POZNAN UNIVERSITY OF TECHNOLOGY

Hubert Andrzejewski

Protokół komunikacji bezprzewodowej dla dedykowanego systemu automatyki domowej

Master's Thesis

Supervisor: dr inż. Mariusz Nowak

Poznań, 2014

Spis treści

Wstęp

System automatyki domowej jest rozszerzeniem systemu zarządzania budynkiem. Wchodzi on w bezpośrednie interakcje z użytkownikiem, umożliwiając mu tym samym zarządzanie oraz definiowanie własności behawioralnych elementów składowych systemu oraz precyzowanie ich wzajemnych oddziaływań. Reguły oraz akcje sformułowane w ten sposób służyć mają podniesieniu komfortu użytkownika, jego bezpieczeństwa oraz obniżeniu kosztów eksploatacji budynku [Har03]. Co więcej, systemy tego typu pozwalają podnieść jakość życia osób starszych i niepełnosprawnych, ułatwiając kontrolę nad mieszkaniem. Jednocześnie, systemy automatyki są z powodzeniem wykorzystywane w budynkach o charakterze biurowym czy korporacyjnym o zdecydowanie większej powierzchni i złożoności planu niż budynki mieszkalne. W tym przypadku, poza obniżeniem kosztów eksploatacyjnych systemy automatyki zapewniają wyższy poziom kontroli dostępu oraz nadzoru przemieszczania się osób niż tradycyjne sieci monitoringu. Dzięki temu systemy tego typu często są stosowane równocześnie z monitoringiem, stanowiąc jego istotne uzupełnienie, a niekiedy nawet całkiem go zastępując.

Systemy automatyki domowej cały czas zyskują na popularności. W roku 2012 jedynie w USA zostało zamontowanych 1.5 miliona systemów automatyki domowej i według prognoz, liczba ta w ciągu następnych 6 lat ma wzrosnąć czterokrotnie[1]. Przyczyn tego sukcesu doszukiwać się można m.in. w rosnącej konkurencji na rynku systemów inteligentnych budynków i powiązaną z nimi automatyką domową. W konsekwencji skutkuje to spadkiem cen oraz poprawą jakości rozwiązań oferowanych przez poszczególne firmy.

Jednocześnie, różnorodność rozwiązań przeznaczonych dla inteligentnych budynków wymaga dostępności urządzeń współpracujących z danym systemem. Spełnienie tego postulatu często stanowi poważny problem, szczególnie w przypadku małych rozwijających się systemów. Zmuszone są one przez to uciekać się do zastosowania istniejącego już standardu komunikacji jako bazy dla autorskiego systemu. Częstym wyborem takiego standardu jest ZigBee®, będący własnością ZigBee®Alliance[2], który zdobył już niemałą popularność wśród producentów sprzętu RTV/AGD[3]. Oczywiście protokół ten nie jest monopolistą na polu bezprzewodowych protokołów komunikacyjnych automatyki domowej. Innymi protokołami, o których warto wspomnieć ze względu na wielką popularność, są Z-Wave, KNX–RF oraz X-comfort. Zostaną one również bliżej opisane w dalszej części pracy. Decyzja o wyborze powszechnie używanego protokołu jest bezpieczniejsza pod względem biznesowym i technologicznym, zwiększając tym samym szansę na rozpowszechnienie się systemu oraz ułatwiając jego implementacje. Wiele rozwiązań próbujących wypromować

[1] Allied Business Intelligence Inc, *1.5 Million Home Automation Systems Installed in the US This Year* https://www.abiresearch.com/press/15-million-home-automation-systems-installed-in-th, 2014

[2] Oficjalna strona ZigBee®Alliance https://www.zigbee.org, 2014

[3] Lista produktów i producentów posiadających certyfikat ZigBee®na oficjalnej stronie ZigBee®Alliance http://www.zigbee.org/Products/ByFunction/Appliances.aspx, 2014

własne standardy komunikacji, stało się jedynie niszową ciekawostką, zdobywając blisko zerowy udział w rynku. Przykładem może być system stworzony przez Hewlett-Packard, będący w praktyce kompilacją wszystkich produktów tej firmy[4] oraz system HomeOS[5], stworzony przez oddział Microsoft Research. W przypadku drugiego z wymienionych systemów warto nadmienić jednak, że jest to jak dotąd wersja rozwojowa, dystrybuowana na zasadzie darmowej otwartej licencji.

Mnogość dostępnych urządzeń współpracujących z systemem automatyki domowej jest jednym z zasadniczych kryteriów podczas ewentualnej decyzji o zakupie i wyborze danego rozwiązania automatyki domowej. Jako podstawowe elementy takiego systemu można wyróżnić kontrolery oświetlenia, dostosowujące jego parametry do aktualnych warunków (pora dnia, wykonywana czynność) i zarządzające nimi w sposób obniżający pobór prądu, kontrolery ogrzewania, wentylacji i klimatyzacji (HVAC) oraz podsystemy odpowiedzialne za bezpieczeństwo. W tym przypadku, pod pojęciem bezpieczeństwa rozumiana jest zarówno ochronę budynku przed potencjalnym włamaniem, jak i również zapobieganie ewentualnym zalaniom, ulatnianiu się gazu czy zwarciom, zapewniane poprzez monitoring obciążenia instalacji dostarczającej te media. Na podsystemy mające na celu poprawę bezpieczeństwa składają się również elektroniczne blokady drzwi, elektryczne sterowniki otwierania okien, drzwi lub rolet, czujniki obecności osób. Poza wymienionymi podstawowymi elementami, wiele dostępnych rozwiązań na rynku posiada w zestawie swoich funkcji strumieniowe przesyłanie multimediów w obrębie systemu, przy czym dotyczy to zarówno obrazu z kamer monitoringu, jak i filmów czy muzyki. Popularne są też zintegrowane z systemem roboty odkurzające czy pielęgnujące ogród, systemy irygacji oraz przydomowe stacje meteorologiczne. Coraz też więcej producentów sprzętu RTV/AGD integruje swoje produkty z protokołami komunikacji stosowanymi w systemach automatyki domowej. Dostępne są na rynku inteligentne pralki, lodówki, ekspresy do kawy, umożliwiające zdalne zarządzenie za pośrednictwem systemu. Pojawiają się też koncepcje wyposażenia smartfonów w moduły komunikacyjne ZigBee[6], umożliwiając tym samym bezpośrednie kontrolowanie za pomocą telefonu systemu opartego na tym protokole.

Podczas projektowania systemu automatyki domowej dąży się do jak największego uproszczenia jego obsługi przez użytkownika. Równie ważnym celem jest dążenie do otrzymania systemu, w którym wszystkie warstwy przenikają się wzajemnie, koordynując swoje działania. Nie jest to zadanie proste ani łatwo osiągalne, dla systemu dobrej jakości wymaga ono zaawansowanej predykcji połączonej ze sztuczną inteligencją, opierającą swoje wnioskowania na odczytach z inteligentnych, odpowiednio rozlokowanych czujników. Może wydawać się to przesadą, ale nieporozumienie między mieszkańcem a jego mieszkaniem, może być o wiele bardziej groźne w skutkach niż komunikat błędu w zwykłych systemach komputerowych. Najczęściej zatem, w związku z tymi trudnościami, systemy automatyki domowej oraz inteligentnych budynków ograniczone są do prostych interakcji, takich jak sterowanie podstawowymi parametrami otoczenia (zapalanie i gaszenie świateł, ustawienie pożądanej temperatury pomieszczenia). Jednakże, tak pozornie skromny asortyment funkcjonalności okazuje się wystarczający dla większości użytkowników i potencjalnych

[4] Oficjalna strona projektu HP SmartHome http://www.hpsmarthome.com, 2011

[5] Oficjalna strona projektu HomeOS http://research.microsoft.com/en-us/projects/homeos, 2014

[6] Chriss Hall, *ZigBee support coming to Samsung phones, HTC and others may follow suit*, strona Pocket-lint ltd http://www.pocket-lint.com/news/122257-zigbee-support-coming-to-samsung-phones-htc-and-others-may-follow-suit, 10.06.2013

nabywców systemów automatyki domowej.

Motywacja

Tematyka inteligentnych budynków pojawiła się w obszarze moich zainteresowań już w roku 2006, kiedy to został przeze mnie stworzony w języku Delphi pierwszy system, ustawicznie ulepszany i rozbudowywany. Niestety, ogólna koncepcja tego systemu uniemożliwiła na pewnym etapie jego dalszy rozwój, wymuszając tym samym stworzenie jego następcy od podstaw. Rozważane były różne koncepcje projektu oraz różne sposoby komunikacji pomiędzy jego elementami składowymi. Ostatecznie zdecydowałem się na realizację nowego systemu na bazie własnego protokołu komunikacji bezprzewodowej, współpracującego jednak na poziomie całego systemu z innymi protokołami jak ZigBee czy Z-Wave. Decyzja o zaprojektowaniu nowego protokołu, obarczona jak wcześniej zostało to wspomniane dużym ryzykiem, wynika z niedostępności całkowicie otwartego i wolnego dla użytku komercyjnego protokołu komunikacji bezprzewodowej dla automatyki domowej.

Rezygnacja z wykorzystania w systemie łączności przewodowej podyktowana była dążeniem do stworzenia system taniego, łatwego i szybkiego w instalacji. Wszystkie te trzy obszary są w pewnym stopniu zależne od sposobu montażu systemu w budynku i poniesionych przy tym nakładów. Urządzenia komunikujące się bezprzewodowo nie wymagają instalacji okablowania strukturalnego, wymaganego do działania ich przewodowych odpowiedników, obniżając tym znacząco zarówno koszt, jak i czas montażu systemu automatyki domowej w wykończonych już budynkach. Kontrolery i sterowniki wykorzystujące technologie radiowe do komunikacji, ułatwiają ich późniejsze przemieszczanie i przebudowę topologii systemu, bez nadmiernego nakładu kosztów i czasu. Oczywiście, łączność bezprzewodowa nie jest pozbawiona wad, z których chyba najbardziej problematyczną jest ograniczony zasięgu sygnału i jego ewentualna utrata, w konsekwencji powodująca zerwanie połączenia i często awarię systemu jako całości. Poważnym zagrożeniem dla łączności bezprzewodowej są również interferencje, obniżające znaczenie wydajność komunikacji lub też całkiem ją uniemożliwiając. Ponadto, urządzenia radiowe są z reguły droższe od tych przewodowych, ze względu na ich stosunkową nowość i w dalszym ciągu mniejszą powszechność na rynku. Dostępne jest jeszcze jedno rozwiązanie problemu komunikacji w obrębie systemu, łączące zarówno wady, jak i zalety obu wspomnianych poprzednio metod łączności. Komunikacja elektroenergetyczną siecią rozdzielczą (Power Line Communication - PLC), zdefiniowana m.in. przez standard IEEE 1901[7], pozwala na wykorzystanie istniejącej sieci elektroenergetycznej do komunikacji pomiędzy urządzeniami. Rozwiązanie to pozwala zniwelować koszt montażu okablowania strukturalnego dla tradycyjnej komunikacji przewodowej oraz uniknąć problemów spowodowanych przez utratę zasięgu sygnału radiowego w łączności bezprzewodowej. Z drugiej jednak strony koszt adapterów PLC jest porównywalny, jeśli czasem nie wyższy, od urządzeń komunikacji radiowej. Ponadto, komunikacja za pomocą sieci elektroenergetycznej cechuje się często niższą przepustowością i prędkością od pozostałych rozwiązań oraz jest podatna na interferencje spowodowane pozostałymi urządzeniami, podłączonymi do sieci elektrycznej budynku.

[7] Latchman, H., Katar S, Yonge L., Gavette S., *Homeplug AV and IEEE 1901:A Handbook for PLC Designers and Users*, 2013

Rozważając cechy wymienionych metod komunikacji oraz mając na względzie prostotę i koszty systemu, została podjęta decyzja o wykorzystaniu bezprzewodowej technologii komunikacji, używającej własnego, darmowego i otwartego protokołu.

Założenie o otwartości dotyczy ponadto całego systemu, od poziomu programowego aż do sprzętowego. Oczywiście dostępne są obecnie systemy otwarte, cieszące się już niemałą popularnością. Jednym z nich jest openHAB[8], pracujący na wirtualnej maszynie Javy, uniezależniając go tym samym od platformy. Pozwala on zintegrować obsługę urządzeń komunikujących się w standardzie ZigBee, Z-Wave, KNX oraz wielu innych[9]. Pomimo, iż system ten umożliwia dowolne modyfikacje i dostosowywanie go do własnych potrzeb, nie została podjęta przeze mnie decyzja o wykorzystaniu tego systemu do obsługi własnego protokołu komunikacyjnego. Została ona głównie podyktowana charakterystyką tworzonego standardu komunikacji, odmiennego od tych w protokołach obsługiwanych przez wspomniany system. Kolejnym argumentem przemawiającym za rezygnacją z wykorzystania systemu openHAB, była jego zależność od wirtualnej maszyny Javy (JVM), która pomimo wzrastającej z każdym wydaniem wydajności, nie wyprzedzi na tym polu programów natywnych, napisanych w języku C lub C++. JVM nie pozwala również na integrację z samym systemem operacyjnym platformy sprzętowej, w stopniu w jakim robi to program dedykowany. Z tych też przyczyn została podjęta decyzja o realizacji nowego systemu od podstaw w języku C/C++, skompilowanego pod system Linux, umożliwiając tym samym uruchomienie na wielu tanich platformach komputerowych, jak Raspberry Pi[10] czy BeagleBone[11].

Pomimo istnienia na rynku wielu systemów automatyki domowej i przybywania nowych, tworzenie nowego systemu wraz z dedykowanym protokołem komunikacji bezprzewodowej jest projektem opłacalnym i uzasadnionym ekonomicznie. Systemy automatyki domowej nieustannie zyskują na popularności, stając się już niejako standardem w nowoczesnym budownictwie i ważnym atutem nowopowstałych osiedli mieszkaniowych. Przykładem tego zjawiska mogą być osiedla powstałe w ostatnich latach w Gorzowie Wielkopolskim, Kołobrzegu, Warszawie i w wielu innych miastach, w których działają spółki budowlane ściśle współpracujące z firmą Fibaro[12], będącą jednym z liderów systemów inteligentnych budynków na rynku polskim. Rosnące ceny eksploatacji budynków wymuszają niejako instalację systemów pozwalających zarządzać zużyciem mediów oraz generowaniem odpadów, w konsekwencji obniżając koszty i chroniąc środowisko naturalne. To właśnie ekologia była podstawą dla stworzenia dyrektywy unijnej 2010/31/UE z dnia 19 maja 2010 r.[13], deklarującej wsparcie finansowe oraz prawne dla inteligentnego, energooszczędnego budownictwa. Można zatem oczekiwać, iż popyt na systemy inteligentnych budynków i automatyki domowej będzie wzrastał, stwarzając tym samym miejsce dla konkurencji i nowych systemów. Ponadto, tworzenie urządzeń inteligentnych, umożliwiających komunikację z innymi sprzętami, staje się coraz powszechniejsze wśród producentów różnorakich

[8] Oficjalna strona projektu openHAB http://www.openhab.org/, 2014
[9] Lista wspieranych technologii systemy openHAB na oficjalnej stronie projektu http://www.openhab.org/features-tech.html, 2014
[10] Oficjalna strona projektu Raspberry PI http://www.raspberrypi.org/, 2014
[11] Oficjalna strona projektu BeagleBone http://beagleboard.org/Products/BeagleBone, 2014
[12] Oficjalna strona firmy Fibaro www.fibaro.com, 2014
[13] Ustawa z dnia 19 maja 2003 roku w sprawie charakterystyki energetycznej budynków, Dz. U. UE z 2010r. L 153/13

urządzeń, naturalnie narzucając potrzebę zastosowania systemu gospodarującego nimi.

Systemy zarządzające budynkiem mogą również znaleźć zapotrzebowanie w instytucjach muzealnych bądź zabytkach, gdzie utrzymanie określonych parametrów środowiska, jak temperatura czy wilgotność, jest nie tyle zabiegiem poprawiającym komfort, co niezbędnym dla prawidłowej konserwacji obiektów muzealnych. Fakt, iż aktualnie w polskich instytucjach o charakterze muzealnym wykorzystywane są wciąż analogowe czujniki pomiarowe, nie sprzężone w żaden sposób z instalacją grzewczą bądź wentylacyjną, jest jedynie pozostałością po poprzednim ustroju w mentalności osób zarządzających oraz brakiem odpowiedniego dofinansowania. Można jednak żywić przekonanie, że stan ten niedługie ulegnie radykalnej zmianie i większość instytucji muzealnych będzie poszukiwała systemu pozwalającego chronić zabytki przed wpływem niekorzystnych warunków otoczenia. Warto zatem podjąć starania wypełnienia tej niszy na polskim rynku inteligentnego wyposażenia dla muzeów za pomocą systemów automatyki domowej, na tyle elastycznymi, aby sprostały różnorodności przeznaczonych dla nich zadań.

Jakoś nawiązań do elstyczności, bo akapit jest z czapy

Cel i zakres pracy

Celem niniejszej pracy jest zaproponowanie specyfikacji nowego standardu komunikacji bezprzewodowej dla systemów automatyki domowej oraz podanie przykładowej implementacji zdefiniowanego protokołu, bazując na aktualnie dostępnych protokołach o podobnych zastosowaniach.

Pracę podzielić można na trzy części, z których pierwsza, stanowiąca wprowadzenie, ma charakter czysto teoretyczny. Dwie kolejne części ilustrują zarówno teoretyczne koncepcje zdefiniowanego protokołu, jak i propozycję ich faktycznej realizacji przy wykorzystaniu dostępnych technologii sprzętowych oraz programowych.

W pierwszej kolejności, w rozdziale drugim szczegółowo scharakteryzowane zostały najpopularniejsze standardy komunikacji bezprzewodowej dla systemów automatyki domowej: ZigBee® oraz Z–Wave. Ponadto, została przybliżona charakterystyka protokołów mniej popularnych, jednak posiadających istotne koncepcje lub znaczenie historyczne.

Rozdział trzeci poświęcony został specyfikacji stworzonego w ramach pracy standardu komunikacji bezprzewodowej. Określone zostały ogólne założenia standardu, architektura jego stosu oraz definicja poszczególnych warstw. Zaproponowana została również przykładowa implementacja standardu dla systemów wbudowanych.

Rozdział czwarty zawiera wymogi stawiane urządzeniu realizującemu funkcję nadzorcy systemu. Podobnie jak w rozdziale poprzednim, podana została przykładowa implementacja nadzorcy systemu, zrealizowana w ramach niniejszej pracy.

Rozdział piąty zawiera podsumowanie pracy, wnioski oraz propozycje dalszego rozwoju projektu.

Załączony został ponadto podręcznik użytkownika, pozwalający na zapoznanie się ze sposobami używania stworzonego w ramach pracy protokołu i systemu.

Rozdział 1

Charakterystyka istniejących standardów komunikacji bezprzewodowej dla systemów automatyki domowej

Nie jestem przekonany co do sformułowania tytułu rozdziału, aby był krótki a precyzyjny

Niniejszy rozdział poświęcony jest specyfikacjom oraz charakterystykom istniejących standardów komunikacji bezprzewodowej dla systemów automatyki domowej. W zestawieniu tym, zostały wzięte pod uwagę wiodące na rynku standardy bezprzewodowe, niestety najczęściej o charakterze zamkniętym. Wynikająca z tego faktu częściowa lub czasem nawet całkowita niedostępność dokumentacji technicznej oraz fragmentarycznej przydatności niektórych standardów dla realizacji tej pracy, był przyczyną powierzchowności oraz wybiórczości zamieszczonego opisu.

1.1 ZigBee®

ZigBee® jest specyfikacją protokołu transmisji danych w sieciach bezprzewodowych typu WPAN. Koncepcja tego protokołu została zawiązana w roku 1998, następnie ustandaryzowana w 2003 i uaktualniona w roku 2006. Prace te odbywały się pod nadzorem konsorcjum ZigBee® Alliance, będącego aktualnym właścicielem protokołu oraz zrzeszeniem firm go rozwijających i wykorzystujących. Protokół ZigBee® umożliwia tworzenie sieci bezprzewodowych o topologii gwiazdy, drzewa lub kraty, charakteryzujących się niewielkimi przepływnościami, rzędu 250 kbps oraz dystansem pomiędzy węzłami około 10 metrów. Urządzenia tworzące węzły sieci, cechują się najczęściej niewielkim poborem energii oraz małą mocą obliczeniową.

Specyfikacja protokołu ZigBee® definiuje cztery warstwy stosu sieciowego, bazując przy tym na standardzie IEEE 802.15.4 [oEEE11]. Standard ten określa zarówno fizyczne aspekty medium, zawarte w warstwie fizycznej PHY, jak i zasady sterowania dostępem do niego, poprzez warstwę MAC. Ponadto, definiuje on parametry urządzeń sieci LR-WPAN oraz interakcje pomiędzy nimi zachodzące. Na tej podstawie zostały oparte warstwy wyższe protokołu ZigBee®: warstwa sieciowa NWK, odpowiadająca za zachowanie sieci i jej topologię oraz warstwa aplikacji, udostępniająca interfejs dla aplikacji końcowej poprzez framework.

Każda z warstw realizuje określony zestaw usług dla warstwy wyższej. Usługi te określane są formalnie jako zestawy tzw. prymitywów, dzielących się na prymitywy żądania, wskazania, odpowiedzi i potwierdzenia. Tworzą one interfejs komunikacyjny pomiędzy warstwami, eksponowany poprzez punkty SAP (ang. *Service Access Point*) jednostek DE (ang. *Data Entity*), świadczących usługi transmisji danych oraz jednostek ME (ang *Management Entity*), świadczących wszystkie pozostałe usługi.

Zarys architektury stosu zdefiniowanego przez specyfikację ZigBee® został przedstawiony na rysunku **??**. Zostanie on opisany w sposób bardziej szczegółowy w poniższych sekcjach. Poglądowy opis specyfikacji znaleźć można w dokumentacji udostępnianej za pośrednictwem oficjalnej strony ZigBee® Aliance[1] [ZA08]. Dostęp do szczegółowej specyfikacji oraz komercyjne wykorzystanie standardu związane jest z wymogiem przynależności do ZigBee® Alliance, obarczonej rocznym abonamentem wysokości $4000.

1.1.1 Warstwa fizyczna

Warstwa fizyczna PHY protokołu ZigBee® zdefiniowana jest przez standard IEEE 802.15.4 [oEEE11]. Zgodnie z tym standardem, możliwa jest komunikacja z wykorzystaniem różnych zakresów częstotliwości, których wybór zależny od kraju przeznaczenia produktu i obowiązujących w nim norm prawnych. W tabeli 1.1 zostały przedstawione wybrane częstotliwości, wykorzystywane przez urządzenia ZigBee®. Pełną listę częstotliwości oraz modulacji używanych przez standard IEEE 802.15.4, można znaleźć w dokumentacji [oEEE11, str. 167].

Tablica 1.1.: Częstotliwości stosowane przez ZigBee®

Obszar	**Częstotliwość**	**Kanały**	**Przepływność**	**Kluczowanie**
Europa	868–868.6 MHz	3	20 kbps	BPSK/OQPSK
Ameryka, Australia	902–928 MHz	13	40 kbps	BPSK/OQPSK
Cały świat	2.4–2.4835 GHz	16	250 kbps	OQPSK

Wszystkie trzy zakresy częstotliwości wykorzystują bezpośrednią metodę rozpraszania widma DSSS (ang. *Direct-Sequence Spread Spectrum*), stosując jednak różne metody kluczowania. Zasięg komunikacji pomiędzy urządzeniami zależy bezpośrednio od prędkości transmisji, wpływającej na osiągalną czułość odbiornika oraz od używanej częstotliwości, której mniejsza wartość redukuje straty propagacji sygnału.

Głównymi funkcjami realizowanymi przez warstwę PHY jest aktywacja nadajnika/odbiornika radiowego, wybór kanału komunikacyjnego, przeprowadzanie detekcji energii, pomiar jakości połączenia, ocena zajętości kanału, skanowanie kanału w poszukiwaniu beaconów oraz wymiana pakietów za pośrednictwem fizycznego medium. Funkcje te zostaną przybliżone w poniższych paragrafach.

1.1.1.1 Detekcja energii przez odbiornik

Pomiar ED (ang. *Energy Detection*) jest estymacją energii sygnału wewnątrz kanału, podczas której nie są podejmowane żadne próby zdekodowania lub rozpoznania odebranego sygnału. Pomiar ten jest wykorzystywany podczas realizacji algorytmu poszukiwania i wyboru wolnego kanału przez warstwę NWK. Wynik detekcji jest przekazywany do warstw wyższych jako wartość mapowana liniowo z zakresu 0–40 dB, z dokładnością ±6 dB.

[1] Oficjalna strona ZigBee®Alliance https://www.zigbee.org, 2014

1.1.1.2 Wskaźnik jakości połączenia

Pomiar LQI (ang. *Link Quality Indicator*) jest charakterystyką siły i/lub jakości odebranego pakietu. Jest on wymagany przez warstwę sieciową stosu dla określenia m.in. optymalnej trasy routingu. Pomiar LQI dokonywany jest przy wykorzystaniu ED, estymacji stosunku SIN (ang. *Signal–to–noise*) lub kombinacji tych dwóch metod.

1.1.1.3 Ocena zajętości kanału

Badanie CCA (ang. *Clear Channel Assessment*) jest wykonywane na zlecenie warstwy NWK w celu znalezienia nieużywanego kanału komunikacji i jego wykorzystania. Ocena ta jest wyznaczana przy wykorzystaniu przynajmniej jednej z metod:

1. Energia powyżej progu — nośnik jest określany jako zajęty po wykryciu energii powyżej progu ED
2. Wykrycie nośnej — nośnik jest określany jako zajęty po wykryciu sygnału o charakterystyce zgodnej ze standardem IEEE 802.15.4. Energia wykrytego sygnału może być zarówno nad, jaki i pod progiem ED
3. Wykrycie nośnej z energią powyżej progu — nośnik jest określany jako zajęty po wykryciu sygnału o charakterystyce zgodnej ze standardem IEEE 802.15.4 oraz energii powyżej progu ED

Poza wymienionymi metodami, w dokumentacji standardu IEEE 802.15.4 opisane są trzy inne realizacje CCA, nie są one jednak stosowane w protokole ZigBee® i z tej też przyczyny, zostaną one w tej pracy pominięte [oEEE11, str. 153].

1.1.1.4 Format ramki

Standard IEEE 802.15.4 określa kilka rodzajów ramek warstwy PHY, uzależnionych od wykorzystanej metody modulacji, różniących się jednak nieznaczenie. W ujęciu ogólnym, ramka warstwy fizycznej PPDU (ang. *Packet PHY Data Unit*) składa się z trzech podstawowych komponentów (Rysunek 1.1), niezbędnych do poprawnej transmisji danych:

- Nagłówek synchronizacji — zawierający w sobie preambułę, pozwalającą na synchronizację odbiornika z nadajnikiem oraz informację o początku danych pakietu
- Nagłówek PHY — zawierający informacje o długości ramki
- Blok danych PHY — zawierający dane przekazane przez warstwy wyższe z poleceniem ich transmisji, w formie PSDU (ang. *Physical Service Data Unit*)

Jak można zauważyć, zachodzi tutaj mechanizm kapsułkowania, w wyniku którego informacje przekazywane warstwom niższym są przez nie zaopatrzone w odpowiednie nagłówki, umożliwiające dalszy transport i podejmowanie odpowiednich akcji. Mechanizm kapsułkowania jest stosowany dla wszystkich warstw stosu ZigBee®.

1.1.2 Podwarstwa sterowania dostępem do medium

Głównym zadaniem podwarstwy MAC (ang. *Media Access Control*) jest zarządzanie PAN oraz kontrola dostępu do medium transmisyjnego, w tym przypadku będącego kanałem

<table>
<tr><td colspan="3"></td><td colspan="2">Oktety</td></tr>
<tr><td colspan="3"></td><td colspan="2">1</td><td>zmienna</td></tr>
<tr><td>Preambuła</td><td>SFD</td><td>Długośćramki (7bitów)</td><td>Zarezerwowane (1bit)</td><td>PSDU</td></tr>
<tr><td colspan="2">SHR</td><td colspan="2">PHR</td><td>Blokdanychd PHY</td></tr>
</table>

Rysunek 1.1.: Format ramki warstwy fizycznej – PPDU
Źródło: opracowanie własne na podstawie [oEEE11, str. 180]

radiowym. Pozostałymi funkcjami realizowanymi przez podwarstwę jest walidacja ramek, dostarczanie ramek ACK, asocjacja i deasocjacja. Ponadto, w ramach realizacji kontroli dostępu do medium, podwarstwa MAC jest odpowiedzialna za generowanie i nadawanie ramek beaconów, z których bezpośrednio wynika kontrola superramki oraz gwarantowanych szczelin transmisji. Z mechanizmami kontroli powiązany jest również algorytm CSMA/CA, który zgodnie ze standardem, jest zaimplementowany w tej podwarstwie. Każde z urządzeń posiada adres 64–bitowy oraz przy spełnieniu odpowiednich warunków, krótki adres 16–bitowy, wykorzystywany do komunikacji wewnątrz sieci PAN (ang. *Personal Area Network*).

1.1.2.1 Rodzaje urządzeń

Standard IEEE 802.15.4 definiuje dwa podstawowe typu urządzeń: FFD (ang. *Full-Functional Device*) oraz RFD (ang. Reduced-Functional Device). Urządzenia FFD, w odróżnieniu od urządzeń RFD, posiadają możliwość pracy jako koordynator PAN, koordynator lub zwykłe urządzenie. RFD mogą pracować jedynie jako zwykłe urządzenia, najczęściej bardzo proste i ograniczone, typu włącznik światła lub pasywny czujnik podczerwieni. Urządzenia o ograniczonej funkcjonalności nie potrzebują przesyłać dużych ilości danych i są zasocjowane z pojedynczym FFD. Urządzenie FFD, pracujące jako koordynator PAN, jest niezbędnym elementem każdej sieci. Jest on odpowiedzialny za jej utworzenie, modyfikację oraz asocjację i deasocjację innych urządzeń.

1.1.2.2 Tryby pracy

W zależności od ustawionych parametrów, urządzenia FFD może operować w dwóch trybach, określanych przez emisję ramek beaconów lub jej brak, przy czym beacony może wysyłać jedynie koordynator PAN lub koordynator zasocjowany z istniejącą siecią PAN. W zależności od trybu pracy przyjętej w sieci, zmienia się jej zachowanie oraz realizacja komunikacji pomiędzy urządzeniami. Zgodnie ze specyfikacją ZigBee®, ramki beaconów nie mogą być emitowane w przypadku sieci o topologii kraty.

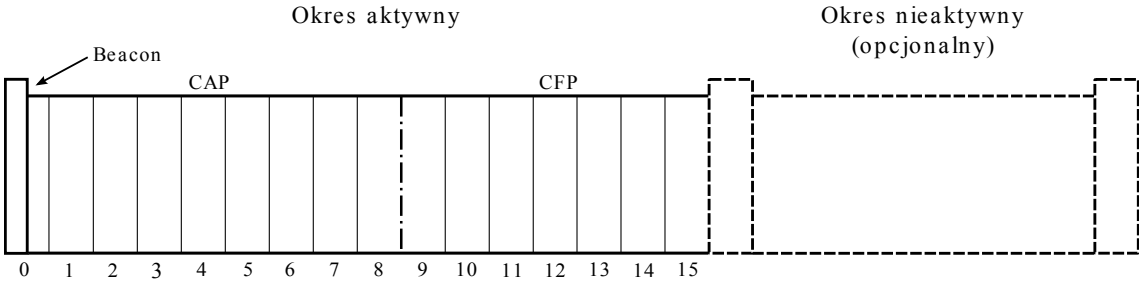
Beacony są specjalnym typem danych, nadawanym w przeznaczonych do tego celu ramkach. Przekazują informacje o sieci nowym urządzeniom, ułatwiając im podjęcie decyzji o wyborze sieci i znalezieniu sąsiadów. Ponadto, beacony umożliwiają komunikację urządzeniom będącym już w sieci, synchronizując je z koordynatorem, pozwalając na identyfikację PAN oraz wyznaczając granicę superramki i informując o jej porządku. W ramce beaconu powinny być zawarte informacje o rodzaju protokołu komunikacyjnego, możliwościach i

typie urządzenia nadającego, adresie, pozycji urządzenia oraz o ewentualnych wiadomościach oczekujących na urządzenie bądź akcjach z nim związanych. Dane udostępniane przez beacony są wykorzystywane przez warstwę sieciową do ustalenia tras routingu oraz zarządzania zasobami sieciowymi.

W przypadku sieci w których zrezygnowano z emisji beaconów, wszystkie te informacje są uzyskiwane w procesie poolingu urządzenia na koordynatorze PAN.

1.1.2.3 Superramka

W sieciach LR–WPAN dopuszczających emisję ramek beaconów, wykorzystywana jest struktura superramki, której format jest określany przez koordynatora nadającego beacony. Superramka jest formą realizacji komunikacji umożliwiającą harmonogramowanie czasów transmisji, a w konsekwencji oszczędzanie energii przez urządzenia należące do sieci LR–WPAN, których asocjacja musi być utrzymywana, pomimo długich okresów braku aktywności.



