

Urządzenia peryferyjne

Prowadzący zajęcia: dr inż. Jan Nikodem

Termin zajęć: Czwartek TN, godz. 14:10

Osoby wykonujące ćwiczenie:

Mateusz Gawłowski, Bartosz Szymański

Oznaczenie grupy:

B

Tytuł ćwiczenia:

KARTY MIKROPROCESOROWE (GSM/SIM)

Data wykonania ćwiczenia:

23.11.2023

Spis treści

1. Cel ćwiczenia	2
2. Zadania do wykonania.....	2
3. Wstęp teoretyczny	2
3.1 Karty GSM/SIM	2
3.2 System plików na karcie SIM	3
3.3. Komendy APDU	4
3.4. PDU SMS	4
4. Użyte oprogramowanie	5
5. Rozwiązania zadań wraz z krótkim opracowaniem	6
5.1 Program do obsługi transmisji GPS	6
Opracowanie.....	10
6. Wnioski	11
7. Literatura	11

1. Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z zasadami obsługi kart mikroprocesorowych GSM/SIM z wykorzystaniem interfejsu PC/SC. Napisanie aplikacji umożliwiającej odczytywanie danych z kart SIM w zakresie komend APDU oraz wyświetlenie książki telefonicznej i odczytywanie listy SMS-ów oraz wyświetlanie ich treści.

2. Zadania do wykonania

- Przetestować przygotowane komendy APDU dla kart GSM. Można skorzystać z gotowej aplikacji i odczytać dane z kart (lista kontaktów oraz SMS-y). Aplikacja powinna znajdować się na komputerze w katalogu C:\SmartCardReader – jeżeli jej brak, to ściągnąć ją i zainstalować.
- Napisać program (może być wersja konsolowa) do odczytywania danych z kart SIM za pomocą czynnika kart mikroprocesorowych, który:
 - Wyświetli nazwę podłączonego do komputera, urządzenia-czytnika
 - Umożliwi wybór czytnika
 - Umożliwi wysłanie (w formacie HEX) komendy do czytnika i wyświetli odpowiedź w formacie HEX oraz znakowo
 - Odczyta listę kontaktów (książkę telefoniczną)
 - Odczyta listę SMS-ów oraz wyświetli dane techniczne oraz ich treść

3. Wstęp teoretyczny

3.1 Karty GSM/SIM

Karty mikroprocesorowe GSM/SIM (Subscriber Identity Module) stanowią integralny element systemów telekomunikacyjnych, zwłaszcza w kontekście sieci komórkowych. Te niewielkie, plastikowe karty zawierają mikroprocesor, pamięć i interfejs komunikacyjny, umożliwiając przechowywanie istotnych informacji, takich jak identyfikator abonenta (IMSI), książka telefoniczna czy wiadomości SMS

Protokół PC/SC (Personal Computer/Smart Card) stanowi standardowy interfejs komunikacyjny między komputerem osobistym a kartami mikroprocesorowymi. Otwarte standardy tego protokołu umożliwiają interakcję pomiędzy oprogramowaniem na komputerze a kartami smart. W kontekście niniejszego ćwiczenia, interfejs PC/SC zostanie wykorzystany do komunikacji z kartą SIM, co umożliwi odczytywanie danych oraz kontrolę nad funkcjami karty.

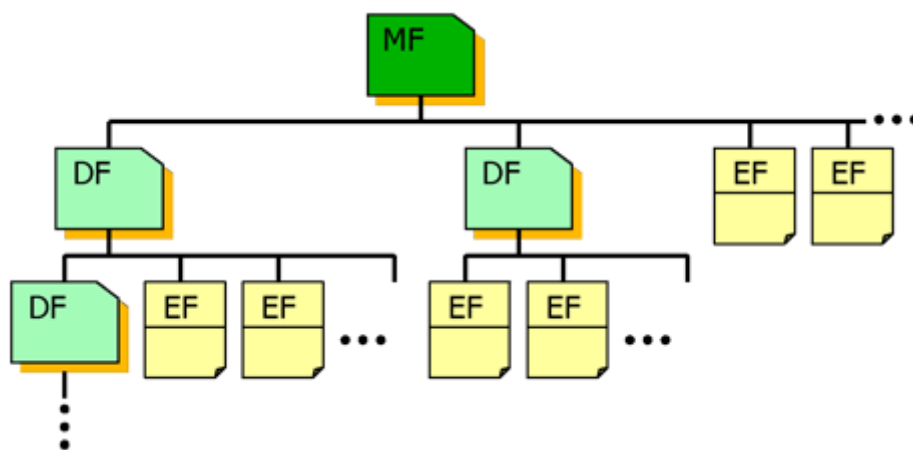
3.2 System plików na karcie SIM

System plików na karcie SIM można zobrazować w formie struktury drzewa. Wyróżniamy trzy główne typy plików:

- MF (Master File) – plik główny, pełniący rolę głównego folderu.
- DF (Dedicated File) – odpowiadający folderom w hierarchii plików.
- EF (Elementary File) – plik zawierający konkretny zestaw danych.

