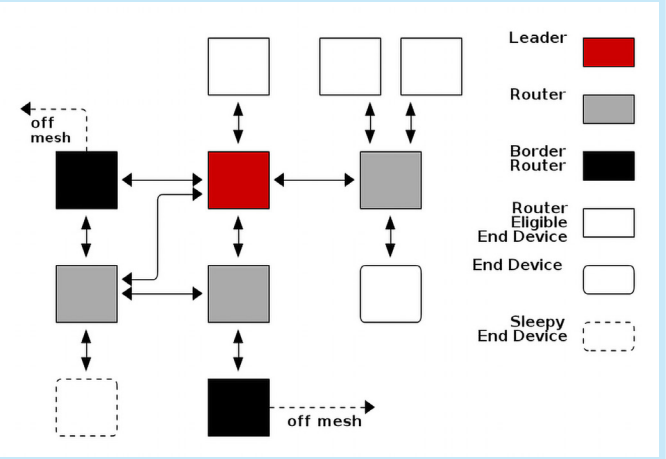
Wybrane pozostałe artykuły kursu „Systemy dla Internetu Rzeczy”

S12. Oprogramowanie narzędziowe dla układów CC26xx i CC13xx platformy SimpleLink, „Elektronika Praktyczna” 11/2017
S13. Zestaw CC26x2R1 LaunchPad, „Elektronika Praktyczna” 1/2018
S14. Podglądanie ruchu w sieci radiowej z protokołem IEEE 802.15.4, „Elektronika Praktyczna” 2/2018
S15. Zestaw CC1352R1 LaunchPad, „Elektronika Praktyczna” 5/2018
S17. Jednoczesna komunikacja radiowa z użyciem dwóch protokołów i w dwóch pasmach, „Elektronika Praktyczna” 8/2018
S18. Praca z jednoczesną komunikacją radiową z użyciem dwóch protokołów i w dwóch pasmach, „Elektronika Praktyczna” 9/2018



Systemy dla Internetu Rzeczy (20)

Sieć z protokołem Thread

Protokół Thread jest przeznaczony do obsługi Internetu Rzeczy, a szczególnie aplikacji związanych z ideą inteligentnego domu (Smart Home) z obsługą urządzeń domowych, klimatyzacji, systemów bezpieczeństwa i oświetlenia czy zarządzaniem zużyciem energii. Nowy protokół Thread zaczyna stawać się zupełnie realną alternatywą dla protokołu ZigBee. Szczególnie jego darmowa wersja OpenThread, obecnie wspierana przez firmę Google. Czyżby szykowała nam się kolejna duża kariera, jak w przypadku Androida?

Thread jest protokołem sieci bezprzewodowej opracowanym przez konsorcjum Thread Group (pierwsza specyfikacja w roku 2015) [15]. W prace konsorcjum zaangażowane są czołowe firmy z branży jak: Google, Samsung, ARM, NXP, Silicon Labs, Nordic Semiconductor, Osram, Qualcomm, Schneider, Somfy, Tyco czy Texas Instruments. Właśnie teraz do konsorcjum (IX 2018) dołączyła firma Apple, twórca systemu inteligentnego domu HomeKit. O roku 2017 konsorcjum Thread Group udziela certyfikatu zgodności z protokołem Thread. Najnowsza wersja specyfikacji jest darmowo dostępna od 7 maja 2018.

Thread pracuje z wykorzystaniem protokołu komunikacyjnego IEEE 802.15.4. Protokół ten specyfikuje dolne warstwy MAC oraz PHY modelu OSI (**rysunek 1**). Asynchroniczny protokół komunikacyjny umożliwia niski pobór mocy przez węzły. W kolejnej warstwie została zastosowana sieć 6LoWPAN (Low-Power Wireless Personal Area Networks), która umożliwia pracę sieci w konfiguracji „mesh” (255 węzłów) oraz adresowanie IPv6. W kolejnej, wyższej warstwie stosowany jest User Datagram Protocol (UDP). Specyfikacja Thread nie definiuje niczego w warstwie aplikacji. Tutaj różni się z protokołem ZigBee. Komunikacja jest prowadzona w pasmie 2,4 GHz.

OpenThread jest otwartą (darmową) implementacją specyfikacji Thread powstałą z inicjatywy firmy NestLabs [14]. Zawiera warstwę sieciową, funkcje urządzeń oraz wsparcie dla routera krawędziowego (Border Router).

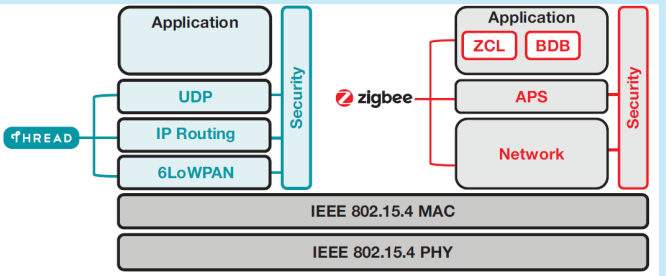
Funkcje układów

W sieci Thread mogą pracować układy (węzły) dwóch typów, z różnymi trybami i pełniące różne funkcje. Układy (device) w sieci Thread mogą być tylko dwóch typów [12]:

- **Full Thread Device (FTD)** – może pełnić funkcję End Device, Router, Leader lub Border Router.
- **Minimal Thread Device (MTD)** – może odgrywać tylko rolę End Device. Niektóre konfiguracje mogą przechodzić w stan uśpienia.

