首先需要两台电脑，可以两个同学共同来完成；当然，如果自己有两台电脑的话就更好了！

在一台电脑上创建一个TCP Socket服务器应用程序

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <WinSock2.h>

#include <stdlib.h>

#pragma comment (lib,"WS2\_32.lib");

#define BUF\_SIZE 64

int main()

{

WSADATA wsd;         //WSADATA变量，用于初始化Windows Socket

SOCKET sServer;     //服务器socket，用于监听客户端请求

SOCKET sClient;     //客户端socket，用于实现与客户端的通信

int retVal;         //调用各种socket函数的返回值

char buf[BUF\_SIZE];

//初始化Socket环境

if (WSAStartup(MAKEWORD(2, 2), &wsd) != 0)

{

//printf("WASStartup failed !\n");

std::cout << "WASStartup failed !" << std::endl;

return 1;

}

//创建用于监听的Socket

sServer = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (sServer == INVALID\_SOCKET)

{

//printf("socket failed !\n");

std::cout << "socket failed !" << std::endl;

return -1;

}

//设置服务器Socket地址

SOCKADDR\_IN addrServ;

addrServ.sin\_family = AF\_INET;

addrServ.sin\_port = htons(9990);

addrServ.sin\_addr.S\_un.S\_addr = htons(INADDR\_ANY);

//绑定Sockets Server 到本地地址

retVal = bind(sServer, (const struct sockaddr \*)&addrServ, sizeof(SOCKADDR\_IN));

if (SOCKET\_ERROR == retVal)

{

std::cout << "bind failed !" << std::endl;

closesocket(sServer);

WSACleanup();

return -1;

}

//在Socket Server上进行监听

retVal = listen(sServer,1);

if (SOCKET\_ERROR==retVal)

{

std::cout << "listen failed !" << std::endl;

closesocket(sServer);

WSACleanup();

return -1;

}

//接受来自客户端的请求

std::cout << "TCP Server Start..." << std::endl;

sockaddr\_in addrClient;    //客户端地址

int addrClientLen = sizeof(addrClient);

sClient = accept(sServer, (sockaddr FAR \*)&addrClient, &addrClientLen);

if (INVALID\_SOCKET == sClient)

{

std::cout << "accept failed !" << std::endl;

closesocket(sServer);

WSACleanup();

return -1;

}

//在服务器与客户端之间发送和接收数据

//循环接受客户端的数据，直接客户端发送quit命令后退出

while (true)

{

memset(buf, 0, BUF\_SIZE);

retVal = recv(sClient, buf, BUF\_SIZE, 0);

if (SOCKET\_ERROR == retVal)

{

std::cout << "recv failed !" << std::endl;

closesocket(sServer);

WSACleanup();

return -1;

}

//获取当前系统时间

SYSTEMTIME st;

GetLocalTime(&st);

char sDataTime[30];

sprintf(sDataTime, "%4d-%2d-%2d %2d:%2d:%2d", st.wYear, st.wMonth, st.wDay, st.wHour, st.wMinute, st.wSecond);

//打印输出信息

std::cout << sDataTime << "Recv from Client [" << inet\_ntoa(addrClient.sin\_addr) << ":";

std::cout << addrClient.sin\_port << "] : " << buf << std::endl;

//如果客户端发送“quit”字符串，则服务器退出

if (strcmp(buf, "quit") == 0)

{

retVal = send(sClient, "quit", strlen("quit"), 0);

break;

}

//否则向客户端发送回显字符串

else

{

char msg[BUF\_SIZE];

sprintf(msg, "Message received - %s", buf);

retVal = send(sClient, msg, strlen(msg), 0);

if (SOCKET\_ERROR == retVal)

{

std::cout << "send failed !" << std::endl;

closesocket(sServer);

closesocket(sClient);

WSACleanup();

return -1;

}

}

}

//释放资源

closesocket(sServer);

closesocket(sClient);

WSACleanup();

system("pause");

return 0;

}

接着在另一台电脑上创建一个TCP Socket客户端应用程序

#include <iostream>

#include <stdio.h>

#include <WinSock2.h>

#include <string>

#include <stdlib.h>

#pragma comment (lib,"ws2\_32.lib")

#define BUF\_SIZE  64

int main()

