Piotr Chłystek 226100
Michał Chojnacki 225936

Data oddania sprawozdania: 20.11.2017 r.
Termin zajęć: Pondziałek 7:30-10:15, TP

Urządzenia peryferyjne

Ćwiczenie 14

Bluetooth – komunikacja z telefonem komórkowym

Prowadzący:
dr inż. Jan Nikodem

1. Cel ćwiczenia

Celem ćwiczenia było napisane programu, który wykrywa adaptery Bluetooth podłączone do komputera, wyświetla informacje o nich, pozwala wybrać z listy wykrytych adapterów jeden, który posłuży do wykrycia w pobliżu urządzeń, obsługujących Bluetooth i nawiązania połączenia z urządzeniem wybranym z listy. Po autoryzacji połączenia urządzenia powinny mieć możliwość wymiany danych.

2. Wstęp teoretyczny

Bluetooth to standard bezprzewodowej komunikacji krótkiego zasięgu (klasa 1 ma zasięg do 100 metrów). Służy do wymiany danych między urządzeniami, takimi jak komputer, telefon lub inne urządzenia peryferyjne przez adapter. Technologia ta korzysta z fal radiowych w paśmie ISM (pierwotnie przeznaczone do zastosowań przemysłowych, naukowych i medycznych – kolejno Industrial, Scientific oraz Medical) 2,4 GHz i wykorzystuje modulację FSK (kluczowanie z przesuwem częstotliwości).

3. Opis programu

Lista bibliotek, użytych do napisania programu

```
#include "stdafx.h"
#include <stdlib.h>
#include <stdio.h>
#include <Winsock2.h>
#include <Ws2bth.h>
#include <BluetoothAPIs.h>
```

Dopisane zostały także biblioteki dla linkera

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

Winsock2.h – biblioteka, która umożliwia tworzenie zaawansowanych gniazd dla połączeń sieciowych (soketów)

BluetoothAPIs.h – biblioteka, która zawiera funkcje do zarządzania urządzeniami Bluetooth

Ws2bth.h – biblioteka, rozszerzająca bibliotekę winsock o funkcje napisane z myślą o technologii Bluetooth

Wykrywanie adapterów odbywa się w następujący sposób: tworzymy tablicę handlerów radios i zakładamy że nie ma ich więcej niż 10. Dodatkowo tworzymy zmienną typu HBLUETOOTH_RADIO_FIND, która będzie przechowywała ilość znalezionych już adapterów i domyślnie inicjujemy jej wartość 0.

```
const int MAX_BT_RADIOS = 10;
// tablica uchwytow przechowujaca znalezione adaptery BT
HANDLE radios[MAX_BT_RADIOS];
// uchwyt przechowujacy pierwszy znaleziony adapter BT
HBLUETOOTH_RADIO_FIND bt_radio_find;
// zmienna przechowujaca ilosc znalezionych adapterów
int bt_radio_id = 0;
```

Znajdowanie adapterów odbywa się według poniższego bloku kodu

```
	// wyszukiwanie pierwszego adaptera BT
bt_radio_find = BluetoothFindFirstRadio(&bt_find_radio_params,&radios[bt_radio_id]);
	bt_radio_id++;

if (bt_radio_find == NULL)
	printf("Nie znaleziono zadnego adaptera Bluetooth! Kod bledu: %d\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!");
			break;
		}
	}
```

Wpierw wyszukiwany jest pierwszy adapter (w celu określenia czy do komputera podłączony jest jakikolwiek adapter Bluetooth), jeżeli nie zostanie wykryty żaden program nas o tym poinformuje. Dopiero po znalezieniu po znalezieniu pierwszego szukane są następne adaptery. Po otrzymaniu listy adapterów program pobiera i wypisuje informacje o nich w pętli

```
for (int i = 0; i < bt_radio_id; i++) {
		// pobieranie informacji o danym adapterze i umieszczanie ich w
strukturze przechowujacej te dane
BluetoothGetRadioInfo(radios[i], &bt_radio_info);
// drukowanie informacji o wybranym adapterze ze struktury
wprintf(L"\nUrzadzenie: %d", i);
wprintf(L"\n\t Nazwa: %s", bt_radio_info.szName);
wprintf(L"\n\t Adres MAC: %02X:%02X:%02X:%02X:%02X:%02X",
bt_radio_info.address.rgBytes[0],
bt_radio_info.address.rgBytes[1], bt_radio_info.address.rgBytes[2],
bt_radio_info.address.rgBytes[3],
bt_radio_info.address.rgBytes[4], bt_radio_info.address.rgBytes[5]);
		wprintf(L"\n\t Klasa: 0x%08x", bt_radio_info.ulClassofDevice);
		wprintf(L"\n\t Producent: 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: ");
scanf_s("%d", &choose_radio);
printf("\nWybrano %d adapter.\n", choose_radio);
```

Gdzie rgBytes to ciągi bitów adresu MAC, ulClassofDevice to klasa urządzenia, manufacturer to producent. Jeżeli nie ma żadnych adapterów, program wypisze właściwy komunikat. Kolejne linie kodu mają na celu umożliwić użytkownikowi wybór adaptera z listy dostępnych adapterów.