W sieci Thread układy mogą pełnić jedną z czterech funkcji [12]:

- **End Device** – układ dołączony do jednego układu Router. Cały ruch danych przechodzi przez rodzicielski Router (parent Router). Jeśli układ utraci kontakt ze swoim rodzicielskim Routerem, to może poszukać nowego.
- **Router** – łączy się z innymi układami Router. Pracuje jako rodzicielski Router dla układów End Device. Router pełni funkcję proxy dla układów potomnych wykonujących operacje w sieci, jak odpowiedź na zapytanie adresowe. Wykonuje wysyłanie, odbiór i przekazywanie dalej komunikatów (messages) w sieci typu mesh. Router przechowuje kopie danych sieci uzyskaną do układu Leader.



Rysunek 1. Warstwy protokołu Thread oraz ZigBee [9]

- **Leader** – jest układem Router, który został wyznaczony do podejmowania decyzji w sieci. Te decyzje dotyczą wyznaczania aktywnego układu dołączania (commissioner), aktualizacji układu Router i wiele innych. Układem Leader staje się w partycji (partition) sieci pierwszy układ Router.
- **Border Router** – układ Router z połączeniem poza sieć. Pełni funkcję mostu (bridge) pomiędzy siecią Thread (IPv6) a inną siecią IPv6, jak Wi-Fi lub Ethernet.

Układy startują zawsze jako End Device. Dopiero potem zmieniają rolę na wymaganą w sieci. Funkcje układów (węzłów) w przykładowej sieci Thread zostały pokazane na rysunku zamieszczonym w nagłówku artykułu [12].

Układy scalone z obsługą protokołu Thread

Już kilka układów scalonych, różnych producentów, umożliwia w pasmie 2,4 GHz jednoczesną transmisję w sieci z protokołem Thread oraz z innym protokołem, jak Bluetooth 5 Low Energy lub protokołem autorskim producenta.

- Układ scalony Nordic Semiconductor nRF52840 umożliwia w pasmie 2,4 GHz transmisję w sieciach z protokołem Bluetooth 5, Thread, ZigBee, IEEE 802.15.4 oraz ANT [8]. Możliwa jest też jednoczesna obsługa transmisji z obsługą stosu BLE 5 oraz stosu OpenThread. Układ ma certyfikację konsorcjum Thread Group.
- Układ scalony Silicon Labs Mighty Gecko EFR32MG1x obsługuje transmisję w pasmie 2,4 GHz z protokołem Zigbee 3.0, Thread, Bluetooth 5 oraz protokół autorski. Możliwe jest tworzenie sieci mesh [13]. Moduł radiowy obsługuje również transmisję w pasmie poniżej 1 GHz (915 MHz, 868 MHz, 433 MHz, 169 MHz). W tym pasmie obsługuje protokoły M-BUS, WAN oraz protokół autorski. Układ ma certyfikację konsorcjum Thread Group. Również moduł sprzętowy Silicon Labs Mighty Gecko Module MGM12P z tym układem ma certyfikację.
- Układ scalony Texas Instruments SimpleLink CC2652R1 umożliwia transmisję w pasmie 2,4 GHz z obsługą wielu protokołów: Bluetooth 5 Low Energy, IEEE 802.15.4g, 6LoWPAN, Thread, ZigBee, KNX RF, Wi-SUN oraz autorski protokół EasyLink. Układ ma certyfikację konsorcjum Thread Group [1]. Układ scalony Texas Instruments SimpleLink CC1352R (oraz jego wersja z dodatkowym wzmacniaczem CC1352P) dodatkowo może pracować w dwóch pasmach: 2,4 GHz i w pasmie poniżej 1 GHz (Sub-1GHz) [S17, 18].

Dokumentacja

Stos TI-OpenThread, dostarczany przez firmę Texas Instruments, oprócz wszystkich funkcji implementacji OpenThread ma dodatkową zaletę pracy na szczycie systemu operacyjnego czasu rzeczywistego TI-RTOS [12]. W ten sposób udostępniany jest cały ekosystem modułów sprzętowych LaunchPad oraz drajwery układów peryferyjnych.

Dokumentacja dotycząca użycia protokołu Thread z układami scalonymi rodziny SimleLink CC13x2/CC26x2 firmy Texas Instruments znajduje się w portalu TIREX [10]. Dla procesora CC1352R (oraz CC1352P) opis znajduje się w pakiecie *SimpleLink CC13x2 Software Development Kit* [5]. Trzeba nawigować do ścieżki *Documents* → *Documentation Overview* oraz przejść do sekcji *TI OpenThread Stack Documentation*. Tam udostępnianych jest kilka odnośników do dokumentów opisu: TI Thread Release Notes, TI Thread Quick Start Guide, TI Thread User's Guide, TI Thread Border Router Setup Guide, TI Thread Runtime APIs oraz OpenThread Stack APIs.

