Implementacja stosu komunikacji w zasilanych bateryjnie urządzeniach NB-IoT

Piotr Szydłowski, Karol Zaręba
Efento Sp. z o.o., ul. Krupnicza 14/5, 31-123 Kraków

Streszczenie: W artykule zaprezentowano sposób implementacji wydajnego stosu komunikacji dla bezprzewodowych, zasilanych bateryjnie rejestratorów NB-IoT. Na podstawie doświadczeń zgromadzonych przy projektowaniu i rozwoju oprogramowania rejestratorów, wykonano analizy mające na celu wybór odpowiednich protokołów komunikacji, formatu serializacji danych, konfiguracji modułu NB-IoT oraz dodatkowe algorytmy opracowane w celu optymalizacji liczby transmisji i ilości przesyłanych danych. Przedstawiono również uzyskane wyniki wraz z analizą wpływu poszczególnych parametrów konfiguracji na czas życia baterii urządzenia i zużycie danych.

Słowa kluczowe: NB-IoT, Internet Rzeczy, CoAP, Protobuf, stos komunikacji, rejestrator, zasilanie bateryjne

1. Wprowadzenie

Obecnie znacząco wzrasta rola zdalnego nadzoru za pomocą różnego rodzaju urządzeń Internetu Rzeczy, IoT (ang. *Internet of Things*) [1]. Coraz bardziej istotne staje się opracowanie niezawodnych sposobów komunikacji dopasowanych do specyfiki tych urządzeń. Po latach testowania różnych modeli i technik komunikacji urządzeń Internetu Rzeczy coraz większą popularność zyskuje model wykorzystujący wydzielone i licencjonowane sieci, przede wszystkim Narrowband IoT (NB-IoT) [2]. Wynika to między innymi z niskich cen modułów komunikacyjnych i kosztów utrzymania urządzenia, coraz większej dostępność sieci (dostęp do sieci NB-IoT jest oferowany przez 168 operatorów na całym świecie [3]) oraz trafnych założeń przy projektowaniu standardu, które umożliwiają producentom urządzeń uzyskanie bardzo długiego czasu życia baterii.

W artykule przedstawiono szereg praktycznych wskazówek projektowych oraz wykonanych analiz, które pozwoliły na opracowanie optymalnych metod komunikacji dla zasilanych bateryjnie urządzeń NB-IoT. Zdobyte doświadczenia zostały zebrane w trakcie projektowania i wdrożenia do produkcji rejestratorów bezprzewodowych NB-IoT, a następnie analizę danych dotyczących ich pracy. Do tej pory zainstalowano ponad 50 000 rejestratorów, które każdego dnia przesyłają do serwerów ponad 24 miliony pomiarów.

Artykuł składa się z kilku sekcji. W Sekcji 2 omówiono standard Narrowband IoT. W Sekcji 3 zaprezentowano podstawową charakterystykę i konstrukcję rejestratora Efento. W Sekcji 4 zawarto opis kluczowych problemów przy projektowaniu metod komunikacji zasilanych bateryjnie urządzeń NB-IoT (dobór protokołów warstwy transportowej i aplikacyjnej, sposób serializacji danych, konfigurację modułu NB-IoT oraz pozostałe algorytmy optymalizujące liczbę transmisji i wielkość przesyłanych ramek). W Sekcji 5 przedstawiono i omówiono uzyskane wyniki.

Autor korespondujący:
Piotr Szydłowski, piotr.szydlowski@efento.pl

Artykuł recenzowany
nadesłany 06.06.2023 r., przyjęty do druku 16.08.2023 r.

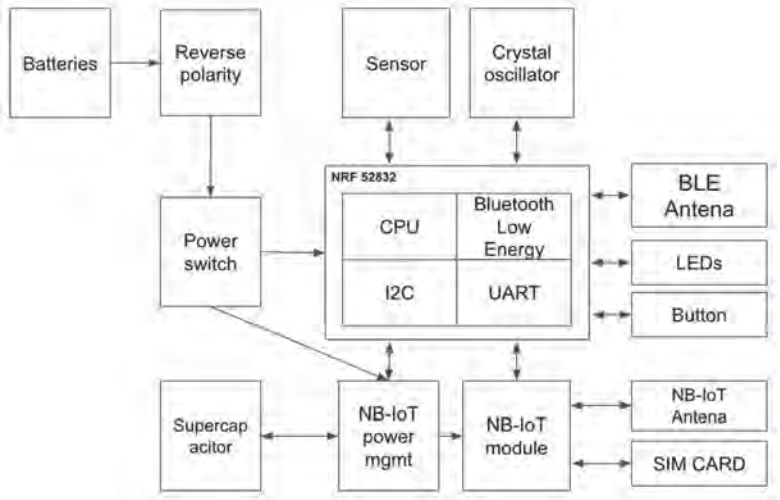
2. Narrowband IoT (NB-IoT)

Narrowband Internet of Things to jeden ze standardów komunikacji radiowej Low Power Wide Area Network (LPWAN), opracowany przez 3GPP i po raz pierwszy udostępniony w 2016 r. (3GPP Release 13 [4]). Główne założenia uwzględnione podczas projektowania standardu komunikacji to:

- zapewnienie łączności urządzeniom o niskiej złożoności (sensory bezprzewodowe, liczniki energii elektrycznej, trackery GPS),
- zminimalizowanie zużycia energii przez urządzenia i zapewnienie im długiego czasu życia baterii,
- zapewnienie zasięgu sieci w trudno dostępnych miejscach (np. piwnice, parkingi podziemne),
- zapewnienie dużej gęstości połączeń, umożliwiając tym samym instalację dużej liczby urządzeń IoT w obrębie pojedynczej stacji bazowej,
- zminimalizowanie kosztów, zarówno po stronie operatorów wdrażających NB-IoT w swoich sieciach, jak i użytkowników urządzeń.

Standard NB-IoT powstał w dużej mierze w oparciu o technologię LTE, zachowując schematy kodowania, modulacji kanałów oraz protokoły wyższych warstw. Dzięki temu wdrażanie NB-IoT w istniejących sieciach LTE jest szybkie i łatwe z punktu widzenia operatora. Aby zmniejszyć złożoność urządzeń NB-IoT (a tym samym obniżyć ich koszty), zrezygnowano z mechanizmów stosowanych w LTE, które nie są użyteczne w urządzeniach IoT. Wprowadzono natomiast dodatkowe funkcje, umożliwiające wydłużenie czasu życia baterii. Do najważniejszych z nich należą extended Discontinuous Reception (eDRX) i Power Saving Mode (PSM) [5]. Dodatkowo w standardzie NB-IoT uzyskano poprawę pokrycia

Rys. 1. Rejestrator NB-IoT Efento. Wnętrze urządzenia z zamontowanym PCB (po lewej) i obudowa (po prawej)
Fig.1. Efento NB-IoT logger. PCB mounted inside the enclosure (on the left) and the enclosure (on the right)



Rys. 2. Schemat blokowy rejestratora NB-IoT Efento
Fig. 2. Block scheme of Efento NB-IoT logger

radiowego dzięki mechanizmom powtarzania transmisji. Standard NB-IoT jest wciąż rozwijany a kolejne wersje 3GPP wprowadzają jego dalsze ulepszenia [6].

Przewagą NB-IoT nad innymi standardami komunikacji LPWAN jest wykorzystanie licencjonowanego pasma komunikacji. Gwarantuje to poprawną pracę urządzeń i zmniejsza ryzyko zakłóceń między nimi.

3. Rejestratory NB-IoT Efento

Rejestratory NB-IoT Efento to urządzenia zasilane bateryjnie umożliwiające pomiary ponad 20 różnych parametrów (m.in. temperatura, wilgotność, ciśnienie atmosferyczne, poziom CO_2, wykrywanie ruchu, wykrywanie zalania). Wyposażone są w moduły komunikacji NB-IoT oraz Bluetooth Low Energy. Dzięki temu mogą komunikować się zarówno z platformami IoT/serwerami, jak i z pobliskimi urządzeniami (smartfon, tablet) w celu konfiguracji lub odczytu danych.

Urządzenia mogą współpracować z udostępnioną platformą do przechowywania i analizy danych lub mogą być zintegrowane z dowolną aplikacją serwerową, stosowaną przez użytkowników. Do głównych zastosowań rejestratorów NB-IoT Efento należą: monitoring warunków przechowywania leków i szczepionek, nadzór nad rozproszonymi obiektami, monitoring jakości powietrza w pomieszczeniach biurowych.

