

Politechnika Warszawska

W Y D Z I A Ł E L E K T R Y C Z N Y



Instytut Elektrotechniki Teoretycznej i Systemów Informacyjno-Pomiarowych

Praca dyplomowa magisterska

na kierunku Informatyka Stosowana
w specjalności Inżynieria Oprogramowania

Nowe możliwości Bluetooth 5 w aplikacjach Internetu Rzeczy

inż. Krzysztof Paweł Kruk
numer albumu 301140

promotor
dr inż. Łukasz Makowski

WARSZAWA 2022

Nowe możliwości Bluetooth 5 w aplikacjach Internetu Rzeczy

Streszczenie

Jednym z dostępnych na rynku standardów komunikacji bezprzewodowej bliskiego zasięgu jest Bluetooth. Obecny od ponad dwudziestu lat wśród elektroniki użytkowej zdobył ugruntowaną pozycję i popularność. Bluetooth 5 rozszerza znane wcześniej funkcjonalności, zwiększa przepustowość i zasięg przy jednoczesnej redukcji zużycia energii oraz wprowadza nowy standard komunikacji sieci Mesh – nowy protokół komunikacji między-węzłowej.

Niniejsza praca wprowadza w świat łączności bezprzewodowej opisując jedne z najpopularniejszych obecnie standardów. Wprowadza się terminologię dla wybranych stosów technologicznych oraz zestawia z modelem referencyjnym ISO OSI. Wśród protokołów opartych o IEEE 802.15.4 wymienia się specyfikację ZigBee oraz Thread. Przedstawiany jest opis zmian Bluetooth od wersji 5.0. Ostatecznie, zestawia się wybrane cechy każdego z wymienionych standardów.

Przegląd literatury wskazuje na szerokie zainteresowanie nauki dziedziną łączności bezprzewodowej. Szereg artykułów prezentuje empirycznie zebrane cechy poszczególnych standardów takie jak opóźnienia transmisji czy maksymalna przepustowość.

Doświadczalnym aspektem pracy jest pomiar zużycia energii i pomiar Packet Error Rate w zależności od wzajemnej odległości węzłów w skonfigurowanych w sieć Bluetooth Mesh i częstotliwości wysłanych komunikatów. Stawiana jest główna hipoteza badawcza, która następnie jest empirycznie weryfikowana. W tym celu konstruowany jest tor badawczy oparty o zestaw uruchomieniowy *P-NUCLEO-WB55* oraz autorskie oprogramowanie. Przedstawiana jest metodologia poszczególnych badań oraz analiza potencjalnych źródeł błędów.

Ostatecznie, prezentowane są wyniki eksperymentów. Zestawia się wartości zużycia energii dla Bluetooth Low Energy i Mesh. Analizowane są przypadki badań Packet Error Rate, które następnie dzieli się na dwie różne klasyfikacje.

Na koniec, praca podsumowuje dokonane prace i wyciąga konstruktywne wnioski.

Słowa kluczowe: Bluetooth, Bluetooth Low Energy, Bluetooth Mesh, Packet Error Rate, Zużycie energii

New Bluetooth 5 features in Internet of Things applications

Abstract

Bluetooth is a widely-available short-range wireless technology. The technology has been present for more than twenty years in the market gaining undisputable popularity among consumer electronics. Bluetooth 5 pushes the limits further, providing a new set of features, increasing throughput and range while it simultaneously reduces energy requirements. Most importantly, the new standard introduces Bluetooth Mesh – a new wireless inter-node protocol.

This thesis introduces into a realm of wireless technologies. It describes a range of wireless standards known to be the most popular. It defines the terminology for a given set of technology stacks and compares them with a conceptual model – Open Systemns Interconnection model (OSI). ZigBee and Thread represent a family of IEEE 802.15.4-based stacks. The thesis describes a set of Bluetooth 5 changes. Finally, it juxtaposes the mentioned standards with each other.

The literature review suggests thats wireless communication is a common and popular field of study. There are many papers that test parameters of each of said stack. These parameters include but not limited to – latency, throughput energy consumption and other.

The practical aspect of the thesis embraces two experiments – energy consumption and Packet Error Rate in Bluetooth Mesh network. The latter experiment includes dependent factors such as distance between nodes or ping frequency. The thesis states the main test hypothesis and tries to empirically verify it. All the experiments are based on a commercial off-the-shelf development board – *P-NUCLEO-WB55*. The board is programmed and operated by custom software specifically designed to run the mentioned experiment. The thesis includes a methodology description as well as an analysis of possible error root causes.

Eventually, the thesis presents results of the abovementioned experiments. It presents Bluetooth Low Energy and Mesh energy consumption charts and compares them. It analyzes Packet Error Rate results by dividing it into two distinct categories and presenting charts accordingly.

Finally, the thesis summarizes all experiments and draws conclusions.

Keywords: Bluetooth, Bluetooth Low Energy, Bluetooth Mesh, Packet Error Rate, Energy consumption

Spis treści

1 Wstęp	9
2 Komunikacja radiowa bliskiego zasięgu	13
2.1 ZigBee	14
2.2 Thread	17
2.3 Bluetooth 5 - Bluetooth Low Energy	19
2.4 Porównanie przedstawionych standardów	23
2.5 Teoretyczne podstawy zaprojektowanych eksperymentów	25
3 Część doświadczalna	31
3.1 Przygotowanie eksperymentu	32
3.1.1 Sprzęt i oprzyrządowanie	32
3.1.2 Narzędzia i elementy dodatkowe	34
3.1.3 Oprogramowanie mikrokontrolera	37
3.1.4 Oprogramowanie PC	45
3.2 Badanie zużycia energii	49
3.2.1 Metodologia badania	50
3.2.2 BLE- Usługa Heart Rate	52
3.2.3 BT Mesh - Model Generic OnOff	54
3.3 Badanie Packet Error Rate	57
3.3.1 Metodologia badania	58
3.3.2 Zależność PER względem częstości zapytań	61
3.3.3 Zależność PER względem odległości między węzłami	65
4 Podsumowanie	69
Bibliografia	73
Wykaz skrótów i symboli	77
Spis rysunków	81

Spis tabelic	83
Spis załączników	85

Rozdział 1

Wstęp

Od czasów powstania pierwszych procesorów, naukowcy i inżynierowie łączyli je z dostępnymi elementami dyskretnymi tworząc urządzenia spełniające określone potrzeby. Z każdym przełomem w miniaturyzacji komponentów cechą wspólną jest tworzenie zespołów współpracujących ze sobą elementów. Od elektroniki analogowej, do obecnych możliwości elektroniki cyfrowej, wymiana informacji, wysyłanie i odbiór sygnałów, jest kluczem do stworzenia zaawansowanego urządzenia.

Intuicyjnie, termin sygnał przynosi na myśl pojęcie nośnika informacji, czy wymiany tejże informacji. Naturalnie łączy się to słowo z wielkościami fizycznymi a nawet namacalnym. Docelowo, pragniemy wysłać pewną treść, co też wiąże się z rozdrobnieniem tak abstrakcyjnej koncepcji na możliwe małe fragmenty, które następnie można przesyłać dalej. Chcąc opisać termin nie tylko jakościowo, ale też ilościowo, kształtuje się taką ideę do postaci *modelu matematycznego*. Sygnałem więc nazwać można pewną funkcję czasu opisującą zjawisko przesyłu tej informacji [30].

Wraz z rozwojem techniki, opracowano właściwe technologie i ustalonono kontrakty definiujące kodowanie owego sygnału. Nie mniej istotną cechą jest wybór zjawiska fizycznego, które tę wiadomość ma przesyłać. Od tego zależy sposób konstrukcji urządzeń odpowiadających za przesył danych. Inne narzędzia należy wykorzystać do komunikacji z użyciem znaków dymnych, a zupełnie innych do kontaktu z łazikiem marsjańskim oddalonym o 20 minut świetlnych od Ziemi. Przesył sygnału nie jest cechą wyłącznie innowacyjności ludzkich dzieł. Natura w wyniku ewolucji opracowała szereg czynników umożliwiający przesyłanie i odbiór informacji – powonienie, smak, transport aktywny jonów w postaci pompy sodowo-potasowej będącą podstawą dla transmisji sygnałów w komórkach nerwowych itd. Sygnał i możliwość manipulacji zdaje się być podstawą funkcjonowania nie tyle cywilizacji ludzkiej, co życia samego w sobie.

Technologicznie, sygnał przesyłać można z użyciem szeregu zjawisk fizycznych, najczęściej powiązanych ze zjawiskami mechanicznymi oraz elektrycznymi. Skupiając się na tych ostatnich, sygnał generuje się bazując na pojęciach napięcia, prądu, częstotliwości, czy ogólnie fal elektromagnetycznych. Telegraf, czy jego współczesne wersje w postaci telefonu czy Internetu, manipulują tą falą z użyciem technologii celem wysłania informacji z punktu A do punktu B. Sposób

w jaki wpływamy na otaczające środowisko, by wysłać sygnał, modelowo nazywane jest warstwą fizyczną [16].

Kolejnym krokiem jest ustalenie pewnego kontraktu pomiędzy wspomnianą warstwą fizyczną, a potencjalnymi warstwami wyższymi. Wymagany jest sposób według którego można zinterpretować ilościowy udział zjawiska fizycznego. W przypadku elektroniki cyfrowej bazującej na *TTL*¹, informacja kodowana jest z użyciem zmian napięcia, gdzie sygnałem niskim (czyli logiczne „0”) nazywamy napięcie w zakresie 0V do 0.8V, a stan wysoki (czyli logiczne „1”) od 2.4 do maksymalnego napięcia 5V. Nowoczesne systemy ten prosty przykład znaczco modyfikują, by przesyłać więcej danych w krótszym czasie, najlepiej na dalszy dystans z minimalizacją zużycia energii. Sposób w jaki jest to zorganizowane, można przyrównywać właśnie do modelu OSI stanowiącego pewien schemat rozumowania przesyłu sygnału cyfrowego w sieci.

Poprzez analogię do wyżej wymienionych zjawisk, niniejsza praca podejmuje się badania właściwości jednej z technologii wykorzystujących podstawy fizyczne do bezprzewodowego przesyłu informacji. Technologią tą jest Bluetooth 5 będąca rozwinięciem standardu funkcjonującego od ponad dwudziestu lat. Wykorzystywana z powodzeniem w elektronice użytkowej wraz z nową wersją wprowadza szereg usprawnień, które umożliwiają zastosowanie aplikacjach IoT.

Internet Rzeczy (z ang. *Internet of Things*) (IoT) jest systemem złożonym z elementów (rzeczy) połączonych ze sobą w sieć, umożliwiającą wysyłanie, przesyłanie i przetwarzanie danych pomiędzy poszczególnymi węzłami. Zagadnienie łączy we wspólną całość mikrokontrolery, czy jednopłytkowe komputery oparte o układ scalony typu system na chipie (z ang. *System on a Chip*) (SoC), wraz z większą siecią nie wykluczając Internetu. Zagadnienie tym samym wiąże nie tylko opis protokołów transmisji danych, ale wymaga skupienia również na sprzęcie stanowiąc kompletny przepis na docelowe rozwiązanie techniczne [10].

Internet Rzeczy koncentruje się na szerokim aspekcie udostępniania urządzeń o małej ilości obliczeniowej do różnych sieci. Obejmuje to definicję metod transmisji danych, właściwe protokoły czy typy transmitowanych danych, ale również kwestię bezpieczeństwa – szyfrowanie. W swej myśl, IoT jest systemem powszechnym, tudzież urządzenia korzystające z tej koncepcji są powszechnie. Przykładami zastosowań są między innymi transport (np. znaki drogowe, zmieniające ograniczenia prędkości w zależności od warunków pogodowych), sprzęt medyczny (np. mobilne rejestratory EKG – monitorowanie metodą Holtera) i konsumencki (np. inteligentne zegarki monitorujące tętno), przemysł, sieci sensoryczne i wiele innych. Oczekiwana najczęściej cechą takich systemów jest energooszczędność, gdyż podłączone urządzenia często zasilane są baterijnie. IoT obejmuje więc szerokie spektrum dziedzin inżynierii oraz codziennego życia.

Celem niniejszej pracy jest zapoznanie się i zbadanie parametrów nowo wprowadzonego standardu począwszy od Bluetooth 5 – Bluetooth Mesh. Technologia ta obecna od niedawna na rynku, wprowadza istotne zmiany, w tym właśnie tworzenie sieci urządzeń, które mogą ze sobą się wzajemnie komunikować w sposób zdefiniowany przez standard. Badane są zużycie energii układów

¹tutaj: Transistor-Transistor Logic

wspierających Mesh i Bluetooth Low Energy oraz jakość łącza transmisji danych poprzez parametr Packet Error Rate (PER).

Rozdział 2 zapoznaje czytelnika z powszechnymi na rynku rozwiązaniami, protokołami:

- ZigBee
- Thread
- Bluetooth 5 z naciskiem na standard Bluetooth Low Energy i Mesh

Dla każdej wyżej wymienionej technologii, wprowadza się podstawowe pojęcia z nią związane. Rozważania poparte są o specyfikacje opracowane przez właściwe organizacje nadzorujące prace nad każdym ze standardów. Rozwiązania powstają wraz z udziałem firm trzecich, technologicznych, czy producentów elektroniki użytkowej, uwzględniając wspólną wizję i realne zapotrzebowania. Przedstawiając stos technologiczny, niniejsza praca odwołuje się do modelu OSI, jako referencyjnego umożliwiając umieszczenie poszczególnych terminów i nazw we wspólnym standardzie.

Wprowadzając powyższe technologie, uwaga koncentrowana jest na aspekcie konfigurowania sieci składających się z wielu elementów. Tym samym, niezbędne jest wprowadzenie terminologii z jakimi wiąże się każda ze specyfikacji. Nie mniej istotne są możliwe topologie czy sposób przesyłu pakietów danych pomiędzy węzłami – tzw. routing.

Po teoretycznym wprowadzeniu do poszczególnych protokołów, następuje przegląd powstałej dotąd literatury. Oczywiste staje się, iż badanie właściwości sieci, ich skalowalności, przepustowości i innych parametrów cieszy się ogromnym zainteresowaniem zarówno naukowców jak i poszczególnych firm produkujących właściwą do nich obsługę elektronikę. Docelowo, zestawiane są parametry poszczególnych protokołów, pozwalając czytelnikowi na samodzielna kontemplację.

Wymieniony rozdział zakończony jest wprowadzeniem do dwóch prowadzonych i opisywanych eksperymentów: badania zużycia energii urządzeń BLE i BT Mesh oraz badania jakości sieci mierzonej wartością Packet Error Rate. Wprowadza się niezbędną terminologię oraz matematyczny opis wymienionych zjawisk i parametrów celem zastosowania ich w dalszej analizie zebranych danych. Definiowane są cele badawcze oraz sprawdzane hipotezy. Dla eksperymentu mierzącego zużycie energii, sprawdza się pobór prądu elektrycznego w czasie. Zebrane wyniki porównywane są następnie z wartościami symulowanymi z użyciem narzędzi dostarczanych przez producenta mikrokontrolera. Packet Error Rate pozwala na określenie jakości i stabilności połączenia. Głównymi hipotezami badawczymi są: wpływ dystansu na PER – wraz ze wzrostem dystansu oczekuje się wzrostu PER; wpływ środowiska na PER – zakłada się, teren o bogatym tle radiowym (tutaj: teren zurbanizowany) negatywnie wpłynie na badany współczynnik, uzyskując większą utratę przesyłanych pakietów, niż w przypadku o uboższym tle radiowym (tutaj: teren miejski).

Główną treścią celem pracy jest rozdział 3, wprowadzający czytelnika w aspekty praktyczne opisywanych rozwiązań. Mając na uwadze wymienione wcześniej planowane doświadczenia, należy przygotować właściwy tor pomiarowy, by zebrać dane, celem dalszej analizy. Do celów pracy, wykorzystano gotowe komercyjnie dostępne zestawy uruchomieniowe *P-NUCLEO-WB* oraz pomiarowy *X-NUCLEO-LPM01A*. Opierając się na wybranym sprzęcie, wprowadza w szczegóły tworzenia niezbędnych urządzeń wspomagających i oprogramowania koniecznego do przeprowadzenia badań.

Rozdział 1. Wstęp

Przedstawiana jest metodologia poszczególnych eksperymentów. Określone są warunki początkowe, parametry techniczne badanych właściwości. Prezentuje się również metodologię oraz nawiązuje do właściwych, wcześniej wymienionych, wzorów i zależności, o które oparto zebrane dane. Ostatecznie, prezentowane są wykresy nakreślające cechy zebranych pomiarów. Wykresy, oprócz rzeczywistych danych, prezentują linie aproksymacyjne, celem określenia tendencji wzrostów/spadków PER.

Finalnie, przedstawiane są wnioski zarówno z przygotowań do przeprowadzenia eksperymentów jak i przede wszystkim wyciągane są należytne obserwacje i ich konsekwencje, prezentując równocześnie dalsze kierunki rozwoju opisywanej tej pracy analizy.

Rozdział 2

Komunikacja radiowa bliskiego zasięgu

Rozdział ten przedstawia szereg komercyjnie dostępnych technologii komunikacji bliskiego zasięgu z nakierowaniem na IoT. Dobór właściwego rozwiązania do potrzeb niekoniecznie jest prostą i oczywistą decyzją [9, 14]. Stąd też rozeznanie wśród dostępnych opcji jest niezbędną koniecznością.

Pierwszy opisany zostanie protokół ZigBee oparty o specyfikację IEEE 802.15.4. Jest to protokół o sprawdzonej użyteczności komercyjnej będący na rynku najdłużej spośród dobranych. Przedstawia się jego ogólną architekturę i podstawową nomenklaturę.

Thread jest stosunkowo nowym standardem na rynku. Urządzenia oparte o ten protokół stają się coraz powszechniejsze o czym świadczy między innymi liczba producentów wspierających jego rozwój [33]. Nie bez znaczenia jest również postrzeganie Thread przez branżę, upatrując w nim przyszłość rozwiązań IoT [4].

Ostatnim, tytułowym protokołem jest Bluetooth 5 wraz z jego energooszczędną odsłoną Low Energy. Jest to najnowsze rozwiązanie na rynku z opisywanych, które ze wsparciem organizacji Bluetooth SIG może stać się jednym z największych graczy wśród protokołów z przeznaczeniem do automatyki domowej.

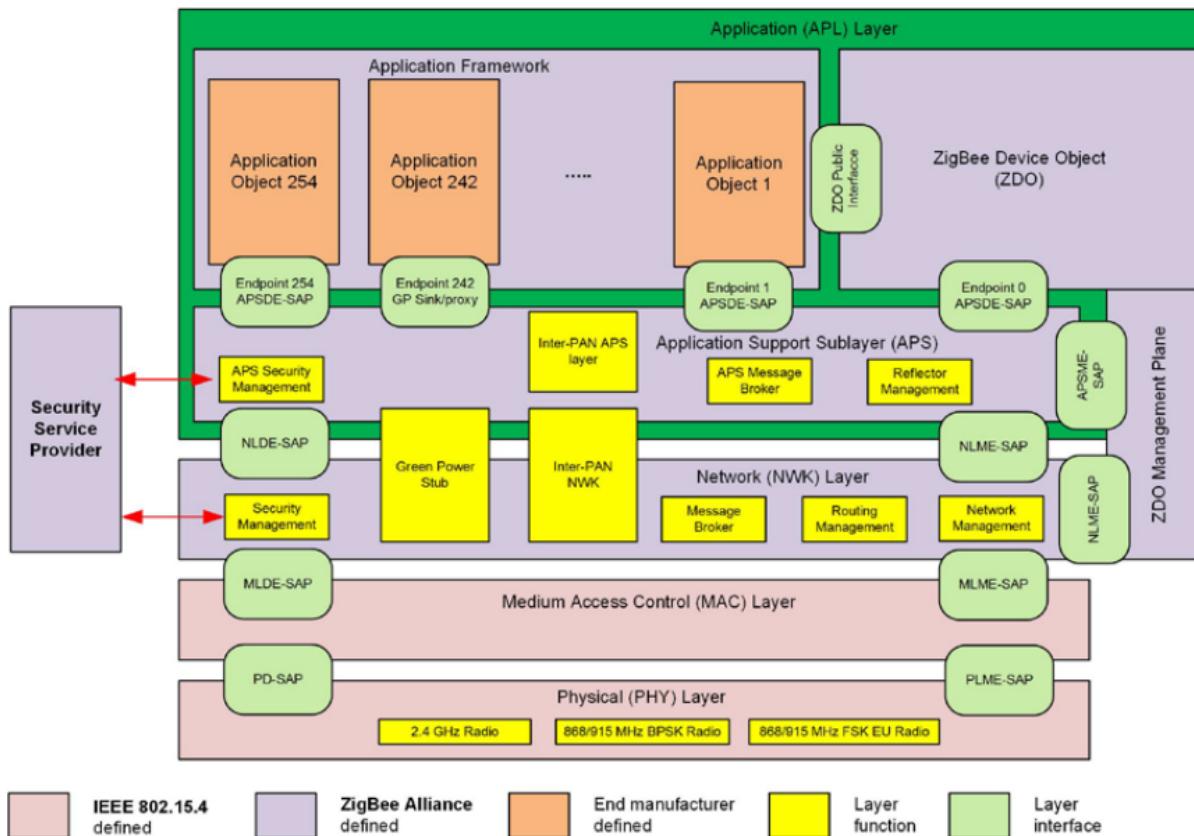
Dla każdego z wymienionych protokołów, rozdział przedstawia krótki rys historyczny, architekturę stosu wraz z próbą odwołania się do modelu referencyjnego OSI. Przedstawia się parametry transmisji danych i niezbędną nomenklaturę wprowadzoną przez standard. Natępnie, następuje próba porównania tychże standardów. Porównywane są takie cechy jak maksymalna dozwolona ilość węzłów, przewidywane topologie, częstotliwości transmisji danych itd.

Bazując na zebranej wiedzy, w szczególności dotyczącej BLE i Mesh, rozdział wprowadza czytelnika teoretycznie do dwóch przeprowadzonych doświadczeń opisywanych w kolejnym rozdziale. Przedstawia się obecny stan wiedzy w danym zakresie, by następnie płynnie przejść do metodologii badań i wyników w kolejnym rozdziale.

2.1 ZigBee

ZigBee jest standardem transmisji bezprzewodowej zapewniający niskokosztową platformę możliwą do zastosowania w elektronice użytkowej, automatyce domowej, wszelkiego rodzaju sensorach (w szczególności przemysłowych i medycznych) jak również grach i zabawkach. Pierwsza specyfikacja opublikowana została w grudniu 2004 roku będąc ciągle aktualizowana, z najnowszą jej wersją będącą datowaną na marzec 2017 roku [44].

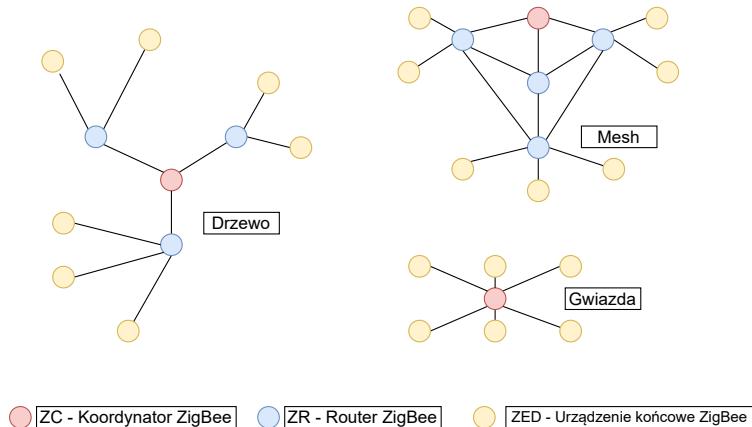
Architektura ZigBee oparta została o IEEE 802.15.4. Definiuje ona fundamentalne zagadnienia: warstwę fizyczną (z ang. *Physical Layer*) (PHY) i warstwę sterowaną dostępem do medium transmisyjnego (z ang. *Medium Access Control*) (MAC). Warstwa fizyczna odpowiada za funkcjonowanie radia, wskaźnik jakości łącza (z ang. *Link Quality Indication*) (LQI), transmisję danych i odbiór pakietów poprzez łącze fizyczne. Definiuje dozwolone częstotliwości działania, szerokości pasma, rodzaj modulacji i dozwoloną przepustowość danych wyrażonych w bitach na sekundę. Warstwa MAC odpowiada za komunikację w wyższych warstwach stosu. Obejmuje to między innymi zarządzanie dostępem do kanałów, walidacja ramek danych, informację zwrotną o otrzymaniu i przetworzeniu danych *Acknowledgement* (ACK) oraz zapewnia odpowiednie uchwyty celem umożliwienia wdrożenia mechanizmów zabezpieczeń.



Rysunek 1. Architektura stosu ZigBee. Źródło: [44]

Standard wprowadza również pojęcie topologii uwzględniając tym samym sposoby, w jakich można zorganizować sieć poszczególnych urządzeń. Definiowane są dwie opcje połączeń: gwiazda, peer-to-peer. Topologia gwiazdy pozwala podłączenie wielu węzłów uwzględniając fakt, iż komunikacja odbywa się za pośrednictwem koordynatora bezprzewodowa sieć o zasięgu osobistym (z ang. *Personal Area Network*) (PAN), będący tożsamy z Urządzeniem w Pełni Funkcjonujące (z ang. *Full Functioning Device*) (FFD). W przypadku konfiguracji rówieśniczej, urządzenia mogą komunikować się dodatkowo między sobą, zapewniając możliwość ustanowienia innych struktur, m.in. Mesh. Standard wprowadza określenie Urządzenia o Zredukowanej Funkcjonalności (z ang. *Reduced Function Device*) (RFD), będące najczęściej urządzeniem o prostej funkcjonalności niewymagającym dużych ilości danych do funkcjonowania, o zredukowanej potrzebie na zasoby sprzętowe [7].

Specyfikacja ZigBee, opierając się na dokumentach IEEE 802.15.4, wykorzystuje częstotliwości 868/915MHz (w zależności od regionu Europa albo USA/Australia) oraz 2.4GHz [44]. Umożliwia transfer z przepustowością do 250 kbps [18]. Omawiany standard wprowadza swoje dodatkowe warstwy komunikacji do stosu: warstwa sieciowa ZigBee (z ang. *ZigBee network layer*) (NWK) oraz warstwa aplikacji (z ang. *Application Layer*) (APL) – Rysunek 1. Warstwa aplikacji, będącą najwyższą w hierarchii, składa się z wielu składowych. Podwarstwa wsparcia aplikacji (z ang. *Application Support Sublayer*) (APS) odpowiada za komunikację pomiędzy NWK a warstwami wyższymi. Oferuje między innymi parowanie urządzeń, przekazywanie wiadomości, adresację, zajmuje się fragmentacją pakietów i zapewnia niezawodny transport danych. Obiekt urządzenia ZigBee (z ang. *Zigbee Device Object*) (ZDO) w głównej mierze odpowiada za wyszukiwanie urządzenia i usług ZigBee [27, 44]. Nadaje on również role urządzeniom sieci.



Rysunek 2. Topologie sieci ZigBee.