W głównej gałęzi *Documents* → *TI Thread* dokumentacji pakietu CC13x2 SDK dostępne są dwa dodatkowe dokumenty: TI Thread API Guide oraz TI Thread Migration Guide.

W pakiecie *SimpleLink Academy 2.20.03 for SimpleLink CC13x2 SDK 2.20* [7] są udostępnione dwa bardzo ciekawe ćwiczenia laboratoryjne *Thread CLI Project Zero* [11] oraz *Thread Border Router Setup*.

W pakiecie CC13x2 SDK [5] dostępnych jest osiem przykładowych projektów. Każdy projekt zawiera plik *README.html*, w którym jest zamieszczony opis aplikacji projektu. Są one też widoczne w portalu TIREX.

Dla procesora CC2652R takie same zasoby znajdują się w pakiecie *SimpleLink CC26x2 Software Development Kit* [6]. Organizacja i dostęp do tych zasobów jest taki sam, jak to zostało opisane powyżej dla procesora CC1352R.

Przygotowanie do pracy

Dokładny opis instalowania oprogramowania jest zamieszczony w poprzednim artykule „Oprogramowanie narzędziowe dla układów CC26xx i CC13xx platformy SimpleLink" [S12]. Opis zestawu startowego CC1352R1 LaunchPad jest zamieszczony w poprzednim artykule „Zestaw CC1352R1 LaunchPad" [S15].

Do wykonania zamieszczonych dalej zadań są potrzebne:

- Dwa zestawy startowe CC1352R1 LaunchPad [3] (lub CC2652R1 LaunchPad [4]).
- Aplikacja CCS 8.1 z zainstalowaną obsługą układów CC13xx/CC26xx.
- Zainstalowany pakiet programowy CC13x2 SDK [5] (lub CC26x2 SDK [6] w przypadku stosowania zestawu startowego CC2652R1 LaunchPad).

Zadania dalej opisane są wykonywane z zastosowaniem zestawu startowego CC1352R1 LaunchPad [S15]. Można je również wykonać z zastosowaniem zestawu startowego CC2652R1 LaunchPad [S13]. Trzeba wtedy jedynie zastosować dedykowany dla niego pakiet programowy CC26x2 SDK [6].

Zadanie 1 – Zaprogramowanie aplikacji Cli_Ftd

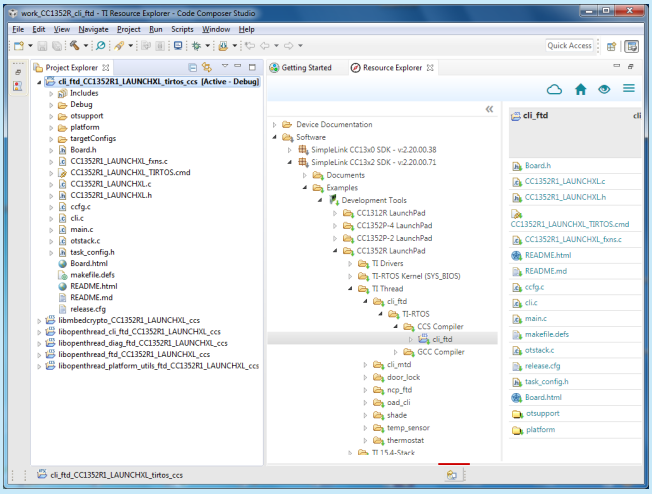
Najłatwiejsze rozpoczęcie pracy ze stosem TI OpenThread to użycie przykładowej aplikacji Cli_Ftd [11]. Nazwa bierze się od dwóch nazw: Command Line Interface (CLI) oraz Full Thread Device (FTD).

Przykładowa aplikacja Cli_Ftd daje podstawowe wprowadzenie do stosu Thread. Pozwala na rozpoczęcie pracy ze stosem. Umożliwia utworzenie sieci i dołączenie węzła. Krok po kroku tworzona jest *ad-hoc* sieć Thread pomiędzy dwoma węzłami z aplikacją Cli_Ftd.

A1. Dołącz pierwszy zestaw startowy CC1352R1 LaunchPad do komputera, używając kabla USB.
A2. W oknie *Menadżer Urządzeń* sprawdź numer portu **COMx1** dla kanału *Aplication/User UART*. Zanotuj ten numer.
A3. Wystartuj środowisko CCS.
A4. Poczekaj na zakończenie sprawdzania aktualizacji środowiska. Jedyny sposób to obserwowanie informacji o postępie sprawdzania wyświetlanych na pasku stanu.

Importuj projekt aplikacji ProjectZero

A5. Kliknij na przycisk *Browse Examples*.
W oknie *Resource Explorer* rozwiń ścieżkę:
SimpleLink CC13x2 SDK – v.2.20.00.71 → Examples → Development Tools → CC1352R LaunchPad → thread → cli_ftd → TI-RTOS → CCS Compiler → cli_ftd
A6. W prawym oknie kliknij na przycisk *Import to IDE* 
A7. W wyświetlanym oknie kliknij na *I Have Read And Agree*.
A8. Poczekaj na zakończenie pobierania wszystkich projektów. To może chwilę trwać. Należy obserwować pasek stanu na dole okna. Szczególnie trzeba poczekać, aż znikną znaczki wykrzyknika nałożonego w oknie *Project Explorer* na ikonki folderów projektów.



