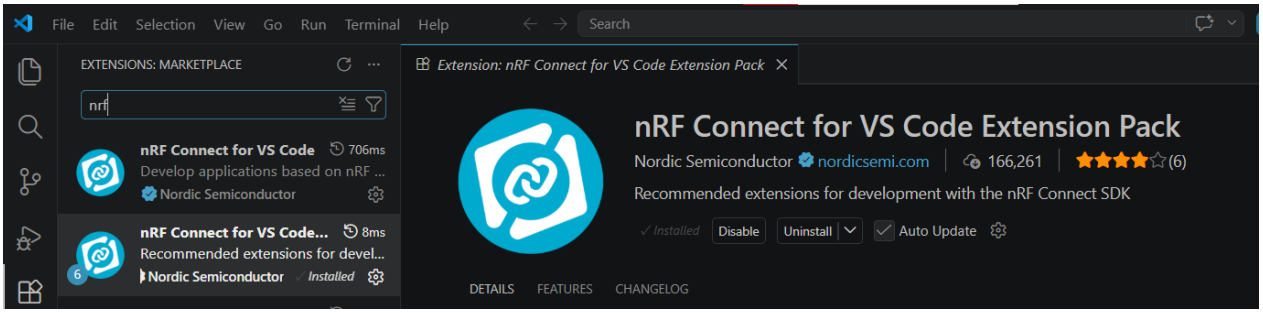
Projekt nRF NFC

Pawel Kasprzyk

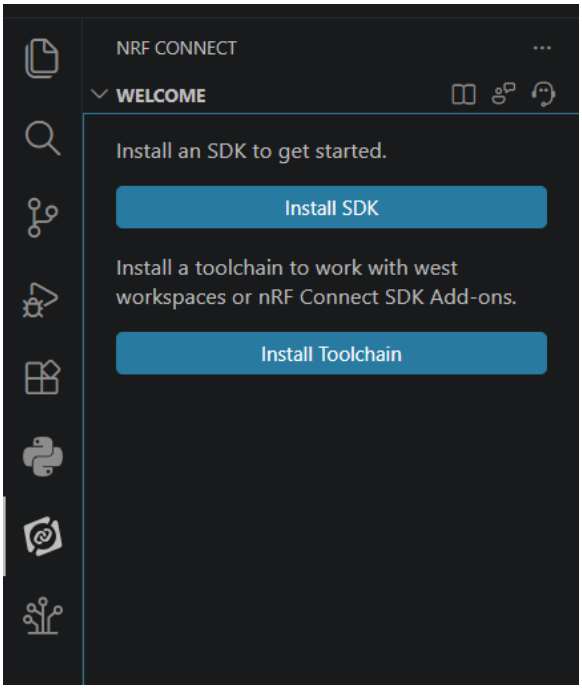
June 2026

1 Jak przygotować środkowisko programistyczne nRF Connect SDK + Visual Studio Code

Po wcześniejszym pobraniu Visual Studio Code należy skierować się do zakładki z wtyczkami oraz wyszukać paczkę wtyczek (nRF Connect for VS Code Extension Pack).



Zainstalowana paczka oferuje pełen zestaw narzędzi do programowania, debugowania oraz kompilacji kodu na mikrokontrolery z rodziny nRF. Następnym krokiem jest zainstalowanie SDK w wybranej wersji.

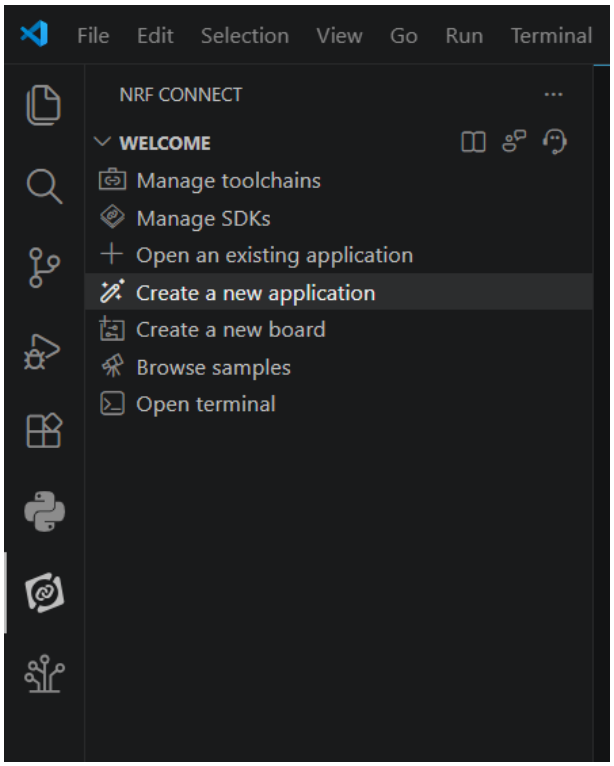


Należy:

1. Przejść do menu wtyczki nRF
2. kliknąć na przycisk install SDK
3. z listy kontekstowej na górze Visual Studio Code wybrać wersję SDK
4. projekt wykonywany był na wersji sdk v3.2.4, nowsze wersje sdk najprawdopodobniej też będą działały
5. program zapyta o lokalizację SDK, najlepiej zostawić domyślną

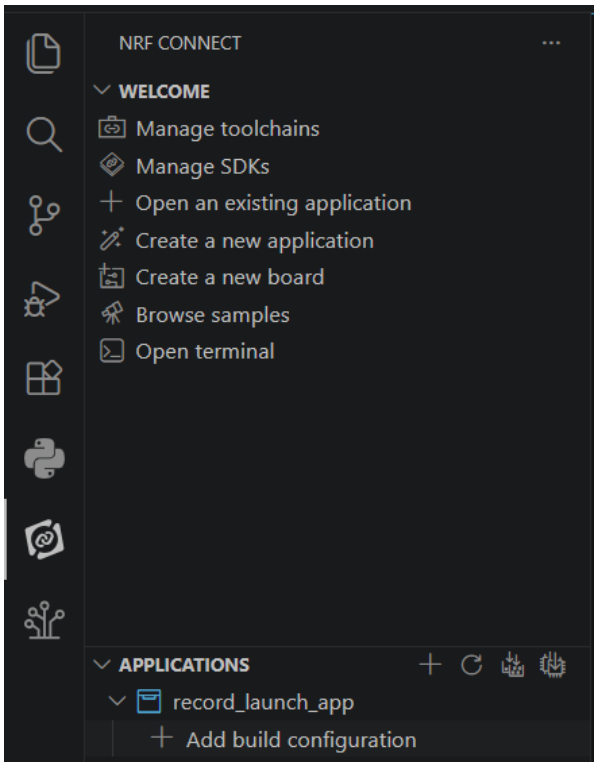
Instalacja SDK trwa długo (5-10 minut). Wraz z instalacją SDK instaluje się również odpowiedni ToolChain. Miejsce instalacji SDK jak i samego projektu (o czym w dalszej części rozdziału) warto zostawić domyślną jeżeli korzystamy z systemu Windows. Windows posiada limit 250 znaków długości ścieżki dostępu. Przekroczenie limitu skutkuje problemami w odnalezieniu plików źródłowych bibliotek zawartych w SDK oraz plików źródłowych naszego projektu. Windows pozwala na zniesienie tego limitu poprzez modyfikację wartości w rejestrze systemu, natomiast podczas wykonywania projektu nie rozwiązało to opisywanego problemu. Dlatego warto zachowywać krótkie ścieżki.

Gdy zainstalujemy potrzebne zasoby menu wtyczki zacznie wyświetlać skróty do właściwych funkcji. By stworzyć projekt programu należy wybrać "Create new application". Następnie, z listy umieszczonej na górze ekranu, wybrać opcję "Copy Sample".