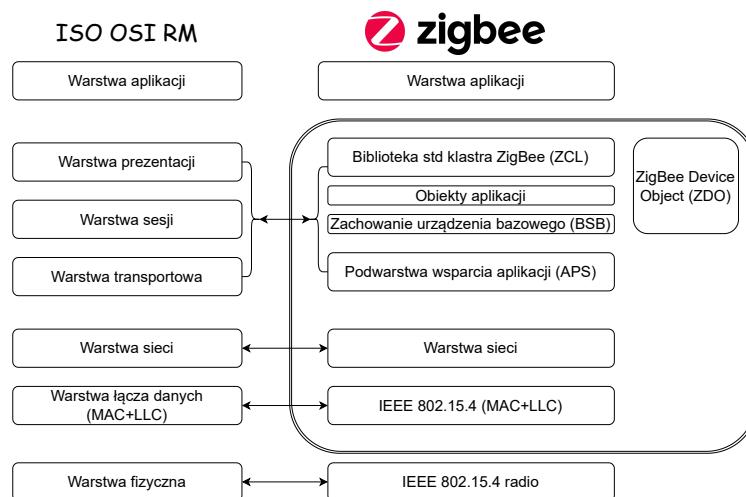
ZigBee wprowadza trzy główne definicje ról urządzeń rejestrowanych do wewnętrz sieci:

- Koordynator ZigBee (z ang. *ZigBee Coordinator*) (ZC) – węzeł odpowiadający za utworzenie i utrzymywanie skoncentrowanej sieci, wybór wymaganych parametrów, dodawanie nowych węzłów.

- Router ZigBee (z ang. *Zigbee Router*) (ZR) – węzeł odpowiadający za przekazywanie danych, który również może przyjąć rolę urządzenia końcowego.
- Urządzenie końcowe Zigbee (z ang. *Zigbee End Device*) (ZED) – węzeł końcowy który odbiera i wysyła dane bez możliwości ich routowania.

Warstwa sieci umożliwia adaptację trzech rodzajów topologii: gwiazda, drzewo i mesh – Rysunek 2. Typ gwiazdy kontrolowany jest przez jednego koordynatora. Topologia drzewa pozwala zastosować hierarchiczne sposoby routingu pakietów. Typ mesh z kolei pozwala na pełną komunikację peer-to-peer między węzłami [44]. Zestawienie poszczególnych warstw z modelem referencyjnym OSI znajduje się na Rysunku 3.

ZigBee umożliwia wykorzystanie następujących metod routingu. Metoda oparta o tablicę trasowania¹ zakłada, iż każdy z węzłów posiada strukturę przechowującą adresy kolejnych, otaczających go węzłów. Raz wysłana wiadomość, będzie korzystać z tej informacji, by przesyłać pakiet do miejsca docelowego. W przypadku niepowodzenia, pierwotny węzeł otrzyma błąd, by ewentualnie podjąć dalszą decyzję o ponownym wyznaczeniu trasy. Standard przewiduje wysyłanie również pakietów przy wykorzystaniu rozgłoszenia z możliwością wyboru roli danego urządzenia. Możliwy jest również multicast. Ostatnią opcją trasowania jest metoda wiele-do-jednego (źródła) [18].



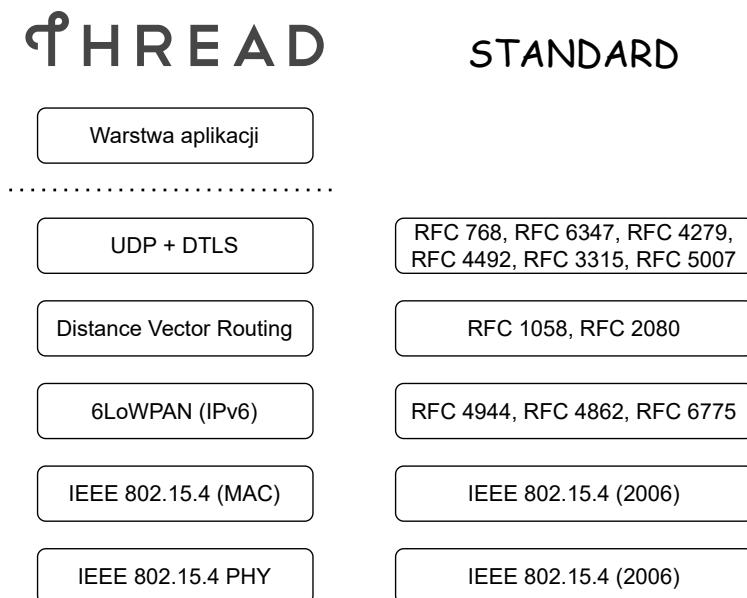
Rysunek 3. Zestawienie warstw stosu ZigBee z modelem referencyjnym OSI. Źródło: [27]

ZigBee wprowadza termin *profile*, będący kontraktem pomiędzy komunikatami wysyłanymi pomiędzy urządzeniami. Definiuje on logiczną strukturę danych i zapewniając kompatybilność pomiędzy platformami różnych producentów. Cechą tą charakteryzują się przede wszystkim profile publiczne zdefiniowane przez ZigBee Alliance. Poszczególni producenci mogą również opracować własnościowe, zamknięte struktury do tworzenia wewnętrznych sieci, gdzie kompatybilność pomiędzy urządzeniami wielu producentów nie jest wymagana [27, 44].

¹z ang. *Routing Table*

2.2 Thread

Thread jest protokół to do zastosowań IoT mający swoje podstawy w standardzie IEEE 802.15.4. Umożliwia on tworzenie rozwiązań o niskim zużyciu energii, przy jednoczesnej koncentracji na bezpieczeństwie opierając adresację o powszechnie znany IPv6. Wybór IPv6 zapewnia płynną integrację z powszechną infrastrukturą i Internetem włącznie. Rozwiązanie jest przez to elastyczne i mniej podatne na starzenie się technologii. Same natomiast produkty oparte o Thread mogą być wdrożone na rynek szybciej dzięki powszechności internetowych narzędzi deweloperskich. Pierwsza wersja specyfikacji została udostępniona w 2014 roku i pozostaje rozwijana do dziś [32].



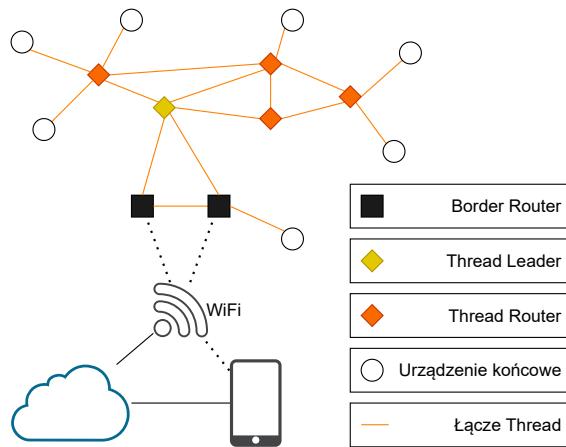
Rysunek 4. Stos Thread i odpowiadające mu specyfikacje RFC/IEEE. Źródło: [8]

Thread wykorzystuje standard IEEE 802.15.4. Transmisja danych odbywa się na częstotliwości $2.4GHz$ z przepustowością do 250 kbps. Protokół jest zoptymalizowany do wykorzystywania dużej liczby węzłów wchodzących w skład sieci [8]. Celem organizacji sieci w spójny zbiór fizycznych i logicznych obiektów, standard wprowadza następującą nomenklaturę. Rolą węzła typu router jest przekazywanie pakietów w sieci oraz nadzorowanie dostępu do sieci, podczas gdy radio takiego urządzenia ciągle jest w aktywnie. Urządzenie końcowe (z ang. *End Device*) (ED) jest urządzeniem przeważnie komunikującym się z jednym routerem będącym jego rodzicem, nie przekazuje pakietów dla innych sieci i możliwością deaktywacji radia celem ograniczenia zużycia energii. Dodatkowo, definiuje się następujące typy urządzeń:

- Pełne urządzenie Thread (z ang. *Full Thread Device*) (FTD) – urządzenie posiadające ciągle włączone radio, mapujące adresację IPv6
 - Router
 - Urządzenie z możliwością działania jako router (z ang. *Router Eligible End Device*) (REED) – urządzenie, które można wykorzystywać jako router

- Pełne urządzenie końcowe (z ang. *Full End Device*) (FED) – urządzenie, którego nie można wykorzystywać jako router
- Minimalne urządzenie Thread (z ang. *Minimal Thread Device*) (MTD) urządzenie przekazujące komunikaty do rodzica.
 - Minimalne urządzenie końcowe (z ang. *Minimal End Device*) (MED) – urządzenie, którego radioodbiornik zawsze pozostaje włączony. Nie wymaga periodycznego pobierania wiadomości z urządzenia-rodzica
 - Usypiane urządzenie Thread (z ang. *Sleepy End Device*) (SED) – urządzenie wzbudzające radioodbiornik okazjonalnie celem pobrania wiadomości.

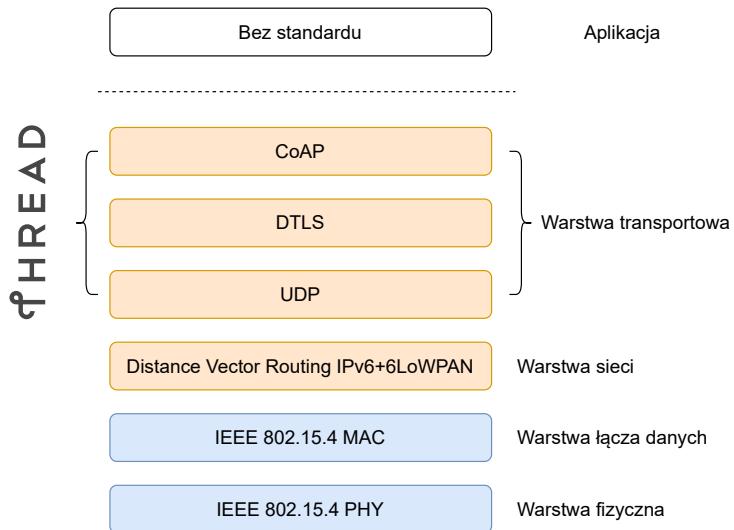
Standard przewiduje dodatkowe typy jak Thread Leader, będący dynamicznie i automatycznie wybieranym węzłem zarządzający pozostałymi Router'ami w sieci. Border Router (router brzegowy) służy za bramkę konwertującą komunikaty przesypane wewnętrz sieci Thread do sieci zewnętrznych takich jak Internet [12].



Rysunek 5. Podstawowa topologia sieci Thread wraz z urządzeniami. Źródło: [34]

Dane w sieci przesyłane są w oparciu o standard *6LoWPAN*². Protokół ten został zoptymalizowany w ten sposób, by wysyłać maksymalną możliwą ilość danych z użyciem jednego pakietu celem minimalizacji fragmentacji pakietów, redukując tym samym narzuć na CPU i zużycie energii. Thread umożliwia stosowanie również protokołów znanych z sieci opartych o model TCP/IP. Tak więc standard ten obsługuje m.in. Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP), Transport Control Protocol (TCP). Topologia tym samym również zależy od ilości węzłów typu router, tworząc albo sieć gwiazdy, albo mesh. Thread obsługuje do 32-óch routerów, gdzie każdy z nich może obsługiwać do 511 ED. Routing odbywa się na zasadach znanych z Internet Protocol (IP) wykorzystując w tym celu protokół zbliżony do Routing Information Protocol (RIP), będący zoptymalizowany do wymagań IoT pod względem zużycia energii [8].

²IPv6 Over Low Power Wireless Personal Networks



Rysunek 6. Thread w zestawieniu z modelem OSI. Źródło: [8]

Thread będący otwartym standardem może zostać wdrożony przez producenta samodzielnie bądź wybrać stos otwartoźródłowy – Rysunek 6. Jednym z takich stosów jest OpenThread³ będący opracowany przez firmę Google.

Co warto dodać, Thread nie definiuje formatu danych jaki jest wysyłany pomiędzy węzłami. Innymi słowy, warstwa aplikacji jest niestandardyzowana [8]. Obecnie trwają prace nad opracowaniem wspólnego i wolnościowego interfejsu. Jednym z takich projektów jest Matter⁴.

2.3 Bluetooth 5 - Bluetooth Low Energy

Komunikacja bezprzewodowa 2.4GHz Bluetooth ma długą historię rozpoczęającą się 1998 roku [3]. Standard przeszedł wiele transformacji do postaci dzisiejszej 5.3 [41]. Niniejsza praca koncentruje się na Bluetooth 5, kładąc szczególny nacisk na Bluetooth Low Energy (BLE).

Bluetooth 5.0, ogłoszony w 2017 roku, wprowadza szereg zmian u podstaw komunikacji radiowej. Względem poprzedzającej wersji Bluetooth (BT), zmodyfikowano działanie warstwy fizycznej osiągając dwukrotnie wyższe teoretyczne transfery danych, wynoszące obecnie 2 Mbps – oznaczane jako LE 2M PHY. Zwiększo również dystans na jakim BLE jest w stanie operować. Poprzednie wersje BLE umożliwiały rozgłaszczenie wyłącznie na 3 kanałach: 37, 37 i 39. Specyfikacja 5.0 zmieniła ten stan rzeczy umożliwiając wykorzystywanie wszystkich kanałów, zwiększając również ilość danych które można wysłać w pakiecie ogłoszeniowym. Usprawniono również algorytm⁵ doboru kanału poprawiając jakość transmisji [42].

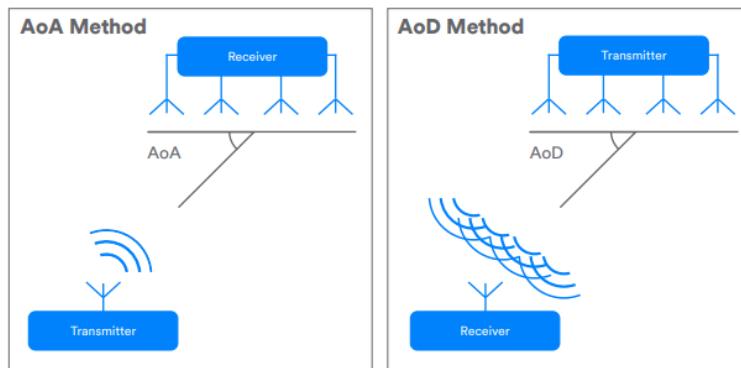
Specyfikacja BT 5.1 z roku 2019 wprowadza szereg usprawnień jak i nową funkcjonalność. Ową funkcjonalnością jest możliwość określenia kierunku badając różnicę w sile sygnałów z nadajników.

³<https://openthread.io/>

⁴<https://csa-iot.org/all-solutions/matter/>

⁵adaptive frequency hopping

Definiuje się dwa sposoby: Kąt Przyjścia (z ang. *Angle of Arrival*) (AoA) i Kąt Przyjścia (z ang. *Angle of Departure*) (AoD). Pierwszy z nich polega na wysłaniu sygnału do macierzy odbiorników. W metodzie AoD to odbiornik odbiera sygnał z wielu nadajników mierząc przy tym fazę fali radiowej w czasie. Te funkcjonalności mają za zadanie przygotować podwaliny pod przyszłe systemy lokalizacji w czasie rzeczywistym⁶ oraz systemów nawigacji wewnętrz pomieszczeń⁷[38].



Rysunek 7. Lokalizowanie kierunku dla standardu Bluetooth 5.1.. Źródło: [38]

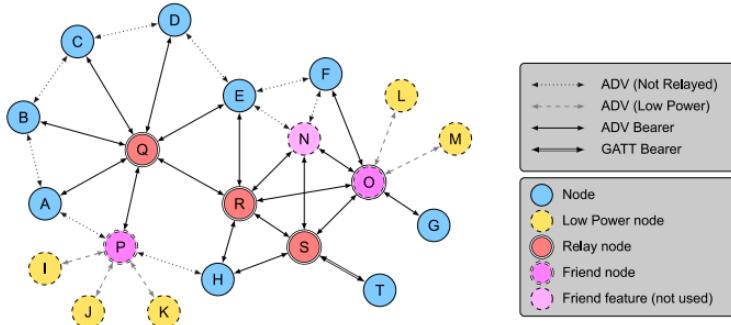
Bluetooth 5.2 z 2020 roku wprowadza nowe funkcjonalności. Ustanawia protokół Ulepszony Protokół Atrybutów (z ang. *Enhanced Attribute Protocol*) (EATT), będący rozwinięciem znanego wcześniej Protokół Atrybutów (z ang. *Attribute Protocol*) (ATT) – protokołu będącego u podstaw komunikacji z użyciem BLE. Wspomniany standard zmienia również sposób zarządzania energią. *LE Power Control* umożliwia dynamiczną optymalizację parametrów transmisji danych pomiędzy połączonymi urządzeniami poprzez monitorowanie siły sygnału. Nowa funkcjonalność poprawia również jakość transmisji w częstotliwościach 2.4GHz, tak popularnych wśród innych protokołów transmisji danych. Kolejnym, nowo wprowadzonym rozwiązaniem jest *LE Isochronous Channels* – kanał izochroniczny. Umożliwiać ma ono transmisję danych do wielu urządzeń docelowych, które zostaną odebrane i przetworzone w tym samym czasie. Funkcjonalność ta przewiduje główne zastosowanie w transmisji audio z użyciem BLE. Potencjalnymi scenariuszami użycia są: współdzielenie audio z osobami w pobliżu, asysta słuchu, telewizja czy nawet ogłoszanie komunikatów w wielu językach np. na pokładzie samolotu [39].

Specyfikacja 5.3 z 2021 roku wprowadza m.in. usprawnienia bezpieczeństwa dla klasycznego BT BR/EDR⁸, określając minimalną, akceptowalną długość klucza. Zapewnia również usprawnienia w kwestii zmiany parametrów transmisji danych – *Connection Subrating*. Zmniejsza ono opóźnienie występujące podczas zmiany wartości parametrów zapewniając lepsze doświadczenia z użytkowania oraz zwiększając żywotność rozwiązań zasilanych baterijnie [41].

⁶RTLS - Real-Time Locating System

⁷IPS - Indoor Positioning System

⁸Basic Rate/Enhanced Data Rate



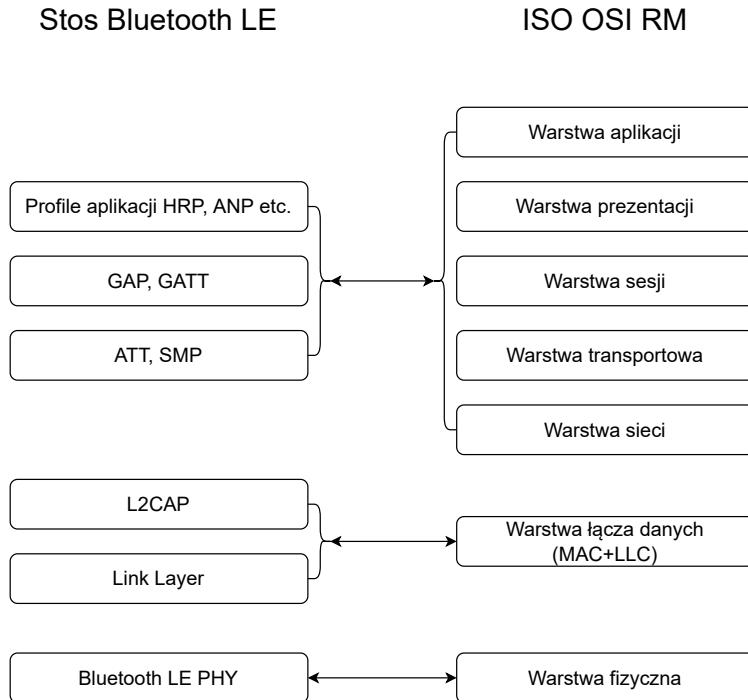
Rysunek 8. Przykładowa topologia BT Mesh z uwzględnieniem rodzajów węzłów. Źródło: [11]

Kolejnym świeżym elementem w stosie BT to Bluetooth Mesh formananie wprowadzony w 2017 roku. Jest to sieć wiele-do-wielu oparta o stos Bluetooth Low Energy rozszerzając jego możliwości. Architektura warstw Mesh wygląda następująco [11]:

- Warstwa modelu (*Model Layer*) – standaryzuje rodzaje operacji i wymienianych wiadomości dla zwyczajowych przypadków użycia
- Warstwa podstawowa modelu (*Foundation Model Layer*) – określa stany, wiadomości i modele niezbędne do konfiguracji i zarządzania siecią
- Warstwa dostępu (*Access Layer*) – definiuje format danych, kontroluje szyfrowanie wiadomości, sprawdza poprawność wiadomości pod kątem przekierowania ich do wyższych warstw
- Góra warstwa transportowa (*Upper Transport Layer*)
- Dolna warstwa transportowa (*Lower Transport Layer*)
- Warstwa sieci (*Network Layer*)
- Warstwa elementu nośnego/okaziciela (*Bearer Layer*) – definiuje sposób jak wiadomości są wysyłane pomiędzy węzłami
- Specyfikacja źródłowa BLE (*Bluetooth Low Energy Core Specification*) – stos BLE opisany zgodnie ze specyfikacją

Specyfikacja źródłowa BLE definiuje niższe warstwy wykorzystywane przez Mesh. Stanowią one analogiczną podstawę, co IEEE 802.15.4 dla protokołów ZigBee czy Thread. Poszczególnym protokołom składowym można przypisać analogiczne warstwy modelu referencyjnego OSI, co zostało ukazane na Rysunku 9.

Węzłem określone jest urządzenie fizyczne będące zarejestrowane wewnętrz sieci Mesh. Proces, podczas którego następuje owa rejestracja nazywa się aprowizacją (z ang. *Provisioning*). Polega on na wygenerowaniu i wymianie kluczy bezpieczeństwa. Szyfrowanie jest domyślne i stanowi podstawę dla konfiguracji sieci, z której nie można zrezygnować. Pojedynczy węzeł może zawierać wiele elementów, a te z kolei mogą zwierać wiele modeli. Elementem będzie więc niezależna składowa, która może być kontrolowana przez dane urządzenie włączone do sieci. Model definiuje natomiast format przetwarzanych wiadomości, stany w których węzeł może się znajdować ustalając jego zachowanie i cechy [20, 40].



Rysunek 9. Zestawienie stosu BLE i modelu ISO OSI. Źródło: [2]

Węzły mogą mieć następujące dodatkowe specjalne cechy będące fundamentem dla stworzenia właściwej topologii [20, 40]:

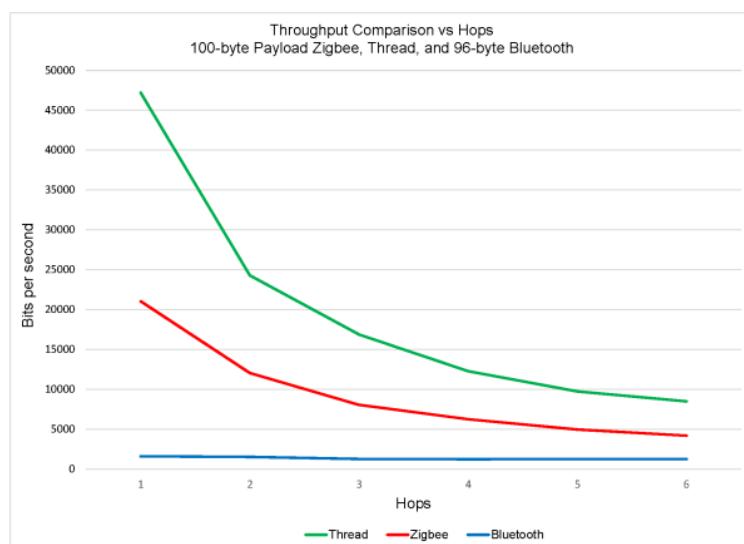
- węzeł przekaźnikowy (*Relay*) – węzeł który przekazuje dane dalej wgłąb sieci zgodnie z algorytmem routingu do określonego momentu Czas Życia *Time-to-live* (TTL)
- węzeł pośredniczący (*Proxy*) – węzeł służący za bramkę, umożliwiający dostęp do sieci z urządzenia BLE. Zrealizowane jest to poprzez udostępnienie właściwej usługi BLE (interfejs GATT).
- para węzłów wykorzystywanych w których jeden z nich jest ograniczony sposobem zasilania:
 - *Low Power Node* (LPN) – węzeł ograniczony zasobami sprzętowymi w tym zasilaniem, mający przeważnie za zadanie działać jak najdłużej na zasilaniu baterijnym
 - *Friend Node* – węzeł będący najczęściej pod stałym zasilaniem, działający w tandemie z LPN, przechowując jego wiadomości i oferując te dane pozostałym węzłom w sieci

Routing w BLE Mesh oparty jest o algorytm *zalewania* (z ang. *Flooding*). W metodzie tej każda wiadomość przekazywana jest do otaczających go węzłów wykluczając węzeł źródła danych. TTL jest również zdefiniowany o ilość przeskoków (z ang. *hops*), które wiadomość musi pokonać nim powinna zostać uznana za martwą.

2.4 Porównanie przedstawionych standardów

Opisane w poprzednich podrozdziałach protokoły i trudność w doborze właściwego rozwiązania do własnych zastosowań jest czymś naturalnym z czym musi zmierzyć się inżynier wraz z zespołem projektowym, by wybrać to *najlepsze*. Poszczególne standardy komunikacji bliskiego zasięgu leżą również w kręgu zainteresowań naukowców badając doświadczalnie ich parametry i zestawiając do przystępnej formy [9, 14].

Istnieje szereg badań zestawiające wymienione wcześniej, aczkolwiek nie ograniczając się wyłącznie do nich, protokoły [1, 6, 14, 15]. Badania te wybierają parametry urządzeń i sieci w postaci czasu oczekiwania⁹ na przesypane wiadomości w zależności od dobranego rozmiaru pakietu czy ilości węzłów, konsumpcji energii, PER, Bit Error Rate (BER), przepustowość i inne parametry sieci.



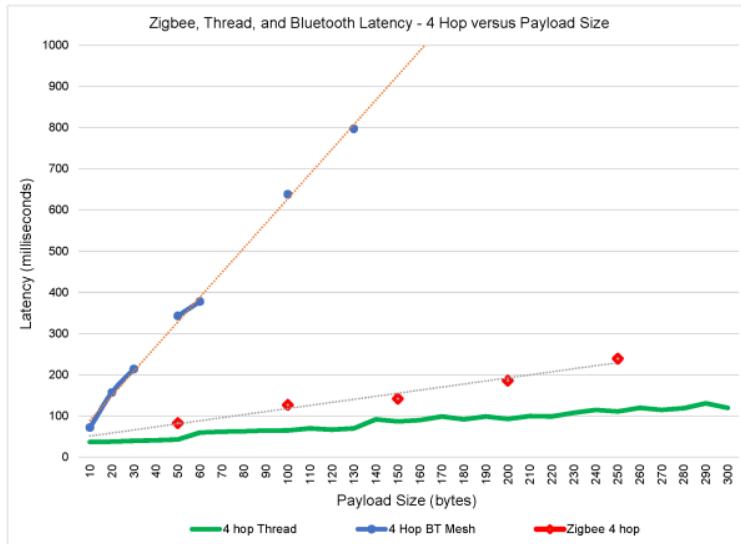
Rysunek 10. Porównanie przepustowości w zależności od ilości przeskoków 100-bajtowej (96-baj dla BT) pakietu podczas transmisji. Źródło: [1]

Badanie przeprowadzone przez firmę *Silicon Labs* bada i porównuje przepustowość i opóźnienie wymienionych rodzajów sieci [1]. Rysunek 10 prezentuje badanie w którym badano maksymalną przepustowość ZigBee, Thread i BT MEsh w zależności od ilości *hopów*. Wykres prezentuje rezultaty testów przesyłu 100-bajtowej wiadomości. Dla protokołów opartych o IEEE 802.15.4 uwidacznia się wykładniczy spadek przepustowości osiągając maksimum w komunikacji peer-to-peer, z przewagą dla Thread. Bluetooth Mesh prezentuje stałe wartości przesyłu wiadomości. Autorzy zaznaczają, że przy tej wielkości wiadomości każdy protokół fragmentuje ją do wielu pakietów.