Rysunek 1.2.: Struktura superramki podwarstwy MAC standardu IEEE 802.15.4
Źródło: opracowanie własne na podstawie [oEEE11, str. 18–21]

Superramka jest obustronnie ograniczona poprzez transmisję beconów i jest podzielona na 16 szczelin czasowych o równym rozmiarze (Rysunek 1.2). Ramka beaconu jest wysyłana w każdej początkowej (zerowej) szczelinie superramki, bez wykorzystania CSMA/CA. Zaraz po nadaniu beaconu rozpoczyna się okres aktywny superramki, podczas którego pozostałe urządzenia mogą skomunikować się z koordynatorem. Następnie może rozpocząć się opcjonalny okres nieaktywny, w czasie którego koordynator nie powinien podejmować interakcji z swoją siecią PAN, mogąc tym samym przejść w stan niskiego poboru energii. Okres aktywny superramki podzielony jest na okres rywalizacji o dostęp CAP (ang. *Contention Access Period*) oraz okres wolny od rywalizacji o dostęp CFP (ang. *Contention Free Period*). Każde urządzenie chcące się skomunikować podczas trwania okresu CAP powinno współrywalizować o dostęp z innymi urządzeniami, wykorzystując do tego algorytm szczelinowego CSMA/CA. Wyjątek tutaj stanowią ramki ACK, które mogą być transmitowane bezpośrednio po otrzymaniu wiadomości.

Natychmiast po szczelinie granicznej kończącej CAP, powinien rozpoczynać się okres CFP i trwać aż do samego końca aktywnego okresu superramki. Podczas tego okresu urządzenia nie muszą rywalizować o dostęp do kanału, ponieważ jest on im zapewniany przez koordynatora, w postaci szczelin GTS (ang. *Guaranteed Time Slot*). Koordynator PAN może zaalokować do siedmiu szczelin GTS, z których każda może zajmować więcej niż jedną szczelinę czasową superramki.

Ten paragraf wchodzi już dość mocno w szczegóły. Usunąć?

Szczelinami GTS zarządza wyłącznie koordynator PAN, alokując je na podstawie dostępnej pojemności superramki oraz otrzymanych wcześniej od urządzeń żądań przyznania GTS, rozpatrywanych w porządku first–come–first–serve. Każda szczelina gwarantowanego czasu jest opisana w pamięci koordynatora numerem szczeliny początkowej, czasem trwania, kierunkiem komunikacji (koordynator — urządzenie lub urządzenie — koordynator) oraz zaasocjowanym urządzeniem docelowym. Każde urządzenie może żądać jednego GTS transmisji i/lub jednego GTS odbioru. Nieużywane lub niepotrzebne szczeliny GTS mogą zostać zdealokowane, w dowolnym momencie, dyskretnie przez koordynatora lub na żądanie urządzenia.

1.1.2.4 CSMA/CA

Algorytm wielodostępu do nośnika z unikaniem kolizji CSMA/CA (ang. *Carrier Sense Multiple Access with Collision Avoidance*) występuje w dwóch wariantach: szczelinowym, stosowanym w sieciach wykorzystujących strukturę superramki oraz bezszczelinowym. W obu przypadkach realizacja algorytmu sprowadza się do odczekiwania losowego interwału czasowego, liczonego w backoff periods, po którego upływie sprawdzana jest zajętość kanału. W przypadku CSMA/CA szczelinowego, sekwencja odczekiwania i badania zajętości kanału realizowana jest kilkukrotnie. Jeśli po upływie wyznaczonego czasu kanał pozostał wciąż wolny, algorytm kończy się powodzeniem, umożliwiając rozpoczęcie transmisji.

1.1.2.5 Tworzenie sieci PAN

Tworzenie sieci PAN może być realizowane wyłącznie przez FFD, stającego się od tego momentu jedynym dopuszczalnym koordynatorem PAN. W celu utworzenie nowej sieci PAN koordynator musi wykonać skan aktywny wszystkich kanałów logicznych, polegający na wysyłaniu żądania emisji beaconów i oczekiwaniu ewentualnej odpowiedzi. Istnieje również możliwość przeprowadzenia skanu pasywnego, nasłuchujący beaconów innych koordynatorów PAN oraz mierzący ED. Po ukończeniu skanowania i określeniu wszystkich potencjalnych sąsiednich sieci PAN, wyniki badania przekazywane są do warstwy NWK, która podejmuje decyzje o wyborze unikalnego identyfikatora sieci PAN.

Dopiero po ukończeniu skanowania oraz wyborze identyfikatora PAN, koordynator może utworzyć nową sieć PAN, o czym informuje inne urządzenia za pomocą emisji ramek beaconów.

1.1.2.6 Asocjacja i deasocjacja

W celu asocjacji, podobnie jak podczas tworzenia nowej sieci PAN, urządzenie niezasocjowane przeprowadza skan aktywny lub pasywny wszystkich kanałów logicznych. Na podstawie wyników skanowania i po wybraniu przez warstwę aplikacji identyfikatora PAN sieci, do której urządzenie chce zostać przyłączone, wysyła ono do koordynatora PAN żądanie asocjacji. Po odesłaniu wiadomości ACK, koordynator ma określony czas na podjęcie decyzji o akceptacji bądź odrzuceniu żądania asocjacji. Jeśli w sieci znajduje się dostateczna ilość zasobów na dołączenie nowego urządzenia (m.in. pula adresowa), koordynator generuje krótki adres i odsyła go urządzeniu wraz z wiadomością o akceptacji asocjacji.

Deasocjacja może zostać zainicjowana zarówno na bezpośrednie żądanie urządzenia, jak i w wyniku decyzji podjętej przez koordynatora PAN, o czym urządzenie zdeasocjowane jest informowane.

Specjalnym przypadkiem deasocjacji jest osierocenie, pojawiające się w przypadku powtarzających się błędów komunikacji (brak zwrotnego ACK od koordynatora). W takiej sytuacji, urządzenie może zresetować warstwę MAC (adres, identyfikator PAN) i przeprowadzić ponownie próbę asocjacji. Drugą alternatywą jest podjęcie próby ponownego dołączenia urządzenia osieroconego. W tym procesie, w każdym kanale logicznym z określonego zbioru nadawana jest komenda notyfikacji osierocenia. Po odebraniu jej przez dotychczasowego rodzica, generowana jest odpowiedź zatwierdzająca ponowne dołączenie.

1.1.2.7 Transfer danych

W standardzie istnieją trzy typy transakcji: koordynator — urządzenie, urządzenie — koordynator oraz urządzenie — urządzenie. Mechanizm realizacji każdego z tych transferów zależy od wsparcia beaconów w sieci.

W sieci nieemitującej beaconów, wykorzystując bezszczelinowe CSMA/CA, urządzenie bezpośrednio dokonuje transmisji ramek danych. W sieci z beaconami, urządzenie chcące dokonać transferu danych, za pomocą beaconów synchronizuje się ze strukturą superramki w celu wyznaczenia właściwego momentu transmisji danych. W obu mechanizmach preferowane jest wykorzystywanie skróconego adresu 16–bitowego, zamiast adresu rozszerzonego 64–bitowego.

W sieciach peer–to–peer, gdzie każde urządzenie może komunikować się każdym bez kontroli koordynatora, możliwe są dwa rozwiązania: synchronizacja pomiędzy węzłami lub nieustanne nasłuchiwanie i wysyłanie przy wykorzystaniu CSMA/CA.

W celu zwiększenia oszczędności energii, został zdefiniowany mechanizm transmisji niebezpośredniej. Zgodnie z nim, dane zaadresowane dla konkretnego urządzenia są przechowywane przez koordynatora PAN. Informacje o oczekujących danych są rozsyłane w samoistnie emitowanych przez koordynatora beaconach lub beaconach wysłanych w odpowiedzi na żądanie urządzenia. Umożliwia to urządzeniom aktywację odbiornika radiowego tylko w momencie nadawania beacona przez koordynatora lub tylko w sytuacji kiedy urządzenie samo zdecyduje o pobraniu danych.

1.1.2.8 Formaty ramki MAC

Standard IEEE 802.15.4 definiuje dla warstwy MAC ogólny format ramki, zaprezentowany na rysunku 1.3. Format ten, w zależności od docelowej zawartości ramki i jej funkcji, może występować z jednej z czterech odmian:

1. ramka danych — służąca jedynie do transmisji danych, poza polami podstawowymi zawierająca jedynie blok danych
2. ramka beacona — służącą do emisji beaconów. Poza polami podstawowymi, zawiera w sobie informacje o specyfikacji superramki, konfiguracji szczelin GTS, informacje o wiadomościach oczekujących na urządzenia oraz pole przeznaczone na blok danych beacona

3. ramka ACK — przeznaczona do przekazywania wiadomości typu Acknowledgement, zawierającej jedynie pole kontrolne ramki, numer sekwencyjny wiadomości oraz jej sumę kontrolną
4. ramka komendy MAC — służąca do przekazywania komend przeznaczonych dla warstw MAC innych urządzeń (np. żądanie szczeliny GTS). Poza podstawowymi polami zawiera w sobie identyfikator komendy oraz blok danych do niej przypisany

<table>
<tr><th>Oktety: 2</th><th>1</th><th>0/2</th><th>0/2/8</th><th>0/2</th><th>0/2/8</th><th>0/5/6/10/14</th><th>zmienna</th><th>2</th></tr>
<tr><td rowspan="2">Pole kontrolne ramki</td><td rowspan="2">Numer sekwencyjny</td><td>Identyfikator docelowego PAN</td><td>Adres docelowy</td><td>Identyfikator źródłowego PAN</td><td>ID nadawcy</td><td rowspan="2">Pomocniczy nagłówek bezpieczeństwa</td><td rowspan="2">Blok danych ramki</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Pola adresowania</td></tr>
<tr><td colspan="7">MHR</td><td>Blok danych MAC</td><td>MFR</td></tr>
</table>

Rysunek 1.3.: Format ogólnej ramki podwarstwy dostępu do nośnika MAC.
Źródło: opracowanie własne na podstawie [oEEE11, str. 57]

Bazując na ogólnym formacie ramki, we wszystkich czterech z wymienionych typów można wyszczególnić pola wspólne, a zarazem obligatoryjne:

- MAC Header (MHR) — nagłówek ramki MAC, odpowiedzialny za przekazywanie informacji o formacie ramki, ustawieniach zabezpieczeń, numerze sekwencyjnym wiadomości oraz danych adresowe zarówno nadawcy jak i odbiorcy,
- MAC Payload — blok danych warstwy MAC, zawierający różne rodzaje danych w zależności od typu ramki,
- MAC Footer (MFR) — stopka ramki MAC, przechowująca sumę kontrolną wiadomości, obliczaną jako 16–bitowy cykliczny kod nadmiarowy.

1.1.3 Warstwa sieciowa

Warstwa sieciowa NWK specyfikacji ZigBee łączy w sobie warstwę sieciową oraz transportową standardu OSI. Zadaniami realizowanymi przez tę warstwę jest zapewnianie poprawności operacji przeprowadzanych przez podwarstwę MAC oraz eksponowanie warstwie aplikacji odpowiednich interfejsów dostępu do usług. Ponadto, warstwa NWK odpowiedzialna jest za routing wiadomości, konfigurację nowych urządzeń, zarządzanie przynależnością do sieci oraz nią samą, nadawanie adresów urządzeniom, odkrywanie sąsiadów oraz możliwych tras routingu.

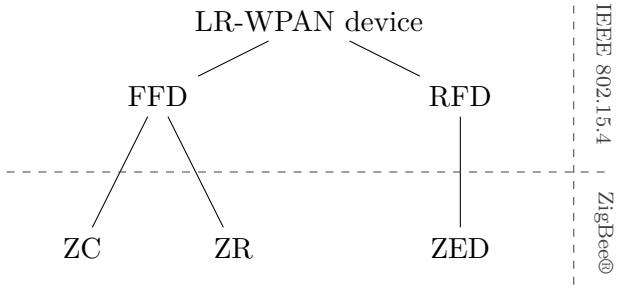
Ważnym składnikiem warstwy NWK jest baza NIB (ang. *NWK Information Base*), stanowiąca zbiór atrybutów i stałych niezbędnych do zarządzania warstwą sieciową, np. numer sekwencji wiadomości, tablica sąsiedztwa, 16–bitowy adres sieciowy, identyfikator PAN sieci oraz zmienne wykorzystywane przez algorytmy warstwy NWK.

1.1.3.1 Typy urządzeń tworzących sieć

Jak zostało wspomniane w poprzednich sekcjach dotyczących warstwy MAC (sekcja 1.1.2.1), zgodnie ze standardem IEEE 802.15.4 zdefiniowane są dwa typy urządzeń tworzących sieć LR-WPAN: Full-Functional Device (FFD) oraz Reduced-Functional Device (RFD). Na tej podstawie specyfikacja ZigBee® określa trzy rodzaje urządzeń wchodzących w skład sieci:

1. ZigBee® Coordinator (ZC) — urządzenie charakteryzujące się największymi możliwościami, bezpośrednio powiązane funkcyjnie z koordynatorem PAN standardu IEEE 802.15.4. Rolą koordynatora jest nadzorowanie pracy całej sieci, formowanie jej korzenia, przechowywanie wszystkich informacji o niej, pełniąc przy tym rolę Centrum Zaufania i repozytorium dla kluczy bezpieczeństwa
2. ZigBee® Router (ZR) — urządzenie FFD przekazujące dane pomiędzy innymi urządzeniami należącymi do sieci, rozszerzając tym samym jej zasięg. Ponadto, ZR zarządza lokalnym przydzielaniem adresów i tablicą sąsiedztwa.
3. ZigBee® End Device (ZED) — odpowiednik urządzenia RFD standardu IEEE 802.15.4, posiadający minimum funkcji wymaganych do komunikacji z urządzeniem nadrzędnym.

Podział ten nie jest całkowicie rozłączony. Bardzo często funkcje routingu oraz koordynatora są scalane w jednym urządzeniu. Zależność pomiędzy urządzeniami zdefiniowanymi przed standard IEEE 802.15.4 oraz urządzeniami z nich się wywodzącymi specyfikacji ZigBee® została zaprezentowana na rysunku 1.4.



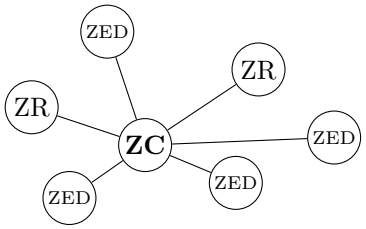
Rysunek 1.4.: Hierarchia typów urządzeń ZigBee® i standardu IEEE 802.15.4.
Źródło: opracowanie własne na podstawie [oEEE11, str. 8–9][ZA08, str. 3]

1.1.3.2 Topologie sieci

Protokół ZigBee®, opierając się na warstwie MAC standardu IEEE 802.15.4, umożliwia utworzenie sieci bezprzewodowej w jednej z trzech topologii: gwiazdy, drzewa oraz kraty, przy czym preferowaną topologią jest ostania z wymienionych.

Topologia gwiazdy

Topologia gwiazdy jest jedną z najprostszych a zarazem najbardziej ograniczonych topologii standardu ZigBee®. Składa się ona ze zbioru urządzeń ZC i ZED, które mogą komunikować się jedynie poprzez centralnego koordyna-

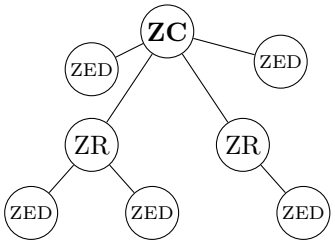


Rysunek 1.5.: Topologia gwiazdy

tora sieci. W przypadku więc awarii centralnego urządzenia, z braku alternatywnych dróg komunikacji działanie całej sieci zostaje zawieszone. Co więcej, centralny koordynator stanowi wąskie gardło dla wszystkich dróg komunikacji w sieci. Ze względu na implementację tej topologii w warstwie MAC, warstwa NWK nie musi brać udziału w procesie przekazywania wiadomości.

Topologia drzewa

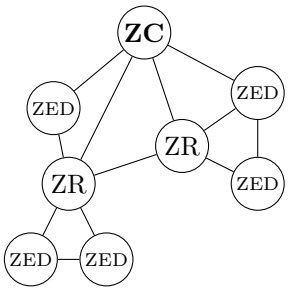
Topologia drzewa składa się z koordynatora PAN, stanowiącego korzeń drzewa, którego potomkami mogą być urządzenia ZC lub ZR, posiadające kolejne poziomy potomków. Urządzenia ZED występują w przypadku tej topologii wyłącznie w formie liści dowolnego poziomu drzewa. Podobnie jak w miało to miejsce w topologii gwiazdy, komunikacja pomiędzy liśćmi odbywać się może jedynie przez rodziców, co w niektórych sytuacjach może prowadzić do przesłania wiadomości wykorzystując korzeń drzewa i przebywając dwukrotnie jego wysokość. Awaria połączenia pomiędzy urządzeniami może nieść skutki o wiele poważniejsze niż było to w przypadku topologii gwiazdy, uniemożliwiając komunikację całej gałęzi z resztą sieci.



Rysunek 1.6.: Topologia drzewa

Topologia kraty

Topologia kraty jest rozwinięciem topologii drzewa, umożliwiającym bezpośrednią komunikację liści drzewa, tworząc w ten sposób gęstą sieć połączań. Taka realizacja struktury połączeń w sieci zwiększa efektywność propagacji wiadomości, umożliwiając wykorzystanie więcej niż jednej trasy komunikacji pomiędzy urządzeniami. Ponadto, istnienie alternatywnych ścieżek wymiany danych zwiększa odporność sieci na ewentualne awarie poszczególnych jej połączeń. Topologia ta musi być zaimplementowana programowo w warstwie NWK, jako że standard IEEE 802.5.4 nie przewidział jej w podwarstwie MAC.



Rysunek 1.7.: Topologia kraty

1.1.3.3 Adresowanie

W sieciach ZigBee® każde urządzenie posiada przypisane dwa typy adresów: rozszerzony, 64–bitowy adres IEEE oraz adres skrócony, 16–bitowy.

Adres rozszerzony, zdefiniowany przez standard IEEE 802.15.4 w podwarstwie MAC, jest przypisywany przez wytwórcę w momencie produkcji danego urządzenia. Adres ten musi umożliwiać jednoznaczną identyfikację urządzenia w skali światowej, wymagając tym samym unikalności adresu względem wszystkich urządzeń standardu IEEE 802.15.4.

Adres skrócony, zdefiniowany przez specyfikację ZigBee®, jest unikalny jednie w obrębie sieci PAN do której urządzenie przynależy. Jest on nadawany przez urządzenie ZR lub ZC, które jako rodzic zajmuje się przyłączeniem nowego urządzenia do istniejącej sieci.

Przydział nowych adresów 16–bitowych jest realizowany zgodnie z jednym z dwóch mechanizmów, opisanych w poniższych paragrafach. Wybór mechanizmu zależy od wartości wpisu w bazie NIB *nwkAddrAlloc*.

Mechanizm adresowania rozproszonego

Mechanizm ten jest używany w celu zapewnienia każdemu potencjalnemu rodzicowi skończonej puli adresów sieciowych dla urządzeń tworzących jego podbloki. ZC określa maksymalną liczbę potomków dla każdego urządzenia C_m, maksymalną liczbę routerów R_m oraz maksymalną głębokość sieci L_m, liczoną w przeskokach pomiędzy koordynatorem a urządzeniem. Parametry te są wykorzystywane podczas obliczania współczynnika *Cskip(d)* (Równanie 1.1), stanowiącego rozmiar puli adresów do dyspozycji danego rodzica, dystrybuowanej wśród jego dzieci typu ZR, znajdujących się na danym poziomie głębokości sieci, oznaczanej jako d.

$$Cskip(d) = \begin{cases} 1 + C_m(L_m - d - 1), & \text{jeśli } R_m = 1 \\ \frac{1+C_m-R_m-C_m*R_m^{L_m-d-1}}{1-R_m}, & \text{w pozostałych przypadkach} \end{cases} \tag{1.1}$$

W przypadku, gdy $Cskip = 0$, urządzenie powinno być traktowane jako ZED, bez możliwości posiadania potomków. Znając $Cskip$ rodzica oraz jego adres A_{parent}, można obliczyć adres sieciowy jego potomka dołączającego do sieci. $Cskip$ staje się wtedy wartością przesunięcia nowego adresu, zgodnie ze wzorem 1.2.

$$A_n = A_{parent} + Cskip(d)R_m + n \tag{1.2}$$

gdzie $1 \leq n \leq C_m - R_m$. Dla urządzenia ZED, adresy powinny być przydzielane w sposób sekwencyjny, z adresem n–tym . Jeśli rodzic wyczerpie przydzieloną mu pulę adresową, nie powinien pozwalać nowemu urządzeniu na asocjację. Przykładowy przydział adresów w sieci został zaprezentowany na rysunku 1.8, z parametrami $C_m = 8$, $R_m = 4$ oraz $L_m = 3$.

Zdecydowanym walorem tego mechanizmu adresowania jest łatwość w określeniu adresu rodzica urządzenia dla którego przeznaczona jest dana wiadomość.

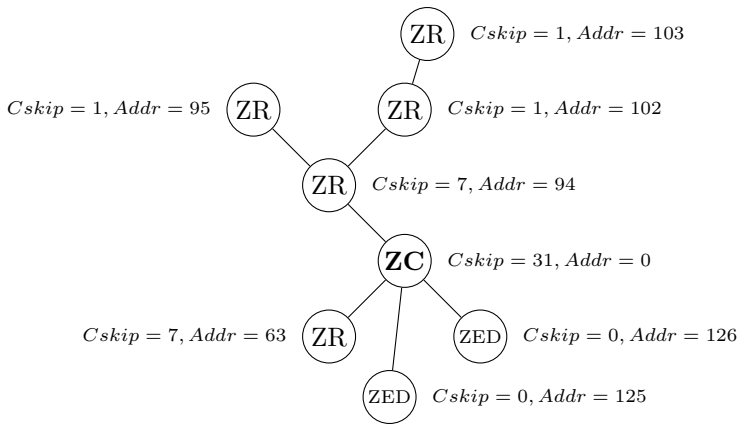
Mechanizm adresowania stochastycznego

Losowy przydział adresów dla nowych urządzeń w sieci realizowany jest zgodnie ze schematem NIST[2]. Jedynym wymogiem dotyczącym wygenerowanego adresu jest jego unikalność w obrębie tablicy sąsiedztwa rodzica asocjowanego urządzenia.

1.1.3.4 Tablica sąsiedztwa

Tablica sąsiedztwa, przechowywana i zarządzana przez warstwę NWK, powinna zawierać informacje o każdym urządzeniu będącym w zasięgu transmisji danego urządzenia ZC lub ZR. Tablica ta jest wykorzystywana zarówno podczas odkrywania sieci i przyłączania się do niej, jak i do przechowywania informacji i relacji zachodzących z sąsiednimi urządzeniami

[2]Strona Centrum Zasobów Bezpieczeństwa Komputerowego, http://csrc.nist.gov/groups/ST/toolkit/rng/index.html, 2013



Rysunek 1.8.: Przykładowy przydział adresów w sieci ZigBee®
Źródło: opracowanie własne na podstawie [ZA08, str. 372]

w sieci. Dane zachowane w tablicy sąsiedztwa powinny by aktualizowane każdorazowo po otrzymaniu ramki z wiadomością od sąsiada.

Informacjami zawartymi w tablicy sąsiedztwa jest zarówno adres 64–bitowy, jak i adres 16–bitowy, typ urządzenia, relacja z urządzeniem, wskaźniki transmisji (ilość błędów, LQI) oraz czas ostatniej aktualizacji.

1.1.3.5 Routing

W celu optymalizacji komunikacji z sieciach ZigBee® stosowany jest routing, którego ścieżki przechowywane są w specjalnych tablicach routingu urządzeń typu ZC oraz ZR. W tablicach tych zapisywane są m.in. informacje od adresie urządzenia do którego dana ścieżka prowadzi, status ścieżki, koszt (najczęściej liczony jako LQI) oraz adres urządzenia następnego przeskoku na drodze do celu. Korzystając ze ścieżek przechowywanych w tablicach routingu, urządzenia posiadające tzw. zdolność routingu (ang. *Routing Capacity*), przekazują odebraną wiadomość wzdłuż ścieżki prowadzącej do urządzenia o danym adresie docelowym lub jego rodzica w przypadku sieci o adresowaniu rozproszonym i urządzeniu docelowym typu ZED. W sytuacji gdy zarówno w tablicy routingu jak i tablicy sąsiedztwa nie istnieje wpis odpowiadający urządzeniu dla którego jest przeznaczona wiadomość lub wpis uległ przedawnieniu, inicjowany jest algorytm odkrywania trasy.

Odkrywanie trasy

Odkrywanie trasy (ang. *Route Discovery*) jest procesem w czasie którego urządzenia należące do sieci podejmują współpracę w celu znalezienia i ustalenia tras w warstwie NWK, przeznaczonych dla różnych typów nadawania:

- *Unicast Route Discovery* — jest realizowane z uwzględnieniem pojedynczego urządzenia źródłowego oraz pojedynczego urządzenia docelowego

- *Multicast Route Discovery* — jest realizowane biorąc pod uwagę pojedyncze urządzenie źródłowe oraz docelową grupę urządzeń
- *Many–to-one Route Discovery* — jest realizowane przez pojedyncze źródłowe urządzenie, w celu ustalenia tras do niego samego ze wszystkich ZR oraz ZC w obrębie danego promienia. Urządzenie źródłowe staje się koncentratorem.

Ramka komendy żądania routingu zostaje rozpropagowana w całej sieci, pośród urządzeń typu ZC i ZR, poszukujących w swoich tablicach sąsiedztwa lub tablicach routingu urządzenia docelowego. Po jego odnalezieniu, zostaje odesłana odpowiedź do urządzenia inicjującego proces odkrywania trasy, przy użyciu której urządzenia ją przekazujące aktualizują swoje wpisy w tablicach routingu. Wartości pośrednie z opisanego procesu są przechowywane w tablicy odkrywania trasy.

Wszystkie trasy oraz komendy żądania odkrywania trasy mają nadane numery sekwencyjne, dzięki którym możliwe jest uniknięcie nadmiernego ruchu w sieci oraz powstania tras zawierających pętle.

Algorytm odkrywania trasy został tutaj opisany w sposób bardzo skrótowy, aczkolwiek dający ogólny obraz algorytmu i wystarczający na potrzeby tego opracowania. Dokładną specyfikację przebiegu procesu odkrywania trasy znaleźć można w dokumentacji ZigBee® oraz w dokumentacji algorytmu AODV, z którego opisany algorytm bezpośrednio się wywodzi [ZA08, str. 422–433][Per03].

Wiadomości statusu połączenia

Połączenia w sieci bezprzewodowej mogą być asymetryczne, z czym związane jest ryzyko pojawienia się błędów podczas transmisji w jednym z kierunków. W konsekwencji może spowodować to niepowodzenie przekazywania odpowiedzi komendy żądania routingu. Aby zapobiec takim sytuacjom, ZR okresowo wymieniają ze swoimi sąsiadami pomiary kosztu połączeń za pomocą nadawania ramek statusu połączenia metodą rozgłaszania w promieniu jednego przeskoku. Zwrotna informacja o koszcie jest wykorzystywana podczas odkrywania trasy w celu zapewnienia użycia wysokiej jakości połączeń w obu kierunkach.

1.1.3.6 Zarządzanie przynależnością do sieci

Podstawowym elementem konstrukcyjnym sieci LR-WPAN jest relacja rodzic-potomek, formowana podczas dołączania nowego urządzenia do sieci, które staje się potomkiem urządzenia nadzorującego dołączenie. Proces dołączania należy do podwarstwy MAC, zarządzany jest on jednak przez warstwę NWK.

W celu dołączenia do istniejącej sieci urządzenie dokonuje odkrywania sieci oraz jej wyboru. Następnie, urządzenie wysyła żądanie dołączenia do sieci i wybiera odpowiedniego rodzica z tablicy sąsiedztwa, z którym podejmuje próbę asocjacji. Jeśli rodzic uzna, iż warunki na przyłączenie są odpowiednie, jest generowana i przesyłana odpowiedź akceptacyjna zawierająca 16–bitowy adres sieciowy dla nowego urządzenia. Nowy potomek może rozpocząć synchronizację z rodzicem (superramka, kanał, identyfikator PAN). Tworzone są też odpowiednie wpisy w tablicach sąsiedztwa obu urządzeń.

W sytuacji utraty połączenia z siecią przeprowadzana jest procedura ponownego dołączania, zbliżona do procesu asocjacji. Jedyną różnicą jest zamiana asocjacji MAC na

wymianę żądań typu *rejoin*. Modyfikacja ta pozwala na ponowne dołączenie do sieci odrzucającej nowe urządzenia. Specjalnym przypadkiem utraty połączenia jest osierocenie, którego procedura rozwiązania jest identyczna z tą opisaną przy okazji przybliżenia specyfikacji podwarstwy MAC.

Istnieje również możliwość bezpośredniego dołączania do sieci. Opiera się ono na predefiniowaniu w urządzeniu ZR lub ZC rozszerzonego, 64–bitowego adresu urządzenia dołączającego. Procedurę bezpośredniego dołączania może zainicjować jedynie ZR lub ZC, które alokują 16–bitowy adresy w tablicy sąsiedztwa, przeznaczony dla nowego urządzenia.

1.1.3.7 Rozgłaszanie

Każde urządzenie znajdujące się w sieci może zainicjować rozgłaszanie wiadomości (ang. *Broadcasting*) do innych urządzeń sieci. Odbiorcami rozgłaszania mogą być wszystkie urządzenia należące do sieci PAN (adres rozgłoszeniowy *0xffff*), wszystkie urządzenie ZR i ZC (adres rozgłoszeniowy *0xfffc*) lub wszystkie routery małej mocy (adres *0xfffc*). Podczas rozgłaszania nie jest stosowana wiadomość ACK pochodząca z podwarstwy MAC, zamiast której wykorzystywany jest pasywny ACK, w którym ZC i ZR śledzą który z sąsiadów przekazał wiadomość. Wszystkie wiadomości rozgłaszane powinny być przechowywane przez określony czas przez wszystkie urządzenia sieci w celu zapobiegania zdublowaniu odebranej wiadomości.

1.1.3.8 Format ramki NWK

Specyfikacja ZigBee® definiuje ogólny format ramki (Rysunek 1.9), na podstawie którego zdefiniowane są ramki specjalnych zastosowań, jak ramka danych czy ramka komendy.

Oktety: **2**	**2**	**2**	**1**	**1**	**0/8**	**0/8**	**0/1**	**zmienna**	**zmienna**
Pole kontrolne ramki	Adres docelowy	Adres źródłowy	Promień	Numer Sekwencyjny	Adres docelowy IEEE	Adres źródłowy IEEE	Kontrola Multicastu	Podramka routingu	Blok danych ramki
Nagłówek NWK									Blok danych

Rysunek 1.9.: Format ogólnej ramki warstwy NWK
Źródło: opracowanie własne na podstawie [ZA08, str. 307]

Ramka warstwy NWK składa się z bloku danych i nagłówka, w którym zawarte są informacje o adresie docelowym ramki w wersji rozszerzonej i skróconej, adresie źródłowym, zasięgu wiadomości, numerze sekwencyjnym oraz przekazywana jest podramka routingu.

1.1.4 Warstwa aplikacji

Warstwa aplikacji (APL) jest najwyższą warstwą zdefiniowaną przez specyfikację ZigBee®, udostępniającą interfejs dostępu do systemu końcowemu użytkownikowi. Elementami składowymi warstwy jest podwarstwa wsparcia aplikacji (APS ang. *Application Support Sub–Layer*), obiekt urządzenia ZigBee (ZDO ang. *ZigBee Device Object*) oraz obiekty aplikacji zdefiniowane przez wytwórcę, umieszczone we frameworku aplikacji. Każdy obiekt aplikacji posiada określony komponent specyfikujący jego funkcjonalności, nazywany punktem

końcowym (ang. *endpoint*), który poprzez APSDE–SAP (ang. *Application support sub–layer data entity - service access point*) pozwala na wymianę danych z podwarstwą APS, a co za tym idzie, komunikację z innymi urządzeniami i ich endpoint–ami (Rysunek 1.10). Związku z tym, wiadomość może zostać zaadresowana do konkretnego punktu końcowego lub do całego urządzenia.

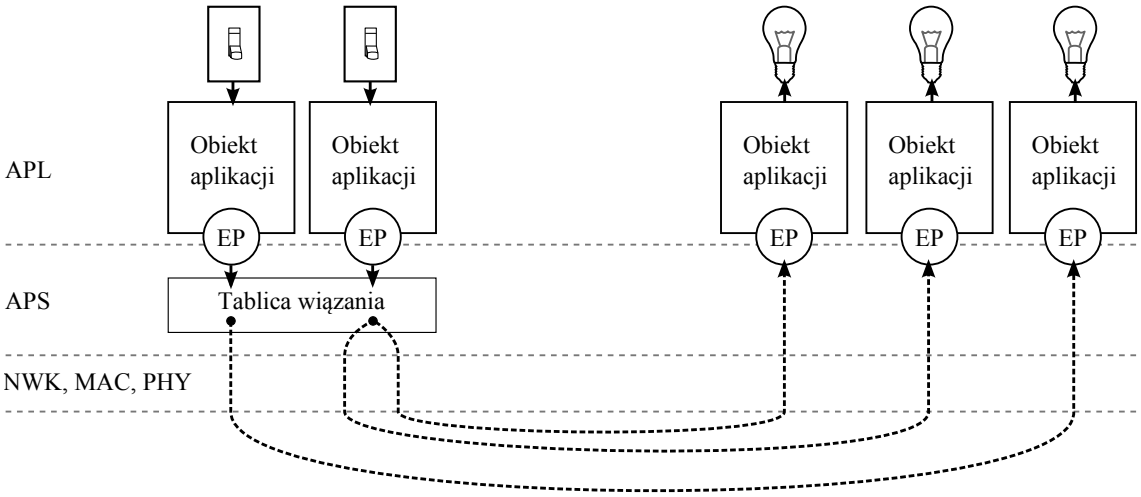
1.1.4.1 Podwarstwa wsparcia aplikacji

Podwarstwa APS zapewnia interfejs pomiędzy warstwą sieciową stosu (NWK) oraz warstwą aplikacji (APL) poprzez ogólny zestaw usług, używanych zarówno przez obiekt ZDO, jak i obiekty aplikacji zdefiniowane przez wytwórcę. Zadaniami realizowanymi przez podwarstwę APS jest generowanie ramek warstwy aplikacji, wiązanie, zarządzenie grupowym adresowaniem wiadomości, fragmentacja wiadomości oraz zapewnianie bezpieczeństwa.