Obwód drukowany rejestratora (PCB) oraz jego obudowę przedstawiono na Rys. 1. Urządzenie zaprojektowano w sposób modułowy. Umożliwia to łatwe dodawanie obsług kolejnych rodzajów sensorów, bez konieczności modyfikowania części wspólnej układu elektronicznego, która zawiera moduły komunikacyjne i procesor.

Rejestratory Efento NB-IoT bazują na układzie SoC Nordic Semiconductor NRF 52832 wyposażonym w 32-bitowy, energooszczędny procesor ARM Cortex-M4, do którego podłączone są poszczególne peryferia. Transmisję NB-IoT zapewnia moduł Quectel BC66. Rejestratory zasilane są pakietem baterii składającym się z trzech baterii litowych o napięciu 3,6 V i łącznej pojemności 6300 mAh (Rys. 2).

4. Implementacja wydajnego sposobu komunikacji urządzeń pracujących w sieci NB-IoT

W przypadku zasilanych bateryjnie urządzeń pracujących w sieciach NB-IoT kluczowe jest takie zaprojektowanie stosu komunikacji, aby zapewnić:

- Jak najmniejszą wielkość przesyłanych ramek danych. Ponieważ operatorzy sieci komórkowych rozliczają się z ich użytkownikami na podstawie użycia danych, zmniejszenie ilości danych zmniejsza koszty użytkowania urządzenia. Dodatkowo sieci NB-IoT zostały zaprojektowane i zoptymalizowane do przesyłania małych ilości danych, zapewniając jednocześnie łączność dużej liczbie urządzeń zlokalizowanych w obrębie tej samej stacji bazowej (do 100 000 urządzeń). W celu zapewnienia poprawnej pracy sieci, urządzenia NB-IoT nie powinny transmitować zbyt dużej ilości danych niepotrzebnie konsumując zasoby sieci.
- Jak najkrótszy czas wysyłki/odbierania danych. Zużycie energii przez urządzenia pracujące w sieciach NB-IoT w trakcie transmisji danych (stan aktywny) jest znacznie większe w porównaniu do trybu oszczędzania energii PSM (ang. *Power Saving Mode*). Aby ograniczyć zużycie energii kluczowe jest zminimalizowanie wielkości przesyłanych ramek danych.
- Jak najmniejsze obciążenie serwera. Dane przesyłane przez urządzenia IoT trafiają do serwera, gdzie są przetwarzane. Aby ograniczyć zużycie zasobów serwera dane przesyłane przez urządzenia IoT powinny być przesyłane w formacie umożliwiającym szybkie przetwarzanie przez serwer.
- Niezawodność transmisji danych. Ponieważ urządzenia IoT często przesyłają dane, na podstawie których podejmowane są decyzje lub wyzwalane są akcje, konieczne jest, aby istotne informacje zawsze trafiły do serwera zajmującego się ich dalszym przetwarzaniem.
- Kompatybilność z popularnymi platformami IoT i łatwą możliwość integracji z aplikacjami firm trzecich. Rozwiązania budowane w oparciu o urządzenia IoT coraz częściej składają się z komponentów pochodzących od różnych dostawców. Wybrany protokół komunikacji powinien być powszechnie znany i używany, aby umożliwić użytkownikom łatwą integrację z wybraną przez nich platformą IoT/aplikacją.

Prace projektowe związane ze stosem komunikacji używanym przez bezprzewodowe rejestratory NB-IoT Efento prowadzone były etapowo. W pierwszym etapie dokonano analizy i wyboru protokołów warstw transportowej i aplikacyjnej. Następnie dobrano optymalny sposób serializacji przesyłanych przez rejestratory danych. W ostatnim etapie przeprowadzono optymalizację algorytmów kompresji i transmisji pod kątem ograniczenia ilości danych i liczby transmisji.

4.1. Wybór protokołu warstwy transportowej i aplikacyjnej

Do najpopularniejszych protokołów warstwy aplikacyjnej w IoT należą obecnie Constrained Application Protocol (CoAP) i MQ Telemetry Transport (MQTT) [7]. Oba protokoły są szeroko stosowane w rozwiązaniach IoT, a biblioteki obsługujące każdy z nich są powszechnie dostępne dla praktycznie wszystkich popularnych języków programowania. Zarówno CoAP jak i MQTT są wspierane przez szereg popularnych platform IoT.

Porównanie wydajności tych protokołów było tematem wielu analiz prowadzonych środowiskach akademickich oraz w przemysłu. Na podstawie przeprowadzonych symulacji [8] omówiono korzyści z zastosowania CoAP i MQTT w sensorach medycznych. Badacze z Ericsson, korzystając z wewnętrznych narzędzi tej firmy [9], symulują pracę 76 500 urządzeń NB-IoT, a następnie analizują wpływ protokołu na wydajność i niezawodność komunikacji.

W artykule skupiono się na porównaniu korzyści płynących z zastosowania CoAP i MQTT w zasilanych bateryjnie urządzeniach NB-IoT oraz pokazano, w jaki sposób podjęto decyzję o wyborze protokołu warstwy aplikacyjnej zastosowanego w rejestratorach Efento.

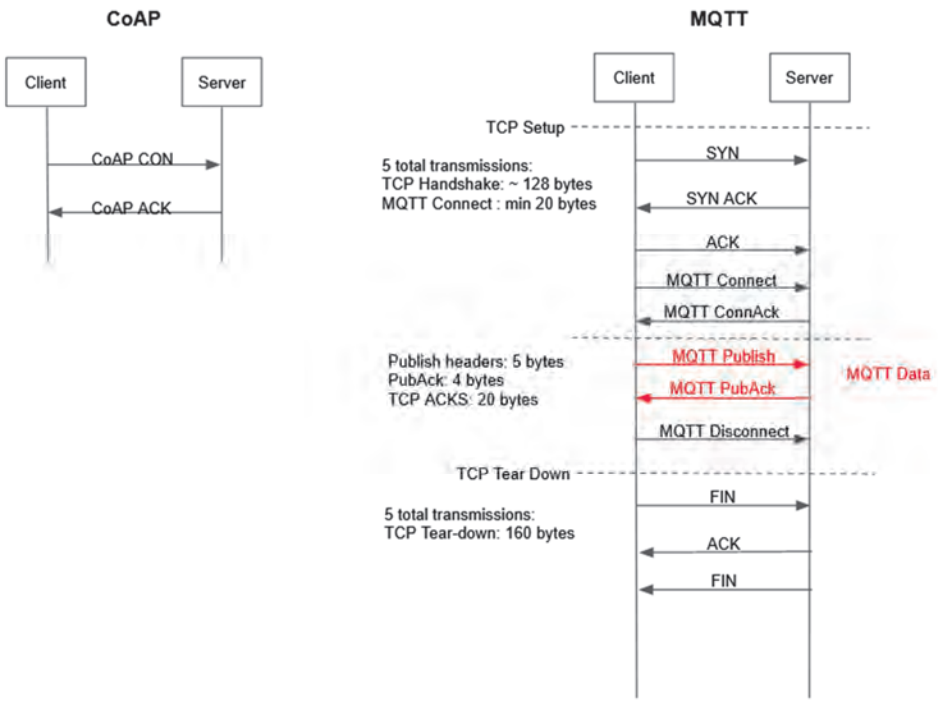
CoAP to protokół warstwy aplikacyjnej opracowany i ustandaryzowany przez IETF [10]. Protokół ten jest podobny do HTTP (używa tych samych metod: GET, POST, PUT, DELETE) i oparty jest na modelu klient/serwer. Został zoptymalizowany do zastosowania w urządzeniach o ograniczonych zasobach. CoAP najczęściej wykorzystuje UPD jako warstwę transportową ale może działać w dwóch różnych trybach: wysyłka ramki danych bez wymagania potwierdzenia otrzymania przez serwer (ramki oznaczone jako NON) lub wysyłka ramki danych z wymaganiem potwierdzenia (ramki oznaczone jako CON). Wybór typu ramki danych jest decyzją klienta i może zmieniać się między ramkami (np. co piąta ramka jest typu CON, pozostałe NON). Protokół CoAP charakteryzuje się dodatkowo małym narzutem – minimalny rozmiar nagłówka protokołu to cztery bajty (wielkość nagłówka może wzrosnąć w przypadku przesyłania pól opcjonalnych).

