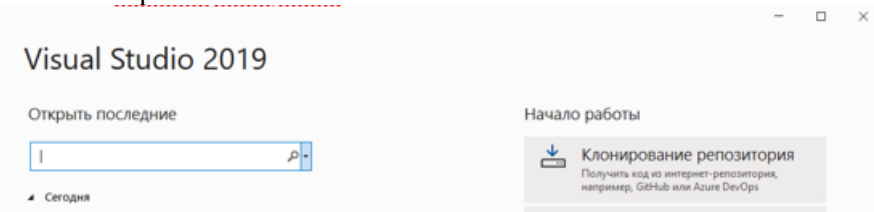
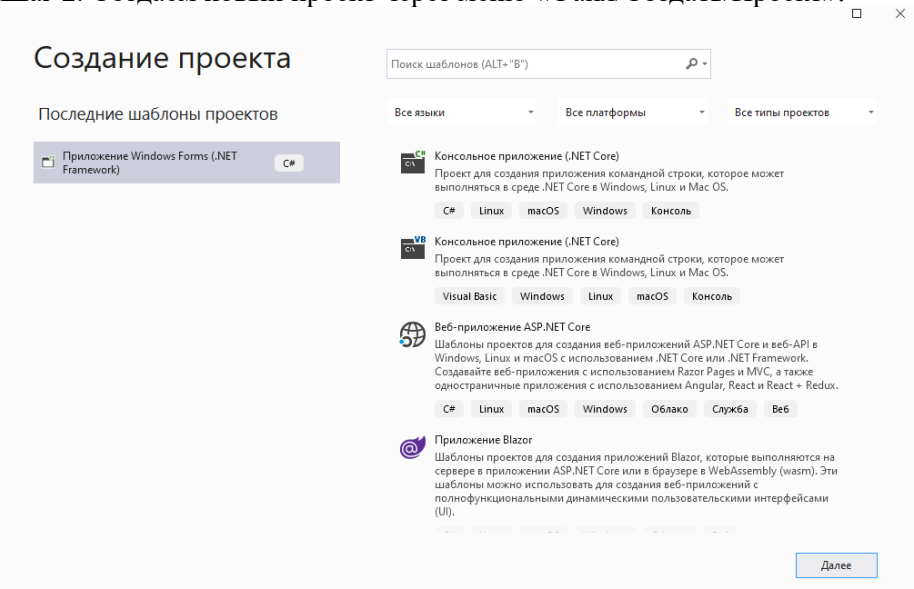
Лабораторная работа №2 – часть 2. Работа с сокетами UDP в .NET

Шаг 1. Открываем Microsoft Visual Studio



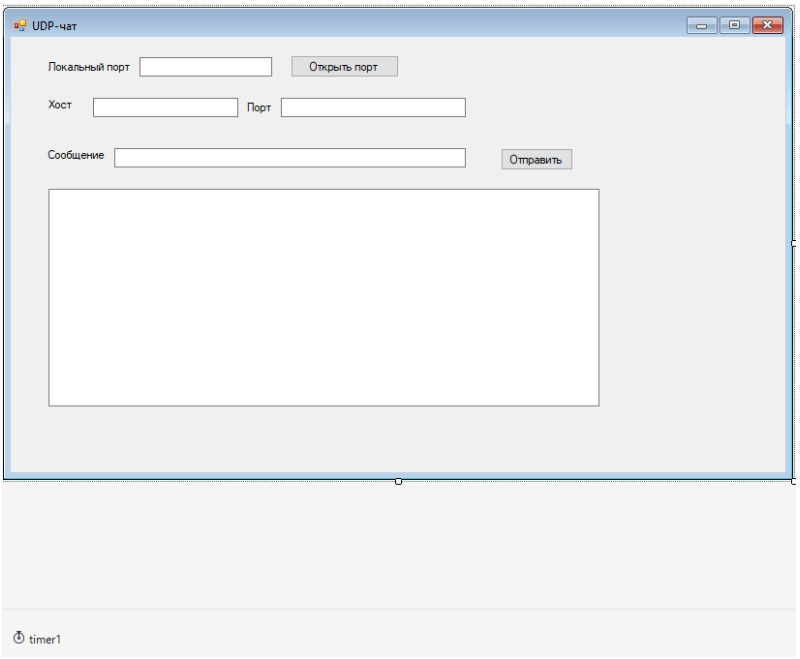
Шаг 2. Создаем новый проект через меню «Файл/Создать/Проект».