Podobnie jak miało to miejsce w przypadku warstwy NWK, podwarstwa APS zawiera specjalną bazę z informacjami niezbędnymi do działania warstwy aplikacji. Przechowywane są w niej dane tablicy wiązania, tablicy grup czy informacje na temat używanego kanału radiowego.

Wiązanie

Wiązanie (ang. *Binding*) pozwala urządzeniom ZigBee® na ustalenie odpowiedniego odbiorcy docelowego dla ramek pochodzących z danego punktu końcowego. Odbiorcą ramki powinien być określony punkt końcowy konkretnego urządzenia lub adres grupowy. Można zatem stwierdzić, iż wiązanie pozwala na utworzenie trwałego połączenia pomiędzy dwoma obiektami aplikacji i późniejszą wymianę danych pomiędzy nimi, w sposób transparentny dla aplikacji użytkownika. Zarys działania wiązania został przedstawiony na rysunku 1.10.



Rysunek 1.10.: Komunikacja pomiędzy punktami końcowymi (EP) różnych urządzeń oferujących różne funkcjonalności.
Źródło: opracowanie własne na podstawie [?, rozdz. 2]

Adresowanie grupowe

Jedną z funkcji podwarstwy APS jest zarządzanie tablicą grup, pozwalającą na skojarzenie poszczególnych punktów końcowych z określonymi grupami oraz na selektywne dostarczanie ramek zaadresowanych grupowo do odpowiednich punktów końcowych. W praktyce oznacza to możliwość jednoczesnego zawiadywania kilkoma różnymi obiektami aplikacji przez inny pojedynczy obiekt, co zostało pokazane na rysunku 1.10. Tablica grup zarządzana przez podwarstwę APS powinna być zgodna z tablicą grup zawartą w warstwie NWK.

1.1.4.2 Obiekt urządzenia ZigBee®

Obiekt urządzenia ZigBee® (ZDO) reprezentuje bazową klasę funkcjonalności, zapewniającą interfejs pomiędzy obiektami aplikacji we frameworku aplikacji, profilem urządzenia oraz APS, umożliwiając tym samym zarządzanie urządzeniem i siecią poprzez obiekty aplikacji. Kontrola ta realizowana jest przez różnego rodzaju menadżery:

- *Device and Servie Discovery* — funkcja wspierająca odkrywanie urządzeń i usług wewnątrz pojedynczej sieci PAN. Ponadto, funkcja ta pozwala na odkrywanie urządzenia będącego w stanie niskiego poboru energii
- *Security Manager* — funkcja odpowiedzialna za bezpieczeństwo, zarządzanie kluczami, uwierzytelnienie oraz kontakt z Centrum Zaufania
- *Network Manager* — funkcja pozwalające na zarządzanie przynależnością do istniejącej sieci PAN
- *Binding Manager* — umożliwia zarządzanie tablicą wiązania
- *Node Manager* — udostępnia komendy zdalnego zarządzania odkrywaniem sieci, odbiorem tablicy routingu i tablicy wiązania. Pozwala na realizację decyzji o opuszczeniu sieci bądź zlecenie tej akcji innemu urządzeniu
- *Group Manager* — zapewnia dołączanie do grup i ich opuszczanie przez obiekty aplikacji

Jednocześnie, ZDO pozwala innym urządzeniom na odkrywanie usług oferowanych przez obiekty aplikacji korzystające z niego.

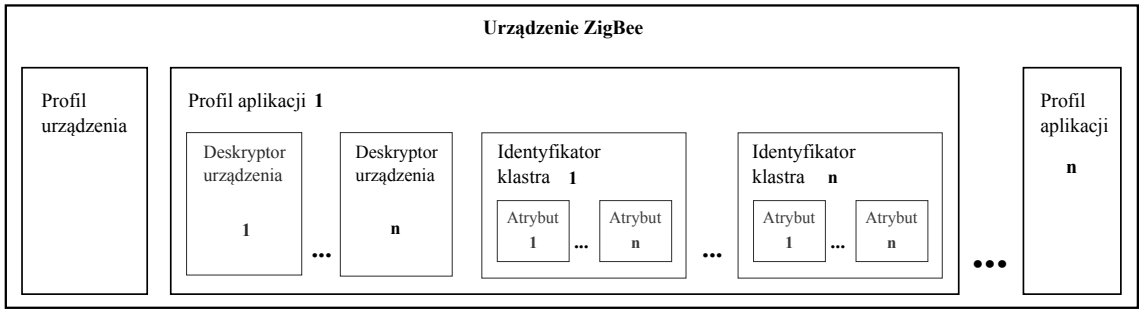
Odkrywanie usług

Obiekt ZDO umożliwia odkrywanie usług, czyli możliwości innych urządzeń. Proces ten może zostać zrealizowany przez sekwencyjne odpytywanie wszystkich punktów końcowych danego urządzenia lub przez użycie usługi dopasowywania. Dzięki odkrywaniu usług innych urządzeń, a w konsekwencji, ich wzajemnego dopasowania, możliwe jest automatyczne utworzenie sieci oraz jej konfiguracja.

1.1.4.3 Framework aplikacji

Framework aplikacji jest środowiskiem hostującym obiekty aplikacji zdefiniowane przez wytwórcę wraz z przypisanymi do nich punktami końcowymi. Zadaniem frameworku jest

filtrowanie ramek przychodzących z podwarstwy APS i prezentowanie jedynie ramek mogących znajdować się w obszarze zainteresowania aplikacji zaimplementowanych przy każdym aktywnym punkcie końcowym. Decyzja o przekazaniu wiadomości podejmowana jest na podstawie zadeklarowanych profili aplikacji i wynikającej z nich zdolności do odbioru i przetwarzania określonych typów wiadomości. Z każdym z profili skojarzony jest zestaw deskryptorów urządzenia oraz identyfikatorów klastra. Tymczasem, każdy klaster jest kontenerem dla jednego lub więcej atrybutów. Relacje te zostały przedstawione na rysunku 1.11.



Rysunek 1.11.: Diagram relacji zachodzących pomiędzy abstrakcyjnymi elementami opisującymi urządzenia ZigBee®
Źródło: opracowanie własne na podstawie [ZA08, podr. 2.3]

Profil aplikacji

Profil aplikacji jest zgodnością wiadomości, formatów wiadomości oraz przetwarzania akcji, umożliwiającą stworzenie iteroperacyjnych aplikacji. Na podstawie zgodności profili aplikacji, urządzenia określają czy możliwa jest pomiędzy nimi komunikacja na poziomie aplikacji, a w konsekwencji, realizacja zadań stawianych urządzeniom. Ponieważ każde urządzenie może posiadać więcej niż jeden profil, urządzenia są zdolne do wykonywania różnorodnych zadań zachowując przy tym interoperacyjność.

Każdy z profili może być zbiorem deskryptorów urządzenia oraz klastrów. Deskryptory opisują urządzenie ZigBee® pod względem jego możliwości, typu, rodzaju zasilania. Dodatkowo, zawierają opis punktów końcowych urządzenia, identyfikatory profilu aplikacji, identyfikatory klastrów oraz dane bardziej skomplikowane, przekazywane w formie XML, np. opis urządzenia lub ikonka urządzenia w systemie.

Klastry są wiadomościami aplikacyjnymi, które mogą być kontenerami dla jednego lub więcej atrybutów, stanowiących swoiste parametry przekazywanych komunikatów. Dzięki klastrom możliwa jest wymiana wiadomości i prawidłowa interpretacja pomiędzy różnymi urządzeniami należącymi do sieci.

Profile aplikacji oraz punkty końcowe pozwalają na rozszerzenie podstawowych funkcji urządzeń oraz ich aktualizację, zachowując jednocześnie kompatybilność wsteczną.

Profil urządzenia

Profil urządzenia jest zbiorem klastrów definiujących funkcje obligatoryjnie wspierane przez wszystkie urządzenia ZigBee®. Profil urządzenia wspiera kluczowe funkcje komunikacji pomiędzy urządzeniami:

- *Device and Service Discovery* — zdolność do odkrywania urządzeń oraz usług przez nie oferowanych
- *End Device Bind, Bind and Unbind* — zdolność do dodawania i usuwania wpisów z tablicy wiązania oraz wsparcie dla uproszczonej metody wiązania, wymagającej udziału użytkownika (np. naciśnięcie przycisku)
- *Network Management* — zarządzanie siecią i przynależnością do niej

Dzięki zdefiniowaniu profilu urządzenia, każde urządzenie należące do sieci ma zagwarantowaną minimalną funkcjonalność, umożliwiającą podstawową kooperację z innymi urządzeniami (np. routing, odkrywanie sieci).

1.1.5 Bezpieczeństwo

Specyfikacja protokołu ZigBee® zakłada wykorzystanie modelu „otwartego zaufania", w którym poszczególne warstwy ufają sobie wzajemnie, chroniąc kryptograficznie jednie komunikację pomiędzy urządzeniami. Do tego celu wykorzystywane są 128–bitowe klucze symetryczne, współdzielone zarówno przez nadawcę jak i odbiorcę wiadomości. Zadaniem kluczy jest jednocześnie szyfrowanie bloku danych przekazywanej wiadomości oraz zapewnianie jej integralności i wykrycie ewentualnych modyfikacji. Klucze mogą być współdzielone pomiędzy jedynie dwoma urządzeniami biorącymi udział w komunikacji — *Link key*, podobnie jak możliwe jest wykorzystanie klucza globalnego — *Network key* — używanego przez wszystkie urządzenia w sieci. Dystrybucja kluczy może być zrealizowana poprzez ich preinstalację, ustalenie w procesie negocjacji lub przekazanie z Centrum Zaufania (ang. *Trust Center*), będącego najczęściej koordynatorem sieci. Poza funkcją repozytorium kluczy, Centrum Zaufania stosuje usługi bezpieczeństwa w celu konfiguracji oraz autoryzacji urządzeń w sieci.

1.1.6 Pozostałe specyfikacje ZigBee®

ZigBee® poza definicją bazowego protokołu komunikacji dla bezprzewodowych sieci kratowych, oferuje również specyfikacje o wyższym stopniu wyspecjalizowania.

1.1.6.1 RF4CE

Specyfikacja RF4CE jest przeznaczona dla prostej, dwukierunkowej kontroli w sieci typu urządzenie-urządzenie, niewymagającej pełnych możliwości sieci kratowej [ZA10]. Komunikacja pomiędzy urządzeniami wykorzystuje jedynie dwa, niezachodzące na siebie kanały radiowe. Standard ten oferuje mniejsze wymagania co do rozmiaru pamięci i mocy urządzeń, umożliwiając tym samym obniżenie kosztów implementacji. Zgodnie z tą specyfikacją, w sieci PAN można wyróżnić dwa typu urządzeń:

- Węzeł docelowy (TN, ang. *Target Node*) — jest węzłem będącym koordynatorem PAN, mogącym zainicjalizować tworzenie sieci
- Węzeł kontrolera (CN, ang. *Controller Node*) — jest rodzajem pilota zdalnego sterowania, przyłączającym się do sieci PAN utworzonej przez węzeł docelowy i kontrolującym urządzenie dołączone do niego

Pomiędzy poszczególnymi sieciami PAN może występować komunikacja. Podobnie jak miało to miejsce w ZigBe® Specyfication, urządzenia są parowana na podstawie dopełniania oraz zgodności funkcjonalności, tak więc np. pilot zdalnego sterowania będzie starał się odnaleźć telewizor.

1.1.6.2 ZigBee® IP Specification

ZigBee® IP Specifiaction jest pierwszym otwartym standardem bezprzewodowych sieci kratowych, bazującym na IPv6. Standard ten umożliwia dostęp do internetu i kontrolę tanich urządzeń o małej mocy i zróżnicowanej architekturze. Urządzenia wykorzystujące tę specyfikację mogą uzyskać dostęp do Ethernetu, Wi-Fi, HomePlug oraz urządzeń do nich podłączonych, wykorzystując do tego protokoły takie jak 6LoWPAN, IPv6, PANA, RPL, TCP, TLS czy UDP.

1.2 Z–Wave

Standard Z–Wave definiuje protokół bezprzewodowy, przeznaczony do komunikacji w sieciach kratowych charakteryzujących się niewielkimi przepływnościami. Głównym celem protokołu jest zapewnienie wymiany krótkich wiadomości sterujących pomiędzy jednostką kontrolującą a jednym lub kilkoma węzłami sieci. Specyfikacja protokołu definiuje stos składający się z czterech warstw sieciowych, opisanych dokładniej w poniższych podsekcjach oraz dokumentacji Z–Wave [Zen07][Zen06]. Warstwa fizyczna protokołu oraz warstwa dostępu do nośnika zostały zdefiniowane w standardzie na podstawie rekomendacji ITU–T G.9959 [ITU12].

Protokół Z–Wave został stworzony przez firmę Zen–Sys, wykupioną następnie przez Sigma Designs wraz ze wszystkimi własnościami intelektualnymi. Zarówno wykorzystanie komercyjne standardu, jak i dostęp do jego pełnej dokumentacji wymaga przynależności do Z–Wave Alliance[3], obciążanej rocznym abonamentem wysokości $3500. Z tego też względu specyfikacja standardu zamieszczona w niniejszej pracy nie obejmuje wszystkich technologicznych niuansów, skupiając się jedynie na ogólnym koncepcie protokołu.

1.2.1 Warstwa fizyczna

Urządzenia Z–Wave komunikują się za pośrednictwem fal radiowych o częstotliwościach znajdujących się w paśmie ISM (ang. *Industrial, Scientific, Medical*), uzależnionym od przepisów obowiązujących w danym kraju (Tablica 1.2). Transmisja odbywa się z prędkością do 100 Kbps, przy wykorzystaniu kluczowania częstotliwości i kodowania Manchester. Maksymalny obsługiwany rozmiar sieci Z–Wave wynosi 232 węzły, co przy zasięgu dochodzącym do 30 metrów i możliwości routingu pozwala na pokrycie znacznego obszaru.

1.2.1.1 Rodzaje węzłów

Protokół Z–Wave definiuje dwa podstawowe rodzaje urządzeń: urządzenia kontrolujące (ang. Controller Nodes), które zajmują się inicjacją komend sterujących i przekazywaniem ich innym urządzeniom oraz węzły podległe (ang. *Salve Nodes*), wykonujące odebrane

[3]Oficjalna strona Z–Wave Alliance http://www.z-wavealliance.org/, 2014

Tablica 1.2.: Częstotliwości stosowane przez Z–Wave

Obszar	**Częstotliwość**
Europa	868.42 MHz
USA	908.42 MHz
Hong Kong	919.82 MHz
Australia i Nowa Zelandia	921.42 MHz

rozkazy. Każde urządzenie należące do sieci posiada możliwość przekazywania komend pochodzących od innych urządzeń, co umożliwia kontrolerowi sterowanie urządzeniami znajdującymi poza bezpośrednim jego zasięgiem.

Urządzenia kontrolujące

Kontroler jest urządzeniem przekazującym rozkazy innym urządzeniom w sieci, związku z czym musi on posiadać pełną, aktualną tablicę routingu. Główny kontroler w sieci (ang. *Primiary Controller*), który ją inicjuje, posiada wyłączne prawo przyłączania oraz wykluczania pozostałych węzłów z sieci. Z tego też względu wymagana jest znajomość aktualnej topologi sieci. Pozostałe urządzenia kontrolujące, określane jako drugorzędne (ang. *Secondary Controllers*), posiadają jedynie prawo nadawania rozkazów w sieci. W zależności od funkcji oraz mobilności, można wyróżnić kilka rodzajów kontrolerów:

- *Portable Controller* — kontroler mogący się przemieszczać, zmieniając tym samym swoją pozycję w sieci, co w konsekwencji wymaga estymacji aktualnej lokalizacji w celu obliczenia najszybszej trasy routingu dla nadawanych wiadomości.
- *Static Controller* — kontroler będący urządzeniem stacjonarnym, posiadającym dostęp do stałego źródła zasilania. Kontroler statyczny może posiadać ponadto funkcję dystrybucji zmian w topologii w sieci otrzymanych od kontrolera głównego. Przykładem takiego urządzenia może być brama Internetowa monitorująca system Z–Wave.
- *Installer Controller* — rodzaj kontrolera przenośnego, posiadający dodatkową funkcjonalność zarządzenia siecią oraz testowania jakości połączeń
- *Bridge Controller* — rozszerzona wersja kontrolera statycznego, umożliwiająca stworzenie mostka pomiędzy siecią Z–Wave a innymi rodzajami sieci, np. UPnP. Kontroler ten umożliwia nadzór maksymalnie 128 wirtualnych węzłów podległych, utworzonych z istniejących węzłów należących do innej sieci.

Urządzenia podległe

Urządzenia podległe (ang. *Slaves*) są węzłami sieci Z–Wave odbierającymi i wykonującymi akcje przypisane do odebranych komend. Węzły te nie są zdolne do inicjacji komunikacji z innymi urządzeniami bez wcześniejszego odebrania od kontrolera komendy rozkazu transmisji . Zgodnie ze standardem, można rozróżnić dwa rozszerzenia urządzenia typu *Slave*:

- *Routing Slave* — urządzenie posiadające ogólną funkcjonalność węzła podległego, rozbudowane o możliwość przechowywania statycznych tras routingu i przekazywania z ich pomocą wiadomości, nie poprzedzonych komendą ich żądania (ang. *unso-*

licited messages. Przykładem takiego urządzenia może być termostat lub pasywny czujnik ruchu na podczerwień.

- *Enhanced Slave* — węzeł posiadający funkcjonalność *Routing Slave*, rozszerzoną o zegar czasu rzeczywistego oraz pamięć EEPROM. Przykładem takiego urządzenia może być stacja pogodowa.

1.2.2 Warstwa kontroli dostępu do nośnika

Warstwa dostępu do nośnika (MAC) zdefiniowana w standardzie Z-Wave jest niezależna od wykorzystanej warstwy fizycznej, wymaga jednak bezpośredniego dostępu do odebranej ramki i binarnych danych pochodzących z kanału. Drugi z wymienionych wymogów jest związany z zaimplementowanym w warstwie MAC algorytmem unikania kolizji.

1.2.2.1 Unikanie kolizji

Mechanizm unikania kolizji (CSMA/CA) zapobiega zakłóceniom powstałym w wyniku jednoczesnej transmisji kilku węzłów. Jego działanie opiera się na nieustannym utrzymywaniu urządzeń w trybie odbioru, aby w momencie podjęcia decyzji o nadawaniu, po odczekaniu losowego odcinka czasu mierzonego w milisekundach, być zdolnym do określenia zajętości kanału. W przypadku braku odbioru danych binarnych, urządzenie może rozpocząć transmisję. Mechanizm unikania kolizji jest stosowany przez wszystkie typy urządzeń należących do sieci Z-Wave.

1.2.2.2 Format ramki

Ramka warstwy MAC jest przekazywana bezpośrednio do nadania warstwie fizycznej, z pominięciem mechanizmu kapsułkowania. Ramka składa się z preambuły umożliwiającej synchronizację odbiornika, bloku wskazującego początek ramki, bloków danych składających się z jednego bajtu oraz bloku wskazującego na koniec ramki (Rysunek 1.12).

Preambuła	Początek ramki	Dana 1	Dana 2	Dana 3	...	Dana n	Koniec ramki

Rysunek 1.12.: Format ramki podwarstwy kontroli dostępu do nośnika MAC
Źródło: opracowanie własne na podstawie [Zen06, str. 7]

1.2.3 Warstwa transferu

Warstwa transferu specyfikacji Z–Wave kontroluje transmisję danych pomiędzy dwoma węzłami sieci i dba o jej poprawność, poprzez wiadomości ACK, ewentualną retransmisję oraz sprawdzanie sumy kontrolnej. Na poziomie warstwy transferu rozróżniane są trzy typy transmisji: singlecast, multicast, broadcast, które wraz z wiadomością typu ACK definiują cztery formaty ramek warstwy transferu standardu Z–Wave.

1.2.3.1 Formaty ramek

Wszystkie formaty ramek warstwy transferu zbudowane są wg. wspólnego schematu (Rysunek 1.13). Zgodnie z nim, na samym początku ramki umieszczany jest identyfikator sieci Z–Wave oraz identyfikator nadawcy wiadomości (p. 1.2.4.2). W następnym bajcie zawarty jest nagłówek, zawierający informacje o formacie przekazywanej ramki. W kolejnych polach przekazywana jest informacja o długości wiadomości oraz blok danych, którego struktura uzależniona jest od formatu ramki. Na samym końcu ramki umieszczana jest suma kontrolna.

Home ID	Source ID	Nagłówek	Długość bloku danych	Blok danych	Suma kontrolna

Rysunek 1.13.: Ogólny format ramki warstwy transferu
Źródło: opracowanie własne na podstawie [Zen06, str. 9]

Ramka Singlecast

Ramka typu *Singlecast* jest zawsze transmitowana do jednego, określonego węzła, nieobligatoryjnie potwierdzającego odbiór wiadomością ACK. W przypadku tego typu ramki, w bloku danych zawarty jest 8 bitowy identyfikator odbiorcy oraz właściwy blok danych.

Ramka Multicast

Ramka typu *Multicast* zostaje zaadresowana jednocześnie do kliku węzłów należących do sieci, unikając wysyłania osobnych wiadomości do każdego z tych urządzeń. W przypadku transmisji tego typu ramki, odbiorcy nie przekazują zwrotnej wiadomości potwierdzenia odbioru, związku z czym ramka ta nie jest rekomendowana dla zastosowań wymagających niezawodnej komunikacji. W bloku danych ramki *Multicast* zawarta jest lista identyfikatorów odbiorców oraz właściwy blok danych.

Ramka Broadcast

Ramki rozgłaszania są odbierane przez wszystkie urządzenia w sieci, bez potwierdzenia odbioru wiadomością ACK. Analogicznie do ramki *Multicast*, rozgłaszanie nie jest zalecane dla komunikacji wymagającej niezawodności. Ramka rozgłaszania zaadresowana jest wartością *0xFF*.

Ramka ACK

Ramka potwierdzająca odbiór jest elementem mechanizmu zapewniającego niezawodną komunikację, dając gwarancję odbioru wiadomości. Ze względu na jej charakter, blok danych tego typu ramki jest pusty.

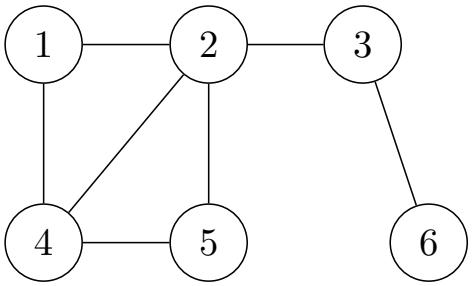
1.2.4 Warstwa routingu

Urządzenia standardu Z–Wave posiadają zakres dochodzący na obszarze otwartym do 30.5 metra, jednak w budynkach, których ściany, wyposażenie i użytkownicy pochłaniają dużą część mocy sygnału obniżając jego zasięg, niezbędne jest użycie routingu. Mechanizm

ten zaimplementowany w warstwie routingu, pozwala na przekazywanie ramek pomiędzy urządzeniami nie będącymi w swoim bezpośrednim zasięgu, wykorzystując do tego celu kratową topologię sieci. Trasę, wzdłuż której przekazywana jest wiadomość, ustala koordynator główny bazując na tablicy routingu.

1.2.4.1 Tablica routingu

Tablica routingu jest tworzona oraz uzupełniania przez koordynatora głównego w reakcji na dołączenia nowego urządzenia do sieci bądź wykluczenia. Podczas dołączania, koordynator główny odpytuje urządzenie o jego sąsiadów, aby na podstawie tej informacji móc określić graf połączeń reprezentowany właśnie za pomocą tablicy routingu, będącej w praktyce macierzą sąsiedztwa (Rysunek 1.14 (b)). W ramce poddawanej routingowi, podawana jest maksymalna ilość przeskoków oraz kolejne identyfikatory urządzeń mających przekazać wiadomość dalej.



(a) Graf połączeń pomiędzy węzłami sieci

	1	**2**	**3**	**4**	**5**	**6**
1	0	1	0	1	0	0
2	1	0	1	1	1	0
3	0	1	0	0	0	1
4	1	1	0	0	1	0
5	0	1	0	1	0	0
6	0	0	1	0	0	0

(b) Tablica routingu

Rysunek 1.14.: Graf połączeń w sieci wraz z odwzorowaniem w tablicy routingu
Źródło: opracowanie własne na podstawie [Zen06, str. 13]

1.2.4.2 Adresy urządzeń

Protokół Z–Wave w celu identyfikacji węzłów wykorzystuje adres hierarchiczny, zbudowany z identyfikatora sieci (ang. *Home ID*) oraz identyfikatora węzła (ang. *Node ID*).

Identyfikator sieci

Identyfikator sieci, przechowywany jako 32 bitowy unikalny adres, pozwala na adresowe odseparowanie sąsiednich systemów Z–Wave od siebie oraz na wymianę informacji pomiędzy tymi sieciami. Identyfikator ten jest zaprogramowany przez wytwórce w pamięci kontrolerów, które po utworzeniu sieci nadają go urządzeniom podległym. Identyfikator urządzeń podległych, zarówno sieciowy jak i węzła, fabrycznie ustawiony jest na wartość 0, która po przyłączeniu do sieci zostaje ustalona przez koordynatora.

Identyfikator węzła

Identyfikator każdego węzła ustalany jest przez głównego koordynatora podczas jego dołączania urządzenia od sieci. Jego 8 bitowa wartość musi unikalna w obrębie danej sieci.

1.2.5 Warstwa aplikacji

Warstwa aplikacji standardu Z–Wave jest odpowiedzialna za dekodowanie i wykonywanie komend pochodzących od koordynatorów sieci. Pośród komend można wyróżnić da typy: komendy protokołu Z–Wave, takie jak komenda dołączenia czy przekazania informacji o węźle oraz komendy specyficzne dla konkretnej aplikacji.

1.2.5.1 Informacje o węźle

Ze względu na różnorodność urządzeń potencjalnie tworzących sieć Z–Wave, konieczne jest stosowanie ramki opisującej funkcjonalność węzła. Część z tych funkcjonalności, zdefiniowana przez standard Z–Wave, odpowiada za podstawowe akcje węzła. Ramka z informacjami o funkcjonalnościach jest rozgłaszana przez węzeł w odpowiedzi na ramkę żądania tych informacji lub w wyniku naciśnięcia przycisku akacji danego urządzenia.

1.2.5.2 Format ramki

Ramka warstwy aplikacji rozpoczyna się z nagłówkiem wiadomości, przechowującym metodę transmisji wiadomości (*Singlecast*, *Multicast*, *Broadcast*). Następnie, przekazywana klasa komendy należąca do standardu Z–Wave (numery 0x00–0x1F) lub należąca do klasy komend konkretnej aplikacji (numer 0x20–0xFF). W następnym polach zawarty jest identyfikator komendy lub akcji oraz parametry, potrzebne do wykonania rozkazu (lista komend jest zawarta w dokumentacji Z–Wave [Zen07]). Wszystkie typy ramek poza ACK mogą służyć do przekazywania komend.

Nagłówek	Klasa komendy	Komenda	Parametr 1 komendy	. . .	Parametr n komendy

Rysunek 1.15.: Format ramki warstwy aplikacji
Źródło: opracowanie własne na podstawie [Zen06, str. 14]

1.3 Pozostałe standardy

Poza opisanymi w poprzednich sekcjach popularnymi w skali światowej standardami komunikacji bezprzewodowej dla systemów automatyki domowej, istnieją standardy mniej powszechne, zawierające jednak ważne dla niniejszej pracy koncepcje.

1.3.1 Xcomfort

Xcomfort jest bezprzewodowym zdecentralizowanym systemem automatyki domowej, wykorzystującym do komunikacji pasmo ISM o częstotliwości 869.3 MHz. Specyfikacja systemu zapewnia routing wiadomości konfigurowalny przy wykorzystaniu komputera. Ze względu na zamknięty charakter tego systemu, właściciel – firma EATON[4], nie udostępnia niestety specyfikacji do wglądu publicznego[5].

[4] Oficjalna strona firmy EATON http://www.moeller.net/de/index.jsp, 2014
[5] W celu otrzymania wglądu w specyfikację techniczną wymagane jest zawarcie umowy o poufności z firmą EATON, która to umowa uniemożliwia jednak przedstawienie informacji o standardzie w niniejszej

Pomimo jednak niedostępności dokumentacji Xcomfort, wspomnieć należy o tym systemie ze względu na możliwość szczególnego mechanizmu konfiguracji zależności zachodzących pomiędzy urządzeniami. Relacje te definiowane są przez użytkownika na zasadzie tworzenie połączeń w grafie obrazujących wszystkie urządzenia znajdujące się w systemie, które to są zapisywane w sposób rozproszony, na samych urządzeniach biorących udział w danej akcji. Rozwiązanie to pozwala na rezygnację z obligatoryjności elementu centralnego systemu, przenosząc sam proces wymiany rozkazów na urządzenia im podlegające, rozpraszającym tym samym sterowanie oraz zwiększając niezawodność systemu i odporność na ewentualne awarie.

1.3.2 Insteon

Standard formy Insteon, którego początki można datować na rok 1997, jest odpowiedzią na istniejące na rynku drogie i skomplikowane protokoły komunikacji w systemach automatyki domowej, takie jak ZigBee® czy Z–Wave [Ins13]. Standard ten definiuje dwuzakresową sieć kratową w modelu peer–to–peer, wykorzystującą do swojego działania równolegle dwa media transmisyjne:

- sieć elektroenergetyczna — standard INSTEON PL, wykorzystujący falę nośną o częstotliwości 131.65 KHz modulowaną binarnym kluczowaniem fazy,
- fale radiowe — standard INSTEAON RF, wykorzystujący falę nośną o częstotliwości pasma ISM 868 MHz (Europa), modulowaną kluczowaniem częstotliwości

Dzięki równości wszystkich węzłów sieci, możliwe było wprowadzenie mechanizmów routingu, ponownego nadawania wiadomości i kontroli błędów, realizowanych przez każde urządzenie w sieci, bez konieczności przypisywania im większych zasobów bądź uprawnień (por. 1.1.3.1).

Zastosowanie dwóch medium transmisyjnych pozwoliło w standardzie Insteon na zwiększenie niezawodności sieci oraz jej zasięgu, bez potrzeby korzystania z mostków przekazujących wiadomości w sieciach innego standardu jak WLAN czy Ethernet. Jednocześnie, urządzenia nie muszą posiadać równolegle interfejsu radiowego jak i elektroenergetycznego, pozwalając na kombinację wewnętrznych standardów dla otrzymania sieci przystosowanej do indywidualnych potrzeb.

1.3.3 KNX–RF

Konnexbus (KNX) jest otwartym standardem zarządzania oraz kontroli urządzeń dla automatyki domowej, stworzonym w wyniku połączenia w roku 2001 trzech niezależnych standardów, z którymi KNX zachował kompatybilność wsteczną[Ass06]:

- *European Installation Bus (EIB)* — rozwijany przez EIBA (ang. *European Installation Bus Association*),
- *European Home Systems (EHS)* — stowarzyszenia EHSA (ang. *European Home Systems Association*),
- *BatiBUS* — stowarzyszenie BCI (ang. *Batibus Club International*).

pracy

Konnexbus jest systemem całkowicie zdecentralizowanym, w którym każde urządzenie wyposażone jest w procesor oraz podzespoły umożliwiające autonomiczną pracę. Wybór mikroprocesora sterującego urządzeniem jest całkowicie uniezależniony od KNX Association, proponujące przykładową implementację m.in. dla procesorów z rodziny ATmega, ATiny, ARM, PIC24F czy 8051. Sterowanie rozproszone w systemie umożliwia kontynuowanie jego poprawnej pracy pomimo awarii niektórych jego elementów oraz rozdział zadań na moc obliczeniową poszczególnych urządzeń.

Standard KNX początkowo przeznaczony do stosowania w sieciach komunikujących się za pomocą skrętki dwuparowi oraz linii elektroenergetycznej, został z czasem rozbudowany o łączność w oparciu o Ethernet, podczerwień oraz fale radiowe.

