2022년 IoT기반 스마트 솔루션 개발자 양성과정

Embedded Application

14-TCP Local Server

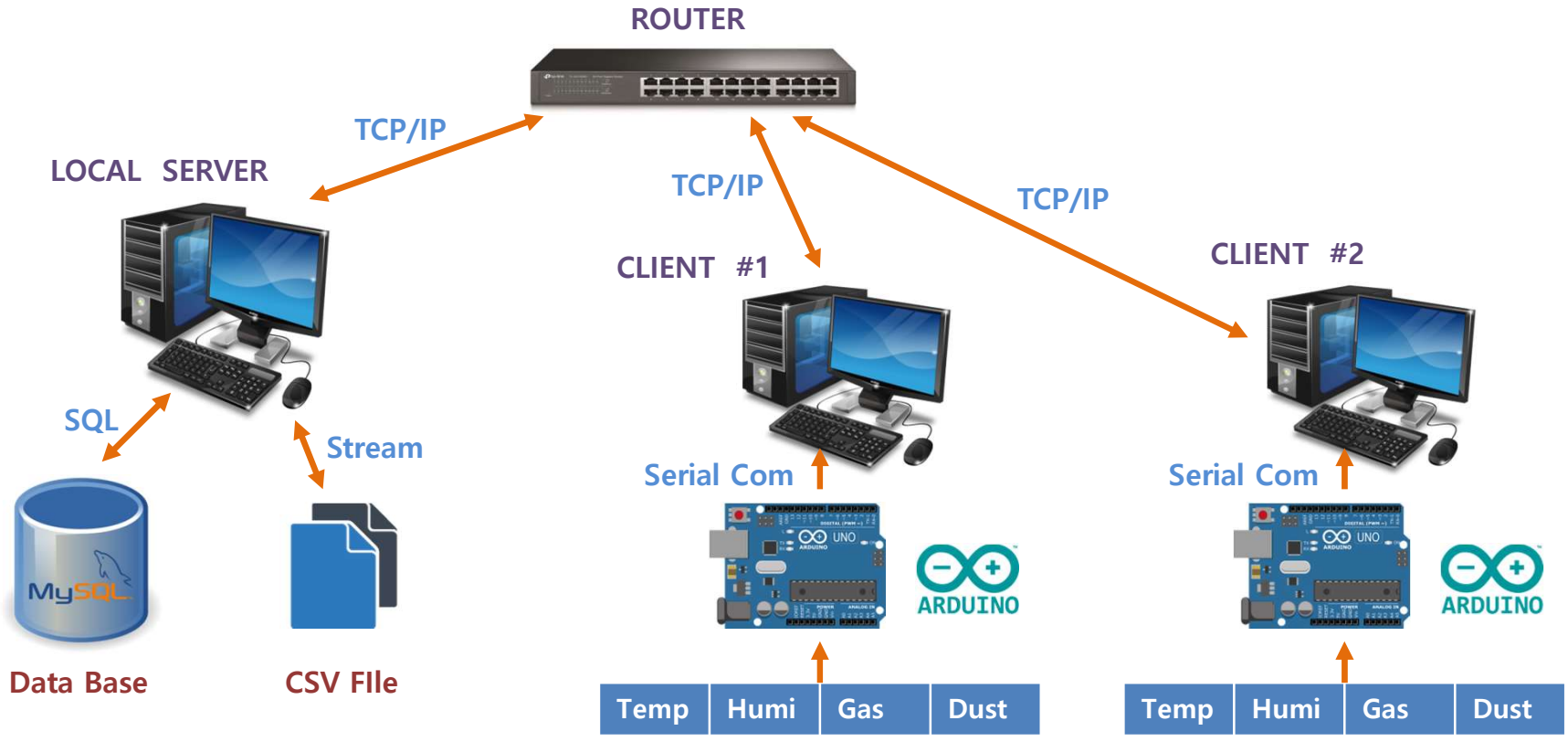
담당 교수 : 윤 종 이

010-9577-1696

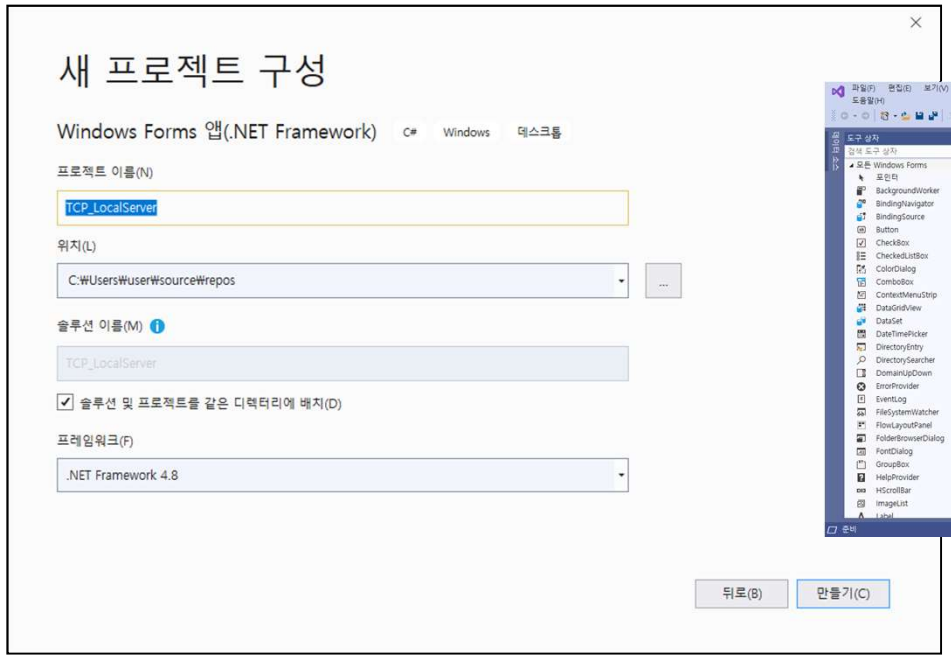
ojo1696@naver.com

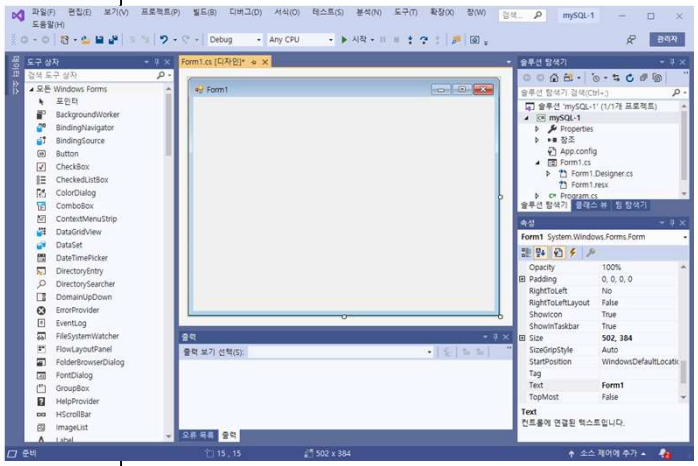
https://cafe.naver.com/yoons2022

System Blockdiagram

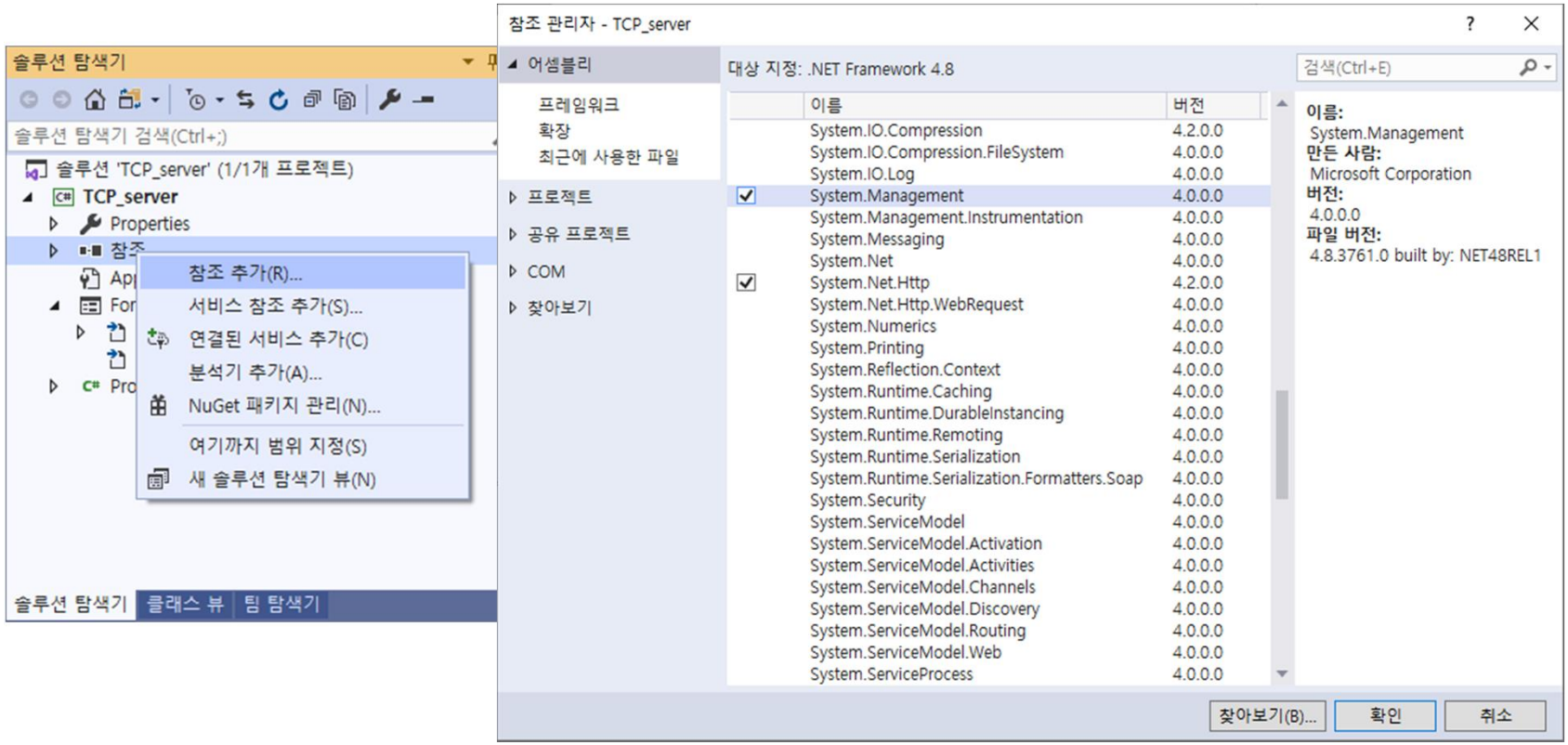


