```
1  using System;
2  using System.Windows.Forms;
3  using System.IO.Ports;
4  using System.Threading;
5  using System.Net.Sockets;
6  using System.Net;
7
8  namespace four_in_one
9  {
10     public partial class Form1 : Form
11     {
12         private readonly SerialPort _serialPort = new SerialPort();
13         private Thread threadSerialPort;
14         private Thread threadTCPServer;
15         private Thread threadTCPClient;
16         private TcpListener server;
17         private NetworkStream stream1;
18         private NetworkStream stream2;
19         private byte[] bytes = new byte[256];
20         private byte[] bytesSP = new byte[256];
21         private bool ButtonccClicked = false;
22         private string msgfromClient = "";
23         private string completemsgfromClient = "";
24         private Int32 TCPport;
25         private string serverIP = "";
26         private TcpClient tcpClient2;
27         private string msgfromServer = "";
28         private string completeMsgfromServer = "";
29         private byte[] bytesfromServer = new byte[256];
30         private byte[] bytestoSend = new byte[256];
31         private bool formisclosing = false;
32         private string[] SPports;
33
34         public Form1()
35         {
36             InitializeComponent();
37             SelectCOMAdd();
38             _serialPort.WriteTimeout = 5000;
39             COMAdd();
40             this.FormClosing += Form1_FormClosing;
41             UpdateBaudRate();
42
43         }
44
45         private void UpdateBaudRate()
46         {
47             if(int.TryParse(SerialPortBaudrate.Text, out int baudRate))
48             {
49                 _serialPort.BaudRate = baudRate;
50             }
51             else
52             {
53                 _serialPort.BaudRate = 9600;
54             }
55         }
56
```

```
57
58          private void SPconnet_Click(object sender, EventArgs e)
59          {
60              if (SelectCOM.SelectedItem != null && !string.IsNullOrEmpty
                  (SelectCOM.SelectedItem.ToString()) && !string.IsNullOrEmpty
                  (SerialPortBaudrate.Text))
61              {
62                  _serialPort.PortName = SelectCOM.SelectedItem.ToString();
63                  UpdateBaudRate();
64                  threadSerialPort = new Thread(() => RunSerialport());
65                  threadSerialPort.Start();
66              }
67              else
68              {
69                  MessageBox.Show("select COM and set a Baudrate");
70              }
71
72          }
73
74          private void COMAdd()
75          {
76              SPports = SerialPort.GetPortNames();
77              foreach (string port in SPports)
78              {
79                  SelectCOM.Items.Add(port);
80              }
81          }
82
83          private void RunSerialport()
84          {
85              try
86              {
87                  _serialPort.Open();
88                  ShowInfo("Connected");
89                  while (true)
90                  {
91
92                      string data = "";
93                      string completeData = "";
94                      int i;
95                      while ((i=_serialPort.Read(bytesSP, 0, bytesSP.Length))!
                        =0)
96                      {
97                          ShowInfo("Message Received");
98                          data = System.Text.Encoding.ASCII.GetString(bytesSP,
                        0, i);
99                          completeData += data;
100                         if (data[data.Length - 1] == '\n')
101                         {
102                             ShowmsgInList("Message Received: " +
                        completeData);
103
104                             completeData = "";
105                         }
106                     }
107                 }
```

```
108             }
109             catch
110             {
111 
112                 if (!(formisclosing || ButtonccClicked))
113                 {
114                     ShowInfo("error by receiving message");
115                 }
116             }
117         }
118         private void Form1_FormClosing(object sender, FormClosingEventArgs e)
119         {
120             _serialPort.Close();
121             formisclosing = true;
122         }
123 
124         private void Confirm_Click(object sender, EventArgs e)
125         {
126             if (oSerialPort.Checked)
127             {
128                 panelSerialPort.Visible = true;
129                 panelTCP.Visible = false;
130                 MessageBox.Show("Please select COM and set a Baudrate");
131             }
132             else if (oTCP.Checked)
133             {
134                 panelSerialPort.Visible = false;
135                 panelTCP.Visible = true;
136 
137             }
138         }
139         private void SelectCOMAdd()
140         {
141             string[] ports = SerialPort.GetPortNames();
142             foreach (string port in ports)
143             {
144                 SelectCOM.Items.Add(port);
145             }
146         }
147 
148         private void StartTCPServer_Click(object sender, EventArgs e)
149         {
150             if (AlsTCPServer.Checked)
151             {
152                 server = new TcpListener(IPAddress.Any, 8000);
153                 threadTCPServer = new Thread(() => RunTCPserver(server));
154                 threadTCPServer.Start();
155             }
156             else
157             {
158                 MessageBox.Show("Please select 'als Server' option ");
159             }
160         }
161 
162 
163         private void RunTCPserver(TcpListener server)
```

```
164         {
165             server.Start();
166             ShowInfo("Started Server");
167             try
168             {
169                 while (true)
170                 {
171                     ShowInfo("Waiting for a connection... ");