Rysunek 2. Okno perspektywy *CCS Edit* po załadowaniu projektów aplikacji *project_zero_app*

A9. W oknie *Project Explorer* rozwiń drzewo głównego projektu (**rysunek 2**) o nazwie ***cli_ftd_CC26x2R1_LAUNCHXL_tirtos_ccs***.

Oprócz projektu głównego widocznych jest także kilka projektów zależnych. Zawierają one rdzeń OpenThread, biblioteki CLI, diagnostykę i wiele innych modułów użytkowych.

W dokumencie *readme.txt* jest zamieszczony opis aplikacji. Można go otworzyć i zobaczyć bezpośrednio w środowisku CCS. Wystarczy dwa razy kliknąć na linię nazwy. Jednak wygodniej jest go otworzyć w przeglądarce internetowej w portalu TIREX [10].

Wykonaj budowanie wszystkich projektów

A10. W oknie *Project Explorer* kliknij na linię projektu ***cli_ftd_CC26x2R1_LAUNCHXL_tirtos_ccs*** (wybierz go).

A11. Kliknij na ikonkę *Build*.

A12. Czekaj na pełne zakończenie budowania wszystkich projektów.

W oknie *Console* pokazywane są informacje o postępach budowania. Aplikacja zawiera wiele projektów, dlatego czas budowania jest raczej **bardzo** długi! Dodatkowo jest on wydłużony przez zastosowanie wysokiego poziomu optymalizacji kodu wynikowego.

Zaprogramuj pierwszy moduł LaunchPad

A13. Kliknij na przycisk „*Debug*”.

A14. Zostaje otwarte okno *CCS Debug* i startuje sesja debugowa.

A15. Czekaj na zatrzymanie programu w pierwszej instrukcji funkcji *main()*. To może trochę trwać. Jest sporo kodu do zaprogramowania. Obserwuj informacje na pasku stanu i w oknie *Console*.

A16. W perspektywie *CCS Debug* zakończ sesję debugową. Kliknij na ikonkę *Terminate*.

A17. Czekaj, aż CCS przełączy widok na perspektywę *CCS Edit*.

A18. Odłącz pierwszy zestaw startowy CC1352R1 LaunchPad od komputera.

A19. Oznacz go jako *Pierwszy*.

Zaprogramuj drugi moduł LaunchPad

A20. Dołącz drugi zestaw startowy CC1352R1 LaunchPad do komputera, używając kabla USB.

A21. W oknie *Menadżer Urządzeń* sprawdź numer portu **COMx2** dla kanału *Aplication/User UART*. Zanotuj ten numer.

A22. Kliknij na przycisk „*Debug*”.

A23. Czekaj na zatrzymanie programu w pierwszej instrukcji funkcji *main()* i wyświetlenie strzałki na lewo od 80. linii kodu.

A24. Zamknij aplikację CCS.

A25. Odłącz drugi zestaw startowy CC1352R1 LaunchPad od komputera.

A26. Oznacz go jako *Drugi*.

Zadanie 2 – Utworzenie sieć Thread

Dwa zadania systemu operacyjnego TI-RTOS są uruchamiane w ramach aplikacji Cli_Ftd. Jedno zadanie realizuje stos TI-OpenThread. Drugie zadanie realizuje moduł poleceń standardu OpenThread. Po wystartowaniu pracy modułów czerwona dioda LED na module LaunchPad zaczyna powoli błyskać. Sygnalizuje to poprawną pracę stosu.

Interfejs poleceń CLI jest udostępniony poprzez port UART modułu LaunchPad i wymaga zastosowania na komputerze programu terminalu szeregowego.

Dołącz pierwszy program terminalu

B1. Dołącz pierwszy zestaw startowy CC1352R1 LaunchPad z zaprogramowaną aplikacją Cli_Ftd do komputera, używając kabla USB.

B2. Dioda zacznie powoli mrugać.

B3. Wystartuj aplikację terminalu *PuTTY*.

B4. Kliknij przycisk *Serial*. W polu *Serial line* wpisz numer portu szeregowego **COMx1**. W polu *Speed* wpisz prędkość transmisji 115200.

B5. W oknie *Category* kliknij na *Colours*. Zaznacz *Use system colours*. Kliknij *OK*.

B6. Rozciągnij pierwsze okno aplikacji PuTTy wszerz (ważne).

Uwaga: Nie odłączaj modułu CC1352R1 LaunchPad od komputera, jeśli jest z nim połączony program terminalu PuTTy.

Dołącz drugi program terminalu

B7. Dołącz drugi zestaw startowy CC1352R1 LaunchPad z zaprogramowaną aplikacją Cli_Ftd do komputera, używając kabla USB.

