POLITECHNIKA WROCŁAWSKA
WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

URZĄDZENIA PERYFERYJNE - LABORATORIA

LAB 5 - Modemy. Transmisja sygnałów cyfrowych

Autorzy:
Adam Szatkowski 259056
Mateusz Bębnowicz 262751
Grupa A

Prowadzący:

dr inż. Dariusz Caban

1 Przygotowanie, wstęp teoretyczny

1.1 Modulacje

Typy modulacji

- podstawowe:
  - AM - modulacja amplitudy (modulacja polega na tym, że im większa wartość sygnału analogowego, tym większa amplituda sygnału sinusoidalnego)
  - FM - modulacja częstotliwości (modulacja polega na tym, że im większa wartość sygnału analogowego, tym większa częstotliwość (gęstość) sygnału sinusoidalnego)
  - PM - modulacja fazy (modulacja polega na tym, że im większa wartość sygnału analogowego, tym większa faza sygnału sinusoidalnego)
- kluczowanie
- modulacja impulsowa

Standardy modulacji

- ASK (pochodne od AM) - amplitudowe
- QAM - amplitudowo-fazowe
- PSK - kluczowanie fazy
- FSK - kluczowanie częstotliwości (tam, gdzie logiczne 0, sygnał ma mniejszą częstotliwość, tam gdzie 1, jest gęstszy)

1.2 Modemy

Modem - urządzenie, które moduluje/demoduluje sygnał (analogowy na cyfrowy oraz cyfrowy na analogowy), aby mógł zostać on przesłany do dalszego urządzenia. Pierwsze modemy powstawały w latach 50. XX wieku. Na przełomie lat 70. i 80. firma Hayes wprowadziła swój model *Hayes Stack SmartModem* z własnym zestawem poleceń, które stały się standardem w późniejszych latach. Wszystkie polecenia z tej rodziny zaczynają się od AT. Obecnie urządzenia sieciowe takie jak routery posiadają wbudowane modemy, aby przetwarzać odbierany sygnał analogowy na cyfrowy.

1.3 Pojęcia dodatkowe

- bit - jednostka danych (0 lub 1)
- Hz - herc, jednostka częstotliwości (1/T)
- Bd - bod (ang. baud) - Przykładowo, prędkość transmisji rzędu 250 Bd oznacza, że w ciągu każdej sekundy sygnał może zmienić się 250 razy, czyli może zostać przesłanych 250 symboli.

1.4 Komendy Hayesa

Wszystkie komendy Hayesa zaczynają się od wprowadzenia "AT" i zatwierdzane są znakiem nowej linii. Oto najważniejsze z nich:

- ATD*numer_telefonu* - wykonanie połączenia na wskazany numer (konsola wyświetla wtedy na ekranie tekst *RING* u odbiorcy),
- ATA - odbieranie połączenia przychodzącego (w konsolach obu uczestników rozmowy pojawia się *CONNECT*,
- ATH - zakończenie połączenia (u obu użytkowników pokazuje się *NO CARRIER*),
- +++ - wejście do trybu komend (polega to na możliwości pisania w swojej konsoli zamiast tak jak zwykle, przeciwnika).
-

2 Elementy zaimplementowane

1. Podłączenie modemów i identyfikacja w menedżerze urządzeń.
2. Połączenie przez terminal PuTTY z drugim modemem i podstawowa komunikacja.
3. Program zawierający:
   - otwieranie i zamykanie portu,
   - wybieranie numeru,
   - dzwonienie na wybrany numer,
   - rozłączanie połączenia,
   - przyjmowanie połączenia,
   - wysyłanie wiadomości tekstowych.

3 Przebieg zajęć

Najpierw zostało wykonane sprawdzanie połączenia portów fizycznych w komputerze i modemie (COM oraz telefoniczne). Nastąpiła próba połączenia poprzez terminal PuTTY i wywołanie podstawowych komend Hayesa (AT). Próba nawiązania połączenia z drugim urządzeniem zakończyła się sukcesem, obie strony mogły być zarówno odbiorcą jak i wykonującym połączenie. Następnym krokiem było napisanie programu w języku C# (z elementami Forms z .NET). Udało się zaimplementować w trakcie zajęć połączenie z portem (i zamykanie portu), odbieranie połączeń, kończenie ich, wybieranie numeru oraz wykonywanie połączeń. Jako osobne pole tekstowe z przyciskiem stworzono wysyłanie wiadomości.

4 Program

Program został napisany w języku C# z elementami Forms (.NET). Do korzystania z portu użyta została biblioteka wewnętrzna `System.IO.Ports`.

4.1 Kod

4.2 Program.cs

Ten plik odpowiada za uruchomienie aplikacji okna (jest generowany przez środowisko):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
namespace ModemModem
{
    static class Program
    {
        static void Main()
        {
            SerialDataReceivedEventHandler(serialPort_DataReceived);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

4.3 Form1.cs

Główny plik programu, w nim wykonywane są wszystkie przypisania, obliczenia i wywołania

