Лабораторная работа №1

Тема: Создание сетевого приложения на основе сокетов.
Цель работы: Создать приложение клиент – сервер.

Ход работы:

1. Передать на сервер полутоновое изображение и значение яркости от 0 до 255. На сервере подсчитать количество точек указанной яркости и вернуть результат клиенту.

Код программы:

Program.cs

```
using System;
using System.Windows.Forms;

namespace WinSocketDemo
{
    static class Program
    {
        static void Main()
        {
            Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new frmSettings());
            Application.Run(new MainForm());
        }
    }
 }
```

Worker.cs

```
using System;
using System.Threading;
using System.Drawing;

namespace WinSocketDemo
{
    class Worker
    {
        // Глобальный объект, характеризирующий персону в чате –
_____
```

```
        public static AppCore GlobalAppObject;
        // Настраивает объект выше
        public static void CreateMainObject(bool IsServer,
string ChatNick)
        {
            GlobalAppObject = new AppCore();
            // Назначаем ник
            GlobalAppObject.ChatNickProp = ChatNick;

            // Тип приложения
            if (IsServer)
            {
                GlobalAppObject.ApplicationTypeProp =
AppCore.AppType.Server;
            }
            else
            {
                GlobalAppObject.ApplicationTypeProp =
AppCore.AppType.Client;
            }
        }
        // Ф-ция, работающая в новом потоке: получение новых
сообщенй ———
        public static void GetMessages(Object obj)
        {
            Object[] Temp = (Object[])obj;

            System.Windows.Forms.ListBox ChatListBox =
(System.Windows.Forms.ListBox)Temp[0];

            // В бесконечном цикле получаем сообщения
            while (true)
            {
                // Ставим паузу, чтобы на время освобождать порт
для отправки сообщений
                Thread.Sleep(50);
                if (GlobalAppObject.ApplicationTypeProp ==
AppCore.AppType.Server)
                {
                    try
                    {
                        // Получаем сообщение от клиента
                        Bitmap Message =
SocketWorker.GetDataFromClient();
                        int[] GIST = new int[256];
                        double r;
                        int q;
                        for (int a = 0; a < Message.Width; a++)
```

```
                        {
                            for (int b = 0; b < Message.Height;
b++)
                            {
                                r = Message.GetPixel(a,
b).GetBrightness() * 255;
                                r = Math.Round(r, 0);
                                q = (int)r;
                                GIST[q]++;
                            }
                        }
                        SocketWorker.SendDataToClient(GIST);

                    }
                    catch { }
                }
                else
                {
                    try
                    {
                        int Message;
                        string tre = "";
                        Message =
SocketWorker.GetDataFromServer();

                        ChatListBox.Invoke((ThreadStart)delegate
()
                        {

ChatListBox.Items.Add(DateTime.Now.ToShortTimeString() + " " +
Message);
                        });
                    }
                    catch { }
                }
            }
        }    }
}
```

