МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ
ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри

_____ І.А. Жуков
(підпис)

«_____»_____2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР

ЗА СПЕЦІАЛЬНІСТЮ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: Система виклику медичного персоналу на базі *BLE* та *iOS*-технологій

Виконавець: студент КС231М Пуліковський Дмитро Сергійович
(студент, група, прізвище, ім'я, по батькові)

Керівник: к.т.н., доцент Єфимець Валентин Микитович
(науковий ступінь, вчене звання, прізвище, ім'я, по батькові)

Нормоконтролер: _____ Малярчук В.О.
(підпис) (ПІБ)

Засвідчую, що у дипломній роботі немає запозичень праць інших авторів без відповідних посилань
Студент _____ Пуліковський Д.С.
(підпис) (ПІБ)

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних систем та мереж

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____І.А. Жуков
(підпис)

« ____» _____ 2020 р.

З А В Д А Н Н Я
на виконання дипломної роботи (проекту)

Пуліковського Дмитра Сергійовича
(прізвище, ім'я, по батькові)

1. Тема роботи: Система виклику медичного персоналу на базі *BLE* та *iOS*-технологій
затверджена наказом ректора від " 25 " вересня 2020 року № 1793/ст
2.Термін виконання проекту (роботи): з 05.10.2020 до 30.12.2020
3. Вихідні дані до роботи: система виклику медичного персоналу
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1. Аналіз можливостей стандарту *ble* в системах передачі даних;
2. Принципи використання низькочастотних радіопристроїв для передачі інформації;
3. Реалізація системи виклику медичного персоналу

5. Перелік обов'язкового графічного матеріалу:
Презентація *Power Point*

6. Календарний план-графік

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1	Ознайомитись з постановкою задачі дипломного проектування	22.09.20	
2	Вивчити спеціальну літературу і технічну документацію	24.09.20	
3	Провести аналіз	28.09.20	
4	Спроектувати систему виклику медичних спіробітників	27.10.20	
5	Дослідити методи оцінки рівня радіосигналу у преиміщенні	07.11.20	
6	Написати пояснювальну записку	12.12.20	
7	Підготовити графічний демонстраційний матеріал	16.12.20	

7. Дата видачі завдання « » вересня 2020 р.

Керівник дипломної роботи _____ Єфимець В.М.
(підпис) (П.І.Б.)

Завдання прийняв до виконання _____ Пуліковський Д.С.
(підпис студента) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Система виклику медичного персоналу на базі *BLE* та *iOS*-технологій”: 94 с., 32 рис., 5 табл., 22 літературних джерел*f*.

BLE, BLUETOOTH, МЕДИЧНА СИСТЕМА, РІВЕНЬ СИГНАЛУ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Мета дипломної роботи – розробити систему виклику медичного персоналу на базі блутус технології.

Об’єкт дослідження – система виклику медичного персоналу.

Предмет дослідження – використання *Bluetooth*-пристроїв в системі виклику медичного персоналу.

Метод дослідження – аналіз рівня радіосигналу Система виклику медичного персоналу на базі блутус технології.

Дипломна робота досліджує методи реалізації системи виклику медичного персоналу з використанням *Bluetooth*-пристроїв. Розроблена система рекомендується до використовування у медичних закладах та у вищих навчальних закладах у якості прикладу використання *Bluetooth*-пристроїв.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ATM	–	*Asynchronous Transfer Mode*
BLE	–	*Bluetooth Low Energy*
FIFO	–	*First input first output*
MDRR	–	*modified weighted round-robin algorithm*
PQ	–	*priority queue*
WFQ	–	*Weighted Fair Queuing*
БКП	–	Багатоканальний пристрій
ЕОМ	–	Електронно обчислювальна машина
ЛКМ	–	Локальна комп’ютерна мережа
ОКП	–	Одноканальний пристрій
ОС	–	Операційна система
ПК	–	Персональний комп’ютер

ВСТУП

Успіхи в області напівпровідникової електроніки, що дозволяють інтегрувати на одному кристалі велику кількість різноманітних пристроїв (в тому числі можливість інтеграції аналогових і цифрових схем), досягнення в технології виробництва інтегральних схем (зниження вартості виробництва) сприяють проникненню в повсякденне життя різних електронних пристроїв і систем. Часто вони стають повсякденними і непомітними, проте за кожною з них стоїть праця багатьох людей і технології. Особливо інтенсивно розвиток йде в сфері вбудованих систем і портативних пристроїв, що використовують радіоканал – часто це ті пристрої, які оточують нас в звичайному житті, що працюють в умовах різних обмежень – законодавчих, медичних, вага і розмір. Такі пристрої відносять до класу малопотужних радіопристроїв.

Бездротові системи міцно увійшли в наше життя:

– в побуті це різні мультимедійні системи, вузли знаходяться, бездротові інтерфейси, різноманітні системи моніторингу;

– в промисловості – системи збору даних, автоматизовані та автоматичні системи управління (від систем освітлення, до автоматизації будівель і їх комплексів);

– в транспорті – відстеження вантажів, моніторинг параметрів руху і т.д.

Однією з найбільш часто зустрічаються в повсякденному житті бездротових мережевих технологій, після стільникового зв’язку, є *Bluetooth*. Завдяки відносно високих швидкостей передачі даних і непоганим енергетичними показниками технологія *Bluetooth* отримала широке поширення в мобільних електронних пристроях, персональних комп’ютерах, ноутбуках, бездротових навушниках, гарнітури, мультимедійних центрах. Стандарт дозволяє підтримувати досить розгалужену і складну мережу пристроїв. Однак, для застосування в сенсорних мережах класичний *Bluetooth* підходить мало через значно для автономних джерел живлення енергоспоживання, внаслідок

особливостей роботи стека протоколів.

У світлі останніх подій (пандемія) і підвищеною увагою до медичної галузі одним з перспективних секторів ринку в Україні є використання різних сенсорів в телемедицині і спецілаізірованних пристроїв термінового виклику.

На даний момент в основному використовуються реалізації бездротового обміну даними представляється низькочастотна частина *ISM* діапазону, а саме частоти менше 1 ГГц. Причини цього наступні:

– в діапазоні 2,4 ГГц присутня велика кількість пристроїв – комп'ютери і бездротове мережеве обладнання, бездротові навушники, гарнітури, системи типу "розумний будинок";

– сигнали з частотами менше 1 ГГц менше схильні до впливу перешкод у вигляді стін, будинків, дерев;

– при рівних потужностях можуть забезпечити більш впевнений прийом даних (зменшення частоти передачі в два рази приблизно в стільки ж збільшує дальність (формула Фріза)).

Але використання пристроїв діпозона 2,4 ГГц дозволяє використовувати дешеві універсальні пристрої для вирішення глобальних завдань.

Якщо перед вами поставлено завдання спроектувати рішення на базі пристроїв, що живляться від однієї батарейки, які збирають інформацію в перебігу багатьох років і передають її по каналу *Bluetooth*, то саме логічне рішення.

Раніше при розробці різних пристроїв використовували звичайний *Bluetooth*. Звичайний *Bluetooth* – це по суті бездротової *UART*. Маємо потік даних, що приймається, і потік даних, що відправляється.

Переваги подібних систем в простоті розробки – команди або інформація приймаються по *UART*, далі відбувається простий парсинг.

Недоліки в тому, що за замовчуванням передача йде весь час. Найпопулярніший *Bluetooth* пердатчік типу *HC*-05, *HC*-06 споживає в такому режимі десятки міліампер постійно, якщо самостійно не вживати заходів по відключенню модуля і включенню на час активного з'єднання. А що якщо,

наприклад, другий пристрій хоче відправити повідомлення про якусь подію, а другий модуль спить? Тоді другий модуль отримає це повідомлення тільки коли прокинеться. Але що якщо ми хочемо з одного боку мати дуже низьке енергоспоживання, а з іншого боку бути в робочому режимі тільки коли йде якась передача даних? А що якщо потрібно з'єднати не два, а три і більше пристроїв? Тут на допомогу якраз і приходить *BLE* і пристрої які отримали назву «маячки» (*beacon*).

Під «маячками» маються на увазі мініатюрні радіопередавачі на основі технології *Bluetooth Low Energy* на батарейках, які періодично відправляють дані в ефір. Давайте детально розглянемо найбільш важливі аспекти, які потрібно враховувати при проектуванні маячків. З використанням *BLE*-стека *Texas Instruments* розробка маячків відбувається простим і інтуїтивно зрозумілим способом.

Протокол *BLE* використовується за сценарієм: рідко передавати дані і обробляти довгий час. Зокрема, можливе використання дворежимних гаджетів *BLE* в смартфонах, планшетних ПК, ноутбуках. Однорежимні можуть використовуватися в безлічі сфер діяльності. Під ці сфери потрапляють пристрої з розділів здоров'я, автоматизації, аналізу, управління.

Безліч завдань можуть вирішуватися, коли в радіусі дворівневого модуля визначаються інші однорівневі *BLE*-прилади. До цих приладів відносяться прилади-сигналізатори, що повідомляють власника про видалення від сумки, барсетки, гаманця, переносний тари та інших персональних речей, оснащених *BLE*-модулем. Відмінне застосування даними брелоки з *BLE* знаходять в якості маячків для дитини, щоб не втратити його в досить людних місцях.

Стійка робота і низьке енергоспоживання протоколу *BLE* дозволяють розглядати його в якості заміни *NFC*, а саме *RFID*-міток. Але варіант суміщеної роботи *BLE + NFC* виглядає більш привабливо. *BLE* дає великий радіус, пов'язаний зі стійкою роботою, другий відповідає за логічне сполучення пари,

плюс забезпечує надійний захист за рахунок малого радіусу дії.

Робота приладів через блутус з низьким енергоспоживанням дозволяє відкривати віддалено двері, ворота і приводити в дію інші механізми з великої відстані, довго не змінюючи акумулятор в бездротовому і компактному органі управління.

Крім того, використання з смартфоном *BLE*-модуля дозволить на великій відстані через пов'язані канали керувати будь-якими приладами і аксесуарами розумного будинку або іншої інфраструктури на базі *BLE*-технологій.

Одним з варіантів застосування маячків є системи виклику медперсоналу, які призначені для організації пошукового виклику персоналу, двостороннього зв'язку між персоналом і хворими, виклику чергових медсестер.

Подібні системи забезпечують наступні функції:

– організацію двостороннього зв'язку між персоналом і пацієнтом;

– організацію двостороннього зв'язку між черговим лікарем і медсестрою;

– виклик чергової медсестри хворим з душової кабіни, палати, туалету;

– можливість об'єднання з системами радіотрансляції і колективного прийому телебачення з забезпеченням вибору програм з пульта системи виклику медперсоналу;

– відображення виклику світловими покажчиками.

Системи виклику персоналу знаходять все більше використання в різних видах медичних установ, таких як клінічні та лікарняні стаціонари, санаторії, будинки для людей похилого віку та хосписи.

Дані системи спрощують процес спілкування медичного персоналу та пацієнтів, забезпечують належний догляд за пацієнтами, їх комфорт і захищеність під час перебування в лікарні, підвищують ефективність роботи персоналу і покращують спостереження медичного персоналу за відділенням в цілому. З впровадженням таких систем медичний персонал своєчасно і повно інформується про виклики пацієнтів, що змушує співробітників відповідальніше підходити до роботи і краще справлятися з повсякденними

обов'язками. Крім того, реєстрація всіх викликів і дій персоналу дозволяє відстежити і оцінити їх дії в неоднозначних ситуаціях.

З огляду на вище сказане можна визначити мету, об’єкт та предмет дослідження.

Мета дипломної роботи – розробити систему виклику медичного персоналу на базі блутус технології.

Об’єкт дослідження – система виклику медичного персоналу.

Предмет дослідження – використання *Bluetooth*-пристроїв в системі виклику медичного персоналу.

РОЗДІЛ 1
АНАЛІЗ МОЖЛИВОСТЕЙ СТАНДАРТУ *BLE* В СИСТЕМАХ ПЕРЕДАЧІ ДАНИХ

1.1. Аналіз методів застосуванням пристроїв, що працюють за технологією *Bluetooth Low Energy*

1.1.1. Відмінності *Bluetooth Low Energy* від стандартного *BlueTooth*

У *BLE* модуль споживає лише в моменти передачі даних, в решту часу він знаходиться в глибокому сні. Завдяки цьому можна істотно знизити енергоспоживання. Однак, чим частіше планується передача даних, тим більше буде енергоспоживання. І при дуже частою передачі даних енергоффектівность *BLE* буде несильно відрізнятися від звичайного *Bluetooth*.

Також *BLE* дозволяє підключатися відразу до декількох пристроїв. Якщо класичний *Bluetooth* підтримує топологію типу «Точка-Точка», то *BLE* підтримує «Мовлення» і «чарункова мережа».

Мовлення – периферійний пристрій передає дані декількох центральних.

Чарункова мережа – пристрої підключені один до одного. Тобто в мережі декілька пристроїв і у кожного по кілька підключень.

В організації роботи *BLE* передбачається переведення пристрою в один з двох типів:

– *CENTRAL*, він же *CLIENT*, він же *GATT CLIENT*, він же *MASTER* – це пристрій, який підключається до периферійних;

– *PERIPHERAL*, він же *SERVER*, він же *GATT SERVER*, він же *SLAVE*.

GATT – це специфікація для відправки та отримання коротких фрагментів даних (атрибутів), вона визначає ієрархію даних, тобто те, за якими правилами пристрої обмінюються повідомленнями.

У кожного пристрою *BLE* є профіль. Він може мати 1 і більше сервісів, у сервісу повинна бути мінімум одна характеристика. Характеристика завжди

долдна мати мінімум два атрибути: оголошення характеристики і значення характеристики.

Центральний пристрій сканує *Advertising* пакети, а периферійне їх відсилає. Після того, як центральний пристрій знаходить шукане периферійний пристрій, вони переходять на канал передачі даних.

У *BLE* для передачі використовується 3 канали для *Advertising* пакетів і 37 каналів для передачі даних.

1.1.2. Властивості *Bluetooth Low Energy Notify* і *Indicate*

По суті – обидва виконують одну функцію оповіщення. Відмінність оповіщення від читання в тому, що вам не потрібно кожен раз робити запит на читання значення у пристрої. Ви отримуєте значення асинхронно, по факту, і не витрачаєте енергію на непотрібні запити. Як правило такий режим використовується, коли потрібно дізнатися, що у пристрої оновилися дані (наприклад відбулося читання з датчика температури). Також якщо, поновлення даних аперіодичне, то періодичний опитування буде марною тратою енергії. Крім того, таке опитування вимагає двостороннього зв'язку, так спочатку ми відправляємо запит на читання, а потім нам приходить відповідь. *Notification* це односторонній вид зв'язку, тому відбувається економія на часі передачі даних, а також і на енергоспоживанні. Однак, *Notify* це передача без підтверджень, це ненадійний канал передачі даних.

1.1.3. Затримки в передачі даних у протоколах *Bluetooth Low Energy*

Вимірювання проводилося для передачі типу *INDICATE*. Як говорилося вище – це асинхронна передача даних з підтвердженням. Відлік часу починався з моменту виклику функції *indicate* на *Peripheral* пристрої і закінчуватися в момент виклику коллбека про зміну характеристики на периферійному

пристрої та читанні її на *Central* пристрої. Діапазон затримок склав від 5 до 50 мс. Це потрібно враховувати в пристроях, які прив'язані до реального масштабу часу. Середнє значення затримки *BLE* становило приблизно 10 мс. Середньоквадратичне відхилення не вважали.

1.2. Аналіз стеку протоколів *Bluetooth Low Energy*

1.2.1. Структура стека

Також як і класичний стек протоколів *Bluetooth,* стек *BLE* складається з двох основних частин: контролера (*Controller*) і вузла мережі (*Host*). Контролер включає в себе фізичний і канальний рівень і часто реалізується у вигляді системи-на-кристалі (СНК) і інтегрованим бездротовим трансивером. Частина стека, іменована вузлом мережі реалізується програмно на мікроконтролері додатків і включає в себе функціональність верхніх рівнів: рівень логічного зв'язку (*Logical Link Control – LLC*), протокол адаптації (*Adaptation Protocol – L2CAP*), атрибутів (*Attribute Protocol – ATT*), протокол атрибутів профілів пристроїв (*Generic Attribute Profile – GATT*), протокол забезпечення безпеки (*Security Manager Protocol – SMP*), протокол забезпечення доступу до функцій профілю пристроїв (*Generic Access Profile* (*GAP*). Взаємодія між верхньою і нижньою частинами стека здійснюється інтерфейсом *Host Controller Interface* (*HCI*). Додаткова функціональність прикладного рівня може бути реалізована поверх рівня вузла мережі. На рис. 1.1 представлена структура стека протоколів *BLE* [15, 17].

Незважаючи на те, що деякі функції контролера *BLE* запозичені від класичного *Bluetooth,* вони не сумісні між собою, тобто пристрій, що підтримує тільки *BLE* (однорежимні пристрій – *single-mode device*) не зможе взаємодіяти з пристроєм, що підтримує тільки *Bluetooth* 2.*x* / 3.0. Для здійснення взаємодії між ними хоча б один з пристроїв має підтримувати обидва стека протоколів (дворежимні пристрій – *dual-mode device*).

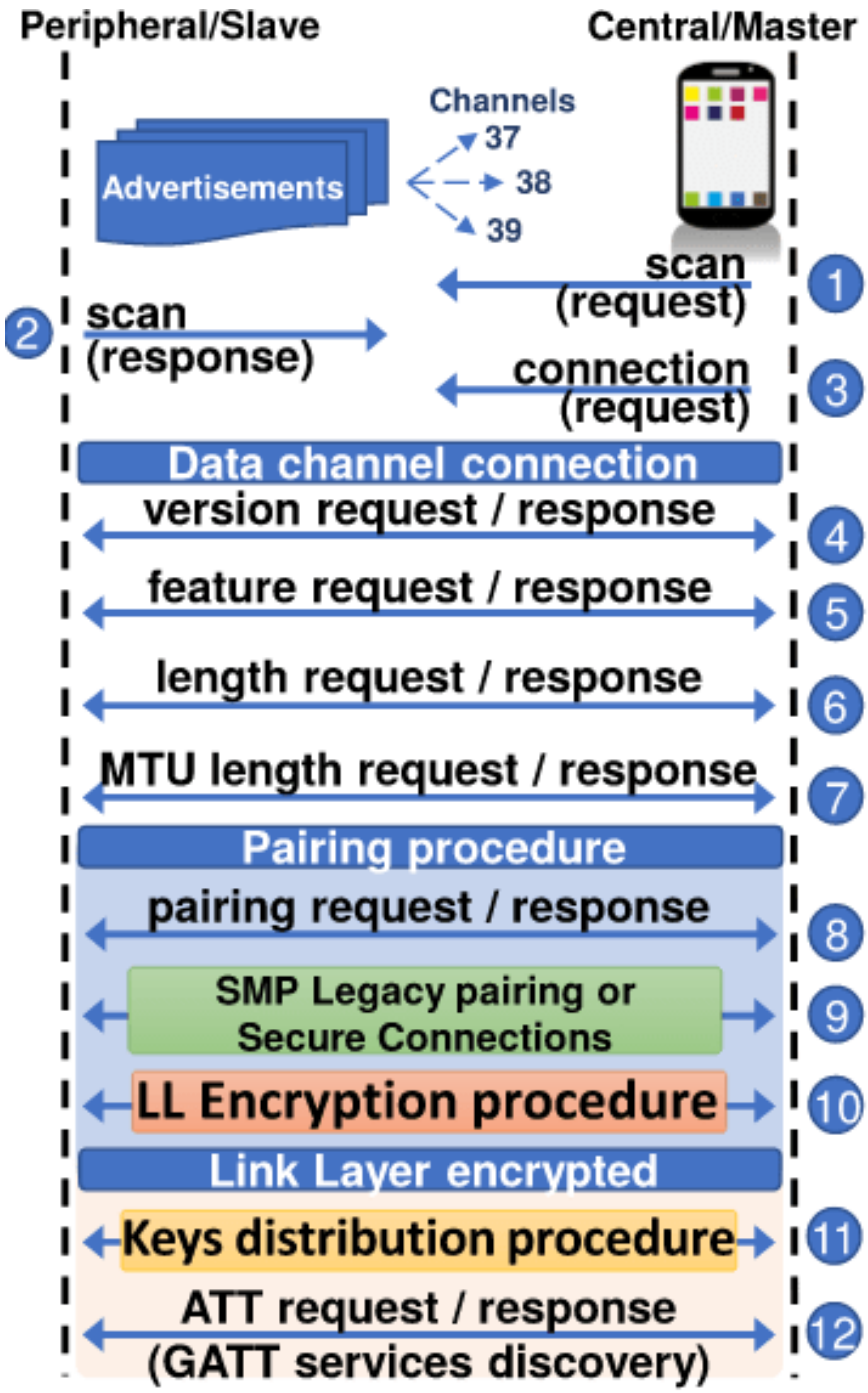

Рис. 1.1. Структура *Bluetooth Low Energy*

Однорежимні пристрої володіють найменшим споживанням і, в основному, являють собою кінцеві виконавчі пристрої. Дворежимні пристрої припускають можливість періодичного отримання енергії, розташовуються на різних мобільних пристроях, а також можуть функціонувати і як звичайні *Bluetooth* пристрою.

1.2.2. Фізичний рівень протоколу *BLE*

Пристрої *BLE* працюють в діапазоні 2.4 ГГц. У стандарті визначено 40 частотних каналів з відстанню в 2 МГц між каналами. На фізичному рівні застосована *GFSK* модуляція (*Gaussian Frequency Shift Keying*) з індексом модуляції в межах від 0.45 до 0.55, що дозволяє зменшити пікове споживання енергії. Швидкість передачі на фізичному рівні 1 Мбіт / с. У стандарті *BLE* чутливість приймача визначена як рівень сигналу на приймачі, при якому частота бітових помилок (*Bit Error Rate – BER*) досягає рівня 10-3, і повинна бути -70 дБм або краще. Виділяють два типи каналів – канали оголошення і канали даних. Канали оголошення використовуються для пошуку пристроїв, встановлення з'єднання, широкомовних передач, тоді як канали даних використовуються для двонаправленого обміну даними між пристроями, між якими встановлено з'єднання.

Для каналів оголошення виділено три частотних каналу в центрі смуги, що мінімізує перекриття з каналами 1, 6 і 11 стандарту *IEEE* 802.11. Решта 37 каналів використовуються для обміну даними (рис. 1.2) [15].

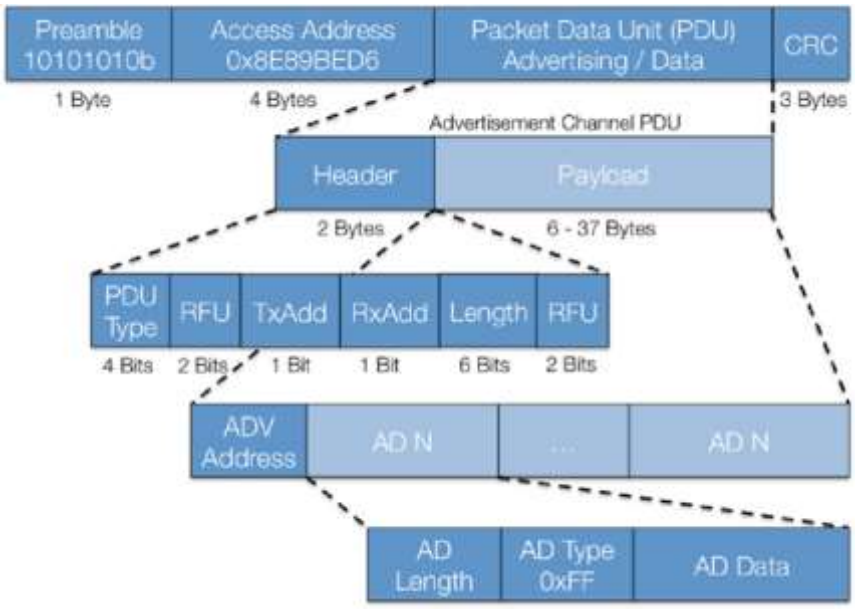

Рис. 1.2. Структура пакету даних

1.2.3. Канальний рівень протоколу *BLE*

У *BLE* для передачі широкомовних пакетів застосовуються канали оголошення. Будь-пристрій, що передає пакети за даними каналах називається проголошувачем. Передача пакетів по каналах оголошень відбувається тільки протягом спеціальних виділених інтервалів часу, які називаються подіями оголошень. Під час цих подій пристрій-проголошувач передає пакети оголошень послідовно по кожному з трьох каналів. Пристрої, тільки приймають пакети оголошень називаються сканерами [17].