New Project

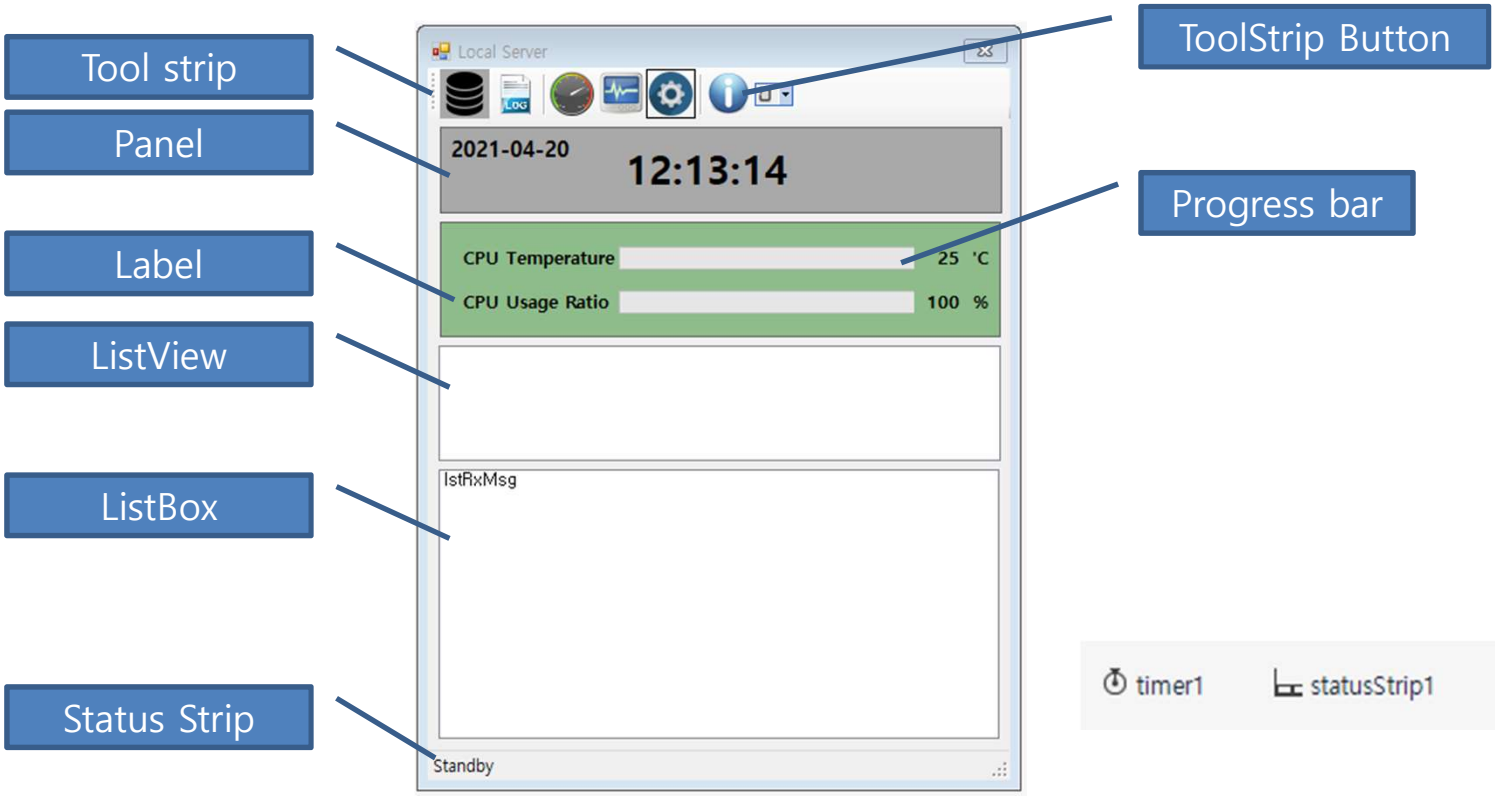

새 프로젝트 구성
Windows Forms 앱(.NET Framework)
C#
Windows
데스크톱
프로젝트 이름(N)
TCP_LocalServer
위치(L)
C:₩Users₩user₩source₩repos
...
솔루션 이름(M)
TCP_LocalServer
솔루션 및 프로젝트를 같은 디렉터리에 배치(D)
프레임워크(F)
.NET Framework 4.8
뒤로(B)
만들기(C)



System.Management



Main Form Design



Property

Object	Property
Form1	frmMain
