Urządzenia peryferyjne laboratorium	
Temat:	Czytnik kart chipowych
Data zajęć:	20.10.2021
Grupa zajęciowa:	TN środa 11:15
Członkowie zespołu:	Aleksandra Ciężka 252701 Mateusz Chalik 252735
Prowadzący:	dr inż. Tomasz Walkowiak

1. Wstęp

Celem ćwiczenia było odczytanie danych z karty SIM przy wykorzystaniu czytnika kart chipowych.

2. Realizacja zadania

Realizacje ćwiczenia rozpoczęto od odczytania danych z karty Sim za pomocą aplikacji zainstalowanej na komputerze w pracownik. Zadanie polegało na odczytaniu ilości SMS-ów zapisanych na karcie. Komunikacja z czytnikiem odbywała się za pomocą komend APDU.

2.1 Testowanie przykładowej aplikacji

W aplikacji zostały wprowadzone komendy:

A0 A4 00 00 02 3F 00 – SELECT MF
A0 A4 00 00 02 7F 10 – SELECT TELECOM
A0 A4 00 00 02 6F 3D – SELECT SMS

Udało odczytać się ilość SMS-ów na karcie SIM, która wynosiła 25.

2.2 Własna aplikacja okienkowa

Kolejnym etapem zadnienia było napisanie aplikacji okienkowej służącej do komunikowania się z kartą chipową przy pomocy standardu PC/S.C.

W aplikacji zaimplementowano funkcje:

1. Do połączenia się z czytnikiem

```
public void ConnectReader()
            {
              //defincja kontekstu
              context = new SCardContext();

             // przekazanie kontekstu
              context.Establish(SCardScope.System);
             // lista czytników
              string[] readerList = context.GetReaders();
             // lista czytników jest pusta
              Boolean emptyList = readerList.Length <= 0;

              if (emptyList)
              {
               // wyjatek
                throw new PCSCException (SCardError.NoReadersAvailable, "Blad czytnika");
              }

               // definicja czytnika
              reader = new SCardReader(context);
              // bład
              err = reader.Connect(readerList[0], SCardShareMode.Shared, SCardProtocol.T0 | SCardProtocol.T1);

             // sprawdza przypisanie
              checkError(err);
            // wybór protokołu
             switch (reader.ActiveProtocol)
              {
              case SCardProtocol.T0:
              protocol = SCardPCI.T0;
              break;

            case SCardProtocol.T1:
            protocol = SCardPCI.T1;
            break;

        default:
              throw new PCSCException(SCardError.ProtocolMismatch, "nieobslugiwany protokol: " +
        reader.ActiveProtocol.ToString());
             }
             Connected = true;
             }
```

2. Sprawdzenia czy nie wystąpił błąd

```
void checkError(SCardError err)
{
    if (err != SCardError.Success)
    {
        throw new PCSCException(err, SCardHelper.StringifyError(err));
    }
}
```

3. Wysyłania komend z CommandTextBox

```
public byte[] sendCommand(byte[] comand, String name)
            {
            // komendy z CommandTextBox

             // byte to string
            //AnswerTextBox.Text = BitConverter.ToString(comand);
            byte[] recivedBytes = new byte[256];

            //przesyła dane APDU do karty
             err = reader.Transmit(protocol, comand, ref recivedBytes);
            checkError(err);

             //sprawdza czy wystąpił błąd
            writeResponse(recivedBytes, name);

             return recivedBytes;
             }
```