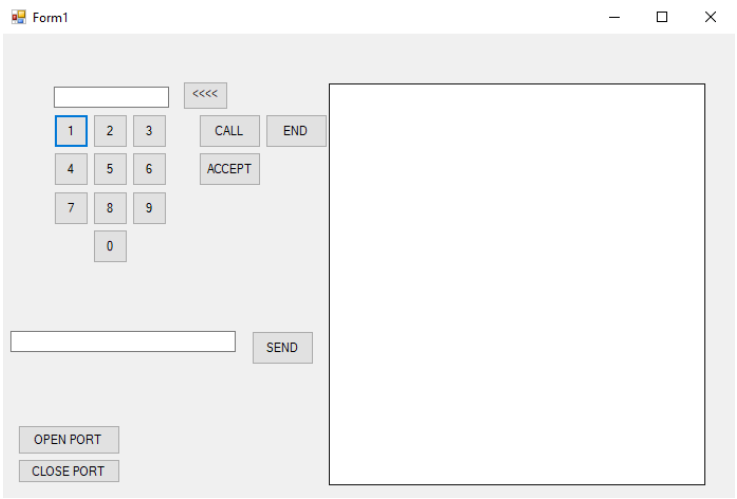
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
namespace ModemModem
{
    public partial class Form1 : Form
    {
        SerialPort _serialPort = new SerialPort("COM1", 9600, Parity.None, 8,
    StopBits.One);
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            textBoxPhoneNumber.Text += '1';
        }
        private void button2_Click(object sender, EventArgs e)
        {
            textBoxPhoneNumber.Text += '2';
        }
        private void button3_Click(object sender, EventArgs e)
        {
            textBoxPhoneNumber.Text += '3';
        }
        private void button4_Click(object sender, EventArgs e)
```

```
33          {
34              textBoxPhoneNumber.Text += '4';
35          }
36          private void button5_Click(object sender, EventArgs e)
37          {
38              textBoxPhoneNumber.Text += '5';
39          }
40          private void button6_Click(object sender, EventArgs e)
41          {
42              textBoxPhoneNumber.Text += '6';
43          }
44          private void button7_Click(object sender, EventArgs e)
45          {
46              textBoxPhoneNumber.Text += '7';
47          }
48          private void button8_Click(object sender, EventArgs e)
49          {
50              textBoxPhoneNumber.Text += '8';
51          }
52          private void button9_Click(object sender, EventArgs e)
53          {
54              textBoxPhoneNumber.Text += '9';
55          }
56          private void button10_Click(object sender, EventArgs e)
57          {
58              textBoxPhoneNumber.Text += '0';
59          }
60          private void buttonBack_Click(object sender, EventArgs e)
61          {
62              if(textBoxPhoneNumber.Text.Length > 0)
63              textBoxPhoneNumber.Text = textBoxPhoneNumber.Text.Substring(0,
    textBoxPhoneNumber.Text.Length - 1);
64          }
65          private void buttonCall_Click(object sender, EventArgs e)
66          {
67              string comendCall = "atd" + textBoxPhoneNumber.Text + "\r";
68              MessageBox.Show(comendCall, "numer");
69              _serialPort.Write(comendCall);
70          }
71          private void buttonEnd_Click(object sender, EventArgs e)
72          {
73              string comendCall = "+++ath\r";
74              MessageBox.Show(comendCall, "ending CALL");
75              _serialPort.Write(comendCall);
76          }
77          private void textBox1_TextChanged(object sender, EventArgs e)
78          {
79          }
80          public void dataReceived(object sender, SerialDataReceivedEventArgs e)
81          {
82              var data = _serialPort.ReadExisting();
83              if (data.Contains("CON"))
84              {
85                  polaczono = true;
86                  WypiszNaEkran("Polaczono z drugim modemem.");
87              }
88              WypiszNaEkran("<-" + data + "\n");
89              }
90          public void WypiszNaEkran(string data)
91          {
92              if (this.InvokeRequired)
93              {
94                  this.Invoke(
```

```
                    new MethodInvoker(
                    delegate ()
                    {
                        textBox1.Text += data + "\r";
                        textBox1.SelectionStart = textBox1.TextLength;
                        textBox1.ScrollToCaret();
                    }));
            }
        }
        private void buttonClosePort_Click(object sender, EventArgs e)
        {
            _serialPort.Close();
        }
        private void buttonAccept_Click(object sender, EventArgs e)
        {
            string comendCall = "ata\r";
            MessageBox.Show("","ACCEPT CALL");
            _serialPort.Write(comendCall);
        }
        private void buttonOpenPort_Click(object sender, EventArgs e)
        {
            _serialPort.Open();
            _serialPort.DataReceived += dataReceived;
        }
        private void Form1_Load(object sender, EventArgs e)
        {
        }
        private void textBoxSend_TextChanged(object sender, EventArgs e)
        {
        }
        private void buttonSend_Click(object sender, EventArgs e)
        {
            _serialPort.Write(textBoxSend.Text + "\r");
        }
    }
}
```

5 Okno programu

Po skompilowaniu i uruchomieniu program prezentuje się następująco (należy otworzyć port, a następnie wybrać numer i wykonać połączenie lub odebrać połączenie przychodzące):



Główne okno programu

6 Wnioski

Trudność zadania z modemem polegała głównie na nietypowej formie polegającej na współpracy z drugim zespołem. Każdorazowo, aby sprawdzić, czy program jest prawidłowo napisany, trzeba było nawiązać połączenie telefoniczne z sąsiednim modemem. Kolejną trudnością była detekcja, po której stronie wystąpił problem z wymianą wiadomości. Ostatecznie udało się zlokalizować problem poprzez terminal PuTTY.

7 Bibliografia

- https://en.wikipedia.org/wiki/Hayes_command_set
- https://pl.wikipedia.org/wiki/Modulacja
- http://nss.et.put.poznan.pl/study/projekty/sieci_komputerowe/modem_compol_iii/html/wprowadzenie.htm
- http://delibra.bg.polsl.pl/Content/37552/BCPS_41947_1982_Teledacja-w-resorcie.pdfhttp://forum.jdtech.pl/Watek-komendy-at-hayesa-podstawy-czyli-inna-komunikacja-z-modemem-wwan-lte
- https://www-users.mat.umk.pl//~zssz/SSO2001/Wyk4/sld054.htm