**Московский государственный технический университет им. Н.Э. Баумана**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Утверждаю: |  |
| Галкин В.А. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |

**Курсовая работа по дисциплине**

**«Сетевые технологии в АСОИУ»**

**«Локальная безадаптерная сеть»**

описание программы

(вид документа)

писчая бумага

(вид носителя)

15

(количество листов)

|  |  |  |
| --- | --- | --- |
| ИСПОЛНИТЕЛИ: | |  |
| студенты группы ИУ5-61Б: |  |
| Андреев А.В.  Ларичева М.В.  Пахомкин К.С. | "\_\_\_"\_\_\_\_\_\_\_\_\_2021 г. |

Москва - 2021 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оглавление

[1. Введение 3](#_Toc305591231)

[2. Forms 3](#_Toc305591232)

[3. Листинг 6](#_Toc305591236)

# Введение

Программный продукт написан с использованием технологии WinForms на языке программирования C#. Для создания графического интерфейса и взаимодействия с COM-портом использовались стандартные библиотеки и элементы управления.

Программа должна реализовывать функцию передачи коротких текстовых сообщений между ПК, соединенными через интерфейс RS-232C нуль-модемным кабелем. Дополнительные функции, не относящиеся к стандартным, приведены ниже.

# 2. Forms

**Chat.cs**

Главное окно программы, предназначенное для передачи сообщений, а также завершения работы программы.

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;

namespace WahChat

{

public partial class Chat : Form

{

public Chat()

{

InitializeComponent();

NetworkService.GetSharedService().chatBox = this.chatBox;

this.usernameLabel.Text = String.Format("Подключен как \"{0}\"", NetworkService.GetSharedService().currentSession.username);

this.chatBox.SelectionMode = SelectionMode.None;

}

private void SendButton\_Click(object sender, EventArgs e)

{

string message = messageBox.Text;

NetworkService.GetSharedService().SendMessage(message);

messageBox.Clear();

}

private void closeButton\_Click(object sender, EventArgs e)

{

NetworkService.GetSharedService().CloseConnection();

}

}

}

**Login.cs**

Окно программы, предназначенное для ввода имени пользователя, которое отображается в чате, и подключения.

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;

namespace WahChat

{

public partial class Login : Form

{

public Login()

{

InitializeComponent();

NetworkService.GetSharedService().notificationLabel = this.notificationLabel;

NetworkService.GetSharedService().connectButton = this.loginButton;

this.loginButton.Text = "Подключиться";

if (NetworkService.GetSharedService().currentConnection.isPortsOpened)

{

this.notificationLabel.Text = "Порты открыты";

}

else

{

this.notificationLabel.Text = "Ошибка открытия портов";

}

}

private void loginButton\_Click(object sender, EventArgs e)

{

int username = int.Parse(textBox.Text);

NetworkService.GetSharedService().CreateSession(username);

Frame frame = new Frame(Frame.Type.Link);

NetworkService.GetSharedService().SendFrame(frame);

this.notificationLabel.Text = "Отправка..";

this.Hide();

Chat chatForm = new Chat();

chatForm.Show();

}

}

}

**Setup.cs**

Окно программы, предназначенное для настройки соединения.

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;

namespace WahChat

{

public partial class Setup : Form

{

public Setup()

{

InitializeComponent();

}

private void Setup\_Load(object sender, EventArgs e)

{

// Показываем список COM-портов.

string[] portNames = NetworkService.GetSharedService().GetPortsNames();

foreach (string portName in portNames)

{

incomePortBox.Items.Add(portName);

outcomePortBox.Items.Add(portName);

}

}

private void connectButton\_Click(object sender, EventArgs e)

{

// Проверяем, выставлены ли порты.

if ((incomePortBox.SelectedItem != null) && (outcomePortBox.SelectedItem != null))

{

string incomePort = incomePortBox.SelectedItem.ToString();

string outcomePort = outcomePortBox.SelectedItem.ToString();

// Если порты одинаковые, то показываем ошибку.

if (incomePort == outcomePort)

{

MessageBox.Show("Выберите различные COM-порты", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

return;

}

// Стартуем соединение.