Inne badanie prezentuje opóźnienia występujące w sieci składających się z 4 węzłów w zależności od wielkości wiadomości – Rysunek 11. Uwidacznia się oczywisty wzrost tej proporcji wraz ze większą ilością wysyłanych bajtów. Różny jest natomiast przyrost. Szczególnie wyróżnia się tutaj Bluetooth Mesh. Ze względu na swój mechanizm routing'u¹⁰ i architekturze stosu, każda wiadomość

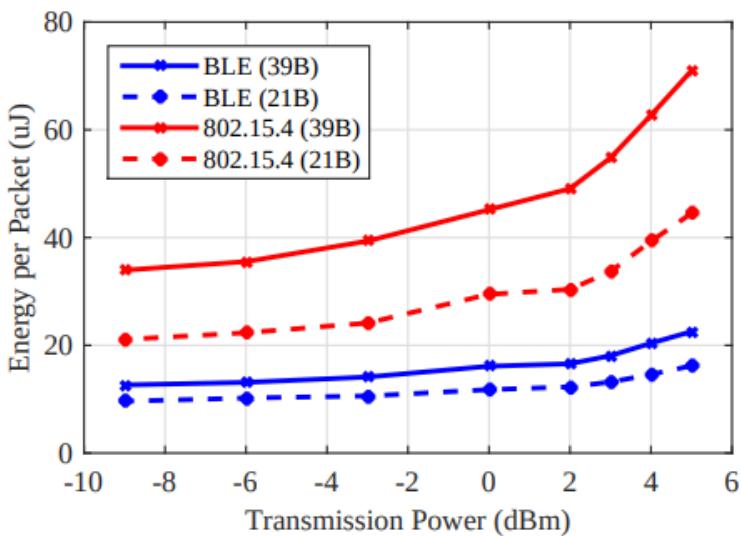
⁹z ang. *latency*

¹⁰przypomnienie - flooding mechanism



Rysunek 11. Porównanie opóźnienia w zależności od wielkości pakietu dla 4-węzłowej sieci różnych standardów komunikacji. Źródło: [1]

powyżej 12 bajtów ulega segmentacji. W połączeniu z mechanizmem routowania sieć ulega zapchaniu powodując *duży ruch* w sieci obniżając jej możliwości transmisji. Jest to oczekiwane zachowanie biorąc pod uwagę architekturę stosu, który udostępniając odpowiednie usługi i usiłuje minimalizować ilość wysyłanych danych ograniczając się najczęściej do jednej ramki.



Rysunek 12. Zależność konsumpcji energii od mocy transmisji danych w zależności od wielkości pakietu i rodzaju protokołu. Źródło: [5]

Badacze analizowali również zużycie energii przez BLE i standard IEEE 802.15.4 z użyciem chipu firmy *Texas Instruments* – CC2650 [5]. Autorzy sprawdzali zależność, w której mierzono ilość energii niezbędnej do transmisji wiadomości o rozmiarach 21 i 39 bajtów na wysokości warstwy PHY przy stałym napięciu zasilania 3.3V. Protokół BLE wymaga niemal dwa razy mniej energii do transmisji

wiadomości niż konkurencyjny protokół. Wraz ze wzrostem mocy nadawanego sygnału, ta proporcja staje się coraz korzystniejsza z energetycznego punktu widzenia dla Bluetooth'a – Rysunek 12.

Protokoły również można porównać statycznie na podstawie dostępnej literatury, a przede wszystkim publicznie dostępnych specyfikacji standardów. Rezultaty takiej analizy obserwuje się w tabeli 1.

Cecha porównywana	Bluetooth Mesh	Thread	ZigBee
Częstotliwość	2.4GHz	2.4GHz	2.4GHz, 868/915MHz
Przepustowość	do 2Mbps	do 250 kbps	do 250 kbps
Max. ilość węzłów	32767 przy max. 126 [0x7f-1] przeskokach[11]	16352 (max. 32 routery, do 511 urządzeń końcowych/router)[12]	64000
Topologia	Mesh (flooding)	Mesh	Mesh, drzewo, gwiazda

Tablica 1. Zestawienie protokołów Bluetooth Mesh, Thread, ZigBee

2.5 Teoretyczne podstawy zaprojektowanych eksperymentów

Celem tego podrozdziału jest zapewnienie minimalnego, teoretycznego wstępu wymaganego do przeprowadzenia właściwych badań. Poniższe punkty prezentują definicje, wzory i symulacje. Zawarte zostaną również hipotezy badawcze, które w praktycznej dalszej części pracy zostaną empirycznie zweryfikowane.

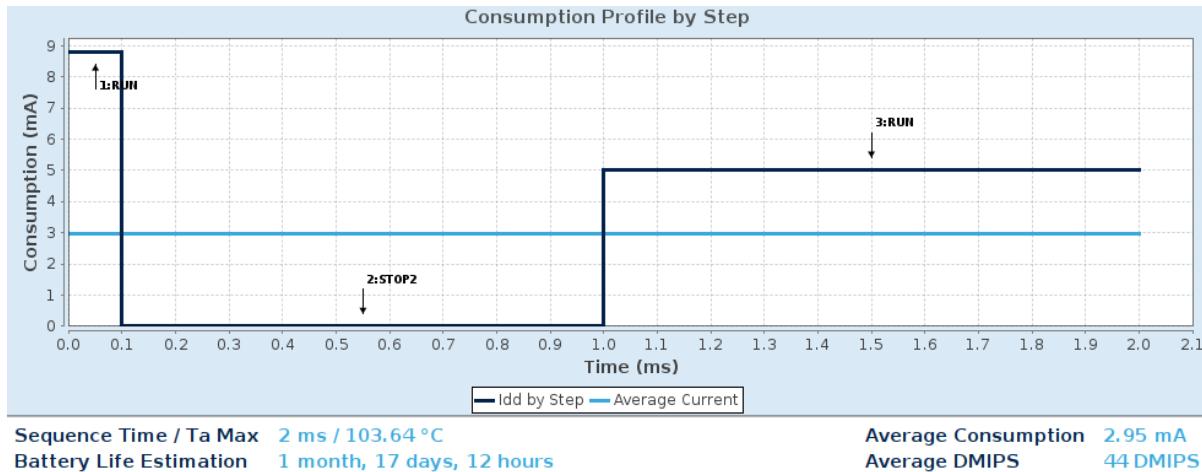
Badanie zużycia energii

Energooszczędność urządzeń elektronicznych ma szczególne znaczenie w przypadku produktów zasilanych ogniwami. Celem optymalizacyjnym jest wydłużenie czasu użytkowania na pojedynczym ładowaniu bądź do kolejnej wymiany baterii.

Przed przystąpieniem do fizycznego eksperymentu, wykonano przykładową symulację z użyciem oprogramowania firmy ST, będącej producentem wybranej platformy opisanej w kolejnym rozdziale. *STM32CubeMX* będący zintegrowanym środowiskiem programistycznym zapewnia moduł o nazwie *PCC* [35]. Umożliwia on symulowanie zużycia energii przez układ zgodnie ze zgrubnymi szacunkami inżyniera.

Postępując zgodnie z dokumentacją, wykonano dwie symulacje. Rysunek 13 odzwierciedla przewidywane zużycie energii urządzenia BLE dla domyślnie proponowanych nastaw sugerowane przez *PCC*. Krok trzeci jest etapem aktywacji radia rozgłaszącego komunikaty o gotowości do połączenia. Szacowany pobór prądu wynosi ok. 5mA, przy średniej 2.95mA. Jako główne źródło zasilania, do celów symulacji, dobrano „wirtualną” baterię 3.3V – Li-SOCL2 (A3400). Umotywowane

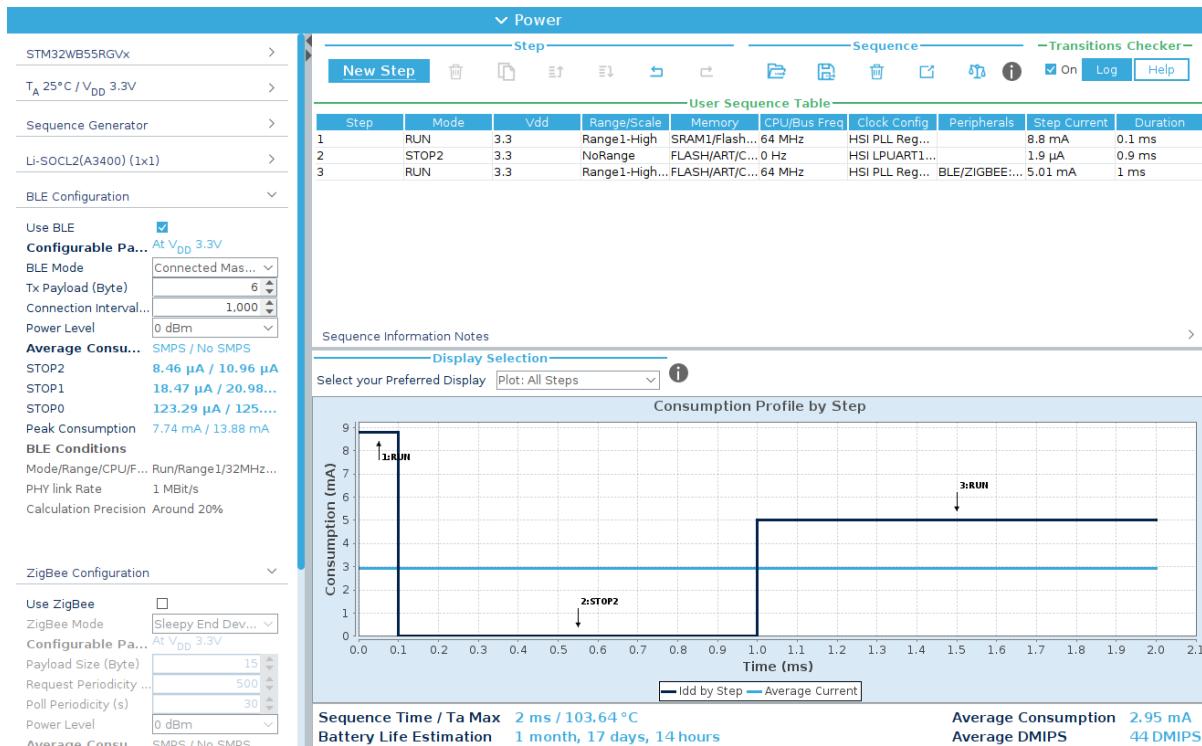
Rozdział 2. Komunikacja radiowa bliskiego zasięgu



Rysunek 13. Symulacja zużycia energii przez urządzenie BLE podczas ogłaszań

jest to wykorzystywaniem analogicznego źródła jak podczas właściwego badania. Oprogramowanie sugeruje, iż bateria zostanie rozładowana po nieco ponad półtora miesiąca użytkowania.

Analogiczną symulację przeprowadzono dla połączonego już urządzenia – Rysunek 14. Dla zadanych parametrów, oczekiwana długość życia układu będzie o dwie godziny dłuższa niż wcześniej wspomniany przypadek.



Rysunek 14. Symulacja zużycia energii przez urządzenie BLE podczas połączenia

Moduł *PCC* niestety nie zapewnia wsparcia do symulacji węzłów sieci Mesh. Stawia się hipotezę, iż zużycie energii będzie zależało od poszczególnie pełnionych ról przez dane urządzenie. Standard przewiduje wykorzystywania zasilania baterijnego przez węzeł LPN. W przypadku takiej konfiguracji, należałoby spodziewać się rezultatów podobnych do symulowanych. Przeprowadzając pomiary, wykorzystano standardowe węzły, stale nasłuchujące za sygnałem będącym serwerem modelu *Generic OnOff*. Spodziewane jest znaczący wzrost zużycia energii właśnie ze względu na wymieniony fakt zaczerpnięty ze specyfikacji.

Zebrawszy dane chwilowych poboru natężenia prądu w czasie, należy wyliczyć właściwe parametry w postaci skonsumowanej energii i mocy. Wykorzystuje się w celu oczywistą zależność fizyczną zaprezentowaną wzorem 1 [19]. Wzór uwzględnia okres trwania akwizycji danych zdefiniowany w docelowym podrozdziale 3.2.

$$E_{\text{całkowita}} = U \cdot \int_{t=0[s]}^{t=50[s]} di \, dt \quad (1)$$

$$P = \frac{E_{\text{całkowita}}}{t} \quad (2)$$

gdzie:

$E_{\text{całkowita}}[J]$ – wykorzystana energia podczas 50s sekundowej sesji rejestracji danych

$P[W]$ – moc użyta podczas 50s sekundowej sesji rejestracji danych

$U[V]$ – napięcie zasilania mikrokontrolera - $3.3V$ - stała

$di[A]$ – prąd w danej chwili

$dt[s]$ – podstawa czasowa całkowania, $0.01s/\text{interwał}$ ($100Hz$) – stała

Badanie Packet Error Rate

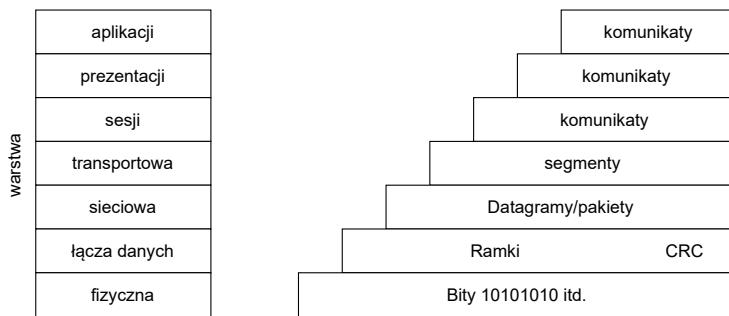
Przed przystąpieniem do badań należy precyźnie zdefiniować wykorzystywaną terminologię.

Pakietem nazywamy pojedynczą porcję danych przetwarzaną na poziomie warstwy sieciowej modelu referencyjnego ISO OSI – Rysunek 15 [16]. Warstwa ta umożliwia routing, adresowanie logiczne oraz przetwarzanie i dostarczanie pakietów.

BLE wprowadza własną nomenklaturę dla poszczególnych warstw sieciowych. Jest to o tyle istotne, iż standard ten nie zapewnia odpowiadających modelowi ISO OSI warstw jeden-do-jednego. Część z tych warstw jest agregowanych w zespół protokołów wyższych warstw – Rysunek 9.

Bazując na definicji modelu OSI oraz stosie BLE, pakietem można nazwać wiadomości będące możliwe blisko warstwy *Link Layer (LL)*. Podobną definicję prezentuje dokumentacja ST: „Pakiet to pojedyncza oznaczona wiadomość wysłana przez jedno i odebraną przez co najmniej jedno urządzenie.”¹¹ [29]

¹¹Tłumaczenie własne



Rysunek 15. Model referencyjny ISO OSI wraz i odpowiadające mu nazwy porcji danych. Źródło: [16]

BLE Mesh dodatkowo wprowadza własne dodatkowe warstwy komunikacji, jak zostało to opisane w podrozdziale 2.3. Każda z wymienionych warstw jest hermetyzowana za pośrednictwem dostarczanego przez producenta Application Programming Interface (API). Middleware może tym samym zostać zamknięty przed dostępem, skutkując brakiem możliwości bezpośredniego nasłuchiwanego pakietów w warstwie LL. Uwzględniając powyższe czynniki, *pakietem* dla BLE Mesh nazywana będzie wiadomość najbliższa warstwie LL.

Posiadając definicję pakietu, PER możliwe staje się zdefiniowanie wzoru, a zarazem znaczenia głównego celu badań. PER jest miarą ilości błędnych pakietów w proporcji do wszystkich wysłanych pakietów, zgodnie ze zdefiniowanym wzorem 3.

$$PER = \frac{s - r}{s} \cdot 100\% \quad (3)$$

gdzie:

s – ilość wysłanych pakietów

r – ilość odebranych pakietów

s-r – ilość niepoprawnych/błędnych pakietów

Eksperyment PER przeprowadzany w warunkach terenowych jest narażony na szereg czynników wpływających na jakość transmisji. Projektowane doświadczenie starało się zawrzeć przynajmniej część z nich, co zostało to opisane w podrozdziale 3.3.

Uznając fakt przeprowadzania badań weryfikujących aspekty komunikacji Bluetooth opartej na częstotliwości 2.4GHz, naturalnym staje się dobór różnych środowisk, w których ten czynnik jest skrajnie różny. Uzasadnia się to faktem interferencji fal. Warunki zurbanizowane zapewniają bogate tło radiowe, w szczególności oparte o częstotliwości będące wykorzystywane przez inne popularne technologie takie jak WiFi. Zakłada się, iż w takim środowisku prawdopodobieństwo kolizji (interferencji) jest większe aniżeli w środowisku względnie radiowo odizolowanym. Tym samym, jedną z badanych hipotez określa się jako dominujący wpływ tła radiowego w warunkach zurbanizowanych na wzrost miary PER. Analogicznie, w warunkach odizolowanych radiowych, oczekiwany jest mniejsze tempo wzrostu PER wraz ze zwiększanym dystansem pomiędzy węzłami.

Czynnikiem który może mieć wpływ na rezultaty jest rozmieszczenie węzłów sieci. Zgodnie z opisem metodologii (podrozdział 3.3), każde z urządzeń układane było bezpośrednio na glebie. Konsekwencją takiego stanu rzeczy jest wpływ zjawiska zwanego rozchodzeniem się fali powierzchniowej. Zgodnie z definicją: „[antena] jest umieszczona na powierzchni ziemi, jeśli zawieszono ją na wysokości mniejszej niż λ nad powierzchnią” [31]. W przypadku wykorzystania komunikacji Bluetooth, definicję spełnia każde położenie urządzenia poniżej linii 12,5cm nad powierzchnią ziemi.

Nie mniej istotna na ostateczne efekty badań jest pogoda w postaci czynników temperatury czy wilgotności. Zmiana w tych parametrach wpływa na wartości tłumienia sygnału.

Rozdział 3

Część doświadczalna

Treść tego rozdziału przedstawia wszelkie kroki podjęte celem realizacji dwóch postawionych zadań: badania zużycia energii oraz PER. Wprowadzając czytelnika w aspekty przygotowawcze eksperymentu i technikalia z nimi związane, przechodzi się do prezentacji wyników i powiązanych analiz opracowanych wykresów.

Pierwszy podrozdział wprowadza w aspekty czysto techniczne i sprzętowe. Uzasadnia wybór platformy jak i oprogramowania w tym Integrated Development Environment - Zintegrowane środowisko programistyczne (IDE). Przedstawia się problem zasilania zestawów uruchomieniowych podczas badań terenowych, jednocześnie prezentując praktyczne, proste i niskokosztowe rozwiązanie problemów. Ze względu na kruchość elementów elektronicznych, wprowadza się projekt a następnie omawia fizyczne wykonanie autorskiej obudowy. Jednakże, głównym elementem tego rozdziału jest skoncentrowana na oprogramowaniu. Zarówno oprogramowaniu mikrokontrolerów jak i PC. Przedstawia się wprowadzone funkcjonalności oraz definiuje ich sposób użycia. Opisuje się kwestię kalibracji uzasadniając potrzebą i potwierdzając jej skuteczność obliczeniami arytmetycznymi bazując na równaniu uzyskanym drogą inżynierii wstępnej. Ostatnim aspektem poruszonym w tym punkcie jest aplikacja PC i jej interfejs użytkownika.

Drugi podrozdział, prezentuje badanie zużycie energii jako całość. Wprowadzi się metodologię według której dokonano akwizycji danych. Następnie, prezentuje się wykresy podając opis do każdego z nich. Badania oparto o dwa główne aspekty – konsumpcja energii przez przez usługę BLE Heart Rate (HRT) oraz model BLE Mesh. Ostatecznie, zestawia się zebrane w ten sposób dane.

Ostatni podrozdział dotyczy eksperymentu PER. Analogicznie do wyżej wymienionego doświadczenia, wprowadza metodologię badań oraz określa kategorie zbieranych zmiennych i definiuje się parametry stałej transmisji. Zebrane dane w próbach terenowych następnie są dzielone na dwie kategorie: PER względem częstości zapytań oraz PER względem odległości między węzłami. Wyniki, w postaci szeregu wykresów, są następnie omawiane.

3.1 Przygotowanie eksperymentu

Eksperimentalna część pracy bezsprzecznie wymaga przygotowania odpowiedniego zestawu sprzętu i narzędzi. Konieczne jest wybranie odpowiedniej platformy sprzętowej wspierającej komunikację bezprzewodową Bluetooth Low Energy w wersji minimum 5.0. Niezbędne jest również oprzyrządowanie pomiarowe, które umożliwia pomiar natężeń prądu poniżej 1mA, ze względu na energooszczędność urządzeń BLE.

Badania zużycia energii wymagają aparatury, która w pełni zarejestruje minima i maksima poboru prądu przy niskich błędach pomiarowych. Na podstawie otrzymanych danych wykonane zostaną właściwe obliczenia ukazujące zużycie energii przez urządzenie dla wielu trybów działania.

Eksperiment PER wymaga przygotowania wielu zestawów uruchomieniowych obsługujących komunikację BLE w konfiguracji Mesh. Dodatkowo, wymagane jest stworzenie dedykowanego oprogramowania na mikrokontroler jak i komputer osobisty. Jest to niezbędne w celu kontroli przepływu doświadczenia jak i akwizycji danych.

Niniejszy rozdział omawia zakres poniesionych przygotowań do przeprowadzenia właściwych eksperymentów.

3.1.1 Sprzęt i oprzyrządowanie

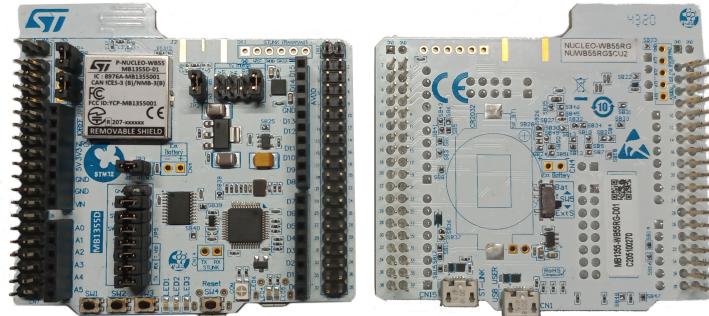
Próby doświadczalne oparto o produkty firmy STMicroelectronics (krócej: ST). Jest to europejska firma obejmująca swym portfolio projektowanie i produkcję elementów elektronicznych. Jedną z ich linii produktów są 32-bitowe mikrokontrolery architektury ARM, znane również jako STM32. Są to rozwiązania cenne na rynku.

Ze względu na powszechność stosowanych rozwiązań, jednak nie ograniczając się wyłącznie do tego kryterium, wybrano mikrokontroler z rodziny STM32WB. Gotowe, komercyjnie dostępne zestawy ewaluacyjne oraz zintegrowany ekosystem stanowią doskonałą podstawę dla badań Bluetooth Low Energy.

P-NUCLEO-WB55

Badania Bluetooth Low Energy wymagały wyboru platformy, która umożliwia eksperimentalną weryfikację wybranych celów badawczych. Ostatecznie, zdecydowano się na wykorzystanie mikrokontrolera *STM32WB55RG*. W celu zapewnienia powtarzalności eksperymentu jak i ze względu na ograniczenia czasowe, docelową platformą badawczą stał się zestaw uruchomieniowy *P-NUCLEO-WB55* [13].

Zestaw ten zgodny jest ze specyfikacją Bluetooth Low Energy v5.0. Dodatkowo, wspiera on inne standardy komunikacji, m.in. Zigbee [25]. Ten fakt może zostać wykorzystany w celu bezpośredniego porównania różnych stow komunikacji bezprzewodowej. Nie jest to jednak celem niniejszej pracy, a stanowi możliwość jej dalszego rozwinięcia.



Rysunek 16. Zestaw uruchomieniowy P-NUCLEO-WB55

X-NUCLEO-LPM01A

Płytkę rozszerzeń X-NUCLEO-LPM01A spełnia wszelkie oczekiwania dotyczące możliwości pomiarowych stawianych przed projektem. Wg oficjalnej dokumentacji [36], układ oferuje:

- Programowalne źródło napięciowe 1,8V do 3,3V
- Dynamiczne próbkowanie w zakresie od 100nA 50mA przy maksymalnej częstotliwości 100kHz z 2%-ową dokładnością pomiarów
- Pomiar statyczny natężenia prądu do 200mA
- Integracja z aplikacją do dedykowaną aplikacją akwizycji danych [22]

Moduł ten nie ogranicza się tylko do wykorzystywania autorskich złącz firmy ST. Schemat wyprowadzeń pinów umożliwia wykorzystanie popularnych platform takich jak Arduino¹. Co istotniejsze, wybrana płytka umożliwia zasilanie dowolnego układu dzięki wyprowadzeniu źródła napięciowego za pośrednictwem pinów (za dokumentacją: złącze CN14). Ten fakt został wykorzystany w badaniach, pozwalając uniknąć kłopotliwej konfiguracji P-NUCLEO-WB55 wymagającej tworzenia dodatkowych ścieżek lutowanych.

Dodatkową cechą tego modułu jest akwizycja danych z użyciem komputera osobistego. Dzięki dedykowanej aplikacji, dostarczanej wraz z modułem, możliwa jest akwizycja danych w czasie rzeczywistym przy wybranych częstotliwościach próbkowania. Zebrane w ten sposób dane służą dalszym analizom, co zostało wykorzystane w niniejszej pracy.

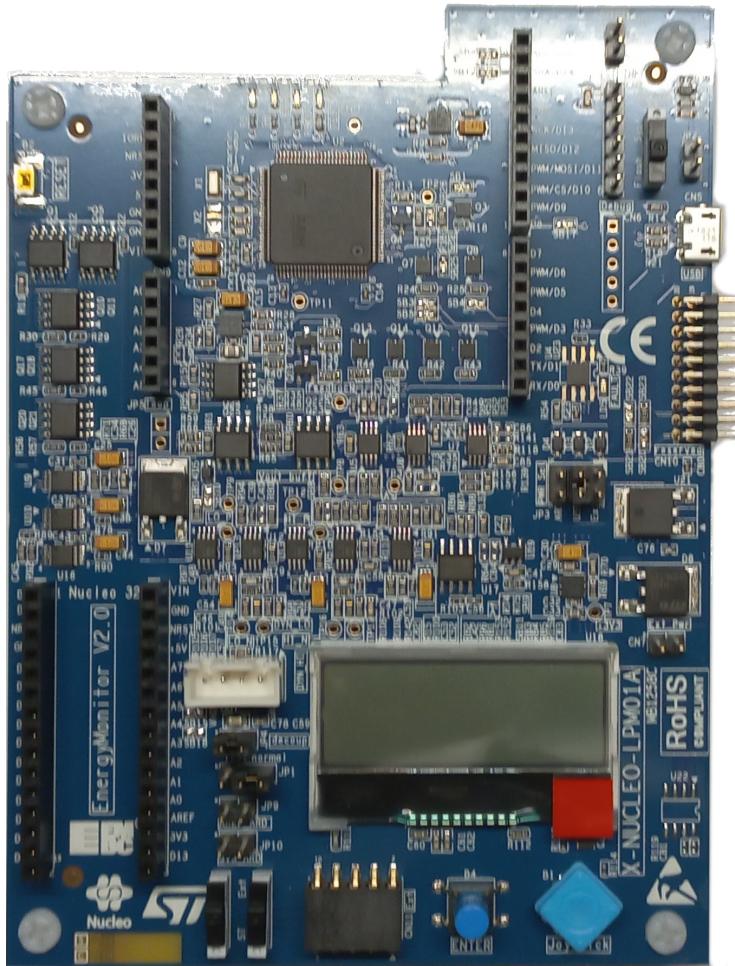
Narzędzia i firmware firmy ST

Firma ST wraz z zestawem uruchomieniowym udostępnia pełne zintegrowane środowisko programistyczne oraz niezbędne biblioteki i certyfikowany firmware:

- **STM32CubeIDE** [21] – multiplatformowe zintegrowane środowisko programistyczne dostarczane przez ST bazujące na otwartoźródłowym środowisku Eclipse².
- **STM32CubeProgrammer** [23] – narzędzie umożliwiające wykonywanie operacji odczytu, zapisu i weryfikacji skompilowanego oprogramowania produktów STM32.

