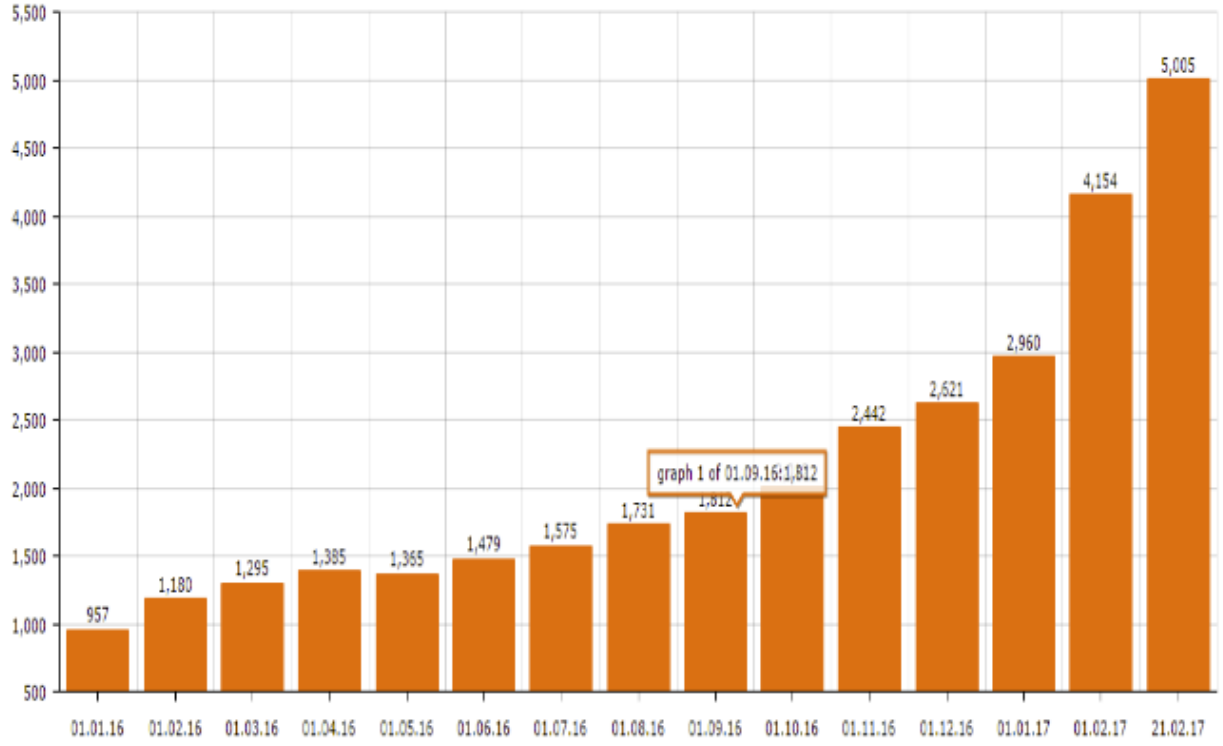

Рисунок 5.1 – Диаграмма роста участников проекта

Для отладки компонентов системы и исследования важных (если не ключевых) параметров системы, таких как помехоустойчивость, надёжность, пропускная способность, время дискретизации, долговечность, время работы и многих других для модульных решений на данном этапе не требуется наличие внешнего сетевого подключения, а обойтись можно минимальной работающей системой, т.е. сетью из трёх узлов.

6. Разработка собственного образца сетевой системы

6.1. Работа с уровнем аппаратных абстракций

6.1.1. Общие сведения о HAL

Hardware Abstraction Layer (HAL, Слой аппаратных абстракций) — это слой абстрагирования, который реализован в ПО, находящемся между программным обеспечением на компьютере и физическим уровнем аппаратного обеспечения. Слой аппаратных абстракций нужен, чтобы скрывать различия в аппаратном обеспечении устройства от основной части ядра ОС, то

есть он нужен затем, чтобы у большей части кода не было необходимости в изменении при запуске на системах с разным аппаратным обеспечением.

То есть, простыми словами, HAL обеспечивает единые обозначения для всех аппаратных единиц в системе и, по обыкновению, он един для всех обозначений в пределах одного стандарта.

Имена функций HAL в стеке протоколов ZigBee имеют следующие префиксные соглашения:

- halCommon: API, который используется стеком EmberZNet и также может вызываться из приложения. Этот API должен быть реализован в обязательном порядке. Пользовательские приложения могут изменять реализацию API, но его функциональность должна оставаться неизменной;
- hal: API, который используется экземплярами приложений. Пользовательские приложения могут удалить этот API или изменить его реализацию по своему усмотрению;
- halStack: API, используемый только стеком EmberZNet. Этот API должен быть реализован и не должен вызываться напрямую из любого приложения. Пользовательские приложения могут изменять реализацию API, но его функциональность должна оставаться неизменной;
- halInternal: API, внутренний для HAL. Стек и приложения EmberZNet никогда не должны вызывать этот API напрямую. Пользовательские приложения могут изменять этот API по своему усмотрению. Однако будьте осторожны, чтобы не повлиять на функциональность любых API-интерфейсов halStack или halCommon.

Стоит заметить, что HAL – это даже более низкий уровень в языках программирования (ЯП), нежели API. HAL взаимодействует с аппаратным обеспечением, а не с ядром системы, то есть он требует меньше процессорного

времени, чем API. HAL и API часто используются в языках высокого уровня, чтобы взаимодействовать с компонентами более низкого уровня.

ОС, имеющие HAL, проще применять на различном оборудовании. Это становится важным для тех систем, которым необходимо работать на большом количестве различных платформ.

6.1.2. HAL в cc2530

В общем и целом под HAL в модулях cc2530 называется способ абстрагироваться от реализации оборудования. В коде приложения используются функции и макросы, которые работают с абстракциями типа *Кнопка 1* или *Светодиод 2*, а конкретное соответствие оборудования и абстракций задается отдельно.

Разберемся что за HAL_LED_2 и как понять, на какой пин он подвешен.

Поиском находим файл *hal_led.h*, где описаны эти константы и функция HalLedSet, в куда передается номер светодиода и режим. Внутри вызывается функция HalLedOnOff для включения и выключения светодиода, которая в свою очередь выполняет либо HAL_TURN_ON_LED2 либо HAL_TURN_OFF_LED2.

HAL_TURN_ON_LED2 и HAL_TURN_OFF_LED2 — это макросы, описанные в *hal_board_cfg.h*. В зависимости от конфигурации оборудования макросы меняются. В нашем случае:

```
#define HAL_TURN_OFF_LED2()       st( LED2_SBIT = LED2_POLARITY (0); )

#define HAL_TURN_ON_LED2()        st( LED2_SBIT = LED2_POLARITY (1);
```

Чуть выше в файле описаны соответствия LED2_SBIT и LED2_POLARITY:

```
/* 2 - Red */
```

#define LED2_BV BV(1)

#define LED2_SBIT P1_1

#define LED2_DDR P1DIR

#define LED2_POLARITY ACTIVE_HIGH

Это означает, что светодиод 2 у нас располагается на пине P1_1 и его уровень включения — высокий. Но, судя по коду, светодиод должен был погаснуть при нажатии на кнопку, а у нас он остается гореть. Если в этом файле *hal_board_cfg.h* поменяем:

#define LED2_POLARITY ACTIVE_HIGH

на *#define LED2_POLARITY ACTIVE_LOW*

Теперь светодиод гаснет при нажатии на кнопку S2, как и должно быть по логике.

