Urządzenia peryferyjne Laboratorium 3 – Bluetooth	
Michał Lewandowski, Dominik Kilijan Grupa F	Czwartek 17:30 – 19:30 TNP 26.10.2023

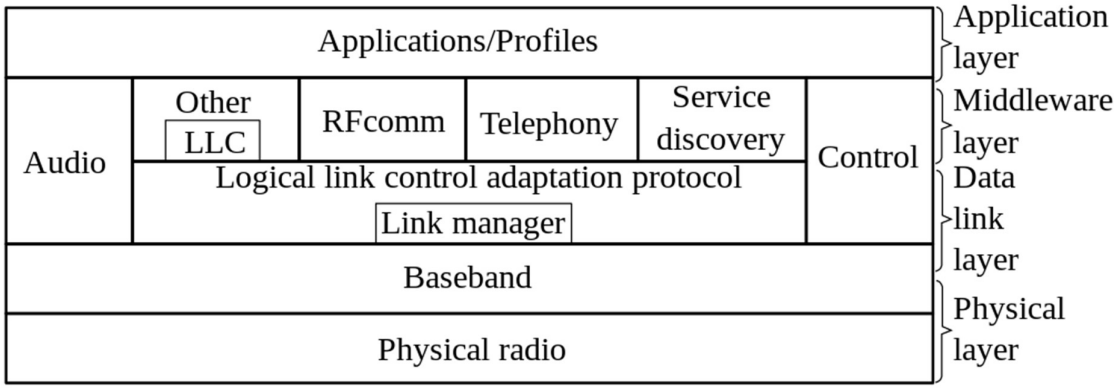
1 Wstęp teoretyczny

1.1 Bluetooth

Bluetooth – standard bezprzewodowej komunikacji krótkiego zasięgu pomiędzy różnymi urządzeniami elektronicznymi, takimi jak klawiatura, komputer, laptop, palmtop, smartfon i wieloma innymi.

Jest to otwarty standard opisany w specyfikacji IEEE 802.15.1. Jego specyfikacja techniczna obejmuje trzy klasy mocy nadawczej ERP 1-3 o zasięgu 100, 10 oraz 1 metra w otwartej przestrzeni. Najczęściej spotykaną klasą jest klasa druga. Standard korzysta z fal radiowych w paśmie częstotliwości ISM 2,4 GHz.

1.2 Zasady realizacji transmisji



1.2.1 Warstwa radiowa

Warstwa ta odpowiedzialna jest za transport danych od urządzenia master do slave i vice versa. Jest to system o małym poborze mocy, działający w zależności od klasy na różnych zasięgach, operujący w paśmie częstotliwości ISM 2,4 GHz. Pasmo jest podzielone na 79 kanałów, po 1 MHz każdy. System wykorzystuje modulację FSK (Frequency Shift Keying), dając prędkości transmisji 1 Mbit/s, jednak duża część tego

widma jest zajęta przez nagłówek. Aby przydzielić kanały sprawiedliwie, wykorzystuje się skakanie częstotliwości (1600 skoków na sekundę). Sekwencję skoków dyktuje węzeł master. Systemy 802.11 oraz Bluetooth operują na tych samych częstotliwościach z takim samym podziałem pasma na 79 kanałów. Z tego powodu zakłócają się wzajemnie. Ponieważ skoki częstotliwości są znacznie szybsze w systemie Bluetooth, jest znacznie bardziej prawdopodobne, że system ten będzie zakłócał transmisje w 802.11. Gdy nowe urządzenie szuka sieci do której mogłoby się przyłączyć skacze po kanałach w sposób losowy. Połączenie zostaje nawiązane, dopiero gdy trafi na ten sam kanał co master. Algorytm wybierania kanału przez mastera jest inny niż algorytm przełączania kanału przez slave'a szukającego mastera. Oba algorytmy opierają się na losowości co powoduje, że nigdy nie ma pewności po jakim czasie slave znajdzie mastera.