NetworkService.GetSharedService().CreateConnection(incomePort, outcomePort, checkBox.Checked);

this.Hide();

Login loginForm = new Login();

loginForm.Show();

}

else

{

MessageBox.Show("Оба COM-порта должны быть выбраны", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

}

}

# 3*.* Листинг

**Connection.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.IO.Ports;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WahChat

{

class Connection

{

public bool isPortsOpened = false;

private byte boundByte = 0xFF;

private SerialPort incomePort;

private SerialPort outcomePort;

public bool isMaster;

private List<byte> bytesBuffer = new List<byte>();

public Connection(string incomePortName, string outcomePortName, bool isMaster)

{

this.isPortsOpened = OpenPorts(incomePortName, outcomePortName, isMaster);

// ..

}

/// <summary>

/// Открытие портов

/// </summary>

private bool OpenPorts(string incomePortName, string outcomePortName, bool isMaster)

{

// Создаем объекты портов.

this.incomePort = new SerialPort(incomePortName);

this.outcomePort = new SerialPort(outcomePortName);

this.isMaster = isMaster;

// Настраиваем порты.

this.incomePort.Parity = Parity.Even;

this.incomePort.Handshake = Handshake.RequestToSend;

this.incomePort.BaudRate = 9600;

//this.incomePort.ReadBufferSize = 4 \* 1024; // TODO: Надо пересчитать размер буфера.

this.incomePort.DataReceived += new SerialDataReceivedEventHandler(RecieveBytes);

this.outcomePort.Parity = Parity.Even;

this.outcomePort.Handshake = Handshake.RequestToSend;

this.outcomePort.BaudRate = 9600;

// Открываем порты.

this.incomePort.Open();

this.outcomePort.Open();

return (this.incomePort.IsOpen && this.outcomePort.IsOpen);

}

/// <summary>

/// Закрытие портов

/// </summary>

private bool ClosePorts()

{

// Закрываем порты.

this.incomePort.Close();

this.outcomePort.Close();

return (this.incomePort.IsOpen && this.outcomePort.IsOpen);

}

/// <summary>

/// Пересылка байтов

/// </summary>

public void SendBytes(List<byte> list)

{

// TODO: Кодирование

List<byte> hamm = list;

// Делаем так, чтобы внутри кадра не встречалось boundByte.

List<byte> safeList = new List<byte>(hamm.Count);

foreach (var b in hamm)

{

if ((b & 0x7F) == 0x7F)

{

safeList.Add(0x7F);

safeList.Add((byte)(b & 0x80));

}

else

{

safeList.Add(b);

}

}

// Добавляем стартовый и конечный байт

safeList.Insert(0, boundByte);

safeList.Add(boundByte);

if (this.outcomePort.WriteBufferSize - this.outcomePort.BytesToWrite < safeList.Count)

{

// Если сообщение не влезло в порт, то надо что-то с этим делать.

// То ли очередь придумать, то ли ещё что.

return;

}

byte[] arr = safeList.ToArray();

this.outcomePort.Write(arr, 0, arr.Length);

}

/// <summary>

/// Получение байтов

/// </summary>

public void RecieveBytes(object sender, SerialDataReceivedEventArgs e)

{

int bytes = incomePort.BytesToRead;

byte[] comBuffer = new byte[bytes];

// Записываем в массив данные от ком порта.

incomePort.Read(comBuffer, 0, bytes);

foreach (byte incomeByte in comBuffer)

{

if (incomeByte == boundByte)

{

if (this.bytesBuffer.Count > 0)

{

NetworkService.GetSharedService().HandleMessage(this.bytesBuffer);

}

this.bytesBuffer = new List<byte>();

}

else

{

this.bytesBuffer.Add(incomeByte);

}

}

}

}

}

**CycleCoder.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace WahChat

{

class CycleCoder

{

/// <summary>

/// Процедура кодирования циклическим кодом с образующим полиномом g(x)=x^3+x+1

/// </summary>

/// <param name="a">информационные разряды</param>

/// <param name="n">образующий полином</param>

/// <param name="code">результат кодовая комбинация</param>

static void CycleCode(byte a, byte n, out byte code)

{

a <<= 3;

// умножаем исходную код

// считаем число разрядов

while (CountBit(a) != CountBit(n)) n <<= 1; // уравняли по двоичному порядку делимое и делитель цикла while-необходим для получения остатка от деления v(x)\*x^n-k на образующий полином.

code = a;

do // цикл, когда a> 111,все что меньше идет в остаток

{

a ^= n;

do

{

n >>= 1;

