24.1.2.2 Socket-Projekte

Sockets als Schnittstelle zur Programmierung von Netzwerk-Anwendungen werden in allen drei Projekten verwendet, die Ihnen in diesem Kapitel vorgestellt werden. Es werden u.a. die beiden Klassen *ServerSocket* und *Socket* eingesetzt.

- Das erste Projekt besteht aus einem Server- und einem Client-Programm. Der Server bietet einen einfachen Chat-Service, der von mehreren Clients genutzt werden kann.
- Ein Server bietet im zweiten Projekt einen Zitat-Service an. Clients können sich zum Server verbinden und durch das Senden einer bestimmten Zeichenkette ein kurzes oder ein langes Zitat anfordern, das der Server an den Client ausliefert.
- Im dritten Projekt ruft ein Client von einem Zeit-Server im Internet auf dem Port 37 die aktuelle Zeit ab und gibt sie formatiert aus.

Für das letzte Projekt wird der Quelltext im Kapitel vollständig angegeben. Alle drei Projekt-Archive finden Sie im Downlad-Bereich.

24.1.2.2.1 Projekt 1 – Chat-Service

Im ersten Projekt werden im Chat-Server ein Server-Sockets sowie Sockets für jede laufende verbindung eingesetzt. Im Chat-Client wird lediglich ein Socket eingesetzt. Als Dienst bietet der Chat-Server auf Port 8088 einen einfachen Chat-Service an. Wenn Sie den Service nutzen wollen, dann müssen Sie die IP-Adresse des Servers kennen und den Port, unter dem der Server seinen Service anbietet. Der Chat-Server und ein Chat-Client A werden auf dem Rechner 1 (IP-Adresse 192.168.2.102) gestartet. Daher ist die Angabe der Server-IP-Adresse 127.0.0.1 (lokale Loopback-Adresse) bei Client A genauso korrekt wie die Alternative 192.168.2.102:

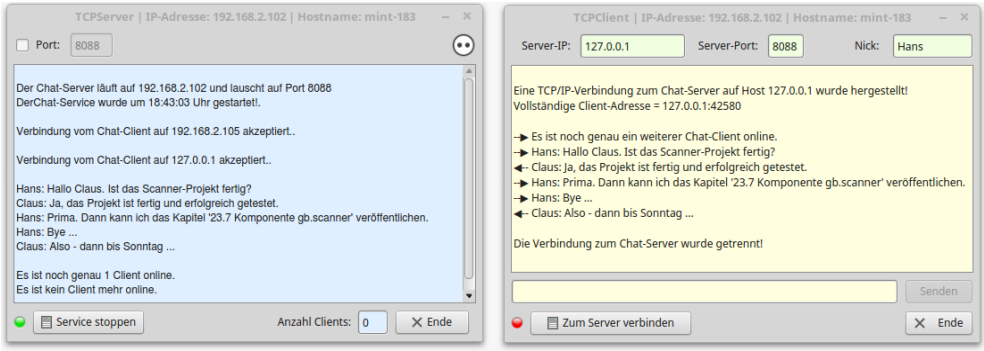

Abbildung 24.1.2.2.1: Chat-Server und Chat-Client A

Auf dem Server werden der komplette Chat-Verlauf und diverse Statusmeldungen protokolliert, während für die Clients der Chat-Verlauf sowie Statusmeldungen zur TCP-Verbindung angezeigt werden. Für einen Test der Server-Client-Applikation befindet sich auf dem Rechner 2 (Laptop) ein weiterer Chat-Client B. Der Client B auf dem Rechner 2 muss die Server-IP-Adresse 192.168.2.102 auswählen. Achtung: Die IP-Adresse des Servers hängt immer vom (lokalen) Netzwerk ab, in dem sich der Rechner mit dem Server befindet! Der Port ist bereits beim Server und den Clients auf 8088 voreingestellt.