Двонаправлений обмін між *BLE*-пристроями можливий тільки після встановлення з'єднання між ними. Створення нового з'єднання між двома пристроями є асиметричною процедурою, протягом якої пристрій-проголошувач по каналах оголошення сигналізує про свою готовність до з'єднання, в той час як інший пристрій (ініціатор з'єднання) прослуховує дані канали. Коли ініціатор виявляє потрібний пристрій, він може надіслати запит на встановлення з'єднання (*Connection Request*) проголошувачю, який встановлює між ними з'єднання. З цього моменту пристрої можуть здійснювати обмін по каналах даних. Пакети, що відносяться до встановленому з'єднанню, будуть відзначені згенерував випадковим чином 32-бітовим кодом доступу.

Також як і в класичному варіанті *Bluetooth*, в *BLE* для встановленого з'єднання один з пристроїв виступає в якості ведучого (*master*), друге – веденого (*slave*). В ході процедури встановлення з'єднання – ініціатор і проголошувач, відповідно. Провідний пристрій може підтримувати кілька з'єднань з відомими, в той час як ведене пристрій може мати тільки одне підключення – до ведучого. Таким чином, *BLE*-пристрій одночасно може належати тільки одній піко-мережі. В цьому криється ще одна відмінність *BLE* від *Bluetooth* – в останньому випадку ведене пристрій в свою чергу могло виступати в якості ведучого пристрою своєї власної піко-мережі.

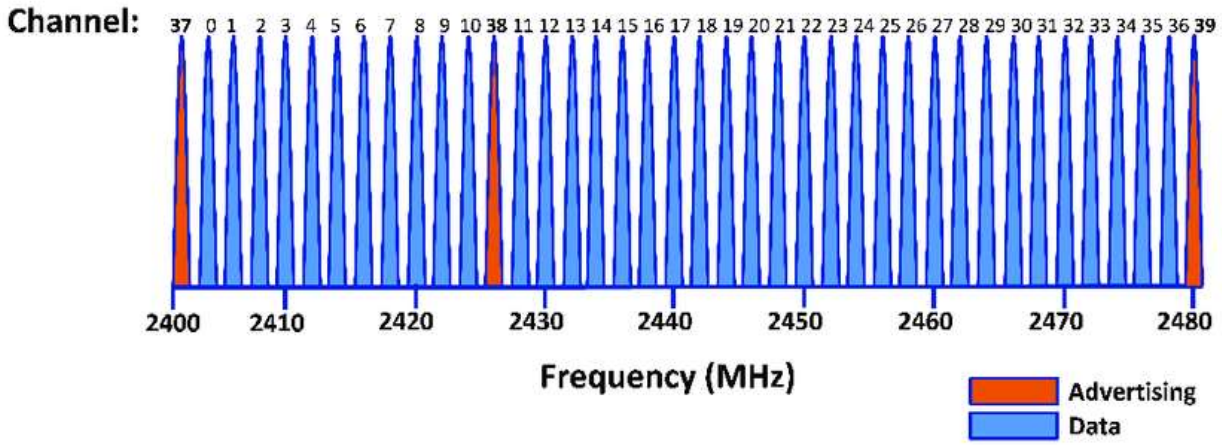

Рис. 1.3. Співвідношення частотних каналів *BLE* і каналів *IEEE* 802.11

Для економії енергії ведене пристрій за умовчанням знаходиться в сплячому стані, періодично прокидаючись для перевірки наявності пакетів даних від ведучого. Ведучий визначає для своїх ведених пристроїв моменти часу, в які ведений прокидається для прослуховування каналу, регулюючи, тим самим, доступ пристроїв до середовища передачі здійснюється за схемою поділу часу (*Time Division Multiple Access – TDMA*). Провідний пристрій також задає веденим схему перемикання частотних каналів. Параметри з'єднання передаються в повідомленні запиту на встановлення з'єднання і можуть бути оновлені при необхідності (наприклад, якщо змінився схеми перемикання каналів для усунення перекриття з частотними каналами інших пристроїв).

Після встановлення з'єднання фізичний канал передачі даних розділяється на неперекривающіеся тимчасові інтервали, звані подіями з'єднання (*connection events*) або фреймами. Протягом фрейма всі пакети передаються по одному частотному каналу. Кожен фрейм починається з передачі пакета провідним пристроєм. У тому випадку, якщо відоме пристрій отримало пакет, воно повинно послати пакет-підтвердження. У той же час, підтвердження від ведучого пристрою відомому не потрібно. Між двома послідовними пакетами повинен бути витриманий інтервал часу, як мінімум 150 мкс – т.зв. міжкадровий інтервал (*Inter Frame Space – IFS*).

До тих пір, поки між ведучим і веденим триває обмін пакетами, подія

з’єднання (або фрейм обміну) вважається відкритим. В пакетах даних, в разі необхідності подальшого обміну, встановлений біт *More Data* (*MD*). Якщо не один з пристроїв не має даних для передачі, подія з’єднання буде закрито, і ведене пристрій вже не повинно прослуховувати канал до початку наступного фрейму. Іншими причинами, що призводять до закриття події з’єднання, є два послідовно прийнятих пакета з помилками, неправильну адресу пристрою в пакеті. Для контролю бітових помилок в пакеті, після поля даних слід поле 24-бітної контрольної суми.

Для нового події з’єднання провідний і ведений використовують новий частотний канал, заданий в карті перемикання каналів. Час між початком двох послідовних подій з’єднання задається параметром *connInterval*, є кратним 1.25 мс і може лежати в межах від 7.5 мс до 4 с. Другим важливим параметром для піко мережі *BLE* є параметр *connSlaveLatency* визначає кількість послідовних фреймів протягом яких ведене пристрій не прослухає канал і може на цей час відключити трансивер. Даний параметр є цілим числом в межах від 0 до 499 і не повинно перевищувати контрольного інтервалу супервізора – параметр *connSupervisionTimeout*. Параметр *connSupervisionTimeout* може приймати значення в діапазоні від 100 мс до 32 с [17].

На канальному рівні для управління потоком даних діє механізм зупинки і очікування (*stop-and-wait mechanism*) на основі т.зв. кумулятивного підтвердження, службовця одночасно і оповіщенням про помилку. Тема кожного пакета, переданого по каналах даних, містить два однобітних поля, які називаються порядковим номером і наступним очікуваним порядковим номером (*Sequence Number* (*SN*) і *Next Expected Sequence Number* (*NESN*) відповідно).

SN ідентифікує пакет, тоді як, *NESN* показує, який пакет очікується від пристрою, з яким встановлено з’єднання. Якщо пакет пристроєм прийнято успішно, поле *NESN* в його наступному пакеті буде збільшено, і такий пакет одночасно буде вважатися пакетом підтвердження. В іншому випадку, якщо пристрій виявляє помилку (не сходиться контрольна сума),

1.2.4. Протокол *L2CAP*

Протокол *L2CAP* в *BLE* є спрощеною і оптимізованою версією відповідного протоколу в *Bluetooth* 2.*x* / 3.*x*. У *BLE* основним завданням *L2CAP* є мультиплексування даних трьох протоколів (*ATT, SMP, Link Layer*) для з'єднання канального рівня. Відповідає за встановлення логічного з'єднання. Не проводиться сегментування пакетів або складання пакетів, тому що максимальна корисне навантаження *L2CAP* в *BLE* становить 23 байта.

Визначає комунікаційні сполучення між двома пристроями, які виступають у контексті даного протоколу в якості клієнта і сервера. Сервер підтримує набір атрибутів, що представляють собою структуру даних, що дозволяє отримувати доступ до інформації, керованої протоколом *GATT*. Ролі клієнта і сервера визначаються протоколом *GATT* і не залежать від ролі пристрою в з'єднанні (провідний / ведений).

Клієнт за допомогою запитів може отримати доступ до атрибутів сервера. Крім того, сервер посилає клієнтові два типи повідомлень, що містять атрибути:

– повідомлення, які не потребують підтвердження;

– індикатори, на які клієнт зобов'язаний відповісти.

Клієнт також може послати сервера команди на зміну значень атрибутів.

Протокол *GATT* визначає середовище виконання, використовувану *ATT* для пошуку послуг і обміну характеристиками між пристроями. Характеристика в даному випадку являє собою набір даних, що включають в себе значення і властивості. Дані, що відносяться до сервісів і характеристикам, зберігаються в атрибутах.

Наприклад, сервер з працюючим сервісом "температурний датчик" може бути пов'язаний з характеристикою "температура", яка використовується для опису датчика, а інший атрибут може застосовуватися для зберігання результатів вимірювань.

1.3. Аналіз питань безпеки *BLE*

BLE пропонує кілька сервісів безпеки для захисту даних, що передаються між парою з'єднаних пристроїв. Більшість з підтримуваних сервісів можуть бути описані в термінах двох режимів: *LE Security Mode* 1 і *LE Security Mode* 2. Ці режими забезпечують сервіси безпеки на канальному рівні і рівні *ATT* відповідно[17].

Канальний рівень *BLE* підтримує шифрування і аутентифікацію на основі алгоритму *Cipher Block Chaining-Message Authentication Code* (*CCM*) і блочного шифру *AES*-128. При використанні шифрування і аутентифікації в з'єднанні, до корисного навантаження (*PDU*) додається 4-байтное повідомлення перевірки цілісності – *Message Integrity Check* (*MIC*), після чого поля *PDU* і *MIC* шифруються.

Захист каналу передачі даних між парою пристроїв в протоколі *BLE* обумовлена двома режимами *LE* Сек'юріті мод 1 і *LE* Сек'юріті мод 2. Перший режим працює на *Data Link layer* (*DLL*), другий на *AT* & *T*.

На *DLL* в протоколі *Bluetooth LE* присутній шифрування і аутентифікація за допомогою технології побудови аутентификационного коду повідомлення з блочного алгоритму шифрування (*CCM*) та шифру *AES*-128. При роботі *CCM* і *AES*-128, додається до них *Protocol Data Unit* і додаткове повідомлення для ідентифікації цілісності, розміром 4-байта, після якого *PDU* і повідомлення шифруються.

У деяких випадках аутентифікацію можна провести поверх нешифрованих з'єднання на канальному рівні. Але в такому разі на *AT* & *T* - рівнем до *PDU* плюсується дванадцяти байтная сигнатура.

Режими безпеки включають в себе кілька рівнів, використовуваних в залежності від типу з'єднання.

Також можлива передача аутентифікаційних даних поверх нешифрованих з'єднання канального рівня. В даному випадку, на рівні *ATT* до корисного навантаження додається 12-байтная сигнатура. Сигнатура обчислюється

шляхом використання алгоритму *AES*-128 як блочного шифру. Одним входом алгоритму є лічильник, що дозволяє запобігти атакам типу повтору повідомлень. Якщо приймача вдається верифікувати повідомлення, вважається, що воно прийшло від достовірного джерела.

На додаток до описаних сервісів, *BLE* підтримує механізм, званий приватним (або приватним) адресою, який дозволяє пристрою безліч часто змінюваних адрес. Цей механізм знижує загрозу відстеження *BLE* пристрою на його адресу. Приватний адреси генеруються на основі публічної адреси пристрою шляхом його шифрування з використанням ключа, отриманого від довіреної пристрою.

Кожен режим безпеки передбачає наявність декількох рівнів, що застосовуються в залежності від типу з'єднання пари пристроїв (табл. 1.1).

Таблиця 1.1

Сервіси і рівні безпеки, визначені в стекупротоколів *BLE*

Режим	Рівень безпеки	Тип з'єднання (*pairing*)	Шифрування	Перевірка цілісності	Рівень стека
LE Security Mode 1	рівень 1	немає	немає	немає	Канальний рівень (*Link Layer*)
	рівень 2	без аутентифікації	є	є	
	рівень 3	аутентифікація	є	є	
LE Security Mode 2	рівень 1	без аутентифікації	немає	є	Рівень *ATT* (*ATT Layer*)
	рівень 2	аутентифікація	є	є	

Логічне з'єднання пристроїв (*pairing*) відбувається в три етапи:

1. Перший – на канальному рівні два модуля обмінюються інформацією про доступні можливості введення-виведення, а після приймають рішення, за яким з виявлених відбудеться взаємодія. По суті на першому етапі з'єднані на канальному рівні пристрої оголошують свої доступні можливості введення-виведення, і на основі їх приймається рішення про метод взаємодії на другому етапі.

2. Другий – створення ключа для третього етапу, який називається «тимчасовий ключ короткострокового значення». Він послужить для надійної

передачі даних про тимчасове ключі. Ключ може бути переданий трьома способами: з використанням альтернативного каналу *NFC*, з введенням шестизначного коду, що вводиться користувачем, або без перевірки аутентифікації, якщо перший і другий спосіб організувати неможливо.

Метою другого етапу є генерація короткоживучого ключа (*Short-Term Key – STK*), який буде використаний на третьому етапі для забезпечення безпеки передачі поширення ключової інформації. На другому етапі пристрої спочатку домовляються про тимчасове ключі (*Temporary Key – TK*) за допомогою одного з методів:

– *Out Of Band*;

– *Passkey Entry*;

– *Just Works*.

Метод *Out Of Band* (передача поза смуги) передбачає передачу тимчасового ключа по альтернативних каналах, наприклад, використовую *NFC*. Метод *Passkey Entry* ключ задає користувач у вигляді послідовності з 6 цифр. Коли застосування обох методів не можливо, використовується метод *Just Work*, хоча він не підтримує перевірку аутентифікації, і не захищений від атаки типу "посередник" (*Man In The Middle – MITM*).

На базі ключа ТК і випадковим числам, що генерується кожним з вузлів, створюється *STK*, що є завершенням другого етапу.

3. Третій – кінцеві точки з'єднання обмінюються трьома 128 бітними ключами і, якщо порушень не помічено, пара успішно синхронізується.

LTK використовується для генерації 128-бітного ключа для шифрування і аутентифікації на канальному рівні, *CSRK* для підпису даних на рівні *ATT*, а *IRK* – для генерації приватних адрес.

Протокол управління безпекою *Security Manager Protocol* (*SMP*), що працює поверх фіксованого каналу рівня *L2CAP* і відстежує виконання всіх трьох етапів. Вразливим місцем *BLE* на поточний момент є незахищеність жодного з реалізованих в ньому методів встановлення з’єднання від пасивного прослуховування.

Так як Блютуз з низьким енергоспоживанням увійшов до специфікації *Bluetooth* 4.0, то технічні порівняння проведені з класичною версією (табл. 1.2).

Таблиця 1.2

Порівняльна таблиця *BLE* з класичним класичний *Bluetooth*

Характеристика	Класичний *Bluetooth*	*Bluetooth Low Energy*
Частота радіосигналу	2.4 ГГц	2.4 ГГц
Дальність дії	100 метрів	Більше 100 метрів
Швидкість передачі	1-3 Мб / сек	1 Мб / сек
Кількість ведених пристроїв	7	Опція залежить від реалізації
Безпека	128-бітний шифр	128-біт *AES + Counter Mode*
Затримка	100 мс	6 мс
Загальний час передачі інформації (мінімум)	100 мс	3 мс
Максимальний струм	До 30 мА	До 15 мА
Потужність	1 Вт	0,01-0,05 Вт

Також не можна не відзначити велику схожість протоколів *BLE* з *BR / EDR* – першим створеним Блутус-протоколом:

– робота в єдиному діапазоні *ISM* 4 ГГц;

– модуляція *GFSK*;

– стрибкоподібне перестроювання частоти;

– схожа схема каналу з демодуляцией аналогового сигналу і перетворенням в цифровий.

Відмінності між *BR / EDR* і *BLE*-модулями в тому, що перший ділить смугу пропускання на 79 каналів з розносом в 1 МГц, а другий працює з передавачем і приймачем для поділу смуги на 40 каналів, з розкидом в 2 МГц.

1.4. Рівень *GAP* і профілі додатків

Протокол *GAP* визначає ролу пристроїв, режими і процедури виявлення пристроїв і сервісів, управління встановленням з'єднання і безпекою. У *BLE GAP* виділяє чотири ролі для контролера – широкомовний, спостерігач, периферійний і центральний.

Широкомовний вузол може тільки передавати пакети по каналах оголошення і не підтримує з'єднання з іншими пристроями. Спостерігач здатний тільки прослуховувати канали оголошень, зокрема, здатний приймати пакети, що передаються широкомовною вузлом. Центральні вузли являють собою пристрої, здатні підтримувати кілька з'єднань, в той час як периферійні пристрої являють собою прості пристрої, здатні підтримувати одне з'єднання з центральним вузлом. Ролі центрального і периферійного вузла припускають, що пристрій здатний виконувати функції ведучого або веденого пристрою відповідно. Пристрій може підтримувати кілька ролей, але водночас активної може бути тільки одна з них.

Поверх *GAP* можуть бути побудовані додаткові профілі додатків, що підтримують необхідну користувачу функціональність. У *BLE* підтримується ієрархія профілів – профіль верхнього рівня може використовувати функції профілю низького рівня.

1.5. Аналіз пристроїв, що підтримують протокол *BLE*

Протокол *BLE* міститься в модулях з вбудованим програмним забезпеченням. Модулями оснащуються кінцеві пристрої. Як модулі можна виділити:

1, *BT*111 – створений для додатків, де потрібна робота зі стандартними протоколами *Bluetooth* і *BLE* (рис. 1.4).

Рис. 1.4. Вигляд модуля *BT*111

2. *BLE*112 – однорежимний *BLE*-модуль для сенсорних систем та інших аксесуарів на батарейках (рис. 1.5).

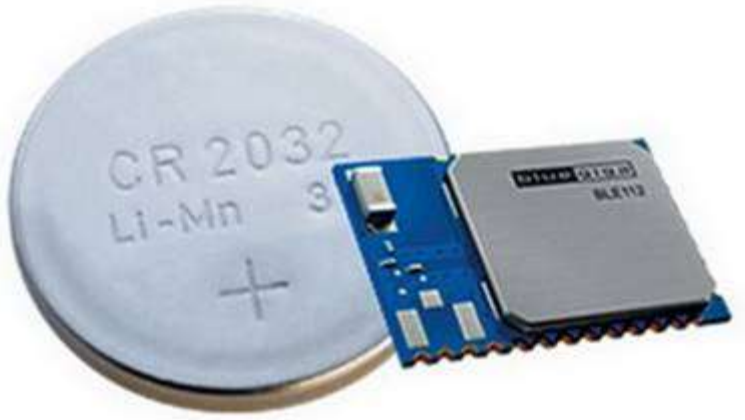

Рис. 1.5. Вигляд модуля *BLE*112

3. *USB BLED*112 – аналогічний однорежимний *BLE*-модуль з усіма властивостями *BLE*112, але виконаний в форм факторі *USB*-флешки.

До *BLE*-пристроїв можна віднести:

1. Спортивні аксесуари на кшталт шагомеров, пульсометрів, рітмометров, які мають форму годин для руки або браслета.

2. Різні сенсори для визначення руху, температури, вологості.

3. Системи читання і відображення інформації з автономним джерелом живлення.

4. Побутова медтехніка по типу вимірювачів глюкози, тонометрів, температурних вимірників.

5. Гаджети для віддаленого виклику, за типом радіо-няня.

6. Прилади побутової електроніки, за типом бездротової периферії (клавіатури, мишки), панелей і пультів.

7. Пристрої для автоматизації в житловому будинку по типу шлюзів між з'єднаної до *Smart House* сенсорної мережею і смартфонами з Блютуз.

8. Пристрої безпеки, за типом тривожних кнопок, безконтактних ключів та інше.

Щоб визначити, чи працює ваш аксесуар з протоколом-*BLE*, слід вивчити інформацію про пристрій на сайті виробника або в інструкції по експлуатації. Для пристроїв, що працюють на *Android*, можна перевірити підтримку за допомогою додатків:

1. *BLE Checker*.
2. *LightBlue*.
3. *Bluetooth LE Scanner*.

Безліч таких «визначників» знаходиться в *Play Market* і доступні для використання безкоштовно. Моделі телефонів, випущені до 2015 року включно, можуть не підтримувати цю опцію.

1.6. Аналіз експлуатаційних характеристик *BLE*

Одним з важливих показників, що визначають можливість застосування технології бездротового зв’язку до тієї чи іншої задачі, є енергоспоживання вузлів мережі, що працює за даною технологією. Це буде визначати час автономної роботи пристроїв, і, відповідно, схему технічного обслуговування мережі.

Для пристроїв *BLE* їх споживання буде залежати від ролі пристрою в з’єднанні і параметрах з’єднання, зокрема від *connInterval*, *connSlaveLatency*, *connSupervisionTimeout*, а також від якості зв’язку.

Середнє енергоспоживання вузла в режимі веденого залежно від величини *connInterval* представлено на рис. 1.4 [17].

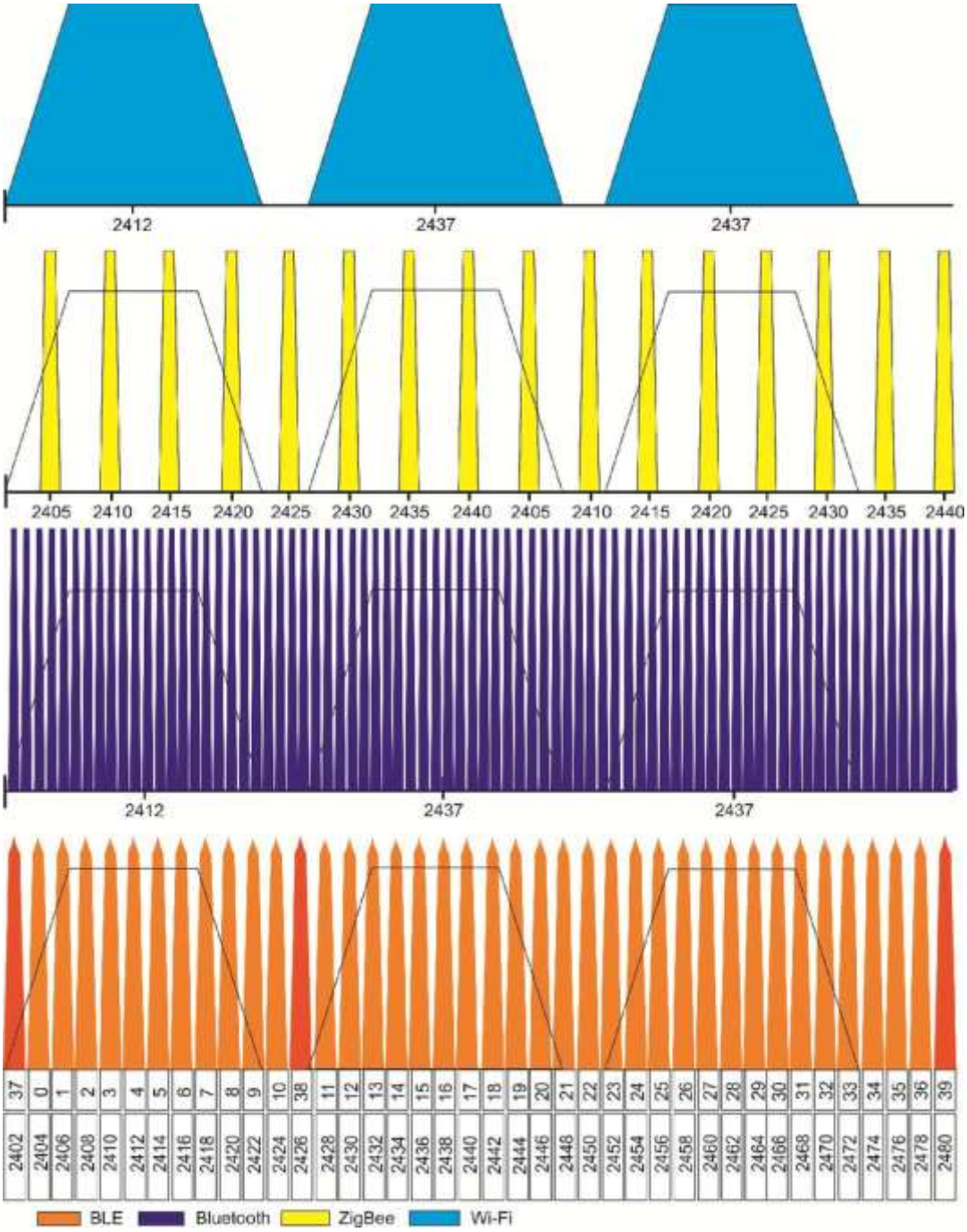

Рис. 1.4. Канали *BLE* пристрою
(вузол побудований на базі РНК *CC*2450, *connSlaveLatency* = 0)

В також [17] представлені результати теоретичного аналізу часу автономної роботи *BLE* пристрою в якості веденого залежно від інтервалу проходження подій зв'язку *connInterval*, рівня бітових помилок (*BER*) (рис. 1.5 і

рис. 1.6, відповідно).

