TCP/IP 网络组件 Lwip 之 UDP Server

RealTouch 评估板 RT-Thread 入门文档

版本号：1.0.0

日期：2012/9/1

修订记录

日期	作者	修订历史
2012/9/1	bloom5	创建文档

实验目的

- 快速了解 Lwip 组件，
- 熟悉 UDP Server 和 UDP Client 之间的通信机制。

硬件说明

本实验使用 RT-Thread 官方的 Realtouch 开发板作为实验平台。涉及到的硬件主要为

- RJ45 接口，作为网络连接的需要，我们需要用网线将 Realtouch 和目标机连接起来，具体请参见《Realtouch 开发板使用手册》
- 串口 3，作为 rt_kprintf 输出，需要连接 JTAG 扩展板

实验原理及程序结构

实验设计

UDP 全称为 User Datagram Protocol， 中文名是用户数据报协议，是 OSI 参考模型中一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务协议是一种简单的网络传输层协议，RFC768 中有详细描述。UDP 完全不同于提供面向连接的、可靠的字节流的 TCP 协议，通常，UDP Client 不需要和 Server 建立连接，就可以进行数据传输。

本实验中 RealTouch 作为 UDP Server，PC 作为 UDP Client，两机进行网络通信。

源程序说明

系统依赖

在 rtconfig.h 中需要开启

- #define RT_USING_HEAP

此项可选，开启此项可以创建动态线程和动态信号量，如果使用静态线程和静态信号量，则此项不是必要的

- #define RT_USING_LWIP

此项必须，本实验使用 LWIP 组件，因此需要开启此项

- #define RT_USING_CONSOLE

此项必须，在开始过程中仍需通过串口进行显示相关的工作

主程序说明

本实验中，在初始化线程中完成了网络硬件的初始化，lwip 初始化，然后开启了 udpsrv。

```
void rt_init_thread_entry(void *parameter)
{
#ifdef RT_USING_LWIP
    /* initialize eth interface */
    rt_hw_stm32_eth_init();
#endif

#ifdef RT_USING_COMPONENTS_INIT
    /* initialization RT-Thread Components */
    rt_components_init();
#endif

    rt_platform_init();
    /* do some thing here. */
    udpserv();
}

int rt_application_init()
{
    rt_thread_t init_thread;

    init_thread = rt_thread_create("init",
                                   rt_init_thread_entry, RT_NULL,
                                   2048, 8, 20);

    if (init_thread != RT_NULL)
        rt_thread_startup(init_thread);

    return 0;
}
```

udpsrv 所有的操作均在 udpserver.c 中的 udpserver()函数中完成。

```
#include <rtthread.h>
#include <lwip/sockets.h> /* 使用 BSD socket，需要包含 sockets.h 头文件 */

#define BUFSZ   1024

void udpserv(void *paramemter)
```

```
  {
      int sock;
      int bytes_read;
      char *recv_data;
      rt_uint32_t addr_len;
      struct sockaddr_in server_addr, client_addr;

      /* 分配接收用的数据缓冲 */
      recv_data = rt_malloc(BUFSZ);
      if (recv_data == RT_NULL)
      {
          /* 分配内存失败，返回 */
          rt_kprintf("No memory\n");
          return;
      }

      /* 创建一个 socket，类型是 SOCK_DGRAM，UDP 类型 */
      if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
      {
          rt_kprintf("Socket error\n");

          /* 释放接收用的数据缓冲 */
          rt_free(recv_data);
          return;
      }

      /* 初始化服务端地址 */
      server_addr.sin_family = AF_INET;
      server_addr.sin_port = htons(5000);
      server_addr.sin_addr.s_addr = INADDR_ANY;
      rt_memset(&(server_addr.sin_zero), 0,
sizeof(server_addr.sin_zero));

      /* 绑定 socket 到服务端地址 */
      if (bind(sock, (struct sockaddr *)&server_addr,
               sizeof(struct sockaddr)) == -1)
      {
          /* 绑定地址失败 */
          rt_kprintf("Bind error\n");

          /* 释放接收用的数据缓冲 */
          rt_free(recv_data);
          return;
      }
```

```
    addr_len = sizeof(struct sockaddr);
    rt_kprintf("UDPServer Waiting for client on port 5000...\n");

    while (1)
    {
        /* 从 sock 中收取最大 BUFSZ - 1 字节数据 */
        bytes_read = recvfrom(sock, recv_data, BUFSZ - 1, 0,
                              (struct sockaddr *)&client_addr,
&addr_len);
        /* UDP 不同于 TCP，它基本不会出现收取的数据失败的情况，除非设置了
超时等待 */

        recv_data[bytes_read] = '\0'; /* 把末端清零 */

        /* 输出接收的数据 */
        rt_kprintf("\n(%s , %d) said : ",
inet_ntoa(client_addr.sin_addr),
                  ntohs(client_addr.sin_port));
        rt_kprintf("%s", recv_data);

        /* 如果接收数据是 exit，退出 */
        if (strcmp(recv_data, "exit") == 0)
        {
            lwip_close(sock);

            /* 释放接收用的数据缓冲 */
            rt_free(recv_data);
            break;
        }
    }

    return;
}

#ifdef RT_USING_FINSH
#include <finsh.h>
/* 输出 udpserv 函数到 finsh shell 中 */
FINSH_FUNCTION_EXPORT(udpserv, startup udp server);
#endif
```