1.2.2 Baseband layer

Warstwa ta jest zbliżona do podwarstwy MAC modelu OSI. Opakowuje ona luźne bity w ramki. Master w każdej pikosieci definiuje sloty czasowe o długości 625 μs. Transmisja mastera zaczyna się od slotów parzystych natomiast transmisja slave od slotów nieparzystych. Jest to tradycyjna multipleksacja w dziedzinie czasu (TDM), gdzie master zajmuje połowę slotów, a slave pozostałą ich część. Ramki mogą mieć długość jednego, trzech lub pięciu slotów czasowych. Skakanie częstotliwości pozwala ustawić czas 250 – 260 μs na skok, aby umożliwić stabilizacje układów radiowych. Dla ramki składającej się z jednego slotu czasowego, po tym czasie, zostaje 366 z 625 bitów. 126 z nich zawierają kod dostępu oraz nagłówek, pozostałe 240 są dla danych. Gdy ramka składa się z pięciu slotów, tylko jeden okres stabilizacji jest wymagany i dla warstwy baseband pozostaje 2781 bitów, więc dłuższe ramki są znacznie bardziej efektywne niż ramki zbudowane z jednego slota czasowego. Każda ramka jest transmitowana przez kanał logiczny, nazywany z angielskiego link, pomiędzy masterem i urządzeniem slave. Istnieją dwa rodzaje kanałów logicznych. Pierwszy nazywa się ACL (Asynchronous Connection-Less), używany w połączeniu z komutacją pakietów, gdzie dane są dostępne w nieregularnych odstępach czasu. Dane te pochodzą od warstwy L2CAP po stronie nadawczej i są dostarczane do warstwy L2CAP po stronie odbiorczej.

1.3 Zasady nawiązywania połączenia

1.3.1 Skanowanie Urządzeń (Inquiry):

- Urządzenie Wysyłające (Inquiring Device): rozpoczyna proces skanowania w poszukiwaniu dostępnych urządzeń w swoim zasięgu.
- Urządzenie Odpowiadające (Responding Device): urządzenia, które są w trybie dostępności, odpowiadają na sygnał zapytania (inquiry) od urządzenia wysyłającego.

1.3.2 Ustanawianie Połączenia (Paging):

- Urządzenie Wysyłające (Paging Device): po zakończeniu procesu skanowania, urządzenie wysyłające wysyła sygnał próbujący nawiązać połączenie z wybranym urządzeniem odpowiadającym.
- Urządzenie Odpowiadające (Responding Device): odpowiada na sygnał próbujący nawiązać połączenie. W tym momencie, oba urządzenia wiedzą, że nawiązały połączenie.

1.3.3 Parowanie (Pairing):

- Następnie urządzenia muszą być sparowane. Podczas parowania urządzenia wymieniają klucze dostępu Bluetooth (czasem jest to PIN lub automatycznie generowany klucz), który zapewnia bezpieczne połączenie między nimi. Proces ten może wymagać akceptacji na obu urządzeniach.

1.3.4 Nawiązywanie Połączenia (Connection Establishment):

- Po sparowaniu urządzenia nawiązują połączenie. W przypadku, gdy jedno z urządzeń jest Masterem, a drugie Slave'em, Master inicjuje połączenie. Połączenie to może być jednorazowe (transmisja jednorazowa) lub stałe (na przykład w przypadku słuchawek i smartfonów).
- Po nawiązaniu połączenia urządzenia mogą zaczynać wymieniać dane. Mogą to być pliki, dźwięk, obrazy, komunikaty tekstowe lub inne rodzaje danych, w zależności od zastosowania.

1.3.5 Zakończenie Połączenia:

- Rozłączanie Połączenia: Po zakończeniu komunikacji, urządzenia mogą zakończyć połączenie.
- Rozładowanie Połączenia: Urządzenia mogą być ponownie użyte w procesie skanowania i nawiązywania nowych połączeń z innymi urządzeniami.

1.4 Obex

Object Exchange – protokół komunikacyjny, określający procedury wymiany danych binarnych między urządzeniami. Rozwojem i utrzymaniem specyfikacji zajmuje się Infrared Data Association. Protokół jest używany do przesyłania danych w takich technologiach jak: Bluetooth, USB czy RS232. Specyfikacja protokołu OBEX opiera się na architekturze klient – serwer. Klient wykorzystuje zaufane medium transportowe do połączenia się z serwerem w celu zażądania transmisji obiektów. Przesyłane obiekty są zapisane w formacie binarnym

2 Przebieg zadania

Celem zadania było napisanie programu, który odszuka i połączy się z telefonem, a następnie przesłane zostaną wybrane pliki. Program napisany został w języku C++. Z korzystano z takich bibliotek jak:

- Winsock2 – biblioteka umożliwiająca tworzenie socketów dla połączeń sieciowych
- Ws2bth – rozszerzenie bibilioteki winsock2 o dodatkowe funkcje dla połączenia Bluetooth
- BluetoothAPIs – API zawierające funkcje interface'u Bluetooth np. sprawdzanie adresu MAC, albo statusu

3 Kod programu

```
#include <cstdio>
#include <Winsock2.h>
#include <Ws2bth.h>
#include <BluetoothAPIs.h>

#pragma comment(lib, "Ws2_32.lib")
#pragma comment(lib, "Bthprops.lib")

// struktura z parametrami wyszukiwania urzadzen BT
BLUETOOTH_DEVICE_SEARCH_PARAMS bt_dev_search_params = {
        sizeof(BLUETOOTH_DEVICE_SEARCH_PARAMS),
        1,
        0,
        1,
        1,
        1,
        20,
        nullptr
};

const int MAX_BT_RADIOS = 10;
const int MAX_BT_DEV = 10;
int bt_radio_id = 0;
int bt_dev_id = 0;

// struktura z parametrami wyszukiwania adaptera BT
BLUETOOTH_FIND_RADIO_PARAMS bt_find_radio_params = {
sizeof(BLUETOOTH_FIND_RADIO_PARAMS) };
// struktura z parametrami adaptera BT
BLUETOOTH_RADIO_INFO bt_radio_info = { sizeof(BLUETOOTH_RADIO_INFO), 0, };
// znalezione adaptery BT
HANDLE radios[MAX_BT_RADIOS];
// znaleziony adapter BT
HBLUETOOTH_RADIO_FIND bt_radio_find;
BLUETOOTH_DEVICE_INFO devices[MAX_BT_DEV];
HBLUETOOTH_DEVICE_FIND bt_dev_find;

int main()
{
    // wyszukiwanie pierwszego adaptera BT
    bt_radio_find = BluetoothFindFirstRadio(&bt_find_radio_params,
&radios[bt_radio_id]);
    bt_radio_id++;
    if (bt_radio_find == nullptr) {
        printf("Nie znaleziono zadnego adaptera Bluetooth! Kod bledu: %lu\n",
GetLastError());
    }   // wyszukiwanie kolejnych adapterów BT
```

```
    else {
        while (BluetoothFindNextRadio(bt_radio_find, &radios[bt_radio_id])) {
            bt_radio_id++;
            if (bt_radio_id == MAX_BT_RADIOS - 1) {
                bt_radio_id--;
                printf("Znaleziono wiecej niz 10 adapterow Bluetooth!\n");
                break;
            }
        }
    }

    for (int i = 0; i < bt_radio_id; i++) {
        // pobieranie informacji o danym adapterze i umieszczanie ich w
strukturze przechowujacej te dane
        Sleep(500);
        BluetoothGetRadioInfo(radios[i], &bt_radio_info);
        wprintf(L"\nUrzadzenie: %d", i);
        wprintf(L"\n\tNazwa: %s", bt_radio_info.szName);
        wprintf(L"\n\tAdres MAC: %02X:%02X:%02X:%02X:%02X:%02X",
bt_radio_info.address.rgBytes[5],
                bt_radio_info.address.rgBytes[4],
bt_radio_info.address.rgBytes[3], bt_radio_info.address.rgBytes[2],
                bt_radio_info.address.rgBytes[1],
bt_radio_info.address.rgBytes[0]);
        wprintf(L"\n\tKlasa: 0x%08x", bt_radio_info.ulClassofDevice);
        wprintf(L"\n\tProducent: 0x%04x\n", bt_radio_info.manufacturer);
    }

    if (!BluetoothFindRadioClose(bt_radio_find)) printf("Blad zamykania
wyszukiwania adapterow BT.");

    int choose_radio = 0;
    printf("\nWybierz adapter:\n>");
    scanf_s("%d", &choose_radio);
    printf("\nWybrano adapter %d.\n", choose_radio);

    printf("\nWyszukiwanie urzadzen bluetooth\n");
    // ustawianie adaptera dla danych parametrow wyszukiwania urzadzen BT
    bt_dev_search_params.hRadio = radios[choose_radio];

    devices[0].dwSize = sizeof(BLUETOOTH_DEVICE_INFO);
    bt_dev_find = BluetoothFindFirstDevice(&bt_dev_search_params,
&devices[0]);

    if (bt_dev_find == nullptr) {
        printf("\nNie znaleziono zadnych urzadzen Bluetooth!");