Panel1	pnlClock
label1	lblDate
label2	lblTime
ListBox	lstRxMsg

ToolStrip Button

Object	Property
btnServer	
btnLog	
btnPerfomance	
btnResouce	
btnSysInfo	
btnAbout	

timer1

Property	Value
Enable	True
interval	250

CPU Status

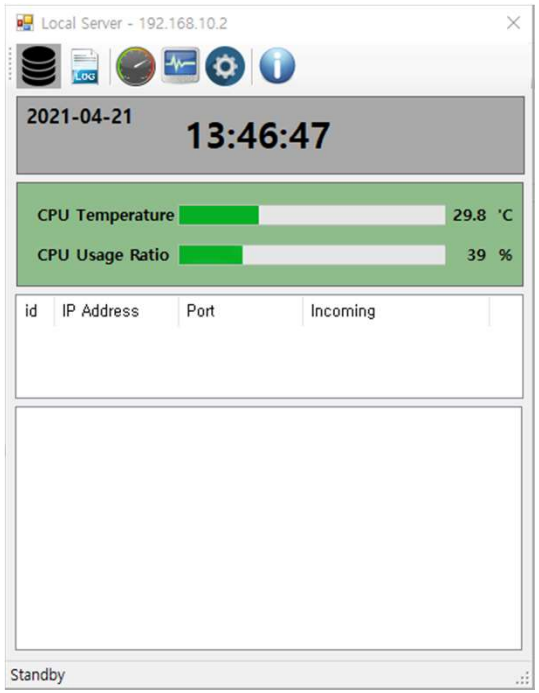
Object	Property
barCPU_Temp	
lblCPU_Temp	
barCPU_Usage	
lblCPU_Usage	