B8. W oknie *Menadżer Urządzeń* sprawdź numer portu COMx2 dla kanału *Aplication/User UART*. Zanotuj ten numer.

B9. Wystartuj aplikację terminalu *PuTTY*.

B10. Kliknij przycisk *Serial*. W polu *Serial line* wpisz numer portu szeregowego **COMx2**. W polu *Speed* wpisz prędkość transmisji 115200.

B11. W oknie *Category* kliknij na *Colours*. Zaznacz *Use system colours*. Kliknij *OK*.

B12. Rozciągnij okno aplikacji *PuTTy* wszerz (ważne).

Tworzenie sieci Thread

Tworzenie sieci Thread zaczyna się od utworzenia nowej partycji z zastosowaniem odpowiedniego kanału transmisji. Pierwszy układ definiuje identyfikator sieci PANID i tworzy sieć składającą się z pojedynczego węzła, tzw. singleton. Ustawiane są domyślne parametry sieci. W przykładzie został zastosowany kanał 14 oraz PANID = 0xface.

Uwaga: Najprostszy sposób przenoszenia poleceń z pliku instrukcji tego przykładu do okna terminalu:

- W oknie instrukcji zaznacz i skopiuj polecenie (Ctrl-C).
- W oknie terminalu kliknij prawym klawiszem myszki i kliknij klawisz *Enter*.

B13. W oknie **pierwszego** terminala naciśnij klawisz *Enter*. Zostanie wyświetlony znak zachęty (kursor) „>” (**rysunek 3**).

B14. Wpisz polecenie: „***channel 14***” ustawienia kanału. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

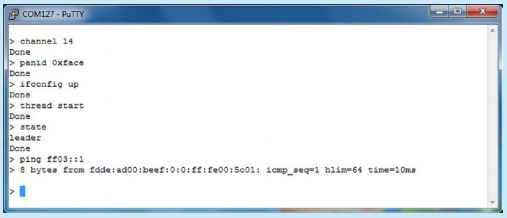
B15. Wpisz polecenie: „***panid 0xface***” ustawienia identyfikatora sieci PAN. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

B16. Wpisz polecenie: „***ifconfig up***”. Polecenie ifconfig up włącza moduł radiowy układu scalonego. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

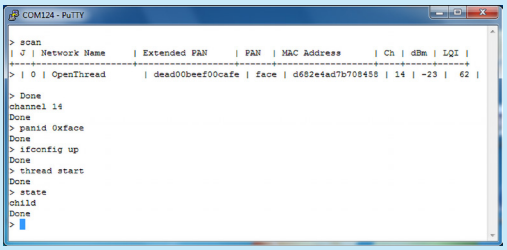
B17. Wpisz polecenie: „***thread start***”. Polecenie thread start powoduje wystartowanie stosu Thread. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

B18. Wpisz polecenie: „***state***”. Zostanie wyświetlony stan węzła: „leader”.

Węzeł zmienił stan z „detatched” na „leader” nowej partycji sieci Thread (rysunek 3). Teraz już pracuje sieć Thread w kanale 14 z identyfikatorem PANID = 0xface.



Rysunek 3. Okno terminalu 1 dla tworzenia sieci Thread



Rysunek 4. Okno terminalu 2 dla tworzenia sieci Thread

Dodawanie węzła do sieci Thread

Można użyć innego modułu z aplikacją Cli_Ftd do zobaczenia rozgłaszania w sieci wykonywanego przez pierwszy moduł.

B19. W oknie **drugiego** terminalu naciśnij klawisz *Enter*. Zostanie wyświetlony znak zachęty „>”.

B20. Wpisz polecenie: „***scan***”. Zostanie wyświetlony stan sieci (**rysunek 4**).

B21. Jeśli w terminalu nie jest wyświetlany kursor, to wciśnij klawisz *Enter*.

Ustaw ten sam numer kanału i identyfikator sieci jak dla pierwszego modułu LaunchPad.

B22. Wpisz polecenie: „***channel 14***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

B23. Wpisz polecenie: „***panid 0xface***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

Teraz wystartuj stos Thread.

B24. Wpisz polecenie: „***ifconfig up***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

B25. Wpisz polecenie: „***thread start***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

B26. Wpisz polecenie: „***state***”. Zostanie wyświetlony stan węzła: „child” (rysunek 4).

Weryfikacja działania sieci Thread

Obecnie dwa urządzenia z aplikacją Cli_Ftd zostały połączone w sieci Thread. Można teraz użyć polecenia „ping” do wysłania żądanie echa ICMPv6 do węzłów sieci mesh. W rezultacie odpowiedzą wszystkie węzły IPv6 w sieci o zgodnym identyfikatorze.