Чтобы не менять общие файлы, не относящиеся к нашему приложению, лучше сделать иначе: создадим копию файла *hal_board_cfg.h* (из папки Z-Stack 3.0.2\Components\hal\target\CC2530EB\) в папку Source и назовём его например *hal_board_cfg_DIYRu.h*

Сделаем так, чтобы наша копия файла подключалась самая первая (тем самым исключив из подключения общего файла). Создадим в нашей папке Source файл *preinclude.h* и запишем туда строку:

#include "hal_board_cfg_DIYRuZRT.h"

Укажем подключение этого файла самым первым — в настройках проекта:

$PROJ_DIR$\..\Source\preinclude.h

Выглядеть это будет как на рисунке ниже.

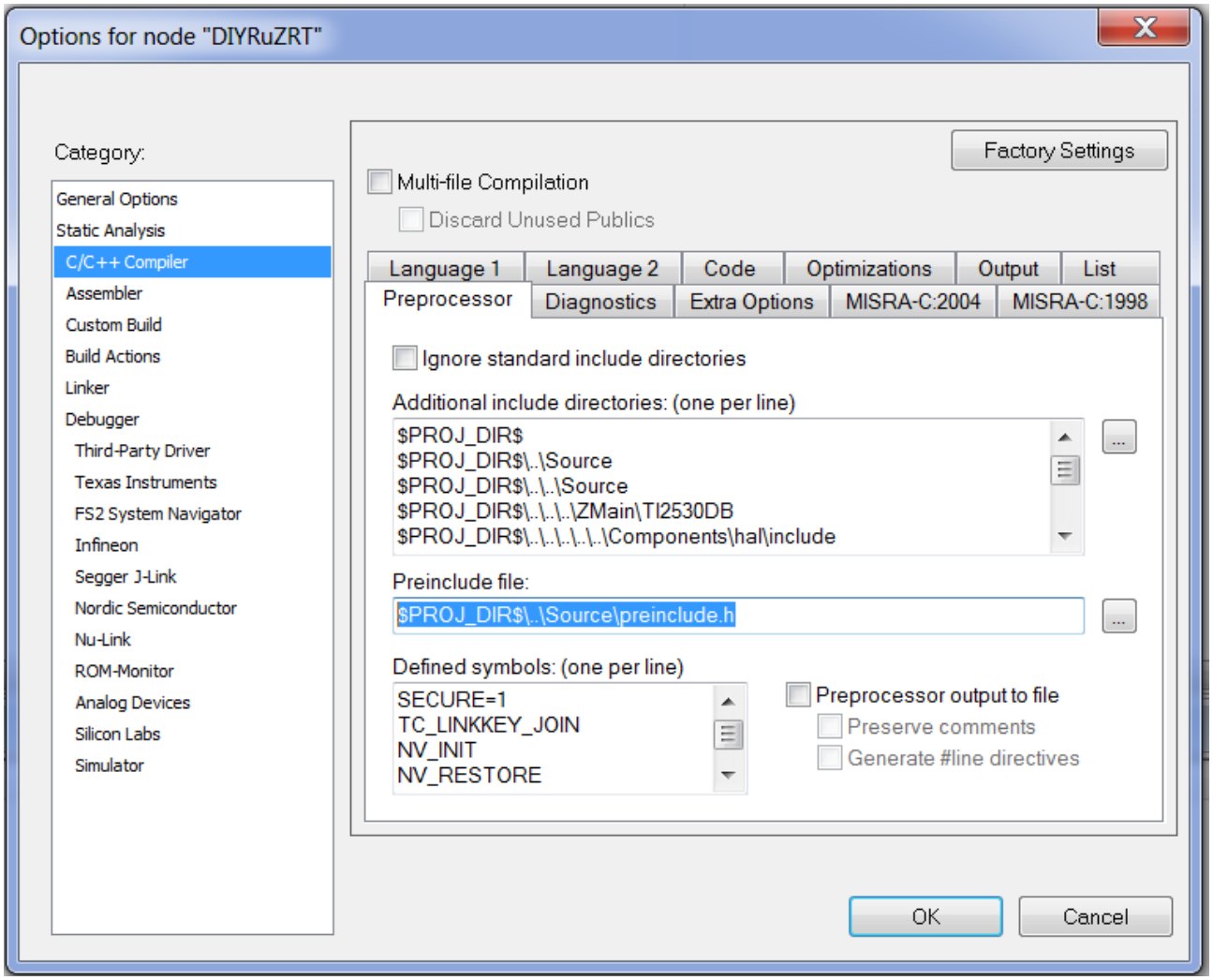

Рисунок 6.1 – Опции для ноды DIYRu.h

Теперь можем менять параметры оборудования в нашем файле hal_board_cfg_DIYRuZRT.h и в файле preinclude.h без необходимости править общие файлы.

В этот же файл preinclude.h я перенес директивы компилятора и удалил их в Options компилятора:

#define SECURE 1

#define TC_LINKKEY_JOIN

#define NV_INIT

#define NV_RESTORE

#define xZTOOL_P1

#define xMT_TASK

#define xMT_APP_FUNC

#define xMT_SYS_FUNC

#define xMT_ZDO_FUNC

#define xMT_ZDO_MGMT

#define xMT_APP_CNF_FUNC

#define LEGACY_LCD_DEBUG

#define LCD_SUPPORTED DEBUG

#define MULTICAST_ENABLED FALSE

#define ZCL_READ

#define ZCL_WRITE

#define ZCL_BASIC

#define ZCL_IDENTIFY

#define ZCL_SCENES

#define ZCL_GROUPS

В том же файле *hal_board_cfg_DIYRuZRT.h* найдём описание кнопки S1 и Joystick Center Press:

/ S1 */*

#define PUSH1_BV BV(1)

#define PUSH1_SBIT P0_1

/ Joystick Center Press */*

#define PUSH2_BV BV(0)

#define PUSH2_SBIT P2_0

#define PUSH2_POLARITY ACTIVE_HIGH

Это соответствует пинам кнопок на плате.

Посмотрим на инициализацию оборудования — макрос HAL_BOARD_INIT в этом же файле.

По умолчанию здесь включается директива HAL_BOARD_CC2530EB_REV17, поэтому смотрим соответствующий вариант макроса.

/ ---------- Board Initialization ---------- */*

*#if defined (HAL_BOARD_CC2530EB_REV17) && !defined (HAL_PA_LNA) && *

*!defined (HAL_PA_LNA_CC2590) && !defined (HAL_PA_LNA_SE2431L) && *

!defined (HAL_PA_LNA_CC2592)

#define HAL_BOARD_INIT()

{

uint16 i;

SLEEPCMD &= ~OSC_PD; / turn on 16MHz RC and 32MHz XOSC */*

while (!(SLEEPSTA & XOSC_STB)); / wait for 32MHz XOSC stable */*

asm("NOP"); / chip bug workaround */*

for (i=0; i<504; i++) asm("NOP"); / Require 63us delay for all revs */*

CLKCONCMD = (CLKCONCMD_32MHZ | OSC_32KHZ); / Select 32MHz XOSC and the source for 32K clock */ *

while (CLKCONSTA != (CLKCONCMD_32MHZ | OSC_32KHZ)); / Wait for the change to be effective */ *

SLEEPCMD |= OSC_PD; / turn off 16MHz RC */*

/ Turn on cache prefetch mode */*

PREFETCH_ENABLE();

HAL_TURN_OFF_LED1();

LED1_DDR |= LED1_BV;

HAL_TURN_OFF_LED2();

LED2_DDR |= LED2_BV;

HAL_TURN_OFF_LED3();

LED3_DDR |= LED3_BV;

HAL_TURN_OFF_LED4();

LED4_SET_DIR();

\

/* configure tristates */ \

```
  P0INP |= PUSH2_BV;
}
```

Именно в этом макросе инициализируются режимы и регистры процессора.