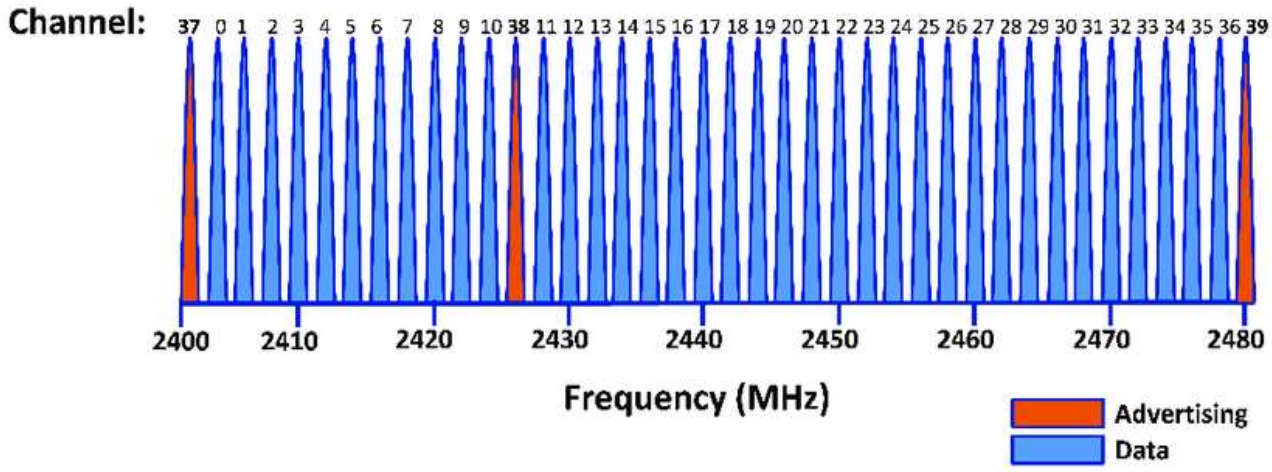

Рис. 1.5. Канли роботи *BLE*-пристрої на основі *CC*2540 від батареї ємністю 230 мАг

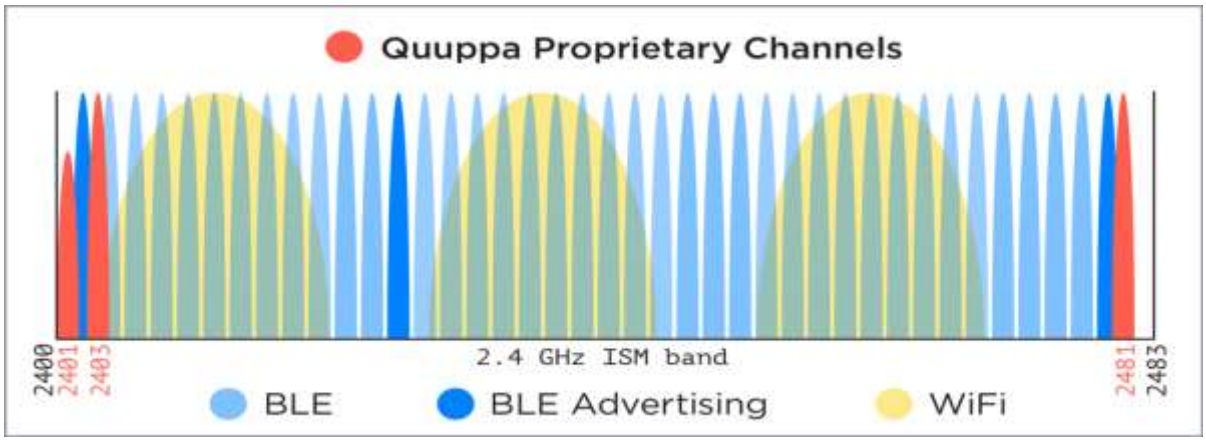

Рис. 1.6. Канли роботи *BLE*-пристрої на основі *CC*2540 від батареї ємністю 230 мАг

Дані результати, хоча і представляють максимальні оцінки часу роботи *BLE* пристроїв, але показують, що *BLE* цілком підходить для сенсорних пристроїв з автономним живленням і середнє споживання *BLE*-пристроїв цілком можна порівняти зі споживанням пристроїв "традиційних" для сенсорних мереж технологій.

Порівняльні характеристики технологій *BLE, Bluetooth, ZigBee,* 6*LoWPAN, Z-Wave* представлені в Табл. 1.3 [15, 17].

Таблиця 1.3

Деякі порівняльні характеристики технологій *BLE, Bluetooth, ZigBee,* 6*LoWPAN, Z-Wave*

Параметр	*ZigBee*	6*LoWPAN*	*Z-Wave*	*BLE*	*Bluetooth*
1	2	3	4	5	6
Частотний діапазон, МГц	868/915/2400	868/915/ 2400	868/908, 2400 (не всі версії пристроїв)	2400	2400
Бітова швидкість, Кбіт / с	20/40/250		9.6 / 40, 200	1000	<721 (*v*1.2), 3000 (*v*2 + *EDR*), <24000 (*v*3 + *HS*)
Тип модуляції сигналу	*BPSK* / *BPSK* / *O-QPSK*	*BPSK* / *BPSK* / *O-QPSK*	*BPSK*	*GFSK*	*GFSK* (*v*1.2), *GFSK* / 4-*DQPSK* / 8*DPSK* (*v*2 + *EDR*), 802.11 (*v*3 + *HS*)
Метод розширення спектра	*DSSS*	*DSSS*	немає	*FHSS* (ширина каналу 2 МГц)	*FHSS* (ширина каналу 1 МГц)
Чутливість приймача, дБм	-92 для 868/915 МГц; -85 для 2400 МГц	-92 для 868/915 МГц; -85 для 2400 МГц	-101	<-70 -87 ... -93	-90
Вихідна потужність передавача, дБм	-32 ... 0	-32 ... 0	-20 ... 0	-20 ... 10	20/4/0 (клас 1/2/3)
Розмір даних пакета, байт	до 127		до 64	Від 8 до 47	до 358
адресація	16 і 64-біт *MAC*, 16-біт ідентифікатор мережі	16 і 64-біт *MAC*, 128-біт адресу *IPv*6	32-біт ідентифікатор будинку; 8-біт адресу вузла	48-біт відкритий адреса *Bluetooth* або випадковий адресу	48-біт відкритий адреса *Bluetooth*

Продовження таблиці 1.3

1	2	3	4	5	6
Типові вимоги до реалізації стека протоколів	45 – 128 Кбайт ПЗУ; 2.7 – 12 Кбайт ОЗУ	~ 24 Кбайт ПЗУ; ~ 3.6 Кбайт ОЗУ	32 – 64 Кбайт ПЗУ; 2 – 16 Кбайт ОЗУ	~ 40 Кбайт ПЗУ; ~ 2.5 Кбайт ОЗУ	~ 100 Кбайт ПЗУ; ~ 30 Кбайт ОЗУ

Висновки за розділом

Безумовно, велика частина областей застосування *Bluetooth* може бути успішно замінена або доповнена пристроями *BLE,* продовживши термін служби пристроїв за рахунок більш ефективного управління енергоспоживанням. Зокрема, можливе застосування дворежимних пристроїв *BLE* в мобільних телефонах, планшетних комп'ютерах, ноутбуках. Однорежимні пристрої можуть застосовуватися в якості бездротового інтерфейсу датчиків на батарейках, що застосовуються як окремо, так і в складі інших пристроїв – годинах, пульсометра, крокомір, домашніх тонометрах, термометрах і тому подібних пристроїв.

У складі мобільних пристроїв *BLE* може бути використаний для управління домашньою автоматикою, пристроями освітлення або охорони як мінімум, в межах одного приміщення. Для керування пристроями в межах всього будинку можливе використання *BLE* в якості шлюзу між керуючим пристроєм і мережею домашньої автоматики.

Низький рівень споживання енергії та більш стійка робота в умовах великої кількості аналогічних пристроїв в ряді випадків дозволяє розглядати *BLE* як альтернативу пристроїв *NFC,* зокрема *RFID* мітками. Але більш цікавий варіант використання *BLE* спільно з *NFC*. У цьому випадку перші забезпечують більший радіус стійкої роботи і велику кількість спільно діючих механізмів, а другі служать для встановлення логічного з'єднання між парою пристроїв, забезпечую більш високий рівень безпеки за рахунок меншого радіуса дії.

РОЗДІЛ 2
ПРИНЦИПИ ВИКОРИСТАННЯ НИЗЬКОЧАСТОТНИХ РАДІОПРІСТРОЇВ ДЛЯ ПЕРЕДАЧІ ІНФОРМАЦІЇ

2.1. Описання пакету даних в протоколі *BLE*

Передані від пристрою *Bluetooth Low Energy* дані мають формат, відповідний базової специфікації *Bluetooth*, і складаються з декількох частин, які показані на рис. 2.1.

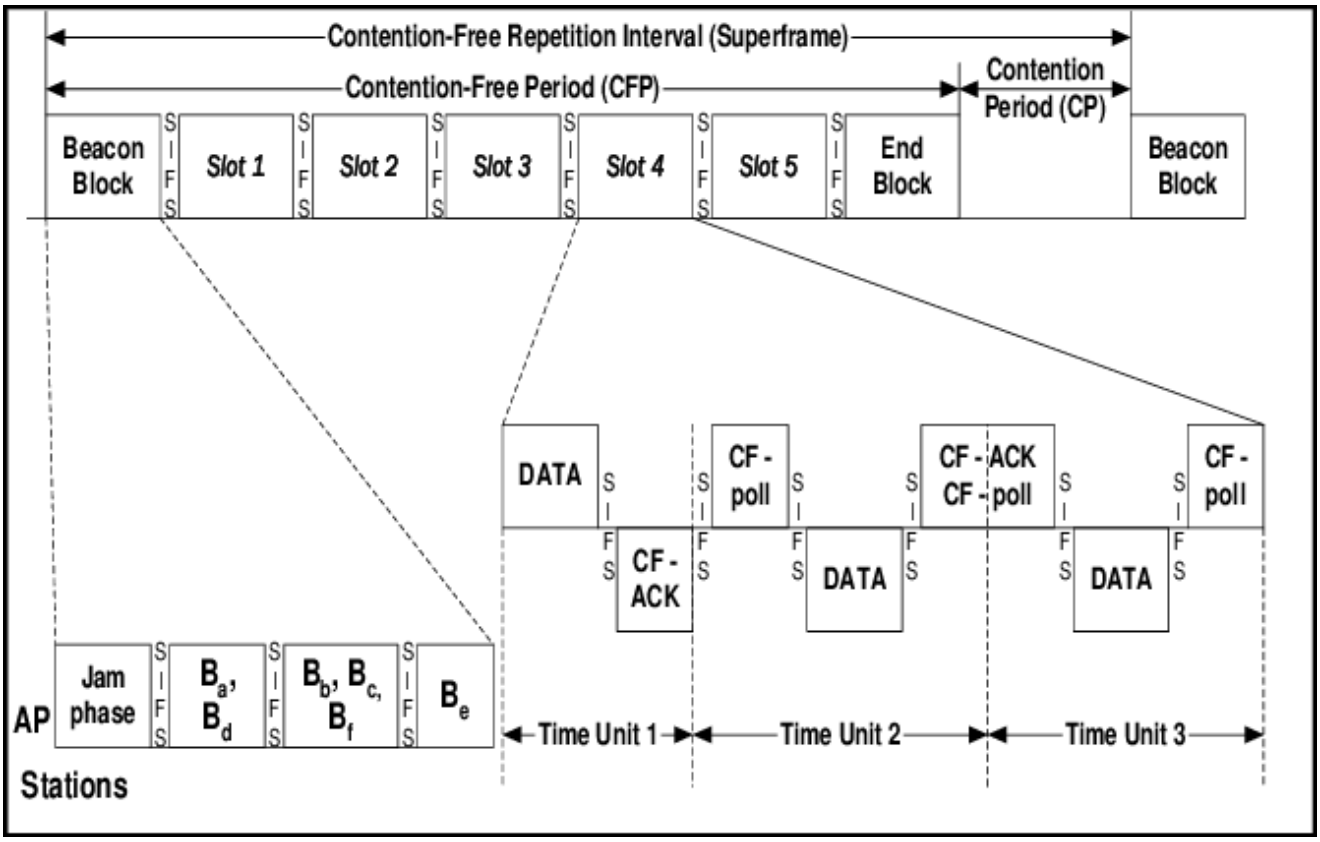

Рис. 2.1. Пакет даних *Bluetooth Low Energy*

Преамбула – однобайтове значення, яке використовується для синхронізації в приймачі. Для широкомовних пакетів воно завжди дорівнює 0*xAA*.

Адреса доступу також фіксований для широкомовних пакетів і встановлений в 0*x*8*E*89*BED*6. Корисне навантаження пакета складається з заголовку і корисного навантаження.

Тема описує тип пакету, а тип *PDU* визначає призначення пристрою. Для широкомовних додатків є три різних типи *PDU* (табл. 2.1).

ADV_SCAN_IND – широкомовний передавач, який може надати додаткову інформацію у відповідь на сканування. Біт *TxAdd* вказує, чи є адреса проголошувачя (що міститься в корисне навантаження) публічним (*TxAdd* = 0) або випадковим (*TxAdd* = 1). *RxAdd* зарезервований для інших типів пакетів, які не розглядаються в даній роботі, оскільки вони не застосовуються до маячків.

Таблиця 2.1

Типи *PDU*-оголошень для широкомовних даних

Тип *PDU*	Назва пакета	Опис
0	*ADV_IND*	Подія з’єднується ненаправленного оголошення
10	*ADV_NONCONN_IND*	Подія непоєднуване ненаправленного оголошення
110	*ADV_SCAN_IND*	Подія сканируемого ненаправленного оголошення

Остання частина переданого пакета – циклічний надлишковий код (*CRC*). Циклічний контроль надмірності – код з виявленням помилок, що використовується для перевірки цілісності пакету від небажаних змін, як правило, внаслідок перешкод в ефірі.

Це гарантує цілісність даних всіх переданих через ефір пакетів.

Корисне навантаження пакета включає адресу проголошувачя поряд з обумовленими користувачем даними, що включаються в пакет оголошення (*user defined advertised data*), як показано на рис. 2.2. Ці поля є широкомовна адреса маячків і дані.

2.1.1. Адреса і прапори пристрою

Широкомовна адреса може бути або публічним, або випадковим. Публічний адресу [4], відповідно до стандарту *IEEE* 802-2001 використовується як унікальний ідентифікатор організації (*OUI*), отриманий з центру реєстрації *IEEE*. Компанія *Texas Instruments* забезпечує адреси *IEEE* для всіх пристроїв *Bluetooth Smart*. Випадкові адреси можуть бути безпосередньо згенеровані маячком і бувають трьох різних типів:

– статичниа;

– нерозв'язна приватна;

– розв'язна приватна (*resolvable private*).

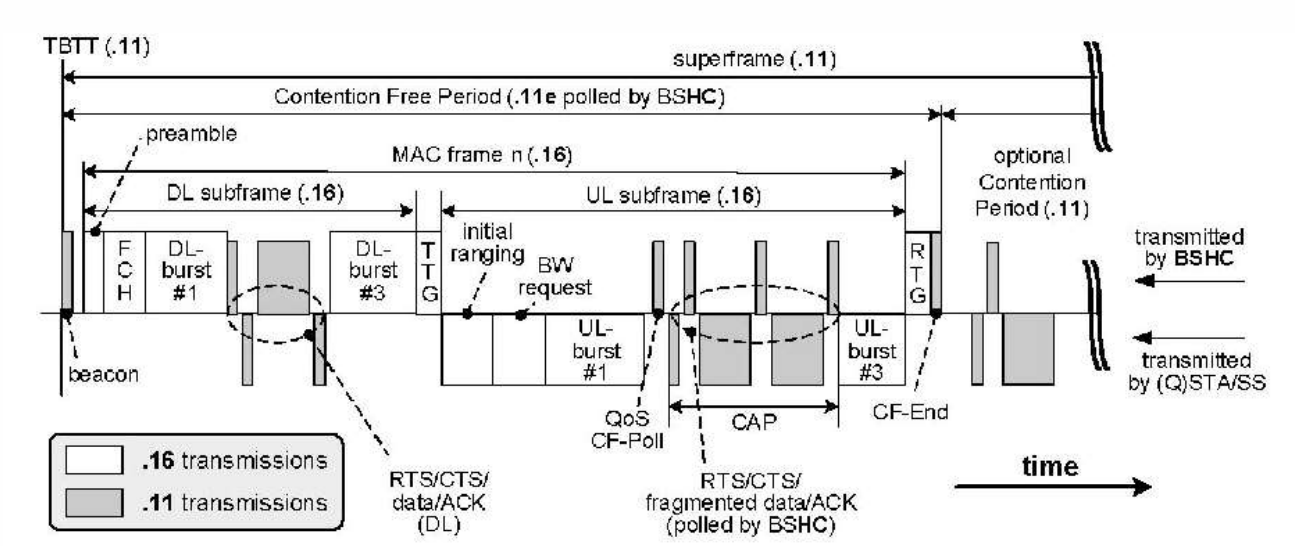

Рис. 2.2. Широкомовні дані *Bluetooth Low Energy*

Статична адреса не дозволяється змінювати, поки пристрій перезавантажиться. Приватна адреса може змінюватися з часом, а розв'язний адреса може використовуватися, щоб отримати істинний адресу. Нерозв'язна адреса також може змінитися з часом, в цьому його відмінність від статичного адреси. Випадкова адреса – засіб забезпечення секретності, яке запобігає стеження за пристроєм. Є певні правила генерації випадкових адрес, подробиці можна знайти в базовій специфікації [1].

Рис. 2.3. Можливі варіанти *BLE*-пристроїв

Передані дані можуть бути відформатовані відповідно до спеціальних форматів даних *SIG Bluetooth,* деякі приклади показані в табл. 2.2 [2, 3]. Далі приділимо увагу прапорів та даними, специфічним для виробника.

Таблиця 2.2

Типи даних оголошень

Тип даних *AD*	значення	Опис
Прапори	0*x*01	Можливості виявлення пристрою
сервіс *UUID*	0*x*02 ... 0*x*07	Служби *G*ATT-пристрої
локальне ім’я	0*x*08 ... 0*x*09	ім’я пристрою
Рівень потужності *TX*	0*x*0*A*	Вихідна потужність пристрою
Дані, специфічні для виробника (*Manufacturer-Specific Data*)	0*xFF*	визначаються користувачем

Перші три байти широкомовних даних визначають можливості пристрою. Ця вимога базової специфікації [4], як формат байтів і прапори режиму виявлення (замасковані біти) представлені в таблиці 2.3.

Якщо ніякі бітові прапори не встановлені, прапор типу даних може бути опущений [6]. Він, наприклад, не потрібно для непоєднуване пакету оголошення (*ADV_NONCONN_IND*).

Таблиця 2.3

Прапори типів даних установки з'єднанн

Байт	Біт	Прапор / значення	Опис
0	-	0*x*02	Довжина цих даних
1	-	0*x*01	Прапор типу *AD GAP*
2	0	Обмежений режим виявлення *LE*	Оголошення 180 *c*
-	1	Загальний режим виявлення *LE*	Невизначений час оголошення
-	2	*BR* / *EDR* не підтримує	-
-	3	Одночасно *LE* і *BR* / *EDR* (контролер)	-
-	4	Одночасно *LE* і *BR* / *EDR* (головний комп'ютер)	-
-	5 ... 7	-	Зарезервовані

2.1.2. Принципи використання інтервалу оголошення

Маячок забезпечує низьке енергоспоживання, переважно перебуваючи в бездіяльності і прокидаючись тільки на короткі періоди для передачі даних. Час між цими подіями мовлення називається інтервалом оголошень (рис. 2.4). Для

несоединяемих маячків інтервал не може бути менше 100 мс, а для з'єднуються маячків – менше 20 мс. До цього інтервалу додається псевдослучайная затримка 0 ... 10 мс, це гарантує, що маячки можуть працювати спільно, навіть якщо вони почали передачу в один і той же час.

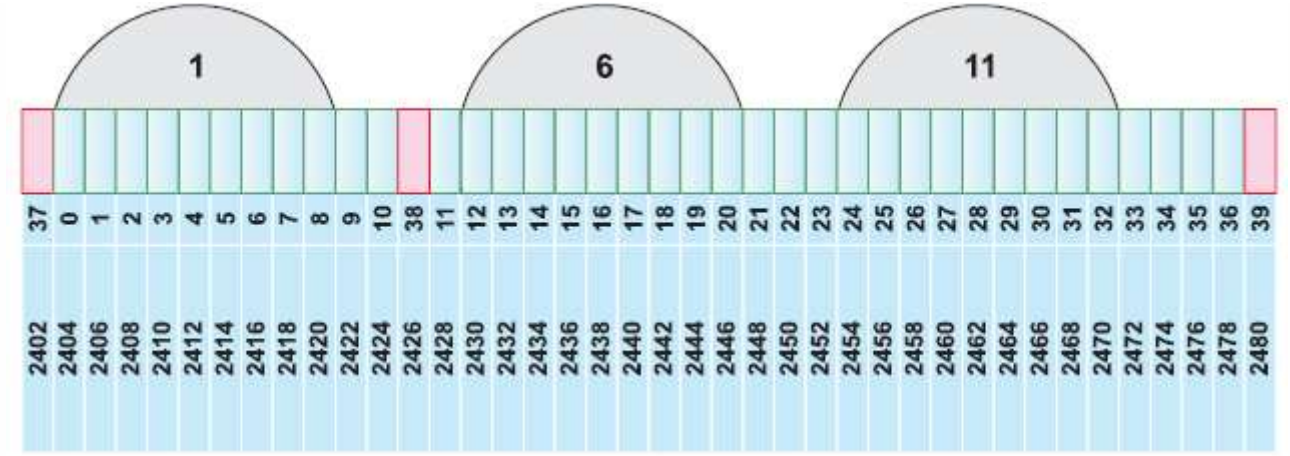

Рис. 2.4. Інтервал оголошень

Інтервал оголошень вибирається на основі компромісу між споживаною потужністю і часом очікування. Більший інтервал дозволяє проводити в режимі очікування більше часу, але при цьому також збільшується період очікування спостерігачем отримання широкомовного пакета.