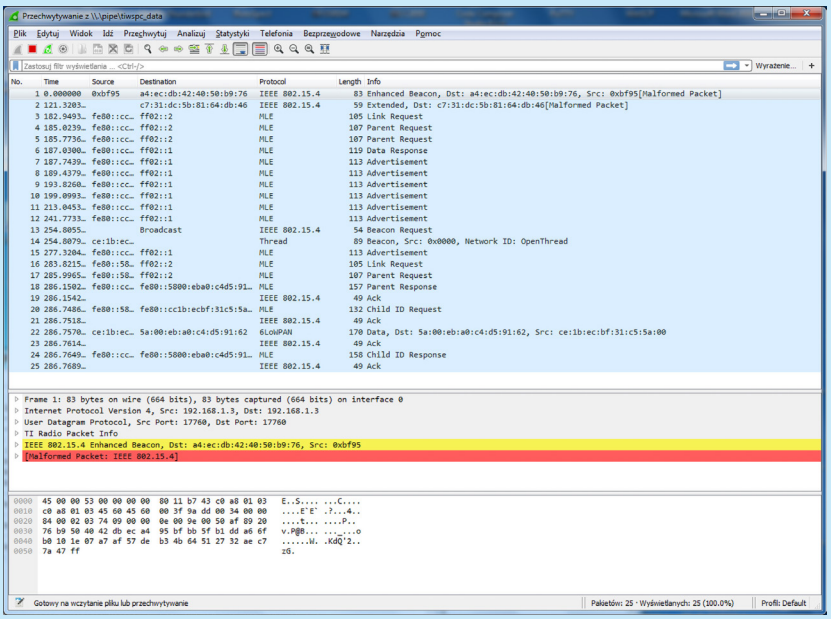
B27. W oknie **pierwszego** terminalu wpisz polecenie: „***ping ff03::1***”. Zostanie wyświetlona odpowiedź drugiego modułu LaunchPad.

Sieć Thread została utworzona i pracuje pomiędzy dwoma układami. Jednak zostały zastosowane domyślne ustawienia parametrów, jak klucz główny (master). Mogą one być niezgodne z ustawieniami stosowanymi przez innych producentów układów transmisji. Dodatkowo taki sposób dodawania węzła do sieci nie jest bezpieczny.

Podglądanie transmisji sieci Thread

Dokumentacja pakietu SDK skrywa ciekawą możliwość. Okazuje się, że modułu CC1352R1 LaunchPad można użyć do podglądania transmisji w sieci Thread. Transmisji, która jest zaszyfrowana! Trzeba do tego użyć darmowej aplikacji SmartRF Protocol Packet Sniffer 2 (Texas Instruments) oraz znanej aplikacji Wireshark. Sposób skonfigurowania aplikacji Wireshark jest ukryty w dokumencie [16]. Trzeba pominąć początkowy opis starych wersji i znaleźć opis dla CC13x2. Tam jest pokazane, jak przekazać poufne informacje sieci do aplikacji Wireshark. Ale i tak trzeba użyć dodatkowego sposobu skonfigurowania aplikacji Wireshark. Jest on dokładnie opisany w poprzednim artykule serii „Podglądanie ruchu w sieci radiowej z protokołem IEEE 802.15.4” [S14].

Przykład podglądania transmisji zachodzącej podczas wykonywania zadania 1 jest pokazany na **rysunku 5**. Linie numer od 1 do 15 odpowiadają za transmisję podczas wykonywania poleceń od C1 do C25. Od linii 16 zaczyna się reakcja na polecenie C26.



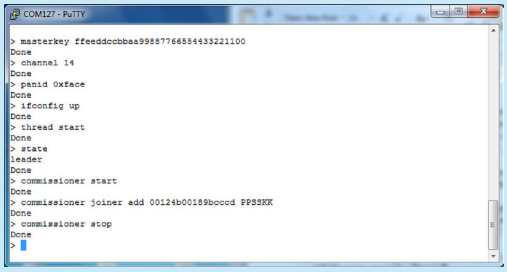
Rysunek 5. Podgląd transmisji sieci Thread w programie Wireshark

Zadanie 3 – Bezpiecznie dołączenie nowego węzła do sieci Thread

Dołączanie (Commissioning) to proces bezpiecznego dodawania nowego węzła (urządzenia) do sieci bezprzewodowej lub zestawianie połączenia pomiędzy dwoma urządzeniami. Thread odróżnia nawiązanie połączenia (jak pokazano w poprzednim zadaniu) od dołączania. Wynika to z bardzo dużych wymagań obliczeniowych procesu dołączania związanych z kryptografią. Węzły oprócz różnych ról, typów i trybów mogą pełnić też różne funkcje, jako układ dołączający (Commissioner) lub układ dołączany (Joiner).

Na wysokim poziomie można opisać proces dołączania następująco:

- Aktywowanie pracy układu dołączającego (Commissioner).
- Dostarczenie danych układu dołączanego (Joiner) układowi dołączającemu (Commissioner)



Rysunek 6. Okno terminalu 1 do dołączania do sieci Thread

- Układ dołączający (Commissioner) powiadamia sieć o potencjalnym układzie dołączanym (Joiner)
- Układ dołączany (Joiner) skanuje sieć.
- Router odpowiada układowi dołączającemu (Commissioner). Ten Router jest teraz nazywany Joiner Router.
- Układ dołączający (Commissioner) oraz układ dołączany (Joiner) zestawiają sesję DTLS (Datagram Transport Layer Security) poprzez Joiner Router z użyciem poufnych (pre-shared) informacji.
- Układ dołączający (Commissioner) oraz układ dołączany (Joiner) zamykają sesję DTLS.
- Joiner Router używa klucza tymczasowego (ephemeral) do bezpiecznego przesyłania informacji do układu dołączanego (Joiner).