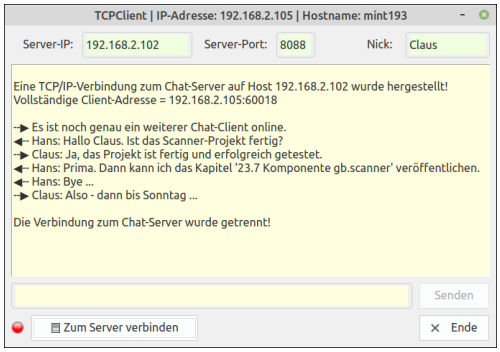

Abbildung 24.1.2.2.2: Chat-Client B auf Rechner 2

Der hinreichend kommentierte Quelltext wird nur in Auszügen angegeben:

```
[1]  Public Sub ServerSocket_Connection(RemoteHostIP As String)
[2]
[3]      Dim ConnectedSocket As Socket
[4]      Dim ConnectionSocket As Socket
[5]      Dim sBuffer As String
[6]      Dim k As Integer
[7]
[8]      If ServerSocket.Status <= Net.Active Then Return
[9]
[10] '-- Es wird ein *neuer* Socket auf dem Server erzeugt, wenn der Server eine Client-Verbindungsanfrage
[11] '-- akzeptiert hat. Dieser Socket wird zur Kommunikation zwischen Server und Client verwendet!
[12]     ConnectionSocket = ServerSocket.Accept()
[13]
[14] '-- Der neue Socket wird der Liste der aktiven ConnectionSockets auf dem Server hinzugefügt:
[15]     aConnetedSockets.Add(ConnectionSocket)
[16]
[17] '-- Wenn eine Verbindungsanfrage von einem neuen Client vom Server akzeptiert wurde, dann wird eine
[18] '-- Nachricht an alle mit dem ChatServer verbundenen Clients gesendet, nur nicht an den ersten Client.
[19]     If aConnetedSockets.Count > 1 Then
[20]        If aConnetedSockets.Count = 2 Then
[21]           sBuffer = "Es ist noch genau ein weiterer Chat-Client online."
[22]        Else
[23]           sBuffer = "Es sind noch weitere " & Str(aConnetedSockets.Count - 1)&" Chat-Clients online."
[24]        Endif
[25]        For k = 0 To aConnetedSockets.Max
[26]          ConnectedSocket = aConnetedSockets[k]
[27]          Write #ConnectedSocket, sBuffer, Len(sBuffer)
[28]        Next
[29]     Endif
[30]
[31]     txbCurrentClients.Text = Str(aConnetedSockets.Count)
[32]     LogMessage(Subst("Verbindung vom Chat-Client auf &1 &2",RemoteHostIP,"akzeptiert.") & gb.NewLine)
[33]
[34] End
[35]
[36] ''Socket ist der Socket, über den die Kommunikation Server <-> 'Aktiver Client' abgewickelt wird
[37] Public Sub Socket_Read()
[38]
[39]     Dim sBuffer As String
[40]     Dim ConnectionSocket As Socket
[41]     Dim ConnectedSocket As Socket
[42]
[43] '-- Es wird der zuletzt aktive Client ermittelt
[44]     ConnectionSocket = Last
[45] '-- Der Daten-Strom des zuletzt aktiven Clients wird ausgelesen und in `sBuffer` gespeichert
[46]     Read #ConnectionSocket, sBuffer, Lof(ConnectionSocket)
[47] '-- Run Service:
[48] '-- Die Chat-Nachricht des zuletzt aktiven Chat-Clients wird an *alle* mit dem Chat-Server
[49] '-- verbundenen Clients gesendet!
[50]     For Each ConnectedSocket In aConnetedSockets
[51]       Write #ConnectedSocket, sBuffer, Len(sBuffer)
[52]     Next
[53]
[54] '-- Die Chat-Nachricht wird auf dem Server protokolliert
[55]     LogMessage(sBuffer)
[56]
[57] End
```