Спостерігач зазвичай виконує сканування з тривалістю циклів менше 100%, щоб знизити споживання енергії або надати іншим бездротовим протоколам час для роботи. Хороший приклад – смартфони, у яких в більшості випадків є один загальний чіп для *Bluetooth* і *Wi-Fi*®. Якщо навушники з'єднані через класичний *Bluetooth*, а через *Wi-Fi* підтримується зв'язок з точкою доступу, то сканування *Bluetooth Low Energy* буде, ймовірно, проводитися лише на коротких інтервалах роботи. Тимчасові інтервали доступу до ефіру на цьому пристрої розділені між декількома 2,4-гігагерцевий протоколами.

Спостерігач може сканувати в пасивному або активному режимі. Якщо буде використовуватися активний режим і маячок його підтримує, то посилається команда "*Scan Request*", на яку маячок повинен видати *Scan Response*. Запит (*Request*) – це порожній пакет (немає ніяких даних), тоді як

відповідь (*Response*) – це, як правило, статична інформація, наприклад, назва або модель пристрою. Відповідь повністю визначається самим пристроєм. Таким чином, це можуть бути дані з будь-яких датчиків або будь-яка інша корисна інформація. Коли спостерігач сканує в пасивному режимі, він не буде посилати запит сканування.

Таблиця 2.4

Типи даних оголошення, формат даних, специфічних для виробника

Байт	Значення	Опис
0	0*x*03 ... 0*x*1*F*	Довжина цих даних
1	0*xFF*	Прапор даних, специфічних для виробника
2	0*x*0*D*	ідентифікатор компанії
3	0*x*00	Ідентифікатор компанії (наприклад, 0*x*000*D* - *Texas Instruments*)
4 ... 31	-	Визначені користувачем (додаткові) дані

2.1.3. Принципи використання систем живлення в маяках

Маячок може отримувати енергію декількома шляхами. Є три різних основних способи:

– джерело живлення постійної доступності (*USB*, електромережу і так далі);

– батарейки (*CR*2032, *AAA*, літієва і так далі);

– збір енергії – сонячної, кінетичної тощо (*energy harvesting*).

Як правило, спочатку вибирають батарейки, часу життя яких досить для більшості додатків, і які дозволяють випускати невеликі і бюджетні продукти. Можна також використовувати акумуляторні батареї, причому в деяких додатках – разом з бездротовою зарядкою. Вибір типу батарейки важливий, тому що деякі батарейки можуть погано працювати в режимі віддачі максимального струму. Ємність батареї вибирається на основі того, як часто необхідна передача і чи потрібно подальша обробка (читання датчика,

алгоритми обробки отриманих даних і так далі). Взаємодія з датчиком зазвичай має на увазі послідовну передачу даних з використанням інтерфейсів *UART, SPI* або *I2C,* що вимагає додаткової енергії, яка може виявитися навіть більше, ніж при виконанні радіообміну по протоколам *Bluetooth Low Energy*.

Збір енергії реалізується частіше для бездротових проектів з низьким енергоспоживанням, і маячок може бути оснащений джерелом збору енергії. Механічний тиск і сонячна енергія – найвідоміші джерела. Навіть світло в приміщенні [4] може використовуватися для живлення маячка.

Зірка - *star* (проста топологія). Кожен сенсор встановлює зв'язок і веде обмін тільки зі своїм батьком, яким є центральний вузол мережі – *PAN* координатор мережі *LR_WPAN*. Ця топологія реалізується тільки за допомогою стандарту *IEEE* 802.15.4. Усі вузли чують лише координатор *PAN* і тому можуть звертатися один одному тільки через нього. У центрі зірки - координатор *PAN*: центр збору даних.

Дерево - *tree* (складніша топологія), при якій у кожного вузла м би. один батьківський і декілька дочірніх вузлів. Тут також використовується один *PAN* координатор, але є і інші вузли-координатори.

Комірка - *mesh*. Комірчаста топологія пропонує альтернативні варіанти вибору маршруту між вузлами. Тут також використовується один *PAN* координатор мережі, усі інші вузли ідентичні. Пакети транслюються від вузла до вузла, поки не досягнуть кінцевого одержувача. Можливі різні шляхи проходження пакетів, що підвищує доступність мережі у разі виходу з ладу тієї або іншої ланки.

Point-to-point. Вузли у великих мережах з топологією кластерне дерево також і пов'язані між собою (без використання координатора), що дозволяє будувати масштабовані системи.

Point - to - multipoint. Вузол по черзі або широкомовно передає пакети усім вузлам-сусідам з таблиці сусідів. Ця топологія – окремий випадок топології зірка при взаємодії двох вузлів і також містить координатор *PAN*. Це означає, що один з двох вузлів д. би. ведучим *master,* інший веденим *slave*.

Взаємодія цих вузлів відбудеться лише за двох умов:

1) у провідного вузла має бути адреса *PAN,* 2) адреса веденого вузла "прописана" в пам'яті у провідного вузла.

Кластерне дерево - *claster_tree*. В порівнянні із зіркою вводяться ретранслятори пакетів з базовою функцією маршрутизації пакетів, які також здійснюють контроль маршрутів в мережі, вибір часу ретрансляції залежно від завантаженості радіоефіру. Вузли в цій топології самі розраховують або прокладають маршрут, залежно від методу маршрутизації. Якщо вузол, вказаний в маршрутній карті, або вийшов з ладу, або сильно погіршав зв'язок з ним, або виник затор, він може змінити маршрут на один з резервних або вибрати інший ретранслятор в області радіовидимості.

Асоціація виконується автоматично. Для цього координатор *PAN* використовує кадр маяка, який розпізнає будь-який вузол. Вузол-кандидат, що отримав кадр маяка, може звернутися до координатора *PAN*, видає запит на приєднання до мережі. Якщо координатор *PAN* дозволить цьому вузлу асоціацію, він присвоїть йому статус координатора і додасть новий координатор в якості дочірнього координатора у свій список сусідів, передавши йому у відповідь номер слоту доступу і адресу "*MAB*".

Потім тільки що асоційований координатор додасть *PAN* координатор, як свого батька, у свій список сусідів, і почне передавати маяки; інші кандидат-координатори потім приєднуються до мережі. Якщо початковий *PAN* координатор не може підключити вузол-кандидат до мережі, останній шукатиме іншого батька.

У кожного вузла є декілька сусідів. Кожного з них він чує і може вести з ним обмін кадрами. Група сусідів утворює осередок *mesh,* в якій усі можуть обмінюватися один з одним. Якщо вузли не сплять постійно і слухають своїх сусідів, не економлячи енергоспоживання, то мережа з багатьох осередків може мати властивість *Self_healing* -самовідновлення - виявлення і виправлення несправності в роботі вузлів і зв'язків між вузлами без участі людини. Ця здатність заснована на перевірці якості ліній зв'язку між вузлами завдяки

постійному прослуховуванню ефіру і наявності функції *LQI*. У мережі доставка даних не координується з центру - шлюзу *MESH BLE_Gate*, а здійснюється за принципом ретрансляції від свого сусіда до його сусіда в спеціально виділених для цього слотах. Мережа м*esh* побудована на базі топології "кожен-з-кожним" *с* децентралізованою доставкою даних. Роль координатора *PAN* зводиться лише до функції першого вузла мережі, адреси *PAN*, що є носієм. Для передачі даних по мережі координатора *PAN* не потрібно. Метод реалізації мережі *LR_WPAN* є предметом технології типу *BlueTooth*. Можливість самоорганізації і самовідновлення мережі на базі топології *Claster_Tree* з осередками *Mesh* дозволяє спонтанно і оперативно створювати мережі. Мережа автоматично формується з великого числа вузлів, розкиданих по території кластера, по "початковому імпульсу" від *PAN* координатора - коли в мережу його сусідам починають поступати від нього кадри маяка. Якщо усі вузли мають автономне живлення, існує проблема обмеженого часу функціонування такої мережі. Зменшується воно і зі збільшенням території охоплення мережею. Оперативність і відсутність витрат на координування є головним критерієм. Спонтанно створювані мережі іменує термін *Ad_Hoc* "випадково" з можливістю розширення в часі і в просторі.

Топологія кластера мережі *MESH-BLE* має дворівневу структуру *Line - Mesh*, яка дозволяє максимально ефективно використовувати виділені 16-ть каналів в смузі частот 2.4-2.5 ГГц, поєднуючи швидкість передачі даних і надійність функціонування мережі. *MESH-BLE* має динамічну мережеву топологію, в якій маршрути передачі даних постійно міняються, але сам вузол мережі встановлений стаціонарно. При додаванні або видаленні вузла мережі відбувається автоматична переконфігурація мережі. Кластер мережі складається з його координатора і асоційованих з ним радіотрактів, до складу яких входять роутери – вузли з включеною функцією маршрутизації. Координатор кластера виконує і функції концентратора – вузла вищого 0-го рівня топології, який може обмінюватися даними з будь-якими вузлами, що входять до складу його мережі.

Роутер є посередник між вузлами мережі для транзиту даних від видалених сенсорів до концентратора. Кожен вузол і сенсор мережі має свою унікальну адресу *MAB*. Сенсор *MESH-BLE* – це вузол без функції маршрутизації; він може бути і одержувачем, і джерелом даних. Будь-який вузол може бути. кінцевим одержувачем даних

MESH-BLE підтримує ланцюжок *line-mesh-tree* топології: сенсори-вузли-радіотракти-кластери-гіпермережа:

1) *line* -топологія: радіотракт має двонаправлений трафік і містить:

- генератор і термінатор суперкадрів, розташовані в крайніх точках радіотракту;

- набір вузлів - роутери і ретранслятори;

- групи асоційованих з вузлами стаціонарних і мобільних сенсорів.

Лінійну топологію визначає структура шахти - тунелі і штреки, або маршрути міського транспорту.

2) *mesh*-топологія: визначає набір радіотрактів, що утворюють кластер мережі – два і більше радіотрактів в межах мережі, що взаємодіють в одному і тому ж фізичному радіоканалі.

Мережевий кластер включає один координатор, який, як правило, виконує і функції концентратора. У *line-mesh* топології координатор відповідає за старт мережі і вибір її ключових параметрів. Використання роутерів дозволяє розширювати мережу. У *line* мережі роутер/ретранслятор пересилає пакети і управляє мережею без прокладення маршрутів, використовуючи метод лінійної маршрутизації з повторним прийняттям. *Line*-мережа застосовує комунікації, орієнтовані на суперкадри. *Mesh*-мережі забезпечують повні *peer-to-peer* зв'язки між радіотрактами, що забезпечуються тільки роутерами. Проект використовує квазі-*mesh* мережа, в якій створення мережі і генерація регулярних суперкадрів відбувається із застосуванням структури *line*-мережі, а обмін даними виконують роутери з логічною структурою *mesh*-сети. Цей проект описує тільки *intra*-сети: зв'язок починається і завершується в одному і тому ж кластері мережі.

MESH-BLE має дворівневу топологію: *line* для організації радіотракту мережі в тунелі або штреку з підключенням сенсорів і квазі-*mesh* для з'єднання радіотрактів між собою. Вузол використовується як точка ініціації, завершення і маршрутизації зв'язку для мережевих комунікацій. Координатор кластера є основним контролером в мережі. Усі вузли в мережі з будь-якою топологією мають унікальну адресу, що складається з двох частин: 16-бит ідентифікатор *MESH-BLE* мережі і власне 16-біт коротка "Мережева Адреса Вузла - *MAB*" в цій мережі, використовуваний для прямого спілкування у рамках одного кластера, і кластера, що виділяється координатором, коли вузол входить до складу мережі (процедура асоціації, оординатором, що проводиться). Сенсори також мають власний *MAB*, призначений йому вузлом-батьком під час асоціації. Мобільні сенсори в мережі асоціюються як в просторі, так і в часі з будь-яким вузлом за наявності ресурсів (слотів і *MAB*), а стаціонарні сенсори - тільки в просторі.

Топології *line-mesh* мають тільки координатори і роутери, при цьому будь-який вузол може спілкуватися з будь-якими іншими сусідніми вузлами (до 14-ти в цьому проекті), якщо вони знаходяться у сфері охоплення один з одним. Однорангова мережа *mesh* являється вузькоспеціалізованою мережею з самовідновленням і дозволяє застосовувати т. н. хопи для маршрутизації повідомлень по методу "маршрутизація_від_джерела" від будь-якого вузла до будь-якого іншого вузла в мережі. Хоп (стрибок, стрибок) є одиницею виміру довжини маршруту в числі проміжних вузлів, включаючи також вузол призначення. У проекті з топологією *line - mesh* застосований на мережевому рівні тільки метод маршрутизації від джерела з повторним прийманнями і обхідними шляхами Кожен кластер мережі має унікальний 16-бітовий ідентифікатор *PAN ID*, який дозволяє вузлам зв'язуватися в мережі з використанням 16-бітових *MAB* і дає можливість передачі між вузлами через кластери гіпермережі.

Механізм вибору ідентифікатора і створення мережі проводиться на *NWK* рівні.

Охоплення мережі може розширюватися за межі *MESH-BLE*. Виразно вираженої області охоплення не існує для радіосередовища, тобто параметри поширення – динамічні і невизначені. Невеликі зміни місця розташування або напряму можуть привести до великої зміни сили сигналу або якості сеансу зв'язку. Ці ефекти відбуваються незалежно від пристрою - нерухомого або мобільного, т. до. переміщення об'єктів може впливати на поширення сигналу від вузла до вузла.

Нижче представлений короткий опис мережі з топологією кластерів "*line-mesh*".

Використовується комбінована топологія датчикової мережі: зірка – двонаправлена радіошина – комірка. Мережа складається з безлічі вузлів, причому один з них виступає в ролі координатора мережі.

Приклад топології кластера мережі *line - mesh* як варіант однорангової мережі *mesh*, де більшість вузлів - роутеры/ретранслятори. Мобільний сенсор-нащадок підключається до кластера в якості тимчасового елементу і він не дозволяє іншим вузлам спілкуватися через нього. До складу вузла-роутера стека ПО мережевого рівня входить програма маршрутизації.

Після активізації 1-го вузла якого-небудь тунеля (штреку) і привласнення йому статусу генератора суперкадру він може створити свій власний радіотракт мережі. Вузол-генератор забезпечує функцію автономного створення радіотракту мережі і синхронізації його вузлів за допомогою своїх суперкадрів. Вузли-генератори в інших тунелях, у свою чергу, створюють свої радіотракти. Можливий режим роботи, коли суміжні радіотракти створюються і працюють на різних радіоканалах 11-26. Один з координаторів є загальним *PAN* координатором з великими обчислювальними ресурсами, чим будь-який інший координатор. Цей координатор оголошується в якості координатора кластера, внаслідок того, що він став першим координатором, що встановив зв'язок в кластері. Базова структура *line-mesh* має вибраний *PAN ID*, зараз не використовуваний будь-якою іншою мережею в межах радіосфери впливу.

Після того, як призначений мережі *PAN ID* вибраний (механізм вирішення конфліктів *PAN ID* не вимагається, т. до. у мережі відсутні координатори з ідентичними *PAN ID*), координатор кластера формує 1-й радіотракт, дозволяючи іншим радіотрактам увійти до складу свого кластера і передаючи широкомовні кадри створення кластера до сусідніх радіотрактів. Далі 1-й радіотракт підключає до себе інші суміжні радіотракти, якщо вони раніше були вже незалежно створені своїми вузлами-генераторами. Структура мережі будується за схемою *mesh* і відображає топологію тунелів і штреків при формуванні мережі.

Процедура асоціації радіотрактів в кластер подібна до асоціації вузлів *PAN* мережі і розглядається нижче. Створення мережі *mesh* відбувається за розкладом хвилями від центру до периферії від координатора 1-го радіотракту, яке упорядковує процес створення мережі. Радіотракти топології *line* одного і того ж кластера не залежать один від одного і кожним з них управляє координатор радіотракту. У топології *mesh* кожен вузол спілкується тільки з сусідніми вузлами у рамках своєї радіосфери впливу, створюючи таблицю сусідів.

MESH-BLE - протокол передачі даних для створення безпровідних систем моніторингу і управління. Протокол *MESH-BLE* реалізований на базі ядра реального часу *MESH-BLECore* і апаратно незалежний, на відміну від *BlueTooth*. Він дозволяє створювати радіомережі датчиків і старанних модулів в діапазоні 2.4 Мгц. Досягається це завдяки модульної структури ПО, для переходу на іншу елементну базу необхідно тільки перепідключити модуль, що відповідає за фізичний рівень протоколу.

У рамках протоколу *MESH-BLE* реалізовані наступні механізми взаємодії вузлів:

– конфігурація;

– самоорганізація і самовідновлення;

– маршрутизація;

– контроль присутності вузлів в мережі;

– перехід на резервний маршрут

– аутентифікація;

– гарантована доставка пакетів даних.

Гіпермережа формується з декількох сусідніх кластерів, наприклад, кластер утворює мережу одного горизонту шахти, а гіпермережа – інтеграція кластерів в межах однієї шахти. Як тільки заздалегідь встановлені Застосування або мережеві вимоги виконані, перший *PAN* координатор може доручити одному з граничних координаторів, щоб стати *PAN* координатором нового кластера, прилеглого до першого.

2.2. Принципи використання низькочастотних пристроїв в системах передачі інформації

При використанні низькочастотних пристроїв в системах передачі інформації необхідно ще на етапі проектування вирішити ряд питань, невирішення яких може привести до зниження продуктивності або виходу з ладу всієї системи.

2.2.1. Методи визначення дальності зв’язку

Одним з факторів, тісно пов’язаних, а в деяких завданнях і впливають на топологію мережі і вибір протоколу передачі даних – це граничне відстань, на які можна рознести вузли в мережі при збереженні стійкого зв’язку і необхідної швидкості обміну даними (хоча б на мінімальних значеннях, що влаштовують додаток).

Відстань між вузлами можна оцінити виходячи з відомостей про вихідної потужності передавача, чутливості приймача і характеристик антен з урахуванням емпіричних відомостей.

В реальній ситуації розрахована дальність передачі буде трохи нижче через різних ефектів поширення сигналу (розсіювання, дисперсія, багатопроменеве поширення і ін.). Оскільки врахувати всі або хоча б частину ефектів практично не реально, користуються емпіричними правилами (правила наближеного рахунки), що дозволяють зробити необхідні оценк.

Найбільш важливими факторами для вузлів мережі є характеристики антени – коефіцієнт посилення, діаграма спрямованості, чутливість до предметів в ближній зоні і ін.

2.1.2. Аналіз методыв визначення енергоспоживання вузлів мережі передачі даних

Енергоспоживання вузлів мережі є важливим для систем з автономним живленням, а також в рамках загальної тенденції до переходу на енергозберігаючі технології.

Крім технічних характеристик мікросхем приймачів, мікроконтролерів і інших вузлів бездротових модулів на енергоспоживання істотно впливає режим роботи мережевого додатки, інтенсивність обміну даними (рис. 2.5).

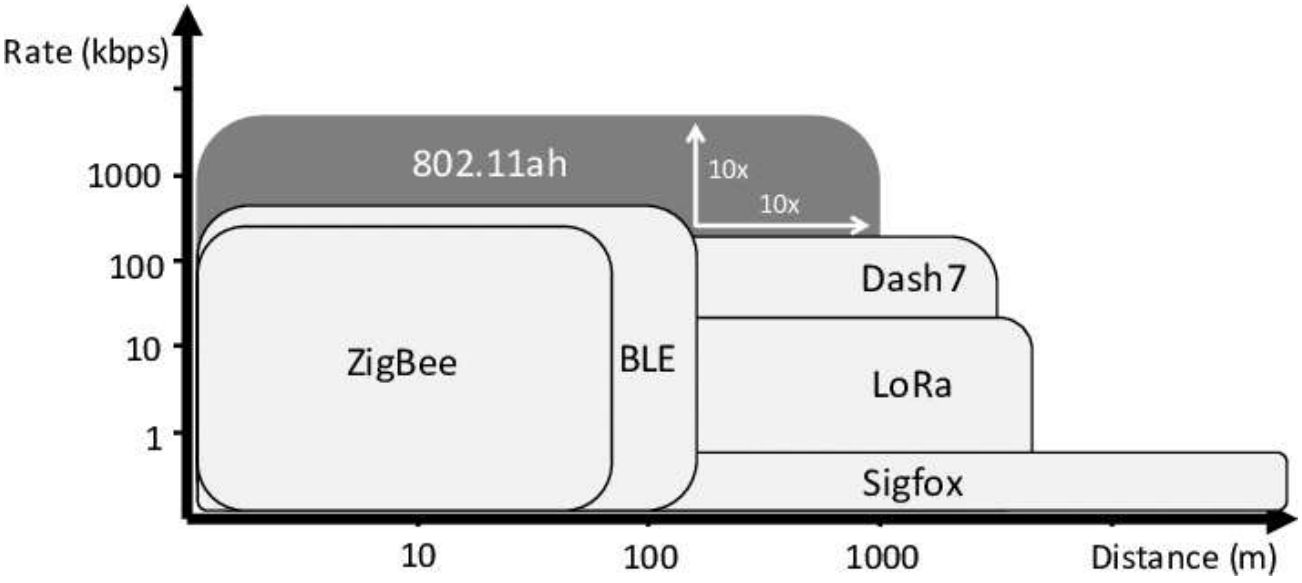

Рис. 2.5. Дальність зв’язку, що забезпечується різними бездротовими технологіями

Виділяють режими роботи з інтенсивним робочим циклом і з малою інтенсивністю обміну. У додатках з інтенсивним робочим циклом основна частка енергоспоживання припадає на радиоинтерфейс – прийом / передача пакетів, синхронізація і автоподстройка частоти. При цьому, в разі переважання в трафіку довгих пакетів, домінує споживання приймача, в разі переважної передачі коротких пакетів на перший план виходить споживання схем ініціалізації родючості і автокалибровки частоти [5].

У додатках з малою інтенсивністю обміну починають грати роль такі показники, як наявність і ефективність режимів зниженого енергоспоживання мікросхем датчиків, мікроконтролерів і приймачів (рис. 2.6).

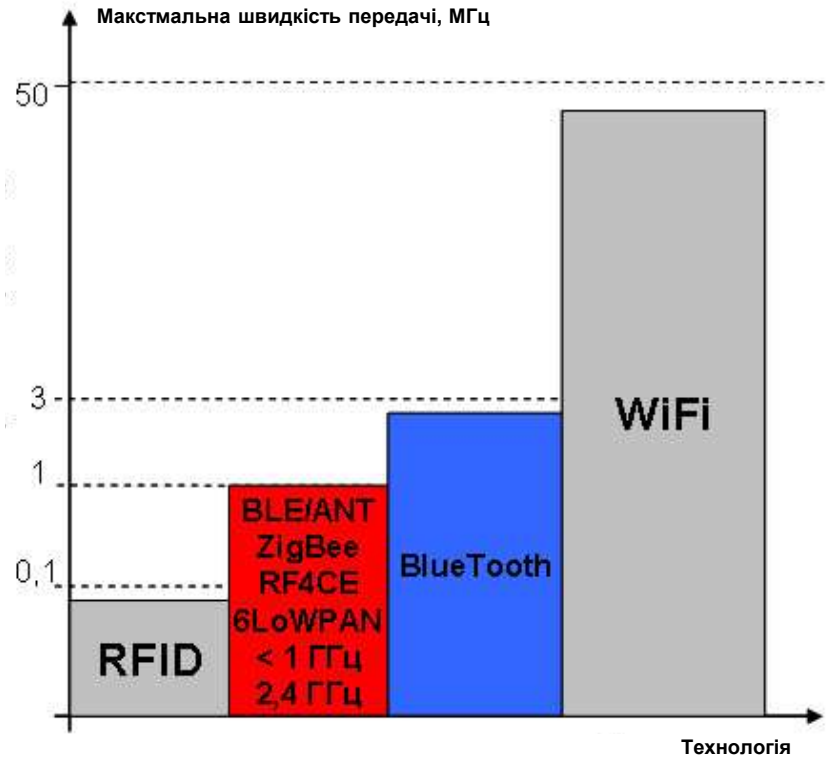

Рис. 2.6. Діапазон швидкостей передачі даних в бездротових технологіях

Типовий профіль енергоспоживання бездротового вузла представлений на рис. 2.7 (абсолютні величини наведені для пристрою діапазону менше 1 ГГц, для пристроїв діапазону 2,4 ГГц струми споживання будуть приблизно в два рази вище).