MQTT to protokół warstwy aplikacyjnej opracowany i rozwijany przez OASIS [11]. Protokół ten został zaprojektowany z myślą o urządzeniach IoT i jest oparty na modelu publikacja/subskrypcja. Każdy z klientów łączy się z brokerem, a następnie subskrybuje dany temat. Każdy z klientów może również publikować dane w wybranym temacie. Jeśli któryś z klientów opublikuje dane w wybranym temacie, każdy subskrybujący go klient otrzyma te dane. Dodatkowo MQTT realizuje mechanizmy Quality of Service (QoS), które zapewniają niezawodność komunikacji między klientami a brokerem. Najczęściej protokół MQTT wykorzystuje TCP jako warstwę transportową. Istnieje implementacja protokołu MQTT zoptymalizowana do zastosowań w urządzeniach o ograniczonych zasobach (MQTT-SN), która nie zyskała takiej popularności jak oryginalna wersja protokołu. Protokół MQTT charakteryzuje się małym narzutem – minimalna wielkość nagłówka to dwa bajty (oraz do 12 bajtów opcjonalnego nagłówka dodatkowego).

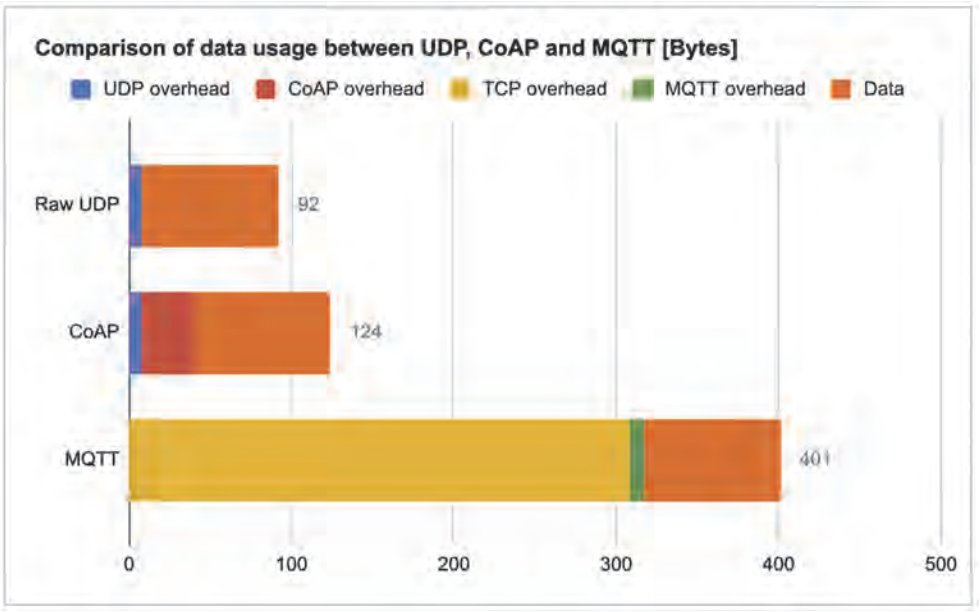
Komunikacja z użyciem protokołu CoAP nie wymaga nawiązywania i rozłączania połączenia między klientem a serwerem, co znacząco zmniejsza ilość przesyłanych danych (Rys. 3). Niezawodność transmisji zapewniona jest przez ramki wymagające potwierdzenia przez serwer (CON). Gwarantuje to, że mimo wykorzystania UDP jako protokołu warstwy transportowej dane przesyłane przez klienta na pewno zostaną otrzymane przez serwer. W przypadku braku potwierdzenia, klient ponawia transmisję.

Komunikacja z użyciem protokołu MQTT wymaga nawiązania i rozłączania połączenia zgodnie z protokołem TCP ("three-way handshake"). Wynikająca z tego konieczność wymiany dodatkowych ramek znacząco zwiększa ilość przesyłanych danych oraz wydatek energetyczny na każdą transmisję. Niezawodność zapewniona jest przez wykorzystanie protokołu TCP w warstwie transportowej oraz dodatkowe mechanizmy MQTT (w przypadku QoS1 i QoS2) – przetworzenie każdej publikacji przez klienta (MQTT Publish) potwierdzona jest przez brokera (MQTT PubAck).

W celu analizy wpływu protokołu na zużycie danych zaimplementowaliśmy testowe rozwiązanie umożliwiające przesyłanie tej samej wiadomość z wykorzystaniem CoAP i MQTT. Aby określić narzut każdego z protokołów, jako poziom odniesienia przyjęto wysyłkę danych z użyciem wyłącznie protokołu UDP. Przesyłana ramka danych w każdym przypadku była identyczna i zawierała typową wiadomość przesłaną przez rejestratory Efento – 84 bajty zawierające pakiet 20 pomiarów temperatury. Zużycie danych zaprezentowano na Rys. 4.



Rys. 3. Schemat komunikacji z wykorzystaniem protokołów CoAP i MQTT (QoS1)
Fig. 3. Communication scheme of CoAP and MQTT protocols



Rys. 4. Porównanie zużycia danych przy wysyłce pojedynczej wiadomości o wielkości 84 bajtów przy użyciu protokołów CoAP (CON), MQTT (QoS1) i "gołego" UPD
Fig. 4. Comparison of data usage when sending a single message of 84 bytes using raw UDP, CoAP (CON) and MQTT (QoS1) protocols

Wielkość ramki wysyłanej przy użyciu protokołu CoAP jest większa o 31 bajtów (33 %), a w przypadku MQTT o 309 bajtów (336 %) w stosunku do "gołego" UDP. Tak duża różnica wynika przede wszystkim z opisanego narzutu protokołu TCP, przez który wykorzystanie protokołu MQTT do wysyłki małych ilości danych, w regularnych (stosunkowo dużych) odstępach czasu jest nieefektywne. Protokół MQTT ma wbudowany mechanizm podtrzymania połączenia (MQTT Pingreq/MQTT Pingresp), który eliminuje konieczność zestawienia połączenia TCP przed każdą wysyłką danych, ale użycie tego mechanizmu jest nieefektywne w przypadku bateryjnie zasilanych urządzeń pracujących w sieciach NB-IoT. Urządzenie musiałoby stale utrzymywać połączenie TCP z serwerem, co uniemożliwiałoby przejście do trybu oszczędzania energii i skutkowałoby znacznym skróceniem czasu życia baterii.

Wykorzystanie protokołu CoAP wprowadza stosunkowo niewielki narzut w stosunku do "gołego" UDP, zapewniając jednocześnie duże korzyści – niezawodność komunikacji (ponowna wysyłka danych w przypadku, gdy urządzenie nie otrzymało potwierdzenia z serwera) oraz łatwą integrację z aplikacjami serwerowymi.

Z opisanych powodów w rejestratorach Efento zdecydowano się zastosować protokół CoAP. Analiza korzyści poszczególnych protokołów komunikacji powinna być wykonana każdorazowo, biorąc pod uwagę specyfikę danego rozwiązania. W niektórych przypadkach (np. urządzenie mające dostęp do zasilania i wysyłające większe ilości danych w ramach pojedynczej transmisji) protokół MQTT (lub inne protokoły warstwy aplikacyjnej) jest lepszym rozwiązaniem niż wybrany przez nas protokół CoAP [12].

4.2. Format i serializacja danych

Istotnym aspektem mającym wpływ na wielkość przesyłanych danych jest wybór odpowiedniego formatu serializacji danych. Projektując sposób komunikacji przeanalizowano następujące aspekty:

- Wielkość danych i szybkość przetwarzania – w celu zapewnienia niskich kosztów obsługi urządzeń NB-IoT oraz długiego czasu życia baterii, sposób serializacji powinien zapewniać minimalizację wielkości ramek danych.
- Szybkość serializacji i deserializacji – wybrana metoda powinna umożliwić minimalizację zasobów niezbędnych do wykonania procesu zarówno po stronie urządzenia, jak i po stronie serwera.
- Możliwości integracji – wybrany format danych powinien być powszechnie stosowany.
- Wersjonowanie – wybrany sposób serializacji danych powinien zapewniać łatwą możliwość dodania/usunięcia pól do ramek danych.

Formaty danych można podzielić na dwie grupy: formaty tekstowe i formaty binarne. Analiza formatów serializacji pod kątem wielkości danych oraz szybkość ich przetwarzania przeprowadzona była wielokrotnie. Najważniejsze wnioski z przeprowadzonych badań:

- Wielkość danych w formatach binarnych jest zdecydowanie mniejsza niż w formatach tekstowych. Różnica ta może być nawet ośmiokrotna [13].
- Czas serializacji/deserializacji danych jest zdecydowanie krótszy w przypadku formatów binarnych. Różnica ta może być nawet dwudziestokrotna [14].
- Narzut wynikający ze specyfiki formatów tekstowych (metadane) jest bardzo duży w stosunku do formatów binarnych. Jedynie 16 % wiadomości JSON to właściwe dane, pozostałe 84 % to metadane. Jeśli ta sama ramka danych została serializowana jako Protobuf, stosunek ten uległ zmianie do 71 % – dane, 29 % – metadane [15].