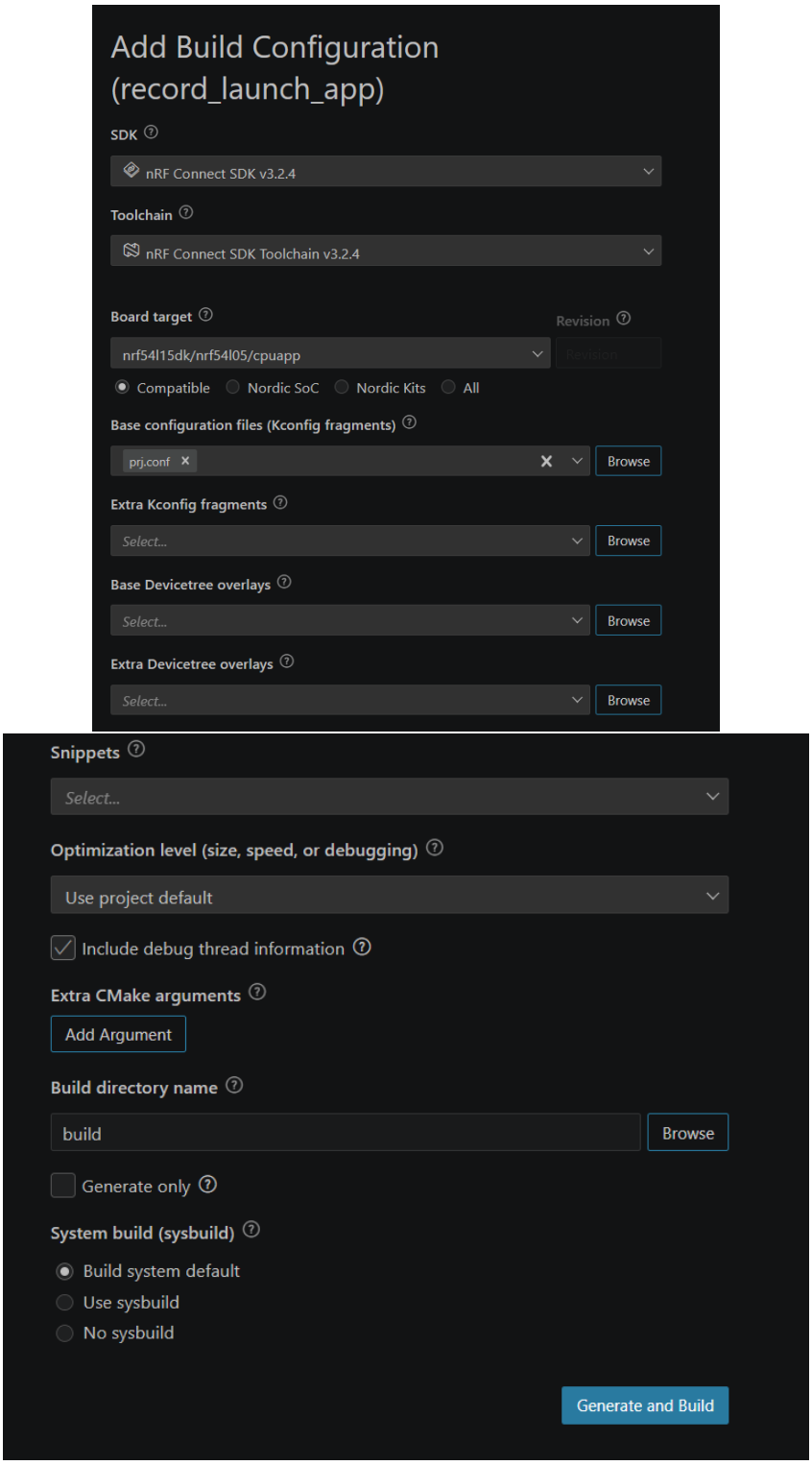


W tym miejscu możemy również wybrać opcję, by zacząć projekt od początku. Jako początkujący z platformą nRF warto wybrać copy sample ponieważ otrzymujemy kompletną kopię projektu która skonfigurowana jest sprzętowo (skonfigurowany plik overlay zawierający informację o wykorzystanych peryferiach mikrokontrolera (np. UART, GPIO, NFC)) i programowo (kod aplikacji korzystająca z wybranego peryferium). By wybrać omawianą w projekcie funkcję NFC należy wybrać przykład "NFC Launch App". Potwierdzamy ścieżkę oraz klikamy "yes i trust the authors"by przejść do skopiowanego kodu.

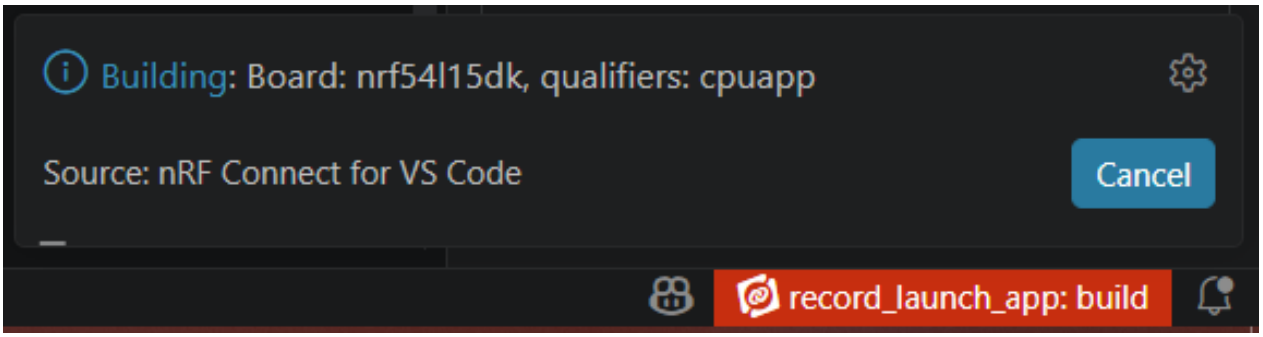
W tym momencie widzimy plik źródłowy naszego programu main.c oraz całą strukturę projektu w zakładce Explorer (Ctrl+Shift+E). By skompilować program oraz wgrać na płytkę wystarczą dwa kroki.



Konfiguracja kompilatora znajduje się w rozwijanej sekcji wtyczki nRF Connect "Applications". W niej klikamy na "Add build configuration"oraz uzupełniamy kilka rubryk jak na poniższych zrzutach ekranu.