¹<https://www.arduino.cc/>

²<https://www.eclipse.org/ide/>



Rysunek 17. Zestaw pomiarowy X-NUCLEO-LPM01A

- **STM32CubeMonitor-Power** [22] – oprogramowanie służące akwizycji danych pod postacią chwilowego poboru prądu elektrycznego m.in. w zestawie X-NUCLEO-LPM01A Rysunek: 17.
- **Firmware STM32CubeWB** [24] – zestaw bibliotek, narzędzi oraz przykładów przeznaczonych dla mikrokontrolerów rodziny STM32WB. W skład tego repozytorium wchodzą zależności takie jak: skompilowany, zamknięty firmware ko-processora dla różnych stosów połączeń bezprzewodowych; przykłady programów wykorzystujące biblioteki HAL jak i również bezpośrednio rejestrów; przykłady BLE; przykłady BLE Mesh.

3.1.2 Narzędzia i elementy dodatkowe

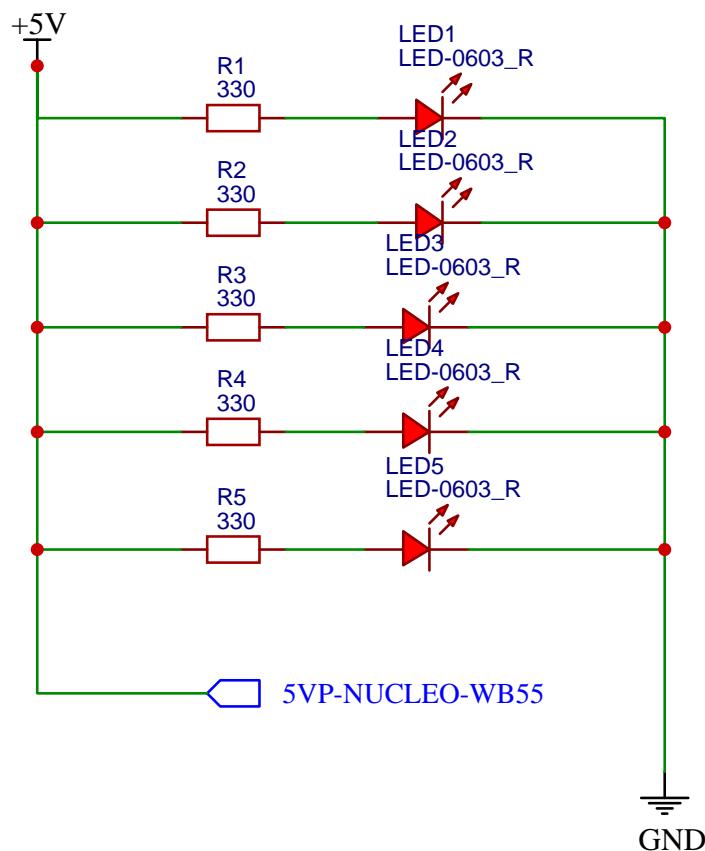
Zasilanie

Próby terenowe wymagają odrębnego, właściwego zasilania. Wybrane zestawy uruchomieniowe wykorzystują ustandaryzowane złącze komunikacyjne USB. Złącze to oprócz komunikacji oferuje zasilanie linią +5V. Przeprowadzając właściwe eksperymenty wykorzystano komercyjnie dostępne banki energii

tzw. powerbanki. Są to akumulatory litowo-jonowe lub litowo-polimerowe z wbudowanym systemem zarządzania baterią³ i właściwymi przetwornicami impulsowymi w celu zapewnienia odpowiedniego napięcia zasilania.

Elektronika wbudowana w magazyny energii automatycznie wyłącza zasilanie w przypadku braku podłączonego urządzenia lub gdy dane urządzenie pobiera marginalnie niskie wartości prądu. Służy to ochronie cennego i delikatnego akumulatora. Jest to cecha korzystna z punktu widzenia konsumenta kupującego powerbank w celu szybkiego ładowania urządzeń elektronicznych. Niestety, zaleta ta staje się wadą w przypadku przeprowadzanych doświadczeń. Podłączając moduł P-NUCLEO-WB55, urządzenie po pewnym okresie działania wyłącza się na skutek uruchomionych zabezpieczeń w źródle energii.

W celu przeciwdziałania temu zjawisku, skonstruowano trywialne urządzenie, które włutowane równolegle w linię zasilania 5V, konsumuje ok. 100mA prądu: $0,5W = 5V \cdot 0,02A \cdot 5[\text{diod}]$. Doświadczalnie stwierdzono, że tak włączony odbiornik energii umożliwia stabilne działanie włączonego do sieci modułu STM32. Schemat zaprezentowano na Rysunku 18



Rysunek 18. Odbiornik energii

³Battery Management System (BMS) – ang. Battery Management System

Obudowa – druk 3D

Kolejnym elementem niezbędnym w celu przeprowadzenia eksperymentów terenowych jest zapewnienie ochrony mechanicznej wybranych zestawów uruchomieniowych. Podczas prób, elementy mogą zostać fizycznie uszkodzone poprzez m.in. wyłamanie fragmentu Printed Circuit Board (PCB) (w tym anteny), uszkodzenie pinów etc.

Wykorzystując techniki projektowania wspomaganego komputerowo⁴, zaprojektowano autorską obudowę. Głównym celem projektowym było zapewnienie minimalnej ochrony mechanicznej, jednocześnie redukując możliwy wpływ materiału na tłumienie sygnału. Elementy interfejsu: przyciski i antena, zostały celowo wyeksponowane. Rzut izometryczny przedstawiony został na Rysunku 19.



Rysunek 19. Model obudowy dla zestawu P-NUCLEO-WB55

Obudowa wykonana została z materiału Polylactic Acid (PLA)⁵ techniką druku 3D Fused Deposition Modeling (FDM)⁶/Fused Filament Fabrication (FFF)⁷. Jednobryłowa konstrukcja podyktowana została chęcią uproszczenia wydruku, kosztem zwiększonej trudności doboru tolerancji dla umieszczanych wewnętrz płytek PCB. Obudowa przewiduje wsuwanie modułu Nucleo do wewnętrz, zapewniając jednoczesne pewne jego mocowanie.

Dobór koloru wydruku również nie był przypadkowy. Przewidując potencjalne lokalizacje przeprowadzanych doświadczeń, wybrano kolor możliwie jaskrawy. Podstawowym założeniem było ułatwienie dostrzeżenia obudowy, a wraz z obudową również zestawu uruchomieniowego, pośród flory leśnej czy terenów zurbanizowanych. Ostateczne wykonanie obudów zaprezentowano na Rysunku 20.

⁴Computer-aided Design (CAD) – ang. *Computer-aided Design*

⁵PLA – ang. *Polylactic acid*, polilaktyd

⁶FDM – ang. *Fused Deposition Modeling*

⁷FFF – *Fused Filament Fabrication*



Rysunek 20. Obudowa zestawu uruchomieniowego Nucleo wykonana w technologii druku 3D – realizacja

3.1.3 Oprogramowanie mikrokontrolera

Oczywistym faktem jest, iż zdefiniowane wcześniej rodzaje eksperymentów wymagają odpowiedniego oprogramowania. Zarówno oprogramowania dla mikrokontrolera jak i komputera PC pozwalającego wykonać i nadzorować wybrane doświadczenia. Podrozdział ten prezentuje kroki podjęte w celu stworzenia odpowiednich programów dla badania poboru zużycia energii i utraty pakietów podczas transmisji danych.

Pierwszym krokiem jest wybór odpowiedniego zestawu narzędzi i frameworka, na którym oparte będą wszelkie dalsze pracy. Rozpatrywane były dwa odrębne stosy technologiczne: Zephyr OS⁸ i natywny dostarczany przez ST. Zephyr OS, będący systemem operacyjnym czasu rzeczywistego (RTOS), jest kompletnym zestawem narzędzi i metodologii wytwarzania oprogramowania dla mikrokontrolerów różnych producentów. Możliwość przeniesienia kodu na inną platformę z minimalną ilością modyfikacji było opcją nadzwyczaj interesującą. Jednakże, pomimo oficjalnego wsparcia mikrokontrolera STM32WB55, nie udało się poprawnie zainstalować dostarczanych przykładów na urządzeniu w sposób umożliwiający ich działanie. Stąd zaniechano dalszych prób wykorzystania tego systemu operacyjnego w projekcie.

Drugim oczywistym kandydatem jest stos oprogramowania dostarczony przez ST. Jest to niejako domyślny wybór. Wybrano więc stos oprogramowania dla mikrokontrolera w wersji v1.13.2, jako najnowszy dostępny w chwili rozpoczęcia prac. Zestaw bibliotek i narzędzi dostarczanych przez producenta został wcześniej przez niego przetestowany dając niejako gwarancję spełnienia minimalnych standardów umożliwiających m.in. certyfikację urządzenia jako wspierającego BLE i BLE Mesh.

Badanie zużycia energii (3.2), wykorzystuje dostarczaną wraz ze stosem przykładową aplikację opracowaną przez producenta mikrokontrolera. Owa aplikacja, BLE HRT, wykorzystuje zdefiniowany przez Bluetooth SIG profil *HeartRate* publikując regularnie, co sekundę odpowiednią wiadomość odbieraną przez odbiornik – telefon/tablet. Do badań konsumpcji wykorzystano niezmodyfikowany przykład.

Doświadczenie PER wymaga bardziej wyrafinowanego podejścia, zdefiniowanego z wymaganiami metodologii w punkcie 3.3.1. Jako podstawę dla kolekcji trzech aplikacji wybrano przykład

⁸<https://www.zephyrproject.org/>

BLE_MeshLightingPRFNode. Przykład ten wprowadza bezpośrednio w pełny stos BLE Mesh umożliwiając jednocześnie wdrożenie pożądanych modeli Mesh jak i przypisanie węzłom konkretnej funkcji w sieci: Proxy, Relay, Friend [20].

Badając właściwości sieci, wprowadza się następujące nazewnictwo węzłów celem łatwiejszej identyfikacji:

- węzeł bliższy (ang. *proximal node*) – węzeł będący połączony bezpośrednio ze stacją akwizycji danych i kontroli przepływu eksperymentu. Dodatkowo, węzeł ten udostępnia tryb *Proxy*⁹
- węzeł środkowy (ang. *intermedial node*) – węzeł działający w trybie przekaźnika (terminologia Mesh: *Relay*). Węzeł ten nie uczestniczy bezpośrednio w badaniach tj. nie są z niego odczytywane jakiekolwiek dane.
- węzeł dalszy (ang. *distal node*) – węzeł zliczający ilość odebranych danych r , udostępniający jednocześnie usługę umożliwiającą odczyt tych wartości.

Posiadając trzy węzły, wymagane jest również dostarczenie trzech zestawów oprogramowania dla każdego z nich, w zależności od pełnionej funkcji.

Węzeł bliższy

Pierwszy węzeł, węzeł bliższy, odpowiada bezpośrednio za przeprowadzany eksperyment. Jego celem jest wysyłanie właściwych pakietów do węzła dalszego w określonych interwałach czasowych. Dodatkowo, służy jako bramka do odbioru liczby mówiącej o całkowitej ilości odebranych pakietów. Trzecim zadaniem tego modułu jest zresetowanie licznika z każdą nową iteracją badania PER.

```
1 // plik: appli_test.c
2 MOBLE_RESULT Test_ApplicationTest_Set05_GenericOnOff(MOBLE_ADDRESS src ,
3 MOBLE_ADDRESS dst)
4 {
5     MOBLE_RESULT result = MOBLE_RESULT_SUCCESS;
6     // [...]
7     meshTest.name = OP_NAME_SET05;
8     meshTest.counter = 0;
9     result = test_set05_generic_initialize(src, dst);
10    if (!result)
11    {
12        run_timer(OP_NAME_SET05, test_generic_subscription, src, dst,
13                  test_set05_generic);
14        result = MOBLE_RESULT_SUCCESS;
15    }
16    else
17    {
18        TRACE_I(TF_VENDOR_M, "%s Could not initialize the test due to error
19         code=%d \r\n", OP_NAME_SET05, result);
```

⁹z ang. pośrednik – tryb umożliwiający dostęp do sieci Mesh urządzeniom poprzez odpowiadającą usługę BLE. Umożliwia on m.in. dostęp do sieci i zarządzania nią z poziomu urządzenia nie posiadającego ani nie wspierającego bezpośrednio stos BLE Mesh

```

17     result = MOBLE_RESULT_FAIL;
18 } // [...]
19
20 return result;
21 }

```

Listing 1. Kod uruchamiający sekwencję badawczą PER

Listing 1 przedstawia kod uruchamiający sekwencję badawczą PER. Linie 4-6 inicjalizują niezbędne parametry m.in. identyfikujące aktualnie uruchomione doświadczenie. Przypisywana jest wartość zero do licznika inkrementującego ilość wysłanych pakietów. Jest to o tyle konieczne, iż pozwala na bezpośrednie porównanie ilości wysłanych komunikatów do ilości odebranych w innym węźle. Linia 8 odpowiada ze wyzerowaniem licznika po stronie węzła dalszego. Wysyłany jest odpowiedni komunikat wykorzystujący model Vendor'a z id APPLI_TEST_CMD. W treści wiadomości wysyłana jest odpowiednia wartość numeryczna interpretowana przez węzeł dalszy polecenie zresetowania węzła: APPLI_TEST_PACKET_ERROR_RATE_COUNTER=0x09U, jak pokazano na listingu 2. W przypadku, w którym zresetowanie licznika się powiedzie, dalsza część funkcji przystępuje do uruchomienia timer'a w linii 11.

```

1 // plik: appli_test.c
2 MOBLE_RESULT test_set05_generic_initialize(MOBLE_ADDRESS src ,
3                                             MOBLE_ADDRESS dst)
4 {
5     MOBLE_RESULT result = MOBLE_RESULT_SUCCESS;
6     AppliBuffer[0] = APPLI_TEST_PACKET_ERROR_RATE_COUNTER;
7
8     result = BLEMesh_SetRemotePublication(VENDORMODEL_STMICRO_ID1, src,
9                                         APPLI_TEST_CMD,
10                                        AppliBuffer, sizeof(AppliBuffer),
11                                         MOBLE_TRUE, MOBLE_TRUE);
12
13     if (result)
14     {
15         TRACE_I(TF_VENDOR_M, "%s Could not initialize the test due to error
16         code=%d \r\n", OP_NAME_SET05, result);
17     }
18     return result;
19 }

```

Listing 2. Kod resetujący wartość licznika w węźle dystalnym

Dodatkową uwagę zwrócić na nazewnictwo samych funkcji. Wskazuję one numer polecenia (Set05, Set06). Numer ten wykorzystywany jest przez *Mesh Serial Gateway* i interpretowany w sposób umożliwiający uruchomienie opisywanej funkcji. Nie jest to ścisłe wymaganie a raczej konwencja, kontrakt wprowadzony przez ST dla danego pliku. Niemniej jednak, pozwala ona w jasny i czytelny sposób zidentyfikować polecenie i odpowiadające polecenie Zbiór poleceń Hayes'a (znany

jako zbiór poleceń AT) (AT) (zbliżone do poleceń AT). Opis działania interfejsu PC-STM32WB55 znajduje się poniżej 3.1.3.

Timer

Timer odpowiada za regularne wykonywanie zdefiniowanej funkcji. Odpowiada on za główny przepływ badania interpretując zebrane parametry doświadczenia i je wykonując. Kod oparto o interfejs programistyczny dostarczony przez ST [28]¹⁰.

```
1 // plik: appli_test.c; Usunięto linie logujące przepływu TRACE_I
2 static void run_timer(char const *setName, MeshTest_t subscriptionType,
3   MOBLE_ADDRESS src, MOBLE_ADDRESS dst, void (*callback)(void) ) {
4   MOBLEUINT16 triggerInterval;
5   MOBLEUINT32 killAfterTimeout;
6
7   triggerInterval = timerTriggerInterval;
8   killAfterTimeout = Totaltest;
9
10  HW_TS_Create(subscriptionType, &(meshTest.timer_subscription_id),
11    hw_ts_Repeated, callback);
12  HW_TS_Create(test_kill_subscription, &(meshTest.
13    timer_kill_subscription_id), hw_ts_SingleShot, kill_subscription);
14
15 }
```

Listing 3. Funkcja aktywująca timer

Listing 3 przedstawia wdrożony kod uruchamiający timer. Najistotniejszym parametrem funkcji jest wskaźnik do funkcji umożliwiający uruchomienie kodu w odpowiedzi na zdarzenie (*callback*). Linie 6-7 przypisują wartości kolejno interwału, z którym periodycznie ma być uruchamiany *callback* oraz wartość po której upłynięciu timer zostanie wyłączone. Obie te zmienne są typu logicznego *tick*. Podawane w nich są wartości ticków procesora aniżeli milisekundy. Przed przypisaniem wartości należy przekonwertować milisekundy do ticków w zależności od konfiguracji zegara RTC mikrokontrolera. Zagadnienie to stanowiło problem na etapie tworzenia procedury badawczej. Ostatecznie zdecydowano się na wprowadzenie pojęcia kalibracji, w której to wyznaczano eksperymentalnie wartości ticków względem oczekiwanej długości okresu czasu. Logika takiego procesu opisana została w punkcie dotyczącym oprogramowania PC 3.1.4.

Linie 9-10, rejestrują wybrane funkcje (poprzez wskaźniki do funkcji) do wewnętrz silnika timera dostarczanego przez ST. Linijki 12-13 kolejno uruchamiają timer. Dzięki temu wprowadzono substytut wielowątkowości wykonując wiele procesów (tutaj: funkcji) pozornie jednocześnie bądź w ustalonej

¹⁰rozdział 4.5 *Timer server*

sekwencji. Rezultatem działania tego kodu jest periodyczne uruchamianie funkcji *callback* oraz jej zamknięcie po ustalonym czasie.

Opierając się na inżynierii wstępnej, znaleziono współczynnik, przeliczający wartości milisekund do ticków:

```

1 // plik: stm32wbxx_hal_conf.h
2 #define LSE_VALUE ((uint32_t)32768) /*!< Value of the External
   oscillator in Hz*/
3
4 // plik app_common.h
5 #define DIVR( x, y ) (((x)+((y)/2))/(y))
6
7 // plik app_conf.h
8 #define CFG_RTCCLK_DIV (16)
9 #define CFG_TS_TICK_VAL DIVR( (CFG_RTCCLK_DIV * 1000000), LSE_VALUE )
10
11 // plik app_ble.c - przykład użycia
12 #define INITIAL_ADV_TIMEOUT (60*1000*1000/CFG_TS_TICK_VAL) /*< 60s */

```

Listing 4. Ścieżka inżynierii wstępnej w celu znalezienia współczynnika umożliwiającego konwersję milisekund do ticków

Zgodnie z listingiem 4, ostateczna wartość współczynnika CFG_TS_TICK_VAL wynosi 488.78125. Wyliczone wartości ticków z ich eksperymentalnie wyznaczonymi odpowiednikami zaprezentowano w tabeli 2. Proces ten, zwany dalej kalibracją, oparty został o dodatkową procedurę zaprezentowaną na listingu 5.

```

1 // plik: appli_test.c
2 MOBLE_RESULT Test_ApplicationTest_Set06_CalibrateTimer(MOBLE_ADDRESS src
   ,MOBLE_ADDRESS dst) {
3   MOBLE_RESULT result = MOBLE_RESULT_SUCCESS;
4   if (TestCount != 0 && TestCount > timerTriggerInterval) {
5     meshTest.name = OP_NAME_SET06;
6     meshTest.counter = 0;
7     run_timer(OP_NAME_SET06, test_generic_subscription, src, dst,
8       test_set06_calibrate_timer);
9   }
10  TestNumber = 0; // kill command
11  command = CMD_TYPE_NONE;
12  return result;
}

```

Listing 5. Kalibracja mikrokontrolera – polecenie kalibracyjne

Przedstawione polecenie kalibracji wykorzystuje jedynie węzeł bliższy, który odpowiada za przeprowadzenie wykonanie przepływu doświadczenia. Usługa ta umożliwia wykonanie akcji kalibracyjnej `test_set06_calibrate_timer` polegającej na inkrementacji licznika. Bazując na ostatecznej

wartości licznika przy znanym określonym czasie wykonywania, można wyznaczyć arytmetycznie ostateczną wartość ticków do milisekund. Fakt ten wykorzystywany jest przez aplikację PC i tam zostaje bliżej opisany.

Węzeł środkowy

Kod węzła środkowego uległ najmniejszym zmianom względem pierwowzoru. Dokonano jedynie zmiany w konfiguracji przykładu uruchamiając tryb przekaźnikowy *Relay*, co ukazano na listingu 6.

```
1  /*
2  *   Different features supported by BLE-Mesh. Uncomment according to
3  *   application.
4  *       Low power feature enabled node do not support other features.
5  *       Do not define any other feature if Low Power feature is defined
6  */
7 #define ENABLE_RELAY_FEATURE
8 //##define ENABLE_PROXY_FEATURE
9 //##define ENABLE_FRIEND_FEATURE
10 //##define ENABLE_LOW_POWER_FEATURE
11 //##define ENABLE_PROVISIONER_FEATURE
12 //##define DYNAMIC_PROVISIONER
```

Listing 6. Konfiguracja węzła środkowego

Węzeł dalszy

Oprogramowanie węzła dalszego stworzono komplementarnie do węzła bliższego. Zapewnia ono dwie główne funkcjonalności: zliczanie odebranych pakietów na poziomie Generic OnOff oraz możliwość odczytu i resetu tegoż licznika. Podstawowy interfejs do powyższych funkcjonalności zdefiniowano w pliku *appli_generic_counter.h* - listing 7 wraz z opisem przepływu wiadomości z perspektywy plików i fizycznych węzłów.

Linia 19 definiuje interfejs inicjalizacyjny doświadczenia PER. Zgodnie z wcześniejszym opisem węzła bliższego, ta funkcja aktywowana jest w odpowiedzi na żądanie wyzerowania licznika. Linia 20 reprezentuje funkcję, która pobiera wartość licznika ze struktury zdefinowanej w liniach 15-17, którą zrealizowano już wewnętrz pliku odpowiedzialnego za obsługę modelu *Generic OnOff*: *appli_generic_client.h*.

```
1 #define ENABLE_LED_BLINKING
2
3 /*
4 * The purpose of this header is to provide an API to cover Packet Error
5 * Rate experiment.
6 * The overall experiment is designed as follows:
7 *   * appli_test (proximal node): Run periodically Generic OnOff client
8 *     model (SET-05)
```

```

7 *   * appli_test: Count the number of sent Generic OnOff requests
8 *   * appli_generic_counter (remote node): Count the number of received
9 *   messages
10 *  * appli_vendor (remote node): Publish the number of received messages
11 *  * appli_test: Collect the results. Make sure the nodes are close
12 *  enough to get the results
13 *
14 *
15 typedef struct {
16 MOBLEUINT32 counter;
17 } GenericOnOffCounter_t;
18
19 void generic_onoff_counter_initialize();
20 MOBLEUINT32 generic_onoff_counter();

```

Listing 7. Interfejs eksperymentu PER dla węzła dalszego

Listing 8 prezentuje wycinek kodu zliczającego ilość odebranych komunikatów. `Appli_Generic_OnOff_Set` jest funkcją odpowiadającą na zdarzenie zarejestrowaną w stosie technologicznym BLE Mesh producenta. Uruchamiana jest ona za każdym razem gdy otrzymywane jest polecenie zgodne z modelem zdefiniowanym przez Bluetooth SIG. Najistotniejszą linią jest linia 20, która bezpośrednio odpowiada za logowanie i inkrementację licznika. Istotna jest również instrukcja warunkowa otaczająca opisywane wywołanie funkcji. Uniemożliwia inkrementację licznika na skutek odbioru wiadomości rozgłoszeniowej. Od użytkownika oczekuje się podanie rzeczywistego adresu węzła zarejestrowanego wewnętrz sieci Mesh.

Linie 24-33 wykonują polecenie włączania i wyłączania Light Emitting Diode (dioda emitująca światło) (LED). Funkcjonalność ta otoczona jest dyrektywami sprawdzającymi istnienie zdefinowanej nazwy. Służy to celu włączeniu i wyłączeniu opcji aktywacji diody w celu ograniczenia zużycia energii. Fakt ten wykorzystywany jest w badaniu zużycia energii 3.2.

```

1 /**
2 * @brief Appli_Generic_OnOff_Set: This function is callback for
3 * Application
4 *           when Generic OnOff message is received
5 * @param pGeneric_OnOffParam: Pointer to the parameters received for
6 * message
7 * @param OptionalValid: Flag to inform about the validity of optional
8 * parameters
9 * @param dstPeer: destination send by peer for this node. It can be a
10 *                  unicast or group address
11 * @param elementIndex: index of the element received from peer for this
12 * node which
13 *                  is elementNumber -1
14 * @retval MOBLE_RESULT

```

```
11 */  
12 MOBLE_RESULT Appli_Generic_OnOff_Set(Generic_OnOffStatus_t*  
13     pGeneric_OnOffParam,  
14     MOBLEUINT8 OptionalValid,  
15     MOBLEUINT16 dstPeer,  
16     MOBLEUINT8 elementIndex) {  
17     // [...]  
18     if (AppliOnOffSet[elementIndex].Present_OnOffValue  
19         == AppliOnOffSet[elementIndex].TargetValue)  
20     {  
21         if (dstPeer != 0xc000) {  
22             increment_generic_onoff_counter_packet_error_rate_experiment(  
23                 AppliOnOffSet[elementIndex].Present_OnOffValue, dstPeer);  
24         }  
25 #ifdef ENABLE_LED_BLINKING  
26         if (AppliOnOffSet[elementIndex].Present_OnOffValue > 0)  
27         {  
28             BSP_LED_On(LED_BLUE);  
29         }  
30         else  
31         {  
32             BSP_LED_Off(LED_BLUE);  
33         }  
34 #endif /* ENABLE_LED_BLINKING */  
35     }  
36     // [...]
```

Listing 8. Inkrementacja licznika realizowana jest poprzez dostosowanie funkcji odpowiadającej na zdarzenie

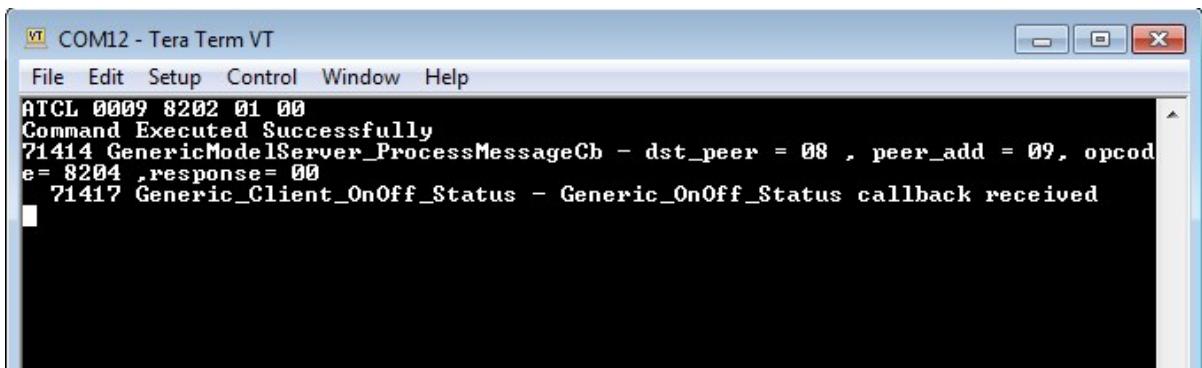
Mesh Serial Gateway

Komunikację na linii węzła bliższego zrealizowano z pomocą autorskiego rozwiązania ST: Mesh Serial Gateway [20]. Rozwiązanie to wykorzystuje komunikację szeregową Universal asynchronous receiver-transmitter (UART) do wysyłania i odbierania komunikatów sterujących. Zgodnie z dokumentacją, polecenie takie może przyjąć postać jak przedstawioną na rysunku 21, wysyłając do węzła 0009 polecenie aktywujące funkcję `Appli_Generic_OnOff_Set` i ustawiając wartość na 1. Innymi słowy, dioda zaświeci się, o ile wcześniej była nieaktywna.