Standard KNX–RF, którego medium transmisyjnym są fale radiowe, został zdefiniowany w roku 2001 oraz zaktualizowany do wersji 1.1 w 2010 r, określanej jako KNX–RF Ready. Rok później zostało wydane jego rozszerzenie, zwane KNX–RF Multi, obsługująca 3 kanały szybkie (pierwszy z nich zgodny z standardem w wersji 1.1) oaz 2 kanały wolne, przeznaczone dla urządzeń zasilanych bateryjnie lub nawet samozaspokajających się energetycznie[6] [Ass09][Wei05][Ass06, str. 40–46]

Specyfikacja tego „otwartego" standardu kosztuje 1000 Euro. Odwołać się jedynie do opracowań?

1.3.3.1 Warstwa fizyczna

Warstwa fizyczna standardu KNX–RF została zdefiniowana w odniesieniu do norm urządzeń bezprzewodowych krótkiego zasięgu. Zgodnie ze specyfikacją, nośnikiem wiadomości kodowanej kodem Manchester jest fala radiowa o częstotliwości środkowej 868.3 MHz, modulowana metodą kluczowania fazy.

Warto zaznaczyć, iż specyfikacja KNX–RF nie określa wprost urządzeń mogących wykorzystywać ten standard komunikacji, pozostawiając pełną dowolność w tej sferze wytwórcom.

Rodzaje urządzeń

W standardzie KNX–RF, ze względu na dostępny sposób komunikacji, rozróżniane są dwa typy urządzeń: komunikujące się jednokierunkowo (ang. *Unidirectional*) oraz dwukierunkowe (ang. *Bidirectional*). Urządzenia obsługujące komunikację jednokierunkową nie posiadają modułu odbierającego, posiadając jedynie zdolność do wysyłania danych. Rozwiązanie to, ograniczające pobór prądu związany z nieustannym nasłuchiwaniem wiadomości dedykowane jest urządzeniom pełniącym rolę czujników, od których często wymagana jest praca przez długi okres czasu na zasilaniu bateryjnym.

1.3.3.2 Warstwa łącza

Zadaniem warstwy łącza jest zapewnienie pozbawionej błędów transmisji telegramu pomiędzy dwoma węzłami sieci lub pomiędzy urządzeniem magistrali a węzłem. W celu realizacji wyznaczonych celów, standard definiuje ramkę (Rysunek 1.16) skonstruowaną z preambuły oraz przynajmniej dwóch bloków, z których każdy zabezpieczany jest cykliczną sumą kontrolną.

[6] Energy harvesting, zdefiniowany m.in. przez standard EnOcean, określający proces pobierania energii ze źródeł zewnętrznych, jak np. energia solarna, termiczna, wiatrowa czy kinetyczna.

Oktety: 6	1	1	1	1	6	10	1	zm.	zm.	10	1...16	10	zm.
	Długość	C-field	ESC	Pole kontrolne	Numer seryjny nadawcy		Pole kontrolne	Adres	Dane				
Preambuła	Blok 1					CRC	Blok 2			CRC	...	CRC	Postambuła

Rysunek 1.16.: Format ramki warstwy łącza.
Źródło: opracowanie własne na podstawie [Wei05, str. 2]

W bloku pierwszym zawarte są informacje o długości ramki, parametrach łącza i nadawcy oraz przekazywany jest numer seryjny lub domena nadawcy. Blok drugi, rozpoczynający się polem kontrolnym z informacjami o kapsułkowanej ramce, adresami docelowymi i źródłowymi, przekazuje właściwą ramkę danych. Następnie, po sumie kontrolnej bloku drugiego może zostać umieszczona postambuła, zakańczając tym samym całą ramkę lub może być ona kontynuowana poprzez dodanie kolejnych bloków.

1.3.3.3 Warstwa sieci

Warstwa sieci dla standardu KNX–RF jest zdecydowanie uproszczona w stosunku do przewodowej wersji tego systemu. Podczas odbioru danych warstwa sieci jedynie interpretuje tryby adresowania. Natomiast podczas nadawania, konstruowane są żądania warstwy łącza dla wszystkich rodzajów ramek przeznaczonych do nadania.

1.3.3.4 Warstwa transportowa

W chwili bieżącej, do komunikacji wykorzystywana jest jedynie metoda bezpołączeniowa. Komunikacja zorientowana na połączanie stosowana w przewodowej wersji systemu w celu zarządzania węzłami.

1.3.3.5 Warstwa sesji i prezentacji

Podobnie jak ma to miejsce w przypadku standardu KNX dla komunikacji przewodowej, warstwa sesji oraz prezentacji nie są zdefiniowane, a ich usługi zostały przeniesione do warstwy aplikacji.

1.3.3.6 Warstwa aplikacji

Warstwa aplikacji standardu KNX–RF została rozdzielona na dwie części: komunikację w czasie działania oraz zarządzanie urządzeniem. Komunikacja w czasie działania posiada jedynie możliwość zapisu wartości (i tym samym przekazanie jej innym urządzeniom). W przypadku urządzeń zdolnych do komunikacji dwukierunkowej zestaw operacji możliwych do przeprowadzenia jest znacznie bardziej rozbudowany, prowadząc w konsekwencji do większego stopnia skomplikowania implementacji danego urządzenia.

Punkty danych

Kluczową koncepcją aplikacji standardu KNX są punkty danych (ang. *Datapoints*), reprezentujące proces oraz jego zmienne, udostępniane pozostałym urządzeniom systemu

poprzez określony zbiór komend odczytu oraz zapisu. Punkty danych, gromadzone w formie grup obiektów, mogą wyrażać wejścia, wyjścia, parametry czy dane diagnostyczne urządzania, przechowywane w formie ustandaryzowanych typów danych.

Aplikacja w perspektywie punktów danych, postrzegana jest jako rozproszona pośród urządzeń sieci, definiowana poprzez kolekcję powiązanych ze sobą odbierających lub transmitujących punktów danych. Poprzez zadeklarowaną w ten sposób konstrukcję aplikacji, poszczególne urządzenia systemu mogą przetwarzać oraz reagować na dane pochodzące z innych urządzeń, traktując je jak lokalne. Ze względu na to podejście, można określić protokół KNX jako sterowany danymi (ang. *Data Driven*).

1.3.3.7 Tryby konfiguracji

Standard Konnex określa dwa dostępna tryby konfiguracji dla swoich urządzeń: tryb systemowy (S–Mode) oraz tryb prosty (E–Mode). W standardzie KNX–RF, jako uproszczonej wersji swojego przewodowego odpowiednika, pomimo implementacji obu wymienionych trybów konfiguracji, dostępny jest jedynie tryb prosty.

Tryb systemowy

Tryb systemowy (ang. *System Mode*) opisuje konfigurację urządzenia przy wykorzystaniu komputera PC z zainstalowanym specjalnym oprogramowaniem[7], oferując największe możliwości względem konfiguracji całego systemu.

Tryb prosty

Głównym celem trybu prostego jest umożliwienie łatwej konfiguracji systemu bez użycia komputera PC. Wszystkie informacje niezbędne do zestawienia połączeń są zapisane przez wytwórcę na poszczególnych urządzeniach, podejmujących próbę powiązania w reakcji na np. naciśnięcie przycisku na urządzeniach, będących w trybie nauki.

[7]Strona aplikacji ETS4 na oficjalnej stronie KNX Association http://www.knx.org/knx-en/software/ets/about/index.php, 2012

Rozdział 2

Specyfikacja protokołu komunikacyjnego

Niniejszy rozdział prezentuje rezultaty prac badawczych oraz konstrukcyjnych, wykonanych w ramach zrealizowanej pracy. W pierwszej kolejności, przedstawiona została ogólna koncepcja architektury protokołu, sieci na nim bazującej oraz mechanizmów określających działanie całego systemu. Następnie, zdefiniowana została warstwa fizyczna, warstwa aplikacji oraz podwarstwa zarządzania zdarzeniami, z uwzględnieniem motywacji kierujących wyborem danych rozwiązań. Finalnie, przedstawiona została zalecana implementacja protokołu w formie urządzenia oraz implementacja eksperymentalnie zrealizowana w ramach pracy, uwzględniając przy tym wymogi stworzonego standardu oraz dostępność poszczególnych modułów na runku.

2.1 Ogólna charakterystyka protokołu

Zdefiniowany w ramach pracy protokół pozwala na tworzenie kratowych sieci bezprzewodowych, składających się z tanich energooszczędnych urządzeń, podejmujących określone akcje w odpowiedzi na zdarzenia zachodzące w całej sieci. Najważniejszymi założeniami, które bezpośrednio wpływały na kształt protokołu, są: łatwość instalacji, energooszczędność, niski koszt wdrożenia oraz uniwersalność. Stworzony standard komunikacji bezprzewodowej oferuje m.in.

- tworzenie, zarządzanie oraz komunikację peer–to–peer w sieci o topologii kraty
- gwarancję poprawności transmisji
- niski pobór mocy
- interakcje pomiędzy urządzeniami w oparciu model *event–driven*
- zdalne wiązanie zdarzeń z akcjami urządzeń, umożliwiające ich nienadzorowaną pracę
- licencyjną wolność i otwartość

Standard definiuje trzy elementy stosu sieciowego — warstwę fizyczną, warstwę aplikacji oraz podwarstwę zarządzania zdarzeniami. Wymienione moduły są wystarczające dla zapewnienia poprawnej komunikacji i realizacji zadań stawianych systemowi opartemu na tym protokole.

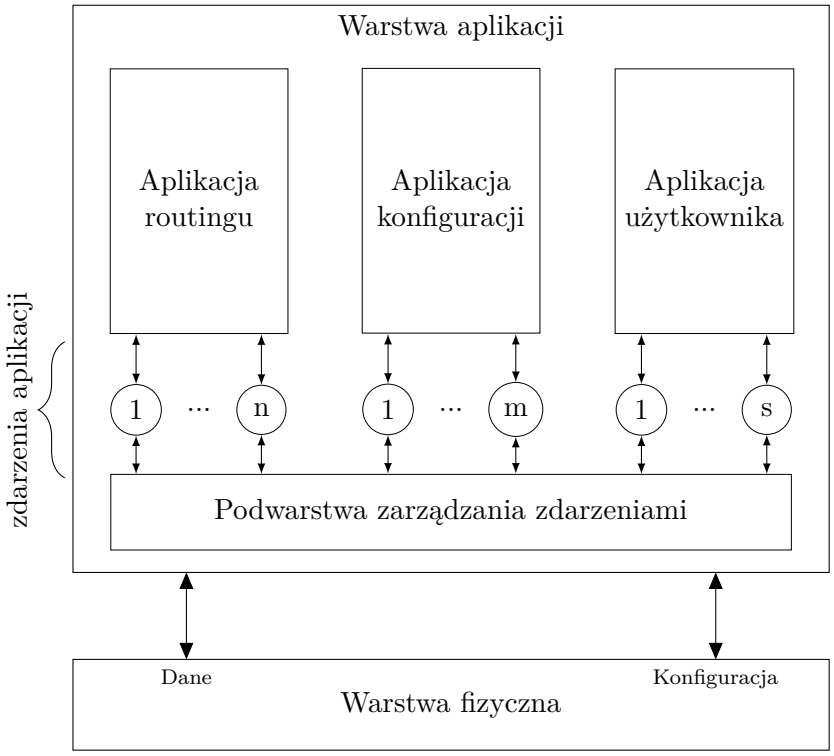
Ze względu na otwartość oraz wolność stworzonego standardu komunikacji, przyjęta została możliwość implementacji za pomocą modułów radiowych aktualnie dostępnych na rynku, rezygnując tym samym z wytwarzania układów z zaimplementowanym już stosem protokołu. Związku z tym, elementy stosu rozpatrywane są w podejściu mniej abstrakcyjnym niż ma to miejsce np. w przypadku standardu IEEE 802.15.4, zakładając dystrybucję warstw stosu na osobne układy (moduł komunikacji bezprzewodowej oraz mikroproce-

sor) [oEEE11]. Podejście takie pozwala przyszłym twórcom urządzeń na wybór dowolnej platformy sprzętowej, wspierając tym samym rozpowszechnienie się standardu.

2.1.1 Architektura protokołu

Stworzony protokół, zgodnie z referencyjnym modelem OSI[1], został zbudowany w oparciu o bloki funkcyjne zwane warstwami. Każda z warstw odpowiedzialna jest za realizację zbioru określonych zadań, oferując usługi warstwom wyższym.

Opisywany protokół komunikacji bezprzewodowej ogranicza się do definicji dwóch warstw modelu — warstwy fizycznej oraz warstwy aplikacji, zawierającej podwarstwę zarządzania zdarzeniami (rysunek 2.1). Funkcjonalności pozostałych warstw modelu referencyjnego (łącza danych, sieciowa, transportowa, sesji oraz prezentacji) zostały rozdystrybuowane w obrębie obu zdefiniowanych warstw stosu protokołu. Podejście takie miało ma celu maksymalne uproszczenie implementacji, rzutujące na możliwość wykorzystania tańszych, mniejszych oraz bardziej energooszczędnych mikroprocesorów.



Rysunek 2.1.: Ogólna architektura stosu
Źródło: opracowanie własne

Zadaniami realizowanymi przez warstwę fizyczną jest pomiar jakości połączenia, ocena zajętości kanału, kontrola dostępu do nośnika, zagwarantowanie oraz sprawdzanie poprawności transmisji, wstępne filtrowanie wiadomości oraz wymiana danych za pośrednictwem fizycznego medium. Funkcje te realizowane są za pośrednictwem udostępnianego przez warstwę fizyczną interfejsu.

[1]International Standard ISO/IEC 7498-1: *Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*, 1996

Podwarstwa zarządzania zdarzeniami odpowiedzialna jest za wiązanie zdarzeń z reakcjami na nie, rejestrowanie oraz wywoływanie zdarzeń. W tym kontekście, zdarzenie rozumiane jest jako asynchroniczne działanie wywołane poprzez wyzwalacz zewnętrzny (przekazany siecią identyfikator zdarzenia) lub wewnętrzny (zmiany stanu danego urządzenia). Po zakończeniu obsługi wywołanego zdarzenia, wywoływane są kolejne zdarzenia do niego dowiązane, tworząc łańcuch zdarzeń w hierarchii typu akcja–reakcja.

Warstwa aplikacji zawiera bloki funkcyjne dotąd należące do osobnych warstw modelu referencyjnego, zaimplementowanych w formie odrębnych aplikacji, jak aplikacja odpowiedzialna za przyłączanie do sieci, routing wiadomości czy konfigurację urządzenia. Zastosowanie takiego rozwiązania umożliwiło rezygnację z kapsułkowania, skutkującego najczęściej narzutem danych oraz czasu przetwarzania. Zamiast przekazywania wiadomości poprzez poszczególne warstwy stosu aż do momentu osiągnięcia tej właściwej, w stworzonym standardzie każda aplikacja udostępnia zestaw zdarzeń wywoływanych bezpośrednio na podstawie identyfikatora zawartego w wiadomości. W praktyce oznacza to przykładowo nadanie wiadomości z identyfikatorem zdarzenia *Przekaż wiadomość*, w przypadku potrzeby wysłania pakietu do innego urządzenia lub nadanie identyfikatora zdarzenia *Wyłącz*, dla wyłączenia wybranego urządzenia.

2.1.2 Komponenty sieci

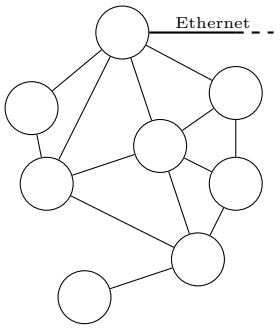
Sieć bezprzewodowa zrealizowana w oparciu o stworzony protokół komunikacyjny, zbudowana jest urządzeń zapewniających, z punktu widzenia sieci, identyczną funkcjonalność i możliwości. W przeciwieństwie więc do standardu ZigBee® lub Z–Wave, nie istnieje żaden logiczny podział na urządzenia umożliwiające routing, mobilne, koordynujące czy wydające rozkazy. Każde urządzenie musi realizować wszystkie funkcjonalności protokołu, posiadając przy tym identyczny uprawnienia, jak inicjacja komunikacji czy zmiana konfiguracji innego urządzenia.

Pojawiające się w dalszej części pracy, urządzenie nadzorujące, w praktyce stanowi definicję bramki dostępowej, która za pośrednictwem np. interfejsu webowego umożliwia użytkownikowi zdalną konfigurację sieci, poszczególnych urządzeń oraz relacji pomiędzy nimi zachodzących. Jednak obecność nadzorcy nie jest absolutnie wymagana dla działania systemu, udostępniającego mechanizm wcześniejszego powiązania urządzeń oraz ustalenia ich relacji.

2.1.3 Topologia sieci

Stworzony standard oferuje topologię sieci w postaci kraty, bez wyszczególnionych elementów centralnych lub koordynujących, w której każde urządzenie może komunikować się każdym, nawet w przypadku braku bezpośredniego połączenia (rysunek 2.2). Wszystkie węzły posiadają przydzielony unikalny w obrębie danej sieci jednobajtowy identyfikator, który wraz z identyfikatorem danej sieci pozwala jednoznacznie zidentyfikować każde urządzenia.

Obecność w sieci urządzeń stanowiących bramki do innych interfejsów, np. sieci Ethernet, Z–Wave lub naściennego panelu dotykowego, nie są w żaden sposób specjalnie traktowane w ujęciu topologii sieci oraz jej zachowania – wszelkie dodatkowe interakcje z siecią



Rysunek 2.2.: Topologia sieci zbudowanej w oparciu o protokół
Źródło: opracowanie własne

zależą od wewnętrznej implementacji bramek i stawianych im celów.

2.1.4 Energooszczędność

Niski pobór energii elektrycznej był jednym z ważniejszych aspektów branych pod uwagę podczas projektowania opisywanego standardu komunikacji bezprzewodowej. Założenie to jest ważne nie tylko w sytuacji urządzenia działającego wyłącznie na zasilaniu bateryjnym, ale również w przypadku urządzeń posiadających stałe źródło zasilania – jednym z zadań stawianych systemom inteligentnego budynku jest obniżenie kosztów eksploatacji. Nadmierny pobór prądu przez urządzenia należące do systemu mógłby nie tylko utrudnić realizację tego celu, ale nawet uniemożliwić.

Energooszczędność na poziomie protokół została zapewniona przez umożliwienie jego prostej i mało skomplikowanej implementacji, ograniczenie zbędnego ruchu sieciowego, zmniejszenie stosowanej długości wiadomości oraz ilości operacji niezbędnych do jej przetworzenia. Dalsza poprawa wydajności energetycznej zależy od jakości implementacji programowej oraz sprzętowej.

2.1.5 Licencjonowanie

Rozwiązania wolne lub otwarte stanowią marginalną mniejszość ogólnej całości systemów automatyki domowej dostępnej na rynku. Przykładowi reprezentanci tej niewielkiej grupy, system zarządzania openHAB współpracujący ze standardem KNX, pojmującym niestety otwartość dość opacznie, nie są w stanie zredukować rynkowego deficytu otwartości.

Wolne i otwarte licencjonowanie stworzonego protokołu nie jest jedynie wybiegiem ideologicznym, posiadającym wyłącznie aspekt moralny bądź światopoglądowy. Otwartość determinuje sposób wytwarzania rozwiązań czy standardów, realizowanych przy ścisłej współpracy przedsiębiorstw, specjalistów oraz użytkowników umożliwiającej powstanie produktów, na które nakład przewyższałby możliwości pojedynczych osób czy nawet korporacji. Możliwe jest to dzięki zaangażowaniu społeczności w rozwój danego projektu, często w formie wolontariatu, co pozwala tym samym na radykalne obniżenie kosztów przedsięwzięcia. Wgląd dowolnie wielkiej grupy osób w sposób realizacji i implementacji pozwala reagować szybko na błędy i wprowadzać odpowiednie zmiany zgodne z oczekiwaniami użytkownika. Analiza realizowana przez większą liczbę niezależnych osób pozwala

wyeliminować potencjalne błędy i luki bezpieczeństwa, w konsekwencji umożliwiając tworzenie projektów o wysokim stopniu niezawodności, bezpieczeństwa czy staranności wykonania. Jest to niezwykle istotne w kontekście znacznego spadku zaufania użytkowników względem oprogramowania w związku z ostatnio ujawnionymi nielegalnymi praktykami służb wywiadowczych[2]. Ponadto, otwartość sprzyja szybszemu rozwojowi przedsięwzięcia, szczególnie w przypadku działania w oparciu o projekt tworzony w internecie, na oczach wszystkich zainteresowanych i przy współudziale chętnych.

Systemy automatyki domowej wymagają najczęściej pełnej integracji z budynkiem mieszalnym, w którym spędza się większość czasu. System zamknięty, nieobdarzony pełnym zaufaniem nie powinien nigdy mieć szansy znaleźć się w sferze tak prywatnej jak własny dom, który od zawsze stanowił swoistą ostoję i autonomiczną strefę życia. Podobnie jednak, jak ma to miejsce w innych dziedzinach informatyki, do prywatności i zaufania nie jest przykładana należyta uwaga i większość ludzi zdaje się na zamknięte, niepodlegające społecznej kontroli rozwiązania. Tworzony w ramach tej pracy system, poza oczywistym wymiarem funkcjonalnym, ma uświadamiać tę potrzebę i dawać gotowe rozwiązania, które ją zaspokoją.

2.2 Warstwa fizyczna

Warstwa fizyczna stworzonego protokołu komunikacyjnego wykorzystuje falę nośną o częstotliwości z pasma ISM (ang. *Industrial, Scientific, Medical*, tablica 2.1), poddaną modulacji cyfrowej MSK (ang. *Minimum Shift Keying*). Transmisja danych realizowana jest z prędkością 250 kbps.

Tablica 2.1.: Częstotliwości ISM stosowane przez protokół

Obszar	**Zakres częstotliwości**
Europa	868–868.6 MHz
Ameryka, Australia	902–928 MHz
Chiny	779–787 MHz
Japonia	915–930 MHz

Wybór częstotliwości fali nośnej oraz sposobu jej modulacji podyktowany był przeznaczeniem systemu do domów mieszkalnych, w których propagacja sygnału radiowego jest utrudniona przez liczne przeszkody fizyczne (elementy konstrukcyjne, wyposażenie, mieszkańcy) oraz zakłócenia generowane przez sprzęty domowe (kuchenka mikrofalowa, odkurzacz czy urządzenia bezprzewodowe). Posługując się modelem propagacji fal radiowych w pomieszczeniach, zdefiniowanej w rekomendacji ITU 1238, wybrana została częstotliwość 868 MHz, charakteryzująca się niższym poziomem tłumienia (98 dB) oraz mniejszym wykorzystaniem pasma, a co za tym idzie generowanym szumem, niż np. 2400 MHz (tłumienie 103 dB) obsługujące Wi-Fi, Bluetooth, ZigBee czy bezprzewodowe peryferia komputera[ITU97]. Co więcej, wyższe częstotliwości posiadają gorszą zdolność przenikania przez przeszkody przy jednoczesnej wyższej podatności na odbicia, mogące rzutować na awaryjność komunikacji.

[2]Informacje o National Security Agency wraz ze wzmiankami o aferze podsłuchowej na stronie angielskiej Wikipedii http://en.wikipedia.org/wiki/National_Security_Agency#Criticism, 2014

Wybór techniki modulacji podlegał niejako dostępności modułów komunikacji bezprzewodowej na rynku, które oferują ograniczony zakres obsługiwanych modulacji, najczęściej: kluczowanie częstotliwości (ang. *Frequency–Shift Keying*) z odmianą ciągłą (ang. *Continuous Phase FSK*, również *Minimum Shift Keying*) i gaussowską (ang. *Gaussian Frequency-Shift Keying*) oraz kluczowanie amplitudy (ang. *Amplitude-Shift Keying*). Wybrana modulacja MSK zapewnia dobre właściwości energetyczne przekazując 99% energii w paśmie kanału i redukując tym samym interferencje pomiędzy kanałami[3].

Zadaniami realizowanymi przez warstwę fizyczną jest pomiar jakości połączenia, ocena zajętości kanału, kontrola dostępu do nośnika, zagwarantowanie oraz sprawdzanie poprawności transmisji, wstępne filtrowanie wiadomości oraz wymiana danych za pośrednictwem fizycznego medium. Zadania dodatkowe, których obsługa na niskim poziomie przyspiesza działanie systemu i poprawia jego wydajność energetyczną, jest szyfrowanie kluczem symetrycznym oraz zarządzanie energią modułu nadajnika/odbiornika poprzez czasowe wyłączenie niewykorzystywanych bloków.

2.2.1 Moc wyjściowa nadajnika

Maksymalna moc wyjściowa nadajnika regulowana jest prawnie ustawami o zasięgu globalnym oraz państwowym. Przykładowo, w Rzeczpospolitej Polskiej, na podstawie rozporządzenia 972, maksymalna moc promieniowania dla pasma 868–868.6 MHz wynosi $25mW$, czyli $11.8dB$[ROZ07].

Ze względu na różne dopuszczalne maksymalne wartości mocy promieniowania na różnych obszarach, wymagana jest możliwość regulacji siły sygnału wychodzącego dla dostosowania dla tych norm. Nie jest wymagane, aby regulacja tej wartości była realizowana płynnie, w praktyce większość modułów komunikacji radiowej oferuje ograniczony dyskretny zakres mocy.

Dostosowywanie mocy wyjściowej nadajnika jest również wykorzystywane podczas realizacji mechanizmów balansowania ruchu sieciowego i zmniejszaniu zapotrzebowania energetycznego urządzeń, podejmowanego przez nadzorcę systemu, co zostało przybliżone w sekcji 3.1.1.

2.2.2 Kontrola dostępu do nośnika

W celu uniknięcia jednoczesnego dostępu do łącza, skutkującego w konsekwencji błędami komunikacji, protokół wykorzystuje mechanizm CSMA/CA (ang. *Carrier Sense Multiple Access with Collision Avoidance*) zgodny z algorytmem w wersji bezszczelinowej standardu IEEE 802.15.4 [oEEE11, str. 21–23].

Przeniesienie mechanizmu CSMA/CA z podwarstwy MAC (ang. *Medium Access Control*) do warstwy fizycznej jest zabiegiem mającym na celu odciążenie głównego mikroprocesora urządzeń, a w konsekwencji podniesienie wydajności energetycznej systemu. W sytuacji braku implementacji mechanizmu CSMA/CA w formie modułu nadajnika radiowego, należy przenieść ten mechanizm do warstwy wyższej, obsługiwanej przez mikroprocesor urządzenia, skutkując niestety skomplikowaniem funkcji odpowiedzialnych za

[3] Singal T.L., *Wireless Communications*, Tata Mcgraw Hill, 2012, rozdział 7.6.7; Chitode J. S., *Communication Systems I*, Technical Publications Pune, 2004, s. 644

komunikację i zbędnym obciążeniem procesora. Ponadto, podejście takie wymaga od układu komunikacji bezprzewodowej udostępnienia mechanizmu detekcji zajętości kanału lub pomiaru energii w kanale, co często jest zrealizowane w sposób niewystarczający dla celów algorytmu. Całkowita rezygnacja z implementacji kontroli dostępu do nośnika nie wpływa w żaden sposób na możliwość wymiany danych pomiędzy urządzeniami wykorzystującymi protokół, oddziałuje jednak znacząco na obniżenie niezawodności komunikacji i pracy całego systemu.

2.2.3 Adresy

Każde urządzenie zgodne ze standardem posiada trzy jednobajtowe, niezależne adresy:

- adres lokalnego urządzenia, ustalany na etapie przyłączania do sieci, unikalny w obrębie danej sieci
- adres rozgłoszeniowy zdefiniowany globalnie dla wszystkich urządzeń standardu, o wartości $0xFF$
- adres sieci do której przynależy dane urządzenie, unikalny w obrębie sąsiadujących sieci

Ograniczenie długości adresów urządzeń do jednego bajtu nakłada limit na rozmiar sieci do 256 urządzeń, co w aktualnych warunkach jest wystarczające dla implementacji w mieszkaniach jednorodzinnych. W przyszłości długość ta może zostać rozszerzona z ograniczonym zachowaniem kompatybilności wstecznej.

2.2.4 Potwierdzenie odbioru oraz retransmisja

Wiadomość ACK (ang. *Acknowledge*), potwierdzająca poprawny odbiór pakietu, powinna być nadawana automatycznie na adres pojedynczego nadawcy odebranej wiadomości. Wyjątek stanowią wiadomości rozgłaszane oraz niewymagające potwierdzenia odbioru poprzez ustawienie bitu NO_ACK w ramce wiadomości, zdefiniowanej w podrozdziale 2.2.8. Ramka odpowiedzi ACK powinna posiadać sekcję danych o zerowej długości oraz ustawiony bit NO_ACK.

Retransmisja wiadomości powinna być realizowana automatycznie po upływie czasu minimalnego $T_{ret\_min} = L_{max}/R + T_{RXtoTX} + T_{TXtoRX}$, gdzie L_{max} jest maksymalną długością ramki liczoną w kilobitach, R przepływnością podaną w kbps, a T_{RXtoTX} oraz T_{TXtoRX} to odpowiednio czas przejścia modułu komunikacyjnego z trybu odbioru do transmisji i vice versa. Czas T_{ret_min} wynosi w przybliżeniu $10ms$, aczkolwiek jest on w dużej mierze zależny od użytego modułu komunikacji radiowej, w związku z czym jego dokładną wartość dobrać należy doświadczalnie.

Ilość retransmisji w przypadku kolejnych niepowodzeń nadawania (braku zwrotnej wiadomości ACK) nie powinna być większa niż $100ms/T_{ret\_min}$, pozostawiając tym samym margines czasu potrzebny dla warstwy wyższej na podjęcie stosownych działań dla pomyślnego ukończenia komunikacji, takich jak zmiana trasy routingu, w czasie akceptowalnym przez użytkownika.

Mechanizm potwierdzenia poprawnego odbioru poprzez odpowiedź ACK oraz mechanizm retransmisji w przypadku braku otrzymania takiej zwrotnej odpowiedzi są obliga-

toryjnie wymagane dla zachowania sprawności działania całego systemu, z zaleceniem implementacji w warstwie fizycznej dla odciążenia głównego mikroprocesora.

2.2.5 Suma kontrolna wiadomości

Suma kontrolna wiadomości w postaci cyklicznego kodu nadmiarowego CRC (ang. *Cyclic Redundancy Check*) jest obligatoryjna dla każdej wiadomości przekazywanej za pośrednictwem protokołu. 16–bitowa suma kontrola jest obliczana ze wszystkich elementów wiadomości, wyłączając preambułę, z wykorzystaniem wielomianu

$$X^{16} + X^{12} + X^5 + 1$$

zgodnego z najbardziej rozpowszechnionym CRC-16-CCITT, dostępnym na wszystkich modułach komunikacji bezprzewodowej, które wspierają cykliczny kod nadmiarowy CRC.

2.2.6 Filtrowanie wiadomości

Odebrane pakiety powinny być filtrowane już na etapie ich przetwarzania przez warstwę fizyczną, powiadamiając warstwy wyższe jedynie w sytuacji poprawnej weryfikacji wiadomości. Filtrowanie odbywać się powinno na podstawie parametrów:

adres docelowy – adres docelowy zawarty w ramce wiadomości porównywany jest z adresem danego urządzenia oraz adresem rozgłoszeniowym. W przypadku braku zgodności wiadomość zostaje odrzucona

adres sieci – adres sieci zawarty w ramce wiadomości porównywany jest z adresem sieci do której przynależy dane urządzenie, w przypadku braku zgodności wiadomość zostaje odrzucona

suma kontrolna – obliczona suma kontrolna odebranej wiadomości (nie uwzględniająca preambuły oraz pola z sumą kontrolną) porównywana jest z sumą przekazaną w wiadomości. W przypadku niezgodności, wiadomość zostaje odrzucona

Dzięki zastosowaniu filtrowania na poziomie warstwy fizycznej i ograniczeniu udziału warstw wyższych w przetwarzaniu niepoprawnych wiadomości, możliwe jest wydajniejsze zarządzanie pracą głównego mikroprocesora i efektywniejsze regulowanie konsumpcji energii.

2.2.7 Wskaźnik jakości połączenia

Pomiar LQI (ang. *Link Quality Indicator*) jest wartością określającą siłę odebranego sygnału i/lub poziom zakłóceń w odebranym sygnale, liczoną w decybelach $[dB]$. Parametr pozwala na poprawę wydajności routingu wiadomości poprzez wskazanie tras charakteryzujących się dobrą siłą sygnału i niskim poziomem zakłóceń, zapewniając dobrą jakość transmisji. Funkcjonalność ta nie jest obligatoryjnie wymagana, aczkolwiek zalecana.

2.2.8 Format ramki

Format ramki warstwy fizycznej jest często już zdefiniowany w module komunikacji bezprzewodowej, w formie meta–protokołu komunikacyjnego danego wytwórcy. Tworząc więc własny protokół komunikacyjny należało uwzględnić możliwości konfiguracyjne ramek danych modułów bezprzewodowych, aby umożliwić ich adaptację. Efektem konsensusu jest format ramki zaprezentowany na rysunku 2.3, której implementację przy wykorzystaniu dostępnego na rynku układu komunikacji radiowej przedstawiono w rozdziale 2.4.