Define

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Collections;
using System.Management;
using System.Diagnostics;
using System.Net;
using TCP_Lib;
```

System Monitoring

- Server system의 동작 상태를 파악
- 위험성 회피 (CPU 온도, CPU 사용율 등)



전역 변수

```
private int LocalPort = 13000;
private static string host = Dns.GetHostName( );
private string strIPaddress = Dns.GetHostEntry(host).AddressList[1].ToString( );

private TcpServer Server;
public static frmMain FormDialog;

PerformanceCounter CpuCounter = new PerformanceCounter("Processor", "% Processor Time", "_Total");
PerformanceCounter RamCounter = new PerformanceCounter("Memory", "Available MBytes");

private bool ServerOn = false;
```

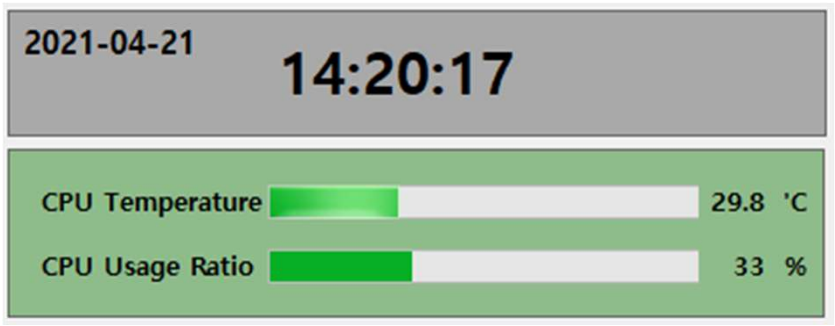
Form Event

```
public frmMain( ) {
        InitializeComponent( );
        FormDialog = this;
}

private void Form1_Load(object sender, EventArgs e) {
        this.Text = "Local Server - " + strIPaddress;
        Init_ListView( );
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e) {
        if (ServerOn) e.Cancel = true;
        else e.Cancel = false;
}
private void Timer1_Tick(object sender, EventArgs e) {
        Reflash_ListView( );
        Get_CPU_Status();
}
```

Get_CPU_Status( )

```
private void Get_CPU_Status( ) {
      if (ServerOn) pnlClock.BackColor = Color.LemonChiffon;
      else pnlClock.BackColor = Color.DarkGray;

      lblDate.Text = DateTime.Now.ToString("yyyy-MM-dd ddd");
      lblTime.Text = DateTime.Now.ToString("HH:mm:ss");

      Int16 CPU_Usage = 0;      Int32 MEM_Usage = 0;      double CPU_Temp = 0;
      ManagementObjectSearcher Sercher = new ManagementObjectSearcher(@"root₩WMI", "SELECT * FROM
MSAcpi_ThermalZoneTemperature");

      foreach ( ManagementObject obj in Sercher.Get( ) )   {
            CPU_Temp = Convert.ToDouble( obj["CurrentTemperature"].ToString( ) );
            CPU_Temp = (CPU_Temp / 10.0) - 273.15;
      }
            lblCPU_Temp.Text = CPU_Temp.ToString( );
            barCPU_Temp.Value = Convert.ToInt32(CPU_Temp);