Выбираем шаблон Visual C#, приложение Windows Forms. В поле «Имя» вводим имя проекта, например, «UDP-чат». В поле «Расположение» указываем папку для хранения проекта. После этого нажимаем «ОК».

На рисунке показан визуальный интерфейс, который необходимо создать:

- текстовые поля textBoxLocalPort (локальный порт), textBoxHost (хост), textBoxPort (порт),
- textBoxMessage (сообщение), textBoxLog (поле текстового вывода);
- кнопки buttonBind (открыть порт) и buttonSend (отправить);
- таймер timer1 (класс Timer) – невизуальный компонент, который позволяет вызывать свой обработчик Tick через равные промежутки времени; для этого компонента установите свойство Enabled равным false, а свойство Interval равным 100 (время измеряется в миллисекундах); компонент добавляется на форму помещением его в любую часть формы и отображается в виде значка под формой.



Шаг 3. Создание исходного кода формы: подключение модулей и создание переменной сокета

Для начала, перед написанием обработчиков событий необходимо подключить к файлу исходного кода формы две библиотеки: System.Net и System.Net.Sockets.

Далее необходимо объявить переменную, в которой будет храниться объект сокета (класс Socket). Эту переменную назовем socket, а поскольку она должна быть видна из всех методов класса формы Form1, то объявим эту переменную внутри класса формы.

```
Form1.cs | Form1.cs [Конструктор]
UDP-чат
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Threading.Tasks;
9   using System.Windows.Forms;
10  using System.Net;
11  using System.Net.Sockets;
12
13
14  namespace UDP_чат
15  {
        Ссылок: 2
16      public partial class Form1 : Form
17      {
18          //Объявление переменной socket
19          Socket socket;
```

Шаг 4. Создание исходного кода формы: создание UDP сокета

Для кнопки buttonBind создадим обработчик события Click. По нажатию кнопки программа будет создавать новый UDP-сокет и привязывать его к номеру порта, который пользователь до этого указал в поле textBoxLocalPort.

Внутри этого обработчика введем следующий код. Этот код открывает новый UDP-сокет и запускает таймер timer1, с помощью которого будем следить за появлением новых сообщений от других программ.

```
ссылка: 1
private void buttonBind_Click(object sender, EventArgs e)
{
    try
    { // 1. Преобразование текста из textBoxLocalPort в число LocalPort
        int LocalPort = Int32.Parse(textBoxLocalPort.Text);

        // 2. Создаем объект socket
        // AddressFamily.InterNetwork - для работы с протоколом IPv4
        // SocketType.Dgram - для работы с дейтаграммами - короткими сообщениями
        // ProtocolType.Udp - работа по протоколу Udp
        socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

        // 3. Создаем пару (адрес, порт) для открытия сокета
        // IPAddress.Any - сокет для работы со всеми интерфейсными картами компьютера
        IPEndPoint LocalPoint = new IPEndPoint(IPAddress.Any, LocalPort);

        // 4. Открываем сокет для адреса LocalEndPoint
        socket.Bind(LocalPoint);

        // 5. Включаем таймер для периодичесой проверки
        timer1.Enabled = true;

        // 6. Вывод сообщения об успешном открытии сокета
        textBoxLog.AppendText("Открыт UDP порт  " + textBoxLocalPort.Text + "\n");
    }
    catch (Exception exc)
    {
        textBoxLog.AppendText(exc.Message+"\n");
    }
}
```