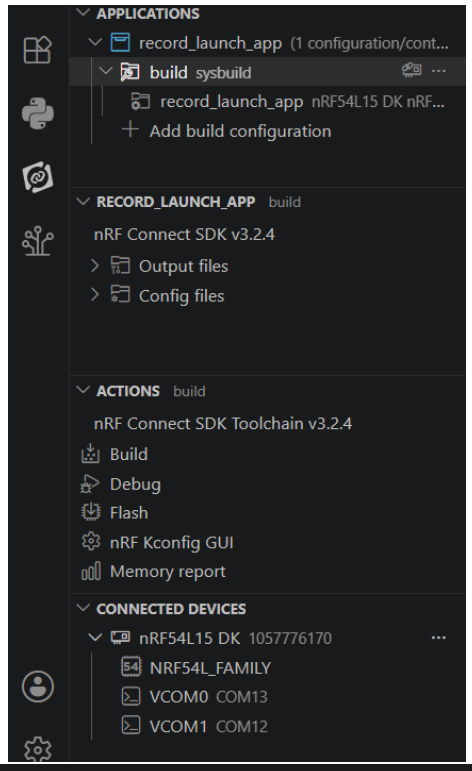
Następnie możemy skompilować program (zbudować) klikając na "build"w rozwijanej sekcji "Actions". Pierwsze budowanie trwa długo ze względu na stworzenie wszystkich plików wyjściowych oraz stworzeniu powiązań z SDK. Proces możemy podejrzeć klikając na niebieski napis "Building"w powiadomieniu w dolnym prawym rogu ekranu.

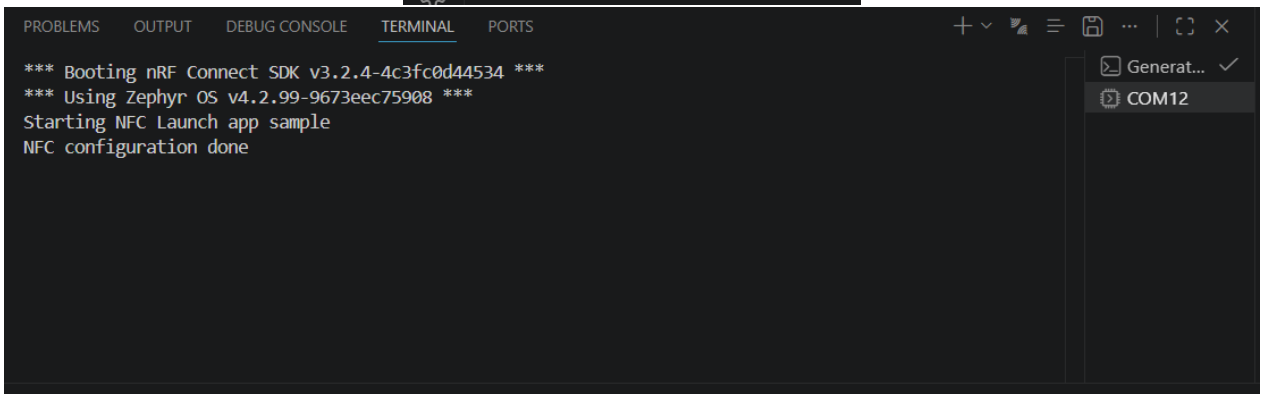


Następne budowania projektu trwają kilkanaście sekund. Podczas tworzenia aplikacji w większości przypadków wystarczy zlecić zwykłe polecenie build. W momencie gdy zmieniamy wykorzystywane biblioteki, dokonujemy zmian w konfiguracji sprzętowej, warto wykonać "pristine build"(zakręcona strzałka która ukazuje się po najechaniu kursorem na przycisk build).

Gdy wszystko poszło dobrze nasz kod jest skompilowany i gotowy do wgrania poprzez przycisk "Flash"w rozwijanej sekcji "Actions".

Sekcja "Connected Devices"wyświetla wszystkie aktualnie podłączone płytki nRF wraz z opcją monitoru portu szeregowego płytki (dla programu omawianego projekcie jak i większość przykładowych programów od Nordic'a należy otworzyć port VCOM1)





2 Omówienie techniki stojącej za emulacją tagu w mikrokontrolerach nRF

Mikrokontrolery firmy Nordic Semiconductor (z serii nRF) wyposażone są w dedykowany peryferial **NFCT**, który umożliwia sprzętową emulację tagów NFC. Poniższe omówienie szczegółowo opisuje architekturę, mechanizmy zarządzania energią oraz obsługę danych, które pozwalają tym układom działać jako urządzenia nasłuchujące (*listening devices*) w standardzie NFC-A.

2.1 Charakterystyka i parametry radiowe peryferialu NFCT

Komponent NFCT stanowi pełną implementację urządzenia nasłuchującego, zgodnego ze specyfikacją NFC Forum dla technologii **NFC-A**. Peryferial ten integruje w sobie kluczowe elementy warstwy fizycznej (PHY), co pozwala na bezpośrednie podłączenie anteny NFC do dedykowanych pinów mikrokontrolera.

Do najważniejszych parametrów radiowych i fizycznych należą:

- **Częstotliwość pracy:** Układ operuje na standardowej częstotliwości **13.56 MHz** (częstotliwość wejściowa fali nośnej generowanej przez czytnik).

- **Prędkość transmisji:** Transfer danych odbywa się z prędkością **106 kbps**, zdefiniowaną przez standard NFC-A.
- **Odbiornik:** Zintegrowany odbiornik AM (amplitudowy) o częstotliwości 13.56 MHz do odbierania zapytań od urządzeń odpytujących (*polling devices/readers*).
- **Nadajnik:** Zintegrowany modulator pracujący na częstotliwości 13.56 MHz, służący do przesyłania odpowiedzi do czytnika.