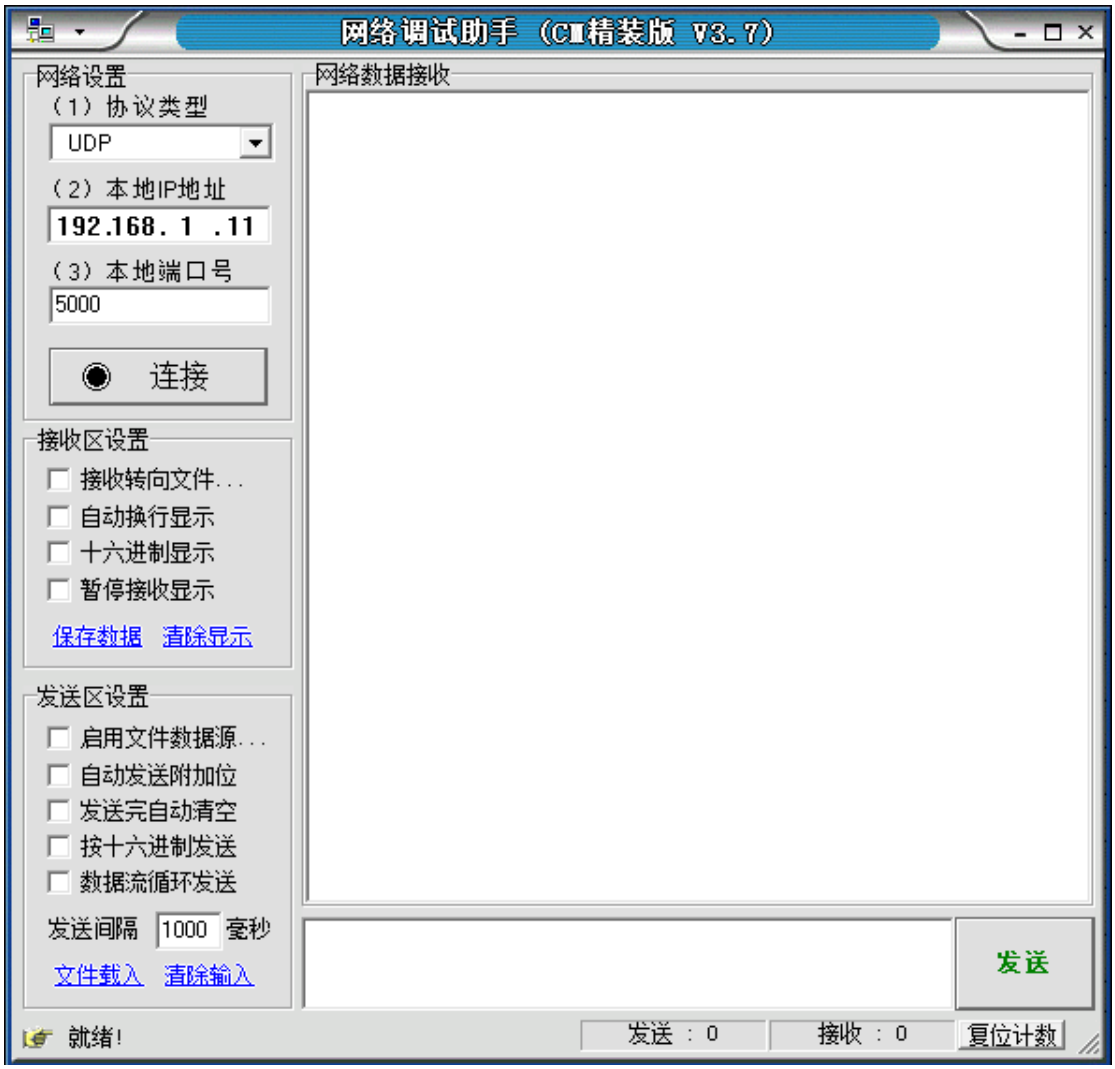
编译调试及观察输出信息

编译请参见《RT-Thread 配置开发环境指南》完成编译烧录，参考《Realtouch 开发板使用手册》完成硬件连接，连接好串口线，连上网线。

首先可以看到串口有如下的信息：

```
 \ | /
 - RT -     Thread Operating System
  / | \     1.1.0 build Aug 29 2012
  2006 - 2012 Copyright by rt-thread team
TCP/IP initialized!
finsh>>UDPServer Waiting for client on port 5000...
```

同时设置好 UDP 客户端上，端口号设置为 5000：



在发送框中输入相应的内容，并发送，Server 将收到内容输出到 Client 的客户端的接受框内。

结果分析

通过以上内容的操作，实现了 UDP Server 和 Client 间的通信。