W analogiczny sposób steruje się poleceniami niezbędnymi do eksperymentu, jak to ukazano na listingu 9.

```
1 ATAP SET-nn iiiiTTTTTTT src dst  
2  
3 przykład:  
4 ATAP SET-05 0a6400009c40 03 05
```

Listing 9. Przykładowe polecenie Mesh Serial Gateway



Rysunek 21. Wykorzystanie interfejsu szeregowego do wysyłania poleceń. Źródło: [20]

ATAP - polecenie testowe

SET-nn - polecenie typu SET dla funkcji numeru nn, np. SET-05

iiii - interwał, częstotliwość zapytań wyrażony w tickach; Wartość 16-bitowa

TTTTTTTT - czas po którym należy przerwać badanie wybrane w tickach; Wartość 32-bitowa

src - adres źródłowego węzła (najczęściej węzła bliższego)

dst - adres węzła docelowego (najczęściej węzła dalszego)

3.1.4 Oprogramowanie PC

Głównym zadaniem oprogramowania PC jest udostępnienie prostego interfejsu użytkownika umożliwiającego swobodne przeprowadzenie badań, w szczególności PER. Stworzenie takiego programu umożliwia nie tylko szybsze wykonanie doświadczeń, co również redukuje błędy możliwe do popełnienia podczas ręcznego wprowadzania komend AT.

Aplikacja realizuje trzy główne cele:

- Ustanawia połączenie z mikrokontrolerem przy użyciu portu szeregowego
- Umożliwia kalibrację podstawy mikrokontrolera czasu względem ticknięć
- Umożliwia wykonanie poleceń uruchamiających cykl badań PER i odbiór wartości zliczeń z węzła dystalnego

Oprogramowanie wykonano z wykorzystaniem framework'a Qt¹¹. Umożliwia on tworzenie kompletnych, wieloplatformowych aplikacji graficznych. Qt nie ogranicza się jedynie do zapewnienia bibliotek graficznych. Framework zapewnia dostęp z poziomu API do urządzeń i peryferiów takich jak drukarki, gamepady, gniazda TCP/IP, stoseu Bluetooth (w tym BLE) jak i również, co jest istotne z punktu widzenia niniejszej pracy, portu szeregowego.

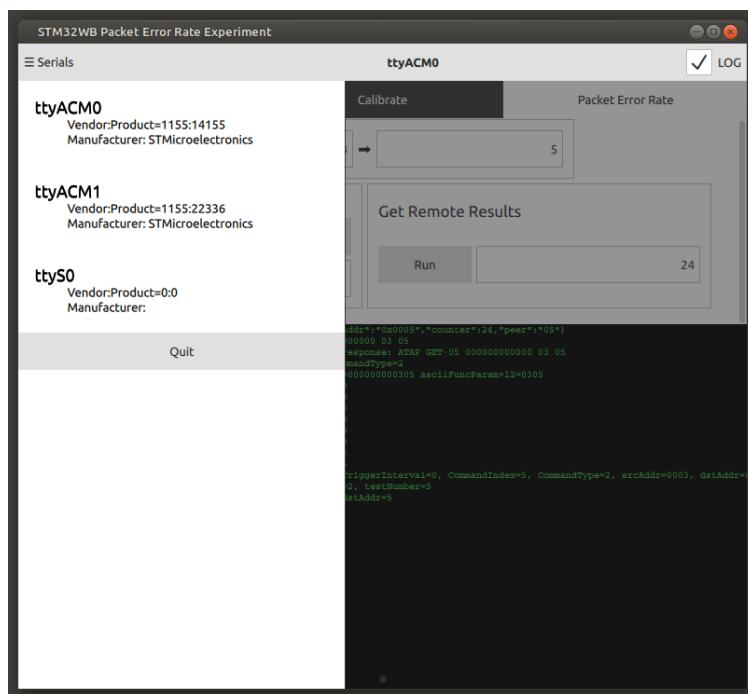
Qt zapewnia kilka możliwości tworzenia GUI wykorzystując tylko kod w języku C++ lub rozdzielając część widzialną od części biznesowej z pomocą generatora interfejsu Qt Designer

¹¹<https://www.qt.io/>

produkując odpowiedni plik Extensible Markup Language (XML) bądź *QML*. *QML* jest językiem deklaratywnym opisujący pożądany efekt graficzny. Do silnika tego narzędzia należy interpretacja i wygenerowanie właściwego efektu. Język ten rozszerza możliwości klasycznego wydania frameworka umożliwiając tworzenie dynamicznych, nowoczesnych interfejsów użytkownika działających niezależnie od platformy docelowej.

Port szeregowy

Aplikację oparto o połączenie części biznesowej tworzonej w C++ oraz interfejsie użytkownika stworzonego w QML. Pierwszym, niezbędnym elementem jest zapewnienie stabilnej komunikacji z użyciem portu szeregowego¹². Aplikacja wpierw wyszukuje listę portów w danym systemie operacyjnym i prezentuje ją w przystępny dla użytkownika sposób – Rysunek 22.



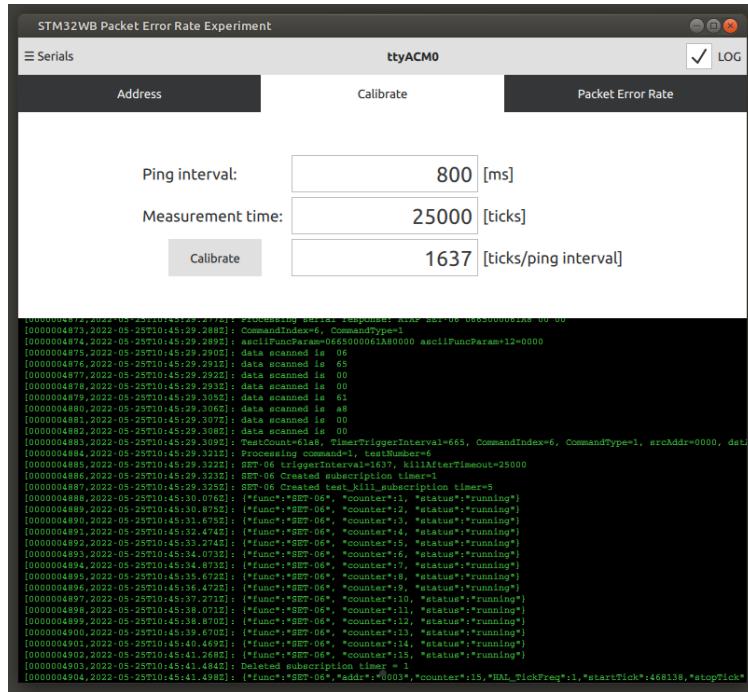
Rysunek 22. Selektor portów szeregowych w aplikacji służącej eksperymentowi PER.

Kalibracja

Kolejnym elementem, wymaganym przez chęć przeprowadzenia badania, jest zapewnienie właściwej podstawy czasu. Wykorzystywany timer (jak został opisany w punkcie 3.1.3) wymaga sprecyzowania wartości podanej w tick'ach. Zauważając potencjalny problem z matematycznym wyznaczeniem odpowiedniego mapowania z milisekund do ticków zdecydowano się na stworzenie empirycznego sposobu na wyznaczenie tej wartości.

¹²<https://doc.qt.io/qt-5/qserialport.html>

Kalibracja urządzenia polega na odnalezieniu wartości ticków przypadającą na pożądany okres wyrażony w milisekundach. Usługa obsługująca taką operację przedstawiona została na listing'u 5. Zadaniem aplikacji PC jest jej wielokrotne wykorzystanie celem wyznaczenia ostatecznej wartości.



Rysunek 23. Selektor portów szeregowych w aplikacji służącej eksperymentowi PER.

Algorytm działania jest następujący. Dla oczekiwanej wartości wyrażonej w milisekundach, wykonuje się pierwsze polecenie kalibracyjne zakładając fałszywie, iż milisekundy równe są ilości ticków. Aplikacja, nasłuchuje przychodzące komunikaty w formacie JSON poszukując słowa kluczowego „status”. Do każdego zliczonego komunikatu, aplikacja przechowuje dodatkowo znacznik czasu. Takie wiadomości akumulowane są w pamięci do momentu otrzymania innego słowa kluczowego: „not_running”. W tym momencie, aplikacja wylicza różnicę między przyległymi znacznikami czasowymi (efektywnie je różniczkując w liniach 4-9 listingu 10). Następnie, wyliczana jest średnia różnica pomiędzy poszczególnymi okresami, w których otrzymano odpowiedź z mikrokontrolera – linia 15.

```

1 // plik calibratecommand.cpp
2 std::pair<uint16_t, double> CalibrateCommand::computeNewInterval() {
3     QVector<qint64> timestampDiffs;
4     std::transform(std::cbegin(timestamps), std::cend(timestamps),
5                  std::back_inserter(timestampDiffs),
6                  [] (const QDateTime &timestamp) -> long { return timestamp.
7                      toMSecsSinceEpoch(); });
8
9     std::adjacent_difference(std::begin(timestampDiffs), std::end(
10        timestampDiffs), std::begin(timestampDiffs));

```

```
9    timestampDiffs.removeFirst();
10
11    qint64 sum = 0;
12    for (qint64 elem : qAsConst(timestampDiffs)) {
13        sum += elem;
14    }
15    double meanTimestampDiff = sum / static_cast<double>(timestampDiffs.
16        size());
16    double accuracy = abs((meanTimestampDiff - expectedIntervalMs) /
17        static_cast<double>(expectedIntervalMs));
17    double interval = currentIntervalTicks + currentIntervalTicks *
18        accuracy; // this isn't stable if we overshoot the correct value...
19
20    return std::make_pair(interval, abs(accuracy));
}
```

Listing 10. Algorytm wyznaczania kolejnej wartości dla funkcjonalności kalibracji

Kolejnym krokiem jest wyliczenie procentowej różnicy pomiędzy obecnie wyliczoną wartością a oczekiwanaą wartością wyrażoną w milisekundach – linia 16. Ostatecznie, wylicza się nową, estymowaną wartość kolejnej iteracji zgodnie z zależnością liniową podaną w linii 17.

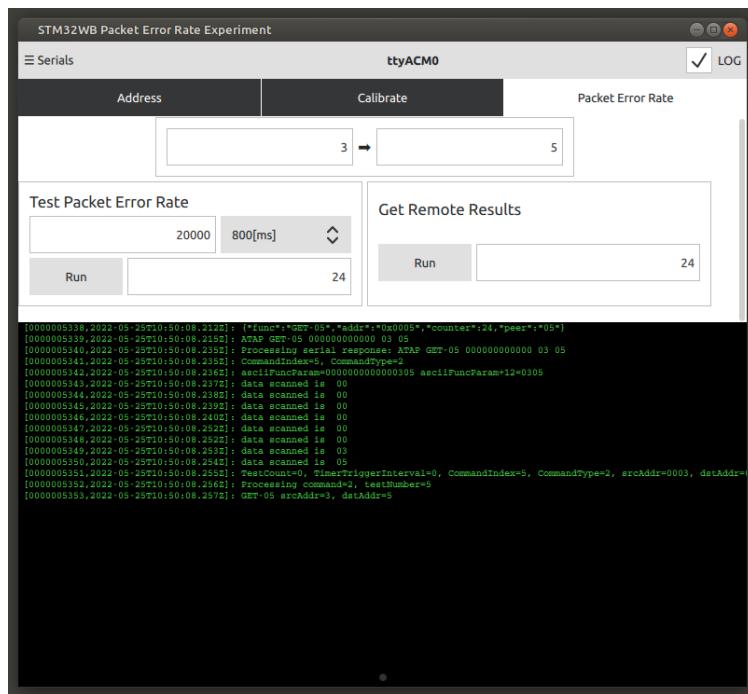
Kalibracja zatrzyma się automatycznie, gdy osiągnięta zostanie zbieżność na poziomie poniżej jednego procenta różnicy. Algorytm jest wrażliwy na „przestrzelanie” kalibrowanej wartości. Jego jakość jest jednak na tyle wystarczająca, iż ostatecznie umożliwił wyznaczenie wartości, które są zbieżne z obliczeniami teoretycznymi, co pokazuje tabela 2. Możliwym usprawnieniem byłoby zastosowanie metody wyznaczania parametrów kalibracyjnych w oparciu o analogię do metody bisekcji.

Ping [ms]	Skalibrowane [tick]	Obliczone [tick]	Różnica [%]
100	204	204.59	0.29
500	1022	1022.95	0.09
800	1637	1636.72	0.02
1300	2661	2659.68	0.05
2100	4299	4296.40	0.06

Tablica 2. Zebrane empirycznie współczynniki kalibracyjne w porównaniu z wartościami obliczonymi

Badanie PER i odczyt wyników

Główym zadaniem aplikacji jest oczywiście przeprowadzenie doświadczenia PER. W tym celu stworzono wygodny interfejs do m.in. funkcji opisywanej w punkcie 3.1.3. Aplikacja wymaga od użytkownika podania adresu (od lewej): węzła bliższego oraz węzła dalszego. W rzędzie poniżej znajdują kontenery z dwoma funkcjami w danej zakładce: właściwy test PER oraz odbiór wartości z węzła zewnętrznego – Rysunek 24.



Rysunek 24. Właściwy interfejs użytkownika do przeprowadzenia doświadczenia PER

Eksperymentator, w celu przeprowadzenia doświadczenia, zobowiązany jest wyznaczyć okres, po którym zakończy się badanie w milisekundach. Czas badania powinien być dostatecznie długi, by błędy inicjalizacji, czy konwersji milisekund do ticków nie miały znacznego wpływu na otrzymany rezultat. Drugi parametr to interwał odpytywania. Aplikacja przechowuje wyniki kalibracyjne¹³, by je wykorzystać właśnie w tym miejscu. Przyśpiesza to procedurę badawczą, a przede wszystkim zmniejsza ryzyko błędu. Ostatnim elementem jest przycisk „Run”, który uruchamia procedurę badawczą. Ekran aplikacji zostanie zablokowany na czas badania.

Analogicznie należy postąpić w przypadku odczytywania licznika z węzła dalszego. Użytkownik zobowiązany jest kliknąć przycisk „Run” w kontenerze „Get Remote Results”. Po krótkiej chwili, odpowiadające pole powinno zostać zaktualizowane o oczekiwana wartość.

3.2 Badanie zużycia energii

Celem niniejszego podrozdziału jest omówienie empirycznej weryfikacji zużycia energii przez wybrany zestaw uruchomieniowy BLE.

Zaprezentowanie zostanie metodologia pomiaru oraz sposób połączenia układu pomiarowego. Omówione zostaną parametry próbkowania i długość trwania pojedynczej sesji badawczej. Ostatecz-

¹³wykorzystując możliwości *QSettings* dane przechowywane są trwale na dysku. W zależności od systemu albo w rejestrze (Windows), albo w pliku *~/.config/Warsaw University of Technology/STM32WB Packet Error Rate Experiment.conf* (*nix)

nie, przedstawia się wzór i sposób przeprowadzonych obliczeń, które docelowo zapewniają oczekiwane wyniki.

Ostatnim, aczkolwiek najistotniejszym elementem poruszonym przez podrozdział, jest prezentacja wyników. Pogrupowane są one na dwie kategorie: BLE Heart Rate i BLE Mesh. Każda z nich przedstawia zebrane empirycznie dane, wskazując na właściwe tryby działania i całkowite zużycie energii.

3.2.1 Metodologia badania

Podstawą badania zużycia energii jest pomiar wartości pobieranego prądu przez zestaw uruchomieniowy P-NUCLEO-WB55 w czasie. Posiadając dane wymienione dane, na podstawie oczywistych zależności fizycznych, wyznacza się parametry jak całkowita wykorzystana energia podczas badania, przedstawiona w przystępnej formie parametru mocy.

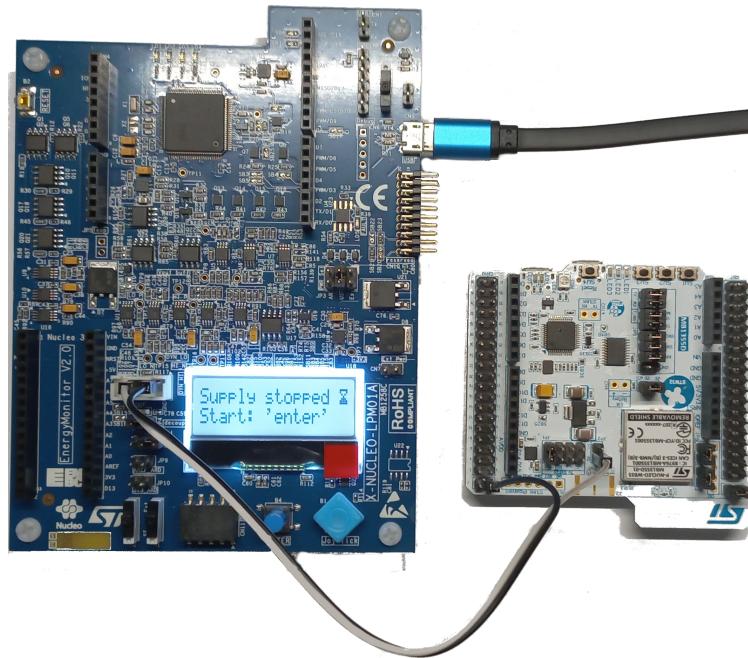
Pomiary wartości chwilowego poboru prądu oparto o moduł opisany w punkcie 3.1.1. Źródłem energii dla płytki pomiarowej jest komputer klasy PC, a precyzyjniej udostępniany port USB. Wykorzystując również ten protokół następuje transmisja danych poprzez komunikację szeregową UART, umożliwiając akwizycję danych.

Połączenie z zestawem uruchomieniowym oparto o konektor CN14 udostępniający piny: PIN1 - masa GND, PIN3 - VCC 3.3V [36]. Wyprowadzone przewody połączono następnie z P-NUCLEO-WB55 poprzez konektor JP2, wcześniej pozbawiony zawleczki. Jest to metoda rekomendowana dla pomiaru prądu opisana w dokumentacji zestawu [26]¹⁴. Porty kompatybilne z Arduino/ST Morpho mogłyby stanowić równorzędny sposóbłączenia zestawu uruchomieniowego do obwodu płytki pomiarowej. Autorska analiza dokumentacji sugeruje jednak, iż opcja ta nie jest wspierana. Wymagane byłoby lutowanie dodatkowego połączenia w (punkcie SB27), by móc wykorzystać zasilanie napięciem 3.3V.

Celem akwizycji danych, wybrano dostarczane przez producenta oprogramowanie *STM32CubeMonitor-Power*. Przykładowy zrzut ekranu zaprezentowany jest na rysunku 26. Pojedyncza sesja pomiarowa trwająca 100 sekund zapewnia niezbędne dane. Węzły sieci były oddalone od siebie nie dalej niż na odległość 1,5 metra. Są one następnie odpowiednio przetwarzane poprzez odcięcie wartości zebranych w ostatnich sekundach. Związane to było ze sposobem działania mikrokontrolera, który po 60 sekundach domyślnie przechodzi w stan zwiększonej oszczędności energii (Low Power Advertising). Parametr ten można zmienić wg własnych upodobań, aczkolwiek zdecydowano się na domyślne nastawy, tak by umożliwić możliwie proste, ponowne przeprowadzenie doświadczenia wykorzystując przykład BLE HRT dostarczony przez ST. Stąd, by rozróżnić różne tryby pracy BLE, zdecydowano o odcięciu połowy okresu pomiarowego, zapewniając jednolite wartości względem trybów zużycia energii w 50 sekundowym oknie.

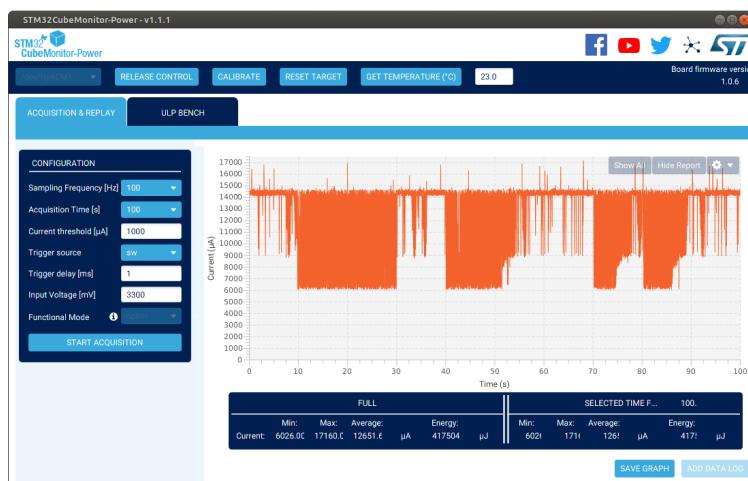
Zebrane dane w postaci wartości chwilowych pobieranego przez mikrokontroler prądu względem czasu przetworzono z użyciem metod numerycznych. W celu wyliczenia łącznej wykorzystanej

¹⁴Rozdział 7.12 Current measurement



Rysunek 25. Podłączony zestaw pomiarowy

energii, a tym samym mocy, stosuje się zależność 1. Uwzględniając fakt działania w domenie dyskretnej, wykorzystano kompozytową metodę całkowania Simpsona [17]. Podstawą dla całkowania są wartości zebrane w równoodległych odstępach. Dokonując akwizycji danych, dobrano częstotliwość próbkowania jako 100Hz . Każda kolejna wartość charakteryzuje się więc 10 milisekundową różnicą w podstawie czasu. Uwzględniając ten fakt, wyliczenie wartości wykorzystanej energii jak i mocy staje się trywialne.



Rysunek 26. Przykładowa sesja pomiarowa dla BLE Mesh - sieć w trybie nasłuchującym

3.2.2 BLE- Usługa Heart Rate

Pierwszą usługą BLE badaną pod kątem zużycia energii jest Heart Rate¹⁵. Wybór ten podyktowany jest chęcią przyszłego zestawienia wartości z rozwiązaniami innych producentów. Usługa zdefiniowana przez Bluetooth SIG stanowi więc wspólny międzyplatformowy język. Porównanie konkurencyjnych rozwiązań względem ST stanowi potencjalny kierunek dalszych rozważań.

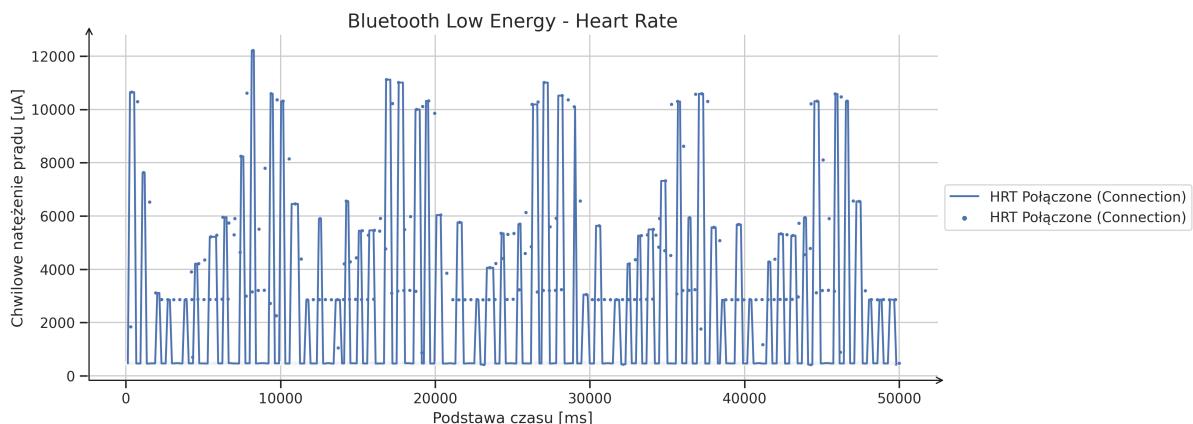
Badania przeprowadzono dla trzech różnych trybów działania aplikacji BLE HRT:

- Usługa połączona (Connected)
- Usługa w trybie szybkiego ogłoszania (Fast Advertising)
- Usługa w trybie niskomocowego ogłoszania (Low Power Advertising)

Emitowana moc ustalona została na $-0.15dBm$. Jest to wartość wspólna dla każdego pomiaru dotyczącego BLE HRT.

Rysunek 27 przedstawia charakterystykę poboru prądu w czasie po ustanowieniu połączenia urządzeniem klienckim. Mikrokontroler publikuje losowe dane z częstotliwością $1Hz$. W konsekwencji radio zestawu uruchomieniowego uruchamiane jest co najmniej raz na sekundę.

Analizując wykres dostrzega się regularne, 10-sekundowe interwały w których pobierany prąd dąży do swojego maksimum wynoszącego ok. $12mA$. Jest to prawdopodobnie związane z ustalonymi parametrami transmisji danych w mikrokontrolerze. Niniejsza praca nie podejmuje jednak próby wyjaśnienia rzeczywistej przyczyny tego zjawiska. Minimalny zarejestrowany pobór prądu ustalony jest na ok. $422\mu A$, czyli o dwa rzędy mniejsze wartości szczytowej.

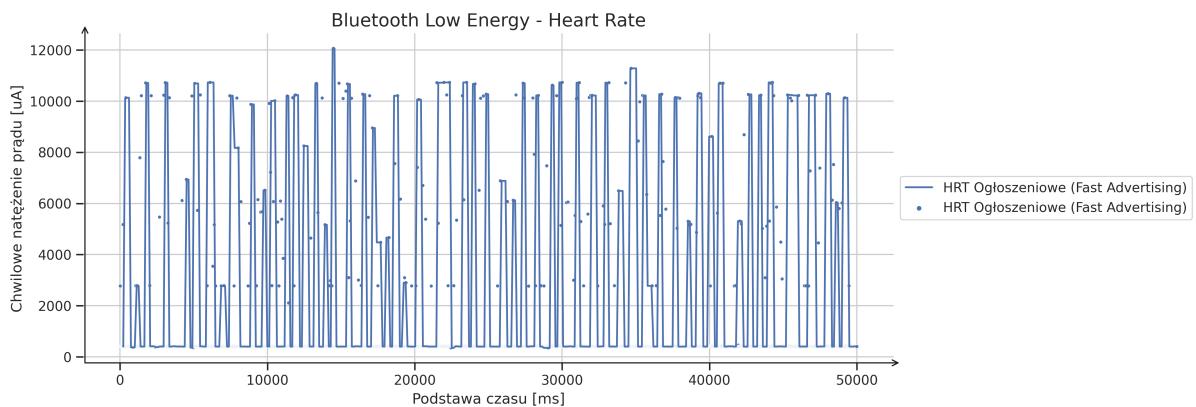


Rysunek 27. Charakterystyka czasowa poboru prądu dla BLE i usługi Heart Rate - Usługa Połączona

Rysunek 28 przedstawia charakterystykę poboru prądu po czasie dla szybkiego trybu ogłoszeniowego. Zgodnie z kodem źródłowym, interwały w których rozgławszane są komunikaty powinny się zawierać w częstotliwości od 80 do 100ms. Wywołuje to konieczność uruchamiania radia co najmniej 10 razy na sekundę. Ma to swoje odzwierciedlenie na wykresie. Charakterystyka jest bardziej burzliwa w porównaniu z wykresem 27.