Вместо LED2_DDR и других переменных будет подставлен P1DIR — это регистр порта P1, который отвечает за режим работы пинов (ввод или вывод). Соответственно LED2_BV — это установка в значения 1 в бит соответствующего пина (в нашем случае в 1й бит, что соответствует пину P1_1):

Регистры и режимы процессора описаны в документации «cc253x User's Guide». Однако нигде не видно настройки кнопок. Кнопки обрабатываются аналогично, но в другом файле — *hal_key.c*.

В нем определены параметры работы кнопок и функции HalKeyInit, HalKeyConfig, HalKeyRead, HalKeyPoll. Эти функции отвечают за инициализацию подсистемы работы с кнопками и считывания значений.

По умолчанию обработка кнопок выполняется по таймеру, каждые 100мс. Пин P2_0 для текущей конфигурации назначен на джойстик и его текущее состояние считывается как нажатие — поэтому запускается таймер мигания светодиодом.

6.2.Отладка системы «Кородинатор – Роутер»

Тут кодешник координатора с описанием работы и роутера, и как они спариваются (видео).

6.3.Разработка системы из двух модулей (архитектура точка-точка)

Программный код в случае развёртки сети без внешнего сетевого подключения будет использовать исключительно внутренние функции модулей и микропроцессоров, задействующих вычислительные способности модулей, радиопередатчики и радиоприёмники, систему GPIO, а также систему, предусматривающую развёртку стека протоколов ZigBee.

Помимо программного кода, написанного на языке C, прошивка модуля содержит минимальный набор параметров и адресации для правильной работы и развёртки сети в стеке протоколов ZigBee.

Программный код представляет набор команд для управления процессом включения входов/выходов или системных светодиодов ввиду упрощения и наглядности, и содержится в файле с расширением .c или .cpp. Файл с расширением .h содержит в себе заголовок (шапку) проекта, необходимые параметры для работы программы и для функционирования модуля.

Есть два пути (по факту больше, но это уже всевозможные костыли) для прошивки модуля нужным программным кодом:

1) непосредственная разработка и загрузка кода в память микроконтроллера из программной среды, разработанной заводом-изготовителем (в нашем случае это Texas Instruments, а программа, о которой говорилось ранее: IAR Embedded Workbench 8051);

2) разработка кода в любом приложении, позволяющем создание и компилирование в нужный формат программ (в нашем случае очень удобно пользоваться опять же заводской программой, которая предусматривает всевозможные варианты компиляции под различные типы ядер и микропроцессоров SmartRF Studio) и дальнейшая загрузка программы в микроконтроллер софтом для прошивки оных (очень удобно пользоваться в данном случае специально проработанной для таких случаев программой

SmartRF Flash Programmer, позволяющей делать тонкие настройки для подобных модулей).

Когда код загружен в память всех модулей, проверяем их работоспособность. Работоспособность збс, при запуске системы модули определяют друг друга за несколько миллисекунд и работают как резервные копии друг друга при отключении одного из них.

Ниже представлены последовательность действий и некоторые комментарии по разработке программного кода, приведённого в приложениях 1 и 2.

Первым делом инициализируем необходимые библиотеки компонентов, нужные нам для работы с модулями:

```
#include <hal_led.h>
#include <hal_assert.h>
#include <hal_board.h>
#include <hal_int.h>
#include "hal_mcu.h"
#include "hal_button.h"
#include "hal_rf.h"
#include "basic_rf.h"
```

Определяем константы, использующиеся в программе:

```
/*
* CONSTANTS
*/
// Параметры

#define RF_CHANNEL 25
// 2.4 GHz канал приёмопередатчика
```

// BasicRF address *definitions*

#define PAN_ID 0x2007 // Pan ID служит для идентификации сети

#define SWITCH_ADDR 0x2520 // Адресация кнопки

#define LIGHT_ADDR 0xBEEF // Адресация лампочки

#define APP_PAYLOAD_LENGTH 1 // Длина строки передачи

#define LIGHT_TOGGLE_CMD 0 // Начальное состояние лампы

// Application states

#define IDLE 0 // Бездействие системы

#define SEND_CMD 1 // Происходит передача

/

*Определяем локальные переменные, использующиеся в программе:

** LOCAL VARIABLES*

**/*

static uint8 pTxData[APP_PAYLOAD_LENGTH];

static uint8 pRxData[APP_PAYLOAD_LENGTH];

static basicRfCfg_t basicRfConfig;

#ifdef SECURITY_CCM

// Security key

static uint8 key[] = {

0xc0, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7,

0xc8, 0xc9, 0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf,

};

#endif

/*

*Основная функция программы идёт ниже:

```
*/
void main(void)
{
   // Конфигурируем базовые функции приёмопередатчика
   basicRfConfig.panId = PAN_ID;
   basicRfConfig.channel = RF_CHANNEL;
   basicRfConfig.ackRequest = TRUE;
#ifdef SECURITY_CCM
   basicRfConfig.securityKey = key;
#endif
   // Инициализируем периферийные устройства
   halBoardInit();
   // Инициализируем протокол HAL
   if(halRfInit()==FAILED) {
     HAL_ASSERT(FALSE);
   }
 // Включаем оповещение о горящем светодиоде
   halLedSet(1);
    // Инициализируем базовые функции приёмопередатчика
   basicRfConfig.myAddr = LIGHT_ADDR;
   if(basicRfInit(&basicRfConfig)==FAILED) {
     HAL_ASSERT(FALSE);
   }
   basicRfReceiveOn();
// (Main loop) Залупиваем основную функцию
   while (TRUE) {
```

```
        while(!basicRfPacketIsReady());
        if(basicRfReceive(pRxData, APP_PAYLOAD_LENGTH, NULL)>0) {
          if(pRxData[0] == LIGHT_TOGGLE_CMD) {
            halLedToggle(1);
          }
        }
      }
    }
```