24.1.2.2.2 Projekt 2 – Zitat-Service

Um den Zitat-Dienst des eingesetzten Servers zu nutzen und nicht nur eine Mitteilung zu erhalten, dass der angeforderte Service gegenwärtig nicht erreichbar ist, sollten Sie die (deutsche) Sammlung von Zitaten auf dem Rechner installieren, auf dem der Server gestartet wird:

```
sudo apt install fortune-mod      ' Zitat-Programm
sudo apt-get install fortunes-de  ' Zitat-Sammlung (DE)
```

Es folgt eine kurze Testbeschreibung für einen Server, der einen Zitat-Service anbietet und für einen Client auf dem gleichen Rechner:

- Zuerst werden der Zitat-Server und ein Client auf einem Rechner gestartet. Danach kann die Kommunikation – hier die Nutzung des Zitat-Services des Servers – beginnen.
- Mit -l (long) oder -s (short) fordert der Client ein langes oder ein kurzes Zitat an.
- Abschließend kann der Service des Servers beendet werden oder der Client sich vom Server trennen. In beiden Fällen werden diese Ereignisse protokolliert.

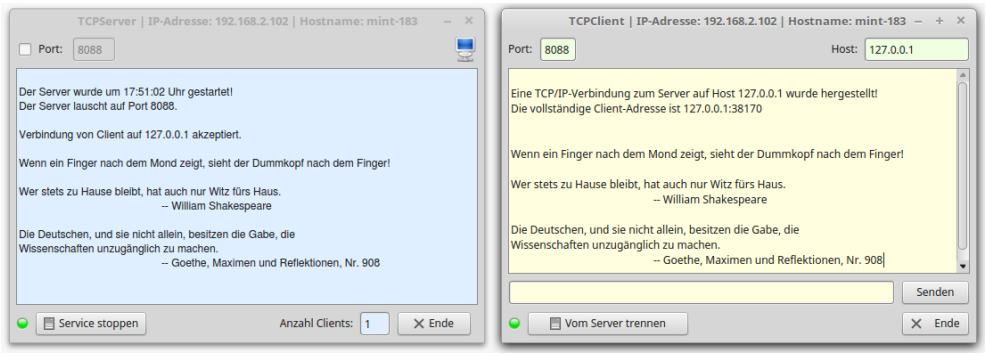

Abbildung 24.1.2.2.3: Zitat-Server und Client 1 in Aktion

Wenn Sie sich die Quelltexte für das Projekt 1 und Projekt 2 genauer ansehen, dann werden Sie feststellen, dass sie sich nur im angebotenen Dienst des Servers und in der Präsentation der übertragenen Daten beim Client unterscheiden. Damit bleibt viel Spielraum für Projekte mit eigenen Diensten!

24.1.2.2.3 Projekt 3 – Client für einen Zeit-Service

Im letzten Projekt nutzt ein Client den Zeit-Service eines ausgewählten Zeit-Servers im Internet auf dem Port 37. Die Liste in einer Combo-Box enthält vier Zeit-Server, die als gut verfügbar gelten. Wenn sich ein Client erfolgreich zum Zeit-Server verbunden hat, dann sendet dieser die aktuelle Zeit in einem speziellen Format und schließt sofort die TCP-Verbindung:

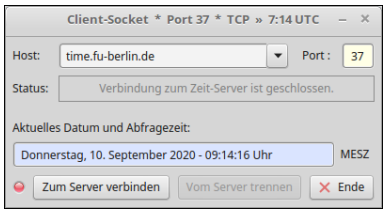

Abbildung 24.1.2.2.4: Client auf Port 37 – Ausgabe des aktuellen Datums und der aktuellen Zeit

Wenn der Zeit-Server nicht erreichbar ist, dann wird das angezeigt. Der Fehler wird hier initiiert, weil der Landeskenner mit 'ita' statt 'it' fehlerhaft angegeben wurde:

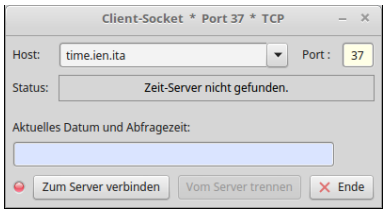

Abbildung 24.1.2.2.5: Client mit Status-Kommentar

Der Verbindungsaufbau wird nach einer im Quelltext fest eingestellten Zeit abgebrochen. Da die Eigenschaft *Socket.Timeout* nach Aussage von B. Minisini (September 2020) nicht als Verbindungstimeout von der Methode connect() verwendet wird, wird im Programm in den Zeilen 44 bis 53 eine eigene TimeOut-Prozedur definiert.
Offensichtlich kann unter der angegebenen IP-Adresse kein Zeit-Server in der vorgegebenen Zeitspanne (2 Sekunden) erreicht werden:

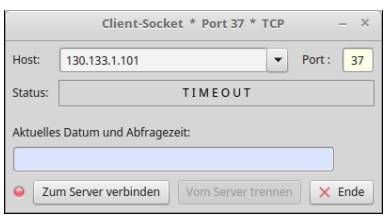

Abbildung 24.1.2.2.6: Client nach einem Timeout

Der Quelltext für den Client wird vollständig angegeben:

```
[1]  ' Gambas class file
[2]
[3]  '-- Liste deutscher Zeitserver: http://www.hullen.de/helmut/filebox/DCF77/ntpsrvr.html
[4]  '-- Test1:  130.133.1.101       -> Fehler TimeOut
[5]  '-- Test2:  time.fu-berlin.net  -> Fehler `Zeit-Server nicht gefunden`
[6]
[7]  Public lTimeResult As Long
[8]  Private iCurrentHost As Integer
[9]
[10] Public Sub Form_Open()
[11]     FMain.Center()
[12]     FMain.Resizable = False
[13]     SetLEDColor(picStatus, "red")
[14] '-- Vorgabe einer Liste mit Zeit-Servern
[15]     cmbHost.Add("134.130.4.17") '-- RWTH Aachen -> timeserver.rwth-aachen.de
[16]     cmbHost.Add("time.ien.it")
[17]     cmbHost.Add("130.133.1.10") '-- time.fu-berlin.de
[18]     cmbHost.Add("time.fu-berlin.de")
[19] '-- Auswahl eines stabilen deutschen Zeit-Servers
[20]     If cmbHost.List.Count > 3 Then cmbHost.Text = cmbHost.List[3]
[21]     lblStatus.Foreground = Color.Gray
[22]     lblStatus.Caption = "Verbindung zum Zeit-Server ist geschlossen."
[23] End
[24]
[25] Public Sub btnConnect_Click()
[26]     txbDateTime.Clear()
[27]     lblStatus.Foreground = Color.Black
[28]     btnConnect.Enabled = False
[29]     btnConnect.SetFocus()
[30] '-- TCP-Verbindung initialisieren (Host und Port)
[31]     TCP_Socket.Host = cmbHost.Text
[32]     If IsInteger(txbPort.Text) Then TCP_Socket.Port = Val(txbPort.Text)
[33]  '-- Verbindung zum Server herstellen
[34]     TCP_Socket.Connect()
[35]     If TCP_Socket.Status > Net.Inactive Then
[36]        btnDisconnect.Enabled = True
[37]        lblStatus.Text = "Verbindung zum Zeit-Server wird aufgebaut ..."
[38]        TimeOut.Delay = 2000 ' TimeOut = 2 Sekunden
[39]        TimeOut.Start()
[40]        SetLEDColor(picStatus, "green")
[41]     Endif
[42] End
[43]
[44] Public Sub TimeOut_Timer()
[45]     TimeOut.Stop()
[46]     If TCP_Socket.Status <> Net.Connected Or lTimeResult = 0 Then
[47]        If TCP_Socket.Status > 0 Then Close #TCP_Socket
[48]        Set_Interface(False)
[49]        lblStatus.Text = ("T I M E O U T")
[50]        SetLEDColor(picStatus, "red")