Oktety: **8**	**2**	**1**	**1**	**1**	**1**	**1 bit**	**0-1024**	**2**
Preambuła	Długość wiadomości	Adres docelowy	Adres źródłowy	Adres sieciowy	Numer sekwen.	NO_ACK	Blok danych	CRC

Rysunek 2.3.: Format ramki warstwy fizycznej protokołu
Źródło: opracowanie własne

Preambuła
: pole pozwalające na synchronizację odbiornika z nadajnikiem, o długości 8 bajtów w postaci 10101010

Długość wiadomości
: 4 bajtowe pole zawierające sumaryczną długość bloku danych oraz pół z adresami, liczoną w bajtach. Pole pozwala na zapis długości o maksymalnej wartości 65 536 bajtów, jednak zalecane jest wykorzystanie jedynie 1024 bajtów, które pozwalają na przesłanie wiadomości w czasie akceptowalny przez użytkownika (ok. $270ms$ dla przepływności 250 kbps)

Adres docelowy
: 1 bajtowe pole zawierające adres docelowy wiadomości, mogący stanowić adres danego urządzenia lub adres rozgłoszeniowy $0xFF$

Adres źródłowy
: 1 bajtowe pole zawierające adres nadawcy wiadomości

Adres sieciowy
: 1 bajtowe pole zawierające adres sieci do której przynależy zarówno nadawca jak i odbiorca

Numer sekwencyjny
: 2 bitowe pole zawierające numer sekwencyjny wiadomości, liczony niezależnie dla każdego adresu docelowego, inkrementowany automatycznie dla każdej nowej wiadomości (retransmisje oraz wiadomości ACK pozostawiają licznik bez zmian)

NO_ACK
: 1 bitowe pole zawierające flagę sterującą nadaniem odpowiedzi ACK w przypadku poprawnego odebrania pakietu. Flaga jest automatycznie ustawiana w przypadku wiadomości typu ACK oraz rozgłaszanych.

Blok danych
: dane przekazywane za pośrednictwem wiadomości, o długości od 0 bajtów (wiado-

mość ACK) do 1024 bajtów

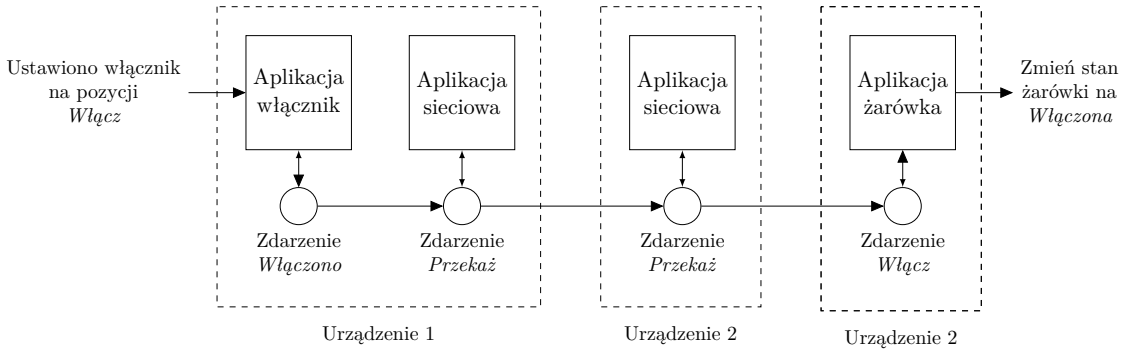
CRC

suma kontrolna wiadomości w postaci cyklicznego kodu nadmiarowego

2.3 Warstwa aplikacji

Warstwa aplikacji zdefiniowanego protokołu stanowi quasi–framework obsługujący aplikacje protokołu, użytkownika oraz udostępniane przez nie zdarzenia. Każda aplikacja stanowi swoisty blok funkcyjny realizujący określony zestaw celów, wymaganych przez protokół lub użytkownika, których wykonanie jest inicjowane poprzez zdarzenia. Zdarzenia rozumiane są jako asynchroniczne działania wywołane poprzez wyzwalacz zewnętrzny (przekazany siecią identyfikator zdarzenia) lub wewnętrzny (zmiany stanu danego urządzenia w wyniku pracy), zarządzane za pośrednictwem Podwarstwy zarządzania zdarzeniami. Po zakończeniu obsługi wyzwolonego zdarzenia, wywoływane są kolejne do niego dowiązane w formie wywołań zwrotnych, tworząc łańcuch zdarzeń w hierarchii typu akcja–reakcja. Zdarzenia dowiązane, czyli wywołania zwrotne, nie muszą być ograniczone do zdarzeń udostępnianych przez lokalne urządzenie, najczęściej są to zdarzenia innych urządzeń, wyzwalane za pośrednictwem sieci/

Przykładowy łańcuch wywołań, przedstawiony na rysunku 2.4, powoduje zapalanie się żarówki podłączonej do Urządzenia 3, w wyniku naciśnięcia włącznika światła obsługiwanego przez Urządzenie 1. W celu wywołania zdarzenia *Włącz* Urządzenia 3, jego identyfikator musi być przekazany przez sieć. Związku z tym, do zdarzenia *Włączono* Urządzenia 1 dowiązane jest zdarzenie *Przekaż*, należące do Aplikacji sieciowej, wraz z parametrem w postaci adresu Urządzenia 3 oraz identyfikatora zdarzenia *Włącz*.



Rysunek 2.4.: Przykładowy łańcuch wywołań w sieci zbudowanej z trzech urządzeń
Źródło: opracowanie własne

Przyjęte rozwiązanie dzielące działania sieciowe na aplikacje zamiast na warstwy, udostępnia jednorodny interfejs dostępu do każdej funkcji urządzenia, bez znaczenia jakie zadania realizuje. Co więcej, błąd mogący pojawić się w działaniu danej aplikacji nie powinien rzutować na działanie pozostałych, umożliwiając dalsze funkcjonowanie urządzenia.

Ze względu na przeniesienie funkcjonalności niektórych niższych warstw stosu sieciowego do warstwy aplikacji, niezbędne jest określenie zestawu aplikacji wymaganych dla działania protokołu, wraz z funkcjami przez nie realizowanymi. Sam sposób wykonania

jest całkowicie zależny od implementacji, jednak dostęp do oferowanych funkcjonalności realizowany za pośrednictwem zdarzeń, musi być być zunifikowany i zgodny z pozostałymi implementacjami protokołu.

Warstwa aplikacji odpowiedzialna jest za obsługę aplikacji użytkownika oraz protokołu, zajmujących się m.in. routingiem, zarządzaniem przynależnością urządzenia do sieci i konfiguracją urządzenia. Składnikiem tej warstwy jest ponadto Podwarstwa zarządzania zdarzeniami, udostępniająca interfejsy obsługi czy wyzwalania zdarzeń lokalnie oraz zdalnie.

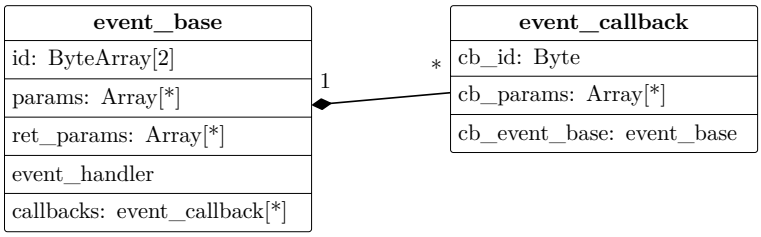
2.3.1 Podwarstwa zarządzania zdarzeniami

Podwarstwa zarządzania zdarzeniami odpowiedzialna jest za definiowanie zdarzeń, rejestrowanie nowych zdarzeń, ich wywoływanie oraz zarządzanie dowiązanymi wywołaniami zwrotnymi na zasadzie operacji CRUD (ang. *Create, Read, Update and Delete*). Podwarstwa wykorzystuje ustawienia przechowywane w utrzymywanej przez nią tablicy zdarzeń, zapewniając jej serializację oraz deserializację. Umożliwia to tym samym zachowanie czy przywrócenia rekordów tablicy w sytuacji problemów z zasilaniem. Musi ona ponadto udostępniać interfejs umożliwiający dostęp i zarządzanie oferowanymi przez podwarstwę usługami.

2.3.1.1 Zdarzenie

Zdarzenie jest akcją wywoływaną asynchronicznie w reakcji na pojawienie się przypisanego do zdarzenia wyzwalacza. W wyniku wywołania, wykonywane jest działanie obsługi danego zdarzenie, po którego zakończeniu uruchamiane są wyzwalacze zdarzeń dowiązanych, przypisanych do obsługiwanego zdarzenia w formie wywołań zwrotnych.

Każde zdarzenie jest opisywane kompletem danych, przedstawionym na rysunku 2.5, których forma reprezentacji pomiędzy poszczególny implementacjami może się różnić, pozostawiając jednak samą organizację bez zmian.



Rysunek 2.5.: Diagram UML reprezentujący zestaw danych i ich relację, opisujących zdarzenie
Źródło: opracowanie własne

Poszczególne pola struktury *event_base*, przechowującej zestaw danych opisujących zdarzenie, posiadają następujące przeznaczenia:

id 16 bitowy identyfikator zdarzenia, na którego podstawie zdarzenie zostaje wyzwolone. Identyfikatory z zakresu $0x0000 - 0x0064$ są zarezerwowane dla zdarzeń należących do aplikacji standardu i muszą pozostać unikalne. W stosunku do pozostałych

adresów warunek unikalności nie musi być spełniony, ze względu na wymóg świadomego wiązania zdarzeń pomiędzy urządzeniami,

params
: parametry wywołania; zestaw atrybutów, o możliwym dostępie na podstawie indeksu, przekazywanych do zdarzenia w momencie wyzwolenia i wpływających na sposób realizacji obsługi zdarzenia. Kolejność atrybutów jest ściśle określana przez deklarację zdarzenia, umożliwiając ich wybiórczą modyfikacją przed wyzwoleniem zdarzenia

ret_params
: parametry zwrotne; zestaw atrybutów, o możliwym dostępie indeksowym, które są ustawiane w trakcie realizacji obsługi zdarzenia, służąc do zwracania danych wynikowych. Są one przekazywane w sposób wybiórczy (na podstawie zapisanego wiązania) w formie parametrów wywołania do zdarzenia wyzwalanego jako wywołanie zwrotne. Indeksy atrybutów *ret_params*, które mają zostać przekazane w formie atrybutów *param* wyzwalanego zdarzenia, zapisane są w polach *event_callback.cb_params*,

event_handler
: zestaw instrukcji odpowiedzialnych za obsługę zdarzenia, uruchamianych przez Podwarstwę zarządzania zdarzeniami w momencie jego wyzwolenia

callbacks
: zestaw struktur typu *event_callback*, określających wywołania zwrotne, realizowane po zakończeniu obsługi danego zdarzenia

Struktura *event_callback* przechowuje dane niezbędne do realizacji wywołania zwrotnego przypisanego do danego zdarzenia i w efekcie wyzwolenia dowiązanego zdarzenia:

cb_id
: 8 bitowy identyfikator wywołania zwrotnego, umożliwiający późniejsze nim zarządzanie za pomocą operacji CRUD. Identyfikator musi być unikalny w obrębie zestawu wywołań zwrotnych przypisanych do danego zdarzenia,

cb_params
: parametry wywołania zwrotnego; zestaw atrybutów, o możliwym dostępie na podstawie indeksu, przekazywany do zdarzenia wyzwalanego dla danego wywołania zwrotnego. Poszczególne atrybuty mogą zawierać stałe dane w postaci np. łańcuchów znaków lub indeksy atrybutów pobieranych ze struktury *event_base.ret_params* zdarzenia do którego dane wywołanie zwrotne jest przypisane. Metoda rozróżnienia typu atrybutu zależna jest od implementacji,

cb_event_base
: wyzwalane w wyniku wywołania zwrotnego zdarzenie, z parametrami przekazywanymi ze struktury *event_callback.cb_params*

Wiązanie zdarzeń

Wiązanie zdarzeń jest procesem, w którym do wybranego zdarzenia zostaje przypisane inne zdarzenie w formie wywołania zwrotnego. Dzięki dowiązaniu możliwe jest kaskadowe wyzwalanie zdarzeń i kształtowanie relacji pomiędzy urządzeniami.Pozwala to na dostoso-

wywanie działania sieci zbudowanej w oparciu o protokół do realizacji wybranych celów, w sposób naturalny dla użytkownika.

Zdarzenia mogą być wiązane zarówno na lokalnym urządzeniu, poprzez interfejs operacji CRUD udostępnianych przez Podwarstwę zarządzania zdarzeniami, jak i zdalnie poprzez zdarzenia Aplikacji konfiguracji. Dla poprawnego związania dwóch zdarzeń wymagana jest znajomość identyfikatorów wiązanych zdarzeń, formatu parametrów zwracanych (*ret_params*) przez zdarzenie początkowe oraz formatu parametrów przekazywanych do zdarzenia wyzwalanego w formie wywołania zwrotnego (*params*). Ze względu na wymóg świadomego wiązania zdarzeń, jeśli nie jest możliwe automatyczne zrealizowanie wiązania, informacje o formatach powinny być w sposób czytelny i zrozumiały przedstawiane użytkownikowi, umożliwiając celowy i świadomy wybór określonych atrybutów dla odpowiedniej konfiguracji systemu. Przykładowo, dla sytuacji przedstawionej na rysunku 2.4, dowiązanie zdarzenia *Włącz* (Urządzenie 3) do zdarzenia *Włączono* (Urządzenie 3) należy do gestii użytkownika. Ponieważ jednak zdarzenia są realizowane na dwóch odseparowanych urządzeniach, niezbędna jest komunikacja sieciowa realizowana przez wyzwolenie zdarzenia *Przekaż* (Urządzenie 1), faktycznie dowiązanego do zdarzenia *Włączono*. Ponieważ użytkownik nie jest zainteresowany sposobem realizacji celu, ale jego samym osiągnięciem, dowiązanie zdarzenia *Przekaż* realizowane jest automatycznie przez urządzenie zlecające stworzenie wiązania.

Po dowiązaniu zdarzenia przekazywania wiadomości, powinna zostać rozpoczęta procedura odnajdywania trasy do docelowego urządzenia, opisana w 2.3.2.1.

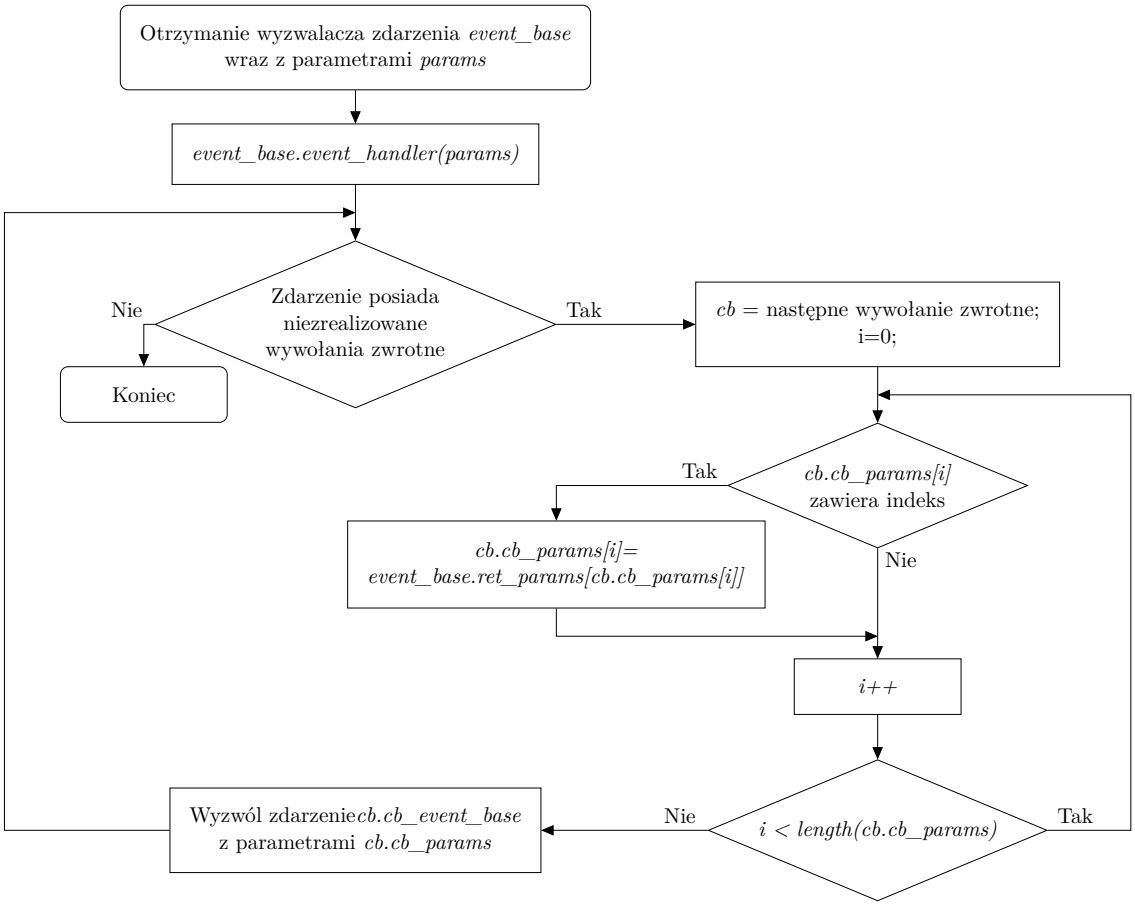
Wszystkie dowiązania powinny być przechowywane w tablicy wiązań, umożliwiając ich zapis na nieulotnej pamięci i odtworzenie w przypadku ponownego uruchomienia urządzenia.

Realizacja zdarzeń

Otrzymanie wyzwalacza przypisanego do określonego zdarzenia, bez znaczenia czy źródło wywołania jest lokalne czy też zewnętrzne, powoduje inicjalizację łańcucha wywołań, przechodzącego od danego zdarzenia, przez zdarzenia dowiązane w formie wywołań zwrotnych, kończąc na zdarzeniu do którego nie zostało utworzone żadne dowiązanie. W poniższym paragrafie zostanie zaprezentowany przebieg realizacji wyzwolonego zdarzenia oraz jego wywołań zwrotnych, uwzględniając proces uzgadniania przekazywanych parametrów wywołania.

Jak zostało przedstawione na schemacie blokowym (rysunek 2.6), pierwszą akcją podejmowaną po otrzymaniu wyzwalacza przypisanego do danego zdarzenia, jest wywołanie zestawu instrukcji obsługi danego zdarzenia (*event_handler*) z przekazanymi parametrami wywołania. Parametry te muszą być przekazane w formacie zgodnym z oczekiwanym przez obsługę zdarzenia, przygotowane do przetwarzania bez potrzeby wcześniejszej ich obróbki bądź formatowania.

Wynikiem realizacji obsługi zdarzenia jest zestaw parametrów zwrotnych *ret_params*, wykorzystywany w dalszym toku przetwarzania zdarzenia. Sekwencyjnie, po zakończeniu obsługi zdarzenia, pobierane są struktury *event_callback* wywołań zwrotnych zdarzeń dowiązanych do aktualnie realizowanego, zapisane w polu *event_base.callback*. Każda z tych struktur posiada pole *cb_params*, zawierające szablon parametrów wywołania, zgodnych w formacie i wymaganych przez zdarzenie wyzwalane w wyniku danego wywołania



Rysunek 2.6.: Schemat blokowy opisujący sposób realizacji zdarzenia oraz jego wywołań zwrotnych
Źródło: opracowanie własne

zwrotnego. Szablon zbudowany jest z atrybutów zawierających wartość stałą, np. łańcuch znaków oraz atrybutów zawierających indeksy. W przypadku indeksowego typu atrybutu, w jego miejsce podstawiana zostaje wartość z *ret_params* o zadanym indeksie. Należy zwrócić uwagę, że sam szablon musi zostać zachowany na potrzeby przyszłych wyzwoleń i obsługi danego zdarzenia.

Po uzgodnieniu wszystkich parametrów wywołania, przekazywany jest wyzwalacz dowiązanego zdarzenia, przypisany do danego wywołania zwrotnego, wraz z wypełnionymi na podstawie szablonu parametrami wywołania. Procedura ta jest powtarzana dla każdego przypisanego wywołania zwrotnego, których ilość ze względu na pojemność indeksu (*event_callback.cb_id*) może wynosić maksymalnie 128. Po wykonaniu wszystkich przypisanych wywołań zwrotnych, realizacja zdarzenia zostaje zakończona.

2.3.1.2 Tablica zdarzeń

Tablica zdarzeń jest strukturą zawierającą w sobie wszystkie zdarzenia udostępniane przez wszystkie aplikacje danego urządzenia, rozpoznawane za pomocą unikalnego (w obrębie urządzenia) 16 bitowego identyfikatora. Zdarzenia przechowywane są w postaci struktury *event_base* wraz z ich wywołaniami zwrotnymi, w postaci struktury *event_callback*.

Zarządzanie zdarzeniami odbywa się za pomocą operacji CRUD (ang. *Create, Read, Update and Delete*) realizowanych za pośrednictwem udostępnianych przez Podwarstwę zarządzania zdarzeniami intefejsów, o postaci zależnej od implementacji. Zdalne dokonywanie operacji CRUD dokonywane jest za pośrednictwem zdarzeń Aplikacji konfiguracji, opisanych w sekcji 2.3.3.2.

2.3.2 Aplikacja sieciowa

Aplikacja sieciowa realizuje zadania przydzielone w referencyjnym modelu OSI warstwie sieciowej, transportowej oraz prezentacji, zajmując się przekazywaniem wiadomości za pośrednictwem dwóch różnych algorytmów, wykrywaniem urządzeń w sieci oraz odkrywaniem tras do nich, dołączaniem do sieci oraz tworzeniem nowej, konwersją wiadomości do formatu wymaganego przez obsługujące je funkcje i vice versa.

Aplikacja sieciowa powinna oferować dwa niezależne mechanizmy przekazywania wiadomości: routing reaktywny, działający w oparciu o algorytm DSR, opisany w 2.3.2.1 oraz propagację wiadomości w oparciu o algorytm zalewania, opisany w 2.3.2.2. Sposób stosowania tych dwóch algorytmów został wyjaśniony i opisany w 2.3.2.4.

2.3.2.1 Routing reaktywny

Routing wiadomości w sieci zrealizowanej na podstawie stworzonego w ramach pracy protokołu, odbywa się przy wykorzystaniu dostosowanego do specyfiki sieci algorytmu routingu DSR (ang. *Dynamic Source Routing*) [BJM96]. Algorytm ten stanowi kompromis elastyczności routingu, prędkości działania oraz ograniczenia ruchu sieciowego niezbędnego do działania algorytmu. Ponadto, algorytmy reaktywne, do których należy DSR są wydajniejsze energetycznie w porównaniu do algorytmów proaktywnych oraz hybrydowych[4].

Algorytm DSR tworzy i przekazuje w nadawanej wiadomości trasę routingu, reprezentowaną jako sekwencja adresów urządzeń, które mają pośredniczyć w przekazywaniu danego pakietu danych. Poszczególne urządzenia po odebraniu wiadomości usuwają swój adres i przekazują wiadomość do następnego urządzenia w sekwencji. W przypadku braku trasy do wybranego urządzenia lub jej zmiany, algorytm zapewnia możliwość poinformowania urządzenia nadającego o błędzie transmisji oraz odnalezienia odpowiedniej trasy.

Każde urządzenie przechowuje pamięć podręczną tras (ang. *route cache*), magazynującą odkryte trasy i umożliwiającą natychmiastowe nadanie wiadomości bez potrzeby wcześniejszej inicjalizacji procesu odkrywania trasy. W przypadku braku odpowiedniego wpisu w pamięci podręcznej, rozpoczynany jest proces odkrywania trasy, w oczekiwaniu na którego zakończenie może być realizowane dalsze przetwarzanie.

Status nadanej wiadomości jest monitorowany przez urządzenie nadające, do którego zostaje zwrócona wiadomość o niepowiedzeniu transmisji na skutek wyłączenia urządzenia docelowego bądź jego przemieszczenia, umożliwiając tym samym podjęcie stosownych działań, jak inicjacja odkrywania nowej trasy do urządzenia.

Zawarty w poniższych paragrafach generalny opis algorytmu jest opisem teoretycznym, bazującym wprost na jego specyfikacji, z którą na potrzeby implementacji należy się za-

[4] N. Kumar, Dr. C. Suresh Gnana Dhass. A Complete Study on Energy Efficient Routing Protocols DSR, ZRP and DSDV In Mobile Ad Hoc Networks. *The International Journal Of Engineering And Science*, Vol. 1, No. 2 (2012), s. 54–60

poznań [BJM96]. Realizacja algorytmu z uwzględnieniem specyfiki protokołu (m.in. zdarzenia, zamiast nagłówków dla wiadomości przeznaczonych dla różnych warstw) została zawarta w specyfikacji zdarzeń Aplikacji sieciowej, opisanych w 2.3.2.8.

Odkrywanie trasy

Odkrywanie trasy pozwala każdemu urządzeniu w sieci na dynamiczne odnalezienie trasy do dowolnego urządzenia należącego do tej samej sieci, w przypadku kiedy urządzenia nie mają bezpośredniego połączenia. Urządzenie inicjujące proces odkrywania trasy, za pomocą algorytmu plotkowania (opisany w następnym paragrafie) propaguje wiadomość typu *RouteRequest*, zawierającą adres poszukiwanego urządzenia, unikalny numer sekwencyjny żądania oraz początkowo pustą listę urządzeń pośredniczących w przekazywaniu wiadomości. Urządzenia posiadają listę w postaci *<adres inicjatora, numer żądania>*, pozwalającą na uniknięcie duplikowania wiadomości oraz przeciążenia sieci poprzez zapętlenie. Każde urządzenie po odebraniu wiadomości typu *RouteRequest* podejmuje następujące kroki:

1. Jeśli lista ostatnich żądań odnalezienia trasy zawiera już dla tego samego nadawcy numer sekwencyjny identyczny z numerem w otrzymanej wiadomości, jest ona odrzucana i proces zostaje na tym urządzeniu przerwany
2. W przeciwnym przypadku, jeśli adres danego urządzenia znajduje się już na przekazanej w wiadomości sekwencji urządzeń pośredniczących, wiadomość jest odrzucana i dalsze przetwarzanie zostaje przerwane
3. W przeciwnym przypadku, jeśli adres poszukiwanego urządzenia jest zgodny z adresem danego urządzenia, do inicjatora odsyłana jest wiadomość *RouteReply* zawierająca sekwencję urządzeń pośredniczących i nadana zgodnie z tą sekwencją
4. W przeciwnym przypadku, jeśli w pamięci podręcznej tras znajduje się adres poszukiwanego urządzenia, do inicjatora odsyłana jest wiadomość *RouteReply* zawierająca połączoną sekwencję adresów urządzeń pośredniczących, adresu danego urządzenia oraz sekwencję adresów z pamięci podręcznej tras danego urządzenia
5. W przeciwnym przypadku, urządzenie dodaje własny adres na końcu sekwencji urządzeń pośredniczących i przekazuje wiadomość korzystając z algorytmu plotkowania

Pamięć podręczna tras może również być uzupełniana w oparciu o wszystkie przekazywane za pośrednictwem urządzenia wiadomości i zawarte w nich trasy.

Przekazywanie wiadomości za pomocą algorytmu DSR realizowane jest w wyniku wyzwolenia zdarzenia *Route*, opisanego w 2.3.2.8. Proces odkrywania trasy obsługiwany powinien być za pomocą algorytmu plotkowania, opisanego w poniższym paragrafie, zamiast oryginalnie przydzielonego do algorytmu propagacji wiadomości przez zalewanie.

Plotkowanie

Plotkowanie (ang. *Gossip*) jest rodzajem komunikacji umożliwiającym tworzenie skalowanych i pewnych systemów rozproszonych. Algorytm plotkowania bazuje na koncepcji iteracyjnej wymiany danych pomiędzy urządzeniami należącymi do sieci, w której każdym kroku dane przekazywane są do losowo wybranego podzbioru urządzeń o liczności f [BHO+99]. Czynność ta jest powtarzana przez urządzenie, aż do momentu ponownego

odebrania wiadomości (porównanie wpisów tablicy *<adres inicjatora, numer żądania>*).

Dzięki zastosowaniu tego algorytmu, możliwe jest przekazanie wiadomości do wybranego urządzenia tudzież wszystkich urządzeń należących do sieci, bez znajomości jej topologii czy tras routingu. Plotkowanie w sieciach bezprzewodowych zapewnia teoretycznie nieskończoną skalowalność, zrównoważone obciążenie sieci, odporność na błędy transmisji oraz wysoki stopień wydajność energetycznej [FGR+07]. W przeciwieństwie do zalewania sieci (ang. *flooding*) wykorzystywanego w oryginalnym algorytmie DSR, plotkowanie nie generuje nadmiernego ruchu sieciowego, zapewniając w skończonym czasie dostarczenie wiadomości do wskazanego urządzenia [HHL02].

Obsługa błędów

Każde urządzenie pośredniczące w routingu wiadomości, musi zapewniać mechanizm potwierdzenia transmisji pomiędzy przeskokami (wiadomości ACK). Dzięki temu możliwa jest detekcja błędu transmisji i w konsekwencji, detekcja niepowodzenia całego routingu z możliwością dokładnego wskazania miejsca przerwania trasy.

Wiadomość typu *RouteError* jest przekazywana do urządzenia inicjującego routing w przypadku niepowodzenia realizacji przekazania wiadomości w kolejnym przeskoku trasy. Urządzenie inicjujące komunikację, po odebraniu wiadomości błędu usuwa pozycje z pamięci podręcznej tras wykorzystujące sekwencję adresów od urządzenia przekazującego komunikat o błędzie do urządzenia docelowego.

2.3.2.2 Algorytm zalewania

Algorytm zalewania (ang. *flooding*) jest jednym z najprostszych algorytmów propagujących wiadomość w obrębie całej sieci, generujący tym samym duży ruch sieciowy oraz straty energetyczne. Jednocześnie jednak, algorytm gwarantuje dotarcie wiadomości do odbiorcy w czasie krótszym niż ma to miejsce w przypadku algorytmu plotkowania, pomimo braku znajomości topologii sieci czy trasy routingu. Algorytm ten powinien być wykorzystywany jednie w sytuacji gdy tradycyjne formy przekazywania wiadomości zawiodą, co zostało opisane w 2.3.2.4.

Algorytm zalewania polega na rozgłaszania wiadomości wśród bezpośrednich sąsiadów urządzenia, którzy z kolei po odebraniu wiadomości powtarzają ten sam proces. W przypadku odebrania ponownie tej samej wiadomości (dla porównania wymagany jest bufor przechowujący identyfikator wiadomości i adres nadawcy), nie jest ona już rozgłaszana.

Przekazywanie wiadomości za pomocą algorytmu zalewania realizowane jest w wyniku wyzwolenia zdarzenia *PanicRoute*, opisanego w 2.3.2.8.

2.3.2.3 Format wiadomości

Wyzwolenie zdarzenia na innym urządzeniu wymaga przekazania identyfikatora danego zdarzenia oraz parametrów wywołania w formie znakowej. W tym celu identyfikator oraz struktura zawierająca parametry musi zostać poddana serializacji oraz odpowiedniemu, wspólnemu dla wszystkich urządzeń formatowaniu, przedstawionemu na rysunku 2.7

Serializacją oraz deserialzacją zajmuje się Aplikacja sieciowa, jako odpowiedzialna za ruch sieciowy realizowany za pośrednictwem protokołu. Każdy atrybut ze struktury *params* przekazywany jest w formie łańcuchów bajtowych znaków, oddzielonych separato-

2 bajty	1 bajt - ...	1 bajt		1 bajt - ...	1 bajt
Identyfikator zdarzenia	Pole 1	Separator ;	...	Pole n	Separator ;

Rysunek 2.7.: Format wiadomości Aplikacji sieciowej
Źródło: opracowanie własne

rem *;* $(0x3B)$. Puste atrybuty parametru przekazywane są w postaci bajtowego znaku o numerze $0x20$. W sytuacji, kiedy atrybut zawiera znak separatora, powinien zostać on zdublowany dla odróżnienia z pojedynczym znakiem separatora pól wiadomości.

2.3.2.4 Przekazywanie wiadomości

Przekazywanie wiadomości jest elementem kluczowym dla umożliwienia wyzwalania zdarzeń pomiędzy urządzeniami i zapewnienia działania całego systemu opartego na protokole. Dla zwiększenia skuteczności komunikacji, w protokole zostały zdefiniowane dwa mechanizmy propagacji wiadomości: routing w oparciu o algorytm DSR (opisany w 2.3.2.1) oraz tzw. zalewanie (opisane w 2.3.2.2). Drugi z wymienionych algorytmów, ze względu na stopień w jakim pochłania zasoby sieci (energia urządzeń i pasmo kanału) powinien być stosowany jedynie w sytuacji gdy nadawana wiadomości ma wysoki priorytet oraz gdy:

a) w pamięci podręcznej tras algorytmu DSR nie ma wybranego adresu docelowego, lub

b) po nadaniu wiadomości za pomocą algorytmu DSR zwracana jest wiadomość *Route-Error*

Motywacją takiego działania jest wymóg zagwarantowania dostarczenia wiadomości w czasie akceptowalnym przez użytkownika, pod warunkiem fizycznej możliwości realizacji tego założenia. Po nadaniu wiadomości za pośrednictwem algorytmu zalewania, powinien zostać rozpoczęty proces odnajdywania trasy do danego urządzenia docelowego. W przypadku niepowodzenia w poszukiwaniu trasy do danego urządzenia, jego adres powinien zostać dodany na listę adresów periodycznego poszukiwania, podejmowanego z malejącą częstotliwością aż do całkowitego zaprzestania poszukiwania i usunięcia adresu z listy.