¹⁵z ang. rytm serca

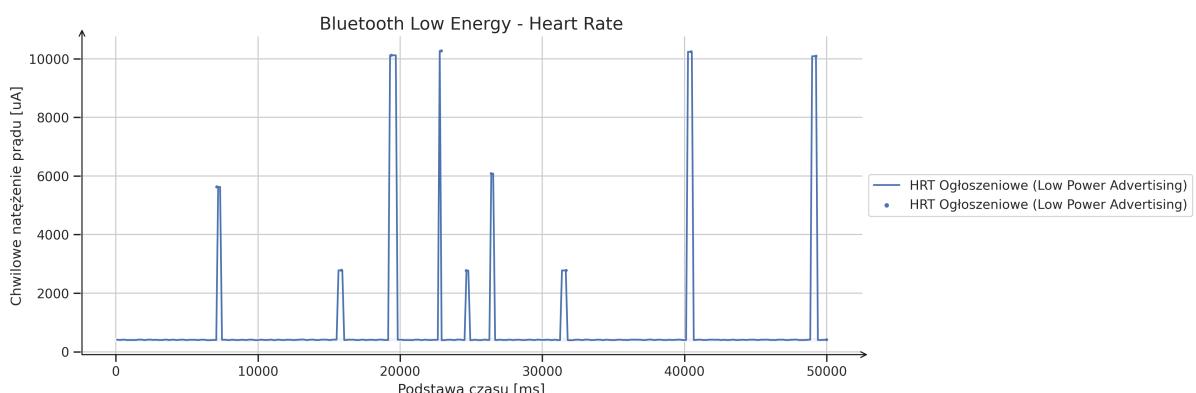
Zauważa się większą częstotliwość szczytów osiągających wartości pomiędzy $10 - 12mA$. Jest to w oczywisty sposób powiązanie ze sposobem działania trybu *Fast Advertising*. Minimalna wartość zarejestrowana wynosi jednak $350\mu A$, co ponownie stanowi różnicę dwóch rzędów wielkości.



Rysunek 28. Charakterystyka czasowa poboru prądu dla BLE i usługi Heart Rate - Tryb szybkiego ogłoszania

Ostatnim wykresem prezentującym charakterystykę poboru prądu w czasie jest rysunek 29. Przedstawia on tryb działania niskomocowego ogłoszania. Różni się on względem trybu *Fast Advertising* częstością z jaką wysyłane są pakiety ogłoszeniowe. W tym przypadku, oprogramowanie wysyła wiadomości z interwałem znajdującym się w przedziale od jednej do dwóch i pół sekundy (parametry: *CFG_LP_CONN_ADV_INTERVAL_MIN* i *CFG_LP_CONN_ADV_INTERVAL_MAX*). Oba wymienione tryby są autorskim rozwiązaniami ST. Przegląd literatury nie wskazuje, jakoby takie zachowanie wymagane jest przez sam standard.

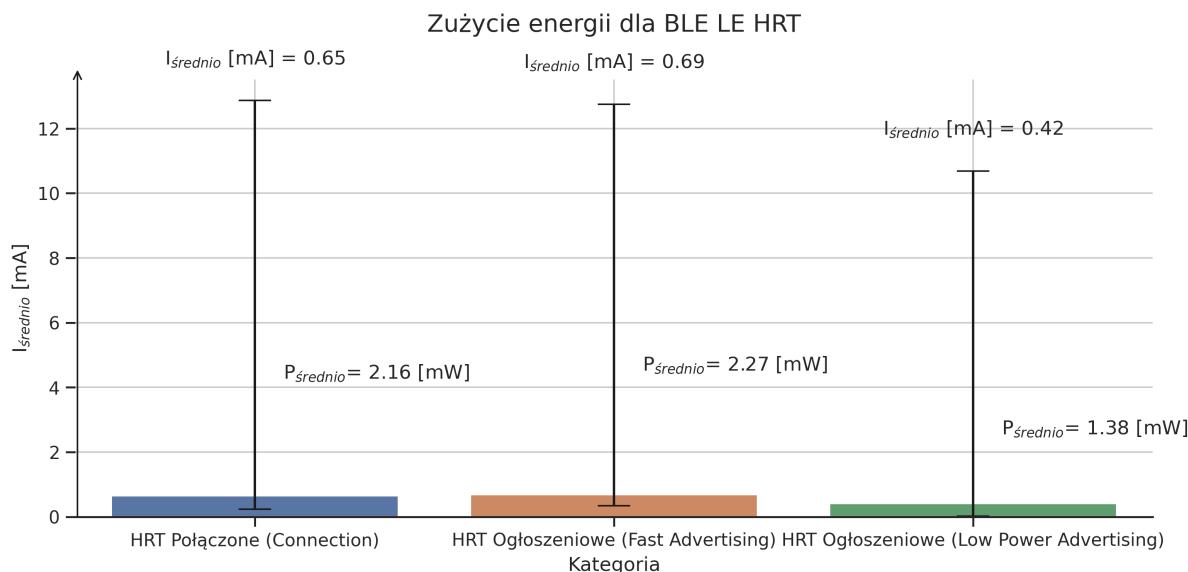
Rysunek 29 wskazuje na podobne zależności. Obserwuje się periodyczne, aczkolwiek stosunkowo rzadkie szczyty w poborze energii rozumianej jako przepływ prądu elektrycznego. Wpisują się jednak w wyżej wymienione parametry transmisji. Brak całkowitego dopasowania względem oczekiwanych parametrów może być spowodowany niedostateczną częstotliwością próbkowania. Z pewnością jest to zagadnienie, które warto przeanalizować w kolejnych iteracjach prób doświadczalnych.



Rysunek 29. Charakterystyka czasowa poboru prądu dla BLE i usługi Heart Rate - Tryb niskomocowego ogłoszania

Ostatecznie, zestawia się wszystkie zebrane jak dotąd wartości pomiarowe na rysunku 30. Wykres słupkowy przedstawia średni pobór prądu wyrażony miliamperach. Porównując, zauważa się znaczącą, blisko dwukrotną różnicę, co do zużytej mocy, pomiędzy trybem ogłoszeniowym niskomocowym a pozostałymi kategoriami.

Porównując wymienione tryby ogłoszeniowe, należy zwrócić uwagę na konfigurację częstotliwości, z którą wysyłane są pakiety danych. Częstotliwość, z którą aktywowane jest radio, może być nawet 30 krotnie wyższa dla przypadku szybkiego ogłoszania. Uwidocznia się również różnica w maksymalnie osiągniętym szczycie poboru mocy, wynosząca ok. $2mA$.



Rysunek 30. Zestawienie zużycia prądu dla usługi Heart Rate w zależności od trybu działania

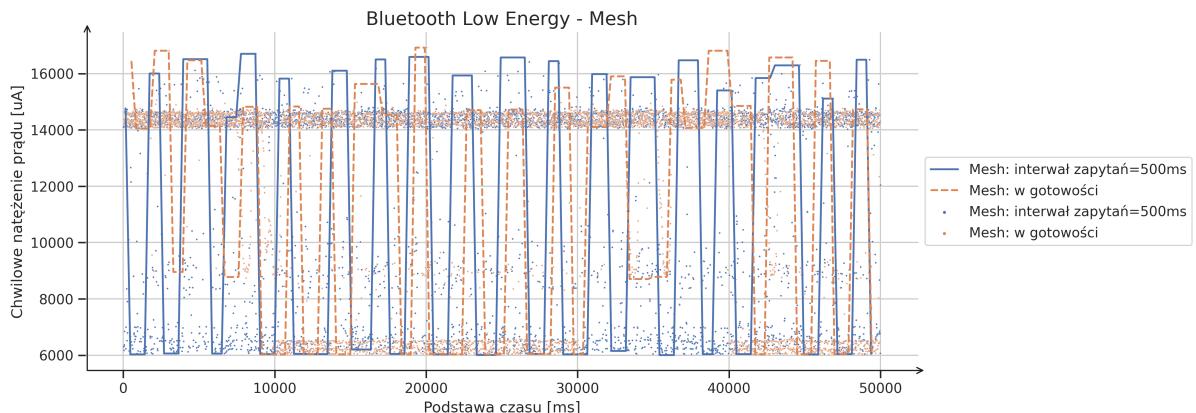
Zużycie energii w zestawieniu trybu ogłoszeniowego szybkiego i ustanowionego już połączenia jest zbliżone. Według zadanych nastawień, *Fast Advertising* aktywuje radio około 10 razy częściej aniżeli po ustanowieniu łączności z aplikacją kliencką. Natomiast, przypadek połączonej już usługi przenosi dodatkowe informacje takie jak puls czy estymowana spalona energia w wyników procesów metabolicznych. Być może, zużycie energii również zależy od ilości przenoszonych danych, a co za tym idzie, wydłużenia czasu działania radia niezbędnego do wysłania takiej wiadomości. Jest to pytanie otwarte.

3.2.3 BT Mesh - Model Generic OnOff

Zużycie energii dla BLE Mesh przeprowadzono dla dwóch wariantów:

- sieć w trybie gotowości (*standby*)
- sieć w której wysyłany jest komunikat z modelu *Generic OnOff*

Poprzez sieć w trybie gotowości należy rozumieć, iż wszystkie zarejestrowane¹⁶ węzły są uruchomione. Zgodnie ze standardem BLE Mesh, każdy z takich węzłów musi mieć zaimplementowane dwa modele: *Configuration Server* i *Health Server* [37]. Odpowiadają one za niezbędną konfigurację jak i również nadzorowanie stanu *zdrowia* sieci, poprzez wysyłanie komunikatów *keep-alive*¹⁷.



Rysunek 31. Charakterystyka czasowa poboru prądu dla BLE Mesh i modelu Generic OnOff

Rozpatrywany przypadek *Generic OnOff* wykorzystuje tenże model do wysyłania komunikatów wewnętrz sieci. Jeden z węzłów wysyła pakiet danych a drugi, odbiorczy, przetwarza wykonując operację zdefiniowaną przez model akcję. W przypadku poniższych badań operacją tą jest kolejne zliczanie otrzymanych pakietów. W tym celu wykorzystano oprogramowanie opracowane na potrzeby kolejnego opisywanego eksperymentu *Packet Error Rate*. Przez cały okres akwizycji wysyłano komunikaty z częstotliwością 2Hz. Zasadne jest wspomnieć, iż na czas trwania doświadczenia wszelkie diody emitujące światło zostały wyłączone celem eliminacji błędów systemowych wpływających na ostateczne zużycie energii przez urządzenie.

Rysunek 31 przedstawia charakterystyki prądu dla wymienionych wcześniej badanych wariantów. Wartości są aproksymowane celem wskazania ogólnej zmiennej tendencji poboru prądu elektrycznego. Jej zmienna reprezentuje wykres punktowy, zlewający się w trzy rozpoznawalne klastry oscylujące wokół wartości 14mA, 9mA i 6mA. Rzeczywista charakterystyka wygenerowana przez aplikację *STM32CubeMonitor-Power* zaprezentowana jest na rysunku 32.

Rysunek 32 wygenerowany przez aplikację produkcji ST odpowiada chmurze punktów z rysunku 31 dla wariantu regularnych zapytań co 500ms. Analogiczny wykres dla trybu nasłuchującego wygenerowany przez *STM32CubeMonitor-Power* reprezentowany jest na rysunku 26. Swoiste zgaszczenie punktów pomiarowych i częste zmiany amplitudy mogą wskazywać na złożoność protokołu Mesh, który wymaga częstych operacji w tym radiowych. Operacje te bezpośrednio oddziałują na zużycie energii, co w oczywisty sposób widoczne jest na wykresach.

Parametry zużycia energii przedstawia rysunek 33. Porównywane są dwa zdefiniowane warianty działania aplikacji. Uwidacznia się niewielka różnica konsumowanej mocy. Węzeł odbierający

¹⁶proces w anglojęzycznej literaturze i dokumentacji znany jest jako *provisioning*

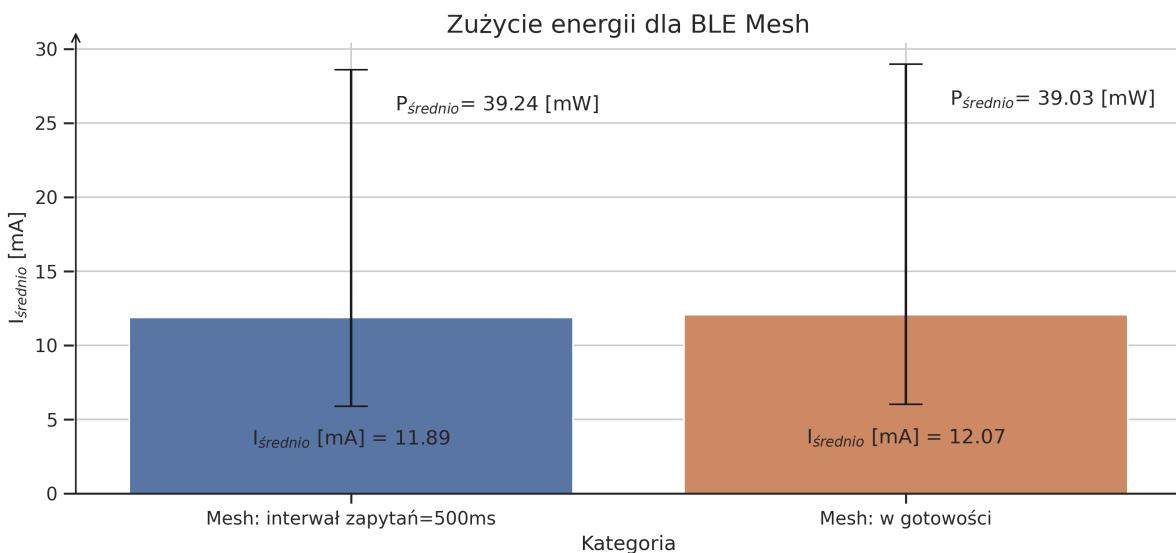
¹⁷standard definiuje takie komunikaty jako *heartbeats*[37][11]

Rozdział 3. Część doświadczalna



Rysunek 32. Rzeczywista charakterystyka poboru energii dla BLE Mesh działającego w trybie gotowości

komunikaty *Generic OnOff* wymaga średnio 39,2mW mocy do funkcjonowania. Konkurencyjny wariant potrzebuje jej nieznacznie mniej. Mesh będący w gotowości potrzebuje średnio 39.0mW mocy. Minimalne i maksymalne natężenia chwilowo pobieranego prądu są porównywalne.

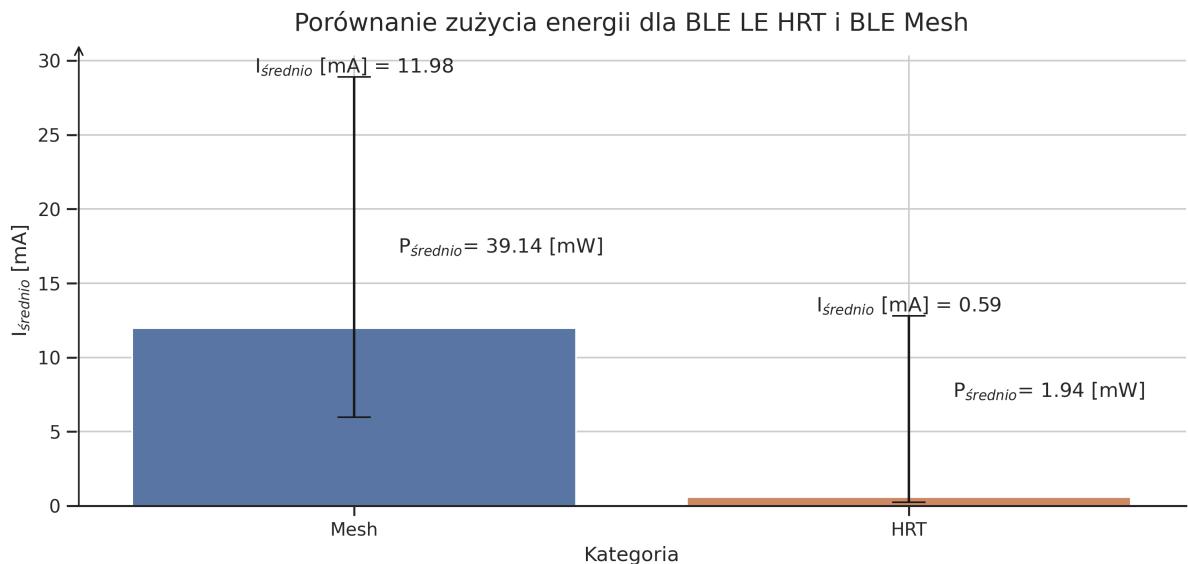


Rysunek 33. Zestawienie zużycia prądu dla BLE Mesh w zależności od trybu działania

Interesującym aspektem jest porównanie BLE Mesh ze standardową usługą BLE pod względem energetycznym. Rysunek 34 wskazuje na różnice pomiędzy dwoma standardami transmisji. Porównując średnie zużycie energii pomiędzy BLE HRT a Mesh, zauważa się ogromne dysproporcje. Mesh wymaga średnio 39.1mW mocy, by funkcjonować. Jest to rząd wielkości więcej aniżeli usługa BLE HRT, wymagająca jedynie 1.9mW. Co naturalne, również różnice pomiędzy skrajnymi wartościami pobieranego przez urządzenie prądu są znaczące. Wykorzystując ten sam sprzęt, średni pobierany

prąd przez skonfigurowany węzeł Mesh wynosi 11.98mA ze szczytem bliskim 30mA. Dla porównania, usługa BLE HRT konsumuje średnio 0.59mA prądu osiągając w *peak'u* ok. 14mA.

Różnica przede wszystkim wynika z różnego zastosowania poszczególnych elementów. Zgodnie za dokumentacją producenta i standardu BLE Mesh, energooszczędnym, głównie zasilanym baterijnie węzłem jest węzeł typu *Low Power Node - LPN* [20][37]. Węzeł tego typu oszczędza energię poprzez rzadkie uruchamianie swojego radia wykorzystując przyjacielski węzeł¹⁸ do przechowywania jego danych w buforze, tak by były dostępne dla pozostałych elementów w sieci.



Rysunek 34. Porównanie średniego zużycia energii pomiędzy BT Low Energy HRT i BLE Mesh

W wykonanym doświadczeniu wykorzystano zwykły, prosty węzeł. Zapewniając niskie opóźnienia w transmisji danych w sieci, węzeł ten spełnia inne zadania niż urządzenie oparte o zwyczajne usługi BLE bądź węzeł LPN. Usługę BLE HRT zaprojektowano już na poziomie standardu jako energooszczędne domyślnie. Urządzenie wykorzystujące takie usługi, będące najczęściej zasilanie baterijnie, może działać miesiące a nawet lata na pojedynczej baterii (np. typu pastylki CR2032). Dostrzeżona różnica jest więc oczekiwana, naturalna, wynikająca z różnych przeznaczeń wykorzystywanych konfiguracji.

3.3 Badanie Packet Error Rate

Celem niniejszego podrozdziału jest omówienie przeprowadzonego eksperymentu PER. Przedstawiona zostanie metodologia badań oraz zbierane parametry wraz z ich opisem i uzasadnieniem.

Wychodząc z definicji, badany problem zdefiniowany jest przedstawionym wcześniej równaniem 3. Stanowi ono podstawę dla eksperymentu. Jego zrozumienie pozwoliło zaprojektowanie właściwego

¹⁸z ang. *Friend Node* - węzeł pośredniczący, najczęściej zasilany ze stałego źródła energii służący nasłuchiwaniu, odbieraniu i przechowywaniu danych pochodzących z węzła LPN

doświadczenia jak i przygotowanie kompletnego stosu technologicznego niezbędnego do jego przeprowadzenia.

Ostatecznym efektem przeprowadzonego eksperimentu jest przedstawienie zebranych danych w postaci wykresów. Prezentują one badane cechy zmienne, zaprezentowane opisane w sekcji opisu metodologii. Końcowym krokiem jest wyciągnięcie wniosków z zebranych danych.

3.3.1 Metodologia badania

Procedurę badawczą skonstruowano bazując na wzorze 3. Niezbędnym mechanizmem, o które oparte jest doświadczenie, to zliczanie ilości pakietów. Zliczanie dotyczy zarówno węzła bliższego jak i również węzła dalszego odbierającego wysypane komunikaty. W przypadku tego drugiego elementu opracowano mechanizm odczytywania licznika zmian badanej wartości, patrz: 3.1.3.

Całość doświadczenia przeprowadzono z wykorzystaniem oprogramowania PC – 3.1.4. Oprogramowanie zapewnia dwie główne funkcjonalności: wysyłanie komunikatów przy określonej częstotliwości przez zadaną ilość czasu; odczytywanie wartości licznika węzła dalszego.

Wysyłanym komunikatem jest polecenie zmiany stanu modelu *Generic OnOff*. Wybrano ten standardowy element stosu BLE Mesh ze względu na jego uniwersalność. Nie ogranicza się on jedynie do wybranej platformy czy właściwościowego modelu. Model ten jest zdefiniowany przez Bluetooth SIG, przez co jest niezależny od producenta urządzeń. Czyni to eksperiment powtarzalny, niezależnie od mikrokontrolera.

Odczytywanie stanu licznika wymagało jednakże wykorzystania autorskiego rozwiązania oparte o właściwość modelu Mesh ST. Nie wpływa to jednak na ostateczny rezultat badań, gdyż stworzone polecenia wykorzystywane jest tylko do odczytu i wysłania wartości licznika węzła dalszego do węzła bliższego i komputera osobistego kontrolującego przepływy doświadczenia. Z każdą kolejną próbą badania PER ten licznik jest automatycznie zerowany, przez co sesja zliczeń zawsze rozpoczyna się od zera.

Eksperiment wyznaczający PER oparto o następujące czynniki zmienne:

- środowisko: teren leśny, teren zurbanizowany
- interwał zapytań
- dystans pomiędzy węzłami
- ilość węzłów składających się na sieć Mesh

Wyznaczanie PER odbyło się w dwóch różnych środowiskach. Jednym z głównych hipotez jest znaczący wpływ środowiska na jakość transmisji danych. Czynniki takie jak temperatura, wilgotność, rodzaj gleby czy tło radiowe może mieć wpływ na komunikację pomiędzy węzłami. Założono, iż tło radiowe może mieć największy wpływ na transmisję danych. Stąd dobrano możliwie skrajne miejsca do badań oceniacąc to jako najistotniejszy czynnik:

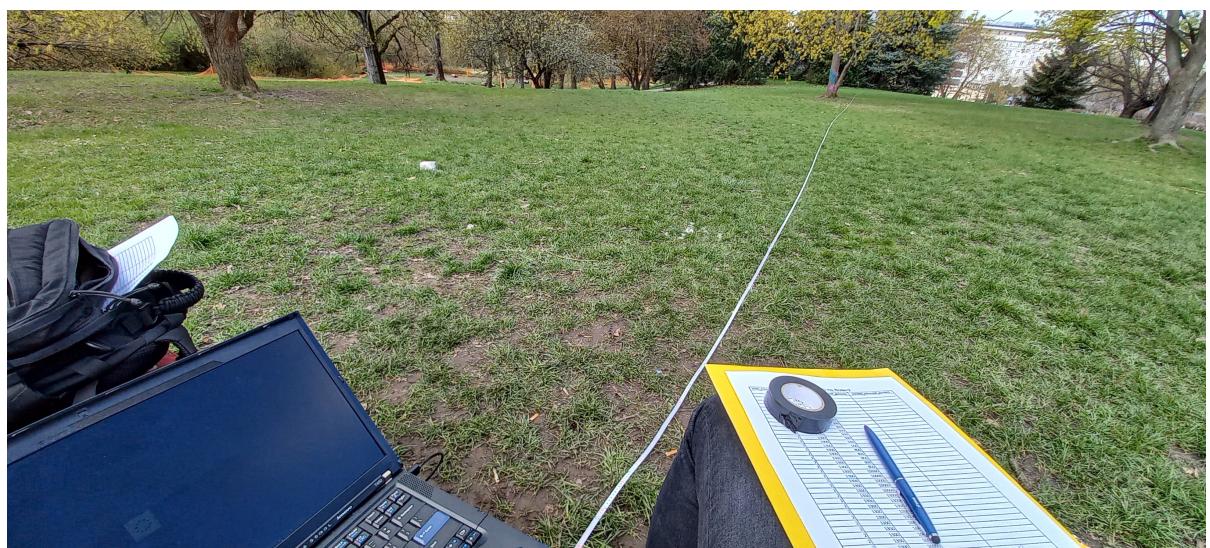
- Kampinoski Park Narodowy (lokalizacja: parking Roztoka) - jako teren leśny oddalony od ośrodka miejskiego ze względu niewielkim tłem radiowym. Pogoda: pochmurnie, wilgotno, temperatura poniżej 20°C.

3.3. Badanie Packet Error Rate



Rysunek 35. Fotografia miejsca badań w Kampinoskim Parku Narodowym

- Park Pola Mokotowskie - jako teren zurbanizowany charakteryzujący się bogatym tłem radiowym działającym w pasmach Industrial, Scientific, Medical (ISM). Pogoda: słonecznie, temperatura ok. 20°C.



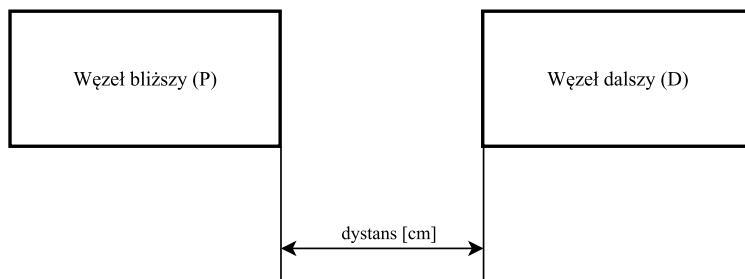
Rysunek 36. Fotografia miejsca badań na Polach Mokotowskich

Kolejnym badanym czynnikiem jest interwał zapytań. Parametr ten został wybrany ze względu na obserwowane problemy z komunikacją podczas etapu tworzenia oprogramowania. Parametry dobrano w takim stopniu, by ów problem ukazać. Obrano następujące interwały:

- 100ms
- 500ms
- 800ms
- 1300ms
- 2100ms

Interesującym parametrem dla bezprzewodowej transmisji danych jest zasięg. Stąd też, jednym z badanych czynników jest określenie jakości PER w zależności od odległości - Rysunek 37.

- 1,5m
- 3,0m
- 5,0m
- 8,0m
- 13,0m
- 16,0m
- 21,0m

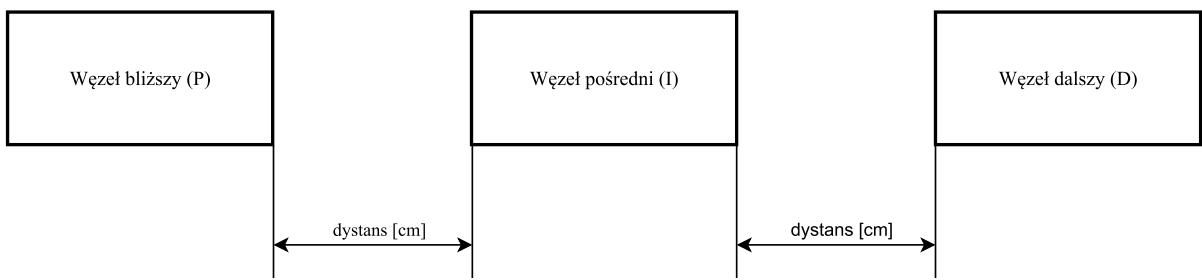


Rysunek 37. Dystans pomiędzy węzłami dla sieci dwóch mikrokontrolerów

Wyżej wymienione odległości stosowano również w przypadku kolejnego badanego parametru, tj. ilości węzłów składających się na sieć BLE. W celu łatwej identyfikacji węzłów, wprowadza się właściwe nazewnictwo opisane w punkcie 3.1.3.