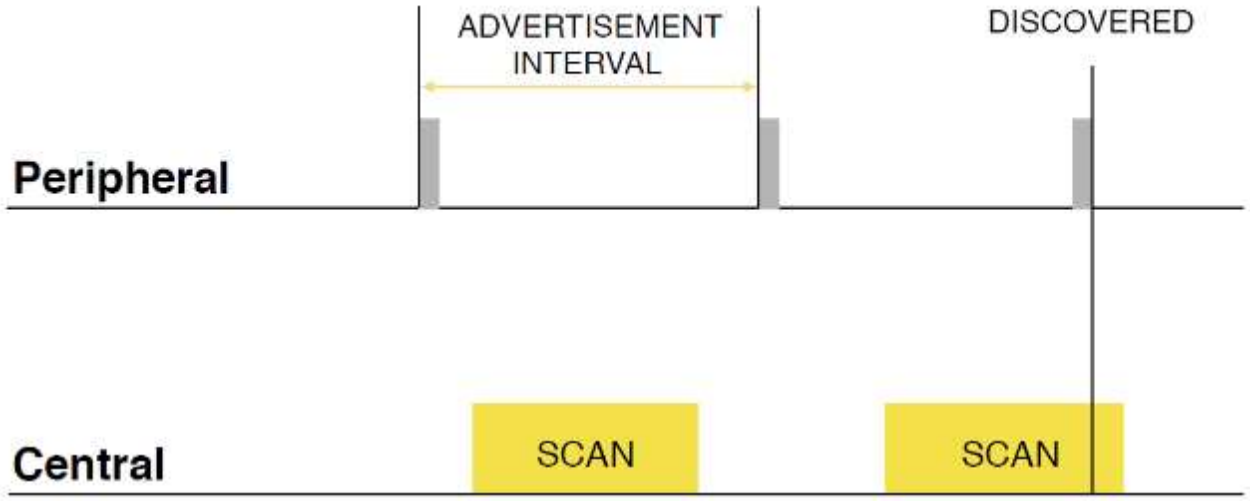

Рис. 2.7. Профілm енергоспоживання бездротового вузла

При цьому відносний рівень споживання пристроїв, що відрізняються за технологіями реалізації можна оцінити по рис. 2.8.

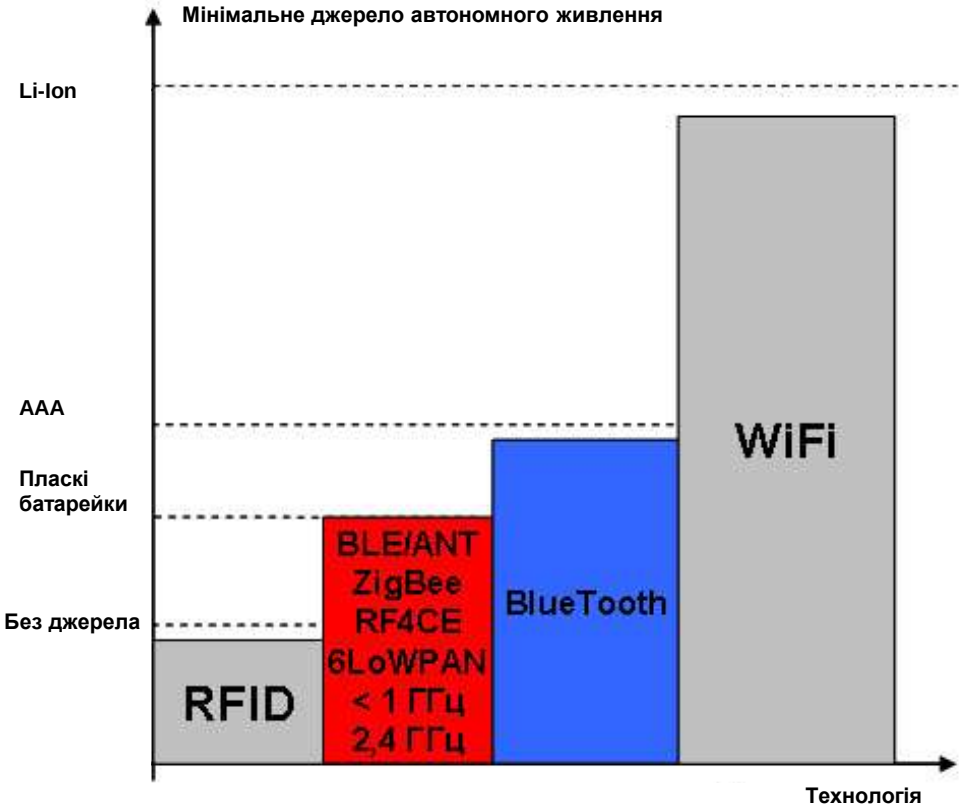

Рис. 2.8. Мінімальні вимоги до джерел живлення бездротових вузлів різних технологій

Найчастіше виробниками пропонуються декілька видів продуктів з низьким енергоспоживанням для бездротових систем з усім необхідним програмним і апаратним забезпеченням. Фактично це позбавляє розробників від необхідності застосування спеціалізованих протоколів для зниження енергоспоживання – ця частина проблеми вирішується на рівні компонентів.

2.2.3. Вибір частотного діапазону

Для обміну даними у всьому світі надаються неліцензованому радіочастотні діапазони. В Україні для цих цілей виділено частотні діапазони 433.075 – 434.750 МГц і 868,7-869,2 МГц, 915 МГц, 2,45 ГГц, 5,8ГГц. Ці частоти можуть використовуватися без оформлення спеціального дозволу ГКРЧ і абсолютно безкоштовно за умови дотримання вимог по ширині смуги, випромінюваної потужності (до 10 мВт в районі частоти 434 МГц і до 25 мВт в районі частоти 868 МГц, до 100 мВт в діапазоні 2,4 ГГц) і призначенням радіосигнали вироби.

Крім законодавчих критеріїв, при виборі частотного діапазону слід враховувати і технічні фактори.

Для діапазону 2,400-2,4835 ГГц є більше частотних каналів, доступні більш високі швидкості передачі, можливий безперервний режим роботи (для родючості), більш компактні антени. З іншого боку, стійка робота реалізується на більш коротких відстанях, зростає вплив різних перешкод (інтерференційні, багатопроменеве поширення, перешкоди).

Для частот менше 1 ГГц дозволені частотні діапазони в різних країнах можуть відрізнятися і не завжди можливо використовувати одну і ту ж елементну базу. Також не скрізь дозволена безперервна радіоактивність пристрою.

Переваги стосуються кращої дальності стійкої роботи в порівнянні з частотами 2,4 ГГц діапазону, при однаковій вихідної потужності передавача, зменшення впливу перешкод на проходження сигналу (особливо актуально для

роботи всередині будівель і офісних приміщень).

На сьогоднішній день проглядається явна тенденція до зростання популярності і повсюдного впровадження пристроїв, що використовують у своїй роботі радіоканал. Число застосовуваних бездротових пристроїв неухильно зростає. З іншого боку через цілого ряду обмежень, що стосуються особливостей формування та прийому радіохвиль, особливостей поширення радіохвиль, а також законодавчих обмежень на діапазони доступних для приватного і комерційного використання частот, вибір частотного діапазону для того чи іншого додатка вельми обмежений. Отже, щільність бездротових пристроїв і ступінь використання будь-якого з діапазонів частот будуть зростати.

Через характеру зростання числа бездротових пристроїв, близькому до експоненціального, розробники змушені вже зараз піклується про стабільну роботу своїх пристроїв і стійкості зв’язків між ними, так як цілком вірогідне зростання власної бездротової мережі, додавання все нових сервісів. Ймовірно також поява в межах дії мережі інших бездротових вузлів або систем.

Продуктивність бездротової мережі в таких умовах буде визначатися ймовірністю успішного прийому пакетів мережі і її керуючих сигналів.

Основні фактори, що впливають на продуктивність бездротової мережі:

– вихідна потужність передавача

– чутливість приймача;

– вибірковість;

– придушення сигналів сусідніх частотних смуг.

Вихідна потужність передавачів для неліцензованому діапазонів частот обмежується регулюючими документами. При необхідності, шляхом застосування зовнішнього підсилювача, потужність вихідного сигналу можна встановити рівний гранично допустимої. У ряді випадків додатково, або замість підсилювача можливе застосування спрямованої антени. Чутливість приймача обмежують технологічні обмеження, власні шуми електронних елементів, взаємний вплив вузлів. Крім цього реальна чутливість приймача може бути

знижена через помилки у виконанні антенно-фідерного тракту. Знову ж зовнішній антенний підсилювач або спрямована антена можуть кілька врятувати становище, правда, швидше за все, в бюджет каналу додасться не більше +20 дБм, найчастіше не більше 10.

В умовах щільного використання частотного діапазону і щільного розміщення пристроїв, все більше вирішальну роль будуть грати здатність приймача фільтрувати (пригнічувати) сигнал сусідніх частот, і вибірковість – здатність придушити сигнальну перешкоду сусіднього (або близького) частотного каналу.

2.2.4. Джерела перешкод в радіомережах мережах

1) Аналогові бездротові телефони

Аналогові бездротові телефони є класичним джерелом перешкод для бездротових локальних мереж стандарту 802.11 (*WLAN*). На відміну від цифрових бездротових телефонів, аналогові бездротові телефони використовують узкополосную передачу, коли передається сигнал займає тільки вузьку смугу частот радіочастотного спектру. Через це такі телефонні апарати можуть надавати серйозні перешкоди точки доступу 802.11, що працює на тому ж каналі або частоті, в той же час, не надаючи значних перешкод точки доступу, що працюють на інших неперекривающіхся каналах.

Одне з лабораторних досліджень показало, що аналоговий радіотелефон, який здійснює передачу на частоті 2,412 ГГц, здатний в момент включення телефону поруч з точкою доступу ефективно перешкодити роботі бездротового з'єднання по цьому каналу. У той же час з'єднання на двох інших неперекритих каналах (6 і 11) були ледь порушені.

Дослідження також дозволило виявити, що пропускна здатність мережі може знижуватися на 99%, коли аналоговий бездротовий телефон знаходиться на відстані 15 метрів від точки доступу, на 20% при відстані в 30 метрів і на 5% при відстані 45 метрів.

У дослідженні зроблено висновок, що якщо аналогові бездротові телефони розташовані близько до точки доступу, то можуть істотно вплинути на бездротовий зв'язок по каналу, на якому вони працюють.

Різними виробниками випускається безліч моделей аналогових бездротових телефонів. Вони широко використовується в будинках і офісах, де також розгорнуті бездротові мережі стандарту 802.11. Щоб усунути вносяться аналоговими бездротовими телефонами перешкоди, спочатку необхідно ідентифікувати і визначити їх місцезнаходження в бездротовій локальній мережі.

2) Радіоняня (монітор спостереження за немовлям). У бездротових моніторах стеження за немовлям (цифрових або аналогових) для передачі сигналу використовується діапазон радіочастот. Цей же радіочастотний діапазон використовується встановлюються в житлових приміщеннях бездротовими мережами. В результаті, коли в одному і тому ж радіохвиль працюють дві конкуруючі системи, виникають радіочастотні перешкоди.

Багато наявні в даний час на ринку бездротові монітори стеження за немовлям використовують частотний діапазон 2,4 ГГц. На рис. 2.10 показана характеристика радіочастотного спектру цифрового монітора стеження за немовлям, що працює на частоті 2,4 ГГц.

Перш за все, коли монітор спостереження за немовлям не використовується, радіочастотні перешкоди відсутні. Однак коли він працює, то може чинити негативний вплив на мережу стандарту 802.11, особливо коли вони знаходяться в безпосередній близькості один від одного. Включений монітор конкурує за смугу пропускання з бездротовою локальною мережею, яка використовує той же радіочастотний діапазон, що викликає зниження пропускної здатності бездротової мережі в результаті впливу радіочастотних перешкод. І такий вплив є взаємним. Вплив більш очевидно при використанні веб-додатків, пов'язаних із завантаженням файлів або переглядом потокового відео.

3) цифрові бездротові телефони. Багато наявних на сучасному ринку

цифрові бездротові телефони працюють в частотному діапазоні 2,4 ГГц або 5,8 ГГц, який також використовується каналами або частотами бездротових локальних мереж стандарту 802.11. Проблема полягає в тому, що це дві абсолютно різні системи, які не розуміють один одного. В результаті радіосигнали від двох різних систем будуть передаватися одночасно, надаючи взаємні радіочастотні перешкоди. Більшою мірою це відноситься до випадку, коли використовуються цифрові мобільні телефони діапазону 2,4 ГГц з технологією *FHSS* (*Frequency Hopping Spread Spectrum* – Псевдослучайная перебудова робочої частоти). При використанні модуляції *FHSS* радіочастотні сигнали цих телефонів перескакують з однієї частоти на іншу в усьому частотному діапазоні 2,4 ГГц. Таке стрибкоподібне «поведінку» буде надавати стійкі радіочастотні перешкоди розташованої в безпосередній близькості бездротової локальної мережі стандарту 802.11. Подібні джерела перешкод можуть викликати суттєві збої в роботі бездротових локальних мереж і знижувати їх пропускну здатність.

Протягом довгих років було випущено величезну кількість цифрових бездротових телефонів. Вони широко використовуються в будинках і офісах, і також є джерелом радіочастотних перешкод, що впливають на роботу бездротових локальних мереж стандарту 802.11.

Існує безліч цифрових бездротових телефонів діапазонів 2,4 / 5 ГГц, які випускаються різними виробниками. Вони широко використовується в будинках і офісах, де розгорнуті бездротові локальні мережі стандарту 802.11. Щоб вирішити проблему з перешкодами від бездротових телефонів діапазону 2,4 / 5 ГГц, необхідно спочатку ідентифікувати і визначити їх місцезнаходження в своєї бездротової мережі.

4) Бездротові камери і цифрові відеомонітори, як правило, складаються з трьох компонентів – відеокамери, передавача для передачі сигналу і приймача для прийому сигналу. Система працює наступним чином – відеосигнал передається з вбудованого передавача бездротової камери на приймач, який підключений до пристрою відображення (монітора) або записуючого пристрою.

Багато бездротових камер і цифрові відеомонітори працюють на частоті 2,4 ГГц.

Подібно до інших пристроїв, які працюють в діапазоні частот 2,4 ГГц, але не є пристроями *WiFi,* встановлені в безпосередній близькості від бездротової локальної мережі стандарту 802.11 бездротові камери і цифрові відеомонітори можуть надавати перешкоди нормальній роботі бездротової мережі. На відміну від інших джерел радіочастотних перешкод, які працюють в частотному діапазоні 2,4 ГГц, радіосигнали від передавача бездротової камери або цифрового відеомонітора в залежності від фізичних умов можуть передаватися на відносно велику відстань (від 60 до 210 метрів при прямої видимості). Зазвичай для того, щоб забезпечити повний перекривається огляд всієї зони спостереження, необхідно кілька камер. І що ще гірше, встановлені в будинках і офісах бездротові камери і цифрові відеомонітори залишаються включеними постійно. Таким чином, вони надають постійні радіочастотні перешкоди знаходяться поруч із бездротовими локальними мережами 802.11.

2.2.5. Оцінка часу і ресурсів на розробку

На поточний момент часу існує два шляхи реалізації апаратних рішень для бездротових систем.

Перший шлях полягає в розробці власної плати пристрою з урахуванням рекомендацій і схем включення елементів.

Типовий перелік елементів включає в себе:

– мікросхему бездротового приймача і мікроконтролер або однокристальні пристрій;

– компоненти радиотракта – антена (друкована / чіп / зовнішня), роз’єми для підключення зовнішньої антени, розв’язують конденсатори, антенний фільтр, в деяких випадках антенний підсилювач;

– стабілізатор живлення або батареї живлення.

Другий шлях полягає у використанні готових модулів і микросборок,

інтегруючих на платах для поверхневого або мезонінного монтажу всі елементи радиотракта, включаючи приймально-передавальні елементи.

У разі використання фірмових бездротових модулів розробник отримує можливість скоротити час виходу кінцевого продукту на ринок, позбавлений розробки топології друкованої плати для високочастотної частини, всі компоненти радиотракта узгоджені між собою, і весь модуль в цілому має узгоджені параметри по температурному діапазону, потужності сигналу та ін .

При невеликих партіях виробів використання готових модулів призводить до скорочення витрат на одиницю продукції.

Розробка бездротового рішення включає в себе вивчення документації на вибрані електронні компоненти, розробку антени, друкованої плати та програмного забезпечення із застосуванням засобів розробки для обраного апаратного рішення.

2.3. Життєвий цикл розробки бездротових систем

Кожна область застосування бездротових систем характеризується наявністю певної кількості вузлів і призначене для вирішення певного класу задач або надання користувачам певних сервісів.

Як правило, кількість вузлів залежить від числа агентів, виявлених в результаті декомпозиції задачі, і може варіюватися від двох до декількох тисяч. При декомпозиції завдання аналізуються потоки даних між агентами, з’ясовується їх спрямованість, інтенсивність, визначається розташування кінцевих вузлів мережі і вимоги до їх мобільності.

Розробнику доступні як пропрієтарні рішення, так і рішення, засновані на міжнародних стандартах, потенційно розширюють можливий ринок споживання кінцевих пристроїв і гарантують сумісність і спільну роботу систем – в частотності підтримка стандарту 802.15.4 і протоколів на його основі *ZigBee* і 6*LoWPAN*.

Кожна з топологій вимагає, природно підтримки у вигляді мережевого

додатки або протоколу.

Фактично, кожна прикладна задача в області бездротових систем представляється як окремий проект зі своїми етапами розвитку і життєвим циклом.

Типовий цикл розробки для бездротових систем наступний:

1. Визначення вимог:

– кількість вузлів мережі;

– аналіз і вибір необхідної топології мережі;

– середня відстань між вузлами;

– діапазон необхідних швидкостей передачі даних;

– енергоспоживання вузлів і джерела їх живлення (стаціонарне живлення, автономні пристрої);

– типове час роботи вузлів в автономному режимі.

2. Вибір технологій реалізації:

– адаптація типових рішень або розробка "з нуля";

– вибір або розробка стека протоколів;

– приведення у відповідність з правовими, технічними та медичними нормами;

– вибір між використанням модулів і микросборок і самостійної розробки вузлів на основі наборів мікросхем – оцінка часових і грошових витрат.

3. проектування:

– вибір лінійки продуктів для побудови рішення;

– вибір антен;

– розробка друкованої плати;

– вибір засобів розробки програмного забезпечення мережевого додатки;

– підтримка розробки – симуляція, налагоджувальні плати, ескізний проект.

4. Тестування програми:

– сертифікація;

– перевірка спільної роботи з іншими бездротовими системами або

забезпечення сумісності з ними;

– тестування роботи програми.

5. виробництво:

– планування життєвого циклу продукту і його супровід;

– система контролю якості.

Висновки за розділом

Принцип роботи *BLE* описаний вже в його назві: *Low Energy*. Протокол передбачає передачу даних короткими пакетами по необхідності, потім – вимкнення передавача. Низький рівень споживання енергії частково досягається застосуванням саме цього принципу. Замість класичного тандема в звичайному *Bluetooth*, пристрої *BLE* зв'язуються один з одним лише при необхідності відправки або отримання інформації.

Протокол *BLE* строго структурований за принципом своєї комунікації з іншими пристроями. Спочатку девайси вивчають доступні сервіси для відправки / прийняття даних; невід'ємна частина цих сервісів – їх характеристики (*characteristics*), що визначають тип даних для майбутньої передачі. Характеристики, з міркувань наочності, можуть мати в своєму складі опису-дескриптори (*descriptors*), які допомагають визначити тип даних. Наприклад, розберемо сервіс під назвою «*Heart Rate Monitor*» (монітор частоти серцебиття) – серед його характеристик присутні такі, як «вимір пульсу».

Більшість *API* для *Bluetooth LE* дозволяють шукати локальні пристрої та визначати доступні в них сервіси, характеристики і дескриптори.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ ВИКЛИКУ МЕДИЧНОГО ПЕРСОНАЛУ

3.1. Опис принципів роботи системи виклику медичного персоналу

Система виклику медперсоналу, по суті, є палатної сигналізацією і призначена для організації пошукового виклику персоналу, двостороннього зв'язку між персоналом і хворими, виклику чергових медсестер.

Залежно від розв'язуваних завдань дана система може включати в себе:

– переговорні модулі;

– центральний блок опитування;

– світлові покажчики виклику;

– модулі з кнопкою виклику.

Подібні системи забезпечують наступні функції:

– організацію двостороннього зв'язку між персоналом і пацієнтом;

– організацію двостороннього зв'язку між черговим лікарем і медсестрою;

– виклик чергової медсестри хворим з душової кабіни, палати, туалету;

– можливість об'єднання з системами радіотрансляції і колективного прийому телебачення з забезпеченням вибору програм з пульта системи виклику медперсоналу;

– відображення виклику світловими покажчиками.

В межах дипломного дослідження було обрано реалізацію спрощеної системи виклику медичного персоналу, яка передбачає зв’язування мережі кнопок виклику з центральним модулем, який перенаправляє повідомлення на телефони медичних співробітників.

3.2. Проектування пристрою *Bluetooth Low Energy*

Як і при будь-розробці, тут також є параметри, які можуть бути оптимізовані з урахуванням різних плюсів і мінусів. Один із прикладів –

інтервал оголошень. Вибір меншого інтервалу, тобто часта передача, збільшує ймовірність того, що інформація буде швидше доходити до пристрою спостерігача, хоча споживана потужність при цьому збільшується.

3.2.1. Засоби розробки

При проектуванні маячка спочатку приймається рішення про те, які кошти розробки використовувати. Є кілька доступних налагоджувальних наборів компанії *Texas Instruments,* які представлені на рис. 3.1. Ці засоби включають в себе як невеликі плати з живленням від дискового елемента (*CC*254*XDK-MINI,CC*2541*DK-SENSOR*), так і багатофункціональні платформи, ідеальні для моделювання пристроїв будь-якої складності (*CC*2540*DK*). Детальна інформація про ці кошти розробки знаходиться на сайті *ti.com*/*ble*.

а) б) в)

Рис. 3.1. Засоби розробки *Bluetooth Smart*: а – CC2540DK-MINI, CC2541DK-MINI, б – CC2540DK, CC2541EMK, в – CC2541DK-SENSOR

3.2.2. Створення програми маячка з *BLE*-стеком *TI*

BLE-стек, що поставляється компанією *Texas Instruments* для бездротових *MCU CC*254*x*, Забезпечує просту і надійну реалізацію з’єднуються і несоединяемих маячків.

Є типові додатки, які можуть використовуватися як програмні шаблони

при проектуванні маячка, вони описані в табл. 3.1. Передбачається, що розробник вже знаком із середовищем розробки *IAR Embedded Workbench* і *BLE*-стеком.