}

while (CountBit(a) != CountBit(n));

} while (a > 7);

code |= a; // приписываем остаток в начало кодового вектора

}

static int CountBit(byte a)

{

int count = 0;

while (!(a == 0))

{

a /= 2;

++count;

}

return (count);

}

static bool CycleDecode(byte a, out byte code, byte n)

{

while (CountBit(a) != CountBit(n)) n <<= 1; // уравняли по двоичному порядку делимое и делитель цикла while-необходим для получения остатка от деления v(x)\*x^n-k на образующий полином.

code = a;

do // цикл, когда a> 111, все что меньше идет в остаток

{

code ^= n;

do

{

n >>= 1;

}

while (CountBit(code) != CountBit(n));

}

while (code > 7);

if (code == 0)

{

code = (byte)(a >> 3);

return (true);

}

else return (false);

}

/// <summary>

/// Процедура кодирования строки текста

/// </summary>

/// <param name="str">строка из текстбокса</param>

static public List<byte> CodeString(string str)

{

char[] ChrArr;

byte LShort, HShort;

byte LWord, HWord; // младшая и старшие части символа

byte Gx = 11; // образующий полином

byte Code;

List<byte> CodingString = new List<byte>();

ChrArr = str.ToCharArray();

short[] ShortArr = new short[str.Length];

for (int i = 0; i < ChrArr.Length; i++)

{

ShortArr[i] = (short)ChrArr[i];

}

for (int i = 0; i < ShortArr.Length; i++)

{

LShort = (byte)(ShortArr[i] & 0x00FF); // получаем младшие 4 разряда маской 0000000011111111;

HShort = (byte)((ShortArr[i] & 0xFF00) >> 8); // получаем старшие (маска 1111111100000000) и сдвигаем вначало

LWord = (byte)(LShort & 0x0F); // получаем младшие 4 разряда маской 00001111;

HWord = (byte)((LShort & 0xF0) >> 4); // получаем старшие (маска 11110000) и сдвигаем вначало

CycleCode(LWord, Gx, out Code);

CodingString.Add(Code);

CycleCode(HWord, Gx, out Code);

CodingString.Add(Code);

LWord = (byte)(HShort & 0x0F); // получаем младшие 4 разряда маской 00001111;

HWord = (byte)((HShort & 0xF0) >> 4); // получаем старшие (маска 11110000) и сдвигаем вначало

CycleCode(LWord, Gx, out Code);

CodingString.Add(Code);

CycleCode(HWord, Gx, out Code);

CodingString.Add(Code);

}

return (CodingString);

}

/// <summary>

/// Метод декодирования исходной строки байтов в строку, введенную пользователем

/// </summary>

/// <param name="str">закодированая строка байтов</param>

/// <returns></returns>

static public string DecodeString(byte[] str)

{

char ch;

string DecodedString = "";

byte LWord, HWord; // младшая и старшие части символа

short LShort, HShort;

byte Gx = 11; // образующий полином

for (int i = 0; i < str.Length; i++)

{

if (!CycleDecode(str[i], out LWord, Gx)) return ("");

if (!CycleDecode(str[++i], out HWord, Gx)) return ("");

LShort = (short)((HWord << 4) | LWord);

if (!CycleDecode(str[++i], out LWord, Gx)) return ("");

if (!CycleDecode(str[++i], out HWord, Gx)) return ("");

HShort = (short)((HWord << 4) | LWord);

ch = Convert.ToChar((HShort << 8) | LShort);

Console.WriteLine("ch={0}", ch);

DecodedString += ch;

}

return (DecodedString);

}

}

}

**Frame.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WahChat

{

class Frame

{

/// <summary>

/// Тип кадра

/// </summary>

public enum Type: byte

{

/// Установка логического соединения.

Link,

/// Получение списка пользователей.

Ask,

/// Отправка сообщений пользователя.

Data,

/// Отправка запроса на переотправку сообщения.

Error,

/// Разъединение соединения.

Downlink

}

public Type type;

public List<byte> data;

public int authorID;

public string message;

public Frame()

{

// ..

