服务端开发说明

1. **通信协议**

与串口设备（客户端）采用**TCP/IP** 的通信方式

1. **通信流程**

服务端软件启动时监听本地的特定端口（建议写到配置文件中），启动监听后等待客户端建立连接（如果需要多台设备连接到服务端请注意对多个客户端建立通信的支持），和客户端建立连接后开始进行通信，设备默认会在一定的时间内（默认是2秒）发送内容为ok?的消息,此时需要服务端给客户端返回ok，其他的消息根据之前约定的协议进行处理。

设备与服务端建立的是长连接，以减少频繁建立连接的性能损耗，所以服务端需要设置一个超时时间，当一段时间以后客户端没有和服务端通信时需要断开与客户端的连接，因为设备可能已经异常中断，但是服务端仍然在维持着这个连接的话会浪费性能。

**注意**，在实际过程中多条消息可能会黏在一起发送过来，需要根据回车换行字符分隔出来进行处理，同时在给客户端回消息时需要在消息的结尾加上$符号，以便客户端处理服务端粘在一起的消息。比如发送的ok信号实际发送的消息时ok$

1. **代码示例（C#）**
   1. **监听端口**

IPAddress ipaddress = IPAddress.Parse(ip);

serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

try

{

serverSocket.Bind(new IPEndPoint(ipaddress, port)); //绑定IP地址：端口

serverSocket.Listen(50); //设定最多50个排队连接请求

portStatusAction(ServerPortStatus.Listen);

tipInfoAction("啟動監聽" + serverSocket.LocalEndPoint.ToString() + "成功");

}

catch (Exception ex)

{

tipErrorAction("無法啟動監聽:" + ex.Message);

logger.Error("無法啟動監聽", ex);

return;

}

* 1. **建立连接**

while (!stop)

{

Socket clientSocket = null;

try

{

//tipInfoAction("等待客戶端連接...");

clientSocket = serverSocket.Accept();

clientSocket.ReceiveTimeout = timeout;

if (clientConnected != null)

{

clientConnected(clientSocket);

}

tipInfoAction("客戶端連接成功，IP：" + ((IPEndPoint)clientSocket.RemoteEndPoint).Address.ToString() + ",端口：" + ((IPEndPoint)clientSocket.RemoteEndPoint).Port);

Task recemsgTask = Task.Factory.StartNew(() =>

{

RecMsg(clientSocket);//消息处理

});

}

catch

{

portStatusAction(ServerPortStatus.NotListen);

tipErrorAction("監聽端口停止");

return;

}

}

* 1. **接收消息并处理**

void RecMsg(Socket clientSocket)

{

bool connected = true;

while (!stop && connected)

{

string result = string.Empty;

List<byte> data = new List<byte>();

byte[] buffer = new byte[1024];

int length = 0;

try

{

while ((length = clientSocket.Receive(buffer)) > 0)

{

for (int j = 0; j < length; j++)

{

data.Add(buffer[j]);

}

if (length < buffer.Length)

{

break;

}

}

}

catch (Exception ex)

{

connected = false;

//logger.Error("接收異常", ex);

tipErrorAction("接收異常：" + ex.Message);

}

if (data.Count > 0)

{

result = Encoding.Default.GetString(data.ToArray(), 0, data.Count);

MessageInfo msg = new MessageInfo()

{

RemoteSocket = clientSocket,

Message = result

};

messageAction(msg);

}

}

tipInfoAction("已斷開客戶端連接，IP：" + ((IPEndPoint)clientSocket.RemoteEndPoint).Address.ToString() + ",端口：" + ((IPEndPoint)clientSocket.RemoteEndPoint).Port);

}

* 1. **发送消息到客户端**

public void SendMsg(Socket clientSocket, string msg)

{

try

{

if (clientSocket != null)

{

clientSocket.Send(Encoding.Default.GetBytes(msg));

}

}

catch (Exception ex)

{

logger.Error(ex);

}

}

**服务端收发消息综合（C#）**

****

**用法:**

try

{

listenner = new ReceiveTcpMsg(listenport, (msg) =>

{

System.Threading.Tasks.Task.Factory.StartNew(() =>

{

this.PorcessMessage(msg);

});

},

(status) =>

{

this.BeginInvoke(new Action(() =>

{

if (status == ServerPortStatus.Listen)

{

btnStart.Enabled = false;

btnStop.Enabled = true;

}

else

{

btnStart.Enabled = true;

btnStop.Enabled = false;

}

}));

},

(info) =>

{

ShowInfo(info);

},

(error) =>

{

ShowInfo(error);

}, null, listenip);

//通信端口监听

listenner.StartListen();

}

catch (Exception ex)

{

MessageBox.Show("啟動異常!" + ex.Message);

logger.Error("啟動異常", ex);

}

private void PorcessMessage(MessageInfo msg)

{

//处理接收到的消息

//1:com口条码消息

//2：sensor消息

//3：通信握手消息

**}**