Таблиця 3.1

Приклади програмного забезпечення маячка для *CC*254*x*

Приклад проекту	Призначення *GAP*	Тип	Підтримка пристроїв
SimpleBLEPeripheral	периферійний пристрій	сполучається	*CC*2540,*CC*2540*T*,*CC*2541, *CC*2541-*Q*1
SimpleBLEBroadcaster	диктор	непоєднувані	*CC*2540, *CC*2540*T*, *CC*2541, *CC*2541-*Q*1

Є також універсальне типове додаток – широкомовний передавач, спеціально спроектований для*CC*2543і працює в непоєднувані режимі [6]. Для *CC*2543 є зразок розробки для багатофункціонального широковещательного передавача [7].

Проект *SimpleBLEPeripheral* повністю описаний в «Керівництві по розробці програмного забезпечення» [8] і є, в загальному випадку, кращою відправною точкою при реалізації з'єднується маячка. *SimpleBLEBroadcaster* – спрощена версія *SimpleBLEPeripheral*, яка підтримує тільки непоєднувані маячки. *API* підтримує ті ж функції маячка, що і для зазначених вище проектів. Таким чином, наведені нижче приклади, застосовні до обох проектів, зокрема, *SimpleBLEBroadcaster* (*BLEv*1.4), можна використовувати як еталонний приклад проекту. Є дві апаратних платформи для *SimpleBLEBroadcaster*: *CC*2541 і *CC*2541*DK-MINI Keyfob*.

У цій роботі використовується плата *CC*2541*EM*. Конфігурація проводиться за допомогою спливаючого списку робочих просторів (рис. 3.3). Крім того, існує версія проекту для *CC*2540 з подібними вбудованими опціями.

Рис. 3.3. Вбудовані опції для *SimpleBLEBroadcaster*

Додаток реалізовано в *SimpleBLEBroadcaster.c*, де широкомовні дані визначені як *advertData*:

*static uint*8 *advertData* [] =

{// Прапори; вони встановлюють пристрій в режим обмеженого виявлення

// (установка з'єднання один раз протягом 30 секунд) замість загального

// режиму виявлення (оголошення не визначені)

0*x*02, // довжина цих даних

GAP_ADTYPE_FLAGS,

GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED,

// трьохбайтовое оголошення даних 111, 112, 113

0*x*04, // довжина цих даних

GAP_ADTYPE_MANUFACTURER_SPECIFIC, 111, 112, 113

// дані, які визначаються виробником

// тип даних оголошення

};

За замовчуванням передаються дані містять обов'язкові прапори, за якими йдуть три байта даних *Manufacture-Specific Data* (номера 111, 112 і 113). Ці дані можуть бути змінені на будь-які інші дані, при цьому в разі потреби слід оновити довжину даних. Залежно від апаратної платформи оголошення встановлюються по-різному. Для *CC*2541 B*uild*, який є загальним для мікросхем *CC*2541, за замовчуванням ця змінна має значення *TRUE*. У типовому додатку тип оголошення за умовчанням встановлено як константа *GAP_ADTYPE_ADV_SCAN_IND*. Це дозволяє під час з'єднання

використовувати команди *Scan Request*/*Response*. Щоб заборонити сканування, що, до речі, зменшить споживану потужність, тип оголошення (*advType*) може бути змінений на константу *GAP_ADTYPE_ADV_NONCONN_IND*. Після цього зміни можна також не ставити три байта прапорів:

Конфігуровані змінні далі надходять на рівень *GAP* для використання стеком *BLE*. Відзначимо, що режим *advertEnable* не починається миттєво, у всякому разі, це не відбувається під час ініціювання додатки (*simpleBLEBroadcaster_Init*). Оголошення почнуть передаватися після того, як запуститься пакет протоколів:

GAPRole_SetParameter (*GAPROLE_ADVERT_ENABLED*, *sizeof* (*uint*8), & *advertEnable*);

GAPRole_SetParameter (*GAPROLE_ADVERT_DATA*, *sizeof* (*advertData*), *advertData*);

GAPRole_SetParameter (*GAPROLE_ADV_EVENT_TYPE*, *sizeof* (*uint*8), & *advType*).

Інтервал оголошення за умовчанням встановлено рівним 100 мс, хоча він може бути збільшений до 10,24 с, що є максимумом, дозволеним базової специфікацією *BLE*. Якщо необхідні довші інтервали – можна вручну дозволяти і відключати передачу даних за допомогою таймера *OSAL*.

Щоб гарантувати виявлення оголошень, є загальне правило: інтервал оголошень + 10 повинен бути менше, ніж вікно сканування спостерігача. Це означає, що маячок повинен бути спроектований з урахуванням можливостей партнерського пристрою, інакше для отримання переданих пакетів буде потрібно багато часу.

Зауважимо, що код (Додаток А), заснований на *BLEv*1.4 *Manufacturer-Specific Data*, не включає код компанії. Щоб використовувати даний код на інших системах, необхідно додати код компанії до *AdvertData*.

3.3. Підключення ОС *Android* до *BLE*-пристроїв

ОС *Android* (*API* 18) представляє вбудовану підтримку *Bluetooth Low Energy* і *API*, за допомогою якого додатки можуть використовувати пошук пристроїв, запит послуг і читання / запис характеристик. На відміну від класичного *Bluetooth*, *BLE* покликаний забезпечити істотно менше енергоспоживання. Це дозволяє додаткам для *Android* спілкуватися з *BLE*-пристроями, які мають низькі вимоги до живлення, таких як датчики, монітори серцевого ритму, фітнес-пристрої і так далі.

Generic Attribute Profile (*GATT*) – профіль *GATT* є спільною специфікацією для відправки та отримання коротких фрагментів даних, відомих як "атрибути" через *BLE*-з'єднання. Всі поточні *LE*-профілі додатків засновані на *GATT*. Творці *BLE* визначили безліч профілів для низькоенергетичних пристроїв. Профіль являє собою визначення того, як пристрій працює в конкретному додатку. Зверніть увагу, що пристрій може реалізовувати більше одного профілю. Наприклад, пристрій може містити профілі пульсометра і датчика рівня заряду батареї.

Attribute Protocol (*ATT*) – *GATT* будується на основі протоколу атрибутів ATT. Це також відноситься до *GATT* / *ATT*. ATT оптимізований для роботи на *BLE*-пристроях. Для цього він використовує настільки мало байтів, наскільки можливо. Кожен атрибут ідентифікується унікальним універсальним ідентифікатором (*UUID*), який являє собою стандартизований 128-бітний строковий ідентифікатор використовується для однозначної ідентифікації інформації. Атрибути переносяться за допомогою ATT у вигляді характеристик і послуг.

Характеристика (*Characteristic*) – містить одне значення, і від 0 до *N* дескрипторів, що описують значення характеристики. Характеристика може розглядатися як тип, аналог класу.

Дескриптор (*Descriptor*) може містити удобочитаем опис, прийнятний діапазон значень або одиницю виміру, конкретні значення характеристики.

Послуга (*Service*) – це набір характеристик. Наприклад, ви можете мати послугу під назвою "пульсометр", що включає в себе таку характеристику, як "Вимірювання пульсу". Ви можете знайти список існуючих на основі *GATT* профілів і послуг на *bluetooth.org*.

3.3.1. Ролі та обов'язки при взаємодії *Android* з *BLE*-пристроєм

Центральна / периферична роль. Це відноситься до самого *BLE*-з'єднанню. Пристрій в центральній ролі сканує, шукає оголошення, а пристрої в периферійній ролі створює оголошення.

GATT-сервер / *GATT*-клієнт. Це визначає, яким чином два пристрої спілкуються один з одним, коли вони встановили зв'язок.

Щоб зрозуміти різницю, уявіть, що у вас *Android*-телефон і фітнес-трекер, який представляє собою *BLE*-пристрій. Телефон підтримує центральну роль; трекер активності підтримує периферійну роль. Щоб встановити *BLE*-з'єднання, вам потрібно по одному пристрою, що підтримує кожну з ролей; два периферійних пристрої не зможуть спілкуватися один з одним, також як і два центральних.

Як тільки телефон і трекер активності налагодили зв'язок, вони починають передачу метаданих *GATT* один одному. Залежно від того, які дані вони передають, той чи інший може виступати в якості сервера. Наприклад, якщо трекер хоче повідомити дані датчика телефону, трекеру має сенс працювати в якості сервера. Якщо трекер активності хоче отримувати оновлення з телефону, то в якості сервера має сенс використовувати телефон.

Як приклад в даному документі представлено *Android*-додаток, що є *GATT*-клієнтом. Додаток отримує дані від *GATT*-сервера на *BLE*-пульсометра. Але ви можете також спроектувати ваше додаток так, щоб воно відігравало роль сервера.

BluetoothAdapter є обов'язковим для будь-яких дій з *Bluetooth*. *BluetoothAdapter* представляє власний *Bluetooth*-адаптер пристрої (*Bluetooth*-

приймач). Є один *Bluetooth*-адаптер для всієї системи, і ваше додаток може взаємодіяти з ним, використовуючи цей об'єкт.

3.3.2. Підключення до *GATT*-серверу

Першим кроком у взаємодії з *BLE*-пристроєм стане підключення до нього – точніше, підключення до *GATT*-серверу на пристрої. Для підключення до *GATT*-серверу на *BLE*-пристрої потрібно використовувати метод *connectGatt* (). Цей метод приймає три параметри: об'єкт контексту, автосоедіненіе (логічне значення, яке вказує, чи слід автоматично підключитися до *BLE*-пристрою, як тільки воно стане доступним), і посилання на *BluetoothGattCallback*:

mBluetoothGatt = device.connectGatt (*this, false, mGattCallback*);

Воно підключається до *GATT*-сервера, що знаходиться на *BLE*-пристрої, і повертає екземпляр *BluetoothGatt*, який потім можна використовувати для проведення клієнтських операцій *GATT*. Додаток для *Android* є *GATT*-клієнтом. *BluetoothGattCallback* використовується для отримання результатів клієнтом, таких як статус підключення, а також будь-які додаткові клієнтські операції *GATT*.

У цьому прикладі *BLE*-програма надає активності (*DeviceControlActivity*) відображення даних про підключення, *GATT*-послуги та характеристиках, підтримуваних пристроєм. На основі введення користувача, ця активність пов'язується зі службою під назвою *BluetoothLeService*, який взаємодіє з *BLE*-пристроєм через *Android BLE API*-інтерфейс.

3.3.3. Методи читання *BLE*-атрибутів та організація отримання *GATT*-повідомлень

Як тільки *Android*-додаток підключається до *GATT*-сервера і виявляє послуги, він може читати і писати атрибути, де це підтримується.

Це загальні для *BLE*-додатків запити для отримання повідомлень про

зміну окремих показників на пристрої. Цей фрагмент показує, як налаштувати повідомлення для будь-якої характеристики за допомогою методу *setCharacteristicNotification* ():

private BluetoothGatt mBluetoothGatt;

BluetoothGattCharacteristic characteristic;

boolean enabled;

...

mBluetoothGatt.setCharacteristicNotification (*characteristic, enabled*);

...

BluetoothGattDescriptor descriptor = characteristic.getDescriptor (

UUID.fromString

(*SampleGattAttributes.CLIENT_CHARACTERISTIC_CONFIG*));

descriptor.setValue

(*BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE*);

mBluetoothGatt.writeDescriptor (*descriptor*);

Коли повідомлення включені для характеристики, виклик *onCharacteristicChanged* () спрацьовує, якщо характеристика була змінена на віддаленому пристрої:

@Override

// Опис характеристики

public void onCharacteristicChanged (*BluetoothGatt gatt,*

BluetoothGattCharacteristic characteristic) {

broadcastUpdate (*ACTION_DATA_AVAILABLE, characteristic*);

}

3.3.4. Використання *BluetoothManager*

Менеджер високого рівня, який використовується для отримання примірника *BluetoothAdapter* і загального управління *Bluetooth*. Використовуйте *getSystemService* (*java.lang.String*) з *BLUETOOTH_SERVICE* щоб створити

BluetoothManager; після цього викликайте *getAdapter* () для отримання примірника *BluetoothAdapter*. Крім того, можна просто викликати *BluetoothAdapter.getDefaultAdapter* ().

Загальнодоступні методи:

1) *BluetoothAdapter getAdapter*() – отримати стандартний *Bluetooth*-адаптер даного пристрою. Повертає: Стандартний *Bluetooth*-адаптер даного пристрою;

2) *List <BluetoothDevice> getConnectedDevices*(*int profile*) – отримати підключені пристрої зазначеного профілю. Повертає набір пристроїв, які знаходяться в стані *STATE_CONNECTED*. Це не відноситься до будь-якої конфігурації програми, але відображає стан *Bluetooth*-підключення даного профілю. Це може використовуватися в додатках як рядок стану, яка просто хотіла б знати стан *Bluetooth*. Потрібен дозвіл *BLUETOOTH*. Параметри: *GATT* або *GATT_SERVER* Повертає: Список пристроїв. При помилку список буде порожній.

3) *int getConnectionState* (*BluetoothDevice device, int profile*) – отримати поточний стан підключення профілю до віддаленого пристрою. Це не відноситься до будь-якої конфігурації програми, але відображає стан *Bluetooth*-підключення даного профілю. Це може використовуватися в додатках як рядок стану, яка просто хотіла б знати стан *Bluetooth*. Потрібен дозвіл *BLUETOOTH*. Параметри: Віддалене *Bluetooth*-пристрій; *GATT* або *GATT_SERVER*. Повертає: Стан з'єднання профілю, одне з: *STATE_CONNECTED, STATE_CONNECTING, STATE_DISCONNECTED, STATE_DISCONNECTING*.

4) *List <BluetoothDevice> getDevicesMatchingConnectionStates* (*int profile, int*[] *states*) – отримати список пристроїв, які відповідають будь-якому із зазначених станів сполуки. Якщо ні один з пристроїв не відповідає жодному із зазначених станів, буде повернуто порожній список. Це не відноситься до будь-якої конфігурації програми, але представляє стан підключення локального адаптера *Bluetooth* для цього профілю;

5) *BluetoothGattServer openGattServer* (*Context context,*

BluetoothGattServerCallback callback) – відкриває *GATT*-сервер. Використовується для отримання результатів, таких як стан підключення, а також результати будь-яких інших серверних операцій в рамках *GATT*. Метод повертає екземпляр *BluetoothGattServer*.

3.3.5. Використання відкритого *API BluetoothGatt* для *Bluetooth*-профілю *GATT*

Цей клас забезпечує функціональність *Bluetooth GATT* для взаємодії зі смарт-пристроями. Щоб підключитися до віддаленого периферійного пристрою, створюється *BluetoothGattCallback* і викликається метод *connectGatt* (*Context, boolean, BluetoothGattCallback*), щоб отримати примірник цього класу. Пристрої, що підтримують *GATT*, можуть бути знайдені за допомогою звичайного виявлення *Bluetooth*-пристроїв або за допомогою сканування *BLE*.

int CONNECTION_PRIORITY_BALANCED

Оновлення параметра сполуки: використовувати параметри підключення, рекомендовані *Bluetooth SIG*. Це параметр за замовчуванням, якщо оновлення параметрів з'єднання не потрібно. Значення константи: 0 (0*x*00000000) – пріоритет з'єднання: високий:

int CONNECTION_PRIORITY_HIGH

Оновлення параметра сполуки: запит високого пріоритету, низькою затримки підключення. Додаток має запитувати високий пріоритет з'єднання тільки для передачі великих обсягів даних через *BLE*. Коли передача буде завершена, додаток повинен запросити параметр *CONNECTION_PRIORITY_ BALANCED* для зниження енергоспоживання. Значення константи: 1 (0*x*00000001):

int CONNECTION_PRIORITY_LOW_POWER

Перелічимо основні методи, що було використано:

– *void abortReliableWrite* (*BluetoothDevice mDevice*) Увага: Цей метод є застарілим для *API* рівня 19. Використовуйте метод *abortReliableWrite* ().

– *void abortReliableWrite* () [Додано в *API* 19] Скасовує надійну транзакцію записи для даного пристрою. Виклик цієї функції призведе до відхилення всіх, хто знаходиться в черзі операцій записи характеристик для даного віддаленого пристрою. Потрібен дозвіл *BLUETOOTH*.

– *boolean beginReliableWrite* () Ініціює надійну транзакцію записи для даного віддаленого пристрою. Після того, як надійна транзакція запису була розпочата, всі виклики *writeCharacteristic* (*BluetoothGattCharacteristic*) відправляються на вилучену програму для перевірки і шикуються в чергу на виконання. Додаток отримує результат у зворотному виклику *onCharacteristicWrite* (*BluetoothGatt, BluetoothGattCharacteristic, int*) у відповідь на кожен виклик *writeCharacteristic* (*BluetoothGattCharacteristic*). У цьому зворотного виклику проводиться перевірка, чи було значення передано точно. Після того, як всі характеристики в черзі були перевірені, *executeReliableWrite* () виконає їх запис. Якщо характеристика була записана неправильно, виклик *abortReliableWrite* () скасує поточну транзакцію без зміни будь-яких значень на віддаленому пристрої. Потрібен дозвіл *BLUETOOTH*. Повертає: *true,* якщо надійна транзакція запису була розпочата.

– *void close* () Закрити *Bluetooth GATT*-клієнт. Додаток має викликати цей метод як можна раніше після того, як це робиться з поточним *GATT*-клієнтом.

– *boolean connect* () Цей метод використовується, щоб повторно з'єднатися з віддаленим пристрою після того, як з'єднання було розірвано. Якщо пристрій не перебуває у межах досяжності, повторне підключення буде вироблено, як тільки пристрій виявиться доступно. Повертає: *true,* якщо спроба підключення була успішно ініційована.

– *void disconnect* () Розриває встановлене з'єднання або скасовує спробу підключення, яка відбувається в даний час. Потрібен дозвіл *BLUETOOTH*.

boolean discoverServices () Виявляє послуги на віддаленому пристрої, а також їх характеристики і дескриптори. Це асинхронна операція. Після

завершення відкриття послуг, спрацьовує зворотний виклик *onServicesDiscovered* (*BluetoothGatt, int*). Якщо виявлення пройшло успішно, вилучені послуги можна отримати за допомогою функції *getServices* (). Потрібен дозвіл *BLUETOOTH*. Повертає: *true,* якщо виявлення віддалених послуг було розпочато.

– *boolean executeReliableWrite* () Виконує надійні транзакції записи для даного віддаленого пристрою. Ця функція дозволяє фіксувати всі характеристики, що знаходяться в черзі операцій записи для зазначеного віддаленого пристрою. Як передзвонити *onReliableWriteCompleted* (*BluetoothGatt, int*) спрацьовує щоб визначити, чи правильно була виконана операція. Потрібен дозвіл *BLUETOOTH*. Повертає: *true,* якщо запит на виконання операції був відправлений.

– *BluetoothDevice getDevice* () Повертає вилучену програму цільового *GATT*-скліента.

– *List <BluetoothDevice> getDevicesMatchingConnectionStates* (*int* [] *states*) Чи не підтримується. Будь ласка, використовуйте *BluetoothManager.getDevicesMatchingConnectionStates* (*int profile, int* [] *states*) з *BluetoothProfile.GATT* як перший аргумент. Параметри: Масив станів; одне з: *STATE_CONNECTED, STATE_CONNECTING, STATE_DISCONNECTED, STATE_DISCONNECTING*. Повертає: Список пристроїв. Список буде порожнім при помилку. Винятки: *UnsupportedOperationException*

– *BluetoothGattService getService* (*UUID uuid*) Повертає *BluetoothGattService,* якщо запитуваний ідентифікатор *uuid* підтримується на віддаленому пристрої. Ця функція вимагає, щоб виявлення послуг було виконано для даного пристрою. Якщо існує кілька примірників однієї послуги (з таким же *UUID*), повертається перший екземпляр служби. Потрібен дозвіл *BLUETOOTH*. Параметри: *UUID* запитуваної послуги. Повертає: *BluetoothGattService,* якщо підтримується, або *NULL,* якщо запитувана послуга не надається віддаленим пристроєм.

– *List <BluetoothGattService> getServices* () Повертає список *GATT*-послуг,

пропонованих віддаленим пристроєм. Ця функція вимагає, щоб виявлення послуг було виконано для даного пристрою. Потрібен дозвіл *BLUETOOTH*. Повертає: Список послуг віддаленого пристрою. Порожній список, якщо виявлення послуг ще не було виконано.

– *boolean readCharacteristic* (*BluetoothGattCharacteristic characteristic*) Запит на читання характеристики з віддаленого пристрою. Це асинхронна операція. Результат операції читання отримує зворотний виклик *onCharacteristicRead* (*BluetoothGatt, BluetoothGattCharacteristic, int*). Потрібен дозвіл *BLUETOOTH*. Параметри: Характеристика для читання з віддаленого пристрою. Повертає: *true,* якщо операція читання була розпочата успішно.

– *boolean readDescriptor* (*BluetoothGattDescriptor descriptor*) Зчитує значення дескриптора віддаленого пристрою. Як тільки операція читання завершена, спрацьовує зворотний виклик *onDescriptorRead* (*BluetoothGatt, BluetoothGattDescriptor, int*), сигналізуючи про результат операції. Потрібен дозвіл *BLUETOOTH*. Параметри: Дескриптор для читання з віддаленого пристрою. Повертає: *true,* якщо операція читання була розпочата успішно.

– *boolean readRemoteRssi* () Вважати *RSSI* підключеного віддаленого пристрою. (*RSSI* – це показник рівня сигналу). Як передзвонити *onReadRemoteRssi* (*BluetoothGatt, int, int*) спрацьовує, коли значення *RSSI* було прочитано. Потрібен дозвіл *BLUETOOTH*. Повертає: *true,* якщо операція читання була розпочата успішно.

– *boolean requestConnectionPriority* (*int connectionPriority*) Запросити оновлення параметра з’єднання. Ця функція відправить запит на оновлення параметра підключення до віддаленого пристрою. Параметри: Запитуваний статус з’єднання, один з: *CONNECTION_PRIORITY_BALANCED, CONNECTION_PRIORITY_HIGH, CONNECTION_PRIORITY_LOW_POWER.*

– *boolean requestMtu* (*int mtu*) Запросити розмір *MTU,* використовуваного для даного підключення. (*MTU* – це розмір одного неподільного блоку даних, переданого в поточній мережі за одну ітерацію). При виконанні запиту на запис (запис без відповіді), відправлені дані будуть усічені до розміру *MTU*. Ця

функція може використовуватися, щоб запросити більший розмір *MTU*, щоб мати можливість відправляти більше даних одночасно. Як передзвонити *onMtuChanged* (*BluetoothGatt, int, int*) буде вказувати, чи пройшла ця операція успішно. Потрібен дозвіл *BLUETOOTH*. Повертає: *true,* якщо нове значення *MTU* було запрошено успішно.

– *boolean setCharacteristicNotification* (*BluetoothGattCharacteristic characteristic, boolean enable*) Включити або відключити повідомлення / індикацію для даної характеристики. Після включення повідомлень для характеристики, зворотний виклик *onCharacteristicChanged* (*BluetoothGatt, BluetoothGattCharacteristic*) буде спрацьовувати, якщо вилучену програму повідомить, що характеристика змінилася. Потрібен дозвіл *BLUETOOTH*. Параметри: Характеристика, для якої необхідно включити повідомлення; *true,* якщо потрібно включити повідомлення. Повертає: *true,* якщо запитувана повідомлення про був встановлений успішно.

– *boolean writeCharacteristic* (*BluetoothGattCharacteristic characteristic*) Записує зазначену характеристику і її значення на віддалене пов'язане пристрій. Після того, як операція запису буде завершена, спрацює зворотний виклик *onCharacteristicWrite* (*BluetoothGatt, BluetoothGattCharacteristic, int*) з результатом операції. Потрібен дозвіл *BLUETOOTH*. Параметри: Характеристика для запису на віддаленому пристрої. Повертає: *true,* якщо операція запису була розпочата успішно.

– *boolean writeDescriptor* (*BluetoothGattDescriptor descriptor*) Записує значення дескриптора на поєднане пристрій. Як передзвонити *onDescriptorWrite* (*BluetoothGatt, BluetoothGattDescriptor, int*) спрацьовує щоб повідомити про результат операції. Потрібен дозвіл *BLUETOOTH*. Параметри: Дескріпотр для запису на віддаленому пристрої. Повертає: *true,* якщо операція запису була розпочата успішно.