Ниже представлена часть программного кода, отвечающая за передачу данных с кнопок. Т.к. эти программы лежат в разных директориях, для этой части необходимо так же провести конфигурацию.

```
// Подключаем библиотеки
#include <hal_led.h>
#include <hal_assert.h>
#include <hal_board.h>
#include <hal_int.h>
#include "hal_mcu.h"
#include "hal_button.h"
#include "hal_rf.h"
#include "basic_rf.h"
// Те же константы и параметры приложения
//
#define RF_CHANNEL              25      // 2.4 GHz RF channel
// BasicRF address definitions
#define PAN_ID              0x2007
#define SWITCH_ADDR         0x2520
```

```
#define LIGHT_ADDR          0xBEEF
#define APP_PAYLOAD_LENGTH      1
#define LIGHT_TOGGLE_CMD        0
// Application states
#define IDLE                    0
#define SEND_CMD                1
/*************************************************************************
**********************
* Локальные переменные
*/
static uint8 pTxData[APP_PAYLOAD_LENGTH];
static uint8 pRxData[APP_PAYLOAD_LENGTH];
static basicRfCfg_t basicRfConfig;
#ifdef SECURITY_CCM
// Security key
static uint8 key[] = {
  0xc0, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7,
  0xc8, 0xc9, 0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf,
};
#endif
void main(void)
{
  // Конфигурируем опять идентификационный номер
  basicRfConfig.panId = PAN_ID;
  basicRfConfig.channel = RF_CHANNEL;
  basicRfConfig.ackRequest = TRUE;
#ifdef SECURITY_CCM
```

```
    basicRfConfig.securityKey = key;
  #endif
    // Инициализируем базовый радиоканал
    halBoardInit();
    // Initalise hal_rf
    if(halRfInit()==FAILED) {
     HAL_ASSERT(FALSE);
    }
    // Моргаем светодиодом, что всё включено
    halLedSet(1);
    pTxData[0] = LIGHT_TOGGLE_CMD;
    // Инициализируем базовую радиочастоту
    basicRfConfig.myAddr = SWITCH_ADDR;
    if(basicRfInit(&basicRfConfig)==FAILED) {
     HAL_ASSERT(FALSE);
    }
    // Main loop. Залупиваем функцию передачи
    while (TRUE) {
      if(halButtonPushed()) {
        basicRfSendPacket(LIGHT_ADDR,pTxData,
APP_PAYLOAD_LENGTH);
  // Переводим MCU в спящий режим. Будить можно только прерыванием
        halIntOff();
        halMcuSetLowPowerMode(HAL_MCU_LPM_3);
        // Прерывание
```

```
            halIntOn();
        }
    }
}
```

6.4.Разработка модели сети с тремя узлами (каждый с каждым)

На основе модулей cc2530 компании Texas Instruments была разработана модель системы освещения. Данная модель включает в себя три модуля, связанные между собой программным кодом, дающим возможность связать их в единую сеть посредством радиочастотного канала с частотой 868 МГц и объединённых протоколом ZigBee; исполнительными устройствами сети должны были являться кнопки управления на модулях, передающих сигналы, и реле, управляющие светодиодными лентами на приёмных устройствах, однако ввиду их размера и избыточности с точки зрения исследования параметров работы приёмника, было принято решение заменить светодиодные ленты на внутренние светодиоды, находящихся в составе устройств, а кнопки решено заменить таймерами.

Графически условно сеть можно представить следующим образом (рисунок 6.1):

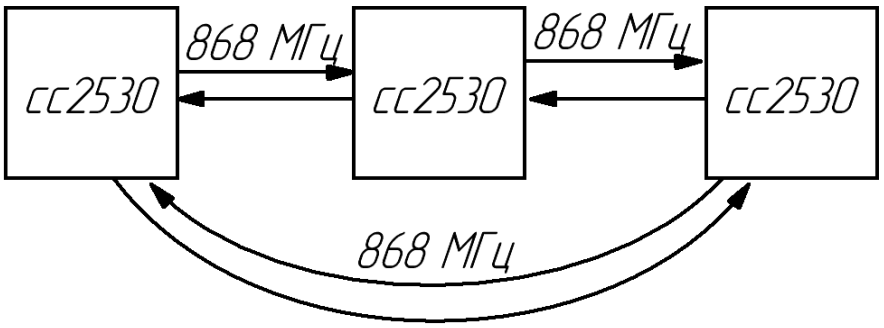

Рисунок 6.1 – Структура сети приёмо-передатчиков cc2530

Здесь каждое устройство является и приёмником и передатчиком, и все они связаны друг с другом.

6.5.Работа при прошивке нескольких модулей по ролям.

Для подключения и прошивки, необходимо присоединить модуль через DEBUG-разьем на плате. Это 10 пинов (2*5) с шагом 1.27 мм. Поэтому, можно приобрести отдельно контакты для подключения к такому разъему, либо взять такую плату с кабелем.

Существует как минимум 2 варианта прошить cc2530:

- используя специализированные устройства, типа CCDebugger;
- используя Arduino-совместимые микроконтроллеры.

6.5.1. Прошивка через CCDebugger

Для подключения CCDebugger используется дополнительная отладочная плата. На ней сразу есть DEBUG-разъем для подключения и готовый UART-USB для работы через компьютер.

Процесс простой: подключаем CCDebugger к плате, плату подключаем через USB к ПК, запускаем Smart RF Flash Programmer, выбираем прошивку для cc2530 и прошиваем.

6.5.2. Подключение к USB

Для подключения такой платы к ПК без отладочной платы потребуется UART-USB кабель с контактами: RX, TX, 3.3v, GND:

RX - P03

TX - P02

3.3v - VCC

GND - GND

Для нормальной работы по UART требуется на плате соединить контакт P20 с GND и выставить скорость порта 115200.

Прошивка модуля зависит от задачи устройства и его вида (координатор, роутер или конечное устройство).

Прошивка определяет конфигурационные данные работы модуля и имеет

шестнадцатеричную кодировку, и, соответственно, расширение hex.

Генерируется прошивка в программе IAR EW8051, после чего загружается в модуль, занимая всю его флэш-память.

В сети на разных сайтах гуляет большое количество написанных “на коленке” прошивок для всех типов устройств.

При помощи координатора Zigbee-сети на базе Texas Instruments SoC cc2530 (и другими) есть возможность создать собственную сеть, в которую подключаются zigbee-устройства. Взаимодействуя напрямую с координатором сети, драйвер позволяет управлять устройствами без дополнительных шлюзов/бриджей от производителей устройств (Xiaomi/TRADFRI/Hue).

Программный код в такой сети можно логически поделить на две части. Первая часть – это код передатчика, и второй – приёмника.

Данный код схож с описанным в предыдущем разделе, однако с существенным отличием, такой код прописывается для всех модулей, различаются лишь каналы для связи, т.е. их адресация, что позволяет универсализировать код для модулей и пихать его в каждый модуль.

Программный код с комментариями для управления приёмником такой сети представлен ниже.