Шаг 5. Создание исходного кода формы: отправка данных

Для кнопки buttonSend создадим обработчик события Click. По нажатию кнопки программа будет отправлять сообщение с текстом их textBoxMessage адресату с координатами, указанными в textBoxHost и textBoxPort. Текстовое сообщение будет передаваться в кодировке UTF-8, что позволит передавать произвольные символы, включая русский текст.

```
private void buttonSend_Click(object sender, EventArgs e)
{
    try
    { // 1. Получаем список IP-адресов для указанного хоста
        IPAddress[] addresses = Dns.GetHostAddresses(textBoxHost.Text);

        if (addresses.Length > 0)
        {
            // Преобразование текста из textBoxPort в число port
            int port = Int32.Parse(textBoxPort.Text);

            // 2. Создаем пару (адрес, порт) для получателя сообщения
            // IPAddress.Any - сокет для работы со всеми интерфейсными картами компьютера
            IPEndPoint reciever = new IPEndPoint(addresses[0], port);

            // 3. Создаем буфер для отправки данных и помещаем в него байты текста из textBoxMessage в кодировке UTF-8
            byte[] data = Encoding.UTF8.GetBytes(textBoxMessage.Text);

            // 4. Отправляем данные
            socket.SendTo(data, reciever);

            // 5. Вывод сообщения об успешноq отправке
            textBoxLog.AppendText("Отправлено  " + textBoxHost.Text + ":" + textBoxPort.Text + ":" + textBoxMessage.Text + "\n");
        }
    }
    catch (Exception exc)
    {
        textBoxLog.AppendText(exc.Message + "\n");
    }
}
```