Postanowiono o równoodległym rozstawieniu węzłów - Rysunek 38. Dla sieci 3 węzłów, maksymalna odległość dzieląca węzeł bliższy od węzła dalszego to 42m. Protokół badawczy zawiera informację tylko o odległości w rozumieniu równoodległego rozstawienia węzłów. Odległość pomiędzy elementami jest oczywistą operacją arytmetyczną.

Odległość pomiędzy węzłami mierzona jest z użyciem taśmy mierniczej z podziałką 1mm. Tolerancję pomiarów należy przyjąć jako najdłuższy wymiar zestawu uruchomieniowego P-NUCLEO-WB55 - 70mm [26]. Pomiar odbywał się na płasko, mierząc odległość pomiędzy leżącymi na glebie węzłami. Błędy pomiarowe wynikłe z ukształtowania terenu są prawdopodobne i wynikają z poza laboratoryjnego charakteru badań.



Rysunek 38. Dystans pomiędzy węzłami dla sieci trzech mikrokontrolerów

W celu zminimalizowania ryzyka pojawienia się losowych błędów o nieznanym pochodzeniu badanie powtarzano pięciokrotnie, w sposób następujący: dla wybranego środowiska, ilości węzłów i zadanej odległości pomiędzy węzłami, wykonaj 5 powtórzeń pomiarowych dla każdego z zadanych interwałów pomiarowych.

$$\exists e, \exists n, \exists d : PER_i = f(e, n, d), i = 1, \dots, 5 \quad (4)$$

gdzie:

e - środowisko

n - ilość węzłów

d - równoodległy dystans pomiędzy węzłami

i - ilość powtórzeń

PER - Packet Error Rate dla wybranego powtórzenia przy stałych parametrach e, n, d

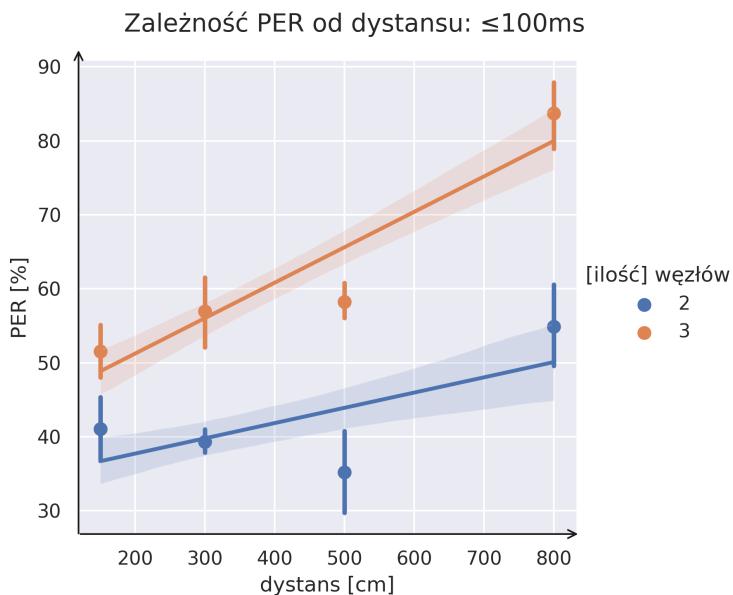
Wykorzystując tak zebrane dane, wyrysowano je na szeregu wykresów wyznaczając jednocześnie linie aproksymacyjne z użyciem modelu liniowego wyznaczonego metodą najmniejszych kwadratów. Odchylenia standardowe zaprezentowane są z użyciem ograniczonego tła otaczającego linię w tym samym kolorze o zmienionym parametrze nieprzezroczystości.

Parametry transmisji danych nie ulegały zmianie podczas przeprowadzanych doświadczeń. Poniższe wartości należy przyjąć za stałe:

- Szybkość transmisji: 2Mbps
- Moc transmisji danych: 0dBm

3.3.2 Zależność PER względem częstości zapytań

Pierwszą sprawdzaną hipotezą jest empiryczna weryfikacja czy częstość zapytań wpływa na PER. W tym celu wykreślono szereg wykresów dla wybranych interwałów czasowych. Pozostałe parametry traktowane są jako stałe.



Rysunek 39. Zależność PER od dystansu dla zapytań o częstotliwość $\leq 100\text{ms}$ dla różnej liczby węzłów

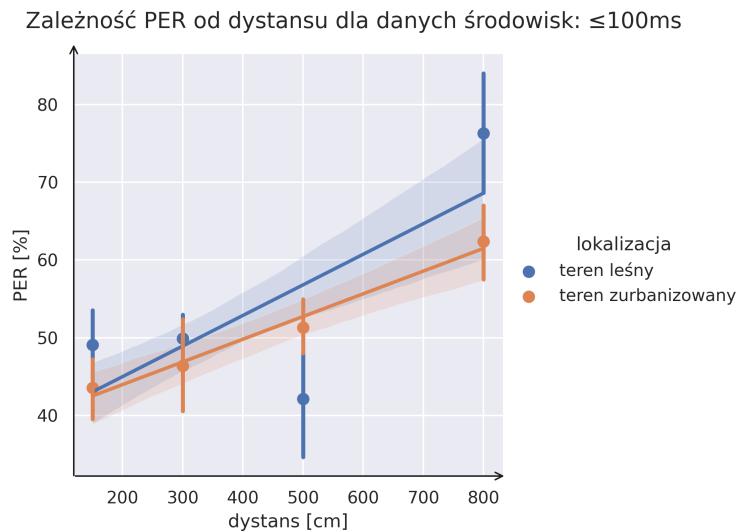
Rysunek 39 przedstawia zależność PER dla interwału 100 milisekund w zależności od odległości dla sieci złożonej z dwóch i trzech węzłów.

Niewątpliwą cechą ukazanych danych jest wysoka wartość PER już przy tak niewielkiej odległości jak 150 cm pomiędzy węzłami. Utrata 40-50% wysyłanych pakietów danych już na pierwszym dystansie pomiarowym może być spowodowana wieloma czynnikami. Przypuszczalnie, wpływ na taki rezultat wywodzi się z czynników środowiskowych lub ze sposobu wykonywania eksperymentu. Niewykluczone są również ograniczenia sprzętowe mikrokontrolera bądź stosu BLE Mesh.

Wraz ze wzrostem odległości pomiędzy węzłami PER wzrasta pomimo wysokiej wartości początkowej. Jest to oczekiwana zależność i zgodna z wiedzą techniczną.

Prowadząc dalszą analizę zależności PER od częstotliwości zapytań, sprawdzono wpływ środowiska na jakość transmisji danych. Rysunek 40 przedstawia wybraną zależność. Rozróżnienie na ilość węzłów nie zostało tutaj uwzględnione. Pomiary przeprowadzone w najbliższym dobranym dystansie ponownie wskazują na 40-50% utratę pakietów w węźle dalszym. Analogiczne rezultaty obserwowane są na pozostałych dystansach z oczekiwana tendencją wzrostową. Wykres dodatkowo przedstawia pewną różnicę w jakości transmisji danych w zależności od środowiska. Różnica ta jednak mieści się w odchyleniu standardowym, posiadając część wspólną dla zadanych parametrów. Nie jest to wystarczające do stwierdzenia jednoznacznego wpływu środowiska.

Zestawiając ze sobą wymienione wcześniej czynniki, obserwuje się interesujące zależności. Rysunek 41 wskazuje na zależność dystansu, ilości węzłów i rodzaju środowiska dla wybranego interwału zapytań 100 milisekund. Po raz kolejny obserwowany jest PER wynoszący 40-50%, niezależnie od środowiska czy ilości węzłów. Sugeruje to wpływ samej testowanej platformy na ostateczny rezultat. Prawdopodobną hipotezą jest niewystarczająca wielkość zaallokowanych



Rysunek 40. Zależność PER od dystansu dla zapytań o częstotliwość $\leq 100\text{ms}$ w wybranych środowiskach bez rozróżnienia na liczbę węzłów

buforów obsługujących transmisję danych. Mikrokontroler, nie będąc w stanie obsłużyć tak częstej transmisji, może doświadczyć awarii, co zaobserwowało podczas badań. Awaria objawiała się brakiem reakcji węzła bliższego na jakiekolwiek komendy AT, co wymagało ponownego uruchomienia urządzenia. Weryfikacja tego zagadnienia wymagałaby zaangażowania zaawansowanych narzędzi programistycznych ingerujących m.in. w pamięć urządzenia. Niniejsza praca nie podejmuje się wyjaśnienia przyczyn obserwowanych anomalii w działaniu mikrokontrolera, udostępniając jednocześnie możliwy punkt dla dalszych prac badawczych z zakresu BLE Mesh.

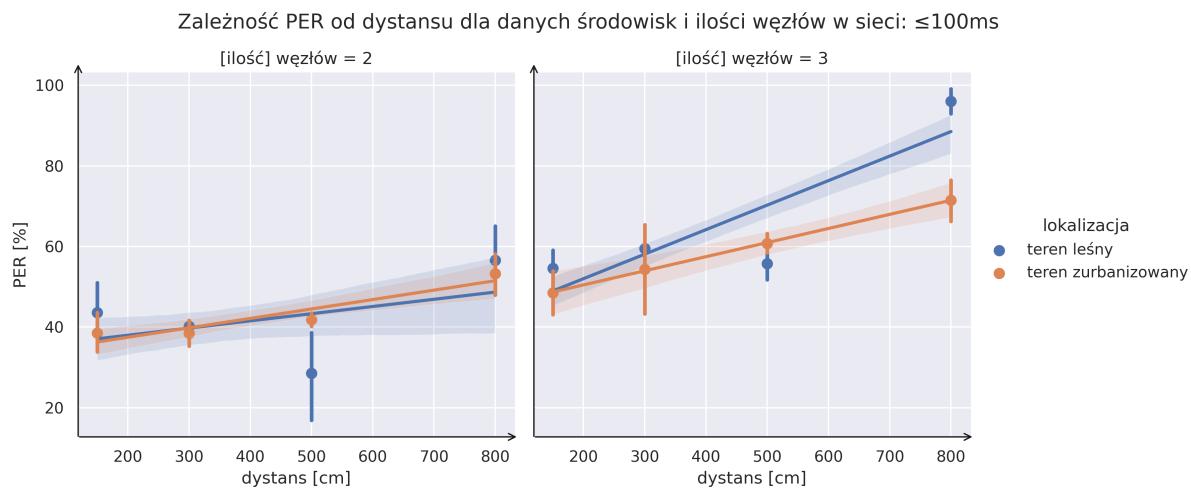
Wpływ samego stosu łączności na PER przy zadanej częstotliwości zapytań zdaje się potwierdzać wykres uwzględniający jakość transmisji dla dwóch węzłów. Niemalże pozioma linia aproksymacji sugeruje niewielki wpływ dystansu na PER. Dotyczy to zarówno terenu zurbanizowanego jak i terenu leśnego. Zbliżone rezultaty zdają się wykluczać czynniki zewnętrzne.

Interesującą zależnością jest nachylenie wykresu względem osi odciętych. Dla przypadku dwóch węzłów sieci, połączenie bezpośrednio między węzłami, PER jest niemal stałe na wybranych odległościach, porównywalne z przypadkiem terenu leśnego. W przypadku trzech węzłów, uwidacznia się potencjalny wpływ węzła środkowego, przekazującego pakiety z punktu bliższego do dalszego. Wraz ze wzrostem dystansu, rośnie wartość PER sięgając nawet 90% w terenie leśnym. Różne nachylenie dla wybranych środowisk również sugeruje znaczący wpływ środowiska na PER.

Znając charakterystykę łączności dla okresów poniżej 100 milisekund, weryfikuje się pozostałe wybrane częstotliwości, zgodnie z podanymi wartościami 3.3.1. Rysunek 42 prezentuje pozostałe interwały w zależności od dystansu i wybranych środowisk bez rozróżnienia na ilość węzłów w sieci.

Przeprowadzone pomiary w terenie leśnym wskazują charakterystykę połączenia zgodną z intuicyjnymi przewidywaniami. Na początkowym dystansie 150cm nie obserwuje się problemów z łącznością. Wszystkie wysłane pakiety zostały odebrane przez węzeł dalszy. Kolejny dystans wskazuje już na

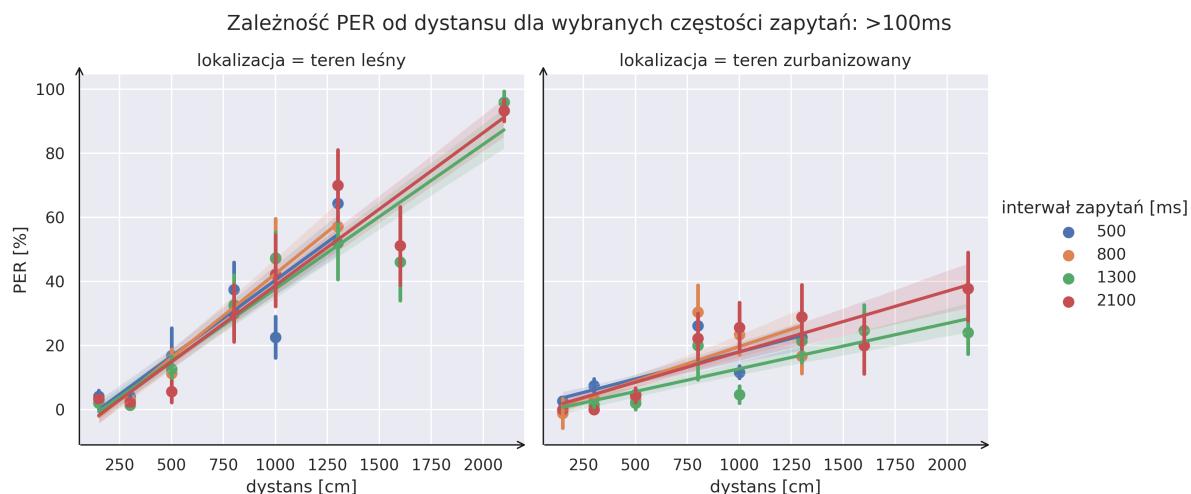
Rozdział 3. Część doświadczalna



Rysunek 41. Zależność PER od dystansu dla zapytań o częstotliwość $\leq 100\text{ms}$ w wybranych środowiskach i liczbę badanych węzłów

pewną utratę pakietów poniżej 20%. Prawdopodobnym czynnikiem jest pogoda lub ukształtowanie terenu leśnego. Co istotne, pomiary dla różnych interwałów odpytywań są zbliżone. Sugerowałoby to brak wpływu częstotliwości na PER. Na pozostałych dystansach pomiarowych, wartości utraty pakietów są do siebie wzajemnie zbliżone osiągając swoje maksimum na dystansie 21m - blisko 100% zaginionych pakietów.

Charakterystyka łączności w terenie zurbanizowanym prezentuje się podobnie. Nie obserwuje się znaczącej utraty pakietów na względnie bliskich dystansach (150, 300 i 500cm). Wartość PER rośnie wraz ze wzrostem odległości pomiędzy węzłami osiągając w swoim szczytce wartość ok. 40%. Co istotne, linie aproksymacyjne są do siebie zbliżone, ponownie sugerując brak korelacji pomiędzy częstotliwością wysyłania komunikatów a wartością PER.



Rysunek 42. Zależność PER od dystansu dla zapytań o częstotliwość $>100\text{ms}$ w wybranych środowiskach

Zależność pomiędzy PER a częstością odpytywań została zbadana wykorzystując narzędzia statystyczne. Traktując model liniowy, jak sugerują zaprezentowane wykresy, tworzonych zgodnie z metodą najmniejszych kwadratów wylicza się współczynnik determinacji dla następujących zmiennych niezależnych: dystans oraz PER. W kolejnym kroku uwzględnia się fakt wykorzystywania korelacji wielu zmiennych, dostosowując odpowiednio wartość.

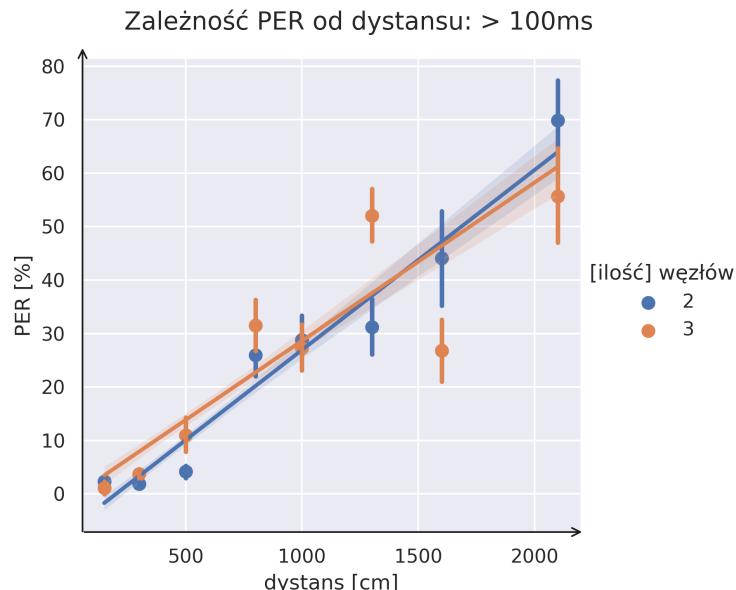
Środowisko	R^2	$R^2_{dostosowany}$
Teren leśny	0.04958	0.04272
Teren zurbanizowany	0.04887	0.04200

Tablica 3. Współczynnik determinacji dla zależności interwału zapytań od dystansu i PER w wybranych środowiskach

Ostatecznie, otrzymany współczynnik determinacji przyjmujący wartość $R^2 = 0,04\text{--}0,05$. Istnieje 4%-owy wpływ interwału zapytań o częstościach większych niż 100 milisekund na ostateczny rezultat PER. Pozwala to wykluczyć ten czynnik wykluczyć z dalszych rozważań uznając go za marginalny.

3.3.3 Zależność PER względem odległości między węzłami

Ustalwszy brak (pomijalnie mały) wpływu częstości zapytań (dla częstości wartości $>100\text{ms}$), praca podejmuje dalszą prezentację danych pod postacią zależności odległości na PER.

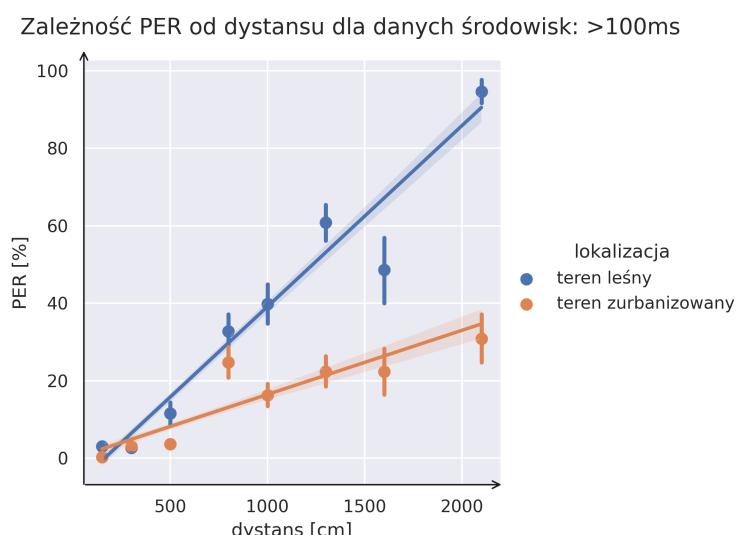


Rysunek 43. Zależność PER od dystansu dla zapytań o częstotliwość $>100\text{ms}$ dla różnej liczby węzłów

Rysunek 43 przedstawia zebrane dane, zestawiając odległość i liczbę węzłów składających się na sieć Mesh. Na początkowych dystansach PER ma wartość bliską bądź równą零. Wraz ze wzrostem odległości pomiędzy węzłami, badany współczynnik rośnie, co jest zgodne z oczekiwaniemi. Na dystansie 16m obserwuje się przełamanie linii aproksymacji. Dodatkowy węzeł środkowy,

prawdopodobnie, znacząco i pozytywnie wpływa na PER przy kolejnych odległościach umożliwiając przekazywanie informacji w sieci. PER w najdalszym punkcie pomiarowym osiąga wartości od ok. 60% do 70%. Należy jednak mieć na uwadze, iż zestawienie nie rozróżnia środowiska w którym następowało zliczanie pakietów.

Rysunek 44 uwzględnia wpływ środowiska na PER. Wykres zdecydowanie wskazuje na różnicę pomiędzy terenem zurbanizowanym a terenem leśnym. Na początkowych dystansach PER jest zbliżone niezależnie od odległości międzywęzłowych. Znaczące różnice w pomiarach, a dzięki temu również względem linii aproksymacyjnej, występują już na dystansie 5m. PER w przypadku miejskim jest bliskie zera, gdzie analogiczne pomiary w środowisku leśnym sugerują piętnastoprocentowy poziom zgubionych pakietów. Wraz ze wzrostem dystansu, wartość PER wzrasta. Niemniej jednak nachylenie linii wskazuje na zdecydowanie wolniejsze narastanie utraty danych podczas transmisji bezprzewodowej dla przypadku miejskiego. Na maksymalnym dystansie międzywęzłowym wynoszącym 21m, PER przybiera wartość ok. 30%. W analogicznym przypadku dla środowiska leśnego następuje niemal całkowita utrata transmisji.



Rysunek 44. Zależność PER od dystansu dla zapytań o częstotliwość $>100\text{ms}$ w wybranych środowiskach bez rozróżnienia na liczbę węzłów

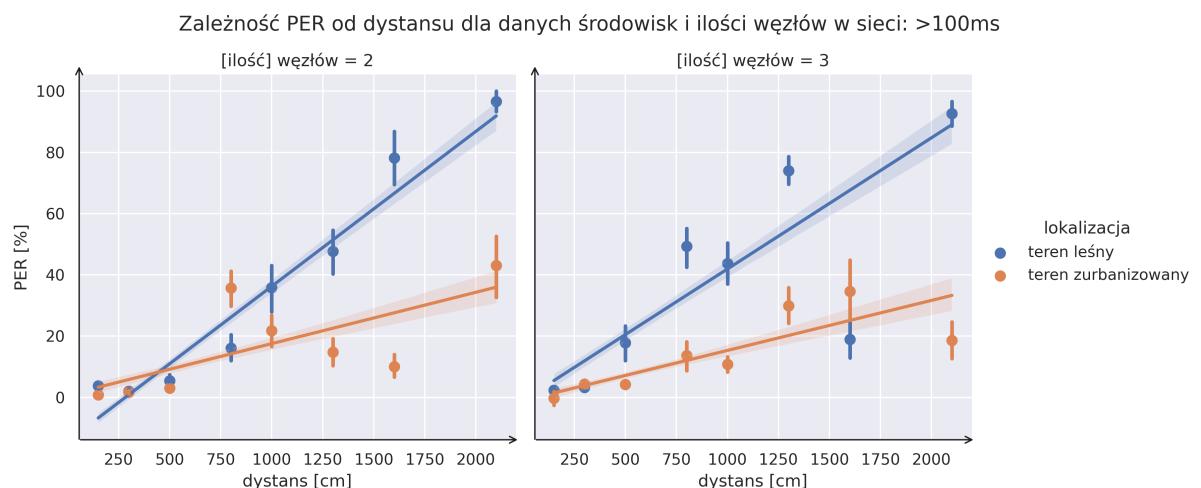
Ostatnia prezentowana zależność ukazana jest na Rysunku 45. Przedstawia on zależność PER od dystansu dla wybranych środowisk i liczby węzłów. Uwidaczniają się wyżej wymienione cechy transmisji opisane w poprzednich akapitach.

Dla przypadku dwóch węzłów, PER jest bliskie zeru na początkowych dystansach do 5m niezależnie od środowiska. W odległości 8m od stacji bazowej obserwuje się miejscową anomalię polegającą na blisko 40%-ej utraty pakietów dla terenu zurbanizowanego. Zarówno jak w poprzednich jak i kilku następujących po sobie punktach pomiarowych, PER nie przybiera takich wartości. Anomalia może być wyjaśniona cechami środowiska – np. węzeł dalszy ułożony został w zagłębiu, co utrudniło odbiór propagowanej fali radiowej. W pozostałych punktach pomiarowych w terenie zurbanizowanym

obserwuje się spadek wartości PER wraz ze zwiększonym dystansem, co znów sugeruje wpływ ukształtowania terenu. Utrata pakietów wynosząca niewiele powyżej 40% zaobserwowana jest na odległości 21m. Jest to o tyle zaskakujące, o tyle dla analogicznego dystansu pomiarowego w terenie leśnym PER wynosi niemal 100%.

Charakterystyka transmisji dla badanego przypadku trzech węzłów przybiera podobną postać. Wartości PER w terenie zurbanizowanym są zbliżone w początkowych dystansach pomiarowych. Dystans 8m (węzeł dalszy na odległości 16m) wskazuje na analogiczną anomalię jak w opisywanym przypadku dwóch węzłów. Obserwuje się maksimum lokalne PER poniżej 20% w otoczeniu najbliższych punktów pomiarowych. Kolejne pomiary wskazują na stały przyrost PER względem równoodległych węzłów. Utrata pakietów na maksymalnym całkowitym dystansie 42m wynosi ok. 20%.

Transmisja danych w środowisku leśnym charakteryzuje się większym przyrostem PER względem odległości w porównaniu do warunków miejskich. Przypadek dwuwęzłowej sieci wskazuje zbliżone wartości utraty pakietów na odległości do 5m włącznie. Na każdym kolejnym dystansie pomiarowym PER wzrasta zbliżając się do 100% w miejscu oddalonym o 21m od węzła bliższego. Oznacza to niemal całkowitą utratę łączności. Charakterystyka dla sieci zbudowanej z trzech węzłów jest analogiczna. Wyjątkiem jest zbiór pomiarów zebranych na dystansie pomiędzy węzłami wynoszącym 16m. PER w tym przypadku spada do ok. 20%. Jest to o tyle niespodziewane iż, poprzedzające i następujące pomiary i wynikłe w konsekwencji PER są blisko 4-krotnie wyższe. Taką konsekwencję należy przypisać ukształtowaniu terenu aniżeli nieznanej właściwości transmisji radiowej.



Rysunek 45. Zależność PER od dystansu dla zapytań o częstotliwość >100ms w wybranych środowiskach i liczbę badanych węzłów

Przedstawione dane prezentują wpływ dystansu na PER w różnych środowiskach. Określona kategoria badanych środowisk nie jest jedynym parametrem charakteryzującym przeprowadzone badania. Dużą odpowiedzialność w ostatecznych zebranych wartościach należy położyć na czynniki takie jak ukształtowanie terenu i pogoda. Pomimo organoleptycznego doboru terenu o możliwie

Rozdział 3. Część doświadczalna

nieurozmaiconej charakterystyce, badane węzły mogły zostać ułożone w lokalnym zagębieniu wynikłym z gleby bądź nawet zagęszczeniu trawy w danym miejscu. Różnica w wilgotności również mogła mieć znaczenie w zebranych danych zwiększąc bądź odpowiednio zmniejszając wartości konduktancji ośrodka, w którym propagowana została fala radiowa. Niepewności te są konsekwencją metodologii przeprowadzonych badań w warunkach nielaboratoryjnych (terenowych). Czynniki te, nie będące częścią formalnie zebranych danych, najprawdopodobniej miały wpływ na ilość odebranych pakietów. Ukształtowanie terenu jak i wilgotność mają bezpośredni wpływ jakość transmisji stanowiąc przeszkody dla strefy propagacji sygnału radiowego (strefy Fresnela).

Rozdział 4

Podsumowanie

Niniejsza praca zrealizowała postawione założenia zdefiniowane we wstępie – rozdział 1. Wprowadzony definicję sygnału płynnie wprowadzono zagadnienia związane z rozpatrywanymi standardami komunikacji bezprzewodowej.