Funkcjonowanie czytnika kart jest łatwe do zrozumienia. Po włożeniu karty chipowej do czytnika, karta jest zasilana poprzez właściwe styki. Kolejno, z karty przesyłana jest wiadomość ATR (Answer to Reset), która dostarcza czytnikowi kluczowych informacji, takich jak:

- Typ karty:
 - Wiadomość ATR informuje czytnik o rodzaju karty, co umożliwia identyfikację, czy jest to karta mikroprocesorowa GSM/SIM, czy też inny typ karty.
- Metoda kodowania bitów 0 i 1:
 - Karta przekazuje informacje odnośnie metody kodowania bitów 0 i 1, co jest istotne dla odpowiedniego zrozumienia formatu danych przesyłanych między kartą a czytnikiem.
- Obsługiwane protokoły komunikacyjne:
 - Wiadomość ATR zawiera również informacje dotyczące obsługiwanych protokołów komunikacyjnych przez kartę. Jest to kluczowe dla właściwego nawiązania połączenia i komunikacji między czytnikiem a kartą.



Rysunek 1 – schemat systemu plików karty SIM

3.3. Komendy APDU

Komendy APDU stanowią standardowy protokół komunikacyjny używany do wymiany informacji między kartą mikroprocesorową a urządzeniem terminalnym. Składają się one z dwóch rodzajów komend: komend poleceń (command) i komend danych (data). W przypadku kart SIM, komendy APDU odgrywają kluczową rolę w zarządzaniu kartą i uzyskiwaniu dostępu do przechowywanych danych. Poniżej przedstawiono dokładny opis struktury wspomnianych komend:

Command APDU						
Header (required)				Body (optional)		
CLA	INS	P1	P2	Lc	Data Field	Le

Rysunek 2 – struktura komend APDU

- CLA - określa klasę instrukcji.
- INS - instrukcja, którą ma wykonać karta.
- P1 i P2 - parametr pierwszy i parametr drugi instrukcji.
- Lc - długość instrukcji.
- Data - dane przeznaczone do instrukcji.

3.4. PDU SMS

PDU, czyli Protocol Data Unit, w kontekście SMS (Short Message Service), oznacza jednostkę danych protokołu używanego do przesyłania krótkich wiadomości tekstowych w sieciach komórkowych. PDU SMS definiuje sposób, w jaki informacje są zakodowane i przesyłane między urządzeniem źródłowym (nadawcą) a urządzeniem docelowym (odbiorcą) podczas wymiany wiadomości SMS.

W PDU SMS zawarte są informacje takie jak numer telefonu odbiorcy, długość wiadomości, typ wiadomości (np. SMS zwykły, Flash SMS) oraz sam tekst wiadomości. PDU umożliwia bardziej efektywne przesyłanie danych, ponieważ pozwala na bardziej zwięzłą reprezentację informacji w porównaniu do innych formatów, takich jak SMS w formacie tekstowym (TP-Text).

PDU SMS jest często używane w zaawansowanych scenariuszach programistycznych, integracjach systemów oraz w pracy z GSM modułami komunikacyjnymi, gdzie konieczne jest precyzyjne zarządzanie danymi przesyłanymi w ramach wiadomości SMS.

Kompletny komunikat po zakodowaniu:
07918406010013F011000B918405112030F00000C405D72435A80C01