{

WSADATA wsd;   //用于初始化Windows socket

SOCKET sHost;   //与服务器进行通信的socket

SOCKADDR\_IN servAddr;  //服务器地址

char buf[BUF\_SIZE];   //用于接收数据缓冲区

int retVal;    //调用各种Socket函数的返回值

//初始化socket动态库

if(WSAStartup(MAKEWORD(2,2),&wsd)!=0)

{

std::cout<<"WSAStartup failed !"<<std::endl;

return -1;

}

//创建用于通信的socket

sHost=socket(AF\_INET,SOCK\_STREAM,IPPROTO\_TCP);

if(INVALID\_SOCKET==sHost)

{

std::cout<<"socket failed !"<<std::endl;

WSACleanup();;

return -1;

}

//设置服务器Socket地址

servAddr.sin\_family=AF\_INET;

//用户需要根据实际情况修改

servAddr.sin\_addr.S\_un.S\_addr=inet\_addr("192.168.88.103");

servAddr.sin\_port=htons(9990);

int sServerAddrlen=sizeof(servAddr);

//连接到服务器

retVal=connect(sHost,(LPSOCKADDR)&servAddr,sizeof(servAddr));

if(SOCKET\_ERROR==retVal)

{

std::cout<<"connect failed !"<<std::endl;

closesocket(sHost);

WSACleanup();

return -1;

}

//在客户端与服务器之间发送和接收数据

while(true)

{

std::cout<<"Please input a string to send: ";

std::string str;

 //接收输入的数据

std::getline(std::cin,str);

 //将用户输入的数据复制到buf钟

ZeroMemory(buf,BUF\_SIZE);

strcpy(buf,str.c\_str());

//向服务器发送数据

retVal=send(sHost,buf,strlen(buf),0);

if(SOCKET\_ERROR==retVal)

{

std::cout<<"send failed !"<<std::endl;

closesocket(sHost);

WSACleanup();

return -1;

}

//接收服务器回传的数据

retVal=recv(sHost,buf,sizeof(buf)+1,0);

std::cout<<"Recv from Server: "<<buf<<std::endl;

if(strcmp(buf,"quit")==0)

{

std::cout<<"quit!"<<std::endl;

break;

}

}

//释放资源

closesocket(sHost);

WSACleanup();

system("pause");

return 0;

}

---------------------

单片机与电脑

#include "stm32f10x.h"

void my\_delay\_ms(int time);

void my\_delay\_ms(int time)

{

int i=0;

while(time--)

{

i=12000;

while(i)

{

i--;

}

}

}

void uart\_init(u32 bound){

//GPIO端口设置

GPIO\_InitTypeDef GPIO\_InitStructure;

USART\_InitTypeDef USART\_InitStructure;

NVIC\_InitTypeDef NVIC\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1|RCC\_APB2Periph\_GPIOA, ENABLE); //使能USART1，GPIOA时钟

//USART1\_TX GPIOA.9

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9; //PA.9

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //复用推挽输出

GPIO\_Init(GPIOA, &GPIO\_InitStructure);//初始化GPIOA.9

//USART1\_RX GPIOA.10初始化

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_10;//PA10

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;//浮空输入

GPIO\_Init(GPIOA, &GPIO\_InitStructure);//初始化GPIOA.10

//Usart1 NVIC 配置

NVIC\_InitStructure.NVIC\_IRQChannel = USART1\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority=3 ;//抢占优先级3

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 3; //子优先级3

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //IRQ通道使能

NVIC\_Init(&NVIC\_InitStructure); //根据指定的参数初始化VIC寄存器

//USART 初始化设置

USART\_InitStructure.USART\_BaudRate = bound;//串口波特率

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;//字长为8位数据格式

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;//一个停止位

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;//无奇偶校验位

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;//无硬件数据流控制

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx; //收发模式

USART\_Init(USART1, &USART\_InitStructure); //初始化串口1

USART\_ITConfig(USART1, USART\_IT\_RXNE, ENABLE);//开启串口接受中断

USART\_Cmd(USART1, ENABLE); //使能串口1

}

uint16\_t str=1;

int main()

{

u16 times=0;

uart\_init(115200);

//key\_init();

while(1)

{

times++;

my\_delay\_ms(10);

if(times%10000)

{

USART\_SendData(USART1, str);//向串口1发送数据

while(USART\_GetFlagStatus(USART1,USART\_FLAG\_TC)!=SET);

}

}

return 0;

}