[51]        FMain.Caption = "Client-Socket  *  Port 37  *  TCP"
[52]     Endif
[53] End
[54]
[55] Public Sub TCP_Socket_Found()
[56] '-- Dieses Ereignis wird ausgelöst, wenn der Hostname erfolgreich
[57] '-- in die Host-IP-Adresse aufgelöst wurde
[58]     lblStatus.Caption = "Hostname in IP-Adresse aufgelöst ..."
[59] End
[60]
[61] Public Sub TCP_Socket_Ready()
[62] '-- Das Ereignis wird ausgelöst, nachdem eine Verbindung zwischen
[63] '-- Client und Server erfolgreich hergestellt wurde.
[64]     TimeOut.Enabled = False
[65]     lblStatus.Text = "Verbunden..." & TCP_Socket.Path
[66]     FMain.Enabled = True
[67]     Set_Interface(True)
[68] End
[69]
[70] Public Sub TCP_Socket_Read()
[71]     Dim sResult, sDatum, sZeit, sShellOutput As String
[72]     Dim i, iOffset As Integer
[73]     Dim iNTPSekunden As Long
[74]     Dim iCurrentSekunden As Integer
[75]     Dim iSekunden19001970 As Long
[76]     Dim dDate1970, dCurrentDateUTC As Date
[77]
[78] '-- In Deutschland gilt UTC+1 als Normalzeit und UTC+2 als Sommerzeit.
[79] '-- Der Zeitstempel im NTP ist 64 Bits lang.
[80] '-- 32 Bits kodieren die iNTPSekunden seit dem 1. Januar 1900 00:00:00 Uhr (!)
[81]     If TCP_Socket.Status = Net.Connected Then
[82]        Read #TCP_Socket, sResult, Lof(TCP_Socket)
[83]    '-- Umwandlung des übermittelten Datenwortes in eine ganze Zahl
[84]        For i = 1 To Len(sResult)
[85]          iNTPSekunden = iNTPSekunden * 256 + Asc(Mid$(sResult, i, 1))
[86]        Next
[87]     Endif
[88] '-- Alternative: Anzahl der Sekunden = b3*256³ + b2*256² + b1*256 + b0
[89]     lTimeResult = iNTPSekunden
[90]     iSekunden19001970 = DateDiff("01/01/1900", "01/01/1935", gb.Second) + DateDiff("01/01/1935", "01/01/1970 ", \
[91]                         gb.Second)
[92]     iCurrentSekunden = iNTPSekunden - iSekunden19001970
[93]     dDate1970 = Date(1970, 01, 01, 0, 0, 0)
[94] '-- Bestimmung der lokalen Zeit
[95]     Shell "date +%Z" To sShellOutput
```

```
[96]  '-- 0 => UTC, 3600 => Normalzeit (CET), 7200 => Sommerzeit -> Central European Summer Time (CEST)
[97]      If Trim(sShellOutput) = "CEST" Then
[98]         iOffset = 7200
[99]         lblTZ.Text = "MESZ"
[100]     Else
[101]        iOffset = 3600
[102]        lblTZ = "MEZ"
[103]     Endif
[104]     dCurrentDateUTC = DateAdd(dDate1970, gb.Second, iCurrentSekunden + iOffset)
[105]     sDatum = Format$(dCurrentDateUTC, "dddd") & ", " & Format$(dCurrentDateUTC, "d. mmmm yyyy")
[106]     sZeit = Format$(Time(dCurrentDateUTC), "hh:nn:ss") & " Uhr"
[107]     txbDateTime.Text = " " & sDatum & " - " & sZeit
[108]     iOffset = 0
[109]     dCurrentDateUTC = DateAdd(dDate1970, gb.Second, iCurrentSekunden + iOffset)
[110]     sZeit = Format$(Time(dCurrentDateUTC), "h:nn")