Binarne formaty danych są znacznie lepszym wyborem w przypadku zasilanych bateryjnie urządzeń IoT. Minimalizacja wielkości danych oraz krótszy czas serializacji niosą korzyści zarówno po stronie urządzeń (mniejsze zużycie energii, a tym

Tabela 1. Porównanie serializacji danych z wykorzystaniem Protobuf i MessagePack
Table 1. Comparison of data serialisation with Protobuf and MessagePack

	Wielkość ramki (bajty)	Średni czas serializacji (µs)	Średni czas deserializacji (µs)
Protobuf	84	< 1	< 1
MessagePack	146	< 1	< 1

samym dłuższy czas życia baterii), jak i platform IoT (mniejsze zasoby potrzebne do przetworzenia danych pochodzących z dużej liczby urządzeń).

Do najpopularniejszych binarnych formatów serializacji danych należą Protobuf [16] i MessagePack [17]. Dla porównania ich efektywności wykonano test polegający na serializacji i deserializacji paczki 20 pomiarów temperatury wykonanych przez rejestrator Efento.

Czasy serializacji i deserializacji w obu przypadkach są na tyle małe, że nie mają istotnego wpływu na pracę urządzenia. Wielkość otrzymanej paczki danych jest natomiast aż o 62 bajty mniejsza niż w przypadku zastosowania Protobufa. Dodatkową przewagą wybranego sposobu serializacji jest rozpoznawalność (format został opracowany i udostępniony przez Google) oraz implementacja praktycznie w każdym języku programowania. Protobuf umożliwia zachowanie kompatybilności między wersjami oprogramowania rejestratorów – jeżeli w nowej wersji urządzenie wysyła dodatkowe informacje (pole), to jest ono dodawane do schematu wiadomości z zachowaniem kompatybilności wstecznej. Pliki .proto, określające strukturę przesyłanych przez rejestratory danych, dostępne są w repozytorium [22].

Obecnie istnieje znacznie więcej binarnych formatów serializacji niż te opisane powyżej (np. BSON [18] czy Apache Avro [19]). Wybierając sposób serializacji należy brać pod uwagę specyfikę danego rozwiązania i przeprowadzić testy, które pozwolą określić, jaki format najlepiej pasuje do konkretnej struktury danych.

4.3. Optymalizacja algorytmów kompresji i transmisji danych

4.3.1. Mechanizm kompresji danych

Mechanizmem, który umożliwił znacząco zmniejszyć wielkość przesyłanych ramek jest algorytm kompresji danych. Pomiary przesyłane są przez rejestratory w postaci paczek danych. Podczas przygotowywania danych do wysyłki rejestrator oblicza wartość średnią ze wszystkich pomiarów z danej paczki ("wartość startowa"), a następnie przesyła do serwera tę wartość wraz z różnicami między nią, a kolejnymi pomiarami. Ponieważ rejestratory mierzą wielkości fizyczne niepodlegające szybkim zmianom, różnice w pomiarach w obrębie jednej paczki są małe i mogą być zapisane na mniejszej liczbie bajtów niż pełne wartości pomiarów. Ten prosty algorytm kompresji danych pozwala zredukować o kilka procent wielkość przesyłanych danych. Wyniki działania algorytmu dla paczek danych składających się z 5 i 10 pomiarów wilgotności zaprezentowano w Tabeli 2.

Przesłana przez rejestrator paczka danych zawiera jedynie znacznik czasu wartości startowej oraz informację o okresie pomiaru, który jest stały w obrębie każdej paczki. Znaczniki czasów poszczególnych pomiarów obliczane są przez serwer, po otrzymaniu danych. To podejście pozwala znacząco zmniejszyć wielkość paczek danych, oszczędzając 4 bajty na każdy pomiar.

4.3.2. Optymalizacja algorytmów transmisji

W przypadku urządzeń NB-IoT znaczna część energii zużywana jest na transmisję danych. Z tego względu, projektując mechanizmy i algorytmy mające na celu wydłużenie czasu życia baterii skupiliśmy się na ograniczeniu liczby transmisji (im rzadziej urządzenie transmituje dane, tym dłuższy czas życia baterii) oraz skróceniu czasu w stanie aktywnym (im krócej urządzenie pozostaje w stanie aktywnym, tym dłuższy jest czas życia baterii).

4.3.3. Zminimalizowanie liczby transmisji

Pierwszym mechanizmem, który ogranicza liczbę transmisji jest wysyłka pomiarów w paczkach. Konfigurując urządzenie użytkownik może ustawić dwa niezależne od siebie parametry: okres pomiaru i okres transmisji. Jeśli dostęp do pomiarów w trybie rzeczywistym nie jest potrzebny, należy wykonywać transmisję rzadziej, z zachowaniem, by każda z wysyłanych paczek danych zawierała wszystkie pomiary wykonane od momentu ostatniej komunikacji z serwerem.

Takie podejście powoduje opóźnienie w transmisji danych, co wpływa na czas reakcji osoby odpowiedzialnej za utrzymanie prawidłowych warunków monitorowanego obiektu. Jeżeli pomiar wykonany przez rejestrator ma wartość leżącą poza bezpiecznym zakresem (np. temperatura w lodówce jest poza zakresem zdefiniowanym przez producenta przechowywanych w niej szczepionek), serwer powinien otrzymać tę informację jak najszybciej. Aby rozwiązać ten problem, przeniesiono część logiki związanej z analizą danych pomiarowych z serwera do rejestratora.

Algorytmy analizy danych w rejestratorze wykorzystują silnik reguł. Użytkownik może konfigurować szereg warunków, które analizowane są po każdym pomiarze/serii pomiarów. W zależności od wyników analizy, rejestrator podejmuje decyzję o wyzwoleniu transmisji ("Report by Exception"). Najprostszy przykład reguły "jeżeli temperatura wzrośnie powyżej określonej wartości, natychmiast wyślij dane do serwera". Opracowany silnik reguł daje użytkownikowi duże możliwości konfiguracji, uwzględniając m.in.:

- Różne rodzaje progów:
  - Próg dolny – jeżeli wartość pomiaru spadnie poniżej wartości progu,
  - Próg górny – jeżeli wartość pomiaru wzrośnie powyżej progu,
  - Próg różnicowy – jeżeli różnica między pomiarem bieżącym a ostatnim wysłanym do serwera jest większa niż wartość progu,
- Mechanizmy niwelowania fluktuacji:
  - Histereza – ograniczenie niepotrzebnych transmisji, gdy pomiary oscylują w okolicy wartości progu,
  - Średnia krocząca – w celu eliminacji krótkich fluktuacji, porównując wartości pomiaru do ustalonego progu urządzenie może brać pod uwagę wartość średnią z ostatnich N pomiarów,
  - Opóźnienie – ograniczenie transmisji w przypadku chwilowych fluktuacji – wartość pomiaru musi przekroczyć wartość progu i pozostać powyżej/poniżej przez określony czas,
- Operatory logiczne AND, OR, NOT, które umożliwiają budowę złożonych reguł, uwzględniających więcej niż jedną mierzoną przez rejestrator wartość (np. „jeżeli temperatura wzrośnie powyżej progu i wilgotność spadnie poniżej progu", natychmiast wyślij dane do serwera),
- Kalendarze reguł – ograniczenie analizy danych pomiarowych jedynie do wybranych dni tygodnia i godzin.