– *void onCharacteristicChanged* (*BluetoothGatt gatt, BluetoothGattCharacteristic characteristic*) Як передзвонити ініціюється в результаті отримання повідомлення від віддаленої характеристики. Параметри:

GATT-клієнт; характеристика, оновлена в результаті віддаленого повідомлення.

– *void onCharacteristicRead* (*BluetoothGatt gatt, BluetoothGattCharacteristic characteristic, int status*) Як передзвонити, повідомляє результат читання віддаленої характеристики. Параметри: *GATT*-клієнт, який викликав *readCharacteristic*; характеристика, чиє значення було прочитано; статус: *GATT_SUCCESS,* якщо операція читання була успено завершена.

void onCharacteristicWrite (*BluetoothGatt gatt, BluetoothGattCharacteristic characteristic, int status*) Як передзвонити з зазначенням результату операції записи характеристики. Якщо цей зворотний виклик спрацьовує в той час, як надійна транзакція запису знаходиться в процесі, то значення характеристики являє собою суму, вказану віддаленим пристроєм. Прикладна програма повинна порівняти цю величину з потрібним значенням для запису. Якщо значення не збігаються, то програма має перервати транзакцію надійної записи. Параметри: *GATT*-клієнт, який викликав *writeCharacheristic*; записується характеристика; статус: *GATT_SUCCESS,* якщо операція запису була проведена успішно.

– *void onConnectionStateChange* (*BluetoothGatt gatt, int status, int newState*) Як передзвонити, що спрацьовує при підключенні до віддаленого *GATT*-сервера або відключенні від нього. Параметри: *GATT*-клієнт, який викликав підключення / відключення; статус: *GATT_SUCCESS,* якщо операція підключення / відключення пройшла успішно; новий стан з'єднання: *STATE_CONNECTED* або *STATE_DICONNECTED*.

– *void onDescriptorRead* (*BluetoothGatt gatt, BluetoothGattDescriptor descriptor, int status*) Як передзвонити, повідомляє про результат операції прочитання дескриптора. Параметри: *GATT*-клієнт, який викликав *readDescriptor*; дескриптор, прочитаний на віддаленому пристрої; статус: *GATT_SUCCESS,* якщо операція запису була проведена успішно.

– *void onDescriptorWrite* (*BluetoothGatt gatt, BluetoothGattDescriptor descriptor, int status*) Як передзвонити, повідомляє про результат операції записи дескриптора.

Програмна частина реалізована за допомогою додатку *Automate*, що дозволив алгоритмізувати і автоматизувати процеси з використанням можливостей смартфона.

Для *Automate* написано процес («*flow*» в термінології додатку) «*Bluetooth connection alert*» для оповіщення про підключення пристрою (рис. 3.4).

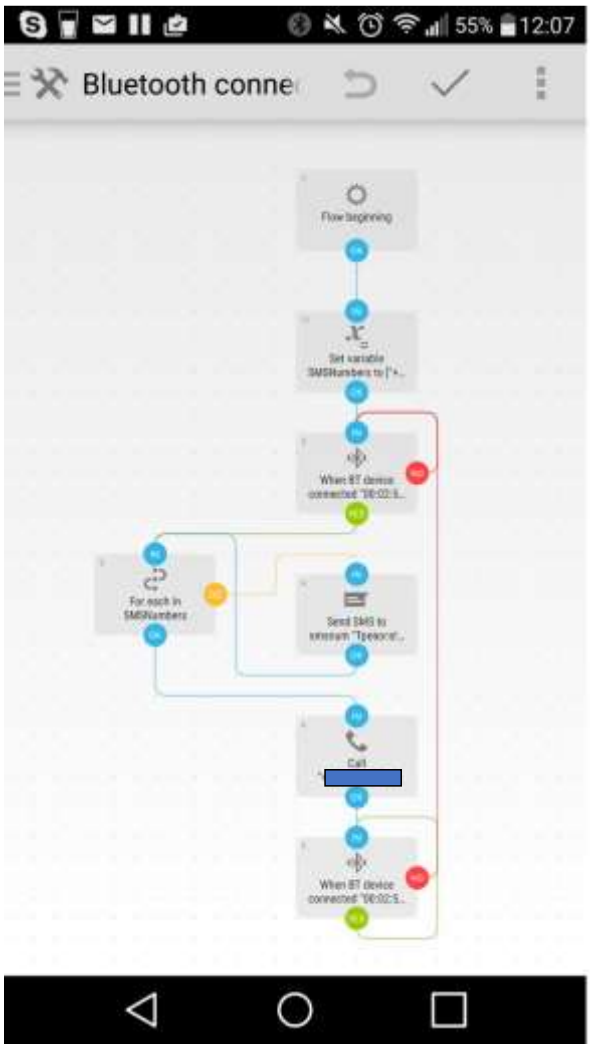

Рис. 3.4. Алгоритмізація в програми в системі *Automate*

Виявивши підключення *Bluetooth*-пристрою з заданою адресою, смартфон розсилає оповіщення і набирає номер.

Для підключення різних кнопок розширюється блок перевірки коду пристрою і відправки повідомлення згідно даного типу.

Додаток запускається на телефоні, який буде виконувати функцію сповіщувача. Його бажано розміщати в точці, яка рівновіддалена від всіх маяків-кнопок.

При включенні кнопки на телефоні буде відображено її код (рис. 3.5). Після визначення пристрою з заданого списку відбувається відправка повідомлення.

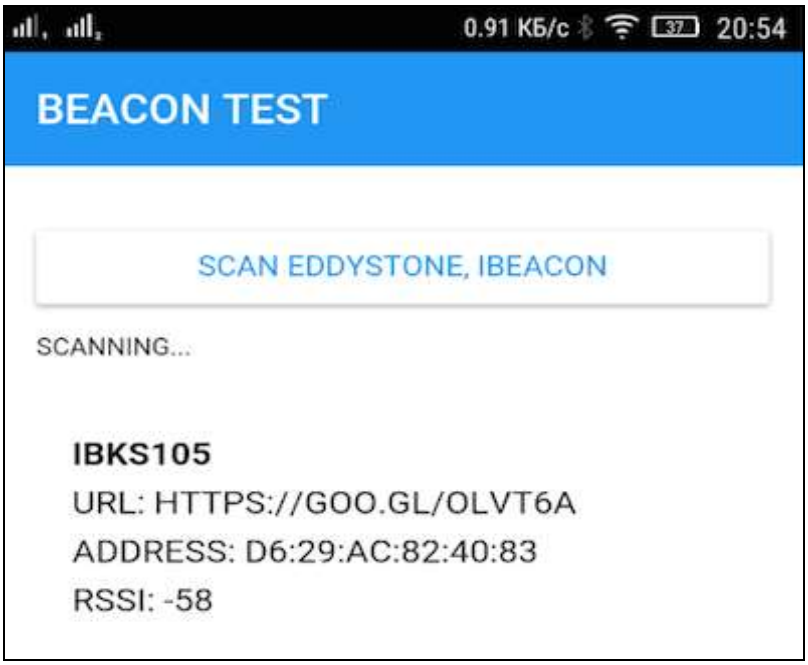

Рис. 3.5. Відображення коду активної кнопки

В даній версії програмного забезпечення не передбачено ведення історії викликів.

3.4. Схема розгортання системи виклику медичного персоналу

Основна відміннсть запропонованого рішення – це відсутність додатковиз пристроїв для агрегації сигналів від кнопок виклику. Замість цього пристрою використовується звичайний смартфон.

В найпростішому форматі можна використовувати стандартний *iBeacon* або її аналог (рис. 3.5).

Максимальна відстань між кнопками виклику та прийомним пристроєм може становити – до 50 метрів на відкритому просторі і близько 10-30 метрів в закритому приміщенні. Для збільшення радіусу прийому радіосигналів від кнопок виклику рекомендується застосовувати додатковий пристрій агрегатор з антеною, для ретранслювання радіосигналів.

Рис. 3.5. *Beacon* – пристрій, який використовується у якості кнопки

Це автоматичні прилади – підсилювачі цифрових радіосигналів від встановлених кнопок виклику. Один ретранслятор може посилити сигнал і передати інформацію навіть через 3-4 поверхи залізобетонного перекриття будівлі лікарні (рис. 3.6). У деяких випадках, при потужній екранування залізобетонних конструкцій, рекомендується використовувати кілька ретрансляторів – для розширення зони впевненого прийому пейджерів і приймачів сигналів кнопок виклику персоналу.

Але для використання в форматі обраного програмного рішення його бажано перепрошивати на визначення обраного набору кодів пристроїв.

Роботу системи можна представити наступною схемою (рис. 3.7). За даною схемою після натискання кнопки, включається пристрій., телефон визначає новий активний пристрій, програма визначає, що даний пристрій міститься у базі, реалізується *SMS*/дзвінок лікарю/лікарям.

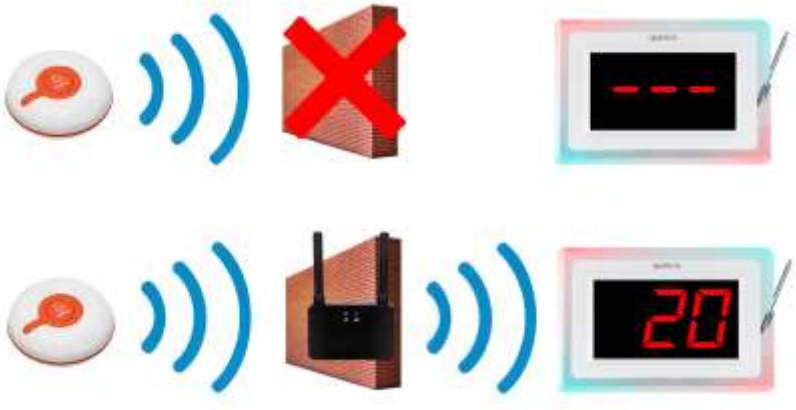

Рис. 3.6. Можливе використання ретранслятору в системі

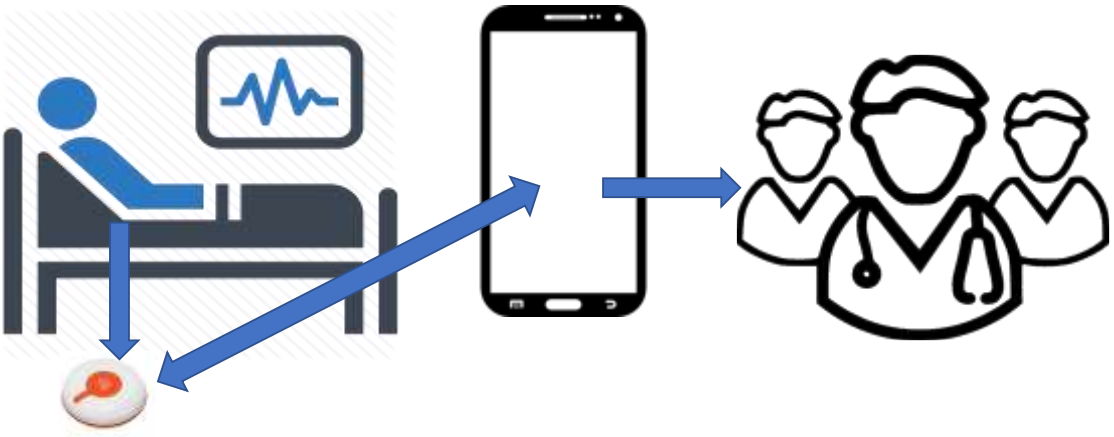

Рис. 3.7. Схема роботи системи виклику медичного персоналу

3.5. Аналіз рівня радіосигналу на точці розгортання системи

У разі використання підсилювачів сигналу необхідно визначити місця в приміщенні, де радіосигнал буде стійким. Якщо розглядається вільний простір, то передбачити який буде рівень електромагнітного сигналу на відстані від точки доступу в місці розташування абонента вкрай проблематично. У сучасних реаліях перед проектуванням мережі будують її плановану електромагнітну карту за допомогою різних програмних і апаратних комплексів. До програмних комплексів відносяться такі як: *TamoGraphSiteSurvey*, *AirMagnet Survey / Planner*, *Site Survey and Planning Tool*от компанії *Ekahau* і ін. Наприклад на малюнку нижче зображений зовнішній вигляд проекту в одній з перерахованих програм.

В основі цих програм лежить математичне ядро, побудоване на базі так званих моделей поширення радіосигналу (моделях втрат потужності). У деяких з них застосовуються і більш складні електродинамічні моделі.

Моделі розрахунку втрат радіосигналу дозволяють оцінити загасання електромагнітної хвилі, випромінюваної адаптером, з урахуванням кількості та типу перешкод на шляху проходження сигналу. У даній статті розглядаються моделі розповсюдження сигналу, що використовуються для розрахунку рівня сигналу всередині будівель. Моделей, про які піде мова, і їх модифікацій існує

велика кількість. У статті розглядаються найбільш прості, якими можна скористатися навіть в польових умовах без глибоких математичних знань.

На рис. 3.8 наведено графік залежності загасання *LFS* зі збільшенням відстані для радіо сигналу на першому частотному каналі (центральна частота 2437 МГц) в діапазоні 2.4 ГГц – синя крива, і в діапазоні 5 ГГц – червона крива. При цьому коефіцієнти посилення приймальні і передавальної антени були прийняті рівними одиниці.

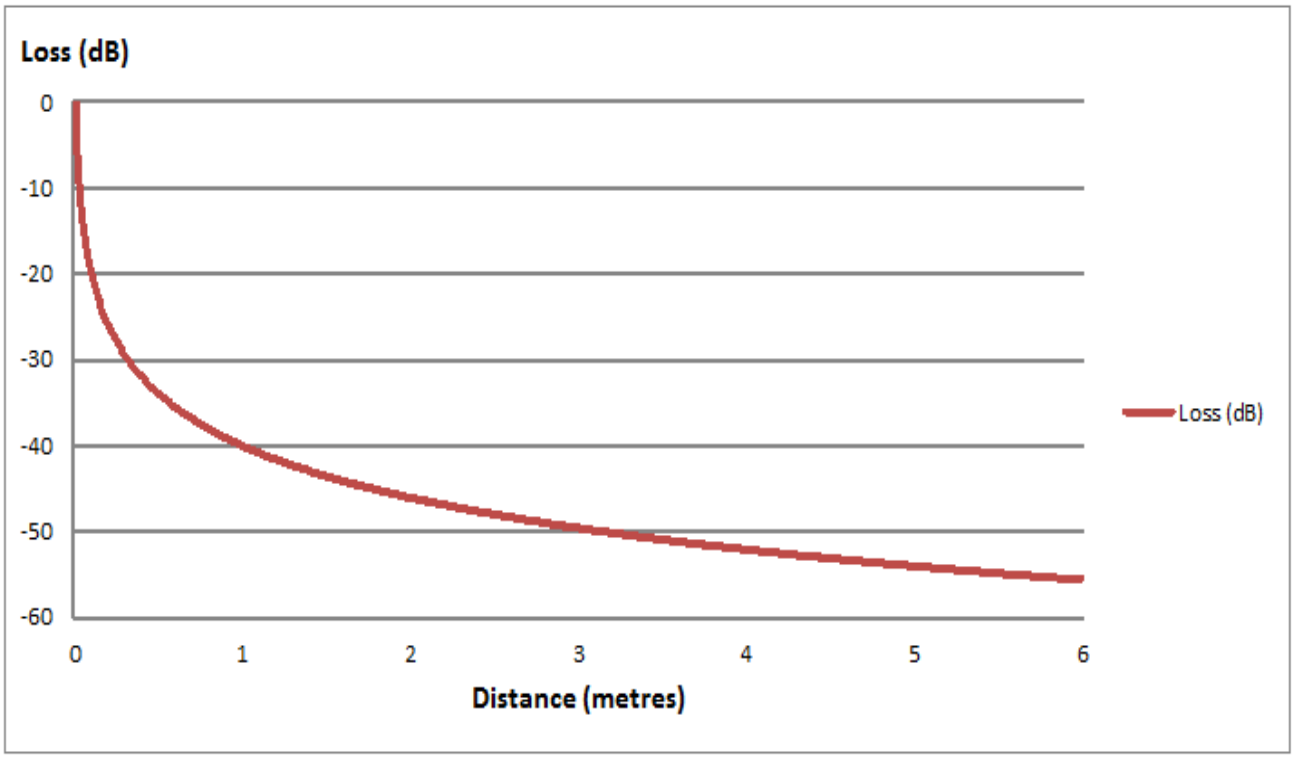

Рис. 3.8. Загасання сигналу *Wi-Fi* зі збільшенням відстаней

Як правило, більшість моделей поширення використовують значення втрат у вільному просторі в якості базового, і додають до нього змінні, що вносять додаткове загасання в залежності від типу перешкод і їх електрофізичних властивостей. До таких моделей належать, наприклад, *One slope* і *Log-distance*. Крім того, існує стандартизована Міжнародним союзом електрозв'язку модель втрат – *ITU-R* 1238. Перераховані моделі втрат відносяться до класу емпіричних статичних моделей, тобто для їх використання потрібно загальний опис типу завдання (типу приміщення). Перераховані моделі втрат з розшифровкою входять до цих змінних, що наведені в формулах.

Метод *One slope*

$$L(d) = L_{FS} + 10n\log\left(\frac{d}{d_0}\right),$$

де d – відстань в метрах, на якому проводиться оцінка загасання;

L_{FS}- втрати на відстані d_0 метрів;

n- коефіцієнт, що залежить від кількості і матеріалу перешкод.

$$L(d) = L_{FS} + 10n\log\left(\frac{d}{d_0}\right) + X_\sigma \ ,$$

де X_σ – нормальна випадкова величина, яка вимірюється в *dB*, що має стандартне відхилення, *DB*.

Розрахунок для *ITU-R* 1 238.

$$L(d) = 20\log f + N\log(d) + L_f(n) - 28 \ ,$$

де d> 1, м – відстань, на якому проводиться оцінка загасання;

f – частота центрального каналу *Wi-Fi*, МГц;

N – коефіцієнт втрати рівня сигналу з відстанню;

$L_f(n)$ – коефіцієнт втрати потужності сигналу при проходженні через стіну (підлога);

n>=1 – кількість стін (підлог) між приймальної і передавальної антен.

Про те, які значення в вищих формулах приймають змінні *N*, *n*, детально розписано безпосередньо в самій рекомендації МСЕ-*R* P. 1238-5 під назвою "Дані про поширення радіохвиль і методи прогнозування для планування систем радіозв'язку всередині приміщень і локальних зонових радіомереж в частотному діапазоні 900 МГц – 100 ГГц "(обсяг – 19 сторінок). Для експерименту, який буде проведено нижче, значення змінних будуть обрані з вказаною рекомендації. У різних ситуаціях змінні можуть набувати різних значень, і щоб перерахувати всі можливі випадки довелося б розмістити в статті мінімум 10 сторінок документа з 19-ти.

На жаль, перераховані моделі не враховують впливу на точку доступу

(точніше на випромінюється їй електромагнітну хвилю) стороннього обладнання, яке функціонує в тому ж частотному діапазоні. Тому всі розрахунки проводяться виходячи з того, що ваш пристрій єдине в усьому радіусі його (обладнання) дії. Як показує практика розрахунків, якщо в радіусі чутності вашої точки доступу знаходиться 20-30 бездротових пристроїв, то радіус дії зменшується на 15-20%. Але варто мати на увазі, що ця цифра суто приблизна і в різних ситуація може проявлятися по-різному, бо дуже залежить від потужності сигналу, який приходить в ваш пристрій, і від того на якій частоті працює навколишній обладнання.

Для аналізу приміщень, де передбачено розгортання системи, було використано систему *NetSpot* та проведено наступні кроки:

1) завантажено карту діапазону дослідження;

2) включення мережі в список сканованих, якщо вона має прихований *SSID* (якщо вона не транслює своє ім'я);

3) провести активне сканування. Для аналізу фактичноїшвидкості інтернет-з'єднаннявашої мережі можемо за бажанням включити Активне сканування на екрані;

4) провести моніторинг бездротової мережі. Необхідно пройтись по всіх кутках тієї території, яку хочемо промоніторити. Відзначаємо точки на карті, які відповідають тому, де знаходимось в момент замірювання сигналу. *NetSpot* фіксує перші показники (або вибірку даних). Проходимо до наступної точки і знову натискаємо на карті, щоб зафіксувати наступні показники. Як тільки відсканованопринаймні три точки, активується можливість Зупинити сканування, оскільки зібрано необхідний мінімум зразків даних для побудови звіту.

5) повторити п.5 у всіх приміщеннях;

6) вивчити отримані теплові карти (рис. 3.9).

«Ефірні» випробування – один із способів вирішення завдання визначення рівня сигналу на просторі, що дозволяє швидко вимірювати параметри якості роботи передавачів і приймачів.

Передавач і приймач однаково важливі, але перевірка таким способом експлуатаційних характеристик приймача стандарту *BLE* представляє особливу складність і вимагає розробки нових методів вимірювання. У статті досліджуються нові методи «ефірних» вимірювань, спеціально розроблені для визначення коефіцієнта пакетних помилок (*Packet Error Rate, PER*) і чутливості приймача пристроїв стандарту *BLE*.

Рис. 3.9. Отримана теплова карта приміщень рівня радіосигналу у прогармі *NetSpot*

Дві основні проблеми, які вирішуються «ефірними» випробуваннями:

1) проблема, що пов’язана з відсутністю можливості прямого дротного зв’зку під час випробувань;

2) проблема, що пов’язана з низькою швидкістю передачі даних

Проаналізуємо більш детально дані проблеми і шляхи їх вирішення.

Проблема № 1: Зазвичай, при «неефірних» випробуваннях для управління випробуваним пристроєм до нього підключаються через цифровий інтерфейс, наприклад *UART* або *USB*. Для випробувань приймача пристрій переводиться в

режим прийому пакетів, після чого випробувальна система передає відоме число пакетів, а потім надсилається запит про те, скільки пакетів було успішно прийнято. За цією інформацією випробувальна система розраховує коефіцієнт пакетних помилок (*PER*) – прийнятий в галузі кількісний показник якості роботи приймача. При «ефірних» випробуваннях відсутня прямий дротяний зв'язок з пристроєм, тому потрібен новий метод, який би дозволяв дізнатися, чи прийняті пакети належним чином.

Вирішення проблеми №1: Щоб розраховувати *PER* без проводового підключення, система для «ефірних» випробувань повинна використовувати стандартні повідомлення ефірного протоколу *BLE* і за ними визначати, прийняті пакети чи ні. Пристрої стандарту *BLE* ведуть передачу на трьох конкретних частотах оповіщення, розосереджених в діапазоні 2,4 ГГц.