2.3.2.5 Dołączanie do sieci

Każde urządzenie w celu dołączania do istniejącej sieci opartej na protokole, jest zobligowane do ustalenia adresu sieci do której ma przynależeć oraz ustalenie własnego, unikalnego adresu w tej sieci. O ile ustalenie adresów poszczególnych urządzeń jest procesem realizowanym automatycznie na poziomie protokołu, to w przypadku obecności kilku sieci bezprzewodowych wybór właściwej podlegać musi decyzji użytkownika, wymagając tym samym interakcji z jego strony.

Wybór adresu sieci

Pierwszym krokiem przyłączania do sieci jest ustalenie jej adresu. W tym celu urządzenie dołączające przechodzi iteracyjnie przez każdy adres sieci, w zakresie 0–127, rozgłaszając

wśród sąsiadów wyzwalacz zdarzenia *IntroduceYourself* i oczekując $200ms$ na odpowiedź zawierająca dane sąsiada (m.in. adres sieci, adres urządzenia w sieci, identyfikator typu urządzenia, opisane w 2.3.3.3). Po zakończeniu tego procesu, w przypadku odkrycia na podstawie otrzymanych odpowiedzi więcej niż jednej sieci, urządzenie wyłącza filtrowanie wiadomości i nasłuchuje na odpowiedź nadzorcy o możliwości przyłączenia się do danej sieci. Umożliwia to tym samym podjęcie decyzji przez użytkownika, do której sieci ma nowe urządzenie się przyłączyć.

Alternatywnym rozwiązaniem, zależnym jednak o funkcji oferowanych przez warstwę fizyczną, jest wybór sieci, do której przynależący sąsiad oferuje największą siłę sygnału. W przeważającej większości przypadków, oznaczać będzie to urządzenie najbliższe, czyli z dużym prawdopodobieństwem urządzenie znajdujące się w tym samym pomieszczeniu i należące do sieci obsługującej dane mieszkanie.

W przypadku nie wykrycia żadnej dostępnej w zasięgu urządzenia sieci, rozpoczynany jest proces tworzenie nowej sieci, opisany w 2.3.2.7.

Wybór adresu urządzenia

Po wybraniu adresu sieci, w której urządzenia będzie funkcjonować, protokół podejmuje działania mające na calu ustalenie unikalnego adresu danego urządzenia w obrębie sieci. W tym celu, urządzenie ponownie rozgłasza wśród sąsiadów wyzwalacz zdarzenia *IntroduceYourself*, a następnie na podstawie otrzymanych odpowiedzi wybiera własny adres większy o jeden od najwyższego adresu sąsiada. Krok też może zostać pominięty, jeśli podczas fazy poszukiwania adresu sieci, odebrane wiadomości od sąsiadów były zachowywane w pamięci podręcznej o postaci <*adres sieci, najwyższy adres sąsiada*>. Wymaga to jednak dodatkowych zasobów pamięci w mikroprocesorze.

Następnie urządzenie przechodzi do drugiej fazy ustalania adresu, w której wywołuje proces Odkrywania trasy algorytmu DSR z zadanym wybranym w poprzednim kroku adresem do wyszukania. W przypadku braku otrzymania odpowiedzi *RouteReply*, urządzenie na stałe przyjmuje wybrany adres. W przeciwnym, zwiększa wybrany adres o jeden i ponawia procedurę poszukiwania urządzenia z identycznym adresem, aż do momentu wybrania wolnego adresu.

2.3.2.6 Ponowne dołączanie do sieci

Sytuacja, w której urządzenie zostaje na jakiś czas pozbawione zasilania w wyniku jego awarii czy zmiany lokalizacji, jest bardzo prawdopodobna. W takim przypadku, po uruchomieniu urządzenia w pierwszej kolejności sprawdzany jest zapisany adres urządzenia oraz sieci do której przynależy. Jeśli takie wpis nie zostanie odnaleziony, urządzenie może rozpocząć proces dołączania do sieci, opisany w 2.3.2.5. W przeciwnym przypadku urządzenie korzystając z zapisanych adresów, powinno rozgłosić wśród sąsiadów wyzwalacz zdarzenia *IntruduceYourself*. Po odebraniu odpowiedzi, protokół porównuje listę otrzymanych adresów sąsiadów z tą przechowywaną w pamięci podręcznej tras. Jeśli lista uległa zmianie w co najmniej 50%, co oznacza w większości przypadków iż topologia lub miejsce w topologii uległo radykalnej zmianie, inicjowany jest proces odkrywania dla wszystkich dotychczas przechowywanych adresów docelowych. Jeśli procent zmian sąsiadów zawiera się w przedziale 1%–49%, procesy odkrywania trasy inicjowane są jedynie dla

tras wykorzystujących utraconych sąsiadów. W przypadku żadnej zmiany w sieci (0%), nie podejmowane są żadne akcje i urządzenie może kontynuować normalną pracę.

Przypadkiem szczególnych jest sytuacja, w której urządzenie nie otrzymało żadnej odpowiedzi na zdarzenie *IntroduceYourself*. W wypadku takim, zachowane adresy oraz pamięć podręczna tras jest czyszczona i rozpoczynany jest proces przyłączania do sieci lub tworzenia nowej.

2.3.2.7 Tworzenie sieci

Tworzenie nowej sieci jest rozpoczynane, w przypadku zakończenia skanowania adresów sieci, nie otrzymując ani jednej odpowiedzi. W takiej sytuacji, urządzenie ustala adres sieci oraz własny na 0, stanowiąc tym samym zerowe urządzenie zerowej sieci.

Urządzenia mogą ponadto być odgórnie przewidziane jako inicjatorzy nowej sieci. W takim przypadku urządzenie podobnie jak miało to miejsce podczas dołączenia do istniejącej sieci, przeprowadza skanowanie wszystkich adresów sieci w poszukiwaniu ewentualnych sieci graniczących. Następnie, po zakończeniu skanowania, urządzenia wybiera adres sieci większy o 1 od największego otrzymanego lub zerowy, w przypadku nie odebrania żadnej odpowiedzi.

2.3.2.8 Zdarzenia aplikacji sieciowej

Tablica 2.2.: Wykaz zdarzeń wymaganych przez Aplikację sieciową, podany wraz z identyfikatorami i opisami

Zdarzenie	**Identyfikator**	**Opis**
Route	$0x0001$	Przekazanie wiadomości do wskazanego urządzenia, przy wykorzystaniu algorytmu routingu opisanego w 2.3.2.1 lub rozgłoszenie wiadomości do urządzeń posiadających bezpośrednie połączenie (sąsiadów)
RouteRequest	$0x0002$	Żądanie odnalezienia trasy, wykorzystywane przez algorytm routingu opisany w 2.3.2.1
RouteResponse	$0x0003$	Odpowiedź na żądanie odnalezienia trasy, wykorzystywana przez algorytm routingu opisany w 2.3.2.1
RouteError	$0x0004$	Zgłoszenie błędu podczas routingu wiadomości
PanicRoute	$0x0006$	Przekazanie wiadomości do wskazanego urządzenia lub wszystkich urządzeń przy wykorzystaniu *zalewania* opisanego w 2.3.2.2
LeaveNetwork	$0x0007$	Opuszczanie sieci przez urządzenie obsługujące zdarzenie

Zdarzenie Route

Przekazanie wiadomości do wskazanego urządzenia, o adresie z zakresu 0–126 ($0x00$–$0xFE$), przy wykorzystaniu algorytmu routingu opisanego w 2.3.2.1. Zdarzenie oferuje również rozgłoszenie wiadomości do urządzeń posiadających bezpośrednie połączenie (sąsiadów), ustawiane za pomocą adresu docelowego o wartości 127($0xFF$).

Format parametrów
sender;receiver;route;message;

Parametry

sender
: Bajtowy adres oryginalnego nadawcy wiadomości zlecającego routing, nie podlegający zmianie przez całą trasę

receiver
: Bajtowy adres docelowego odbiorcy wiadomości, niewykorzystywany bezpośrednio podczas przekazywania sobie wiadomości, jednak użyteczny dla szybkiego sprawdzenia adresu odbiorcy

route
: Sekwencja bajtowych adresów urządzeń pośredniczących w komunikacji, posortowana w kolejności przechodzenia wiadomości przez urządzenia, udostępniana przez pamięć podręczną tras algorytmu DSR. Sekwencja może być dowolnej długości, zapewniającej jednak jednocześnie routing wiadomości w czasie akceptowalnym przez użytkownika

message
: Dane przekazywanej wiadomości, których format oraz długość są całkowicie dowolne, pod warunkiem nie przekraczania maksymalnego rozmiaru ramki warstwy fizycznej. W przypadku potrzeby wysłania większej ilości danych, funkcja zlecająca routing jest odpowiedzialna za podział danych oraz funkcja odbierająca za ich ponowne złożenie

Zdarzenie RouteRequest

Zdarzenie odpowiedzialne za mechanizm Odkrywania trasy należący do algorytmu DSR. Jego wyzwolenie na urządzeniu inicjującym odkrywanie powoduje rozpoczęcie propagacji wyzwalacza tego zdarzenia na pozostałych urządzenie, przy użyciu algorytmu plotkowania. Wyzwolenie tego zdarzenia na pozostałych urządzeniach, zgodnie z algorytmem DSR, powoduje dodanie adresu danego urządzenia na końcu pola *route* (które początkowo zawiera jedynie adres inicjatora odkrywania trasy) oraz dalsze rozpropagowanie wyzwalacza z zaktualizowanym parametrem przy użyciu plotkowania.

Po osiągnięciu poszukiwanego urządzenia, wyzwalane jest na nim zdarzenie *RouteRespone* z odwrócą sekwencją urządzeń z parametrów zdarzenia *RouteRequest* jako trasą.

Format parametrów
sender;receiver;request_id;route;

Parametry

sender
: Bajtowy adres oryginalnego nadawcy wiadomości inicjującego proces odkrywania trasy, wykorzystywany podczas sprawdzania, czy dane urządzenie uczestniczyło już w odkrywaniu trasy, zainicjowanemu przez urządzenie o podanym adresie i danym identyfikatorze procesu odkrywania

receiver

Bajtowy adres urządzenia do którego trasa jest poszukiwana w danym procesie odkrywania trasy. Adres musi zawierać się w przedziale 0–126 ($0x00$–$0xFE$), ze względu na rezerwację adresu 127($0xFF$) dla rozgłaszania

request_id

Numer identyfikacyjny danego procesu odkrywania, liczony niezależnie przez każde urządzenie zlecające Odkrywanie trasy i ikrementowany przy każdym nowym zainicjowaniu tego mechanizmu. Maksymalna wartość identyfikatora nie jest określona, nie powinna być jednak zbyt duża dla nadmiernego rozrostu długości wiadomości oraz zbyt mała, dla odrzucania *RouteRequest* przez algorytm DSR

route

Sekwencja bajtowych adresów urządzeń, które pośredniczyły w przekazywaniu wyzwalacza *RouteRequest* a co za tym idzie, urządzeń tworzących trasę do poszukiwanego urządzenia. Sekwencja ta jest rozbudowywana przez każde urządzenie biorące udział w procesie odkrywania trasy poprzez dodanie własnego adresu na końcu listy

Zdarzenie RouteResponse

Odpowiedź na żądanie odkrywania trasy, wykorzystywana przez algorytm routingu DSR. Zdarzenie wyzwalane w momencie osiągnięcia przez wyzwalacza zdarzenia*RouteRequest* urządzenia o poszukiwanych adresie, z odwrócą sekwencją urządzeń z parametrów zdarzenia *RouteRequest* jako trasą powrotu. Trasa otrzymana w wyniku *RouteRequest* jest również przekazywana bez żadnych modyfikacji za pomocą *RouteReponse* do urządzenia inicjującego proces odkrywania trasy, za pośrednictwem parametru *saved_route*.

Format parametrów

sender;receiver;request_id;route;saved_route;

Parametry

sender

Bajtowy adres urządzenia inicjującego odpowiedź na *RouteRequest*. Parametry niewykorzystywany bezpośrednio przez algorytm aczkolwiek użyteczny dla określania oryginalnego nadawcy wiadomości

receiver

Bajtowy adres urządzenia które pierwotnie zleciło proces Odkrywania trasy, do którego jest kierowana zapisana trasa. Parametr niewykorzystywany bezpośrednio przez algorytm aczkolwiek użyteczny dla określanie docelowego odbiorcy wiadomości

request_id

Identyfikator żądania *RouteRequest* w wyniku którego otrzymania zostało wyzwolone zdarzenia *RouteRespone*

saved_route

Sekwencja bajtowych adresów urządzeń pośredniczących w komunikacji, posortowana w kolejności przechodzenia wiadomości przez urządzenia, pozyskana z żądania *RouteRequest*, o odwróconej kolejności bajtowych adresów

zapisana_trasa

Sekwencja bajtowych adresów urządzeń pośredniczących w przekazywaniu żądania *RouteRequest* wyzwalającego *RouteRespone* i pozyskana z tego żądania, nie poddana żadnym modyfikacjom

Zdarzenie PanicRoute

Zdarzenie odpowiedzialne za propagacje wiadomości w sieci przy wykorzystaniu algorytmu zalewania, opisanego w 2.3.2.2. W efekcie jego wyzwolenia, każde urządzenie w sieci otrzymuje kopię zawartej w parametrach wywołania wiadomości poprzez kaskadowe wyzwolenie tego samego zdarzenia u wszystkich urządzeniach pozostających w bezpośrednim połączeniu (sąsiadów). Każde urządzenie może jedynie raz pośredniczyć w propagacji wiadomości o danym identyfikatorze, rozsyłanej przez danego nadawcę, jak zostało to przedstawione w opisie algorytmu zalewania.

Format parametrów

sender;receiver;message_id;message;

Parametry

sender

Bajtowy adres urządzenia inicjującego propagacje wiadomości realizowaną przez algorytm zalewania, wykorzystywany wraz z identyfikatorem do określania czy dane urządzenie uczestniczyło już w rozsyłaniu danej wiadomości

receiver

Bajtowy adres urządzenia docelowego

message_id

Identyfikator wiadomości propagowanej za pomocą algorytmu zalewania, liczony niezależnie od algorytmu DSR dla każdego urządzenia w sieci, wykorzystywany wraz z adresem nadawcy do określania czy dane urządzenie uczestniczyło już w rozsyłaniu danej wiadomości

message

Dane przekazywanej wiadomości, których format oraz długość są całkowicie dowolne, pod warunkiem nie przekraczania maksymalnego rozmiaru ramki warstwy fizycznej. W przypadku potrzeby wysłania większej ilości danych, funkcja zlecająca propagacje wiadomości przez zalewanie, jest odpowiedzialna za podział danych oraz funkcja odbierająca za ich ponowne złożenie

Zdarzenie LeaveNetwork

Zdarzenie powoduje opuszczenie urządzenia sieci do której przynależy. W jego wyniku urządzenie czyści swoją tablicę zdarzeń z dowiązań, pamięć podręczną tras oraz zapisany adres urządzenie oraz adres sieci, aby następnie przejść w stan uśpienia. Po ponownym włączeniu urządzenia, będzie się ono zachowywało jak urządzenie fabryczne nowe, rozpoczynając proces dołączania do sieci.

Zdarzenie najczęściej wyzwalane zdalnie przez nadzorcę systemu, na zlecenie użytkow-

nika.

(brak parametrów)

2.3.3 Aplikacja konfiguracji

Aplikacja konfiguracji odpowiedzialna jest za szeroko rozumianą konfigurację pracy urządzenia, jego aplikacji oraz zachowania czy relacji z innymi urządzeniami działającymi w sieci. Poprzez udostępnione przez aplikację zdarzenia, możliwe jest dokonywanie odpowiednich zmian w urządzeniu zdalnie, dostosowując jego działanie do oczekiwań użytkownika.

Aplikacja konfiguracji udostępnia zdarzenia operacji CRUD realizowanych na tablicy routingu oraz tablicy zdarzeń, przekazywania statusu urządzenia oraz podstawowych informacji o nim.

2.3.3.1 Status urządzenia

Aplikacja konfiguracji przechowuje specjalną strukturę zawierającą parametry definiujące i opisujące aktualny stan urządzenia. Poza parametrami oczywistymi i wspólnymi dla wszystkich urządzeń, jak aktualny tryb pracy (sleep, ready czy standby), każde rodzaj urządzenia posiada własny zestaw parametrów określających elementy stanu charakterystyczne dla niego. Przykładowo, dla urządzeń zasilanych bateryjnie będzie to aktualny poziom naładowania, dla urządzeń działających jako stacje pogodowe będzie to temperatura powietrza i wilgotność.

Dla zachowania integralności, ważne jest aby każda aplikacja reprezentująca określony zestaw funkcji urządzenia modyfikowała tylko i wyłącznie parametry do niej przypisane. Rozwijając to założenie, parametry określające status urządzenia nie mogą być modyfikowane przez jakiekolwiek inne urządzenie należące do sieci, a jedynie odczytywane po wcześniejszym dowiązania do zdarzenia *StatusChanged*.

Poszczególne parametry stanu urządzenia mogą zmieniać się w wyniku zdarzeń wewnętrznych (zmiana temperatury na czujniku) bądź zewnętrznych (otrzymanie wyzwalacza zdarzenia powodującego włączenie żarówki). W każdym z tych przypadków, zmiana parametru stanu powinna skutkować przesłaniem całej struktury do urządzeń oczekujących na to, poprzez dowiązanie wywołania zwrotnego do zdarzenia zmiany statusu *StatusChanged*.

Parametry do dużej dynamice, jak np. położenie akcelerometru czy wartość temperatury, nie powinny za każdą zmianą powodować wyzwolenia zdarzenia *StatusChanged*, wpływając tym samym niekorzystnie na konsumpcję energetyczną danego urządzenia (wyzwalanie dowiązanych zdarzeń) jak i dużej części urządzeń w sieci (routing wiadomości zawierającej aktualizację statusu urządzenia). Należy ustalić próg zmiany, po którego przekroczeniu zostaje wyzwolone zdarzenie.

Celem statusu urządzenia jest jego prezentacja użytkownikowi za pośrednictwem nadzorcy systemy lub innego urządzenia pełniącego rolę interfejsu dostępowego dla człowieka. Biorąc pod uwagę to kryterium, parametrami statusu urządzenia powinny być wartości interesujące bądź ważne dla użytkownika, a nie dla całego systemu.

Zalecanym formatem dla przekazywanych wartości statusu, jest ten proponowany dla formatu wiadomości Aplikacji sieciowej, opisany w 2.7, w którym każde pole oddzielone

jest za pomocą znaku średnika. Dopuszczalne jest również formatowanie danych w formie JSON[5] lub XML[6], jednak formaty te ze względu na narzut rozmiaru i czasu przetwarzania nie są zalecane.

2.3.3.2 Operacje CRUD

CRUD (ang. *Create, Read, Update, Delete*) stanowią cztery podstawowe funkcje umożliwiające pełne zarządzanie zbiorem danych przechowywanym przez aplikację. W przypadku urządzeń działających zgodnie ze standardem, operacje tego typu mogą być dokonywane na:

- tablicy zdarzeń, przechowującej zdarzenia danego urządzenia i ich wywołania zwrotne (Podwarstwa zarządzania zdarzeniami). Operacje CRUD realizowane na tej tablicy pozwalają na wiązanie zdarzeń w formie wywołań zwrotnych, usuwanie dowiązań, ich aktualizację (np. zmiana wartości stałej w parametrze wywołania) oraz usunięcie
- pamięci podręcznej tras (Aplikacja sieciowa). Operacje CRUD wykonywane na tej strukturze umożliwiają zdalne zarządzenie routingiem, tworząc nowe połączenia bez potrzeby inicjowania procesu Odkrywania trasy, balansowanie routingu czy usuwanie ścieżek niezalecanych (np. omijając urządzenie bateryjne, któremu pozostało już niewiele energii)

Zarówno operacje wykonywane na tablicy zdarzeń oraz na pamięci podręcznej tras podejmowane są na zlecenie urządzenia działającego jako nadzorca systemu, realizujący te akcje najczęściej w wyniku działań użytkownika (np. wiązanie zdarzeń) oraz opcjonalnych algorytmów działających na tym urządzeniu.

Sposób działania operacji CRUD wykonywanych na tablicy zdarzeń oraz pamięci podręcznej tras zdefiniowany jest w opisach odpowiednich zdarzeń, odpowiedzialnych za dane operacje.

2.3.3.3 Informacje o urządzeniu

Każde urządzenie w warstwie aplikacji przechowuje zestaw danych określających model urządzenia, jego wytwórcę, wersję oprogramowania oraz adres w sieci. Informacje te pomagają m.in. nadzorcy określić funkcje urządzenia i jego możliwości, natomiast inne urządzenia w sieci mogą za pośrednictwem tych danych określić adresy sąsiadów. Dane przekazywane są w ściśle określonym formacie, przedstawionym na rysunku 2.8 i propagowane w wyniku wyzwolenia zdarzenia *IntroduceYourself*.

Adres urządzenia
: Bajtowy adres danego urządzenia w sieci, określany przez Aplikację sieciową podczas dołączania urządzenia do sieci lub tworzenia nowej (2.3.2.7).

Adres sieci
: Bajtowy adres sieci do której przynależy dane urządzenie

[5] JSON (ang. JavaScript Object Notation) – lekki tekstowy format wymiany danych komputerowych, opisany w dokumencie RFC 4627

[6] XML (ang. Extensible Markup Language) język znaczników pozwalający na tekstową reprezentację różnych w sposób strukturalny, standard W3C

1 bajt	1 bajt	2 bajty	2 bajty	2 bajty	0 - ...
Adres urządzenia	Adres sieci	Wytwórca	Model	Wersja oprogramowania	Nazwa urządzenia

Rysunek 2.8.: Format danych przechowujących informacje o urządzeniu
Źródło: opracowanie własne

Wytwórca
Dwubajtowy identyfikator wytwórcy danego urządzenia, nadawany na etapie produkcji. Ułatwia on określenie modelu urządzenia, tworząc logiczny podział zbioru wszystkich modeli urządzeń na poszczególnych wytwórców

Model
Dwubajtowy identyfikator określający model urządzenia, stanowiący po identyfikatorze wytwórcy kolejny stopień wyszczególnienia identyfikacji danego urządzenia

Wersja oprogramowania
Dwubajtowy identyfikator określający wersję oprogramowania obsługującą dane urządzenie, może ulegać zmianie po kolejnych aktualizacjach firmware. Jest on ostatecznym klasyfikatorem pozwalającym określenie funkcji oferowanych przez dane urządzenie

Nazwa urządzenia
Parametr opcjonalny, o długości nieokreślonej. Pozwala on wytwórcy na nadawanie poszczególnym urządzeniom predefiniowanych nazw lub numerów produkcyjnych, umożliwiający identyfikację urządzenia co do sztuki, a nie tylko modelu czy wersji oprogramowania

Urządzenie nadzorujące, kierując się przedstawionymi danymi, powinno być zdolne do zaprezentowania użytkownikowi pełnych możliwości i funkcji oferowanych przez dane urządzenie.

2.3.3.4 Zdarzenia aplikacji konfiguracji

Zdarzenie IntroduceYourself

Zdarzenie odpowiedzialne za przekazanie podstawowych informacji o danym urządzeniu, opisanych w 2.3.3.3, do urządzenia zlecającego wyzwolenie zdarzenia.

Format parametrów
orderer;

Parametry

orderer
Adres w sieci urządzenia zalecającego odczyt informacji o danym urządzeniu, do którego też zwrotna informacja ma docelowo dotrzeć

Tablica 2.3.: Wykaz zdarzeń wraz z identyfikatorami i opisami, wymaganych przez Aplikację konfiguracji

Zdarzenie	**Identyfikator**	**Opis**
IntroduceYourself	$0x000A$	Przekazanie podstawowych informacji o danym urządzeniu, opisanych w 2.3.3.3.
StatusChanged	$0x000B$	Zdarzenie wyzwalane dla zmiany któregoś z parametrów definiujących status danego urządzenia i przekazuję aktualizację statusu do urządzeń oczekujących.
RoutingTableCRUD	$0x000C$	Zdarzenie odpowiedzialne za zarządzanie trasami zapisanymi w pamięci podręcznej tras algorytmu DSR.
RoutingTableReturn	$0x000D$	Zdarzenie zwrotu odczytanych danych w wyniku operacji CRUD na pamięci podręcznej tras innego urządzenia
EventsBindingCRUD	$0x000E$	Zdarzenie odpowiedzialne jest za zdalne dokonywanie operacji CRUD na tablicy zdarzeń danego urządzenia.
EventsBindingReturn	$0x0010$	tu odczytanych danych w wyniku operacji CRUD na tablicy zdarzeń innego urządzenia
SetOutputPower	$0x0020$	Zdarzenie odpowiedzialne za ustalenie mocy wyjściowej nadajnika podczas transmisji

Zdarzenie StatusChanged

Zdarzenie wywoływane w momencie zmiany któregoś z parametrów definiujących status danego urządzenia. W wyniku jego wyzwolenia, status jest przekazywany w formie wywołania zwrotnego do dowiązanych zdarzeń na pozostałych urządzeniach, stanowiących najczęściej bramki do interfejsu użytkownika.

(brak parametrów)

Zdarzenie RoutingTableCRUD

Zdarzenie odpowiedzialne za zarządzanie trasami zapisanymi w pamięci podręcznej tras algorytmu DSR. Dzięki użyciu tego zdarzenia, możliwe jest zdalnie tworzenie nowych bądź modyfikowanie istniejących tras routingu, umożliwiając tym samym pominięcie procesu Odkrywania trasy. Urządzenie zlecające wykonanie operacji CRUD, przy znajomości topologii całej sieci oraz po zaimplementowaniu odpowiednich algorytmów, może dokonywać balansu obciążania sieci, rozkładając ruch równomiernie lub np. starając się przekierować go na urządzenie posiadające stałe źródło zasilania, odciążając tym samym urządzenia bateryjne.

Format parametrów
orderer;operation;[route;]EOO;operation...

Parametry

orderer

Bajtowy adres urządzenia zlecającego daną operację. Wymagany dla realizacji operacji odczytu

operation

Bajtowy identyfikator operacji CRUD:

- C $(0x43)$ – utworzenie trasy
- R $(0x52)$ – odczyt wybranego rekordu lub wszystkich rekordów
- U $(0x55)$ – aktualizacja wybranej trasy
- D $(0x44)$ – usunięcie wybranej trasy

route

Parametr opcjonalny, wykorzystywany podczas aktualizacji oraz tworzenia trasy. Sekwencja bajtowych adresów ustawionych w porządku przechodzenia wiadomości od danego urządzenia do docelowego, bez formatowania w postaci separatorów. W zależności od wykorzystanej komendy, dana trasa może być zaktualizowana bądź utworzona na podstawie podanego parametru *route*

EOO

Słowo końca komendy operacji CRUD o postaci $0xA5A$. Po jego dodaniu może zostać przekazana kolejna komenda operacji CRUD, pod warunkiem dostępności wolnego miejsca w ramce warstwy fizycznej

Zdarzenie RoutingTableReturn

Zdarzenie zwrotu odczytanych danych w wyniku operacji CRUD na pamięci podręcznej tras innego urządzenia. Wyzwalane przez urządzenie na którego pamięci podręcznej tras było realizowana operacja CRUD, na podstawie przekazanego w parametrze wywołania adresu urządzenia zlecającego (*orderer*). Poszczególne trasy przekazywane są w postaci sekwencji oddzielonych średnikami.

Format parametrów

sender;route;...

Parametry

sender

Bajtowy adres urządzenia na którym realizowana była operacja CRUD generująca odpowiedź

route

Sekwencja bajtowych adresów urządzeń bez separatorów, tworzących trasę

Zdarzenie EventsBindingCRUD

Zdarzenie *EventsBindingCRUD* odpowiedzialne jest za zdalne dokonywanie operacji CRUD na tablicy zdarzeń danego urządzenia, czyli tworzenie, aktualizację, usuwanie oraz odczyt dowiązań. Funkcje te są o tyle istotne, iż pozwalają na dostosowanie zachowania systemu oraz relacji zachodzących pomiędzy urządzeniami do potrzeb użytkownika.

Format parametrów
orderer;operation;event_id;cb_id;[cb_event_id;cb_params;]EOO;operation...

Parametry

orderer
: Bajtowy adres urządzenia zlecającego daną operację. Wymagany dla realizacji operacji odczytu

operation
: Bajtowy identyfikator operacji CRUD:
  - C ($0x43$) – utworzenie dowiązania
  - R ($0x52$) – odczyt wybranego rekordu lub wszystkich rekordów
  - U ($0x55$) – aktualizacja wybranego rekordu
  - D ($0x44$) – usunięcie wybranego rekordu

event_id
: Dwubajtowy identyfikator zdarzenia, którego wywołania zwrotne mają być edytowane. W przypadku operacji odczytu, dopuszczalne jest użycie zamiast identyfikatora kwalifikatora gwiazdki *, oznaczającego wybór wszystkich zdarzeń przechowywanych w tablicy zdarzeń danego urządzenia

cb_id
: Bajtowy identyfikator wywołania zwrotnego poddawanego operacjom CRUD, dopuszczalny jest również kwalifikator podwójnej gwiazdki **, oznaczający wybór wszystkich wywołań zwrotnych dla danego zdarzenia

cb_event_id
: Parametr opcjonalny, wykorzystywany dla operacji aktualizacji oraz tworzenia, pozwalający na dowiązanie zdarzenia o podanym bajtowym identyfikatorze w formie wywołania zwrotnego

cb_params
: Parametr opcjonalny, wykorzystywany dla operacji aktualizacji oraz tworzenia. Kolekcja atrybutów przekazywanych do zdarzenia wyzwalanego w formie wywołania zwrotnego w formacie opisanym w 2.7 (pole1;pole2...)

EOO
: Słowo końca komendy operacji CRUD o postaci $0xA5A$. Po jego pojawieniu może zostać przekazana kolejna komenda operacji CRUD, pod warunkiem dostępności wolnego miejsca w ramce warstwy fizycznej

Zdarzenie BindingTableReturn

Zdarzenie zwrotu odczytanych danych w wyniku operacji CRUD na tablicy zdarzeń innego urządzenia. Wyzwalane przez urządzenie na którego tablicy zdarzeń było realizowana operacja CRUD, na podstawie przekazanego w parametrze wywołania adresu urządzenia zlecającego (*orderer*). W przypadku odczytu więcej niż jednego rekordu z tablicy zdarzeń, poszczególne rekordy oddzielone są słowem końca komendy *EOO*.

Format parametrów
sender;event_id;cb_id;cb_event_id;cb_params;]EOO;event_id...

Parametry

sender
: Bajtowy adres urządzenia na którym realizowana była operacja CRUD generująca odpowiedź

event_id
: Dwubajtowy identyfikator zdarzenia, którego wywołania zwrotne są zwracane w danej sekcji odpowiedzi (sekcję są oddzielone słowem *EOO*)

cb_id
: Bajtowy identyfikator wywołania zwrotnego zwracanego w wyniku operacji CRUD

cb_event_id
: Dwubajtowy identyfikator dowiązanego zdarzenia wyzwalanego w formie wywołania zwrotnego

cb_params
: Kolekcja atrybutów przekazywanych do zdarzenia wyzwalanego w formie wywołania zwrotnego w formacie opisanym w 2.7 (pole1;pole2...), dla danego odczytywanego rekordu

EOO
: Słowo końca operacji o postaci $0xA5A$. Po jego pojawieniu może zostać przekazana kolejny odczytany rekord, pod warunkiem dostępności wolnego miejsca w ramce warstwy fizycznej

Zdarzenie SetOutputPower

Zdarzenie odpowiedzialne za ustawienie mocy wyjściowej nadajnika układu komunikacji bezprzewodowej w czasie transmisji. Zdarzenie wykorzystywane podczas dostosowywania balansu ruchu sieciowego przez nadzorcę systemu, mające na celu ograniczenie konsumpcji energii przez urządzenia.

Format parametrów
percentage;

Parametry

percentage
: Wartość z przedziału 0–100. Procent mocy wyjściowej nadajnika, który ma zostać ustawiony. Większość modułów komunikacji bezprzewodowej oferuje parę wartości dopuszczalnych do ustawienia, wartość powinna więc być mapowana na te oferowane przez urządzenie, z zaokrąglaniem w górę.

2.4 Implementacja protokołu

Podczas definiowana stosu protokołu oraz stawianych mu wymogów i zadań, bardzo istotnym kryterium była możliwość implementacji protokołu przy użyciu ogólnodostępnych narzędzi oraz podzespołów. Co ważne, protokół został zaprojektowany nie uwzględniając możliwości konkretnego mikrokontrolera czy modułu komunikacji bezprzewodowej, pozostając niezależnym od platformy sprzętowej.

Niezależność od platformy sprzętowej, osiągnięta pośrednio przez otwartość, pozwolić ma docelowo na tworzenie szerokiej gamy urządzeń współdziałających z systemem, posiadających możliwości i charakterystyki dopasowane do realizowanych przez nie zadań. Przykładowo, budowa zwykłego dwustanowego włącznika światła w oparciu o procesor ARM jest daleko idącą przesadą, jednak ten sam procesor będzie wymagany dla stabilnej pracy urządzenia służącego rozpoznawaniu twarzy. Jednocześnie, uniwersalność konstrukcyjna powinna oferować identyczny sposób implementacji wymaganej funkcjonalności niezależnie od wybranej platformy sprzętowej, z wykorzystaniem nawet tego samego kodu. Ujednolicenie procesu tworzenia aplikacji dla różnych opcji sprzętowych umożliwić ma szybką popularyzację systemu i protokołu, udostępniając jednorodny zbiór przykładów oraz gotowych rozwiązań. Podejście takie jest niezwykle istotne dla popularyzacji proponowanych rozwiązań, czego doskonałym przykładem jest otwarta programistyczna platforma Arduino[7], stanowiąca swoistą bazę dla wszystkich projektów wymagających wcześniejszego stworzenia prototypu.