Rysunek 3 – przykładowy komunikat PDU SMS

Tabela 1 – opis struktury komunikatu PDU SMS

Opis parametru	Wartość (szesnastkowo)	Uwagi
Długość pola numeru Centrum Usług	07	Liczba oktetów numeru Centrum Usług z uwzględnieniem pola typu numeracji
Typ numeru Centrum Usług	91	Typ numeru Centrum Usług - dla numeracji międzynarodowej 91H
Numer Centrum Usług	8406010013F0	Numer Centrum Usług, tu podany dla operatora sieci Plus GSM (+48601000310)
Pierwszy oktet	11	Wartość 11H przy wywołaniu funkcji SMS-SUBMIT oraz relacyjny format okresu ważności SMS
Numer odniesienia	00	Pierwszy komunikat z wiadomości dzielonej lub pojedynczy komunikat
Długość numeru odbiorcy	0B	Liczba cyfr numeru odbiorcy
Typ numeru odbiorcy	91	Typ numeru odbiorcy - dla numeracji międzynarodowej 91H
Numer odbiorcy	8405112030F0	Numer odbiorcy zakodowany w identyczny sposób jak numer Centrum Usług
Identyfikator protokołu	00	Oznacza dostarczenie wiadomości jako normalnej wiadomości SMS
Schemat kodowania	00	Oznacza kodowanie domyślne w formacie 7-bitowych znaków ASCII
Okres ważności	C4	Okres ważności wiadomości SMS - 30 dni (format relatywny)
Długość pola danych użytkownika	06	Tekst użytkownika zawiera 5 oktetów (po konwersji septetów)
Dane użytkownika	D72435A80C01	Tekst komunikatu SMS: WITAJ!

4. Użyte oprogramowanie

- Środowisko programistyczne wykorzystane podczas tworzenia programu - Visual Studio.
- Aplikacja służąca do weryfikacji funkcjonalności karty - SmartCardReader.exe.
- Czytnik użyty do odczytu danych z karty - Omnikey 5321v2.
- Karta użyta należy do firmy Play.

5. Rozwiązania zadań wraz z krótkim opracowaniem

5.1 Program do obsługi transmisji GPS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using PCSC;

namespace ConsoleApplication1
{
    class Program
    {
        private static SCardError error;
        private static SCardReader reader;
        private static System.IntPtr intptr;
        private static SCardContext hContext;
        static void Main(string[] args)
        {
            byte[] commandB;
            try
            {
                connect();

                // get in TELECOM
                commandB = new byte[] { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x7F,
0x10 }; // adress constructed with 2 chars 107F
                sendCommand(commandB, "SELECT(TELECOM)");

                commandB = new byte[] { 0xA0, 0xC0, 0x00, 0x00, 0x16 }; //
wait for response with length 22(10)
                sendCommand(commandB, "\nGET RESPONSE");

                // get in SMS
                commandB = new byte[] { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x6F,
0x3C };

                sendCommand(commandB, "\nSELECT SMS");
                commandB = new byte[] { 0xA0, 0xC0, 0x00, 0x00, 0x16 };
            }
            catch { }
        }
    }
}
```

```

        // SMS read
        commandB = new byte[] { 0xA0, 0xB2, 0x01, 0x04, 0xB0 };
        sendCommand(commandB, "\nREAD RECORD");
        commandB = new byte[] { 0xA0, 0xC0, 0x00, 0x00, 0x16 };
        sendCommand(commandB, "\nGET RESPONSE");

        Console.WriteLine("\nPress any buton to continue...\n");
        Console.ReadKey();
    }
    catch (Exception e)
    {
        Console.WriteLine("Error occured");
        Console.ReadKey();
    }
}

public static void toTelecom()
{
    byte[] commandBytes = new byte[] {0xA0, 0xA4, 0x00, 0x00, 0x02,
0x7F, 0x10};

    sendCommand(commandBytes, "SELECT TELECOM");

    // GET RESPONSE
    commandBytes = new byte[] {0xA0, 0xC0, 0x00, 0x00, 0x16};
    sendCommand(commandBytes, "GET RESPONSE");

    // SELECT SMS
    commandBytes = new byte[] {0xA0, 0xA4, 0x00, 0x00, 0x02, 0x6F,
0x3C};

    sendCommand(commandBytes, "SELECT SMS");

    // GET RESPONSE
    commandBytes = new byte[] {0xA0, 0xC0, 0x00, 0x00, 0x0F};
    sendCommand(commandBytes, "GET RESPONSE");
}

```

```

public static void connect()
{
    hContext = new SCardContext();
    hContext.Establish(SCardScope.System);

    string[] readerList = hContext.GetReaders(); // Load available
readers
    Boolean noReaders = readerList.Length <= 0;

    if (noReaders)
    {
        throw new PCSCException(SCardError.NoReadersAvailable, "Reader
was not found");
    }

    int counter = 1;
    Console.WriteLine("Choose reader: ");

    foreach (string element in readerList)
    {
        Console.WriteLine("F" + counter + " -> " + element);
        counter++;
    }

    var input = Console.ReadKey();
    string tmp = readerList[0];

    switch (input.Key)
    {
        case ConsoleKey.F1:
            tmp = readerList[0];
            break;

        case ConsoleKey.F2:
            tmp = readerList[1];
            break;
    }
}

```