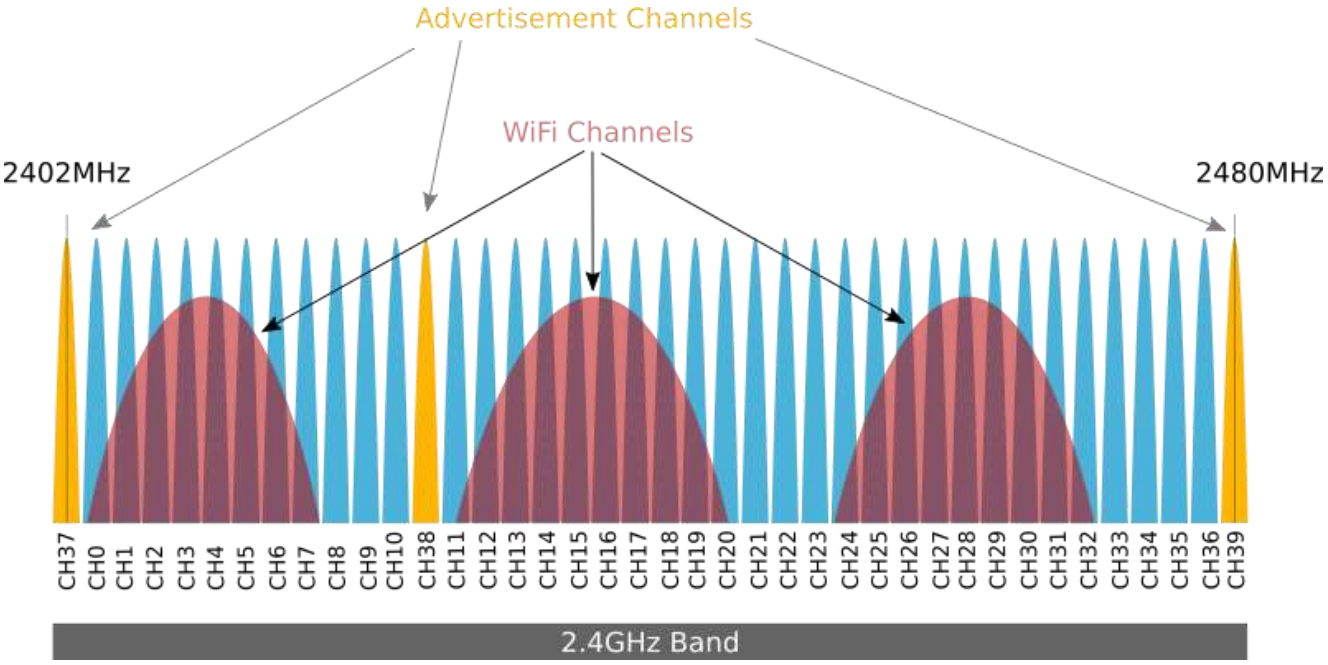

Рис. 3.10. Залежність потужності сигналів стандарту *BLE* (оповіщення, запит, відгук) від каналів

Щоразу після того, як пристрій з приймачем передає пакет оповіщення (*Advertising*), воно протягом короткого періоду часу прослуховує ефір на предмет стандартного повідомлення *BLE* під назвою *Scan_Request* (запит в режимі активного сканування). Це повідомлення зазвичай передають розташовані поблизу пристрої, яким потрібно зв'язатися з джерелом

оповіщення. Якщо джерело оповіщення отримує повідомлення *Scan_Request*, він відповідає на нього повідомленням *Scan_Response* (відгук в режимі активного сканування). Цей процес схематично зображено на рис. 1. Джерело оповіщення передає пакет оповіщення (показаний синім кольором). За ним слідує повідомлення *Scan_Request* від розташованої поблизу станції (показано помаранчевим кольором), на яке джерело оповіщення відповідає повідомленням *Scan_Response* (показано синім кольором, як і пакет оповіщення).

Цей обмін повідомленнями відбувається при нормальній роботі пристроїв стандарту *BLE*, і система для «ефірних» випробувань користується ним, щоб вимірювати *PER*. В ході випробування приймача досліджуваного пристрою (ДП) передає оповіщення, випробувальна система відповідає повідомленням *Scan_Request*, а потім ДП, якщо воно отримало пакет, підтверджує його отримання повідомленням *Scan_Response*. Випробувальна система веде облік відправлених повідомлень *Scan_Request* і отриманих повідомлень *Scan_Responses*. За цією інформацією розраховується *PER* приймача.

Для вимірювання чутливості випробувальна система підбирає такий рівень радіосигналу, при якому досягається певне значення *PER*. Цей рівень, званий чутливістю приймача, служить поширеним показником якості приймача. Метод з використанням пакетів оповіщення, повідомлень *Scan_Request* і повідомлень *Scan_Response* реалізований в «ефірному» тестері пристроїв стандарту *BLE* і забезпечує точне визначення *PER* і чутливості приймача без безпосередньої дротового зв'язку з ДП.

Проблема № 2. Технологія *BLE* характеризується низькими швидкостями передачі даних в порівнянні з іншими поширеними технологіями бездротового зв'язку, наприклад *Wi-Fi* або стільниковим зв'язком. При випробуваннях приймачів потрібно передавати велику кількість пакетів, щоб результати вимірювань були статистично значущими і точними.

Відповідно, зважаючи на низькі швидкості передачі даних, випробування приймачів стандарту *BLE* на виробництві будуть займати занадто багато часу,

через що істотно підвищиться собівартість тестування. Зрозуміло, це неприйнятно, тому виникає друга проблема: необхідний новий метод вимірювання, який дозволяє точно визначати *PER* по значно меншій кількості пакетів, ніж прийнято в традиційних методах визначення цього параметра.

Рішення проблеми № 2: чутливість приймача зазвичай визначається шляхом вимірювання в деякому діапазоні рівнів радіосигналу. Організація *Bluetooth SIG* (*Special Interest Group*), яка відає стандартизацією технології *Bluetooth*, наказує вимірювати чутливість приймача по 1500 пакетам. Якщо застосовувати описаний вище метод оповіщення, у типового пристрою стандарту *BLE* піде кілька хвилин на передачу такої кількості пакетів при кожному визначеному рівні потужності сигналу. У припущенні, що вимірювання проводиться при декількох значеннях рівня, весь цикл визначення чутливості може зайняти більше 10 хв.

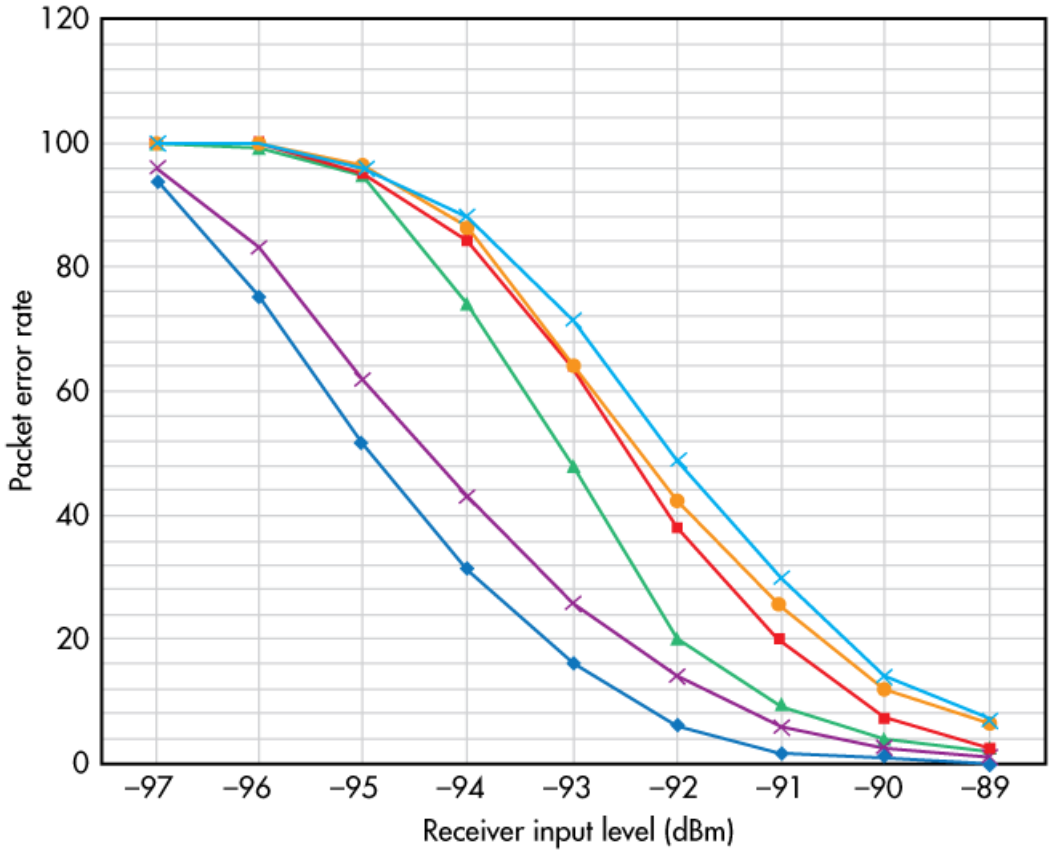

Рис. 3.11. Типові криві розподілу коефіцієнта пакетних помилок для пристроїв стандарту *BLE*

Це явно неприйнятно в багатьох випадках, тому потрібен новий, більш швидкий метод. Вирішити цю проблему можна, застосувавши в системі для «ефірних» випробувань алгоритм швидкого визначення кривої розподілу *PER*. Цей т.зв. «Метод швидкого визначення *PER*» дозволяє за короткий час визначити рівень сигналу, при якому *PER* дорівнює 50%.

Випробування за методом швидкого визначення *PER* починається з того, що тестер передає одиночний пакет (*Scan_Request*) при довільному значенні потужності сигналу. Якщо цей пакет приймається, рівень радіосигналу знижується, і наступний пакет передається на цьому зниженому рівні. Якщо цей пакет не приймається, рівень радіосигналу підвищується, і наступний пакет передається на цьому підвищеному рівні. Використовуючи інтелектуальний алгоритм коригування кроку зміни рівня сигналу та запам'ятовуючи, на яких рівнях пакети приймалися, а на яких немає, можна швидко і точно побудувати криву розподілу коефіцієнта пакетних помилок по мінімальному числу пакетів.

На рис. 2 показана звичайна крива розподілу *PER* для пристрою стандарту *BLE*. Крива демонструє найбільшу чутливість до зміни рівня радіосигналу в точці, де *PER* = 50%, тому алгоритм швидкого визначення *PER* шукає саме цю точку. В ході варіювання рівня радіосигналу більшість пакетів передається при значеннях рівня, близьких до точки *PER* = 50%, і в міру сходження алгоритму в її околиці крок зміни рівня зменшується. Тим самим забезпечується добре повторюване і дуже точне вимірювання чутливості приймача ДП, що дозволяє обійтися менш ніж 5% від того числа пакетів, яке треба було б для визначення *PER* традиційним методом прямої розгортки за рівнем сигналу в широких межах.

Висновки до розділу

В розділі потупово розкрито всі етапи створення системи виклику медичного персоналу на базі блутус технології: від проектування *Bluetooth* пристрою до написання коду програми.

Для вирішення проблеми з мертвими зонами. Додатково в даному розділі був показаний практичний приклад застосування стандартизованої моделі розрахунку загасання сигналу всередині будівлі. Ця та інші моделі допоможуть досить швидко, без застосування спеціалізованого ПЗ, оцінити кількість необхідного обладнання для Вашого офісу. Звичайно, цей підхід не замінить якісних проектних розрахунків в спеціалізованих програмних продуктах, але дозволить що називається "зорієнтуватися на місцевості", потрібно лише враховуватися геометрію будівлі для отримання більш коректних результатів.

ВИСНОВКИ

Безумовно, велика частина областей застосування *Bluetooth* може бути успішно замінена або доповнена пристроями *BLE*, продовживши термін служби пристроїв за рахунок більш ефективного управління енергоспоживанням. Зокрема, можливе застосування дворежимних пристроїв *BLE* в мобільних телефонах, планшетних комп'ютерах, ноутбуках. Однорежимні пристрої можуть застосовуватися в якості бездротового інтерфейсу датчиків на батарейках, що застосовуються як окремо, так і в складі інших пристроїв – годинах, пульсометра, крокомір, домашніх тонометрах, термометрах і тому подібних пристроїв.

У складі мобільних пристроїв *BLE* може бути використаний для управління домашньою автоматикою, пристроями освітлення або охорони як мінімум, в межах одного приміщення. Для керування пристроями в межах всього будинку можливе використання *BLE* в якості шлюзу між керуючим пристроєм і мережею домашньої автоматики.

Низький рівень споживання енергії та більш стійка робота в умовах великої кількості аналогічних пристроїв в ряді випадків дозволяє розглядати *BLE* як альтернативу пристроїв *NFC*, зокрема *RFID* мітками. Але більш цікавий варіант використання *BLE* спільно з *NFC*. У цьому випадку перші забезпечують більший радіус стійкої роботи і велику кількість спільно діючих механізмів, а другі служать для встановлення логічного з'єднання між парою пристроїв, забезпечую більш високий рівень безпеки за рахунок меншого радіуса дії.

Принцип роботи *BLE* описаний вже в його назві: *Low Energy*. Протокол передбачає передачу даних короткими пакетами по необхідності, потім – вимкнення передавача. Низький рівень споживання енергії частково досягається застосуванням саме цього принципу. Замість класичного тандема в звичайному *Bluetooth*, пристрої *BLE* зв'язуються один з одним лише при необхідності відправки або отримання інформації.

Протокол *BLE* строго структурований за принципом своєї комунікації з іншими пристроями. Спочатку девайси вивчають доступні сервіси для відправки / прийняття даних; невід'ємна частина цих сервісів – їх характеристики (*characteristics*), що визначають тип даних для майбутньої передачі. Характеристики, з міркувань наочності, можуть мати в своєму складі опису-дескриптори (*descriptors*), які допомагають визначити тип даних. Наприклад, розберемо сервіс під назвою «*Heart Rate Monitor*» (монітор частоти серцебиття) – серед його характеристик присутні такі, як «вимір пульсу».

Більшість *API* для *Bluetooth LE* дозволяють шукати локальні пристрої та визначати доступні в них сервіси, характеристики і дескриптори.

В розділі потупово розкрито всі етапи створення системи виклику медичного персоналу на базі блутус технології: від проектування *Bluetooth* пристрою до написання коду програми.

Для вирішення проблеми з мертвими зонами. Додатково в даны роботі був показаний практичний приклад застосування стандартизованої моделі розрахунку загасання сигналу всередині будівлі. Ця та інші моделі допоможуть досить швидко, без застосування спеціалізованого ПЗ, оцінити кількість необхідного обладнання для Вашого офісу. Звичайно, цей підхід не замінить якісних проектних розрахунків в спеціалізованих програмних продуктах, але дозволить що називається "зорієнтуватися на місцевості", потрібно лише враховуватися геометрію будівлі для отримання більш коректних результатів.

Конструкторам радіоапаратури необхідно перевіряти експлуатаційні характеристики готових виробів, а не покладатися виключно на вимірювання радіочастотних параметрів на рівні плат, які можуть давати кардинально інші результати. Випробувачі на виробництві часто не мають доступу до радіочастотним і цифровим ланцюгах на рівні плат, тому їм потрібна можливість швидко проводити «ефірні» випробування і отримувати при цьому точні і повторювані результати.

Вирішувати ці завдання дозволяють системи для «ефірних» випробувань пристроїв стандарту *BLE*. За допомогою цих систем інженери можуть

удосконалювати свої конструкції, а виробники – перевіряти якість готових виробів за основними параметрами. В контексті технології *BLE* експлуатаційні характеристики радіочастотної частини мають першорядне значення, тому що погана якість бездротового зв'язку означає погану роботу всього пристрою в цілому. Системи для «ефірних» випробувань пристроїв стандарту *BLE* полегшують конструктивне вдосконалення виробів і перевірку якості їх виготовлення.

Основна відміннсть запропонованого рішення – це відсутність додатковиз пристроїв для агрегації сигналів від кнопок виклику. Замість цього пристрою використовується звичайний смартфон.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – 39 с.

2. Бойченко С.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету / С.В. Бойченко, О.В. Іванченко – К.: НАУ, 2017. – 63 с.

3. *Akyildiz I.F. Wireless multimedia sensor networks*: *applications and testbeds* / *Akyildiz I.F., Melodia T., Chowdury K.R.// Proceedings of the IEEE* (*invited paper*), 2008. – *Vol*. 96. – № 10 – *pp*. 1588-1605.

4. *Akyildiz I.F. Wireless Sensor Networks / I.F. Akyildiz, M.C. Vuran .– New York*: *John Wiley & Sons,* 2010. – 571 *p.*

5. *Core specification v*4.1. – [Електронний ресурс]: *Bluetooth.org*

6. *Supplements to the Core Specification* (*CSS*) *v*.4. – [Електронний ресурс]: *Bluetooth.org*

7. *Assigned Numbers, GAP*, 22.10.2014 13:37. – [Електронний ресурс]: *Bluetooth.org*

8. *Indoor Light Energy Harvesting Reference Design for Bluetooth Low Energy* (*BLE*) *Beacon Subsystem*: *http*://*www.ti.com*/*tool*/*tida*-00100;

9. *TI Online Store*. *https*://*estore.ti.com*/;

10. *CC*2543 *miniBLEBroadcaster*. – [Електронний ресурс]: *http*://*www.ti.com*/*product*/*cc*2543 /*toolssoftware*;

11. *Mini Bluetooth® Low Energy Broadcaster*;

12. *TI BLE Software Developer's Guide, included with the BLE SDK Installer*: *www.ti.com*/*ble-stack*;

13. *TI Packet Sniffer*: *http*://*www.ti.com*/*tool*/*packet-sniffer*.

14. Архипкин В. *Bluetooth*. Информационно-технический центр «Мобильные коммуникации». – М:2017. – 660 с.

15. Болілий В.О. Комп'ютерні мережі. Навчальний посібник / В.О.

Болілий, В.В. Котяк – Кіровоград: ЦОП Авангард, 2008.- 146с.

16. Вишневский В.М. Широкополосные беспроводные сети передачи информаци / Вишневский В.М., Ляхов А.И., Портной С.Л., Шахнович И.В. – М: Эксмо, 2005. – 224 с.

17. Высокопроизводительные сети. Энциклопедия пользователя./ Под ред. Марка А. Спортака: Пер. с англ. – К.: ДиаСофт, 2008. – 836 с.

18. Гепко И., Олейника В. Современные беспроводные сети. Состояние и перспективы развития /И.Гепко, В.Олейника – Воронеж: Принт, 2016. – 484 с.

19. Григорьев В.А. Системы и сети радиодоступа / Григорьев В.А., Лагутенко О.И., Распаев Ю.А. – М:ЭкоТрендз, 2005. – 460 с.

20. Коробейников А.В. Структура системы мониторинга для медицинских учреждений / Математическое моделирование и интеллектуальные системы. – Ижевск: ИжГТУ, 2003. – № 1. – С. 18-21.

21. Кузьменко Н.Г. Компьютерные сети и сетевые технологии. – М.: Наука и Техника, 2013. – 368 с.

22. Педжман Р. Основы построения беспроводных локальных сетей стандарта 802.11 / Рошан Педжман, Лиэри Джонатан – М.:, 2006 – 315 *с*

ДОДАТКИ

Додаток А. Лістинг програмних кодів

```
public class BluetoothLeService extends Service {
 private final static String TAG = BluetoothLeService.class.getSimpleName ();

 private BluetoothManager mBluetoothManager;
 private BluetoothAdapter mBluetoothAdapter;
 private String mBluetoothDeviceAddress;
 private BluetoothGatt mBluetoothGatt;
 private int mConnectionState = STATE_DISCONNECTED;

 private static final int STATE_DISCONNECTED = 0;
 private static final int STATE_CONNECTING = 1;
 private static final int STATE_CONNECTED = 2;

 public final static String ACTION_GATT_CONNECTED =
 "Com.example.bluetooth.le.ACTION_GATT_CONNECTED";
 public final static String ACTION_GATT_DISCONNECTED =
 "Com.example.bluetooth.le.ACTION_GATT_DISCONNECTED";
 public final static String ACTION_GATT_SERVICES_DISCOVERED =
 "Com.example.bluetooth.le.ACTION_GATT_SERVICES_DISCOVERED";
 public final static String ACTION_DATA_AVAILABLE =
 "Com.example.bluetooth.le.ACTION_DATA_AVAILABLE";
 public final static String EXTRA_DATA =
 "Com.example.bluetooth.le.EXTRA_DATA";
```

// Встановлюємо *UUID*, який використовується для послуг вимірювання пульсу

public final static UUID UUID_HEART_RATE_MEASUREMENT =
UUID.fromString (*SampleGattAttributes.HEART_RATE_MEASUREMENT*);

// Різні методи зворотного виклику, певні в *BLE API*
private final BluetoothGattCallback mGattCallback =
new BluetoothGattCallback () {
@Override
public void onConnectionStateChange (*BluetoothGatt gatt, int status,*
int newState) {
String intentAction;
if (*newState == BluetoothProfile.STATE_CONNECTED*) {
intentAction = ACTION_GATT_CONNECTED;
mConnectionState = STATE_CONNECTED;
broadcastUpdate (*intentAction*);
Log.i (*TAG, "Connected to GATT server*.");
Log.i (*TAG, "Attempting to start service discovery*:" +
mBluetoothGatt.discoverServices ());

} *Else if* (*newState == BluetoothProfile.STATE_DISCONNECTED*) {
intentAction = ACTION_GATT_DISCONNECTED;
mConnectionState = STATE_DISCONNECTED;
Log.i (*TAG, "Disconnected from GATT server*.");
broadcastUpdate (*intentAction*);
}
}

@Override
// При виявленні нового сервісу

```
public void onServicesDiscovered (BluetoothGatt gatt, int status) {
if (status == BluetoothGatt.GATT_SUCCESS) {
broadcastUpdate (ACTION_GATT_SERVICES_DISCOVERED);
} Else {
Log.w (TAG, "onServicesDiscovered received:" + status);
}
}

@Override
// Результат читання характеристики
public void onCharacteristicRead (BluetoothGatt gatt,
BluetoothGattCharacteristic characteristic,
int status) {
if (status == BluetoothGatt.GATT_SUCCESS) {
broadcastUpdate (ACTION_DATA_AVAILABLE, characteristic);
}} ...}; ...}
```

Фрагмент коду, перебирає послуги і характеристик сервера і відображає їх в інтерфейсі

```
public class DeviceControlActivity extends Activity {
...
// Демонструє, як перебрати підтримувані GATT-послуги / характеристики.
// У цьому прикладі ми заповнюємо структуру даних, яка прив’язана до ExpandableListView
// в інтерфейсі користувача.
private void displayGattServices (List <BluetoothGattService> gattServices) {
if (gattServices == null) return;
String uuid = null;
```

String unknownServiceString = getResources ().
getString (*R.string.unknown_service*);
String unknownCharaString = getResources ().
getString (*R.string.unknown_characteristic*);
ArrayList <HashMap <String, String >> gattServiceData =
new ArrayList <HashMap <String, String >> ();
ArrayList <ArrayList <HashMap <String, String >>> gattCharacteristicData
= New ArrayList <ArrayList <HashMap <String, String >>> ();
mGattCharacteristics =
new ArrayList <ArrayList <BluetoothGattCharacteristic >> ();

// Перебирає доступні *GATT*-послуги
for (*BluetoothGattService gattService*: *gattServices*) {
HashMap <String, String> currentServiceData =
new HashMap <String, String> ();
uuid = gattService.getUuid (). *toString* ();
currentServiceData.put (
LIST_NAME, SampleGattAttributes.
lookup (*uuid, unknownServiceString*));
currentServiceData.put (*LIST_UUID, uuid*);
gattServiceData.add (*currentServiceData*);

ArrayList <HashMap <String, String >> gattCharacteristicGroupData =
new ArrayList <HashMap <String, String >> ();
List <BluetoothGattCharacteristic> gattCharacteristics =
gattService.getCharacteristics ();
ArrayList <BluetoothGattCharacteristic> charas =
new ArrayList <BluetoothGattCharacteristic> ();

// Перебирає доступні *GATT*-характеристики

```
for (BluetoothGattCharacteristic gattCharacteristic:
gattCharacteristics) {
charas.add (gattCharacteristic);
HashMap <String, String> currentCharaData =
new HashMap <String, String> ();
uuid = gattCharacteristic.getUuid (). toString ();
currentCharaData.put (
LIST_NAME, SampleGattAttributes.lookup (uuid,
unknownCharaString));
currentCharaData.put (LIST_UUID, uuid);
gattCharacteristicGroupData.add (currentCharaData);
}
mGattCharacteristics.add (charas);
gattCharacteristicData.add (gattCharacteristicGroupData);
} ...} ...}
```