[111]     FMain.Caption = "Client-Socket  *  Port 37  *  TCP  " & String.Chr(187) & "  " & sZeit & " UTC"
[112]     SetLEDColor(picStatus, "red")
[113]
[114] End
[115]
[116] Public Sub TCP_Socket_Error()
[117]     Select Case TCP_Socket.Status
[118]       Case Net.CannotCreateSocket
[119]         lblStatus.Text = "Socket generieren nicht erlaubt."
[120]       Case Net.HostNotFound
[121]         lblStatus.Text = "Zeit-Server nicht gefunden."
[122]         FMain.Caption = "Client-Socket  *  Port 37  *  TCP"
[123]       Case Net.ConnectionRefused
[124]         lblStatus.Text = "Verbindung zum Zeit-Server nicht möglich."
[125]       Case Net.CannotRead
[126]         lblStatus.Text = "Fehler beim Lesen des Zeitstempels."
[127]     End Select
[128]     Set_Interface(False)
[129] End
[130]
[131] Public Sub TCP_Socket_Closed()
[132]     lblStatus.Foreground = Color.Gray
[133]     lblStatus.Caption = "Verbindung zum Zeit-Server ist geschlossen."
[134]     FMain.Enabled = True
[135]     Set_Interface(False)
[136] End
[137]
[138] Public Sub btnDisconnect_Click()
[139]     Try TCP_Socket.Close()
[140]     Set_Interface(False)
[141]     lblStatus.Text = "Verbindung zum Zeit-Server ist geschlossen."
[142]     SetLEDColor(picStatus, "red")
[143]     btnDisconnect.SetFocus()
[144] End
[145]
[146] Public Sub cmbHost_Click()
[147]     iCurrentHost = cmbHost.Index
[148] End
[149]
[150] Public Sub cmbHost_Change()
[151]     If TCP_Socket.Status > 0 Then Close #TCP_Socket
[152]     lblStatus.Foreground = Color.Gray
[153]     lblStatus.Caption = "Verbindung zum Zeit-Server ist geschlossen."
[154]     txbDateTime.Clear()
[155]     lblTZ.Text = ""
[156]     SetLEDColor(picStatus, "red")
[157] End
[158]
[159] Private Sub Set_Interface(bState As Boolean)
[160]     btnConnect.Enabled = Not bState
[161]     txbPort.Enabled = Not bState
[162]     btnDisconnect.Enabled = bState
[163]     TimeOut.Enabled = bState
[164]     SetLEDColor(picStatus, "red")
[165] End
[166]
[167] Private Sub SetLEDColor(picBox As PictureBox, sLEDColor As String)
[168]     picBox.Picture = Picture["LED/led_" & Lower(sLEDColor) & ".svg"]
[169] End
[170]
[171] Public Sub btnClose_Click()
[172]     FMain.Close()
[173] End
[174]
[175] Public Sub Form_Close()
[176]     If TCP_Socket.Status > 0 Then Close #TCP_Socket
[177] End
```

Hinweise:

- Es werden vier stabile Zeitserver verwendet, von denen bekannt ist, dass sie zuverlässig ihren Dienst anbieten.
- Unter http://www.hullen.de/helmut/filebox/DCF77/ntpsrvr.html finden Sie eine Liste deutscher Zeitserver, die von Helmut Hullen zusammengestellt wurde.
- Die Hauptlast im Programm trägt das Ereignis *TCP_Socket_Read()*. Zuerst wird die Zeit ausgelesen, dann bearbeitet und abschließend formatiert als Sommerzeit oder als Normalzeit ausgegeben (→ Abbildung 24.1.2.2.4).