172                     TcpClient tcpClient1 = server.AcceptTcpClient();
173                     ShowInfo("Connected to Client");
174                     stream1 = tcpClient1.GetStream();
175                     int i;
176                     while ((i = stream1.Read(bytes, 0, bytes.Length)) != 0)
177                     {
178                         msgfromClient = System.Text.Encoding.ASCII.GetString
                        (bytes, 0, i);
179                         completemsgfromClient += msgfromClient;
180                         ShowmsgInList("Reiceived: " + msgfromClient);
181                         ShowInfo("Received:");
182                     }
183                 }
184             }
185             catch (Exception)
186             {
187                 if (ButtonccClicked)
188                 {
189                     ShowInfo("connection closed.");
190                 }
191                 else
192                 {
193                     ShowInfo("connection failed. Please try again.");
194                 }
195             }
196         }
197 
198         private void ConnectToTCPServer_Click(object sender, EventArgs e)
199         {
200             if (AlsTCPClient.Checked)
201             {
202                 TCPport = int.Parse(PortTCP.Text);
203                 serverIP = IPTCP.Text;
204                 tcpClient2 = new TcpClient(serverIP, TCPport);
205                 threadTCPClient = new Thread(() => RunTCPClient());
206                 threadTCPClient.Start();
207             }
208             else
209             {
210                 MessageBox.Show("Please select 'als Client' option ");
211             }
212         }
213         private void RunTCPClient()
214         {
215             try
216             {
217                 ShowInfo("trying to connect");
218                 msgfromServer = null;
```

```
219                 stream2 = tcpClient2.GetStream();
220                 while (true)
221                 {
222                     int i;
223                     ShowInfo("Connected to Server");
224                     while ((i=stream2.Read(bytesfromServer, 0,
                    bytesfromServer.Length))!=0)
225                     {
226                         msgfromServer = System.Text.Encoding.ASCII.GetString
                    (bytesfromServer, 0, i);
227                         completeMsgfromServer += msgfromServer;
228                         if (msgfromServer[msgfromServer.Length-1] == '\n')
229                         {
230                             ShowmsgInList("Message Received: " +
                    completeMsgfromServer);
231                             completeMsgfromServer = "";
232                         }
233                     }
234                 }
235             }
236             catch
237             {
238                 //this.Invoke(new Action(() => { MessageBox.Show(this, "error
                      by receiving message from Server"); }));
239                 if (!(formisclosing || ButtonccClicked))
240                 {
241                     ShowInfo("error by receiving message from Server");
242                 }
243             }
244 
245         }
246 
247         private void ShowInfo(string info)
248         {
249             Info.Invoke(new Action(() => { Info.Text = info; }));
250         }
251 
252         private void ShowmsgInList(string msg)
253         {
254             listBox1.Invoke(new Action(() => { listBox1.Items.Add(msg); }));
255         }
256 
257         private void CloseConnection_Click(object sender, EventArgs e)
258         {
259             ButtonccClicked = true;
260             if (AlsTCPServer.Checked)
261             {
262                 server.Stop();
263                 stream1.Close();
264             }
265             else if (AlsTCPClient.Checked)
266             {
267                 tcpClient2.Close();
268                 stream2.Close();
269             }
270             else if (oSerialPort.Checked)
```

```
271             {
272                 _serialPort.Close();
273             }
274             ShowInfo("Connection closed");
275         }
276 
277         private void Send_Click(object sender, EventArgs e)
278         {
279             bytestoSend = System.Text.Encoding.ASCII.GetBytes
                  (MessageToSend.Text + "\n");
280             ShowmsgInList("Message sent: " + MessageToSend.Text);
281             if (oSerialPort.Checked)
282             {
283                 _serialPort.Write(bytestoSend, 0, bytestoSend.Length);

284             }
285             else if (oTCP.Checked)
286             {
287                 if (AlsTCPServer.Checked)
288                 {
289                    stream1.Write(bytestoSend, 0, bytestoSend.Length);
290                 }
291                 else if (AlsTCPClient.Checked)
292                 {
293                    stream2.Write(bytestoSend, 0, bytestoSend.Length);
294                 }
295             }
296             ShowInfo("Sent");
297         }
298 
299         private void ShowMyIP_Click(object sender, EventArgs e)
300         {
301             try
302             {
303                 MessageBox.Show(GetLocalIPAddress());
304             }
305             catch (Exception)
306             {
307 
308                 MessageBox.Show("No network adapters with an IPv4 address in
                      the system!");
309             }
310         }
311 
312 
313         public static string GetLocalIPAddress()
314         {
315             var host = Dns.GetHostEntry(Dns.GetHostName());
316             foreach (var ip in host.AddressList)
317             {
318                 if (ip.AddressFamily == AddressFamily.InterNetwork)
319                 {
320                     return ip.ToString();
321                 }
322             }
323             throw new Exception("No network adapters with an IPv4 address in
```

```
                the system!");
324
325         }
326     }
327 }
328
```