Первое, что необходимо сделать – подключить нужные библиотеки:

```
#include <hal_led.h>
#include <hal_assert.h>
#include <hal_board.h>
#include <hal_int.h>
#include "hal_mcu.h"
#include "hal_button.h"
#include "hal_rf.h"
#include "hal_uart.h"
#include "basic_rf.h"
```

Прописываем необходимые константы

Определяем нужные параметры

Прописываем радио-канал приёмо-передатчика

```
#define RF_CHANNEL              25
```

Устанавливаем базовые настройки для радио-канала (идентификационный номер, адрес передатчика и приёмника, длительность передаваемого сигнала и команду на включение).

```
// BasicRF address definitions
#define PAN_ID              0x2007
#define SWITCH_ADDR         0x2520
#define LIGHT_ADDR          0xBEEF
#define APP_PAYLOAD_LENGTH      1
#define LIGHT_TOGGLE_CMD        0
```

Указываем возможные состояния системы передатчика
Здесь 0 – бездействие, 1 - пересылка

```
#define IDLE                    0
#define SEND_CMD                1
```

Указываем локальные переменные

```
static uint8 pTxData[APP_PAYLOAD_LENGTH];
static uint8 pRxData[APP_PAYLOAD_LENGTH];
static basicRfCfg_t basicRfConfig;
```

Для безопасности сети прописываем защитный ключ

```
#ifdef SECURITY_CCM
// Security key
static uint8 key[] = {
   0xc0, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7,
   0xc8, 0xc9, 0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf,
};
#endif
```

```
/******************************************************************
********************* в этом месте прописываются локальные функции
*при их наличии
* LOCAL FUNCTIONS
*/
```

Теперь входим в основную (главную) функцию и кнофигурируем наш основной радио-канал:

```
void main(void)
{
```

```
  char buf[20];
  // Config basicRF
  basicRfConfig.panId = PAN_ID;
  basicRfConfig.channel = RF_CHANNEL;
  basicRfConfig.ackRequest = TRUE;
#ifdef SECURITY_CCM
  basicRfConfig.securityKey = key;
#endif
```

Инициализируем базовую периферию (внутренний светодиод в нашем случае).

```
  halBoardInit();
```

Инициализируем функции абстрактного уровня железа.

```
  if(halRfInit()==FAILED) {
   HAL_ASSERT(FALSE);
  }
```

Следующей строчкой запускаем индкатор о его успешной работе.

```
  halLedSet(1);
```

Инициализируем базовый радио-канал передачи

```
  basicRfConfig.myAddr = LIGHT_ADDR;
  if(basicRfInit(&basicRfConfig)==FAILED) {
   HAL_ASSERT(FALSE);
  }
  basicRfReceiveOn();
```

Далее идёт тело основного цикла, запущенного на устройстве (в простонародье «мэйн луп» ☺) В нём зациклено условие, которое проверяет возникновение логической единицы по готовности пакета отправки, который каждый цикл проверяется, и в случае поступления единицы, меняет выход отправляемого сообщения на противоположный.

```
  while (TRUE) {
    while(!basicRfPacketIsReady());

    if(basicRfReceive(pRxData, APP_PAYLOAD_LENGTH, NULL)>0) {
      if(pRxData[0] == LIGHT_TOGGLE_CMD) {
        halLedToggle(1);
      }
```

```
        }
    }
}
```

Далее стоит описать часть проекта, который отвечает за персылку пакетов по нажатию кнопки по тем же радио-каналам.

Все базовые настройки сохраняются такие же, как в начале.

Ниже функция, отключающая передатчик, когда он не используется, чтобы минимизировать расход энергии, запасённой в батарее.

```
basicRfReceiveOff();
```

Ниже представлено тело цикла, отвечающего за отправку сообщений по выделенному радиоканалу при нажатии кнопки на плате.

```
while (TRUE) {
  if(halButtonPushed()) {

basicRfSendPacket(LIGHT_ADDR,pTxData, APP_PAYLOAD_LENGTH);
```

Ниже формируем функцию, которая также отключает радиоканал, если не поступает прерываний от кнопки, в противных случаях передатчик находится в режиме сна.

```
        halIntOff();
        halMcuSetLowPowerMode(HAL_MCU_LPM_3); // Will turn on
global
```

Включаем функцию перываний:

```
        halIntOn();
      }
    }
}
```

Код выше формирует каналы передачи, оправку и приём посылок в модулях и позволяет отслеживать их приём с помощью встроенного диода, либо же вывести их на соответствующие выводы.

Этот код, записанный в микроконтроллер cc2530 даёт ему возможность

передавать сигнал по адресу, указанному в соответствующей строчке адресата.

Однако для анализа параметров сети этого недостаточно. Нужен вывод на дисплей программатора или персонального компьютера.

Ниже представлена часть кода, заставляющая микроконтроллер с соответствующим адресным значением на канале получения сигналов, получать сигнал, обрабатывать его, и отдавать команду на включение его системного светодиода.

Все данные мы, благодаря последовательному интерфейсу, получаем на экране компьютера и можем отследить работу осветительных приборов и всей сети в целом.

Часть кода ниже отвечает за вывод информации на встроенный в модуль последовательный порт, физически передающий данные через USB-порт:

```
Устанавливаем направленность сигналов.
  MCU_IO_DIR_OUTPUT(1, 0);
  MCU_IO_SET_LOW(1, 0);

Инициализируем базовый канал радиопередачи.
  basicRfConfig.myAddr = LIGHT_ADDR;
  if(basicRfInit(&basicRfConfig)==FAILED) {
   HAL_ASSERT(FALSE);
  }
  basicRfReceiveOn();

Запускаем чтение сигналов по интерфейсу UART.
  halUartInit(HAL_UART_BAUDRATE_38400, 0);
```

Основной цикл, он запускает слушание поступающих команд и отсылку сигнала в случае получения на выходе логической единицы и запись этой единицы по последовательному порту в программатор.

```
  while (TRUE) {
    while(!basicRfPacketIsReady());

    if(basicRfReceive(pRxData, APP_PAYLOAD_LENGTH, NULL)>0) {
      if(pRxData[0] == LIGHT_TOGGLE_CMD) {
        MCU_IO_TGL(1, 0);
        halUartWrite(pRxData, 1);
```

```
            }
        }
    }
}
```

7. Исследования работы модулей cc2530

7.1.Исследование зависимости сигнала от дальности связи (RSSI)

Показатель уровня принимаемого сигнала, RSSI (англ. received signal strength indicator) (телекоммуникации) — полная мощность принимаемого приёмником сигнала. Измеряется приёмником по логарифмической шкале в дБм (dBm, децибел относительно 1 милливатта) [47].

Тест RSSI был проведён на основе программного кода и аппаратной платформы, описанных выше.

Для теста использовались модуль E18-MS1-PCB в качестве передатчика и 2,4 ГГц Zigbee модуль cc2530 (плата разработчика) в качестве приемника.

В таблице 7.1 представлены условия измерения Значение RSSI.

Таблица 7.1 – Условия измерения RSSI

Условия измерения Значение RSSI	Значение RSSI, дБм
20 см	96-97
1 метр	91-97
2 метра	75-91
4 метра	61-75
3 метра + тонкая стена	72
2 метра + стена	58-60
7 метров	55
6 метров	64-67
6 метров + стена	56
10 + препятствия (стены)	42
11-12 метров + стены	25-31
12+ метров + стены	Нет связи

Сигнал приёмника, выведенный через последовательный порт на монитор компьютера, представляет собой инвертированную шкалу, где RSSI обратно пропорционально зависит от качества сигнала.

Для устройств, работающих по стандартам Wi-Fi и Bluetooth 4.0, RSSI является единственным параметром, позволяющим измерить расстояние от устройства до базовой станции или маяка. Уравнение для вычисления

расстояния (за пределами ближней зоны передатчика) имеет следующий вид[5]:

$$P_d = P_0 - 10 \cdot n \cdot \lg\left(\frac{d}{d_0}\right)$$

где:

- d — расстояние от устройства до передатчика, м;
- d_0 — расстояние от устройства до точки, где выполнялось измерение мощности сигнала устройства, м (выбранное единичное (калибровочное) расстояние, например, 1 м);
- lg — десятичный логарифм;
- P_0 — мощность сигнала устройства, измеренная на единичном расстоянии от устройства, dBm;
- n — коэффициент потерь мощности сигнала при распространении в среде, безразмерная величина (для воздуха ; увеличивается при наличии препятствий);
- P_d — RSSI, dBm.