SocketWorker.cs

```
using System;
using System.Text;
using System.Net;
using System.Net.Sockets; // Для работы с сокетами нужно
подключить это пространство имен
using System.Windows.Forms;
using System.IO;
```

```
using System.Drawing;

namespace WinSocketDemo
{
    class SocketWorker
    {
        private static IPHostEntry ipHost; // Класс для сведений
об адресе веб-узла
        private static IPAddress ipAddr; // Предоставляет IP-
адрес
        private static IPEndPoint ipEndPoint; // Локальная
конечная точка

        private static Socket Server; // Создаем объект сокета-
сервера
        private static Socket Client; // Создаем объект сокета-
клиента
        private static Socket Handler; // Создаем объект
вспомогательного сокета

        // Деструктор
        ~SocketWorker()
        {
            // Вместо проверки сокетов на подключение, просто
используем блок try-catch
            try
            {
                // Сразу обрываем соединения
                Server.Shutdown(SocketShutdown.Both);
                // А потом закрываем сокет!
                Server.Close();

                Client.Shutdown(SocketShutdown.Both);
                Client.Close();

                Handler.Shutdown(SocketShutdown.Both);
                Handler.Close();
            }
            catch { }
        }
        // ————————————————————
        // Создание сокета
        public static bool IsConnected = false;
        public static void CreateSocket(Object obj)
        {
            Object[] TempObject = (Object[])obj;
            // IP-адрес сервера, для подключения
            string HostName = (string)TempObject[0];
```

```
            // Порт подключения
            string Port = (string)TempObject[1];
            bool ServerApp = (bool)TempObject[2];
            // Разрешает DNS-имя узла или IP-адрес в экземпляр
IPHostEntry.
            ipHost = Dns.Resolve(HostName);
            // Получаем из списка адресов первый (адресов может
быть много)
            ipAddr = ipHost.AddressList[0];
            // Создаем конечную локальную точку подключения на
каком-то порту
            ipEndPoint = new IPEndPoint(ipAddr,
int.Parse(Port));

            if (!ServerApp)
            {
                try
                {
                    // Создаем сокет на текущей машине
                    Client = new
Socket(AddressFamily.InterNetwork,
                    SocketType.Stream, ProtocolType.Tcp);
                    while (true)
                    {
                        // Пытаемся подключиться к удаленной
точке
                        Client.Connect(ipEndPoint);
                        if (Client.Connected)
                            IsConnected = true;
                        break;
                    }
                }
                catch (SocketException error)
                {
                    // 10061 — порт подключения занят/закрыт
                    if (error.ErrorCode == 10061)
                    {
                        MessageBox.Show("Порт подключения
закрыт!");
                        Application.Exit();
                    }
                }
            }
            else
            {
                try
                {
```

```
                    // Создаем сокет сервера на текущей машине
                    Server = new
Socket(AddressFamily.InterNetwork,
                    SocketType.Stream, ProtocolType.Tcp);
                }
                catch (SocketException error)
                {
                    // 10061 — порт подключения занят/закрыт
                    if (error.ErrorCode == 10061)
                    {
                        MessageBox.Show("Порт подключения
закрыт!");
                        Application.Exit();
                    }
                }

                // Ждем подключений
                try
                {
                    // Связываем удаленную точку с сокетом
                    Server.Bind(ipEndPoint);
                    // Не более 10 подключения на сокет
                    Server.Listen(10);

                    // Начинаем "прослушивать" подключения
                    while (true)
                    {
                        Handler = Server.Accept();
                        if (Handler.Connected)
                            IsConnected = true;
                        break;
                    }
                }
                catch (Exception e)
                {
                    throw new Exception("Проблемы с
подключением");
                }
            }
        }
        // ————————————————————
        // Получение данных от сервера
        public static int GetDataFromServer()
        {
            string GetInformation="";
            int[] inform = new int[256];
            // Создаем пустое “хранилище” байтов, куда будем
получать информацию
```

```
            byte[] GetBytes = new byte[2048];
            int BytesRec = Client.Receive(GetBytes);

            GetInformation +=
Encoding.Unicode.GetString(GetBytes, 0, BytesRec);

            // Переводим из массива битов в строку
            string[] a = GetInformation.Split(',');
            for (int j = 0; j < 256; j++)
            {
                inform[j] = Convert.ToInt32(a[j]);
            }
            int perem = inform[110];
            return perem;
        }
        // ____________________________
        // Получение информации от клиента
        public static Bitmap GetDataFromClient()
        {
            byte[] GetBytes = new byte[2048];
            int BytesRec = Handler.Receive(GetBytes);
            MemoryStream ms = new MemoryStream(GetBytes);
            Bitmap bmp = new Bitmap(ms);
            return bmp;
        }
        // ____________________________
        // Отправка информации на сервер
        public static void SendDataToServer(byte[] Data)
        {
            // Используем unicode, чтобы не было проблем с
кодировкой, при приеме информации
            Client.Send(Data);
        }
        // ____________________________
        // Отправка информации на клиент
        public static void SendDataToClient(int[] Data)
        {
            string mes="";
            //int[] result = new int[256];
            for (int i = 0; i < Data.Length; i++)
            {
                    mes += Data[i] + ",";
            }
            byte[] SendMsg = Encoding.Unicode.GetBytes(mes);
            Handler.Send(SendMsg);
        }
    }
}
```