Uniwersalność powinna być jednak rozumiana w kontekście zarówno dowolności wytwórcy w wyborze wykorzystywanej platformy sprzętowej czy środowiska programistycznego, jak i swobody końcowego użytkownika w dopasowaniu działania całego systemu do realizacji wielu różnorodnych celów.

Uniwersalność rozpatrywana w kontekście końcowego użytkownika powinna zapewniać możliwość adaptacji całego systemu w różnych środowiskach ukierunkowanych na realizację zróżnicowanych celów. Jednocześnie, system powinien zachowywać swój ogólny, statyczny charakter systemu automatyki domowej. Niezbędność takiej restrykcji wynika z konieczności wyważenia proporcji pomiędzy uniwersalnością a prostotą, której zachwianie prowadzić może do nadmiernego skomplikowania protokołu komunikacyjnego, jak ma to miejsce np. w przypadku ZigBee® lub do przesadnego uproszczenia, które ograniczałoby funkcjonalność i możliwe zastosowania systemu.

Realizacja wymogów dualistycznego podejścia do uniwersalności, rozpatrywanej zarówno kontekście użytkownika, jak i twórcy, wymaga integralności i interoperacyjności proponowanego rozwiązania. Zakładanym sposobem realizacji tych postulatów jest przyjęcie zdarzeń jako podstawowego sposobu nawiązywania jakiekolwiek komunikacji pomiędzy urządzeniami. System sterowany zdarzeniami (ang. *Event driven*) poprzez konfigurację odbywającą się na zasadzie przypisywania reakcji na zdarzenia zachodzące pomiędzy komponentami systemu, pozwala na zachowanie pełnej elastyczności i uniwersalności względem dowolnych reguł pracy. Jednocześnie, architektura event driven narzuca konieczność nieustannej oraz natychmiastowej zmiany kontekstów programów (dane w pamięci, kolejka rozkazów) wykonywanych na poszczególnych urządzeniach. Zadanie to zrealizowane może

[7] Oficjalna strona otwartej elektronicznej platformy do prototypowania Arduino http://www.arduino.cc/, 2014

być przy wykorzystaniu systemu operacyjnego przeznaczonego dla urządzeń wbudowanych, który umożliwiałby zarządzanie kontekstem i przetwarzaniem w sposób transparentny dla twórcy poszczególnych komponentów systemu. Systemy operacyjne tego typu często są tworzone na wiele platform sprzętowych jednocześnie, realizując tym samym postawiony wcześniej wymóg uniwersalności w ujęciu dualistycznym.

W poniższych paragrafach zostaną przedstawione rozwiązana mające umożliwić sprostanie postawionym systemowi celom. W pierwszej kolejności przedstawiony zostanie ogólny schemat blokowy reprezentujący przykładową budowę urządzenia. Następnie, zostanie omówiona kwestia obecności systemu operacyjnego oraz jego wyboru. Finalnie, przedstawione zostaną dostępne opcje względem platformy sprzętowej: mikrokontrolera oraz modułu komunikacji bezprzewodowej.

2.4.1 System operacyjny

Zastosowanie systemu operacyjnego pozwala na odseparowanie kodu odpowiedzialnego za realizację funkcji protokołu od platformy sprzętowej, czyniąc kod przenośnym na dowolną platformę wspieraną przez wybrany system operacyjny. Ponadto, większość obecnie oferowanych systemów operacyjnych przeznaczonych dla systemów wbudowanych, posiada zintegrowane mechanizmy ułatwiające proces tworzenia aplikacji, takie jak wątki, semafory, zdarzenia czy sterowniki interfejsów komunikacyjnych. Narzut wynikający z warstwy abstrakcji systemu operacyjnego, zapewniający separację platformy sprzętowej od aplikacji, jest na tyle mały iż można go zaakceptować, szczególnie w zestawieniu korzyściami jakie oferuje.

Na rynku dostępnych jest ponad setka darmowych i otwartych systemów operacyjnych przeznaczonych na systemu wbudowane. Wybór konkretnego podlegał przede wszystkim wymogowi wsparcia jak największej liczby architektur, w tym Atmel AVR[8], ARM[9], MIPS[10], PIC[11] oraz Intel 8051[12]. Nie mniej jednak ważnym kryterium był aspekt oferowanego przez twórców wsparcia i dokumentacji systemu oraz jego ciągłego rozwoju i ulepszania. Z systemów spełniających wszystkie te kryteria, warto wymienić (w kolejności alfabetycznej):

BeRTOS system operacyjny czasu rzeczywistego, zapewniający niezbędne sterowniki modułów obsługiwanych mikrokontrolerów oraz biblioteki, m.in. wspierające kompletny interfejs graficzny. System posiada w pełni statyczny model pamięci, pomocny szczególnie w przypadku procesorów o ograniczonych zasobach pamięci. Dostępny jest również emulator systemu dla środowisk Unixowych
Strona internetowa: *http://www.bertos.org/*

ChibiOS/RT system operacyjny czasu rzeczywistego dla systemów wbudowanych, którego

[8] AVR – rodzina mikrokontrolerów 8, 16 i 32 bitowych produkcji firmy Atmel
[9] ARM (ang. *Advanced RISC Machine*) – głównie 32 bitowa architektura mikroprocesorów typu RISC (ang. *Reduced Instruction Set Computer*)
[10] MIPS (ang. *Microprocessor without Interlocked Piped Stages*) – architektura typu RISC, w wersji 32 i 64 bitowej, rozwijana przez firmę MIPS Technologies
[11] PIC (ang. *Peripheral Interface Controller*) – rodzina 8, 16, i 32 bitowych mikrokontrolerów typu RISC produkowana przez firmę Microchip Technology
[12] Intel 8051 – 8 bitowy mikrokontroler zbudowany w architekturze CISC (ang. *Complex Instruction Set Computer*

architektura została zoptymalizowana dla szybkich zmian kontekstu przetwarzania. Oferuje wszystkie mechanizmy wymagane od systemu operacyjnego, zapewniając ponadto sterowniki dla większości funkcji wspieranych przez niego mikrokontrolerów, moduły testów systemu oraz stos uIP, lwIP i FatFs.
Strona internetowa: *http://www.chibios.org/*

Contiki system operacyjny dedykowany dla Internetu przedmiotów, przeznaczony dla małych urządzeń typu czujniki albo sterowniki, komunikujących się i kontrolowanych przez sieć bezprzewodową. Zawiera wbudowany stos TCP/IPv6 oraz v4, a ponadto wsparcie dla graficznego interfejsu użytkownika.
Strona internetowa: *http://www.contiki-os.org/*

freeRTOS najpowszechniej znany system operacyjny czasu rzeczywistego dla systemów wbudowanych, dostępny na większość możliwych mikrokontrolerów, napisany w C. Jego budowa, poza standardowymi mechanizmami oferowanymi przez system operacyjny jak kolejki, semafory, mechanizmy wzajemnego wykluczania, przerwania, nie posiada przewidzianych elementów takich jak sterowniki, zarządzanie pamięcią czy obsługa sieci. Dostępne jest jednak sporo opracowań dla systemu, dokumentacji i pomocy.
Strona internetowa: *http://www.freertos.org/*

Nut/OS mały system operacyjny zaprojektowany głównie dla obsługi stosu TCP/IP oraz związanych z nim mechanizmów (DHCP[13], ARP[14], DNS[15], Ethernet). Posiada ponadto wbudowany interpreter języka skryptowego Lua[16]
Strona internetowa: *http://www.ethernut.de/en/software/*

QP framework dla aplikacji sterowanych zdarzeniami, który może również pełnić rolę systemu operacyjnego. Oferuje maksymalnie 64 obiekty sterowane zdarzeniami, które tworzyć można za pomocą oferowanych za darmo narzędzi, na zasadzie diagramów UML[17]. Największa jego zaletą jest możliwość wykorzystania tego samego kodu w formie systemu operacyjnego systemu wbudowanego jak i aplikacji dla zwykłego systemu typu Linux czy Windows.
Strona internetowa: *http://www.state-machine.com/qp/*

TinyOS system operacyjny przeznaczony do bezprzewodowych sieci czujników, napisany w języku nesC. Zbudowany jest w oparciu o jeden stos, unikając w ten sposób problemu blokowania. Architektura oparta jest o paradygmat *event–driven*, oferujący asynchroniczne zdarzenia oraz przetwarzane rozdzielnie zadania.
Strona internetowa: *http://www.tinyos.net/*

Kwestia wyboru systemu operacyjnego zależy oczywiście w zupełności od wytwórcy

[13] DHCP (ang. *Dynamic Host Configuration Protocol* – protokół pozyskiwania od serwera danych konfiguracyjnych dla sieci

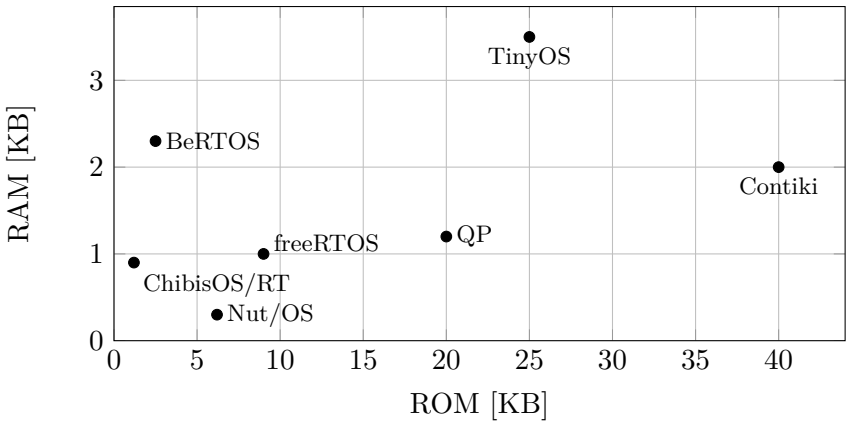
[14] ARP (ang. *Address Resolution Protocol*) protokół sieciowy mapujący adresy warstwy sieciowej na adresy warstwy fizycznej

[15] DNS (ang. *Domain Name System* – protokół komunikacyjny oraz usługa zajmującą się rozproszoną bazą danych adresów sieciowych

[16] Język skryptowy o składni zbliżonej do Pascala, z konstrukcjami opisu danych opartymi na tablicach asocjacyjnych i rozszerzalnej semantyce.

[17] UML (ang. *Unified Modeling Language* – zunifikowany język modelowania, najczęściej graficzny, wykorzystywany do tworzenia modelów systemów, definiując ich struktury, zachowania i interakcje

danego urządzenia oraz możliwości i zasobów wykorzystanego mikrokontrolera. Poszczególne systemy różnią się m.in. w kwestii minimalnego zapotrzebowania na pamięć, co zostało zobrazowane na rysunku 2.9, co należy uwzględnić szczególnie podczas tworzenia rozbudowanych aplikacji.



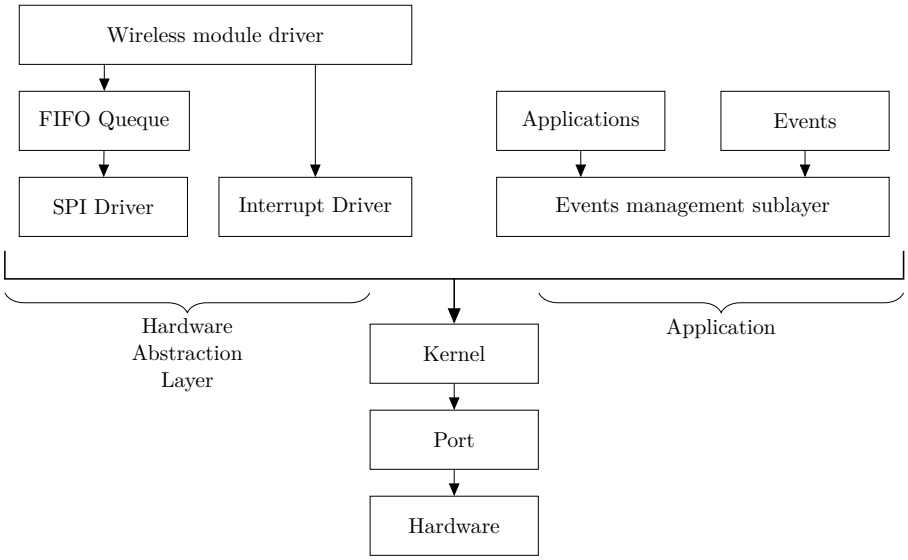
Rysunek 2.9.: Porównanie minimalnej konsumpcji pamięci przez poszczególne systemy operacyjne
Źródło: opracowanie własne na podstawie danych producentów

Implementacja protokołu zrealizowana w ramach niniejszej pracy, oparta została o system ChibiOS/RT z gałęzi 2.6.x, oferujący pełne wsparcie dla wykorzystanych mikrokontrolerów ATmega328p[18] oraz STM32L152[19] w formie sterowników modułów oraz ich konfiguracji. System ten został wybrany również ze względu na solidną dokumentację, małą zajętość pamięci oraz nastawienie na szybką zmianę kontekstów, wymaganą przy systemie sterowanym zdarzeniami. System został stworzony w roku 2006 przez Giovanniego Di Sirio, który zajmuje się przez cały czas jego nieustannym rozwojem, wprowadzając w pierwszym kwartale 2015 roku stabilną wersję z gałęzi 3.x[Sir14].

Korzystając z udostępnionego API systemu ChibiOS/RT, napisany został sterownik HAL (ang. *Hardware Abstraction Layer*) modułu komunikacji bezprzewodowej nRF24L01+ (opisanego w 2.4.2.2) wykorzystujący interfejs komunikacyjny SPI[20], przerwania, kolejki FIFO odczytu i zapisu sterowane przerwaniami pochodzącymi z modułu komunikacji bezprzewodowej oraz mechanizmy wzajemnego wykluczania dla uniknięcia błędów komunikacji z modułem. Realizacja ta została przedstawiona na rysunku 2.10, w sekcji *Hardware Abstract Layer*. Dzięki implementacji wykorzystującej jedynie obiekty stanowiące składniki systemu operacyjnego, przenoszenie sterownika pomiędzy poszczególnymi mikrokontrolerami nie wymaga żadnych zmian w kodzie.

Równolegle, jako opcjonalny składnik jądra systemu, zrealizowana została implementacja Podwarstwy zarządzania zdarzeniami, udostępniająca interfejsy automatycznego rejestrowania nowych zdarzeń na danym urządzeniu i dodawania ich do tablicy zdarzeń,

[18] Dokumentacja techniczna mikrokontrolera Atmel ATmega328p http://www.atmel.com/Images/doc8161.pdf (dostęp 30.08.2014 r.)
[19] Dokumentacja techniczna mikrokontrolera STM32L152RBT6 http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00277537.pdf (dostęp 30.08.2014 r.)
[20] SPI (ang. *Serial Peripheral Interface*) – szeregowy interfejs urządzeń peryferyjnych, wykorzystujący do komunikacji 3 linie.



Rysunek 2.10.: Schemat blokowy implementacji protokołu w systemie ChibiOS/RT 2.6.x
Źródło: opracowanie własne w oparciu o dokumentację systemu [Sir14]

modyfikacji dowiązanych zdarzeń oraz ich wyzwalania. Zadania te zostały wdrożone w oparciu o mechanizm skrzynek pocztowych (Mailboxes) oraz zdarzeń(Events) oferowany przez system ChibiOS/RT. Skrzynki pocztowe stanowią interfejs asynchronicznych kolejek wiadomości, umożliwiających pozostawienie oczekującemu na nie wątkowi wskaźników do określonych struktur danych, wykorzystywanych jako parametry wyzwolenia zdarzenia. Zdarzenia w kontekście protokołu, zostały zrealizowane w oparciu o wątki przetwarzające nieskończoną pętlę, na której samym początku wywoływana jest funkcja API systemu zawieszają dany wątek w oczekiwaniu na zdarzenie systemowe. W momencie otrzymania przez interfejs Podwarstwy zarządzania zdarzeniami identyfikatora zdarzenia, po jego odnalezieniu w tablicy zdarzeń, wyzwalane jest przypisane do niego zdarzenie systemowe, wyzwalające z kolei funkcję obsługi zdarzenia protokołu. Parametry przekazywane podczas wyzwolenia zdarzenia oraz zdarzeń zwrotnych zostały zaimplementowane w formie tablic wskaźników, zapewniając wymagany dostęp indeksowy oraz dowolność w rodzaju przechowywanych danych.

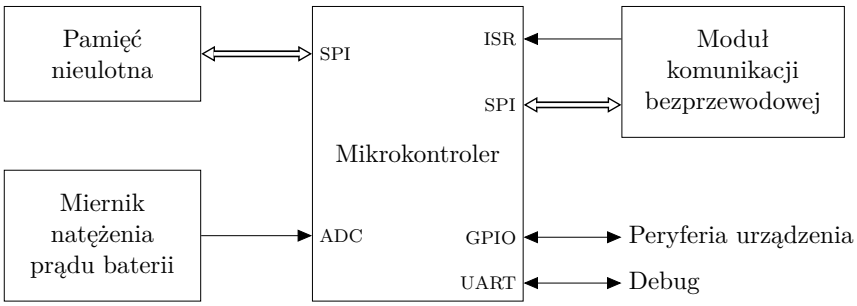
Zastosowanie osobnych wątków dla każdego zdarzenia protokołu, pozwoliło na uniknięcie zatrzymania działania systemu w przypadku awarii przetwarzania w danym wątku. Potencjalnie groźne pozostały wycieki pamięci, wykraczające poza przydzielony wątkowi stos pamięci i powodujące zawieszenie całego systemu. Problem ten jednak rozwiązuje zastosowanie zalecanej architektury ChibiOS/RT, wykorzystującej statyczne użycie pamięci, zarządzane przez usługi obsługi pamięci udostępniane przez jądro systemu.

System operacyjny jest wygodnym usprawnieniem w procesie tworzenia nowych urządzeń, pozwalającym uniknąć ponownego pisania tego samego kodu dla innego mikrokontrolera. W przypadku jednak niewielkich mikrokontrolerów lub prostych urządzeń, wykorzystanie systemu operacyjnego o ile jest możliwe, może stanowić pewną przesadę. Przykładem takiej sytuacji jest zrealizowany w ramach pracy prosty pilot zdalnego sterownia, którego oprogramowanie zostało zbudowane na zasadzie skończonej maszyny stanów, co jest rozwiązaniem całkowicie akceptowalnym. Ponadto, ze względu na mobilność tego

urządzenia i małej energetycznej wydajności algorytmu routingu DSR, pominięte zostały w tym urządzeniu zdarzenia odpowiedzialne za routing (*Route*, *RouteRequest*,*RouteResponse* oraz *RouteError*), zastąpione propagacją wiadomości poprzez zalewanie. Rozwiązanie takie jest wysoce nierekomendowane, wykorzystane zostało jedynie dla uproszczenia konstrukcji oraz zbadania jednego z wielu wariantów implementacji.

2.4.2 Platforma sprzętowa

Obecność systemu operacyjnego pozwala na wybór dowolnej platformy sprzętowej przez niego wspieranej, z możliwością jej zmiany na inną w zależności o potrzeb. Dobór modułów użytych do urządzeń zbudowanych w ramach pracy realizowany był na podstawie czasu, łatwości oraz cenowej dostępności układów na polskim rynku, który nie jest niestety zaopatrzony w takim stopniu na rynki zachodnie. W poniższych paragrafach zostały przedstawione układy, których użycie jest zalecane oraz te faktycznie wykorzystane podczas realizacji pracy. Ogólną koncepcję urządzenia przedstawiono na rysunku 2.11, uwzględniając mikrokontroler, moduł komunikacji bezprzewodowej, pamięć nieulotną oraz miernik natężenia prądu baterii.



Rysunek 2.11.: Schemat blokowy implementacji protokołu w systemie ChibiOS/RT 2.6.x
Źródło: opracowanie własne w oparciu o dokumentację systemu [Sir14]

Pamięć nieulotna, bez znaczenia w jakiej postaci, wymagana jest dla zapisu tablicy zdarzeń wraz z ich dowiązaniami, pamięci podręcznej tras oraz ogólnej konfiguracji urządzenia. Dane te muszą być przechowywane przez urządzenia w przypadku utraty zasilania, aby po jego przywróceniu możliwa była kontynuacja pracy. Niektóre mikrokontrolery oferują zapis na pamięci wbudowanej, jednak jej rozmiar w większości przypadków jest niewystarczający.

Miernik natężenie prądu w baterii jest elementem nieobligatoryjnym, aczkolwiek przydatnym dla bardziej świadomego zarządzania energią przez urządzenie oraz raportowania nadzorcy systemu ewentualnych problemów z nią związanych, co pozwala użytkownikowi na podjęcie odpowiednich akcji. Wiele modułów komunikacji bezprzewodowej zapewnia taką funkcjonalność, czyniąc osobny moduł pomiarowy zbędnym. Większość układów zmierzone wartości zwraca w formie proporcjonalnego napięcia w zakresie 0V–V_{dd}, związku z czym ważne jest podczas wyboru mikrokontrolera zwrócenie uwagi na obecność portu i układu przetwornika analogowo–cyfrowego ADC (ang. *Analog to Digital Converter*.

2.4.2.1 Mikrokontroler

System ChibiOS/RT oficjalnie wspiera większość powszechnie używanych mikrokontrolerów, reprezentujących różne architektury, co zostało przestawione w tabeli 2.4 oraz na oficjalniej stronie systemu[21]. Każda z architektur oraz jej przedstawicieli posiada cechy charakterystyczne stanowiące ich wady lub zalety, trudno więc wskazać jeden uniwersalny układ, odpowiedni do realizacji każdego urządzenia.

Tablica 2.4.: Architektury i odpowiadające im mikrokontrolery oficjalnie wspierane przez system ChibiOS/RT

Architektura	**Mikrokontrolery**
ARM Cortex–M0	LPC11xx, LPC11Uxx, STM32F0xx
ARM Cortex–M3	LPC13xx, STM32F1xx, STM32F2xx, STM32L1xx
ARM Cortex–M4	STM32F3xx, STM32F4xx
ARM7	AT91SAM7x, LPC214x
MegaAVR	ATmega128, AT90CAN128, ATmega328p, ATmega1280
MSP430	MSP430F1611
Power Architecture e200z	SPC56x (wszystkie)
STM8	STM8L, STM8S

Podczas wyboru mikrokontrolera, na którym oparte ma zostać budowane urządzenie, kierować się należy nie tylko jego zasobami pamięci czy prędkością zegara taktującego. Czynnikami czasem nawet ważniejszymi jest dostępność odpowiednich portów komunikacyjnych w wystarczającej liczbie, niski pobór energii, cena czy obsługa zewnętrznych przerwań, które pozwalają na wybudzanie uśpionego mikroprocesora jedynie w sytuacjach tego wymagających. Ważna jest również dostępność narzędzi przeznaczonych do prototypowania oraz debugowania tworzonych urządzeń, niezbędnych dla zapewnienia użytkownikowi produktu dobrej jakości.

W ramach realizacji pracy wykorzystane zostały jedne z najbardziej popularnych i najłatwiej dostępnych na rynku układów:

- ATmega328p, zbudowany wg architektury MegaAVR, działający na platformie Arduino Uno
- STM32L152, zbudowany wg architektury ARM Cortex–M3, działający na platformie STM32–Discovery Board

Oba wybrane układu osadzone były na platformach służących do prototypowania, co ułatwiło oraz przyspieszyło tworzenie wersji eksperymentalnych urządzeń. Warto zaznaczyć, iż układ STM32L152 wyróżnia się pozytywnie w zestawieniu z innymi mikokonrolerami, charakteryzując się wysoką oszczędnością energetyczną, stosunkowo dużymi zasobami, dostępnością wielu interfejsów komunikacyjnych oraz dość niską ceną.

[21] Architektury oficjalnie wspierane przez system ChibiOS/RT http://www.chibios.org/dokuwiki/doku.php?id=chibios:archit (dostęp 30.08.2014)

2.4.2.2 Moduł komunikacji bezprzewodowej

Podczas wyboru modułu komunikacji bezprzewodowej, poza parametrami jak dostępne częstotliwości transmisji czy interfejs komunikacyjny, zwrócić uwagę należało na format ramki wbudowany w moduł przez producenta. Formaty te często są w ograniczonym stopniu konfigurowalne, przez co może okazać się niemożliwym wykorzystanie modułu dla zdefiniowanego standardu komunikacji.

Biorąc pod uwagę różnorodność dostępnych na rynku modułów komunikacji bezprzewodowej[22], należało ustalić kryteria wstępne ograniczające liczbę potencjalnych modułów:

- komunikacja w zakresie częstotliwości 868–868.6 MHz (zgodnie ze standardem)
- modulacja fali nośnej MSK (zgodnie ze standardem)
- dostępna przepływność 250 kbps (zgodnie ze standardem)
- brak wbudowanego protokołu komunikacyjnego typu ZigBee czy Z–Wave
- zasilanie napięciem w zakresie 2.3–3.6 V, jako najczęstsze zasilanie energooszczędnych mikrokontrolerów
- pobór prądu w trybie odbioru poniżej 50 mA, dla większej energooszczędności
- pobór prądu w trybie transmisji poniżej 80 mA, dla większej energooszczędności
- zasięg w pomieszczeniach większy niż 10 m, umożliwiający komunikację nawet w rzadkich sieciach[ITU97]

Na podstawie wymienionych kryteriów, wybrana została grupa modułów komunikacji bezprzewodowej ze sklepu DigiKey.com[23], z której zostały z kolei odfiltrowane moduły różniące się jedynie typem obudowy. Rezultat został przedstawiony na rysunku 2.12 w zależności zasięgu komunikacji do poboru prądu w trybie odbioru. Zależność ta pozwala na łatwe wskazanie urządzeń charakteryzujących się niskim poborem mocy w trybie odbioru, który stanowi około 90% czasu życia urządzenia oraz dobrym zasięgiem, umożliwiającym komunikację w sieciach o niewielkiej gęstości i dużym rozproszeniu na obszarze pomieszczenia.

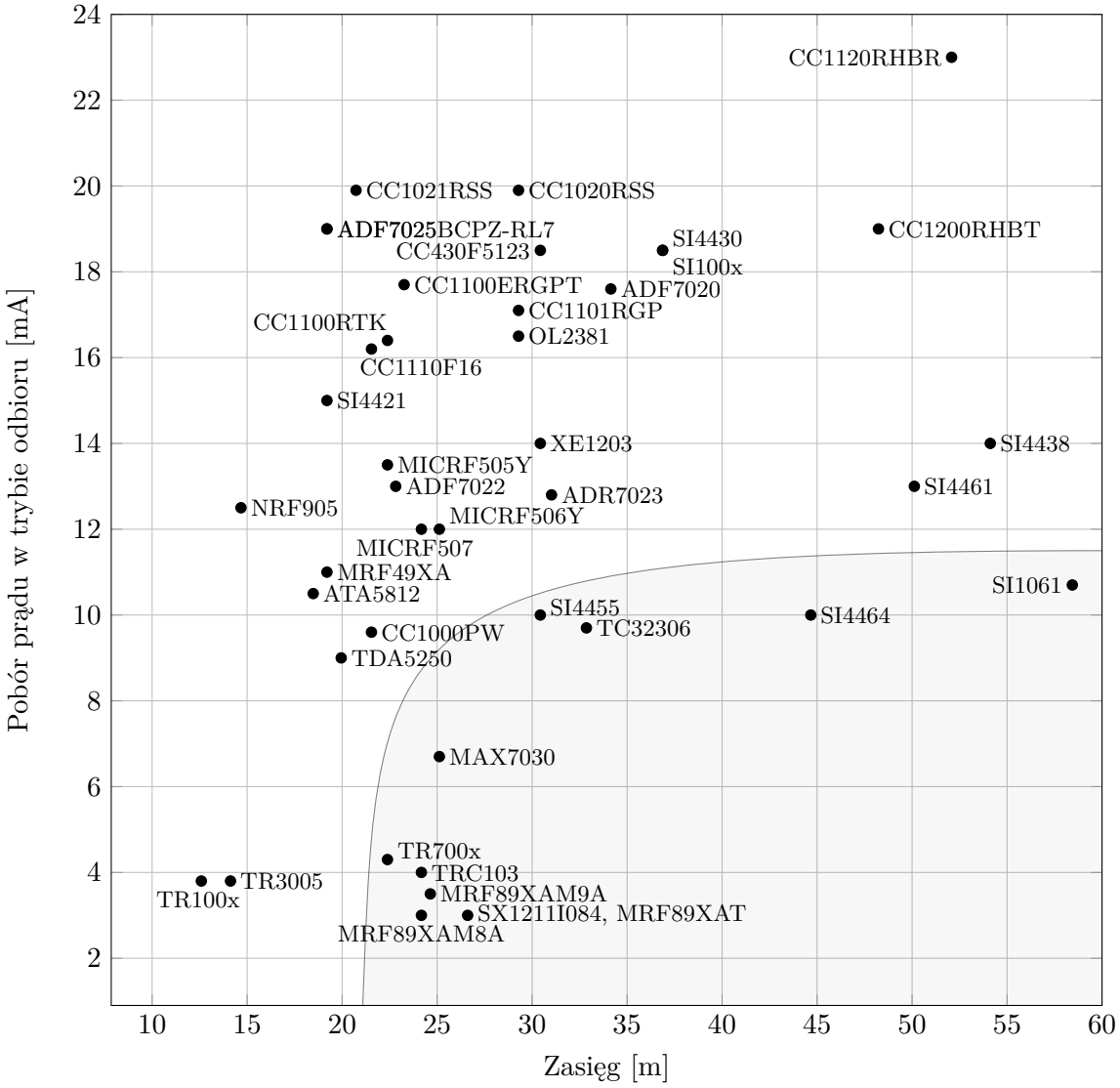
Analizując uzyskany wykres, można dostrzec grupę urządzeń (szary obszar na rysunku 2.12) o korzystnym stosunku zasięgu do poboru prądu, odbiegającym od ogólnej tendencji pozostałych urządzeń. Z tego podzbioru, na podstawie dostarczonej przez producentów dokumentacji, wybrane zostały trzy moduły pozwalające na implementację zdefiniowanego w ramach pracy protokołu komunikacji bezprzewodowej.

SX1211 [24] wyprodukowany przez Semitech, jeden z najbardziej energooszczędny z analizowanych modułów (3 mA w trybie odbioru). Posiadając surową ramkę wiadomości (brak wbudowanego protokołu) pozostawia możliwość i miejsce na realizację protokołu. Oferuje filtrowanie wiadomości, przerwania dla wewnętrznych zdarzeń, automatyczne obliczanie sumy kontrolnej oraz pomiar siły sygnału.

[22] W sklepie internetowym DigiKey.com, stosowanym jako punkt odniesienie, znajduje się 3853 modułów komunikacji radiowej (stan z dnia 30.08.2014)

[23] Strona główna dystrybutora podzespołów elektronicznych DigiKey.com http://www.digikey.com/ (dostęp 30.08.2014)

[24] Dokumentacja techniczna http://www.semtech.com/images/datasheet/sx1211.pdf (dostęp 30.08.2014)



Rysunek 2.12.: Porównanie modułów komunikacji bezprzewodowej na podstawie kryterium zasięgu oraz poboru prądu w trybie odbioru
Źródło: opracowanie własne na podstawie sklepu internetowego Digi-Key.com

Si4464 [25] wyprodukowany przez Silicon Labs, układ posiadający największy zasięg spośród badanych modułów (44 m). Wymaga bardzo ograniczoną ilość zewnętrznych podzespołów (wbudowany oscylator), ułatwiając kompozycję płytki oraz oszczędzając na niej miejsce. Podobnie jak pozostałe moduły, zapewnia przerwania dla zdarzeń wewnętrznych, surowy format ramki danych, miernik poziomu baterii oraz czujnik temperatury.

MRF89XA [26] wyprodukowany przez Microchip, zapewnia energooszczędność na poziomie

[25] Dokumentacja techniczna http://www.silabs.com/Support20Documents/TechnicalDocs/Si4464-63-61-60.pdf (dostęp 30.08.2014)
[26] Dokumentacja techniczna http://ww1.microchip.com/downloads/en/DeviceDoc/70622C.pdf (dostęp 30.08.2014)

modułu SX1211 (3 mA w trybie odbioru). Posiada ponadto wbudowany oscylator, filtrowanie wiadomości, przerwania. Wbudowana w moduł ramka wiadomości pozwala na dostosowanie jest do protokołu poprzez przekazywanie w słowie synchronizacji następujących wartości: długość wiadomości, adres docelowy, adres nadawcy. Należy również zrezygnować z opcjonalnego pola długości wiadomości zdefiniowanego przez wbudowany format ramki.

Istnieją oczywiście inne moduły, które mogą umożliwiać implementację protokołu, mimo iż nie spełniły one wszystkich postawionych tutaj kryteriów. Analiza jednak wszystkich możliwości nie jest głównym przedmiotem niniejszej pracy, ostateczny wybór pozostawiony jest więc ewentualnym wytwórcom urządzeń.