            CPU_Usage = Convert.ToInt16(CpuCounter.NextValue( ));
            lblCPU_Usage.Text = CPU_Usage.ToString( );
            barCPU_Usage.Value = CPU_Usage;
}
```

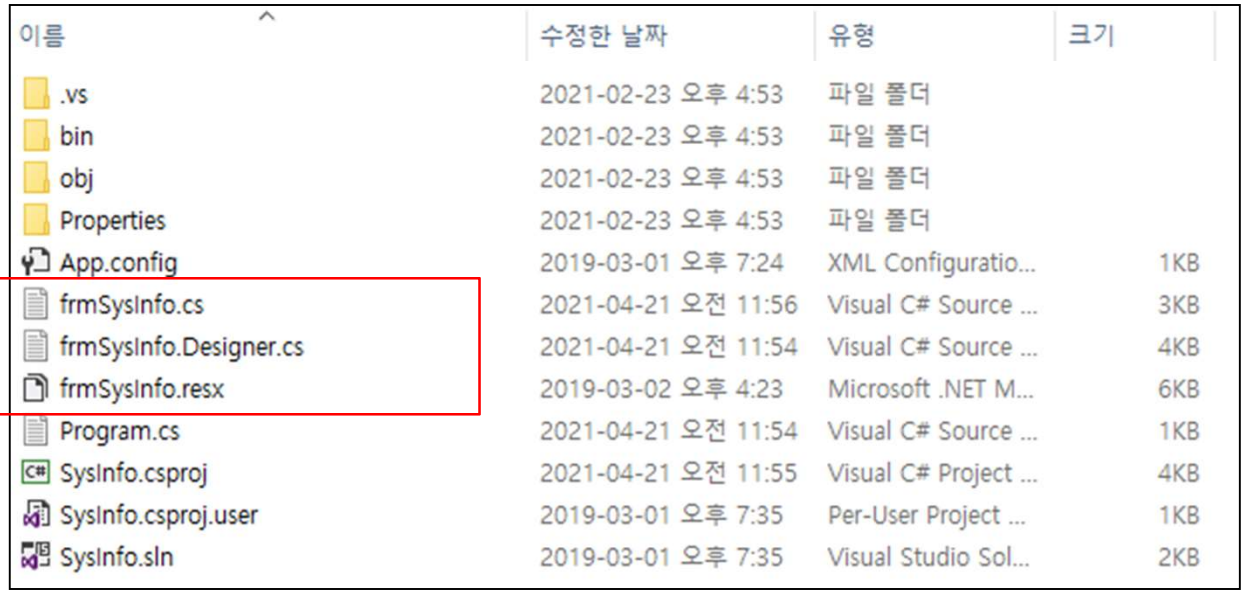
System Monitoring Run



추가 : 기존 항목 [Form]

- SysInfo project

복사



이름	수정한 날짜	유형	크기
.vs	2021-02-23 오후 4:53	파일 폴더	
bin	2021-02-23 오후 4:53	파일 폴더	
obj	2021-02-23 오후 4:53	파일 폴더	
Properties	2021-02-23 오후 4:53	파일 폴더	
App.config	2019-03-01 오후 7:24	XML Configuratio...	1KB
frmSysInfo.cs	2021-04-21 오전 11:56	Visual C# Source ...	3KB
frmSysInfo.Designer.cs	2021-04-21 오전 11:54	Visual C# Source ...	4KB
frmSysInfo.resx	2019-03-02 오후 4:23	Microsoft .NET M...	6KB
Program.cs	2021-04-21 오전 11:54	Visual C# Source ...	1KB
SysInfo.csproj	2021-04-21 오전 11:55	Visual C# Project ...	4KB
SysInfo.csproj.user	2019-03-01 오후 7:35	Per-User Project ...	1KB
SysInfo.sln	2019-03-01 오후 7:35	Visual Studio Sol...	2KB

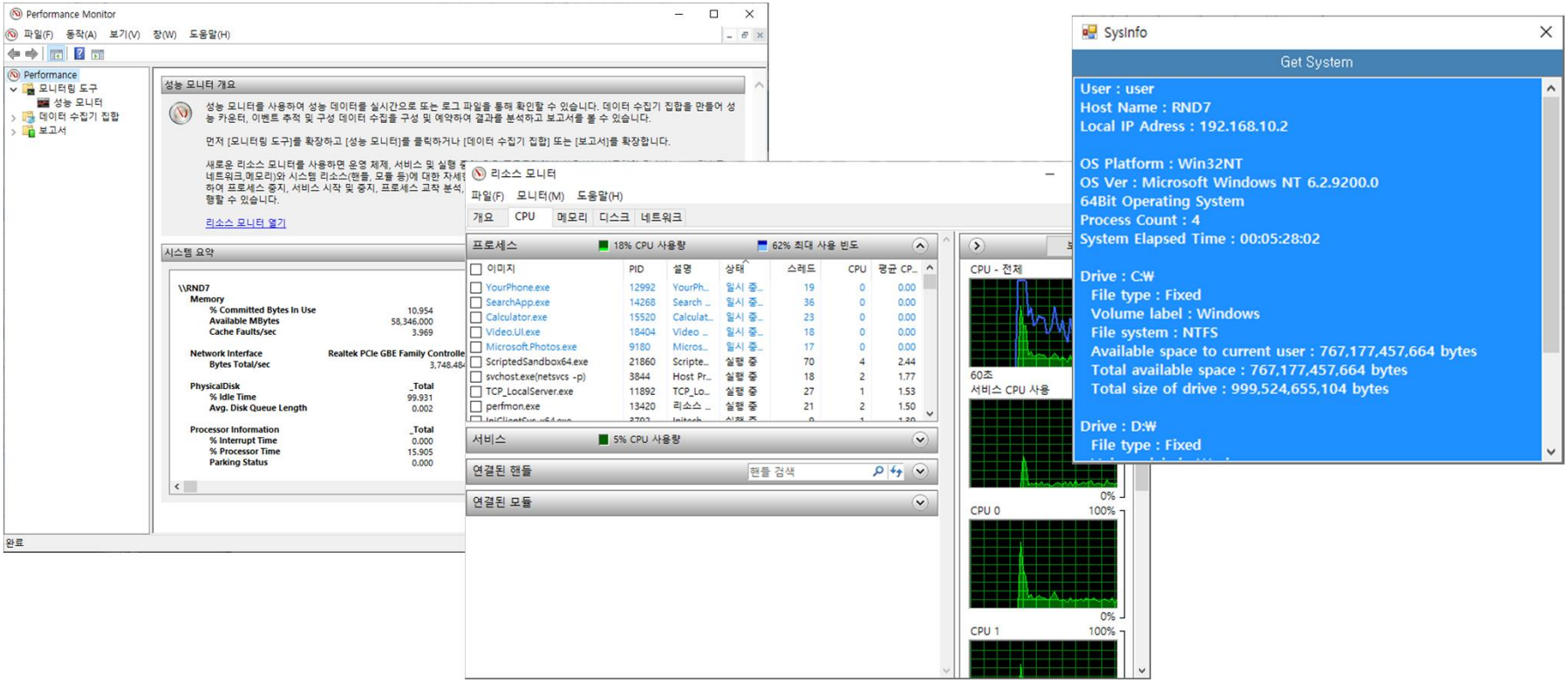
속성 바꾸기

- 현재 폴더에 붙여넣기
- [솔루션 탐색기] [프로젝트 : TCP_LocalServer] – [추가] [기존항목] { Form1.cs }
- 속성
  - Name Space : frmMain의 NameSpace와 동일하게
  - File Name : frmSysInfo

Untility Button

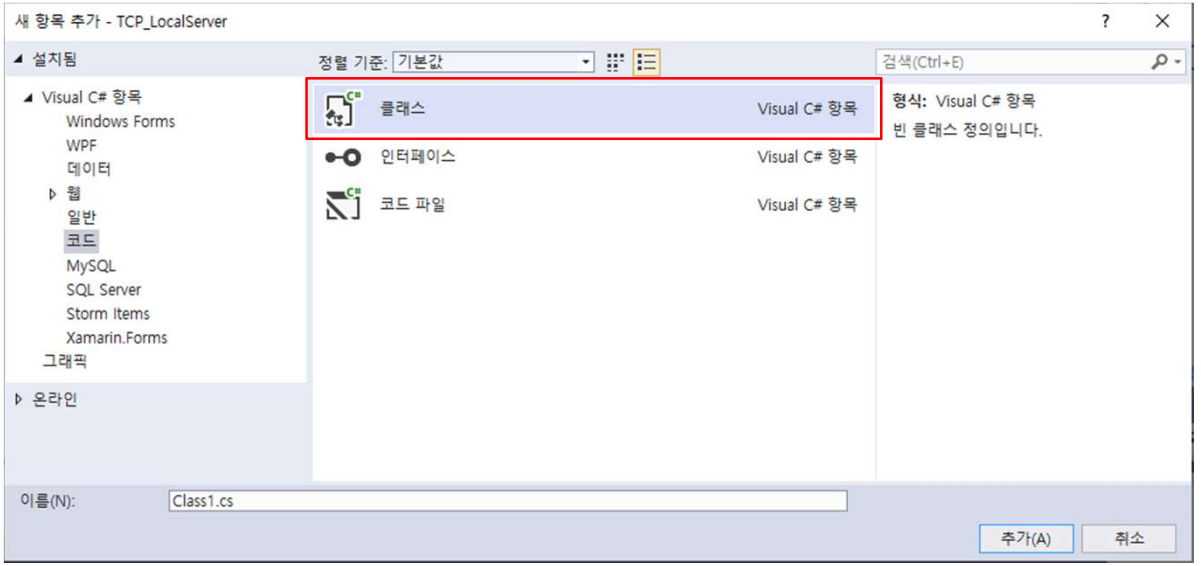
```
private void btnPerfomance_Click(object sender, EventArgs e)  {
        Process.Start("Perfmon.exe");
}