2.2 Architektura transmisji danych i mechanizm EasyDMA

W celu zminimalizowania obciążenia rdzenia procesora (CPU) podczas krytycznych czasowo operacji radiowych, peryferial NFCT wykorzystuje bezpośredni dostęp do pamięci RAM za pomocą mechanizmu **EasyDMA**.

Odbiór danych (RX)

Gdy czytnik wysyła ramkę danych, odbiornik AM rejestruje sygnał, a peryferial NFCT automatycznie przeprowadza proces demontażu ramki (*frame disassembly*). Proces ten jest konfigurowany i kontrolowany za pomocą rejestru `RXD.FRAMECONFIG`. Po wyodrębnieniu czystych danych z ramki NFC-A, mechanizm EasyDMA zapisuje sekcję danych bezpośrednio do pamięci RAM. Po zakończeniu poprawnego odbioru całej ramki, system jest powiadamiany dedykowanym zdarzeniem (*event*).

Nadawanie danych (TX)

W przypadku wysyłania odpowiedzi, dane są pobierane bezpośrednio z pamięci RAM za pomocą EasyDMA. Następnie układ sprzętowo dokonuje montażu ramki (*frame assembly*) zgodnie z regułami zapisanymi w rejestrze `TXD.FRAMECONFIG`. Parametry takie jak typ ramki oraz opóźnienie czasu nadawania są w pełni konfigurowalne. Układ generuje zdarzenie systemowe natychmiast po zakończeniu transmisji kompletnej ramki.

> **Wsparcie dla integralności danych i logiki transmisji:**
> Peryferial NFCT posiada wbudowane, automatyczne funkcje obliczania sumy kontrolnej **CRC** (*Cyclic Redundancy Check*) oraz kontroli parzystości (*parity*). Dodatkowo zaimplementowano sprzętowy, automatyczny mechanizm **rozwiązywania kolizji (anticollision)**, wymagany przez NFC Forum, co pozwala na poprawną identyfikację tagu w obecności innych urządzeń NFC.

2.3 Zarządzanie energią i tryby detekcji pola (Wake-on-field)

Zaletą implementacji NFC ma platforie nRF jest system niskiego poboru prądu oparty na detekcji obecności pola magnetycznego czytnika, znany jako **SENSE mode** (*Wake-on-field*). Funkcja ta pozwala na oszczędzanie energii i może działać w dwóch głównych stanach zasilania mikrokontrolera:

Tryb System ON

Gdy mikrokontroler znajduje się w trybie uśpienia z włączonym zasilaniem, moduł NFCT monitoruje poziom energii indukowanej w antenie.

- Jeśli energia przekroczy określony próg (zdefiniowany w specyfikacji elektrycznej NFCT), generowane jest zdarzenie `FIELDDETECTED`. Informuje ono system o konieczności aktywacji pełnej funkcjonalności NFCT do obsługi nadchodzących ramek.
- W momencie, gdy czytnik zostanie oddalony i siła pola spadnie poniżej progu uniemożliwiającego komunikację, generowane jest zdarzenie `FIELDLOST`.

Tryb System OFF

W trybie głębokiego uśpienia, funkcja *Low Power Field Detect* potrafi wybudzić cały mikrokontroler poprzez wygenerowanie **resetu systemowego**.

- Źródło wybudzenia można zidentyfikować po restarcie w rejestrze `RESETREAS`.
- Konsekwencją resetu jest wyłączenie peryferialu NFCT. Z tego powodu programista w procedurze obsługi resetu (*reset handler*) musi ponownie aktywować moduł NFCT i odpowiednio go skonfigurować.
- Jeśli system zostanie wprowadzony w tryb System OFF w momencie, gdy pole magnetyczne jest już obecne, funkcja detekcji natychmiast wybudzi układ, generując reset.

W celu zapewnienia wymogów czasowych narzuconych przez specyfikację NFC Forum, NFCT zawiera programowalny kontroler czasu ramek *(frame timing controller)*. Pozwala on na kontrolę czasu opóźnienia między ramką przychodzącą od czytnika (*inter-frame delay*) a odpowiadającą jej ramką wychodzącą z mikrokontrolera.

3 Wsparcie programistyczne dla NFC - ramki NDEF, przykładowe implementacje tagów i aplikacji

3.1 Architektura oprogramowania NFC w nRF Connect SDK (NCS)

Narzędzia programistyczne dostarczane przez Nordic Semiconductor w ramach ekosystemu nRF Connect SDK (opartego na systemie operacyjnym czasu rzeczywistego Zephyr RTOS) w pełni izolują programistę od niskopoziomowej manipulacji rejestrami peryferialu NFCT. Wsparcie programistyczne zostało zorganizowane modułowo, dzieląc zadania na warstwę obsługi protokołu sprzętowego (tagów konkretnego typu) oraz warstwę logiczną formatowania danych.