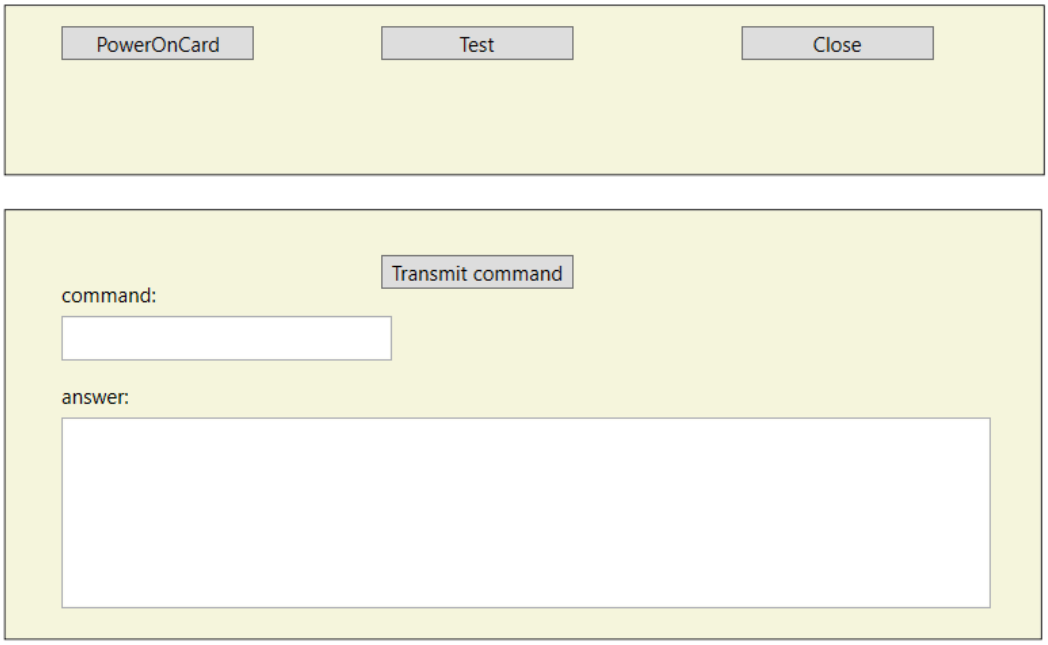
4. Wypisania odpowiedzi w AnswerTextBox

```
        public void writeResponse(byte[] recivedBytes, String responseCode)
            {
            // odpowiedx w AnswerTextBox

            // napisanie odpowiedzi w polu tekstowym
             AnswerTextBox.Text = responseCode + ": ";

            for (int i = 0; i < recivedBytes.Length; i++)
            {
                AnswerTextBox.Text = "{0:X2} " + recivedBytes[i];
            }
             AnswerTextBox.Text = "\n";
            }
```

Podjęto próbę napisania funkcji umożliwiającej odczytanie zapisanych na karcie SMS-ów.

```
case "SELECT SMS":
  // byteCommand = new byte[] { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x3F, 0x00 };
  //sendCommand(byteCommand, "MASTER FILE");

    byteCommand = new byte[] { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x7F, 0x10 };
    sendCommand(byteCommand, "SELECT TELECOM");

    byteCommand = new byte[] { 0xA0, 0xC0, 0x00, 0x00, 0x20 };
    sendCommand(byteCommand, "GET RESPONSE");

    byteCommand = new byte[] { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x3F, 0x00 };
    sendCommand(byteCommand, "SELECT MASTER FILE");

    byteCommand = new byte[] { 0xA0, 0xC0, 0x00, 0x00, 0x20 };
    sendCommand(byteCommand, "GET RESPONSE");

    byteCommand = new byte[] { 0xA0, 0xA4, 0x00, 0x00, 0x02, 0x6F, 0x3C };
    var response = sendCommand(byteCommand, "SELECT SMS");

    byteCommand = new byte[] { 0xA0, 0xC0, 0x00, 0x00, 0x20 };
    sendCommand(byteCommand, "GET RESPONSE");

    byteCommand = new byte[] { 0xA0, 0xB2, 0x00, 0x04, 0xB0 };
    sendCommand(byteCommand, "READ RECORD");

    foreach (byte bb in response)
    {
              if (bb > 0x19 && bb < 0x7B)
              {
                char anwser = Convert.ToChar(bb);
                AnswerTextBox.Text = anwser.ToString();
              }
               else
               {
                AnswerTextBox.Text = "{0:X2} "+ bb;
              }
    }
    break;
```

Okno aplikacji okienkowej:



3. Wnioski

Przy komunikacji z kartą chipową kluczową rolę odgrywa znajomość układu plików i struktury katalogów. Do komunikacji z kartą SIM za pomocą czytnika kart chipowych wykorzystuje się komendy APDU. Informacje umożliwiające poprawny przepływ danych opisane są w standardzie GSM.

Aplikacja została napisana w języku C#. W aplikacji udało stworzyć się okienko będące interfejsem użytkownika. Nie udało się jednak zaimplementować metody pozwalającej odczytać ilość oraz treść SMS-ów zapisanych na karcie SIM. Jednym z możliwych przyczyn niepowodzenia w odczytaniu danych z karty mogło być nie poprawne otwarcie katalogów, przez co odczyt danych mógł pochodzić z katalogu innego niż zawierającego dane o SMS-ach.