Данное уравнение следует из формулы передачи Фрииса для распространения радиосигнала в свободном пространстве[1].

Построенный на основе таблицы 7.1. график зависимости качества сигнала от дальности передачи показан на рисунке ниже.

Рисунок 7.1 – Зависимость качества сигнала от дальности передачи

Следуя этой зависимости, можно сказать, что качество сигнала, а стало быть, скорость передачи внутри сети ZigBee, практически линейно (с небольшими искажениями) зависит от дальности передачи.

7.2.Концентрация устройств в помещении

Задачей сети является бесперебойная передача пакета данных из одного узла в другой с незначительным снижением скорости. Логика и предыдущий подраздел подсказывают, что с увеличением дальности устройств друг от друга, то есть уменьшением их количества на единицу объёма, должна уменьшиться скорость передачи данных в сети.

В то же время, при слишком плотном размещении устройств, то есть при превышении какого-то критического количества устройств в сети (или на единицу объёма) количество передаваемой служебной информации будет расти, а это означает, что скорость передачи также будет неумолимо падать.

Весь пакет передачи данных состоит из … бит информации, из них … бит – служебная информация, которая увеличивается в зависимости от количества ретрансляций сигнала при проходе от одного узла к другому.

7.3. Устойчивость к помехам

Так как стек протоколов ZigBee базируется на частоте от 2,4 до 2,48 ГГц, имеет 16 радиочастотных каналов шириной по 5 МГц каждый, его устойчивость к помехам обусловлена их наличием на этих частотах.

Здесь на передачу данных будут влиять помехи сетей Wi-fi и Bluetooth, находящихся в том же диапазоне частот.

7.4. Энергопотребление при длительной работе

Большинство устройств работает по следующему алгоритму: устройство находится в «спящем» состоянии практически все время, обеспечивая оптимальный режим энергосбережения. При поступлении новой информации либо во время очередного сеанса связи оно активизируется, быстро передает данные и снова переходит в режим пониженного энергопотребления. Типовые временные задержки при этом составляют 30 мс для подключения нового устройства к сети, 15 мс для перехода из «спящего» в активное состояние и 15 мс для доступа к каналу. Так удается продлить срок службы батарей до 10 лет и более в зависимости от типа приложения и длительности рабочего цикла, причем ток при передаче может составлять порядка 15-30 мА, а в «спящем» режиме - менее 2 мкА. В результате задержки по отклику настолько малы, что человек, войдя в комнату и щелкнув переключателем беспроводной связи ZigBee, даже не заметит, что свет появился почти мгновенно, в то время как задержки при подключении устройств к сети Bluetooth составляют до трех секунд.

7.4.1. Структура узла и типовой алгоритм модели энергопотребления

Узел беспроводной сенсорной сети представляет собой своего рода микро встроенную систему, которая состоит из четырех частей: сенсорного модуля, процессорного модуля, модуля беспроводной связи и модуля энергоснабжения.

Обычно энергопотребление процессора и модуля датчика является относительно низким, и большая часть энергопотребления происходит в модуле беспроводной связи.

Ввиду объема чувствительных узлов часто невозможно использовать батареи большой емкости для их питания, и из-за характеристик беспроводных сенсорных сетей часто невозможно заменить батареи для чувствительных узлов.

Следовательно, то, как повысить эффективность связи и снизить потребление энергии, является одной из ключевых проблем, решаемых в беспроводных сенсорных сетях.

Типичная модель энергопотребления для беспроводных сенсорных сетей принимается:

$$E_{Tx}\{k\times E_{elec}+k\times\varepsilon_{ft}\times d_2, d<d_{0k}\times E_{elec}+k\times\varepsilon_{mpd4}, d_0.$$

(1)

$$E_{Rx}=k\times E_{elec}.$$

(2)

Формула (1) указывает на потерю энергии при отправке k-битных данных в приемник с расстоянием d, включая две части: потери в цепи излучения и потери на усиление мощности.

E_{elec} - это энергия потерь в передающем контуре, а для потерь при усилении мощности используются разные модели в соответствии с разным

расстоянием между отправителем и получателем: модель свободного пространства и модель замирания при многолучевом распространении.

Когда расстояние передачи находится в пределах определенного порогового значения d_0 (d_0 является константой), когда $d < d_0$, принимается модель свободного пространства, мощность расстояния передачи пропорциональна квадрату расстояния, ε_{ft} - энергия, необходимая для мощности усиление в модели канала в свободном пространстве; когда $d \geq d_0$, принимается модель многолучевого замирания, потребляемая мощность пропорциональна четырехкратному расстоянию, εmp энергия, необходимая для усиления мощности в модели многолучевого канала с замиранием.

Формула (2) указывает энергию, потребляемую приемником при получении k-битных данных.

7.4.2. Алгоритм модели энергопотребления для линейных беспроводных сенсорных сетей

В соответствии со структурой беспроводной сенсорной сети вдоль железной дороги определяется местоположение узлов приемника и чувствительных узлов в беспроводной сенсорной сети, которая развернута в соответствии с фактическими требованиями мониторинга и соответствующими правилами вдоль железнодорожной линии.

Развертывание ретрансляционных узлов в беспроводных сенсорных сетях напрямую связано с временем жизни беспроводных сенсорных сетей.

В целом, ретрансляционные узлы применяют единую стратегию развертывания, то есть все ретрансляционные узлы, расположенные во всех узлах восприятия, развертываются с одинаковым интервалом.

В соответствии с этой стратегией развертывания расстояние между всеми ретрансляционными узлами одинаково, а количество энергии, потребляемой данными одного блока пересылки, равно, но узлам, находящимся

на близком расстоянии от узла приемника, требуется больше пересылки данных, чем тем, которые находятся на большом расстоянии, поэтому легко сформировать горячую зону энергии, что приводит к проблеме «энергетической дыры», из-за которой беспроводная сенсорная сеть преждевременно умирает.

Ввиду вышеупомянутых проблем предложена стратегия развертывания неоднородной оптимизации, и политика описывается следующим образом: между двумя смежными узлами восприятия или узлом приемника и узлом восприятия существует равномерное развертывание, то есть расстояние между два соседних узла ретрансляции равны между двумя смежными узлами восприятия или узлом приемника и узлом восприятия.

Узлы ретрансляции, расположенные ближе к узлу приемника, предполагают большую емкость пересылки данных, и следует принять меньший интервал развертывания; однако узлы ретрансляции, находящиеся на большом расстоянии от узла-получателя, предполагают меньшую емкость пересылки данных, и следует принимать больший интервал развертывания, насколько это возможно, чтобы каждый узел ретрансляции потреблял одинаковую энергию при пересылке отдельных данных.

Следовательно, для двух смежных чувствительных узлов, которые находятся ближе к узлу приемника или между узлом приемника и чувствительным узлом, ретрансляционные узлы должны быть развернуты более плотно.

Когда число ретрансляционных узлов, которые могут быть развернуты, такое же, как и у средней стратегии развертывания, неравномерно оптимизированная стратегия развертывания может сбалансировать энергопотребление всех ретрансляционных узлов в сети, что может продлить жизненный цикл сети и улучшить эффективность сети.