Шаг 6. Создание исходного кода формы: получение данных

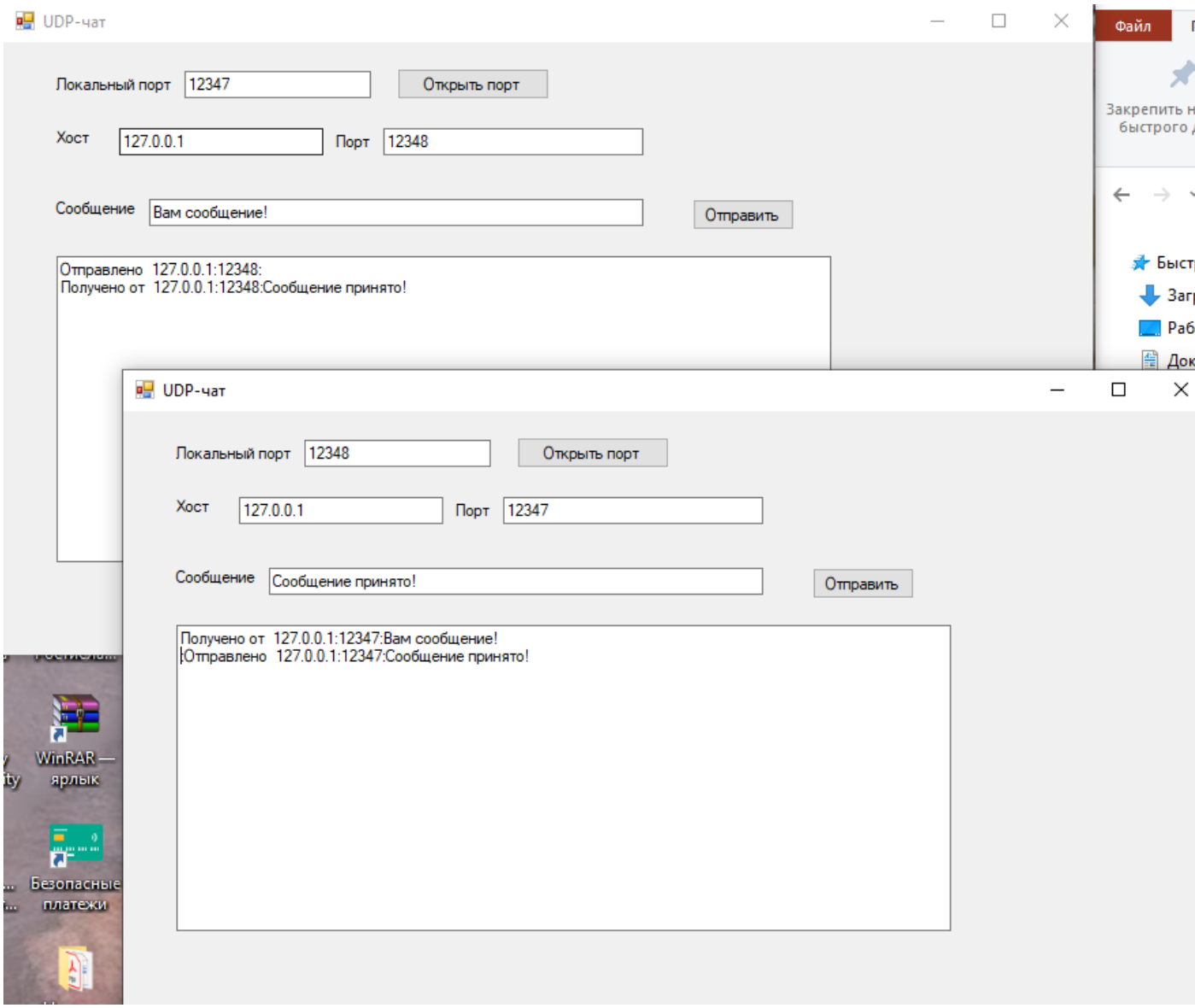
Для таймера timer1 создадим обработчик события Tick. После создания сокета этот обработчик будет вызываться каждый 100 мс и проверять наличие новых полученных данных. Если таковые будут иметься, то будет вызываться процесс получения сообщения и вывод его на экран.

```
private void timer1_Tick(object sender, EventArgs e)
{
    if (socket.Available > 0)
    {
        try
        { // 1. Создаем объект для хранения адреса-порта отправителя сообщения
            EndPoint dataSender = new IPEndPoint(IPAddress.Any,0);

          // 2. Создаем буфер для хранения полученных данных
            byte[] data = new byte[10000];

          // 3. Считываем сообщение
            int bytesRead = socket.ReceiveFrom(data, ref dataSender);
            if (bytesRead > 0) // сообщение точно получено!
            {
                // 4. Преобразуем байты в текст
                string recievedText = Encoding.UTF8.GetString(data,0,bytesRead);

                // 5. Вывод сообщения
                textBoxLog.AppendText("Получено от  " + dataSender.ToString() + ":" + recievedText + "\n");
            }

        }

        catch (Exception exc)
        {
            textBoxLog.AppendText(exc.Message + "\n");
        }
    }
}
```

Шаг 7. Тестирование программы

Для тестирования программы понадобятся 2 ее экземпляра. Для этого дважды выполните команду «Отладка/Запуск без отладки».

1. В одном приложении откройте порт 12347, а в другом 12348.
2. В первом приложении перешлите сообщение на адрес 127.0.0.1 (адрес «сам себе») и порт 12348.
3. Отправьте из второго приложения ответ первому.



Узнайте свой IP-адрес в локальной сети и адрес вашего соседа. Попробуйте вести переписку друг с другом. Используйте для этого порт 12347.

Попробуйте устроить массовую рассылку внутри локальной сети на порт 12347. Для этого в качестве адреса хоста надо указать адрес для групповой рассылки. Определить его можно следующим образом. Если ваш адрес, например, 192.168.1.13, а маска вашей сети 255.255.255.0, то адрес массовой рассылки будет равен 192.168.1.255 (часть адреса «192.168.1» - это адрес всей локальной сети, число «13» - это номер вашего компьютера в сети, а число «255» означает – рассылать всем в сети).