Tabela 2. Wpływ algorytmu kompresji na wielkość ramki danych
Table 2. Impact of the compression algorithm on the data frame size

Pomiary [%RH]	Wielkość ramki danych bez kompresji (bajty)	Wielkość ramki danych z kompresją (bajty)
[0, 45, 47, 50, 45]	96	90
[0, 45, 47, 50, 45, 43, 49, 50, 45, 42]	114	106

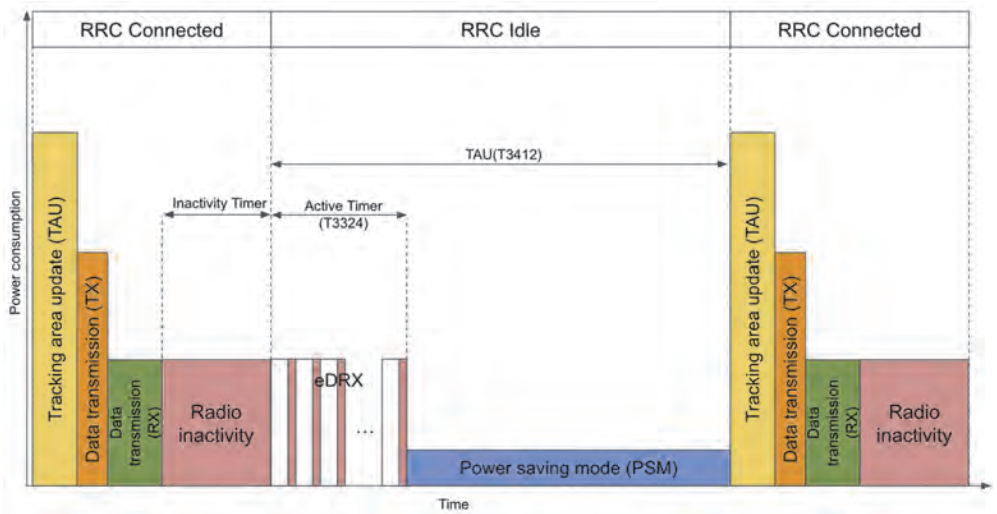
4.3.4. Skrócenie czasu transmisji

W przypadku urządzeń NB-IoT kluczowe jest wykorzystanie mechanizmów oszczędzania energii oferowanych przez tę technikę komunikacji. Analiza ich wpływu na zużycie energii przez urządzenia NB-IoT w warunkach laboratoryjnych wykonywana była wielokrotnie [20]. Typowy schemat komunikacji urządzeń NB-IoT zaprezentowano na Rys. 5. Po wybudzeniu i włączeniu radia urządzenie inicjuje komunikację z siecią (TAU), następnie przesyła dane (Data transmission (TX)) oraz otrzymuje odpowiedzi z serwera (Data transmission (RX)). Po zakończeniu komunikacji urządzenie pozostaje połączone z siecią przez zdefiniowany okres czasu (Inactivity timer). Następnie urządzenie nasłuchuje dane przychodzące z serwera nie utrzymując pełnego połączenia sieciowego (eDRX). W tym trybie zużywa mniej energii niż w przypadku stanu aktywnego (RRC connected – pełne połączenie sieciowe), ale więcej niż w przypadku pracy w trybie oszczędzania energii (PSM – brak komunikacji z siecią). Po upływie czasu zdefiniowanego przez Active Timer (T3324) urządzenie przechodzi do trybu oszczędzania energii (PSM), w którym zużycie energii jest minimalne.

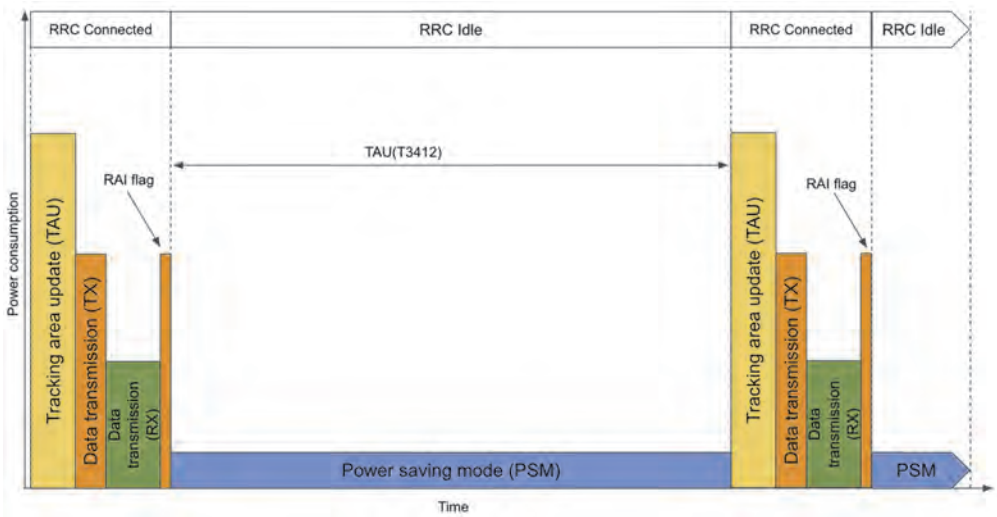
W celu skrócenia czasu transmisji, a tym samym wydłużenia życia baterii, w oprogramowaniu rejestratorów Efento wykorzystano oferowaną przez standard NB-IoT funkcję Release Assistance. Przez wysłanie flagi Release Assistance Indicator (RAI), urządzenia NB-IoT mogą informować sieć, że transmisja danych jest zakończona, zasoby radiowe mogą być zwolnione, a urządzenie przechodzi do trybu oszczędzania energii. W naszym oprogramowaniu używamy tej flagi po wysłaniu każdej ramki niewymagającej potwierdzenia przez serwer (NON), jak również po otrzymaniu odpowiedzi z serwera w przypadku transmisji ramki wymagającej potwierdzenia (CON). Gdyby rejestratory nie używały flagi RAI, pozostawałyby w stanie aktywnym do momentu zwolnienia zasobów przez sieć (Inactivity Timer). Ta optymalizacja ma znaczący wpływ na zużycie energii w trakcie transmisji danych – przykładowo w polskiej sieci T-Mobile wykorzystanie flagi RAI skraca czas o około 6 s, w którym urządzenie pozostaje połączone z siecią.

Dodatkowo całkowicie zrezygnowano z użycia eDRX. Po wysłaniu flagi RAI rejestrator natychmiast przechodzi w tryb oszczędzania energii. W naszym przypadku komunikacja z serwerem jest zawsze inicjowana przez urządzenie, a ewentualne dane (np. nowa konfiguracja) są przesyłane do niego w odpowiedzi na ramki typu CON. Umożliwia to całkowite wyłączenie eDRX (Active Timer ustawiony na 0), minimalizując w ten sposób zużycie baterii podczas transmisji (Rys. 6).

Inny sposób skrócenia czasu transmisji jest możliwy dzięki wykorzystaniu protokołu CoAP i wbudowanego w niego mechanizmu wymagania (lub nie) potwierdzenia otrzymania ramki przez serwer. W przypadku wysyłki ramki wymagającej potwierdzenia, urządzenie musi ponad dwukrotnie dłużej pozostać w stanie aktywnym, czekając na odpowiedź serwera – średni czas w trybie aktywnym dla ramki CON: 7,9 s, dla ramki typu NON: 3,6 s (pomiary wykonano przy następujących parametrach sygnału: RRSP – 65 dBm, RSRQ – 4 dBm, RSSI – 61 dBm, ECL 0). Zaprojektowany algo-



Rys. 5. Typowy schemat komunikacji urządzeń NB-IoT
Fig. 5. Typical scheme of NB-IoT devices communication



Rys. 6. Schemat komunikacji rejestratora NB-IoT Efento
Fig. 6. Scheme of Efento NB-IoT logger communication

rytm komunikacji umożliwia użytkownikom określić, jak często urządzenie ma wysyłać dane korzystając z ramek CON. Pozwala to dostosować pracę rejestratora do potrzeb użytkownika. Jeśli otrzymanie wszystkich pomiarów przez serwer nie jest kluczowe, użytkownik może ustawić rejestrator tak, że ramka typu CON wysyłana jest raz na dobę, wydłużając tym samym czas życia baterii. W celu zapewnienia niezawodności transmisji w istotnych momentach (np. przekroczenie ustalonego progu alarmowego), kluczowe dane zawsze wysyłane są za pomocą ramek CON i powtarzane w przypadku braku potwierdzenia otrzymania danych przez serwer.

W celu skrócenia czasu transmisji zrezygnowano z wykorzystania eDRX oraz skorzystano z funkcji Release Assistance. Teraz rejestrator przechodzi do trybu oszczędzania energii natychmiast po zakończeniu transmisji. Dodatkowo użytkownicy mogą skonfigurować typ wysyłanych ramek danych (CON/NON). W ten sposób udało się skrócić czas, w którym urządzenie pozostaje połączone z siecią o około 10 s, co znacząco wpływa na żywotność baterii.

5. Wyniki

Opracowane rejestratory zostały poddane testom w celu zbadania i określenia parametrów kluczowych dla zasilanych bateryjnie urządzeń NB-IoT, jak niezawodności komunikacji, czas życia baterii oraz zużycia danych. Badania przeprowadzono w sieci NB-IoT o dobrym zasięgu (RRSP – 65 dBm, RSRQ – 4 dBm, RSSI – 61 dBm, ECL 0).