Следовательно, математическая модель, основанная на этой стратегии, может быть создана для решения проблемы общего количества ретрансляционных узлов; количество ретрансляционных узлов должно быть развернуто между двумя соседними чувствительными узлами или узлом приемника и чувствительным узлом, чтобы эффективность сети могла быть максимизирована.

Исследуемая беспроводная сенсорная сеть представляет собой типичную линейную сеть, и для того, чтобы избежать проблемы «энергетической дыры», энергетический баланс всех узлов должен потребляться.

Затем проблема энергопотребления узлов будет подробно проанализирована.

Согласно этой гипотезе может быть известно, что объем данных, собранных каждым узлом датчика в 1 цикле, составляет 1 бит, объем данных, принимаемых каждым узлом в 1 цикле, равен rbit, объем данных, которые каждый узел отправляет в 1 цикле, равен jbit, очевидно, объем данных, полученных и отправленных $l \leq j \leq l + r$ для узлов ретрансляции в 1 цикле, равен jbit.

Поскольку энергия, потребляемая данными зондирования, пренебрежимо мала, известно, что ретрансляционный узел потребляет больше энергии, чем зондирующий узел.

Следовательно, узкое место жизненного цикла беспроводных сенсорных сетей лежит в ретрансляционных узлах.

По расчётным данным, для работы устройства на основе модуля cc2530 с режимом передачи раз в две минуты пакета данных при средней длительности передачи пакета в 10 секунд, получается следующая арифметика:

Модуль работает от аккумулятора и настроен на напряжение 3,3 В, при этом в спящем режиме модуль потребляет 2 мкА, а в режиме передачи усреднённо 30 мА.

То есть за сутки его потребление можно посчитать с большой точностью, перемножив ежесекундное потребление на всё время цикла рассмотрения этой задачи.

По расчёту выходит, что за сутки в спящем режиме модуль тратит 0,5 Вт*с энергии.

При передаче раз в две минуты длительностью импульса передачи в 10 сек модуль потратил 712, 8 Вт*с энергии, что соответствует 0,2 Вт*ч потраченной энергии.

Заключение

К достоинствам технологии ZigBee следует отнести хорошую масштабируемость, возможность самовосстановления в случае сбоев и простоту настройки. Это так, поскольку ZigBee-устройства при включении питания благодаря специальному алгоритму, реализуемому встроенным программным обеспечением, умеют сами находить друг друга и формировать сеть, а в случае выхода из строя какого-либо из узлов умеют устанавливать новые маршруты для передачи сообщений. Таким образом, технология Zigbee может быть использована как для реализации простых соединений «точка-точка» и «звезда», так и для образования сложных сетей с топологиями «дерево» и «ячеистая сеть». При применении 64-разрядной адресации, кстати, свойственной 1-Wire-технологии, в единую сеть могут быть объединены свыше 60 тысяч ZigBee-устройств.

Список используемой литературы

1. Используем ZigBee в Home assistant // indahomekit.ru URL: https://www.indahomekit.ru/2019/01/10/ispolyzuem-zigbee-v-home-assistant/ (дата обращения: 17.05.2019).
2. Анисимов Дмитрий Владимирович Механизм распределенного доступа к среде передачи данных, обеспечивающий стабилизацию пропускной способности на максимальных значениях при высокой нагрузке в сетях стандарта IEEE 802. 11 // T-Comm. 2016. №10. URL: https://cyberleninka.ru/article/n/mehanizm-raspredelennogo-dostupa-k-srede-peredachi-dannyh-obespechivayuschiy-stabilizatsiyu-propusknoy-sposobnosti-na-maksimalnyh
3. Бобин А.Ю., Борисов А.П. Исследование помехозащищенности сетей ZigBee // Информационное пространство в аспекте гуманитарных и технических наук – 2015 / Материалы междисциплинарной межвузовской конференции студентов, магистрантов и аспирантов. Барнаул, 2015. – С.7 – 9
4. Коптев Д.С., Щитов А.Н., Шевцов А.Н. Сравнительный анализ наиболее перспективных стандартов беспроводных сетей связи // Международный журнал гуманитарных и естественных наук. 2016. №1. URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-naibolee-perspektivnyh-standartov-besprovodnyh-setey-svyazi
5. Кукунин Сергей Валерьевич, Лысенков Николай Александрович Система дистанционного управления техническими устройствами на основе технологии ZigBee // АСУ и приборы автоматики. 2010. №152. URL: https://cyberleninka.ru/article/n/sistema-distantsionnogo-upravleniya-tehnicheskimi-ustroystvami-na-osnove-tehnologii-zigbee
6. Легков К. Е. Методы повышения производительности беспроводных mesh-сетей специального назначения // T-Comm. 2017. №3. URL: https://cyberleninka.ru/article/n/metody-povysheniya-proizvoditelnosti-

besprovodnyh-mesh-setey-spetsialnogo-naznacheniya-1

7. Самойлов Дмитрий Олегович, Семенов Валерий Дмитриевич, Упаев Антон Борисович, Федотов Владимир Александрович, Беспроводная система охранно-пожарной сигнализации с использованием технологий ZigBee и GSM // Доклады ТУСУР. 2011. №2-2 (24). URL: https://cyberleninka.ru/article/n/besprovodnaya-sistema-ohranno-pozharnoy-signalizatsii-s-ispolzovaniem-tehnologiy-zigbee-i-gsm
8. Соколов Михаил, Воробьев Олег, Реализация беспроводных сетей на основе технологии ZigBee стандарта 802.15.4 // Компоненты и Технологии. 2005. №46. URL: https://cyberleninka.ru/article/n/realizatsiya-besprovodnyh-setey-na-osnove-tehnologii-zigbee-standarta-802-15-4
9. Шнайдер Дмитрий Александрович, Крахмалев Евгений Игоревич, Кинаш Александр Викторович Организация распределенной системы управления уличным освещением на основе беспроводной сети стандарта ZigBee // Вестник ЮУрГУ. Серия: Компьютерные технологии, управление, радиоэлектроника. 2010. №2 (178). URL: https://cyberleninka.ru/article/n/organizatsiya-raspredelennoy-sistemy-upravleniya-ulichnym-osvescheniem-na-osnove-besprovodnoy-seti-standarta-zigbee
10. Аладов Андрей Вальменович, Валюхов Владимир Петрович, Закгейм Александр Львович, Черняков Антон Евгеньевич, Цацульников Андрей Федорович Динамически управляемые светодиодные источники света для новых технологий освещения // Научно-технические ведомости СПбГПУ. Физико-математические науки. 2014. №4 (206). URL: https://cyberleninka.ru/article/n/dinamicheski-upravlyaemye-svetodiodnye-istochniki-sveta-dlya-novyh-tehnologiy-osvescheniya
11. Крахмалев Евгений Игоревич Энергосервис в системах уличного освещения: технико-экономические аспекты // Вестник ЮУрГУ. Серия: Компьютерные технологии, управление, радиоэлектроника. 2012. №35. URL: https://cyberleninka.ru/article/n/energoservis-v-sistemah-ulichnogo-