Ponowne tworzenie sieci Thread

C1. Wykonaj jednocześnie Reset pierwszego i drugiego modułu LaunchPad. Przyciśnij i przytrzymaj przycisk Reset znajdujący się na górnej krawędzi płytki drukowanej modułu.

C2. Kliknij w oknie **pierwszego** terminalu i wciśnij klawisz *Enter*. Zostanie wyświetlony znak zachęty (kursor) „>”.

C3. Wpisz polecenie ustawienia nowego klucza głównego (zamiast wartości domyślnej):
„***masterkey ffeeddccbbaa99887766554433221100***”.

C4. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

C5. Wpisz polecenie: „***channel 14***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

C6. Wpisz polecenie: „***panid 0xface***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

C7. Wpisz polecenie: „***ifconfig up***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

C8. Wpisz polecenie: „***thread start***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

C9. Wpisz polecenie: „***state***”. Zostanie wyświetlony stan węzła: „leader”.

Węzeł zmienił stan z „detatched” na „leader” nowej partycji sieci Thread (**rysunek 6**).

Odczyt danych poufnych

Ponieważ w sieci jest tylko jeden węzeł, to zostanie on użyty jako układ dołączający (Commissioner). Drugi moduł LaunchPad zostanie użyty jako układ dołączany (Joiner). Najpierw trzeba sprawdzić jego EUI64 (extended unique identifier).

C10. Kliknij w oknie **drugiego** terminalu i wciśnij klawisz *Enter*. Zostanie wyświetlony znak zachęty „>”.

C11. W oknie drugiego terminalu wpisz polecenie: „***eui64***”. Zostanie wyświetlone identyfikator oraz potwierdzenia zakończenia operacji: „Done” (**rysunek 7**).

Rozpoczęcie dołączania

Teraz należy wystartować układ dołączający (Commissioner).

C12. W oknie **pierwszego** terminalu wpisz polecenie: „***commissioner start***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

Teraz trzeba układowi dołączającemu (Commissioner) podać dane (joiner entry) układu dołączanego (Joiner) z jego UID64 i hasłem PSK. W tym zadaniu jako hasło PSK (pre-shared key) zostanie użyte „PPSSKK”.

C13. W oknie pierwszego terminalu wpisz polecenie, ale z zastosowaniem UID64 **uzyskanego w kroku C11** (rysunek 7):
„***commissioner joiner add 00124b00189bcccd PPSSKK***”

C14. Zostanie wyświetlone potwierdzenie zakończenia operacji: „Done”.

Układ dołączający (Commissioner) ma już teraz informacje o układzie, który może starać się (petitioning) dołączyć do sieci.

Teraz trzeba się upewnić, że klucz główny tego nowego urządzenia jest ustawiony na wartość domyślną.

C15. W oknie **drugiego** terminalu wpisz polecenie: „***masterkey***”. Zostanie wyświetlona wartość domyślnego klucza oraz potwierdzenia zakończenia operacji: „Done” (rysunek 7).

Dołączanie węzła do sieci Thread

Dwa kolejne polecenia spowodują wystartowanie procesu dołączania. Układ dołączany (Joiner) rozpoczyna negocjować sesję DTLS (Datagram Transport Layer Security) oraz uwierzytelniania z układem dołączającym (Commissioner). Ten proces może zająć dłuższy czas.

C16. W oknie **drugiego** terminalu wpisz polecenie: „***ifconfig up***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

C17. W oknie drugiego terminalu wpisz polecenie: „***joiner start PPSSKK***”. Zostanie wyświetlone potwierdzenie zakończenia operacji: „Done” oraz potwierdzenie sukcesu „Join Success” (rysunek 7).

C18. Wciśnij klawisz *Enter*.

Teraz trzeba sprawdzić, czy informacje sieci Thread dotarły do nowego węzła.

C19. W oknie drugiego terminalu wpisz polecenie: „***masterkey***”.

C20. Zostanie wyświetlony aktualny klucz główny oraz potwierdzenie zakończenia operacji: „Done” (rysunek 7).

Teraz trzeba wystartować stos Thread.

C21. W oknie drugiego terminalu wpisz polecenie: „***thread start***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done”.

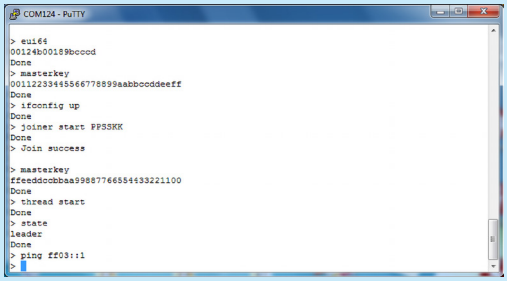
C22. W oknie drugiego terminalu wpisz polecenie: „***state***”.