5.1. Niezawodność komunikacji

Niezawodność komunikacji została zbadana dla dwóch konfiguracji urządzeń:

1. Żadna z ramek wysyłanych przez urządzenie nie wymaga potwierdzenia otrzymania od serwera (są ramkami typu NON),
2. Wszystkie ramki wysyłane przez urządzenie wymagają potwierdzenia otrzymania od serwera (są ramkami typu CON).

Oba urządzenia wysłały do serwera 10 000 ramek danych. Za sukces uznano każdorazowe otrzymanie przez serwer ramki danych. W przypadku ramek wymagających potwierdzenia rejestrowano dodatkowo, w której próbie ramka została dostarczona do serwera (Tab. 3).

W przypadku ramek niewymagających potwierdzenia, sukcesem zakończyło się 99,95 % transmisji. Utrata 0,05 % ramek danych wynika ze sporadycznych problemów z siecią NB-IoT. W przypadku ramek wymagających potwierdzenia, sukcesem zakończyło się 100 % transmisji. Wzrost zużycia energii wynikający z retransmisji jest pomijalny w stosunku do wydatku energetycznego na transmisję wszystkich 10 000 ramek danych. Obie konfiguracje zapewniają zadowalającą niezawodność, a dzięki możliwości wyboru typu ramek danych, użytkownik może balansować między kompletnością pomiarów a żywotnością baterii.

5.2. Czas życia baterii

W celu określenia uzyskanego czasu życia baterii wykonaliśmy pomiary zużycia energii przez rejestrator temperatury i wilgotności w poszczególnych fazach jego pracy. Pomiary dla każdej fazy wykonywano 30 razy, a następnie obliczano z nich wartość średnią (Tabela 4). Wykres poboru prądu w poszczególnych fazach pracy rejestratora zaprezentowano na Rys. 7.

Na podstawie danych pomiarowych opracowano narzędzie umożliwiające symulację czasu życia baterii z uwzględnieniem wpływu jakości sygnału NB-IoT [17]. Przykładowe czasy życia baterii prezentuje Tabela 5.

Otrzymane wyniki wskazują, że przy odpowiedniej konfiguracji czas życia baterii rejestratorów temperatury i wilgotności NB-IoT Efento może wynieść nawet 11 lat. Istotny wpływ na czas życia baterii mają przede wszystkim okres transmisji i typ wysyłanych ramek. W przypadku rejestratora temperatury i wilgotności okres pomiaru nie ma istotnego wpływu na czas życia baterii. Ponadto czas życia baterii znacząco skraca się w przypadku gorszego zasięgu sieci. Wynika to ze specyfiki standardu NB-IoT: moduł NB-IoT automatycznie dostosowuje moc nadawania do parametrów sygnału (im gorszy sygnał tym większa moc nadawania) oraz, w przypadku gorszego zasięgu, zwiększa maksymalną dopuszczalną liczbę retransmisji w celu wysyłki danych.

Opracowany model ma szereg ograniczeń i nie uwzględnia wielu parametrów mających wpływ na czas życia baterii. Obecnie gromadzimy dane dotyczące pracy urządzeń, które w przyszłości pozwolą nam oszacować czas życia baterii na podstawie danych rzeczywistych, uwzględniając m.in. warunki środowiskowe w jakich pracuje rejestrator, dodatkowych transmisji wynikających z wyzwolenia reguł, zmiany konfiguracji czy zdalnej aktualizacji oprogramowania.

Tabela 3. Wyniki analizy niezawodności transmisji danych
Table 3. Results of data transmission reliability analysis

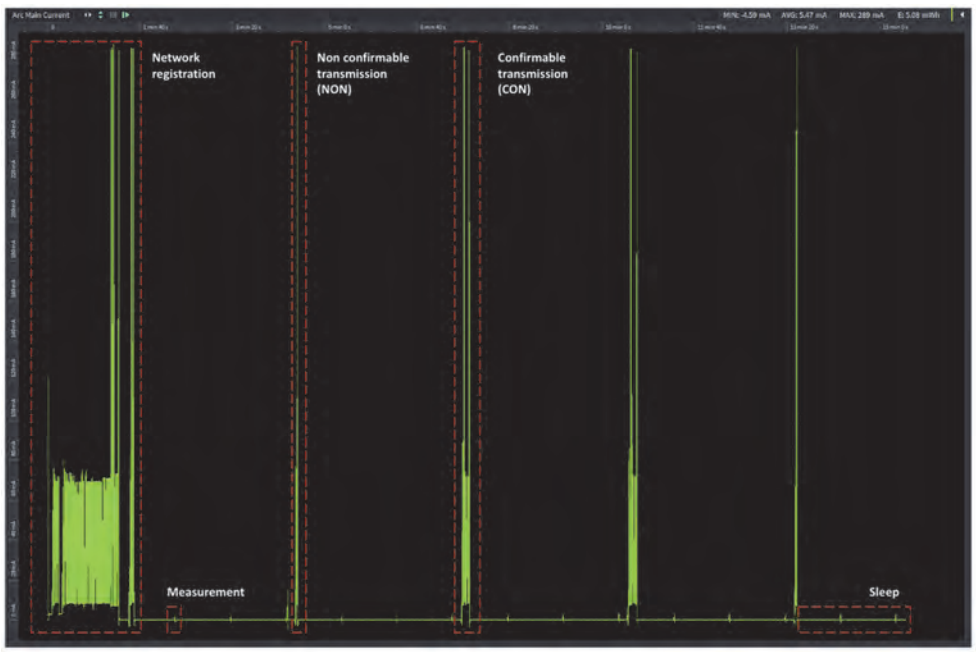
Typ ramki danych	Liczba wysłanych ramek danych	Liczba sukcesów w pierwszej transmisji	Liczba sukcesów w drugiej transmisji	Liczba sukcesów łącznie
NON	10 000	9995	brak retransmisji	9995
CON	10 000	9996	4	10 000

Tabela 4. Analiza poboru prądu w poszczególnych fazach pracy rejestratora temperatury i wilgotności NB-IoT
Table 4. Analysis of current consumption in each phase of logger's operations

Faza pracy	Średni pobór prądu [mA]	Czas trwania [s]
Uruchomienie, rejestracja do sieci	13,22	25,72
Transmisja bez potwierdzenia	8,04	3,6
Transmisja z potwierdzeniem	8,36	7,9
Tryb oszczędzania energii	0,01	–
Wykonanie pomiaru	0,747	0,06

Tabela 5. Symulacja czasu życia baterii w zależności od konfiguracji rejestratora
Table 5. Simulation of the impact of logger's configuration on the battery life time

Konfiguracja	Dobry zasięg (ECL0)	Średni zasięg (ECL1)	Słaby zasięg (ECL2)
Okres pomiaru: 15 min Okres transmisji: 180 min Potwierdzenie: zawsze	11 lat	4,7 lat	2,35 lat
Okres pomiaru: 15 min Okres transmisji: 60 min Potwierdzenie: zawsze	9,79 lat	1,96 lat	0,98 lat
Okres pomiaru: 15 min Okres transmisji: 180 min Potwierdzenie: zawsze	10,53 lat	2,11 lat	1,05 lat
Okres pomiaru: 5 min Okres transmisji: 180 min Potwierdzenie: zawsze	11 lat	4,69 lat	2,35 lat
Okres pomiaru: 5 min Okres transmisji: 60 min Potwierdzenie: zawsze	6,47 lat	1,3 lat	0,65 lat



Rys. 7. Wykres poboru prądu przez rejestrator Efento w poszczególnych fazach pracy
Fig. 7. Efento logger's current consumption in its operations phases

5.3. Zużycie danych

W analizie zużycia danych uwzględniono wszystkie dane przesyłane przez rejestrator (nagłówki protokołów IP, UDP, CoAP i przesyłane dane pomiarowe) oraz dane przesyłane do rejestratora (nagłówki protokołów, ramki ACK). Po opracowaniu modelu, wyniki zostały zweryfikowane z platformą operatora komunikacyjnego, prezentującą zużycie danych przez urządzenia.

Pierwsza analiza polegała na zbadaniu wpływu okresu transmisji na zużycie danych. Okres pomiaru ustawiony był na stałe na 15 minut, a wszystkie wysyłane ramki były typu CON. Miesięczne zużycie danych przez rejestrator w zależności od skonfigurowanego okresu transmisji przedstawiono na Rys. 8.