W zależności od wymagań projektu, programista konfiguruje stos za pomocą makr systemu Kconfig (np. `CONFIG_NFC_T2T_NRFXLIB` lub `CONFIG_NFC_T4T_NRFXLIB`), co automatycznie dołącza odpowiednie biblioteki:

- **Biblioteka Type 2 Tag (`libnfct2t_nrf52`):** Obsługuje emulację tagu typu 2. Jest prostsza, lżejsza i idealnie nadaje się do udostępniania statycznych informacji lub prostych operacji odczytu/zapisu.
- **Biblioteka Type 4 Tag (`libnfct4t_nrf52`):** Implementuje emulację tagu typu 4. Posiada wielopoziomową strukturę wymiany danych, obsługę APDU (*Application Protocol Data Unit*) oraz realizację protokołów komunikacyjnych.

Uruchomienie obsługi NFC z poziomu kodu aplikacji sprowadza się do rejestracji funkcji zwrotnej (*callback*) dedykowanej dla zdarzeń warstwy fizycznej (np. wykrycie pola `NFC_T2T_EVENT_FIELD_ON` lub jego utrata `NFC_T2T_EVENT_FIELD_OFF`). Biblioteki te automatycznie przejmują zarządzanie zegarem wysokiej częstotliwości (HFCLK), aktywując go tylko w momentach ważnych dla transmisji radiowej.

3.2 Obsługa formatu NDEF (NFC Data Exchange Format)

Warstwa wymiany danych opiera się na uniwersalnym standardzie **NDEF (NFC Data Exchange Format)**, zdefiniowanym przez NFC Forum. Ekosystem NCS dostarcza zaawansowaną bibliotekę do tworzenia, parsowania oraz kodowania komunikatów NDEF (`CONFIG_NFC_NDEF`).

Struktura danych NDEF w nRF Connect SDK składa się z dwóch głównych komponentów programistycznych:

1. **Rekord NDEF (`nfc_ndef_record`):** Pojedyncza porcja informacji zawierająca typ rekordu, unikalne ID oraz *payload*.
2. **Komunikat NDEF (`nfc_ndef_msg`):** Kontener grupujący jeden lub więcej rekordów w logiczną całość, przygotowaną do transmisji.

Biblioteka NCS oferuje gotowe makra i funkcje pomocnicze do natychmiastowego generowania standardowych typów rekordów:

- **Text Record (Rekord tekstowy):** Służy do udostępniania ciągów znaków (np. informacji o urządzeniu). Wykorzystuje deskryptory typu `nfc_ndef_text_record_desc`, gdzie definiuje się kod języka (np. „pl", „en") oraz treść.
- **URI Record (Rekord adresu URL):** Pozwala na automatyczne przekierowanie urządzenia odpytującego (np. smartfona) do określonej witryny internetowej, zasobu sieciowego lub wywołania akcji systemowej (np. `tel:`, `mailto:`).
- **Launch App Record (Rekord uruchamiania aplikacji):** Wykorzystuje specjalny format pakietu (w przypadku systemu Android jest to *Android Application Record* – AAR). Przy użyciu funkcji takich jak `nfc_launchapp_msg_encode`, programista może zakodować w tagu nazwę pakietu aplikacji (np. `com.example.myapp`). Smartfon po zbliżeniu do anteny automatycznie uruchomi tę aplikację, a jeśli nie jest zainstalowana – otworzy sklep Google Play na stronie pobierania.

Proces przygotowania danych (Workflow):

Tworzenie komunikatu przebiega według ściśle określonego schematu:

[Definicja Rekordów] → [Powiązanie z Komunikatem] → `nfc_ndef_msg_encode()` → [Bufor w RAM] → [Przekazanie do biblioteki Tagu]

Po zakodowaniu wiadomości do binarnego bufora w pamięci RAM, długość i wskaźnik na dane są przekazywane do peryferialu za pomocą funkcji typu `nfc_t2t_payload_set()`. Od tej pory układ sprzętowy nRF przy pomocy mechanizmu EasyDMA samodzielnie odpowiada na żądania czytnika.

3. Przykładowe implementacje tagów (NFC Samples)

W dokumentacji nRF Connect SDK v2.5.0 kluczowym elementem wsparcia są gotowe szablony implementacji (*samples*), które pokazują praktyczną integrację peryferialu z logiką biznesową urządzenia.

- **NFC: Text Record & URI Record:** Podstawowe przykłady demonstrujące konfigurację pasywnego tagu. Po zainicjalizowaniu stosu urządzenie przechodzi w stan uśpienia, a transmisja danych odbywa się w pełni autonomicznie w momencie zbliżenia telefonu.
- **NFC: Writable NDEF Message (Zapisywalna wiadomość NDEF):** Implementacja wykorzystująca emulację tagu typu 4 w trybie odczytu i zapisu (*Read-Write emulation mode*). Przykład ten pokazuje, jak urządzenie zewnętrzne może nadpisać wiadomość NDEF przechowywaną w mikrokontrolerze. Stos programistyczny odbiera strumień danych, weryfikuje poprawność struktur NDEF, generuje zdarzenie aktualizacji danych, a aplikacja zapisuje nową treść w nieulotnej pamięci flash mikrokontrolera.
- **NFC: System OFF:** Kluczowa implementacja z punktu widzenia systemów zasilanych bateryjnie. Pokazuje architekturę, w której mikrokontroler zostaje wprowadzony w tryb głębokiego uśpienia. Zbliżenie smartfona generuje indukcję prądu w antenie, co sprzętowo wybudza mikrokontroler (*Wake-on-field*), uruchamia funkcję `reset_handler`, inicjalizuje peryferial i w ułamku sekundy udostępnia dane NDEF urządzeniu mobilnemu.