        BluetoothFindDeviceClose(bt_dev_find);
        return 0;
    }
```

```
    else {
        bt_dev_id++;
        devices[bt_dev_id].dwSize = sizeof(BLUETOOTH_DEVICE_INFO);
        while (BluetoothFindNextDevice(bt_dev_find, &devices[bt_dev_id])) {
            bt_dev_id++;
            devices[bt_dev_id].dwSize = sizeof(BLUETOOTH_DEVICE_INFO);
        }
        if (BluetoothFindDeviceClose(bt_dev_find)) printf("\nKoniec
wyszukiwania urzadzen.");
        else printf("\nBlad konca wyszukiwania urzadzen.");
    }

    printf("\nZnaleziono %d urzadzen.", bt_dev_id);

    for (int i = 0; i < bt_dev_id; i++) {
        Sleep(500);
        wprintf(L"\nUrzadzenie: %d", i);
        wprintf(L"\n\tNazwa: %s", devices[i].szName);
        wprintf(L"\n\tAdres MAC: %02X:%02X:%02X:%02X:%02X:%02X",
devices[i].Address.rgBytes[5],
                devices[i].Address.rgBytes[4], devices[i].Address.rgBytes[3],
devices[i].Address.rgBytes[2],
                devices[i].Address.rgBytes[1], devices[i].Address.rgBytes[0]);
        wprintf(L"\n\tKlasa: 0x%08x", devices[i].ulClassofDevice);
        wprintf(L"\n\tPolaczone: %s\r\n", devices[i].fConnected ? L"true" :
L"false");
        wprintf(L"\tUwierzytelnione: %s\r\n", devices[i].fAuthenticated ?
L"true" : L"false");
        wprintf(L"\tZapamietane: %s\r\n", devices[i].fRemembered ? L"true" :
L"false");
    }

    int choose_dev = 0;
    printf("\nWybierz urzadzenie:\n>");
    scanf_s("%d", &choose_dev);
    printf("\nWybrano urzadzenie numer %d.\n", choose_dev);

    // autentykacja polaczenia miedzy wybranym adapterem z wybranym
urzadzeniem BT
    if (radios[choose_radio] != nullptr &&
!devices[choose_dev].fAuthenticated)
        BluetoothAuthenticateDeviceEx(nullptr, radios[choose_radio],
&devices[choose_dev], nullptr, MITMProtectionRequired);

    // czesc odpowiedzialna za otwarcie gniazda i nawiazanie polaczenia
    WSADATA wsaData;
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        printf("Nieudana proba inicjalizacji WinSock");
        return 1;
```

```
    }
    // ustawianie gniazda
    SOCKET btSocket = socket(AF_BTH, SOCK_STREAM, BTHPROTO_RFCOMM);
    if (btSocket == INVALID_SOCKET) {
        printf("Nie udalo sie utworzyc socketa bluetooth: %d",
WSAGetLastError());
        WSACleanup();
        return 1;
    }

    SOCKADDR_BTH serverAddr;
    serverAddr.addressFamily = AF_BTH;
    serverAddr.serviceClassId = OBEXObjectPushServiceClass_UUID;
    // OBEXObjectPushServiceClass_UUID;
    serverAddr.port = BT_PORT_ANY;
    serverAddr.btAddr = devices[choose_dev].Address.ullLong;

    if (connect(btSocket, (SOCKADDR*)&serverAddr, sizeof(serverAddr)) ==
SOCKET_ERROR) {
        printf("Nie udalo sie polaczyc!\n Error numer: %d",
WSAGetLastError());
        closesocket(btSocket);
        WSACleanup();
        return 1;
    }
    printf("Polaczenie udane!\n");


    closesocket(btSocket);
    WSACleanup();


    return 0;
}
```

4 Wnioski

Zadanie pozwoliło na zrozumienie działania połączeń Bluetooth, jednak pomimo wiedzy na temat OBEX, ze względu trudność implementacji w wybranym języku, nie udało się zrealizować przesyłania plików.

5 Bibliografia

https://en.wikipedia.org/wiki/Bluetooth

https://www.winsocketdotnetworkprogramming.com/winsock2programming/winsock2advancedotherprotocol4k.html

https://learn.microsoft.com/en-us/windows/win32/api/bluetoothapis/