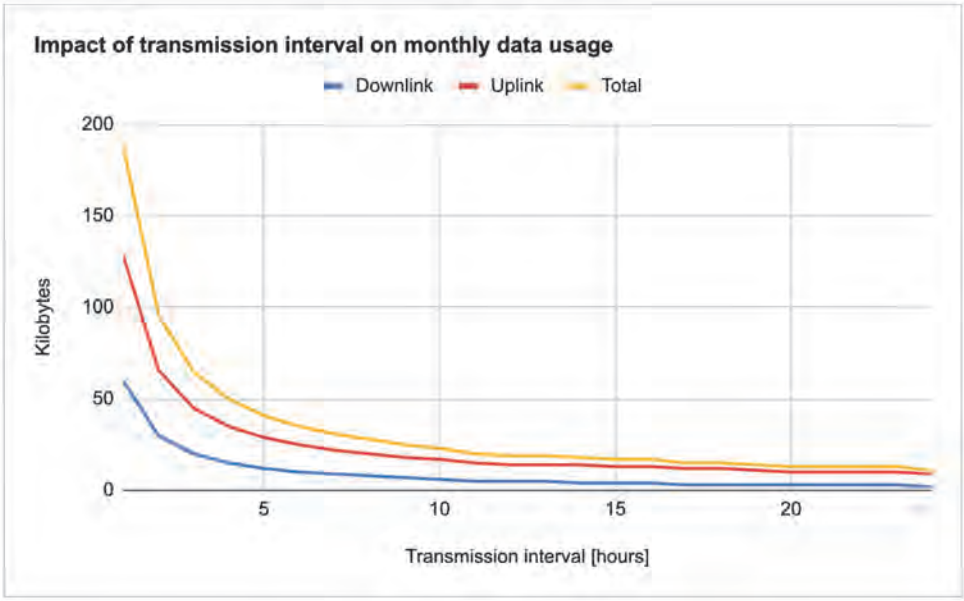
Zgodnie z przewidywaniami, zwiększenie okresu transmisji powoduje zmniejszenie miesięcznego zużycia danych przez rejestrator. Wynika to z narzutu protokołów IP, UDP i CoAP – im mniej ramek danych wysyła urządzenie, tym mniejszy jest sumaryczny narzut w skali miesiąca. Potwierdza to wspomnianą wcześniej konieczność odpowiedniego wyboru protokołów warstw transportowych i aplikacyjnej dla urządzeń IoT.

Kolejna analiza miała na celu zbadanie wpływu okresu pomiaru na zużycie danych. Przy zachowaniu stałego okresu transmisji (60 min) i zastosowaniu ramek typu CON zmodyfikowano okres pomiaru. Wpływ okresu pomiaru na miesięczne zużycie danych prezentuje Rysunek 9.

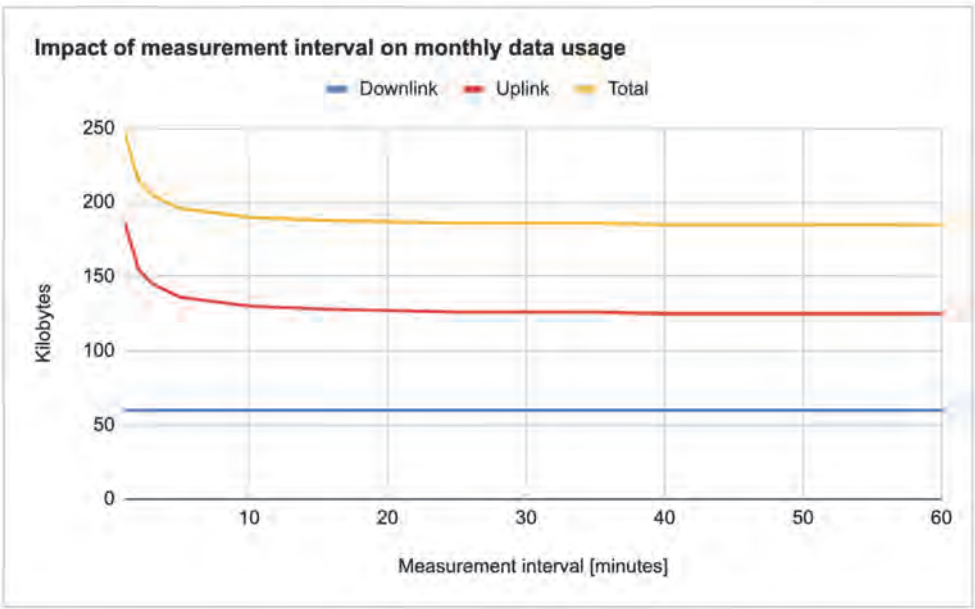
Zwiększenie okresu pomiaru powoduje zmniejszenie ilości danych wysyłanych przez rejestrator (uplink), nie ma jednak wpływu na ilość danych wysyłanych przez serwer do rejestratora w celu potwierdzenia otrzymania ramek. Zużycie danych jest odwrotnie proporcjonalne do okresu pomiaru i asymptotycznie zmierza do stałych wartości, które wynikają z narzutu protokołów IP, UDP i CoAP.

W ostatniej analizie sprawdzono wpływ typu ramki (CON i NON) na zużycie danych. Okres transmisji wynosił 60 minut, a okres pomiaru 15 minut. Zmieniano okres wysyłki ramki typu CON. Przykładowo, jeżeli okres wysyłki ramki CON wynosił 1 godzinę, to każda wysłana przez rejestrator ramka była ramką CON. Jeżeli okres wysyłki ramki CON wynosił 24 godziny, to 23 ramki wysyłane były jako ramki NON, a co 24. ramka była ramką typu CON. Wyniki analizy zostały zaprezentowane na Rys. 10.

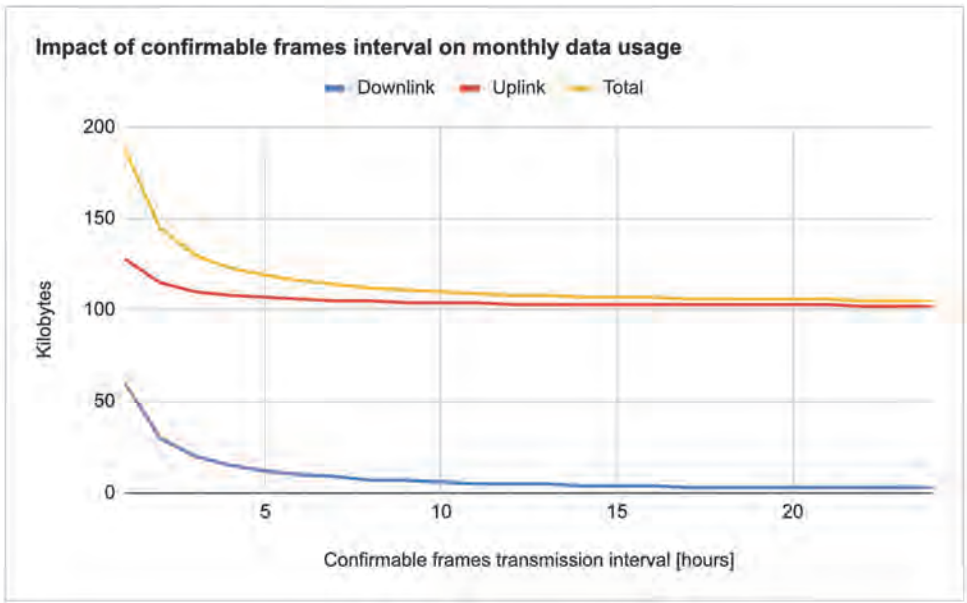
W tym przypadku widoczne jest zarówno zmniejszenie ilości danych wysyłanych przez rejestrator (uplink), jak i ilości



Rys. 8. Analiza wpływu okresu transmisji na miesięczne zużycie danych
Fig. 8. Analysis of the impact of the transmission interval on the monthly data usage



Rys. 9. Analiza wpływu okresu pomiaru na miesięczne zużycie danych
Fig. 9. Analysis of the impact of the measurement interval on the monthly data usage



Rys. 10. Analiza wpływu okresu wysyłki danych w ramkach typu CON na miesięczne zużycie danych
Fig. 10. Analysis of the impact of the transmission interval of the CON frames on the monthly data usage

danych otrzymywanych przez rejestrator (downlink) wraz ze zwiększaniem okresu transmisji danych z użyciem ramki CON. Spadek ilości danych wysyłanych przez rejestrator (uplink) przy większym okresie komunikacji z użyciem ramek typu CON wynika z wysyłki flagi RAI po otrzymaniu potwierdzenia z serwera. Im mniej transmisji wymaga potwierdzenia z serwera, tym urządzenie rzadziej wysyła te ramki (w przypadku ramek typu NON flaga zawarta jest w ramce z pomiarami), co przekłada się na ilość danych wysłanych przez rejestrator.