private void btnResouce_Click(object sender, EventArgs e)    {
        Process.Start("resmon.exe");
}

private void btnSysInfo_Click(object sender, EventArgs e)    {
        Form SysInfo = new frmSysInfo( );
        SysInfo.Show( );
}
```

Utility Run


Performance Monitor
파일(F) 동작(A) 보기(V) 창(W) 도움말(H)
Performance
모니터링 도구
성능 모니터
데이터 수집기 집합
보고서
성능 모니터 개요
리소스 모니터 열기
시스템 요약
\\RND7
Memory
% Committed Bytes In Use 10.954
Available MBytes 58,346.000
Cache Faults/sec 3.969
Network Interface Realtek PCIe GBE Family Controlle
Bytes Total/sec 3,748.484
PhysicalDisk _Total
% Idle Time 99.931
Avg. Disk Queue Length 0.002
Processor Information _Total
% Interrupt Time 0.000
% Processor Time 15.905
Parking Status 0.000
완료
리소스 모니터
파일(F) 모니터(M) 도움말(H)
개요 CPU 메모리 디스크 네트워크
프로세스 18% CPU 사용량 62% 최대 사용 빈도
이미지 PID 설명 상태 스레드 CPU 평균 CP...
YourPhone.exe 12992 YourPh... 일시 중... 19 0 0.00
SearchApp.exe 14268 Search ... 일시 중... 36 0 0.00
Calculator.exe 15520 Calculat... 일시 중... 23 0 0.00
Video.UI.exe 18404 Video ... 일시 중... 18 0 0.00
Microsoft.Photos.exe 9180 Micros... 일시 중... 17 0 0.00
ScriptedSandbox64.exe 21860 Scripte... 실행 중 70 4 2.44
svchost.exe(netsvcs -p) 3844 Host Pr... 실행 중 18 2 1.77
TCP_LocalServer.exe 11892 TCP_Lo... 실행 중 27 1 1.53
perfmon.exe 13420 리소스 ... 실행 중 21 2 1.50
서비스 5% CPU 사용량
연결된 핸들
핸들 검색
연결된 모듈
CPU - 전체
60초
서비스 CPU 사용
0%
CPU 0 100%
0%
CPU 1 100%
SysInfo
Get System
User : user
Host Name : RND7
Local IP Adress : 192.168.10.2
OS Platform : Win32NT
OS Ver : Microsoft Windows NT 6.2.9200.0
64Bit Operating System
Process Count : 4
System Elapsed Time : 00:05:28:02
Drive : C:₩
File type : Fixed
Volume label : Windows
File system : NTFS
Available space to current user : 767,177,457,664 bytes
Total available space : 767,177,457,664 bytes
Total size of drive : 999,524,655,104 bytes
Drive : D:₩
File type : Fixed

TCP_Lib.cs

- [솔루션 탐색기] [프로젝트 : TCP_LocalServer] – [추가] [새 항목] { TCP_Lib.cs }



Define