osvescheniya-tehniko-ekonomicheskie-aspekty

12. Rusbase. [Электронный ресурс]: документация. – режим доступа: https://rb.ru/opinion/umnyj-svet-ekonomen/
13. Официальный сайт фирмы Arduino Software [Электронный ресурс] URL: http://arduino.cc (Дата обращения: 02.02.2020).
14. Барашко, О.Г. Проектирование систем домашней автоматизации: учеб. пособие. / О. Г. Барашко, А. В. Овсянников. – «Белорусский государственный технологический университет», 2006 – 57с.
15. Баранов В. Н. Применение микроконтроллеров AVR: схемы, алгоритмы, прораммы, 2-е изд. испр. - М.: Издательский дом "ДодэкаXXI", 2014. – 288 с.
16. Богданов, С. В. Умный дом: монография / С. В. Богданов. - 2-е изд., перераб. и доп. - СПб. : Наука и Техника, 2015. – 208 с
17. Е.А. Тесля. «Умный дом» своими руками. Строим интеллектуальную систему в своей квартире / Тесля Е.А. - Санкт Петербург, 2008.
18. Амперка/Вики [Электронный ресурс] URL: http://wiki.amperka.ru/ (Дата обращения: 02.03.2020).
19. Arduino Uno: Основы программирования [Электронный ресурс] URL: http://developer.alexanderklimov.ru/arduino/arduino-minimum.php (Дата обращения: 08.03.2020).
20. Описание основных функций языка Arduino [Электронный ресурс] URL: http://freeduino.ru/arduino/lang.html (дата обращения: 11.02.2020)
21. История возникновения Умного дома. [Электронный ресурс]. URL: http://dic.academic.ru (Дата обращения: 15.01.2020).
22. Харке, В. Умный дом. Объединение в сеть бытовой техники и систем коммуникаций в жилищном строительстве: монография / В. Харке; пер. с нем. И. В. Рядченко. - М. : Техносфера, 2014 (Чебоксары). – 287 с
23. Марк, Э.С. Практические советы и решения по созданию "Умного дома" / НТ Пресс, 2007
24. Широтно-импульсная модуляция [Электронный ресурс] //

Информационный ресурс «Амперка», 2015. – Режим доступа: http://wiki.amperka.ru/конспект-arduino:шим (Дата обращения: 17.04.2020).

25. Сколько стоит умный дом [Электронный ресурс] // Home Sapiens. – Режим доступа: http://home-sapiens.ru/skolko-stoit-umnyiy-dom/ (Дата обращения: 29.05.2020).
26. Besmart . Управление Умным домом. [Электронный ресурс] — Режим доступа: http://www.besmart.su/upravlenie (Дата обращения: 15.04.2020).
27. Steven Goodwin. Smart Home Automation with Linux. Learn how to control your home from your PC / Steven Goodwin. Apress. New York, 2015. 269 p.
28. Росляков, А.В. Р75 Интернет вещей: учебное пособие [текст] / А.В. Росляков, С.В. Ваняшин, А.Ю. Гребешков. – Самара: ПГУТИ, 2015. – 200 с.
29. Лавкрафт Г.Ф., Комната с заколоченными ставнями / Г.Ф. Лавкрафт; Аркхэм: Arkham House, 1959. – 176 с.
30. Компания SMART HOME [Электронный ресурс] - URL: http://umniydomspb.ru (Дата обращения: 08.05.2020).
31. Palle Andersen, Tom S. Pedersen, Kirsten M. Nielsen. An Investigation of Energy Storage Possibilities in Single Family Houses for Smart Grid Purposes. IFAC Proceedings Volumes, 2014.
32. Tao Jin, Fuliang Chu, Cong Ling, Daniel Legrand Mon Nzongo. A Robust WLS Power System State Estimation Method Integrating a Wide-Area Measurement System and SCADA Technology. Energies, 2015.
33. Mike Riley «Programming Your Home Automate with Arduino, Android, and Your Computer» - « The Pragmatic Bookshelf Dallas, Texas • Raleigh, North Carolina », 2014. — 242 p.
34. Jonathan Oxer and Hugh Blemings. Practical Arduino: Cool Projects for Open Source Hardware Copyright © 2009. [Электронный ресурс]. URL:

https://docviewer.yandex.ru/view/0 (Дата обращения: 03.04.2020).

35. Скуснов Александр ZigBee: взгляд вглубь // Компоненты и Технологии. 2005. №48. URL: https://cyberleninka.ru/article/n/zigbee-vzglyad-vglub (дата обращения: 22.11.2018).
36. Пушкарев Олег , Ильин Павел Cc2530 — новый ZigBee-трансивер для широкого спектра применений // Компоненты и Технологии. 2009. №99. URL: https://cyberleninka.ru/article/n/ss2530-novyy-zigbee-transiver-dlya-shirokogo-spektra-primeneniy (дата обращения: 12.11.2018).
37. Чуркин Иван Михайлович, Синичкин Олег Игоревич, Певчев Владимир Павлович Некоторые технические решения проблемы повышения частоты повторения силовых воздействий кодоимпульсного сейсмоисточника // Вестник НГИЭИ. 2015. №12 (55). URL: https://cyberleninka.ru/article/n/nekotorye-tehnicheskie-resheniya-problemy-povysheniya-chastoty-povtoreniya-silovyh-vozdeystviy-kodoimpulsnogo-seysmoistochnika (дата обращения: 22.11.2018).
38. Крахмалев Евгений Игоревич Энергосервис в системах уличного освещения: технико-экономические аспекты // Вестник ЮУрГУ. Серия: Компьютерные технологии, управление, радиоэлектроника. 2012. №35. URL: https://cyberleninka.ru/article/n/energoservis-v-sistemah-ulichnogo-osvescheniya-tehniko-ekonomicheskie-aspekty
39. Rusbase. [Электронный ресурс]: документация. – режим доступа: https://rb.ru/opinion/umnyj-svet-ekonomen/
40. Официальный сайт фирмы Arduino Software [Электронный ресурс] URL: http://arduino.cc (Дата обращения: 02.03.2020).
41. Барашко, О.Г. Проектирование систем домашней автоматизации: учеб. пособие. / О. Г. Барашко, А. В. Овсянников. – «Белорусский государственный технологический университет», 2016 – 57с.
42. Deployment of smart home management system at the edge: mechanisms and protocols, , J.M. & Gonciarz, F. Neural Comput & Applic (2018). https://doi.org/10.1007/s00521-018-3545-7

43. Digital LED Pixels: Instructions for use and a characterization of their properties, 2015, Pete R. Jones, Sara E. Garcia, Marko Nardini, https://link.springer.com/article/10.3758/s13428-015-0653-5
44. Servo signal processing for flying height control in hard disk drives, 2011, Uwe Boettcher, Christopher A. Lacey, Hui Li, Kensuke Amemiya, Raymond A. de Callafon, Frank E. Talke https://link.springer.com/article/10.1007/s00542-010-1193-7
45. Simulation of multibody systems with servo constraints through optimal control, 2016, R. Altmann, J. Heiland, https://link.springer.com/article/10.1007/s11044-016-9558-z
46. Bluetooth in Intelligent Transportation Systems: A Survey, 2014, M. R. Friesen, R. D. McLeod, https://link.springer.com/article/10.1007/s13177-014-0092-1
47. HAL, Wikipedia.org [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B9_%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D1%8B%D1%85_%D0%B0%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D0%B9 (Дата обращения: 02.03.2020).