```

        Console.WriteLine("\nPress any button to continue...\n");
        Console.ReadKey();

        reader = new SCardReader(hContext);

        error = reader.Connect(tmp, SCardShareMode.Shared,
SCardProtocol.T0 | SCardProtocol.T1);
        checkError(error);

        if (reader.ActiveProtocol == SCardProtocol.T0)
        {
            intptr = SCardPCI.T0;
        }
        else if (reader.ActiveProtocol == SCardProtocol.T1)
        {
            intptr = SCardPCI.T1;
        }
        else
        {
            Console.WriteLine("Protocol is not available");
            Console.WriteLine("\nPress any button to continue...\n");
            Console.ReadKey();
        }
    }

    public static void sendCommand(byte[] command, String name) // send
command to card
    {
        byte[] recievedBytes = new byte[256];
        error = reader.Transmit(intptr, command, ref recievedBytes);
        checkError(error);
        writeResponse(recievedBytes, name);
    }

    public static void writeResponse(byte[] recievedBytes, String
responseCode) // read response from card
    {
        Console.Write(responseCode + ": ");

        for (int i = 0; i < recievedBytes.Length; i++)
            Console.Write("{0:X2} ", recievedBytes[i]); // print binary
answer

        Console.WriteLine();
    }

```

```

        static void checkError(SCardError error) // check if card is inserted
        {
            if (error != SCardError.Success)
            {
                throw new PCSCException(error,
SCardHelper.StringifyError(error));
            }
        }
    }
}

```

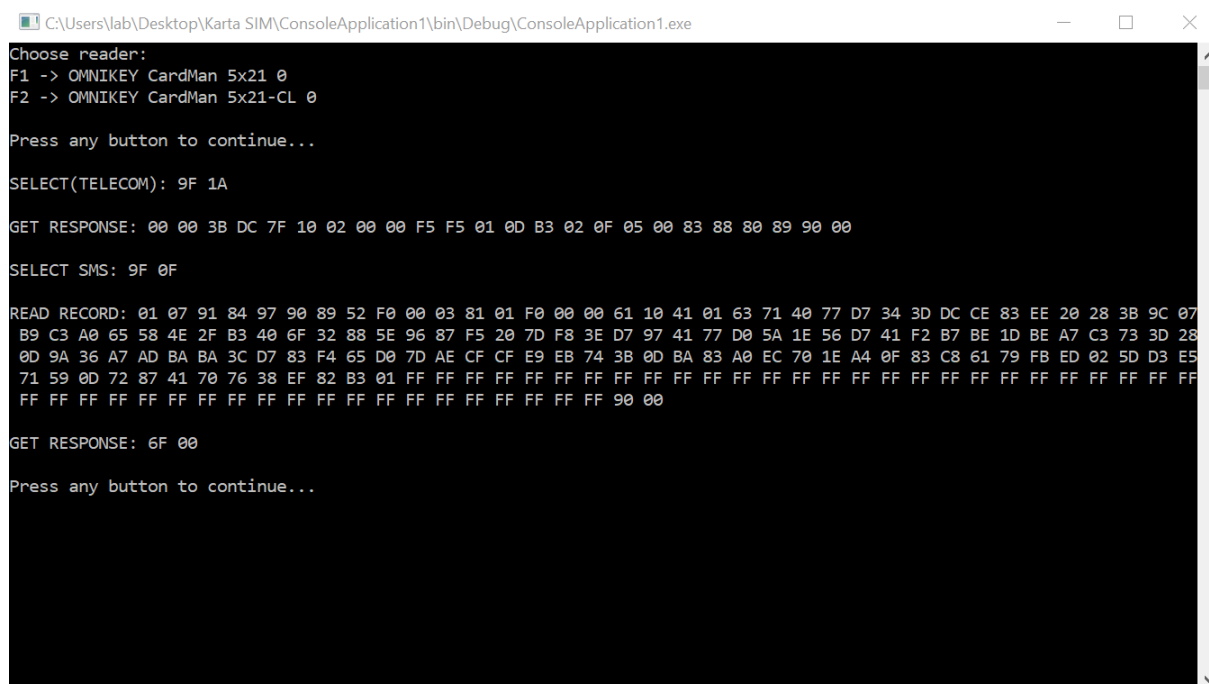
Opracowanie

Kod jest przykładową implementacją komunikacji z kartą smart card wykorzystującą bibliotekę PCSC w języku C#. Program skupia się na komunikacji z kartą SIM w celu odczytu informacji związanego z usługami telekomunikacyjnymi i wiadomościami SMS.

- Metoda *Main()*:
 - Inicjuje połączenie z kartą poprzez *connect()*.
 - Wysyła kilka poleceń do karty (np. wybór aplikacji *TELECOM*, odczyt odpowiedzi itp.) za pomocą *sendCommand()*.
 - Wyświetla komunikat i oczekuje na dowolne naciśnięcie klawisza przez użytkownika.
- Metoda *connect()*:
 - Inicjuje kontekst (*SCardContext*) do obsługi operacji z kartą.
 - Pobiera listę dostępnych czytników.
 - Prosi użytkownika o wybór czytnika z listy.
 - Nawiązuje połączenie z wybranym czytnikiem i określa protokół komunikacji.
- Metoda *sendCommand(byte[] command, String name)*:
 - Wysyła komendę do karty za pomocą *reader.Transmit()*.
 - Odbiera odpowiedź i wywołuje *writeResponse()*.
- Metoda *writeResponse(byte[] recivedBytes, String responseCode)*:
 - Wyświetla otrzymane dane w postaci szesnastkowej w konsoli.
- Metoda *checkError(SCardError error)*:
 - Sprawdza, czy operacja na karcie zakończyła się sukcesem. W przypadku wystąpienia błędu, wywołuje wyjątek *PCSCException*.

Kod skupia się na podstawowych operacjach komunikacyjnych z kartą SIM, takich jak wysyłanie poleceń do karty, odbieranie odpowiedzi i wyświetlanie danych. Program interaktywnie pozwala użytkownikowi wybrać czytnik, z którym chce się połączyć, a następnie wykonuje zestaw operacji zdefiniowanych w *Main()* i innych pomocniczych metodach.

Poniżej przedstawiony został także ciąg szesnastkowy reprezentujący odpowiedź zwrotną (odczytane dane) z karty SIM w ramach komunikacji zaimplementowanej w podanym kodzie programu.