Zestawiając wprowadzone protokoły, na szczególną uwagę zasługują dwa z nich: badany Bluetooth 5 (z wyróżnieniem Mesh) oraz Thread. Opierając się na wiedzy pochodzącej z powszechnie dostępnej literatury, Thread osiągnął znaczaco *lepsze parametry transmisji danych*. Korzyść ta rozumiana jest jako najwyższa przepustowość dla pofragmentowanych danych (Rysunek 10) oraz najmniejsze opóźnienia niezależnie od wielkości wiadomości (Rysunek 11) spośród testowanych rozwiązań.

Thread nie definiuje warstwy aplikacji, jak to umożliwiają bądź wymuszają pozostałe rozważane standardy. Czyni to protokół uniwersalnym, niemalże ogólnego przeznaczenia, niczym protokoły TCP/IP. Porównanie jest również nie bez znaczenia, gdyż Thread opiera się właśnie na IPv6, co znów zmniejsza nabylenie krzywej uczenia się tego standardu i umożliwia niemalże bezpośrednio wpiecie do Internetu i rozwiązań opartych o publiczną chmurę obliczeniową. Protokół ten, będąc oparty o IEEE 802.15.4, nie jest energoszczędny do tego stopnia, co protokół BLE (Rysunek 12). Nie czyni go to jednak gorszym a raczej przypisuje go do innych zastosowań. Potencjalne problemy może stwarzać niewielka popularność tego protokołu wśród oferowanych na rynku produktów. Jest to niemniej powiązane ze względnie nową architekturą danej specyfikacji. Utrudnieniem, które bezpośrednio wpływa na kompatybilność produktów różnych producentów, jest brak wprost zdefiniowanej warstwy aplikacji. To wyzwanie adresują projekty, takie jak Matter¹ udostępniając specyfikację integrującą rozwiązania oparte o Thread.

Bluetooth Mesh, pomimo rezultatów wywodzących się z powszechnej literatury, również spełnia zdefiniowane przed nim standardem zadania. Protokół kładzie szczególny nacisk na organizację i hierarchię transmitowanych wiadomości w warstwie. Można wystosować wręcz opinię, że został stworzony do wysyłania *tylko jednego komunikatu*, by wykonać konkretną akcję. Takie twierdzenie można poprzeć zarówno sposobem routingu i wspomnianej hierarchii wiadomości (modele Mesh), ale

¹<https://csa-iot.org/all-solutions/matter/>

Rozdział 4. Podsumowanie

również biorąc uwagę wyniki reprezentowane na Rysunku 11. Pojedynczy wysłany komunikat cechuje się nie gorszym opóźnieniem niż pozostałe badane protokoły. Uwzględniając energooszczędność tego standardu, czyni go nadzwyczaj interesującą opcją dla konkretnych zastosowań, np. inteligentnego oświetlenia.

Dalsza część pracy to ćwiczenie praktyczne polegające na pomiarze zużycia energii. Zaprezentowano charakterystyki poboru prądu w czasie dla różnych trybów funkcjonowania urządzenia dla dwóch różnych firmware'ów. Pierwszy rozważany przypadek oparty o BLE HRT jest zgodny z przeprowadzoną symulacją – Rysunki 13 oraz 14. Wykonana zgrubna symulacja, bazująca na domyślnych parametrach wskazywanych przez dokumentację, wskazywała na średni pobór prądu elektrycznego na poziomie 2,95mA (moc: 9,8mW) [35]. Jest to co prawda ponad 4-krotna różnica względem uzyskanych pomiarów – maksymalny pobór prądu 0,69mA (moc: 2,27mW) – jest ona jednak na korzyść eksperymentu. Sugeruje to błąd dokonanych nastawów symulatora. Dopracowanie tych ustawień może stanowić kontynuację badań tego zagadnienia. Dopracowanie tych ustawień może stanowić kontynuację

Przypadek zużycia Mesh wskazuje na zużycie energii węzła klasycznego serwera *Generic OnOff*. Węzeł tego najczęściej włączony jest do stałego źródła zasilania. Stąd, porównanie z Rysunku 34, sugeruje na konieczność porównywania BLE do węzła typu *LPN*, który w zamыśle – potwierdzonym z oficjalną specyfikacją – funkcjonuje z wykorzystaniem zasilania baterijnego. Słuszny rozwinięciem tego badania jest wykorzystania *LPN* z jednosekundowym odświeżaniem swojego stanu (zakładając model *Sensors*). Energooszczędność BLE i Bluetooth Mesh wprost wynika z optymalizacji wykorzystania radia, co udowadnia choćby porównanie wyników zaprezentowanych na charakterystykach 28 oraz 29, gdzie podczas rozgłaszenia niskomocowego (zgodnie z opisywanymi w danym rozdziale założeniami), radio przez większość czasu nie funkcjonuje.

Eksperiment PER wskazuje na kilka interesujących zależności. Pierwszą z nich jest wydajność stosu BT mikrokontrolera STM32WB55RG. Przed przystąpieniem do formalnego doświadczenia, przeprowadzono szereg prób z częstotliwościami wykonywanych zapytań znaczaco mniejszymi od 100ms – w analogii do ciągu Fibonacciego testowano interwały 10, 20, 30, 50 i 80ms. Każdy z nich wykazywał tę samą cechę utraty pakietów, jak na przedstawionym Rysunku 39. Przeważnie, po kilku-kilkunastu sekundach pracy węzeł bliższy nie odpowiadał na jakiekolwiek komunikaty AT. Po ponownym uruchomieniu tego urządzenia, działało bez zarzutu umożliwiając pobranie informacji o odebranej liczbie pakietów z węzła dystanego. Sugeruje to raczej problemy ze stosem i ewentualnym wyciekiem pamięci uniemożliwiające poprawne funkcjonowanie sprzętu. Jednym z kroków do dalszych analiz jest weryfikacja kodu i jego działania w czasie rzeczywistym. Narzędzia jakie debugger czy valgrind² byłyby pierwszym krokiem ku weryfikacji zagadnienia.

Ostatnią, aczkolwiek najistotniejszą zbadaną zależnością, jest wpływ dystansu na PER. Zgodnie ze zdefiniowaną hipotezą w punkcie 2.5 i opisywaną później metodologią, oczekiwano dominującą rolę otaczającego środowiska radiowego w przeprowadzanym doświadczeniu. Projektując eksperiment

²<https://valgrind.org/>

nie założono znaczącego wpływu środowiska, ani wpływu rozchodzenia się fal powierzchniowych. Zgodnie z założeniami zaobserwowano spadek PER wraz ze wzrostem odległości. Nie obserwuje się jednak znaczącego wpływu zależności ilości węzłów na PER. Rysunek 45 prezentuje PER w postaci aproksymacji liniowej, gdzie porównywane rezultaty dla danego środowiska od ilości węzłów są zbliżone. W punktach pomiarowych (teren zurbanizowany) 13m i 16m równoodlegle, widać znaczny spadek PER dla dwóch węzłów, gdzie w przypadku trzech węzłów tendencja ta jest nadal wzrostowa. Dopiero na maksymalnym dystansie pomiarowym uwidacznia się ok. 20% PER dla konfiguracji 3-ech urządzeń – mniejszy niż dla 2-óch urządzeń.

Rezultaty są nieprzekonywujące. Wpływ wilgotności i temperatury dla propagacji sygnału omawia praca [43]. Wilgotność ma negatywną korelację względem siły sygnału. Tłumaczone jest to stratami wynikłymi z absorpcji energii sygnału. Wymieniona zależność najprawdopodobniej uwidacznia się dla badania w terenie leśnym. Dodatkowo, wahania w PER na różnych dystansach przy środowisku zurbanizowanym może sugerować nierówności terenu, czyli przeszkodę z punktu widzenia sygnału. Ten czynnik również nie został formalnie uwzględniony.

Podsumowując badanie PER, zauważa się wpływ odległości pomiędzy węzłami na wzrost wartości PER. Zawarty w hipotezie badania wpływ tła radiowego (i wynikające z niego interferencja i osłabianie fali) jest pomijalnie mały lub czynniki trzecie w postaci wilgotności i nierówności terenu uniemożliwiają klarowną identyfikację tego zjawiska.

Poza wymienionymi już możliwościami dalszego rozwoju badań, sugeruje się kolejne, następujące. Uwzględniając fakt utraty pakietów ze względu na krótki interwał odpytywań, oczywistym jest wyszukanie punktu granicznego, w którym omawiana zależność nie występuje. Celem byłoby wyznaczenie minimalnego interwału, który nie powoduje utraty pakietów na niewielkich odległościach w kontrolowanych warunkach radiowych i środowiskowych.

Badanie PER można dodatkowo rozwinać uwzględniając czynniki takie jak wilgotność i temperatura otoczenia oraz wpływ fali powierzchniowej na transmisję sygnału. Oznacza to konieczność wykonania pomiarów w zblighonych, mierzalnych warunkach. Wpływ fali powierzchniowej można zminimalizować poprzez umieszczenie węzłów na czas badań na odpowiednio wysokim statywie. Umożliwiłoby to dodatkowo eliminację przeszkód ze strefy Fresnela dla danych urządzeń nadawczo-odbiorczych, ujednolicając warunki badań.

Praca zrealizowała postawione przed nią cele.

Bibliografia

- [1] „AN1142: Mesh Network Performance Comparison,” en, s. 11,
- [2] *Bluetooth Protocol Stack - MATLAB & Simulink*. adr.: <https://www.mathworks.com/help/bluetooth/ug/bluetooth-protocol-stack.html> (term. wiz. 15.05.2022).
- [3] *Bluetooth® technology media information*, en-US. adr.: <https://www.bluetooth.com/media/> (term. wiz. 29.05.2022).
- [4] Curtis, B., *CES 2022: Matter And Thread Win The IoT Connectivity Wars*, en, Section: Cloud. adr.: <https://www.forbes.com/sites/moorinsights/2022/01/11/ces-2022-matter-and-thread-win-the-iot-connectivity-wars/> (term. wiz. 30.05.2022).
- [5] Fafoutis, X., Tsimbalo, E., Zhao, W., Chen, H., Mellios, E., Harwin, W., Piechocki, R. i Craddock, I., „BLE or IEEE 802.15.4: Which Home IoT Communication Solution is more Energy-Efficient?” *EAI Endorsed Transactions on Internet of Things*, t. 2, nr. 5, grud. 2016, ISSN: 2414-1399. adr.: <https://eudl.eu/doi/10.4108/eai.1-12-2016.151713> (term. wiz. 28.05.2022).
- [6] Georgakakis, E., Nikolidakis, S. A., Vergados, D. D. i Douligeris, C., „An Analysis of Bluetooth, Zigbee and Bluetooth Low Energy and Their Use in WBANs,” en, w *Wireless Mobile Communication and Healthcare*, Lin, J. C. i Nikita, K. S., red., ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Berlin, Heidelberg: Springer, 2011, s. 168–175, ISBN: 978-3-642-20865-2. DOI: 10.1007/978-3-642-20865-2_22.
- [7] IEEE P802.15 Working Group, „IEEE Standard for Low-Rate Wireless Networks,” IEEE, spraw. tech., ISBN: 9781504466899. DOI: 10.1109/IEEESTD.2020.9144691. adr.: <https://ieeexplore.ieee.org/document/9144691/> (term. wiz. 28.05.2022).
- [8] Laboratories, S., „UG103.11: Thread Fundamentals,” en, s. 25, 2022.
- [9] Lethaby, N., „Wireless connectivity for the Internet of Things: One size does not fit all,” en, s. 16, 2017.
- [10] McEwen, A. i Cassimally, H., *Designing the Internet of Things*, 1 wyd. Wiley, 2013, ISBN: 978-1-118-43062-0. adr.: <https://www.wiley.com/en-us/Designing+the+Internet+of+Things-p-9781118430620>.
- [11] Mesh Working Group, *Mesh Profile Bluetooth® Specification*, sty. 2019. adr.: <https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/>.

Bibliografia

- [12] *Node Roles and Types / OpenThread*, en, 2022. adr.: <https://openthread.io/guides/thread-primer/node-roles-and-types> (term. wiz. 29.05.2022).
- [13] *P-NUCLEO-WB55 - Bluetooth 5 and 802.15.4 Nucleo Pack including USB dongle and Nucleo-64 with STM32WB55 MCUs, supports Arduino Uno V3 and ST morpho connectivity - STMicroelectronics*, en. adr.: <https://www.st.com/en/evaluation-tools/p-nucleo-wb55.html> (term. wiz. 10.05.2022).
- [14] Ray, P. P., Dash, D. i De, D., „Edge computing for Internet of Things: A survey, e-healthcare case study and future direction,” en, *Journal of Network and Computer Applications*, t. 140, s. 1–22, sierp. 2019, ISSN: 1084-8045. DOI: 10.1016/j.jnca.2019.05.005. adr.: <https://www.sciencedirect.com/science/article/pii/S1084804519301651> (term. wiz. 29.05.2022).
- [15] Rzepecki, W. i Ryba, P., „IoTSP: Thread Mesh vs Other Widely used Wireless Protocols – Comparison and use Cases Study,” w *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*, sierp. 2019, s. 291–295. DOI: 10.1109/FiCloud.2019.00048.
- [16] SA, H., *TCP/IP. Szkoła programowania*, pl, ISBN: 978-83-246-0293-3. adr.: <https://helion.pl/ksiazki/tcp-ip-szkola-programowania-heather-osterloh,tcpiszp.htm> (term. wiz. 15.05.2022).
- [17] *scipy.integrate.simpson — SciPy v1.8.1 Manual*. adr.: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.simpson.html> (term. wiz. 25.05.2022).
- [18] Silicon Laboratories, „UG103.02: Zigbee Fundamentals,” en, s. 31, 2021.
- [19] Skoro, M., *Fizyka*. Państwowe Wydawnictwo Naukowe, 1973.
- [20] ST, *AN5292 - How to build a Bluetooth ® Low Energy mesh application for STM32WB Series microcontrollers*, en, grud. 2021.
- [21] *STM32CubeIDE - Integrated Development Environment for STM32* - STMicroelectronics, 2022. adr.: <https://www.st.com/en/development-tools/stm32cubeide.html> (term. wiz. 14.05.2022).
- [22] *STM32CubeMonPwr - Graphical tool displaying on PC power data coming from X-NUCLEO-LPM01A* - STMicroelectronics, 2022. adr.: <https://www.st.com/en/development-tools/stm32cubemonpwr.html> (term. wiz. 14.05.2022).
- [23] *STM32CubeProg - STM32CubeProgrammer software for all STM32* - STMicroelectronics, 2022. adr.: <https://www.st.com/en/development-tools/stm32cubeprog.html> (term. wiz. 14.05.2022).
- [24] *STM32CubeWB MCU Firmware Package*, original-date: 2019-04-19T13:27:35Z, maj 2022. adr.: <https://github.com/STMicroelectronics/STM32CubeWB> (term. wiz. 14.05.2022).
- [25] *STM32WB - Bluetooth, Wireless Microcontrollers (MCU)* - STMicroelectronics, 2022. adr.: <https://www.st.com/en/microcontrollers-microprocessors/stm32wb-series.html> (term. wiz. 14.05.2022).

- [26] STMicroelectronics, *UM2435 - Bluetooth® Low Energy and 802.15.4 Nucleo pack based on STM32WB Series microcontrollers*, 2019.
- [27] ——, *AN5506 - Getting started with Zigbee® on STM32WB Series*, lip. 2020. adr.: https://www.st.com/resource/en/application_note/dm00710974-getting-started-with-zigbee-on-stm32wb-series-stmicroelectronics.pdf.
- [28] ——, *AN5289 - Building wireless applications with STM32WB Series microcontrollers*, 2021.
- [29] ——, *PM0271 - STM32WB BLE stack programming guidelines*, grud. 2021. adr.: https://www.st.com/resource/en/programming_manual/pm0271-stm32wb-ble-stack-programming-guidelines-stmicroelectronics.pdf.
- [30] Szabatin, J., *Podstawy teorii sygnałów*. Wydawnictwa Komunikacji i Łączności, 2007, ISBN: 978-83-206-1331-5. adr.: <https://www.wkl.com.pl/podstawy-teorii-sygnalow,1,1,283?> (term. wiz. 31.05.2022).
- [31] Szóstka, J., *Fale i anteny*, 3. Wydawnictwa Komunikacji i Łączności, 2006, ISBN: 978-83-206-1626-2.
- [32] *Thread - Overview*. adr.: <https://www.threadgroup.org/What-is-Thread/Overview> (term. wiz. 29.05.2022).
- [33] *Thread Group*. adr.: <https://www.threadgroup.org/thread-group> (term. wiz. 30.05.2022).
- [34] Thread Group, *Thread Network Fundamentals*, maj 2020. adr.: https://www.threadgroup.org/Portals/0/documents/support/Thread%20Network%20Fundamentals_v3.pdf.
- [35] *UM1718 - STM32CubeMX for STM32 configuration and initialization C code generation*, maj 2022. adr.: https://www.st.com/resource/en/user_manual/dm00104712-stm32cubemx-for-stm32-configuration-and-initialization-c-code-generation-stmicroelectronics.pdf.
- [36] *UM2243 - STM32 Nucleo expansion board for power consumption measurement*, mar. 2018. adr.: https://www.st.com/resource/en/user_manual/um2243-stm32-nucleo-expansion-board-for-power-consumption-measurement-stmicroelectronics.pdf.
- [37] Wooley, M., *Bluetooth Mesh Models - Technical Overview*, mar. 2019. adr.: https://www.bluetooth.com/wp-content/uploads/2019/04/1903_Mesh-Models-Overview_FINAL.pdf (term. wiz. 26.05.2022).
- [38] Woolley, M., *Bluetooth Core Specification v5.1*, 2020.
- [39] ——, *Bluetooth Core Specification Version 5.2 Feature Overview*, 2020.
- [40] ——, *Bluetooth Mesh Networking - An Introduction for Developers*. Grud. 2020.
- [41] ——, *Bluetooth® Core Specification Version 5.3 Feature Enhancements*, 2021.
- [42] ——, „*Bluetooth® Core Specification Version 5.0 Feature Enhancements*,” en, s. 22,

Bibliografia

- [43] Yi Lim, N. C., Yong, L., Su, H. T., Yu Hao Chai, A., Vithanawasam, C. K., Then, Y. L. i Siang Tay, F., „Review of Temperature and Humidity Impacts on RF Signals,” w *2020 13th International UNIMAS Engineering Conference (EnCon)*, Kota Samarahan, Malaysia: IEEE, paź. 2020, s. 1–8, ISBN: 978-1-72819-293-2. DOI: 10.1109/EnCon51501.2020.9299327. adr.: <https://ieeexplore.ieee.org/document/9299327/> (term. wiz. 01.06.2022).
- [44] ZigBee Alliance, „ZigBee Specification,” en, s. 599, 2017.

Wykaz skrótów i symboli

ACK *Acknowledgement* 14

AoA Kąt Przyjścia (z ang. *Angle of Arrival*) 20

AoD Kąt Przyjścia (z ang. *Angle of Departure*) 20

API Application Programming Interface 28, 46

APL warstwa aplikacji (z ang. *Application Layer*) 15

APS Podwarstwa wsparcia aplikacji (z ang. *Application Support Sublayer*) 15

AT Zbiór poleceń Hayes'a (znany jako zbiór polecen AT) 40, 45, 63

ATT Protokół Atrybutów (z ang. *Attribute Protocol*) 20

BER Bit Error Rate 23

BLE Bluetooth Low Energy 19, 20, 24, 26–28, 32, 50, 51, 55

BMS Battery Management System 35

BT Bluetooth 19, 20, 70

CAD Computer-aided Design 35

EATT Ulepszony Protokół Atrybutów (z ang. *Enhanced Attribute Protocol*) 20

ED Urządzenie końcowe (z ang. *End Device*) 17, 19

FDM Fused Deposition Modeling 35

FED Pełne urządzenie końcowe (z ang. *Full End Device*) 18

FFD Urządzeniem w Pełni Funkcjonujące (z ang. *Full Functioning Device*) 15

- FFF** Fused Filamnet Fabrication 35
- FTD** Pełne urządzenie Thread (z ang. *Full Thread Device*) 18
- HRT** Heart Rate 31, 37, 51, 70
- ICMP** Internet Control Message Protocol 18
- IDE** Integrated Development Environment - Zintegrowane środowisko programistyczne 31
- IoT** Internet Rzeczy (z ang. *Internet of Things*) 10, 13, 17
- IP** Internet Protocol 19
- ISM** Industrial, Scientific, Medical 59
- LED** Light Emitting Diode (dioda emitująca światło) 43
- LL** Link Layer 28
- LPN** Low Power Node 22, 26
- LQI** wskaźnik jakości łącza (z ang. *Link Quality Indication*) 14
- MAC** warstwę sterowaną dostępem do medium transmisyjnego (z ang. *Medium Access Control*) 14
- MED** Minimalne urządzenie końcowe (z ang. *Minimal End Device*) 18
- MTD** Minimalne urządzenie Thread (z ang. *Minimal Thread Device*) 18
- NWK** warstwa sieciowa ZigBee (z ang. *ZigBee network layer*) 15
- OSI** Open Systemns Interconnection model 5, 10, 11
- PAN** bezprzewodowa sieć o zasięgu osobistym (z ang. *Personal Area Network*) 15
- PCB** Printed Circuit Board 35
- PER** Packet Error Rate 11, 23, 28, 29, 31, 32, 38, 58, 62–67, 70
- PHY** warstwę fizyczną (z ang. *Physical Layer*) 14
- PLA** Polylactic Acid 35
- REED** Urządzenie z możliwością działania jako router (z ang. *Router Eligible End Device*) 18
- RFD** Urządzenia o Zredukowanej Funkcjonalności (z ang. *Reduced Function Device*) 15

RIP Routing Information Protocol 19

SED Usypiane urządzenie Thread (z ang. *Sleepy End Device*) 18

SoC system na chipie (z ang. *System on a Chip*) 10

TCP Transport Control Protocol 19

TTL Czas Życia *Time-to-live* 22, 23

UART Universal asynchronous receiver-transmitter 44

UDP User Datagram Protocol 18

XML Extensible Markup Language 46

ZC Koordynator ZigBee (z ang. *ZigBee Coordinator*) 16

ZDO Obiekt urządzenia ZigBee (z ang. *Zigbee Device Object*) 15

ZED Urządzenie końcowe Zigbee (z ang. *Zigbee End Device*) 16

ZR Router ZigBee (z ang. *Zigbee Router*) 16

Spis rysunków

1	Architektura stosu ZigBee. Źródło: [44]	14
2	Topologie sieci ZigBee.	15
3	Zestawienie warstw stosu ZigBee z modelem referencyjnym OSI. Źródło: [27]	16
4	Stos Thread i odpowiadające mu specyfikacje RFC/IEEE. Źródło: [8]	17
5	Podstawowa topologia sieci Thread wraz z urządzeniami. Źródło: [34]	18
6	Thread w zestawieniu z modelem OSI. Źródło: [8]	19
7	Lokalizowanie kierunku dla standardu Bluetooth 5.1.. Źródło: [38]	20
8	Przykładowa topologia BT Mesh z uwzględnieniem rodzajów węzłów. Źródło: [11] . .	21
9	Zestawienie stosu BLE i modelu ISO OSI. Źródło: [2]	22
10	Porównanie przepustowości w zależności od ilości przeskóków 100-bajtowej (96-baj dla BT) pakietu podczas transmisji. Źródło: [1]	23
11	Porównanie opóźnienia w zależności od wielkości pakietu dla 4-węzłowej sieci różnych standardów komunikacji. Źródło: [1]	24
12	Zależność konsumpcji energii od mocy transmisji danych w zależności od wielkości pakietu i rodzaju protokołu. Źródło: [5]	24
13	Symulacja zużycia energii przez urządzenie BLE podczas ogłoszania	26
14	Symulacja zużycia energii przez urządzenie BLE podczas połączenia	26
15	Model referencyjny ISO OSI wraz i odpowiadające mu nazwy porcji danych. Źródło: [16]	28
16	Zestaw uruchomieniowy P-NUCLEO-WB55	33
17	Zestaw pomiarowy X-NUCLEO-LPM01A	34
18	Odbiornik energii	35
19	Model obudowy dla zestawu P-NUCLEO-WB55	36
20	Obudowa zestawu uruchomieniowego Nucleo wykonana w technologii druku 3D – realizacja	37
21	Wykorzystanie interfejsu szeregowego do wysyłania poleceń. Źródło: [20]	45
22	Selektor portów szeregowych w aplikacji służącej eksperymentowi PER.	46
23	Selektor portów szeregowych w aplikacji służącej eksperymentowi PER.	47
24	Właściwy interfejs użytkownika do przeprowadzenia doświadczenia PER	49

25	Podłączony zestaw pomiarowy	51
26	Przykładowa sesja pomiarowa dla BLE Mesh - sieć w trybie nasłuchującym	51
27	Charakterystyka czasowa poboru prądu dla BLE i usługi Heart Rate - Usługa Połączona	52
28	Charakterystyka czasowa poboru prądu dla BLE i usługi Heart Rate - Tryb szybkiego ogłoszania	53
29	Charakterystyka czasowa poboru prądu dla BLE i usługi Heart Rate - Tryb niskomocowego ogłoszania	53
30	Zestawienie zużycia prądu dla usługi Heart Rate w zależności od trybu działania	54
31	Charakterystyka czasowa poboru prądu dla BLE Mesh i modelu Generic OnOff	55
32	Rzeczywista charakterystyka poboru energii dla BLE Mesh działającego w trybie gotowości	56
33	Zestawienie zużycia prądu dla BLE Mesh w zależności od trybu działania	56
34	Porównanie średniego zużycia energii pomiędzy BT Low Energy HRT i BLE Mesh	57
35	Fotografia miejsca badań w Kampinoskim Parku Narodowym	59
36	Fotografia miejsca badań na Polach Mokotowskich	59
37	Dystans pomiędzy węzłami dla sieci dwóch mikrokontrolerów	60
38	Dystans pomiędzy węzłami dla sieci trzech mikrokontrolerów	61
39	Zależność PER od dystansu dla zapytań o częstotliwość $\leq 100\text{ms}$ dla różnej liczby węzłów	62
40	Zależność PER od dystansu dla zapytań o częstotliwość $\leq 100\text{ms}$ w wybranych środowiskach bez rozróżnienia na liczbę węzłów	63
41	Zależność PER od dystansu dla zapytań o częstotliwość $\leq 100\text{ms}$ w wybranych środowiskach i liczbie badanych węzłów	64
42	Zależność PER od dystansu dla zapytań o częstotliwość $>100\text{ms}$ w wybranych środowiskach	64
43	Zależność PER od dystansu dla zapytań o częstotliwość $>100\text{ms}$ dla różnej liczby węzłów	65
44	Zależność PER od dystansu dla zapytań o częstotliwość $>100\text{ms}$ w wybranych środowiskach bez rozróżnienia na liczbę węzłów	66
45	Zależność PER od dystansu dla zapytań o częstotliwość $>100\text{ms}$ w wybranych środowiskach i liczbie badanych węzłów	67

Spis tablic

1	Zestawienie protokołów Bluetooth Mesh, Thread, ZigBee	25
2	Zebrane empirycznie współczynniki kalibracyjne w porównaniu z wartościami obliczonymi	48
3	Współczynnik determinacji dla zależności interwału zapytań od dystansu i PER w wybranych środowiskach	65

Spis załączników

1 Kod źródłowy	87
-------------------------	----

Załącznik 1

Kod źródłowy

Wraz z pracą dyplomową załączone są wszystkie pliki związane z projektem, włączając w to kod źródłowy.

Publiczne repozytorium dostępne jest na platformie GitHub pod poniższym linkiem:

- https://github.com/krkruk/stm32wb_mesh_packet_error_rate