4. Zaawansowane aplikacje i komunikacja dwukierunkowa

Poza standardową emulacją prostego tagu pasywnego, wsparcie programistyczne obejmuje zaawansowane scenariusze, w których mikrokontroler nRF przyjmuje aktywną rolę lub realizuje wymianę informacji:

- **NFC: Tag Reader (Czytnik Tagów):** Przykładowa implementacja, w której mikrokontroler zmienia swoją rolę z urządzenia nasłuchującego (*listening device*) na urządzenie odpytujące (*polling device*). Pozwala to układowi nRF na aktywne generowanie pola 13.56 MHz (o ile pozwala na to architektura sprzętowa lub zewnętrzny kontroler) i odczytywanie danych z innych fizycznych tagów zbliżeniowych.

- **NFC: TNEP (Tag NDEF Exchange Protocol):** Protokół TNEP wprowadza mechanizm bezstanowej i stanowej dwukierunkowej wymiany danych pomiędzy tagiem a aplikacją na smartfonie. W dokumentacji NCS znajdują się implementacje zarówno dla roli *TNEP poller*, jak i *TNEP tag*. Protokół ten pozwala na realizację skomplikowanych procedur, takich jak:
  - **OOB (Out-of-Band) Pairing:** Bezpieczne przesyłanie kluczy kryptograficznych i parametrów połączenia w celu błyskawicznego sparowania interfejsu Bluetooth Low Energy (BLE) lub sieci Wi-Fi (tzw. *NFC Connection Handover*).
  - **Wymiana telemetryczna:** Przesyłanie sparametryzowanych danych konfiguracyjnych i odczytów z czujników w ramach jednej sesji przyłożenia telefonu, bez konieczności nawiązywania pełnego połączenia radiowego innych typów.

4 Analiza kodu programu

Układ peryferyjny **NFCT** (NFC Management Tag) wbudowany w mikrokontrolery z serii nRF (np. nRF52, nRF53) firmy Nordic Semiconductor umożliwia emulację pasywnego tagu NFC pracującego w standardzie *NFC Forum Department Devices*. Omawiany kod realizuje funkcjonalność znacznika **NFC Forum Type 2 Tag**, którego celem jest przekazanie do urządzenia odczytującego (np. smartfona) sformatowanej wiadomości **NDEF** (*NFC Data Exchange Format*).

Wiadomość ta należy do typu **Launch App**. Po zbliżeniu telefonu do anteny NFC, system operacyjny automatycznie rozpoznaje pakiet danych i uruchamia dedykowaną aplikację mobilną (*nRF Toolbox*), bądź przekierowuje użytkownika do sklepu z aplikacjami, jeśli nie jest ona zainstalowana.

4.1 Ogólny algorytm działania programu

Program realizowany jest w środowisku RTOS **Zephyr** przy użyciu bibliotek nRF Connect SDK (NCS). Główny przepływ programu w funkcji `main()` przebiega następująco:

1. Inicjalizacja peryferiów płytki deweloperskiej (diody LED).
2. Konfiguracja i inicjalizacja biblioteki *NFC Type 2 Tag* wraz z rejestracją funkcji zwrotnej (*callback*).
3. Zakodowanie danych aplikacji mobilnej (nazwa pakietu Android oraz Universal Link dla iOS) do formatu binarnego NDEF w przygotowanym buforze.
4. Załadowanie gotowej wiadomości jako *payload* dla emulowanego tagu.
5. Uruchomienie emulacji oraz nasłuchiwania na zewnętrzne pole magnetyczne o częstotliwości 13,56 MHz.

4.2 Szczegółowa analiza kodu źródłowego

4.2.1 Dyrektywy preprocesora i stałe globalne

W kodzie zdefiniowano rozmiar bufora danych `NDEF_MSG_BUF_SIZE` na wartość 256 bajtów, co stanowi bezpieczną przestrzeń dla przechowywania wiadomości uruchomieniowej. Makro `NFC_FIELD_LED` przypisuje pierwszą diodę LED zestawu deweloperskiego (`DK_LED1`) jako wskaźnik obecności pola magnetycznego czytnika NFC.

Tablice `android_pkg_name[]` oraz `universal_link[]` przechowują surowe ciągi znaków (reprezentowane jako tablice bajtów `uint8_t`):

- `no.nordicsemi.android.nrftoolbox` – unikalny identyfikator aplikacji nRF Toolbox w systemie Android.
- `nrf-toolbox://main/` – adres URI odnośnika uniwersalnego obsługiwanego przez systemy iOS.