Eksperymentalne egzemplarze urządzeń zbudowane w ramach realizacji pracy, oparte zostały o moduł komunikacji bezprzewodowej nRF24L01+[27], wytwarzany przez Nordic Semicondutor. Wykorzystuje ondo komunikacji pasmo 2.4 GHz, posiada ponadto wbudowany własny niskopoziomowy protokół komunikacyjny, definiujący niekonfigurowalną ramkę wiadomości. Moduł ten jest całkowicie niezgodny ze zdefiniowanym protokołem i niezalecany nawet to prototypowania. Wybrany został ze względu na łatwość dostępu na polskim rynku, stosunkowo niską cenę oraz dystrybucje w postaci gotowej płytki drukowanej z wyprowadzeniami dla płytki uniwersalnej. Praca z tym modułem pozwoliła doświadczalnie udowodnić słuszność rezygnacji z wykorzystania pasma 2.4 GHz (2.2 Warstwa fizyczna), jako pełnego zakłóceń oraz szumów pochodzących od urządzeń typu routery Wi–Fi bądź sprzęt Bluetooth (sprawność komunikacji wynosiła około 60% przy dystansie 1m).

Parę słów podsumowania rozdziału?

[27] Dokumentacja techniczna modułu nRF24L01+ dostępna na stronie producenta http://www.nordicsemi.com/eng/content/download/2726/34069/file/nRF24L01P_Product_Specification_1_0.pdf (dostęp 30.08.2014)

Rozdział 3

Nadzorca systemu

Nadzorca systemu stanowi równorzędne urządzenie w sieci, nie posiadające z punktu widzenia protokołu żadnych dodatkowych praw ani przywilejów w stosunku do innych urządzeń. Nie jest też elementem niezbędnym dla wykorzystania standardu komunikacji w formie systemu automatyki domowej, stanowiąc opcjonalny dodatek pozwalający na zarządzanie systemem zdalnie. Jego obecność jednak pozwala na realizacje pewnych zadań, usprawniających działanie sieci i poprawiających doświadczenia użytkownika z nią związanych.

3.1 Zadania nadzorcy systemu

Nadzorca systemu, jako urządzenie w założeniu posiadające większą moc obliczeniową niż pozostałe urządzenia sieci oraz dostęp do stałego źródło zasilania, umożliwia wdrożenie funkcjonalności, które mogłby stanowić nadmierne obciążenie dla większości urządzeń w sieci. W poniższych paragrafach zostały przedstawione zadania, które powinny być realizowane przez urządzenie nadzorcy systemu, poza standardowym uczestnictwem w działaniu protokołu komunikacyjnego.

3.1.1 Balansowanie ruchu w sieci

Nadzorca systemu, poza standardowym uczestnictwem w mechanizmach protokołu komunikacyjnego, powinien realizować zadania o większym stopniu skomplikowania i konsumpcji zasobów, które przyczyniałyby się do poprawy funkcjonowania sieci bezprzewodowej opartej o stworzony standard komunikacji.

Warunkiem niezbędnym do wdrożenia takiej funkcjonalności, jest pełna świadomość topologii całej sieci, z uwzględnieniem informacji o każdym urządzeniu (element opisany w 2.3.3.3) oraz jego aktualnym statusie (element opisany w 2.3.3.1). Topologia powinna być aktualizowana na bieżąco, w reakcji na pojawienie się w sieci zdarzenia *Route*, *RouteRequest*, *RouteReply*, *RouteError*, a w szczególności *PanicRoute*. Korzystać można również mechanizmy kopii zapasowych pamięci podręcznej urządzeń, opisanych w 3.1.4, na podstawie których można uzupełniać informacje o aktualnie wykorzystywanej w przez sieć topologii.

Bazując na jak najpełniejszej bazie danych dotyczących obsługiwanej sieci, nadzorca systemu powinien cyklicznie bądź w reakcji na pojawienie się dużej liczny błędów transmisji, dokonywać zdalnie zmian w routingu poszczególnych urządzeń. Działanie takie ma na celu odpowiednie zbalansowanie i rozłożenie równomiernie ruchu sieciowego, uniknięcie przeciążeń i zredukowanie błędów transmisji. Ponadto, zastosowanie takich mechanizmów pozwolić ma na wydłużenie żywotności urządzeń zasilanych z baterii, poprzez ich odcią-

żenie i przesunięcie większości ruchu sieciowego wynikającego z routingu na urządzenia o stałym źródle zasilania, a nawet w przypadku bardziej wyrafinowanych algorytmów, eliminacja urządzeń o niskim stanie baterii z całego ruchu sieciowego.

Co więcej, posługując się informacjami o relacjach dotyczących poszczególnych urządzeń i zdarzeń na nich powiązanych oraz wynikającymi z tego statystykami komunikacji, możliwe jest dopasowanie ruchu sieciowego w jeszcze większym stopniu do wymogów mu faktycznie stawianych.

Realizacja tego zadania nie jest oczywiście zagadnieniem prostym w wykonaniu ze względu na mobilny charakter sieci, podlegającej niekiedy częstym modyfikacją topologii, tymczasowymi problemami z łącznością oraz innymi czynnikami losowymi wpływającymi na unieważnienie przyjętego routingu. Pomocne może być zastosowanie mechanizmów zapożyczonych z algorytmów QoS (ang. *Quality of Service*) dedykowanych dla mobilnych sieci Ad Hoc, jak na przykład wybór trasy o większej ilości przeskoków o mniejszych dystansach połączony ze zmniejszeniem mocy sygnału lub wykorzystanie szczelin czasowych, wymagające już jednak ingerencji w protokół[LL99].

3.1.2 Zarządzenie działaniem systemu

Nadzorca, ze względu na pełnioną punkcję interfejsu dostępowego użytkownika do sieci opartej na protokole, musi zgodnie z dyspozycjami użytkownika wpływać na konfigurację poszczególnych urządzeń oraz w konsekwencji, działanie całego systemu.

Podstawowym zestawem opcji urządzenia podlegającym zleconej konfiguracji, jest wynikające z definicji stworzonego protokołu wiązanie zdarzeń, które zachodzi najczęściej pomiędzy różnymi urządzeniami w sieci. Zlecenie wiązania zdarzeń powinno być wydawane za pośrednictwem ogólnodostępnego po przejściu procesu weryfikacji użytkownika, interfejsu konfiguracyjnego opisanego w 3.1.3.

Nadzorca systemu, po odebraniu zlecenia danej konfiguracji wiązań, powinien w pierwszej kolejności dokonać translacji reprezentacji przedstawionej użytkownikowi, nie uwzględniającej np. potrzeby przesłania wyzwalacza zdarzenia przez sieć, na postać uwzględniającą detale konfiguracji. Ten wewnętrzny format musi zawierać m.in. informację o parametrach wywołania (ustalenie indeksów i atrybutów stałych, opisanych w 2.3.1.1), parametrach zwrotnych zdarzenia oraz wymaganej transmisji przez sieć (wyzwolenie zdarzenia *Route*). Następnie nadzorca musi uzyskaną konfigurację rozpropagować wśród wszystkich uwzględnionych w niej urządzeń sieci, dokonując tej czynności w sposób oszczędny energetycznie. Nową konfigurację zdarzeń można zaaplikować na przykład poprzez zlecenie usunięcie wszystkich dowiązań w danym urządzeniu, a następnie wysłanie w całości nowej konfiguracji. Rozwiązanie to jednak przyczynia się do zbędnych strat energii oraz czasu, spowodowanych transmisją dużych pakietów danych. Zalecane jest, aby nadzorca porównał zleconą nową konfigurację urządzeń z konfiguracją aktualnie na nich zapisaną (kopia zapasowa konfiguracji, opisana w 3.1.4), a następnie przekazanł do urządzeń jedynie wiązania różnice w stosunku do pierwotnej konfiguracji.

Niezwykle istotnym zdaniem nadzorcy, jest również sprawdzenie nowej konfiguracji pod kątem wiązań mogących naruszyć stabilność sieci, jak np. zapętlenie wyzwoleń zdarzeń czy kaskada wyzwoleń, która nie nigdy nie może zostać zainicjowana poprzez błędnie sformułowane wyzwolenie warunkowe (przewidziane w przyszłych wersjach standardu).

W przypadku wykrycia wadliwych ustawień, nadzorca powinien natychmiast za pośrednictwem interfejsu konfiguracji poinformować użytkownika o zaistniałym problemie i w żadnych wypadku nie powinien rozsyłać nowej konfiguracji do urządzeń.

Nadzorca systemu musi również agregować informacje o stanie wszystkich urządzeń w systemie, dla ich późniejszej prezentacji użytkownikowi oraz dla celów statystycznych, jak np. algorytmy predykcyjne. Obligatoryjnym jest więc dowiązanie zdarzenia przekazania wiadomości do nadzorcy (*Route*) do zdarzenia *StatusChanged* na każdym urządzeniu przynależącym do sieci. Dzięki takiemu rozwiązaniu, natychmiast po zmianie któregokolwiek z parametrów urządzenia w sieci, nadzorca będzie mógł tę informację przekazać użytkownikowi i podjąć odpowiednie dla sytuacji akcje, jak np. odciążanie urządzenia o niskim stanie baterii z routingu. Jednocześnie, nadzorca musi znać semantykę poszczególnych struktur zawierających status urządzeń. Jeśli więc urządzenie nie przekazuje danych statusu w formie samoopisującego się XML, wytwórca urządzenia musi udostępnić opis formatu nadzorcy, który z kolei musi posiadać mechanizmy pozwalające na identyfikację przekazanych danych i ich późniejszą prezentację użytkownikowi.

Kolejnym aspektem podlegającym zadaniu zarządzeniem konfiguracji systemu jest przeprowadzanie aktualizacji zarówno własnego oprogramowania, jak i oprogramowania pozostałych urządzeń należących do sieci. Realizacji tego zadania jest niezbędna dla zapewnienia użytkownikowi dostępu do najnowszych funkcjonalności oferowanych przez standard komunikacji bezprzewodowej oraz dla łatania błędów pojawiających się w poprzednich wydaniach protokołu lub oprogramowania urządzeń. Aktualizacja powinna być realizowana z wykorzystaniem jedynie zaufanych repozytoriów oprogramowania, unikając w ten sposób narażenia użytkowników na wyciek ich prywatnych danych lub zdalny atak na dom.

3.1.3 Udostępnianie interfejsu konfiguracji

Jednym z podstawowych zadaniach realizowanych przez nadzorcę systemu, jest udostępnianie interfejsu umożliwiającego dokonywanie zmian w konfiguracji systemu czy poszczególnych urządzeń oraz pośredniczenie pomiędzy tym interfejsem a całą siecią. Należy zwrócić uwagę, iż nie jest postawiony wymóg, aby udostępniony interfejs był prezentowany w formie graficznej. Jako interfejs konfiguracji rozumiany może być specjalny kanał komunikacyjny, pozwalający na wydawanie komend tekstowych czy API udostępniane stronom internetowym oraz urządzeniom mobilnym posiadającym dedykowane aplikacje do obsługi systemu. Proponowany sposób realizacji takiego kanału konfiguracji, przedstawiony został w opisanej przykładowej implementacji, w 3.2.

Niezależnie od oferowanej przez nadzorcę metody dostępu do interfejsu konfiguracji, powinien on pozwalać na realizację określonego zestawu operacji i komend, które umożliwiają realizację dowolnych celów za ich pośrednictwem.

Przede wszystkim, oferowany interfejs powinien umożliwiać szczegółową prezentację struktury sieci opartej na protokole, przekazując jak najpełniejszą informację o urządzeniach. Dostępna powinna być lista zdarzeń udostępnianych przez aplikacje urządzenia, wraz z dokładnym opisem poszczególnych parametrów wywołania oraz listą wszystkich dowiązanych zdarzeń w formie wywołań zwrotnych, również zaopatrzonych w dokładny opis parametrów wywołania. Opisy te, powinny być dostarczane przez wytwórców urządzeń w formie samoopisujących się plików XML lub JSON, umożliwiających automatycznie

parsowanie oraz prezentację za pośrednictwem interfejsu konfiguracji.

Co więcej, wybrany kanał dostępu do ustawień sieci powinien pozwalać na pobranie stanu wszystkich urządzeń wraz z ich opisem, podobnie jak to miało miejsce w przypadku zdarzeń. Informacje o statusie poszczególnych urządzeń powinny być aktualizowane na bieżąco, bez potrzeby ponownego otwierania interfejsu komunikacyjnego lub podawania dedykowanych komend. O ile jednak komenda wymuszenia przekazania statusu urządzenia jest jak najbardziej pożądana w implementacji, to sama informacja o zmianie statusu powinna być przekazywana automatycznie we wszystkich otwartych interfejsach konfiguracji, z jednoznacznym wskazaniem na urządzenie zmieniające status oraz jego nowe wartości.

Kolejną obligatoryjnie wymaganą od interfejsu konfiguracji funkcją jest możliwość zlecenia konfiguracji sieci oraz poszczególnych jej urządzeń, dzięki wykorzystaniu mechanizmów opisanych w 3.1.2. Interfejs zlecający konfigurację powinien być w stanie przekazać aktualny stan sieci oraz wiązań pomiędzy zdarzeniami urządzeń, pozwalać na tworzenie nowych wiązań z pełnym zestawem parametrów definiujących wiązanie oraz usuwać wiązania istniejące. Ponadto, interfejs powinien przekazywać informację o pomyślnym bądź błędnym zakończeniu proces propagacji nowej konfiguracji wśród urządzeń należących do sieci, ze wskazaniem na powód lub miejsce pojawienia się błędu, np. wskazanie łańcucha zdarzeń tworzącego pętlę.

Interfejs konfiguracji powinien umożliwiać również przekazanie wyzwalaczy zdarzeń do dowolnie wybranego urządzenia w sieci, pozwalając tym samym manualne generowanie akcji w sieci oraz badanie jej zachowań. Udostępnienie takiej funkcji pozwala również na manualną konfigurację urządzeń, w przypadku jeśli np. inny oferowany, interfejs graficzny nie pozwala na zrealizowanie wymaganego działania. Jednocześnie, opcja taka pozwala na całkowitą dowolność w zaopatrywaniu interfejsów graficznych w różne jednorazowo wywoływane funkcje, jak przykładowo przycisk *Wyłącz wszystkie urządzenia*, powodujący rozgłoszenie przez zalewanie zdarzenia *TurnOff*, wyłączającego wszystkie urządzenia dane zdarzenie obsługujące. Dzięki temu, nie ma potrzeby wiązania zdarzeń pomiędzy urządzeniami dla wykonania jednorazowo jakiejś akcji.

3.1.4 Bezpieczeństwo dostępu oraz konfiguracji

Motywacją dla stworzonego protokołu komunikacyjnego stanowiła w dużym stopniu potrzeba wypełnienia luki w kwestii otwartych standardów komunikacji bezprzewodowej dla systemów automatyki domowej, która miała między innymi zapewniać gwarancję zachowania prywatności poufnych danych użytkownika. Oferowanie stworzonego w ramach pracy protokołu na otwartej licencji rozwiązuje ten problem jedynie w kwestii wiedzy użytkownika o wykorzystaniu jego danych przez system. Dostęp jednak nieautoryzowany do przechowywanych danych przez nadzorcę systemu, interfejsu komunikacyjnego lub przechwycenie przekazywanym nim danych prowadzić może do sytuacji jeszcze bardziej niebezpiecznej w konsekwencjach. Zapewnienie więc bezpieczeństwa interfejsu konfiguracji jest kolejnym z kluczowych zadań realizowanych przez nadzorcę systemu.

Realizacja tego celu może zostać wykonana na wiele różnych sposobów, zaczynając od prostego uwierzytelniania użytkownika, a kończąc na logowaniu hasłem przekazanym przez SMS do szyfrowanego tunelu komunikacyjnego. Sposób zapewnienia bezpieczeństwa komunikacji i dostępu do interfejsu konfiguracji zależy całkowicie od implementacji, stawianej

jej wymogów oraz celów. Niezbędne jest jednak zapewnienie jakiegokolwiek mechanizmu bezpieczeństwa.

Do dziedziny ochrony konfiguracji zaliczyć można również bezpieczeństwo aktualnie wykorzystywanej konfiguracji w sieci. Ze względu na rózne sytuacje losowe, konfiguracja na poszczególnych urządzeniach może ulec zniszczeniu, dekompletacji bądź błędom podczas zapisu. W takich sytuacjach niezbędne jest przechowywanie kopii zapasowej konfiguracji wszystkich urządzeń należących do sieci i jej ewentualne zabezpieczenie poprzez przechowywanie na zewnętrznym serwerze.

3.2 Implementacja

Nadzorca systemu, jako urządzeniu nie zdefiniowane wprost w stworzonym standardzie komunikacji bezprzewodowej, nie posiada jasno określonych wymogów co do konstrukcji czy oferowanych możliwości, poza tymi zdefiniowanymi w protokole dla każdego urządzenia sieci. Implementacja zrealizowana w ramach pracy jest konstrukcją eksperymentalną, dającą pogląd na zalecany sposób budowy nadzorcy systemu oraz oprogramowania na nim działającego.

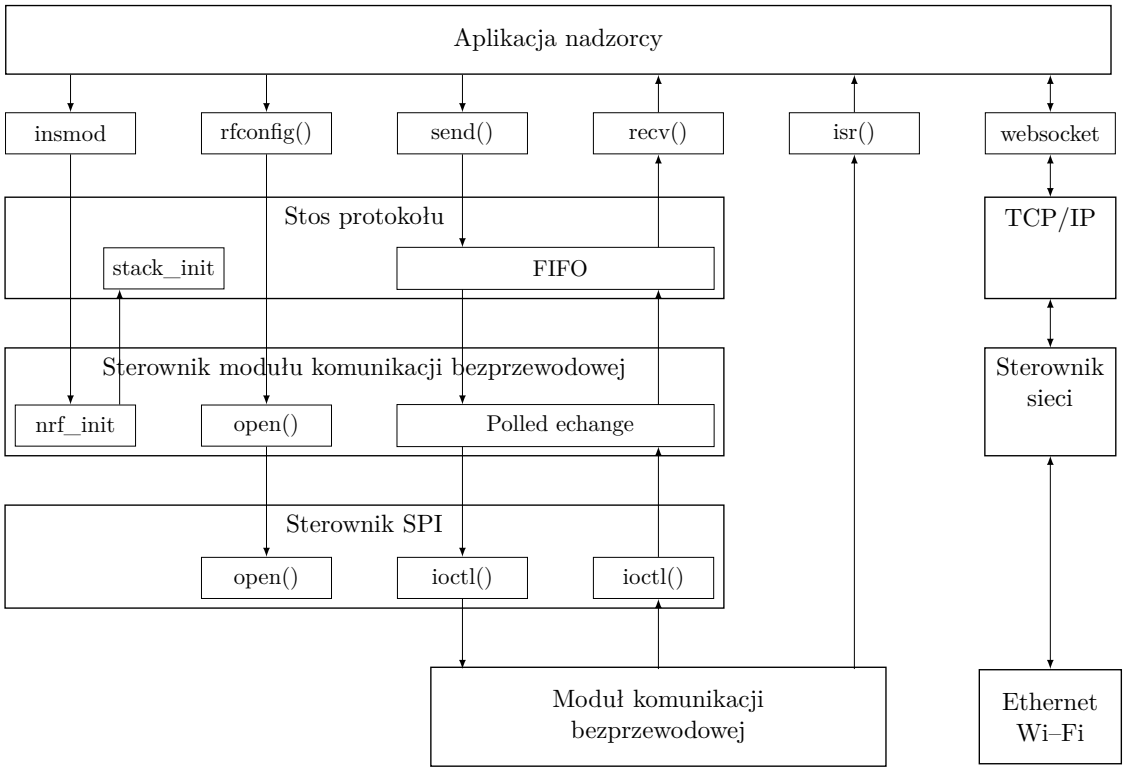
Implementacja sterownika modułu komunikacji bezprzewodowej, stosu zdefiniowanego protokołu oraz aplikacji realizując niektóre z zadań powierzonych nadzorcy została podyktowana i zrealizowana w oparciu o system operacyjny Linux[1]. Wybór tego systemu operacyjnego oparty został jego dobrym wsparciu dla niewielkich platform sprzętowych działających na procesorze typu ARM, otwartości oraz elastyczności w konfiguracji i dostosowywaniu do potrzeb wynikających z jego zastosowania.

Jak zostało zaprezentowane na diagramie blokowym implementacji, przedstawionym na rysunku 3.1, w ramach pracy stworzona została aplikacja wykorzystująca protokół websocket[2], przerwania pochodzące wprost z warstwy sprzętowej oraz stos protokołu komunikacji bezprzewodowej, korzystający z kolei ze sterownika modułu komunikacji bezprzewodowej.

Protokół websocket jest wykorzystywany do udostępniania zdalnego interfejsu konfiguracji, pozwalającego na pobieranie informacji o sieci, jej konfigurowanie oraz sprawdzenie statusu urządzeń, co zostało opisane w 3.1.3. Protokół ten został wybrany ze względu na prostotę jego użytkowania oraz wsparcie na wszystkich nowoczesnych platformach mobilnych oraz jako standard internetowy, we wszystkich nowoczesnych przeglądarkach internetowych. Dzięki temu zbudowanie graficznego interfejsu dostępowego dla użytkownika oraz komunikacja za jego pośrednictwem z nadzorcą była zadaniem o wiele łatwiejszym niż w przypadku wykorzystania np. tradycyjnych socketów TCP/IP, wymagających tła w postaci skryptów PHP.

Stos protokołu komunikacyjnego został zbudowany na zasadzie modułu ładowanego przez system Linux na poziomie jądra, wykorzystującego do komunikacji sterownik modułu komunikacji bezprzewodowej oraz buforowanie wiadomości przez niego przetwarzanych poprzez kolejkę FIFO (ang. *First In First Out*). Sterownik modułu komunikacji bezprzewodowej wykorzystuje do komunikacji z modułem bezprzewodowych wbudowany w system operacyjny sterownik portu SPI, realizujący zapis oraz odczyt danych z urządzenia za pomocą wymiany danych. Sterownik SPI jest przed użyciem konfigurowany,

[1]Strona główna portalu informacyjnego o Linuxie http://www.linux.org/, (dostęp 30.08.2014)
[2]IETF. *RFC6455: The WebSocket Protocol*, 2011



Rysunek 3.1.: Schemat blokowy implementacji nadzorcy w oparciu o system operacyjny Linux
Źródło: opracowanie własne

dla umożliwienia poprawnej komunikacji z modułem, poprzez komendę otwarcia *open()* i przekazane w niej parametry.

Obsługa aplikacji poprzez interfejs konfiguracyjny oparty na protokole websocket, została zrealizowana za pośrednictwem aplikacji webowej udostępniającej graficzny interfejs dostępowy dla użytkownika. Zbudowany został z wykorzystaniem frmeworku Emberjs[3], udostępniającego automatyczne interakcje z protokołem websocket i aktualizację obiektów za jego pośrednictwem. Aplikacja internetowa przechowywana jest lokalnie na urządzeniu nadzorcy i dystrybuowana w sieci poprzez prosty serwer www, zainstalowany na urządzeniu.

Całość została oparta na platformie sprzętowej Raspberry Pi[4] oferującej 24 złączą ogólnego przeznaczenia (w tym SPI), jednordzeniowy procesor typu ARM oraz 512 MB pamięci RAM. Wybór podyktowany był niską ceną urządzenia oraz dostępnością obszernego zbioru materiałów pomagających rozpoczęcie pracy na nim. Platforma ta jednak nie jest zalecana dla przyszłych, bardziej profesjonalnych implementacji, ze względu na jej fatalne wyniki energooszczędności, małą liczbę złączy oraz niezbyt wydajny procesor.

[3]Oficjalna strona frameworku Emberjs http://www.emberjs.com (dostęp 30.08.2014)
[4]Oficjalna strona platformy Raspberry Pi http://www.raspberrypi.org/, (dostęp 30.08.2014)

3.2.1 Zdarzenia

Nadzorca systemu, podobnie jak każde urządzenie sieci, udostępnia zestaw zdarzeń umożliwiających realizację powierzonych mu zadań. Prezentowany zestaw jest zestawem minimalnym, który możesz zostać rozszerzony wraz z rozrostem funkcjonalności nadzorcy systemu oraz samego standardu komunikacji bezprzewodowej.

Tablica 3.1.: Wykaz zdarzeń wymaganych przez Aplikację sieciową, podany wraz z identyfikatorami i opisami

Zdarzenie	Identyfikator	Opis
NewConf	$0x0066$	Zdarzenie wyzwalane dla zmienionej konfiguracji urządzeń w sieci
NewStatus	$0x0067$	Zdarzenie wyzwalane dla otrzymania zaktualizowanego statusu urządzenia w sieci
PublishMessage	$0x0068$	Zdarzenie powoduje przekazanie wiadomości użytkownikowi za pośrednictwem interfejsu komunikacyjnego

Zdarzenie NewConf

Zdarzenie wyzwalane dla zmienionej konfiguracji urządzeń w sieci. Jest ono wymagane w przypadku obecności więcej niż jednego urządzenia w sieci wpływającego na jej konfigurację. Nadzorca po zaaplikowaniu nowej konfiguracji musi powiadomić pozostałych nadzorców i tej zmianie, poprzez przekazanie im wyzwalacza tego zdarzenia.

Format parametrów
sender;

Parametry

sender
: Bajtowy adres nadawcy wiadomości zlecającego zmianę konfiguracji urządzeń w sieci

Zdarzenie NewStatus

Zdarzenie wyzwalane dla otrzymania zaktualizowanego statusu urządzenia w sieci, pozwalające na jego prezentację użytkownikowi

Format parametrów
sender;status

Parametry

sender
: Bajtowy adres oryginalnego nadawcy wiadomości, którego status uległ zmianie

status
: Zaktualizowany status urządzenia, którego rozpoznanie musi być dokonane przez nadzorcę na podstawie udostępnionej przez wytwórcę dokumentacji formatowania

Zdarzenie PublishMessage

Zdarzenie powoduje przekazanie wiadomości użytkownikowi za pośrednictwem interfejsu komunikacyjnego. Wykorzystanie tego zdarzenia pozwala na prezentację pewnym sytuacji w sieci, o których użytkownik powinien zostać niezwłocznie poinformowany, a nie są zaimplementowane wśród mechanizmów nadzorcy

Format parametrów
sender;message;

Parametry

sender

Bajtowy adres nadawcy wiadomości do opublikowania

message

Dane przekazywanej wiadomości, których format oraz długość są całkowicie dowolne, pod warunkiem nie przekraczania maksymalnego rozmiaru ramki warstwy fizycznej. Najczęściej będzie to format stałego łańcucha znaków.

Rozdział 4

Podsumowanie

Systemy automatyki domowej nieustannie zyskują na popularności, stając się stopniowo elementem wymaganym w nowoczesnym budynku mieszkalnym. Oparcie systemu o komunikację bezprzewodowej pozwala nie tylko uprościć instalację i obniżyć jej koszt, ale również dodatkowo podnosi wygodę użytkownika. Jednak dostępne standardy bezprzewodowe są produktami zamkniętymi, narażającymi prywatność oraz bezpieczeństwo użytkownika końcowego. Stworzenie nowego, otwartego protokołu komunikacji wraz z w pełni otwartym systemem automatyki domowej wypełnić ma ten rynkowy deficyt oraz zaoferować rozwiązanie konkurencyjne w stosunku do obecnie dostępnych.

Standard komunikacji bezprzewodowej zrealizowany w ramach pracy określa obie warstwy stosu: warstwę fizyczną oraz aplikacji. Podczas definiowania specyfikacji warstwy fizycznej, rozpatrzona została kwestia wykorzystywanej częstotliwości komunikacji z uwzględnieniem jej dostępności, zasięgu oraz możliwych zakłóceń w paśmie. Ponadto, wykorzystane zostały mechanizmy zapewniające gwarancję poprawności transmisji, jak sumy kontrolne, wiadomości potwierdzające odbiór, algorytm unikania kolizji, uwzględniając jednocześnie możliwości dostępnych na rynku modułów komunikacji bezprzewodowej. Warstwa aplikacji, realizująca zadania niższych warstw referencyjnego modelu OSI, dodatkowo zwiększa jakość komunikacji poprzez wykorzystanie dwóch algorytmów routingu reaktywnego. Jednocześnie, takie przesunięcie funkcji pozwoliło na bezpośrednie wykorzystanie modelu sieci sterowanej zdarzeniami, co w połączeniu z mechanizmem wiązania zdarzeń, odwzorowuje charakterystykę zachowania elektrycznych instalacji domowych.

Zdarzenia oraz ich wiązanie, jako kluczowy element zdefiniowanego protokołu, umożliwia spełnienie zadań mu stawianych, czyli przede wszystkim uniwersalność w realizacji celów i maksymalna prostota implementacji stosu protokołu. Asynchroniczność modelu sterowania zdarzeniami pozwala również na obniżenie zapotrzebowania energetycznego urządzeń zgodnych z stworzonym standardem oraz zwiększenie responsywności całego systemu. Dzięki tym czynnikom możliwa jest implementacja systemu przy wykorzystaniu aktualnie dostępnych na rynku, tanich przy zakupie oraz w eksploatacji modułów, co w połączeniu z otwartością protokołu, licencyjną wolnością oraz wygodą użytkowania urządzeń pozwala na szybsze rozpropagowanie standardu na rynku.

Dla empirycznej weryfikacji poprawności założeń stawianych w niniejszej pracy, zrealizowane zostały prototypy urządzeń oparte na zdefiniowanym standardzie komunikacji bezprzewodowej. Na potrzeby implementacji, wybrane zostały przykładowe platformy sprzętowe oraz programowe. Bazując na tej podstawie, stworzone zostały sterowniki dla modułu komunikacji bezprzewodowej, programowe implementacje stosu protokołu oraz aplikacje realizujące funkcje poszczególnych urządzeń, dedykowane dla systemów operacyjnych GNU/Linux oraz ChibiOS/RT. Ponadto, dla systemu Linux napisana została aplikacja nadzorcy systemu, zarządzająca zdarzeniami sieci oraz routingiem, udostępniająca

za pośrednictwem wbudowanego w nią serwera protokołu WebSocket interfejs zdalnego przekazywania komend konfiguracji do systemu. Dla umożliwienia wygodnego zarządzania system, stworzona została graficzny panel administracyjny w formie aplikacji internetowej, pozwalającej na zdalne sterowanie urządzeniami, wiązanie zdarzeń oraz konfigurację routingu.

Stworzony w ramach pracy standard komunikacji bezprzewodowej oraz jego przykładowa implementacja powstały z założeniem dalszego rozwoju, aż do momentu uzyskania statusu technologii gotowej do sprzedaży użytkownikom końcowym. Planowane jest stworzenie mechanizmu wyzwalaczy zdarzeń tworzonych dynamicznie np. wyzwoleń warunkowych oraz rozszerzenie istniejącego zbioru zdarzeń. Urządzenia mają zostać wyposażone w parsery języka Lua, pozwalając na zdalne tworzenie skryptów realizujących fakultatywne funkcje urządzenia. Co więcej, planowane jest stworzenie mechanizmów zdalnej aktualizacji oprogramowania urządzeń, usprawnienie algorytmów routingu czy dalsze obniżenie poboru energii przez urządzenia.

Bibliografia

[Ass06] KNX Association. *Handbook for Home and Building Control: Basic Principles.* 2006.

[Ass09] KNX Association. *KNX System Specifications: Architecture*, 2009.

[BHO+99] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, 1999.

[BJM96] J. Broch, D.B. Johnson, and D.A. Maltz. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. Technical report, Carnegie Mellon University Pittsburgh, 1996.

[FGR+07] R. Friedman, D. Gavidia, L. Rodrigues, A. Viana, and S. Voulgaris. Gossiping on manets: the beauty and the beast. *ACM Operating Systems Review*, 41(5), 2007.

[Har03] R. Harper. *Inside the Smart Home.* Springer, Londyn, 2003.

[HHL02] Z. Haas, J.Y Halpern, and Li Li. Gossip-based ad hoc routing. Technical report, School of Electrical and Computer Engineering/Department of Computer Science, Cornell University, 2002.

[Ins13] Insteon. *Insteon®; Whitepaper: The Details*, 2013.

[ITU97] ITU. *ITU-R P.1238: Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 900 MHz to 100 GHz*, 1997.

[ITU12] ITU. *Short range narrow-band digital radiocommunication transceivers – PHY and MAC layer specifications*, 2012.

[LL99] C.R. Lin and Jain-Shing Liu. Qos routing in ad hoc wireless networks. *IEEE Journal on selected ares in communications*, 17(8), sierpień 1999.

[oEEE11] The Institute of Electrical and Inc. Electronics Engineers. *IEEE Standard for Local and metropolitan area networks — Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 2011.

[Per03] C. Perkins. Ad hoc on-demand distance vector (aodv) routing. Technical report, Nokia Research Center, 2003.

[ROZ07] Rozporządzenie ministra transportu z dnia 3 licpa 2007 w sprawie urządzeń radiowych nadawczych lub nadawczo–odbiorczych, które mogą być używane bez pozwolenia radiowego (dz. u. 138, poz. 972, 2007.

[Sir14] G. Sirio. Dokumentacja techniczna systemu chibios/rt, 2014. (dostęp 30.08.2014).

[Wei05] T. Weinzierl. Stack implementation for knx-rf, 2005.

[ZA08] Inc. ZigBee Allicane. *ZigBee Specification*, 2008.

[ZA10] Inc. ZigBee Allicane. *ZigBee RF4CE Specification*, 2010.

[Zen06] Zensys. *Software Design Specification, Z-Wave Protocol Overview*, 2006.

[Zen07] Zensys. *Software Design Specification, Z-Wave Device Class Specification*, 2007.

Dodatek A

Przewodnik użytkownika