```
C:\Users\lab\Desktop\Karta SIM\ConsoleApplication1\bin\Debug\ConsoleApplication1.exe
Choose reader:
F1 -> OMNIKEY CardMan 5x21 0
F2 -> OMNIKEY CardMan 5x21-CL 0

Press any button to continue...

SELECT(TELECOM): 9F 1A

GET RESPONSE: 00 00 3B DC 7F 10 02 00 00 F5 F5 01 0D B3 02 0F 05 00 83 88 80 89 90 00

SELECT SMS: 9F 0F

READ RECORD: 01 07 91 84 97 90 89 52 F0 00 03 81 01 F0 00 00 61 10 41 01 63 71 40 77 D7 34 3D DC CE 83 EE 20 28 3B 9C 07
B9 C3 A0 65 58 4E 2F B3 40 6F 32 88 5E 96 87 F5 20 7D F8 3E D7 97 41 77 D0 5A 1E 56 D7 41 F2 B7 BE 1D BE A7 C3 73 3D 28
0D 9A 36 A7 AD BA BA 3C D7 83 F4 65 D0 7D AE CF CF E9 EB 74 3B 0D BA 83 A0 EC 70 1E A4 0F 83 C8 61 79 FB ED 02 5D D3 E5
71 59 0D 72 87 41 70 76 38 EF 82 B3 01 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

GET RESPONSE: 6F 00

Press any button to continue...
```

Rysunek 4 – odpowiedź zwrotna programu do odczytu danych z karty SIM

6. Wnioski

Realizacja ćwiczenia w znaczący sposób wpłynęła na nasz poziom rozumienia tematu w obszarze sposobu działania i odczytu danych z kart GSM/SIM. Kod wykonany na laboratoriach zaprezentowany został w niezmienionej formie w sprawozdaniu, gdyż jego działanie było zgodne z założeniami ćwiczenia.

7. Literatura

Poprzez literaturę należy rozumieć wszelkie materiały potrzebne do zrozumienia założeń laboratoriów, sposobu realizacji ćwiczenia oraz źródeł pomocy podczas rozwiązywania problemów napotkanych w trakcie realizacji zajęć.

- Schemat komunikatu PDU SMS:
<https://ep.com.pl/files/4408.pdf>
- System plików na karcie SIM:
<https://www.forensicsware.com/digital-forensics/sim-card.html>