Wyniki pokrywają się z przewidywaniami teoretycznymi. Analizując zużycie danych przez rejestrator można stwierdzić, że opracowany sposób komunikacji wraz z metodą serializacji danych przyniósł oczekiwane rezultaty. Przy odpowiedniej konfiguracji (np. okres pomiaru 15 minut, okres transmisji 60 minut, potwierdzenie raz na dobę) rejestratory Efento zużywają mniej niż 1 MB danych rocznie. Biorąc pod uwagę stawki operatorów za korzystanie z sieci NB-IoT w 2023 r., przy takim zużyciu danych roczny koszt utrzymania pojedynczego rejestratora wynosi 1 Euro.

Zaprezentowane wyniki (zarówno czas życia baterii jaki i zużycie danych) mogą być odtworzone/zasymulowane dla innych konfiguracji z wykorzystaniem opracowanego narzędzia [22].

6. Podsumowanie

W oparciu o zdobyte doświadczenia przedstawiono istotne tematy, które należy brać pod uwagę projektując sposób komunikacji zasilanych bateryjnie urządzeń NB-IoT. Pokazano, w jaki sposób wykorzystano popularne protokoły komunikacji, funkcje oferowane przez standard NB-IoT i dodatkowe algorytmy minimalizujące zużycie danych przez urządzenia oraz wydłużenie czasu życia baterii. Przy odpowiedniej konfiguracji urządzenie może zużywać mniej niż 1 MB danych rocznie i pracować ponad 10 lat bez konieczności wymiany baterii. Wydaje się, że uzyskane wyniki potwierdzają poprawny projekt stosu komunikacji i algorytmów optymalizujących transmisję danych. Należy podkreślić, że ze względu na mnogość zastosowań urządzeń IoT i specyfikę ich pracy, projektując nowe rozwiązania należy każdorazowo przeprowadzić opisaną w artykule analizę.

Zawarta analiza nie uwzględnia wszystkich parametrów mających wpływ na pracę urządzeń NB-IoT. Pominięto również kwestie bezpieczeństwa (DTLS dla CoAP oraz konfigurację prywatnego APN), co ze względu na rozległość może być tematem na osobny artykuł.

Podziękowania

Artykuł powstał w oparciu o wyniki prac badawczo rozwojowych przeprowadzonych w ramach projektu „Opracowanie systemu monitoringu danych fizycznych opartego na technologii NB-IoT/LTE Cat M1 i platformie w chmurze, z wykorzystaniem nowej generacji bezprzewodowych sensorów" współfinansowany przez Unię Europejską ze środków Regionalnego Programu Operacyjnego Województwa Małopolskiego na lata 2014–2020 w ramach programu Gospodarka wiedzy.

Bibliografia

1. Sinha S., *State of IoT 2023: Number of connected IoT devices growing 16% to 16.7 billion globally*, IoT Analytics [https://iot-analytics.com/number-connected-iot-devices/]
2. Sinha R., Yiqiao W., Hwang S.-H., *A survey on LPWA technology: LoRa and NB-IoT*. „ICT Express", Vol. 3, No. 1, 2017, 14–21, DOI: 10.1016/j.icte.2017.03.004.
3. *NB-IoT & LTE-M March-2023 – Summary*, GSA [https://gsacom.com/paper/nb-iot-lte-m-march-2023-summary/]
4. *Release 13*, 3GPP [www.3gpp.org/specifications-technologies/releases/release-13]
5. Sultania A., Blondia C., Famaey J., *Optimizing the Energy--Latency Tradeoff in NB-IoT with PSM and eDRX*. „IEEE Internet of Things Journal", Vol. 8, No. 15, 2021, 12436–12454, DOI: 10.1109/JIOT.2021.3063435.
6. Hoglund A., Lin X., Liberg O., Behravan A., Yavuz E.A., Van Der Zee M., Sui Y., Tirronen T., Ratilainen A., Eriksson D., *Overview of 3GPP Release 14 Enhanced NB--IoT*. „IEEE Network", Vol. 31, No. 6, 2017, 16–22, DOI: 10.1109/MNET.2017.1700082.
7. Ralte V., Chawngsangpuii R., *Comparative Analysis of MQTT and CoAP Using Wireshark.* Evolution in Computational Intelligence, 2023, 369–380, DOI: 10.1007/978-981-19-7513-4_33.
8. Chen Y., Kunz T., *Performance evaluation of IoT protocols under a constrained wireless access network*. 2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT), Kair, Egipt, 2016, 1–7, DOI: 10.1109/MoWNet.2016.7496622.
9. Larmo A.; Ratilainen A., Saarinen J., *Impact of CoAP and MQTT on NB-IoT System Performance*. „Sensors", Vol. 19, No. 1, 2019, DOI: 10.3390/s19010007.
10. Shelby Z., Hartke K., Bormann C., *The Constrained Application Protocol* (*CoAP*), „Internet Engineering Task Force (IETF)", 2014, ISSN: 2070-1721.
11. MQTT v 5.0, Oasis, 2019 [https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html]
12. Thakur R., Kumari R., *A Comparison of Various IoT Application Layer Protocol*. „American Journal of Electronics & Communication", Vol. 3, No. 1, 2022, 28–34. DOI: 10.15864/ajec.3106.
13. Popić S., Pezer D., Mrazovac B., Teslić N., *Performance evaluation of using Protocol Buffers in the Internet of Things communication Protobuf vs. JSON/BSON comparison with a focus on transportation's IoT*, International Conference on Smart Systems and Technologies, Osijek, Chorwacja, 2016, DOI: 10.1109/SST.2016.7765670.
14. Hunter G., *A Comparison Of Serialization Formats* [https://blog.mbedded.ninja/programming/serialization-formats/a-comparison-of-serialization-formats/], 2019.
15. Martincevic J., *Data exchange in embedded systems – JSON vs Protocol Buffers, Infinium* [https://infinum.com/blog/json-vs-protocol-buffers/], 2023.
16. Specyfikacja Protobuf, Google [https://protobuf.dev/]
17. Specyfikacja MessagePack [https://msgpack.org/]
18. Specyfikacja BSON [https://bsonspec.org/]
19. Specyfikacja Apache Avero [https://avro.apache.org/]
20. Abbas M., Eklund J., Grinnemo K., Brunstrom A., Alfredsson S., Alay O., Katona S., Seres G., Rathonyi B., *Guidelines for an Energy Efficient Tuning of the NB-IoT Stack.* The 45th IEEE Conference on Local Computer Networks (LCN), Sydney, Australia, 2020, DOI: 10.1109/LCNSymposium50271.2020.9363265.

Inne źródła

21. Narzędzie umożliwiające symulację czasu życia baterii i zużycia danych przez rejestratory Efento [https://dev.efento.io/tools/battery]
22. Pliki .proto definiujące strukturę wiadomości przesyłanych między rejestratorami, a serwerem [https://github.com/efento/Proto-files/]

Implementation of Communication Stack for Battery-Powered NB-IoT Devices

Abstract: In this article we present how to implement an efficient communication stack for wireless, battery-powered NB-IoT loggers. Based on the experience gained by designing and developing loggers' software, we present our analyses performed in order to select the appropriate communication protocols, data serialisation format, configuration of the NB-IoT module and proprietary algorithms developed to optimise the number of transmissions and the amount of data sent. We also present the obtained results together with an analysis of the impact of individual configuration parameters on the devices' battery life and data usage.

Keywords: Narrowband IoT, IoT, CoAP, Protobuf, communication stack, logger, battery-powered

mgr inż. Piotr Szydłowski
piotr.szydlowski@efento.pl
ORCID: 0009-0007-1384-2011

Współzałożyciel i CEO Efento, firmy zajmującej się projektowaniem, rozwojem i produkcją rejestratorów bezprzewodowych oraz platformy do gromadzenia i przetwarzania danych pomiarowych. Przed założeniem Efento pracował w firmie doradczej (technologicznym i strategicznym). Absolwent Wydziału Fizyki i Informatyki Stosowanej, AGH Akademii Górniczo-Hutniczej, kierunek Fizyka Techniczna.

mgr inż. Karol Zaręba
karol.zareba@efento.pl
ORCID: 0009-0009-7886-5163

Współzałożyciel i CTO Efento, firmy zajmującej się projektowaniem, rozwojem i produkcją rejestratorów bezprzewodowych oraz platformy do gromadzenia i przetwarzania danych pomiarowych. Przed założeniem Efento zajmował różne stanowiska związane z rozwojem oprogramowania (programista, architekt oprogramowania). Absolwent Wydziału Elektrotechniki, Automatyki, Informatyki i Elektroniki, AGH Akademii Górniczo-Hutniczej, kierunek Automatyka i Robotyka.