}

public Frame(List<byte> data)

{

this.data = data;

byte typeByte = data[0];

switch (typeByte)

{

case (byte)Type.Link:

this.type = Type.Link;

break;

case (byte)Type.Ask:

this.type = Type.Ask;

break;

case (byte)Type.Data:

this.type = Type.Data;

this.authorID = (int)data[1];

byte[] byteArray = data.ToArray();

int messageLength = data.Count - 2;

byte[] messageData = new byte[messageLength];

Array.Copy(byteArray, 2, messageData, 0, messageLength);

this.message = System.Text.Encoding.UTF8.GetString(messageData, 0, messageData.Length);

break;

case (byte)Type.Error:

this.type = Type.Error;

break;

case (byte)Type.Downlink:

this.type = Type.Downlink;

break;

default:

this.type = Type.Link;

break;

}

}

public Frame(Type type)

{

List<byte> data = new List<byte>();

data.Add((byte)type);

this.data = data;

}

}

}

**NetworkService.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.IO.Ports;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WahChat

{

class NetworkService

{

public Label notificationLabel;

public Button connectButton;

public ListBox chatBox;

private NetworkService()

{

// ..

}

private static readonly NetworkService \_sharedService = new NetworkService();

public static NetworkService GetSharedService()

{

return \_sharedService;

}

/// Текущее соединение

public Connection currentConnection;

/// Текущая сессия

public Session currentSession;

/// <summary>

/// Создание соединения

/// </summary>

public void CreateConnection(string incomePortName, string outcomePortName, bool isMaster)

{

this.currentConnection = new Connection(incomePortName, outcomePortName, isMaster);

// формирование LINK кадра..

// отправка LINK кадра..

}

/// <summary>

/// Закрытие соединения

/// </summary>

public void CloseConnection()

{

Frame frame = new Frame(Frame.Type.Downlink);

this.SendFrame(frame);

}

/// <summary>

/// Обработка пришедшего сообщения

/// </summary>

public void HandleMessage(List<byte> message)

{

Frame frame = new Frame(message);

this.HandleFrame(frame);

}

/// <summary>

/// Обработка пришедшего кадра

/// </summary>

public void HandleFrame(Frame frame)

{

switch (frame.type)

{

case Frame.Type.Link:

this.notificationLabel.Invoke((MethodInvoker)delegate {

// Running on the UI thread

this.notificationLabel.Text = "Соединение установлено";

this.connectButton.Text = "Войти";

});

// Если станция не ведущая, то отправляем дальше

if (currentConnection.isMaster == false)

{

this.SendFrame(frame);

}

break;

case Frame.Type.Ask:

// Если станция не ведущая, то отправляем дальше

if (currentConnection.isMaster == false)

{

this.SendFrame(frame);

}

break;

case Frame.Type.Data:

this.chatBox.Invoke((MethodInvoker)delegate {

// Running on the UI thread

this.chatBox.Items.Add(string.Format("{0} ({1}) {2}", DateTime.Now.ToString("hh:mm"), frame.authorID, frame.message));

});

// Если станция не ялвяется отправителем, то отправляем дальше

if (currentSession.username != frame.authorID)

{

this.SendFrame(frame);

}

break;

case Frame.Type.Error:

// Если станция не ведущая, то отправляем дальше

if (currentConnection.isMaster == false)

{

this.SendFrame(frame);

}

break;

case Frame.Type.Downlink:

// Если станция не ведущая, то отправляем дальше

if (currentConnection.isMaster == false)

{

this.SendFrame(frame);

}

System.Windows.Forms.Application.Exit();

break;

}

}

public void SendFrame(Frame frame)

{

this.currentConnection.SendBytes(frame.data);

}

public void SendMessage(string message)

{

byte[] byteStr = System.Text.Encoding.UTF8.GetBytes(message);

List<byte> data = new List<byte>();

data.Add((byte)Frame.Type.Data);

data.Add((byte)currentSession.username);

foreach (byte b in byteStr)

{

data.Add(b);

}

Frame frame = new Frame();

frame.data = data;

this.SendFrame(frame);

}

/// <summary>

/// Список доступных портов

/// </summary>

public string[] GetPortsNames()

{

return SerialPort.GetPortNames();

}

/// <summary>

/// Создание сессии

/// </summary>

public void CreateSession(int username)

{

this.currentSession = new Session(username);

// формирование кадра c username..

// отправка кадра c username..

}

/// <summary>

/// Закрытие сессии

/// </summary>

public void CloseSession()

{

// ..

}

}

}

**Session.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WahChat

{

class Session

{

public int username;

public Session(int username)

{

this.username = username;

}

}

}