Zostanie wyświetlony stan węzła. Według opisu powinien to być stan „child”. Ale w praktyce zawsze wychodzi stan „leader” (rysunek 7). Próba użycia polecenia ping do nawiązania komunikacji też nie daje skutku (rysunek 7).

W wielu próbach już wydanie polecenia *joiner start PPSSKK* kończy się odpowiedzią *Join failed [NotFound]*. Znalezienie przyczyn tego problemu wymaga jednak dokładniejszego zapoznania się ze stosem TI-OpenThread.

Na koniec trzeba wyłączyć pracę układu dołączającego (Commissioner).

C23. W oknie **pierwszego** terminalu wpisz polecenie: „***commissioner stop***”. Zostanie wyświetlone potwierdzenia zakończenia operacji: „Done” (rysunek 6).



Rysunek 7. Okno terminalu 2 do dołączania do sieci Thread

Literatura

Procesory

1. CC1352R (PREVIEW) SimpleLink Multi-Band CC1352R Wireless MCU, Texas Instrumets, ***http://bit.ly/2vJkmqm***
2. CC2652R (PREVIEW) SimpleLink(TM) CC2652R Multi-Standard Wireless MCU, Texas Instrumets, ***http://bit.ly/2yVwDGz***

Moduły sprzętowe

3. SimpleLink Multi-Band CC1352R Wireless MCU LaunchPad Development Kit, LAUNCHXL-CC1352R1, ***http://bit.ly/2I0oP99***
4. SimpleLink CC26x2 wireless MCU LaunchPad Development Kit, LAUNCHXL-CC26X2R1 (CC2652R1), ***http://bit.ly/2Pd4ghd***

Pakiety programowe

5. SimpleLink CC13x2 Software Development Kit, SIMPLELINK-CC13X2-SDK, Ver 2.20.00.71, ***http://bit.ly/2M4Anub***
6. SimpleLink CC26x2 SW Development Kit, SIMPLELINK-CC26X2-SDK, Ver 2.20.00.49, ***http://bit.ly/2CZkxAL***

SimpleLink Academy

7. SimpleLink Academy 2.10.02 for SimpleLink CC13x2 SDK 2.20.03.05, ***http://bit.ly/2vSOzDm***
8. nRF52840 High-end Bluetooth 5/Thread/802.15.4/ANT/2.4GHz multiprotocol SoC, Nordic Semiconductor, ***http://bit.ly/2r29E8H***

Opisy

9. Thread and Zigbee for home and building automation, SWAY012, 01 Mar 2018, Texas Instrumets, ***http://bit.ly/2yVmfPd***
10. TI Resource Explorer (TIREX) , Texas Instruments, ***http://bit.ly/2HHDqqo***
11. Command Line Example Application, CC13x2 SDK v.2.20, ***http://bit.ly/2R4Jo9q***
12. SimpleLink CC13x2 TI Thread User's Guide, TI-OpenThread Stack Overview, v1.02.00.00, ***http://bit.ly/2yTcb9j***
13. EFR32 Mighty Gecko Mesh Networking Wireless SoCs for Zigbee and Thread, Silicon Labs, ***http://bit.ly/2JYgB39***
14. An open foundation for the connected home., ***http://bit.ly/2yspwWG***
15. Thread Group, ***http://bit.ly/2CXYHgP***
16. Packet Sniffer, Setting Up Wireshark, Texas Instrumets, ***http://bit.ly/2q5cZUt***

Pokazany został przykład pracy sieci Thread z metodą (out-of--band) bezpośredniego przekazywania danych poufnych. Ilustruje to jeden ze sposobów zabezpieczenia przed atakiem MITM (Man in the Middle protection) – zabezpieczenia przed atakiem poprzez podsłuchanie transmisji radiowej.

Zaprezentowany przykład bezpiecznego dołączania węzła do sieci TI-OpenThread pokazuje, że w tej implementacji standardu OpenThread to działa, ale nie do końca. Jednak oprogramowanie ukazało się niedawno. Należy też pamiętać, że nowe moduły LaunchPad i pakiety programowe SDK są testowane z przedprodukcyjną wersją (PG1.1) układów scalonych CC1352R1.

Prób z wykorzystaniem układu scalonego CC2652R1 nie udało się wykonać, ponieważ do dyspozycji były tylko moduły z bardzo wczesną wersją PG1.0 układu scalonego. Nie pracuje ona z obecną wersją pakietu programowego CC26x2 SDK.

Powyższy opis jest tylko pierwszym wprowadzeniem do użytkowania sieci Thread. W pakiecie programowym CC13x2 SDK jest jeszcze siedem innych bardzo ciekawych projektów przykładowych.

Możliwość podglądania transmisji w sieci Thread z zastosowaniem aplikacji SmartRF Protocol Packet Sniffer 2 oraz aplikacji Wireshark jest bardzo dobrą zapowiedzią uzyskania bardzo silnego narzędzia uruchamiania komunikacji z siecią Thread. Po dalszym rozwoju stosu TI-OpenThread trzeba będzie wrócić do dokładniejszych prób.

Henryk A. Kowalski
Instytut Informatyki
Politechnika Warszawska