Экранная форма приложения:

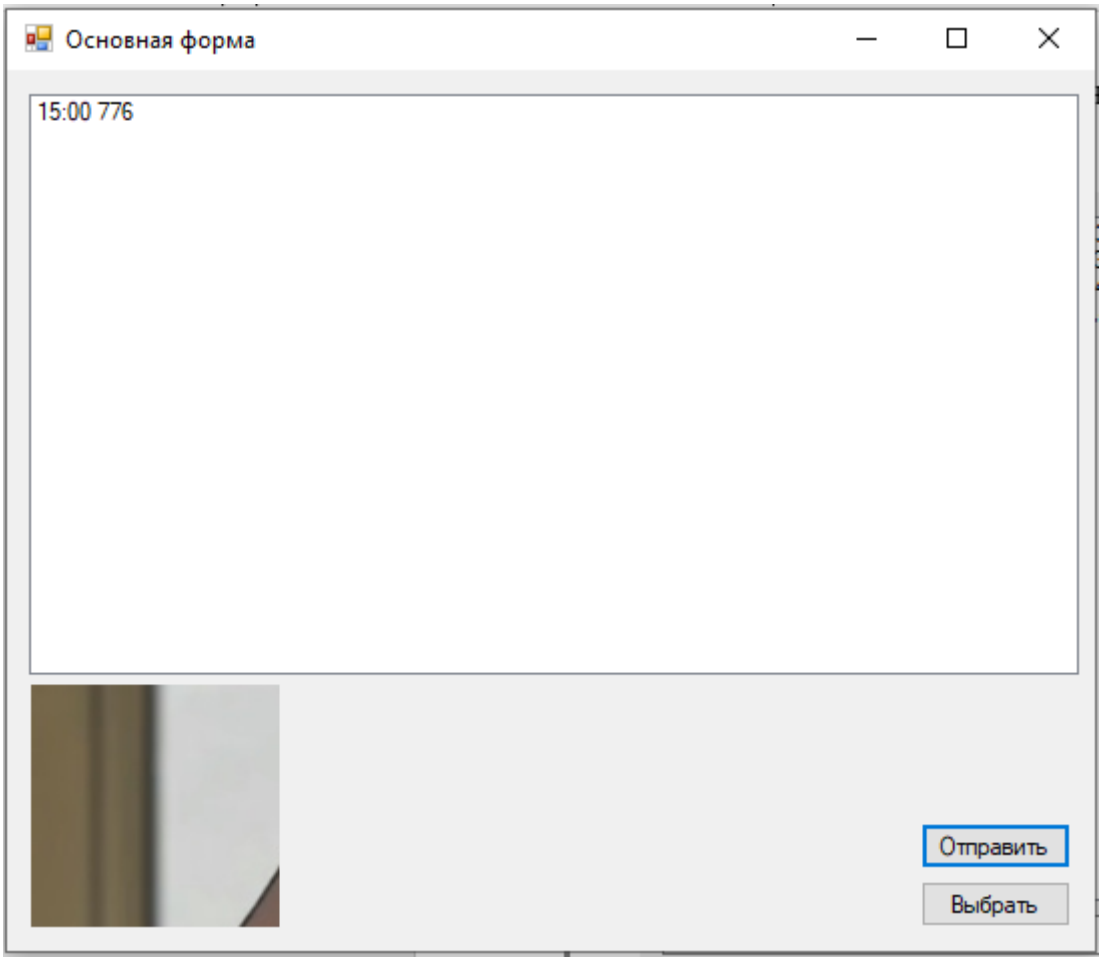

Рисунок 1 – Результат работы программы

Вывод: в ходе лабораторной работы было создано клиент – серверное приложение, а также выполнено практическое задание.