Po wybraniu adaptera odbywa się wyszukiwanie dostępnych urządzeń, które wymaga ustawienia parametrów szukania. Parametry szukania są przechowywane w specjalnej strukturze BLUETOOTH_DEVICE_SEARCH_PARAMS. Przechowywanie znalezionych urządzeń odbywa się analogicznie do przechowywania wykrytych adapterów. Zakładamy że urządzeń w pobliżu może być co najwyżej 10, informacje o urządzeniach są przechowywane w specjalnej tablicy typu BLUETOOTH_DEVICE_INFO.

```
// 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,
	NULL
};

const int MAX_BT_DEV = 10;
BLUETOOTH_DEVICE_INFO devices[MAX_BT_DEV];
HBLUETOOTH_DEVICE_FIND bt_dev_find;
int bt_dev_id = 0;
```

Następny blok kodu to poinformowanie użytkownika o rozpoczęciu wyszukiwania dostępnych urządzeń, ustawienie adaptera dla parametrów wyszukiwania urządzeń Bluetooth. Utworzona zostaje tablica, przechowująca znalezione urządzenia a następnie program rozpoczyna wyszukiwanie pierwszego urządzenia. W zależności od tego czy zostanie ono znalezione, program poinformuje użytkownika o nieznalezieniu żadnych urządzeń lub kontynuuje poszukiwania.

```
printf("\nWyszukiwanie urzadzen...\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 == NULL) {
	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);
```

Następnie program wypisuje informacje o znalezionych urządzeniach, również w sposób analogiczny do wypisywania informacji o znalezionych adapterach

```
for (int i = 0; i < bt_dev_id; i++) {
	wprintf(L"\nUrzadzenie: %d", i);
	wprintf(L"\n\t Nazwa: %s", devices[i].szName);

wprintf(L"\n\t Adres MAC:%02X:%02X:%02X:%02X:%02X:%02X",devices[i].Address.rgBytes[0],
```

```
devices[i].Address.rgBytes[1], devices[i].Address.rgBytes[2],
devices[i].Address.rgBytes[3], devices[i].Address.rgBytes[4],
devices[i].Address.rgBytes[5]);

	wprintf(L"\n\t Klasa: 0x%08x", devices[i].ulClassofDevice);
wprintf(L"  \Polaczone: %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: ");
scanf_s("%d", &choose_dev);
printf("\nWybrano %d urzadzenie.\n", choose_dev);
```

Kolejne dwie linijki dotyczą autoryzacji połączenia między adapterem i urządzeniem. Program wpierw sprawdza czy wybrane urządzenie znajduje się na liście urządzeń i czy połączenie nie było wcześniej autoryzowane. Autoryzacja odbywa się przez wywołanie funkcji BluetoothAuthenticateDeiceEx, której parametrami są uchwyt adaptera, wskaźnik na urządzenie oraz kod autoryzacji, który musi się zgadzać zarówno na urządzeniu i komputerze by połączenie zostało nawiązane.

```
if (radios[choose_radio] != NULL && !devices[choose_dev].fAuthenticated)
BluetoothAuthenticateDeviceEx(NULL, radios[choose_radio], &devices[choose_dev], NULL,
MITMProtectionRequired);
```

Następnie należy utworzyć nowe gniazdo Winsock, służącego do komunikacji w technologii Bluetooth. By utworzyć gniado należy zastosować strukturę SOCKADDR_BTH z biblioteki Ws2bth.h oraz strukturę WSADATA zaprojektowaną do inicjalizacji soketów. Następuje ustawienie atrybutów sckadr, czyli naszego soketu, po czym dochodzi do próby nawiązania połączenia, użytkownik otrzymuje komunikat i socket zostaje zamknięty.

```
// czesc odpowiedzialna za otwarcie gniazda i nawiazanie polaczenia
WSADATA wsaData;
int sck;
SOCKADDR_BTH sckadr;
int result = WSAStartup(MAKEWORD(2, 2), &wsaData);
if (result != NO_ERROR) {
	printf("Blad funkcji WSAStartup!");
	return 0;
	}

// ustawianie gniazda
sck = socket(AF_BTH, SOCK_STREAM, BTHPROTO_RFCOMM);
if (sck == INVALID_SOCKET) {
	printf("Blad tworzenia gniazda!");
	return 0;
	}
else
{
	sckadr.addressFamily = AF_BTH;
	sckadr.btAddr = devices[choose_dev].Address.ullLong;
	sckadr.port = BT_PORT_ANY;
	sckadr.serviceClassId = OBEXObjectPushServiceClass_UUID;
// zestawianie polaczenia
	if (connect(sck, (SOCKADDR *) &sckadr, sizeof(sckadr))) {
		printf("Blad polaczenia!");
		closesocket(sck);
		WSACleanup();
		return 0;
		}
	printf("Poprawnie polaczono z urzadzeniem nr: %d", choose_dev);
```

```
	closesocket(sck);
	WSACleanup();
}
	// czyszczenie strummienia stdin
	fseek(stdin, 0, SEEK_END);
	getchar();
	return 0;
}
```

Transfer danych odbywa się przez protokół OBEX przez polecenia CONNECT, PUT oraz DISCONNECT. Obiekty OBEX składają się z sekwencji nagłówków, przeważnie jednobajtowych. Bloki kodu, mające realizować przesyłanie danych między urządzeniami okazały się wadliwe, dlatego zostały usunięte z programu i nie zdążyliśmy w czasie zajęć napisać poprawnie funkcjonujących procedur.

4.Wnioski

Zajęcia pozwoliły nam na zapoznanie się z bibliotekami, służącymi do dwukierunkowej komunikacji z urządzeniami zewnętrznymi, obsługującymi technologię Bluetooth. Problematyczny okazał się protokół OBEX i jego implementacja, w celu wymiany plików.