```
using System;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Collections;
using TCP_LocalServer;

namespace TCP_Lib {
    public class ConnectionState {

    }

    public class TcpServer {

    }
}
```

class ConnectionState

```
internal Socket _conn;
internal TcpServer _server;
internal DateTime _incoming;

internal byte[ ] _buffer;

public EndPoint RemoteEndPoint   {
        get { return _conn.RemoteEndPoint; }
}

public int AvailableData  {
        get { return _conn.Available; }
}

public bool Connected  {
        get { return _conn.Connected; }
}
```

```
public int Read(byte[ ] buffer, int offset, int count)  {
      try   {
          if (_conn.Available > 0)  return  _conn.Receive(buffer, offset, count, SocketFlags.None);
               else return  0;
      } catch  {     return 0;      }
}
public bool Write(byte[ ] buffer, int offset, int count)  {
      try  {
           _conn.Send(buffer, offset, count, SocketFlags.None);
           return  true;
      } catch {     return  false;   }
}
public void EndConnection()  {
      if (_conn != null && _conn.Connected)  {
              _conn.Shutdown(SocketShutdown.Both);
              _conn.Close();
       }
       _server.DropConnection(this);
}
```

class TcpServer

```
private string _IPaddress;
private int _port;
private Socket _listener;

private ArrayList _connections;
private ArrayList _ClientStatus;
private int _maxConnections = 3;

private string _receivedStr;

private AsyncCallback ConnectionReady;
private WaitCallback AcceptConnection;
private AsyncCallback ReceivedDataReady;

private delegate void SetTextDelegate(string
getString);
```

```
public TcpServer( int port)  {
    _port = port;
    _listener = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);

    _connections = new ArrayList( );
    _ClientStatus = new ArrayList( );

    ConnectionReady = new
AsyncCallback(ConnectionReady_Handler);

    AcceptConnection = new
WaitCallback(AcceptConnection_Handler);

     ReceivedDataReady = new
AsyncCallback(ReceivedDataReady_Handler);
}
```

Start / Stop

```
public bool Start( ) {
    try  {
        _listener.Bind(new IPEndPoint(
IPAddress.Parse(_IPaddress), _port));

        _listener.Listen(100);
        _listener.BeginAccept(ConnectionReady, null);
        return true;
     }  catch  {
        return false;
     }
}
```

```
public void Stop( )  {
    try  {
        lock (this)  {
            _listener.Close( );
            _listener = null;

            foreach (object obj in _connections)  {
                ConnectionState state = obj as ConnectionState;
                try  {
                    DropConnection(state);
                }  catch  {
                    //some error
                }
                state._conn.Shutdown(SocketShutdown.Both);
                state._conn.Close( );
             }
            _connections.Clear( );
            Update_Connection( );
         }
    }  catch (Exception ex)  {
        MessageBox.Show(ex.Message.ToString( ), "LocalServer",
MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

ConnectionReady_Handler

```
private void ConnectionReady_Handler(IAsyncResult asyncResult) {
    lock (this)  {
        if (_listener == null) return;
        Socket conn = _listener.EndAccept(asyncResult);
        if (_connections.Count >= _maxConnections)  {
            string msg = "$busy";
            conn.Send(Encoding.UTF8.GetBytes(msg), 0, msg.Length, SocketFlags.None);
            conn.Shutdown(SocketShutdown.Both);
            conn.Close();
        }  else  {
            ConnectionState state = new ConnectionState();
            state._conn = conn;
            state._server = this;
            state._incoming = DateTime.Now;
            state._buffer = new byte[4];
            _connections.Add(state);

            Update_Connection();

            ThreadPool.QueueUserWorkItem(AcceptConnection, state);
        }
        _listener.BeginAccept(ConnectionReady, null);
    }
}
```

Update_Connection

```
private void Update_Connection( )  {
      _ClientStatus.Clear( );

      foreach (object obj in _connections)  {
          ConnectionState state = obj as ConnectionState;
          Socket conn=state._conn;
          IPEndPoint ip_point = (IPEndPoint)conn.RemoteEndPoint;
          string status = ip_point.Address.ToString( ) + "," + ip_point.Port.ToString( ) + ","+ state._incoming.ToString("yyyy-
MM-dd HH:mm:ss");
          _ClientStatus.Add(status);
     }
}
```

DropConnection

```
internal void DropConnection(ConnectionState state)   {
     lock (this)  {
          state._conn.Shutdown(SocketShutdown.Both);
          state._conn.Close( );

          if (_connections.Contains(state))  {
               _connections.Remove(state);
               Update_Connection( );
          }
     }
}
```

AcceptConnection_Handler

```
private void AcceptConnection_Handler(object ConnState)  {
    ConnectionState state = ConnState as ConnectionState;
    try  {
        _receivedStr = "";
        if (!state.Write(Encoding.UTF8.GetBytes("$Accept₩n"), 0, 8)) state.EndConnection();
    }  catch   {
        //
    }

    if (state._conn.Connected) state._conn.BeginReceive(state._buffer, 0, 0, SocketFlags.None, ReceivedDataReady, state);
}
```

ReceivedDataReady_Handler

```
private void ReceivedDataReady_Handler(IAsyncResult asyncResult)   {
    try  {
        ConnectionState state = asyncResult.AsyncState as ConnectionState;
        state._conn.EndReceive(asyncResult);

        if (state._conn.Available == 0)  {
            DropConnection(state);
        }  else  {
            try  {
                byte[ ] buffer = new byte[1024];
                //--------------------------->
            }  catch(Exception ex)  {
                MessageBox.Show(ex.Message.ToString(), "LocalServer", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
           if (state._conn.Connected) state._conn.BeginReceive(state._buffer, 0, 0, SocketFlags.None, ReceivedDataReady, state);
        }
    }  catch(Exception ex)  {
        MessageBox.Show(ex.Message.ToString(),"LocalServer",MessageBoxButtons.OK,MessageBoxIcon.Error);
    }
}
```