4.2.2 Funkcja obsługi zdarzeń (nfc_callback)

Funkcja `nfc_callback` odpowiada za asynchroniczną obsługę zdarzeń generowanych przez kontroler NFCT:

- `NFC_T2T_EVENT_FIELD_ON`: Generowane, gdy zewnętrzny czytnik (np. smartfon) zbliży się do anteny i indukuje odpowiednie napięcie. Następuje zapalenie diody informacyjnej za pomocą `dk_set_led_on()`.
- `NFC_T2T_EVENT_FIELD_OFF`: Generowane w momencie utraty sygnału (oddalenie telefonu). Dioda zostaje zgaszona przez `dk_set_led_off()`.

Zastosowanie makra `ARG_UNUSED()` zapobiega generowaniu ostrzeżeń kompilatora o niewykorzystanych argumentach funkcji w ciele handlera.

4.3 Opis funkcji z bibliotek NFC

4.3.1 nfc_t2t_setup

Sygnatura:

```
int nfc_t2t_setup(nfc_t2t_callback_t callback, void *context);
```

Opis: Funkcja ta służy do początkowej konfiguracji biblioteki emulacji *NFC Type 2 Tag*. Rejestruje ona funkcję zwrotną (`callback`), która będzie wywoływana przez sterownik niskiego poziomu w momencie wykrycia zdarzeń związanych z warstwą fizyczną i protokołem transmisji (np. pojawienie pola, odczyt sektorów przez czytnik). Drugi parametr pozwala na przekazanie wskaźnika do kontekstu użytkownika (w kodzie przekazano `NULL`). Zwraca wartość 0 w przypadku sukcesu lub ujemny kod błędu.

4.3.2 nfc_launchapp_msg_encode

Sygnatura:

```
int nfc_launchapp_msg_encode(const uint8_t *android_package_name,
                             uint32_t android_package_name_len,
                             const uint8_t *universal_link,
                             uint32_t universal_link_len,
                             uint8_t *ndef_msg_buf,
                             uint32_t *ndef_msg_len);
```

Opis: Wysokopoziomowa funkcja pochodząca z modułu `launchapp_msg`. Odpowiada za automatyczne wygenerowanie struktury danych zgodnej ze specyfikacją **NDEF**. Formatuje ona rekordy typu *Android Application Record (AAR)* oraz rekordy *URI* powiązane z *Universal Links* dla systemów Apple. Funkcja przyjmuje wskaźniki na zdefiniowane ciągi znaków identyfikujące aplikację, ich długości oraz wskaźnik na bufor docelowy `ndef_msg_buf`, w którym zapisana zostanie binarna postać wiadomości. Parametr `ndef_msg_len` przekazuje wejściowy rozmiar bufora, a po zakończeniu działania funkcji zwraca rzeczywistą liczbę zapisanych bajtów.

4.3.3 nfc_t2t_payload_set

Sygnatura:

```
int nfc_t2t_payload_set(const uint8_t *payload, size_t payload_length);
```

Opis: Funkcja ta mapuje przygotowany wcześniej bufor danych (zawierający binarną postać wiadomości NDEF) jako główny payload znacznika Type 2 Tag. Od tego momentu struktura danych przygotowana przez enkoder jest gotowa do przesłania drogą radiową. Każda operacja odczytu (komenda `READ` standardu NFC T2T) zainicjalizowana przez smartfon zwróci fragmenty tego bufora.

4.3.4 nfc_t2t_emulation_start

Sygnatura:

```
int nfc_t2t_emulation_start(void);
```

Opis: Uruchamia właściwą emulację sprzętową. Powoduje włączenie peryferium NFCT wewnątrz SoC nRF. Mikrokontroler przechodzi w tryb oszczędnego nasłuchiwania fali elektromagnetycznej emitowanej przez aktywne urządzenia NFC. Od tego momentu układ reaguje na bodźce zewnętrzne ze strony czytników.

4.4 Obsługa błędów i mechanizm awaryjny

Program posiada sekwencyjny mechanizm kontroli błędów realizowany przy użyciu instrukcji skoku `goto fail;`. W przypadku niepowodzenia którejkolwiek z operacji inicjalizacji (zwrócenie wartości niezerowej przez funkcje API), sterowanie zostaje przekazane do sekcji awaryjnej.

Jeżeli w konfiguracji projektu aktywne jest makro `CONFIG_REBOOT` (pochodzące z systemu operacyjnego Zephyr), wywoływana jest funkcja:

```
sys_reboot(SYS_REBOOT_COLD);
```

Wykonuje ona tzw. *Cold Boot*, czyli pełny restart sprzętowy procesora. Rozwiązanie to zapobiega pozostaniu urządzenia w stanie zawieszenia (np. z powodu błędu pamięci lub awarii magistrali sprzętowej) w systemach wbudowanych pracujących autonomicznie.