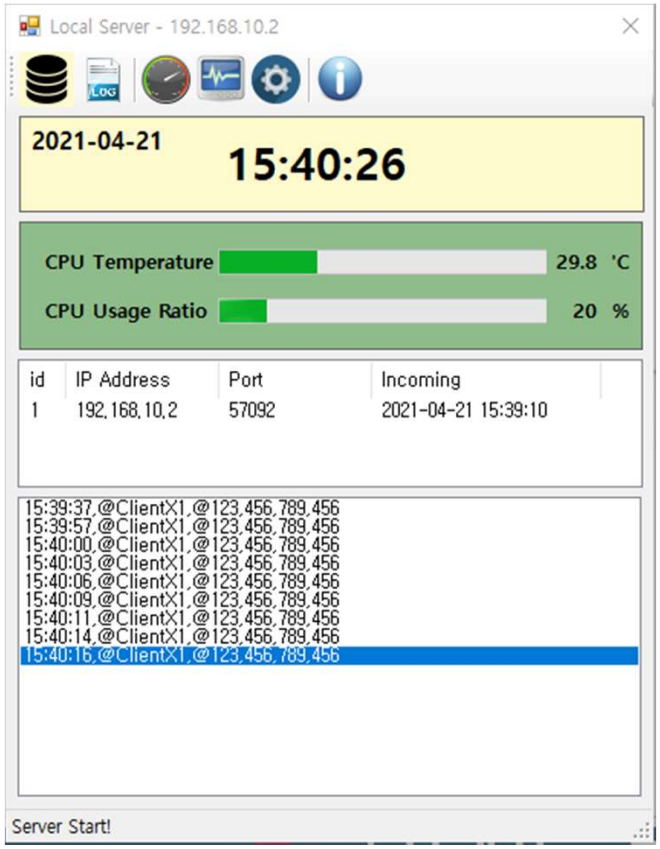
ReceivedDataReady_Handler < Insert >

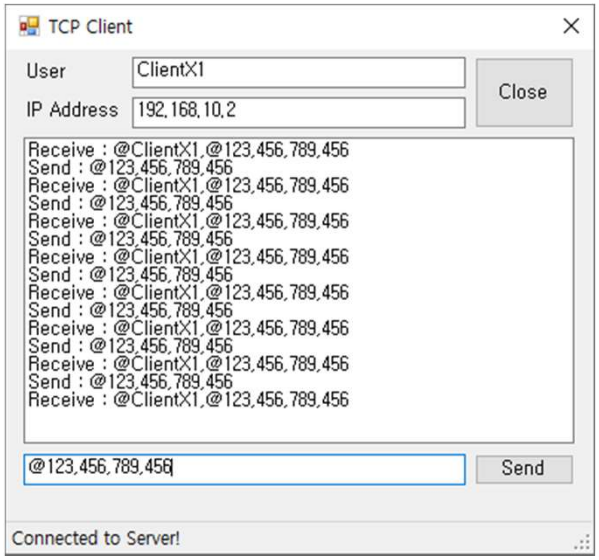
```
while (state.AvailableData > 0)  {
    int readBytes = state.Read(buffer, 0, 1024);
    if (readBytes > 0)  {
        _receivedStr += Encoding.UTF8.GetString(buffer, 0, readBytes);
        if (_receivedStr.IndexOf("₩n") >= 0)  {
            //-----------------------------------------------------------------
            string rxmsg = DateTime.Now.ToString("HH:mm:ss") + "," + _receivedStr;
            frmMain.FormDialog.BeginInvoke(new SetTextDelegate(frmMain.FormDialog.TCPmsgReceive), new object[]
{ rxmsg });

            state.Write(Encoding.UTF8.GetBytes(_receivedStr), 0, _receivedStr.Length);
            _receivedStr = "";
        }
    }  else  {
        state.EndConnection( );
    }
}
```

전달 변수

```
public int MaxConnections {
        get { return _maxConnections;  }
        set { _maxConnections = value;  }
}

public ArrayList ClientStatus  {
        get { return _ClientStatus; }
}

public string IPaddress  {
        get { return _IPaddress; }
        set { _IPaddress = value; }
}
```

Server Run


Local Server - 192.168.10.2
2021-04-21
15:40:26
CPU Temperature
29.8 'C
CPU Usage Ratio
20 %
id
IP Address
Port
Incoming
1
192.168.10.2
57092
2021-04-21 15:39:10
15:39:37,@ClientX1,@123,456,789,456
15:39:57,@ClientX1,@123,456,789,456
15:40:00,@ClientX1,@123,456,789,456
15:40:03,@ClientX1,@123,456,789,456
15:40:06,@ClientX1,@123,456,789,456
15:40:09,@ClientX1,@123,456,789,456
15:40:11,@ClientX1,@123,456,789,456
15:40:14,@ClientX1,@123,456,789,456
15:40:16,@ClientX1,@123,456,789,456
Server Start!


TCP Client
User
ClientX1
IP Address
192.168.10.2
Close
Receive : @ClientX1,@123,456,789,456
Send : @123,456,789,456
Receive : @ClientX1,@123,456,789,456
Send : @123,456,789,456
Receive : @ClientX1,@123,456,789,456
Send : @123,456,789,456
Receive : @ClientX1,@123,456,789,456
Send : @123,456,789,456
Receive : @ClientX1,@123,456,789,456
Send : @123,456,789,456
Receive : @ClientX1,@123,456,789,456
Send : @123,456,789,456
Receive : @ClientX1,@123,456,789,456
Send : @123,456,789,456
Receive : @ClientX1,@123,456,789,456
@123,456,789,456
Send
Connected to Server!