以太网及硬件 TCP/IP 协议栈应用

以太网在各个领域和行业有着非常广泛和深入的应用，这主要源于以太网的高度灵活性和较易实现的特点。因为以太网具有组网简单，成本低廉，兼容性优秀，连接可靠，以及拓扑调整方便的优点，在作为智能家居，物联网或者无线传感网络的网关方面有其他的网络技术所不具备的优势，从而得到大力的发展和应用。本文将详细介绍如何使嵌入式系统接入到以太网，如何采用硬件协议栈的方式使您的方案或应用快速高效的连接到互联网，如何实现 TCP/IP 的通信，以及如何实现上层应用层协议等等。

第 1 章 以太网模型

以太网的实现采用层次结构的概念，每一层都有自己的功能，就像建筑物一样，每一层都靠下一层支持，每一层也都为上一层功能的实现打好基础。

实际上，用户接触到的只是最上面的一层，根本感觉不到底层的存在。要理解以太网，必须从最下层开始，自下而上理解每一层的功能。

1.1 五层结构

以太网模型有不同的分层方式，ISO（国际标准组织）提出 OSI 七层网络模型，自上而下分别为：应用层、表示层、会话层、传输层、网络层、数据链路层、物理层。OSI 七层网络模型主要是为了解决异种网络互联时所遇到的兼容性问题。它的最大优点是将服务、接口和协议这三个概念明确地区分开来，也使网络的不同功能模块承担起不同的职责。由于互联网网络体系结构以 TCP/IP 协议为核心，因而基于 TCP/IP 的参考模型将以太网可以分成四层，自上而下分别为：应用层、传输层、网络互联层、网络接口层。

根据我自己的理解，把以太网分成五层比较容易解释。这五层结构不仅符合 OSI 结构强调的不同层次承担不同职责的特点，同时也符合 TCP/IP 协议参考模型协议之间相互支撑、相互调用的逻辑关系。

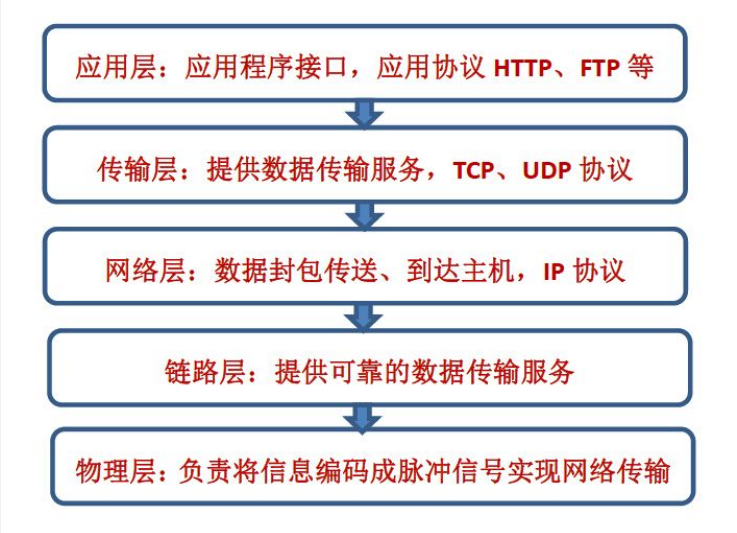


图 1-1-1 以太网五层模型

如上图所示，最底下的一层叫做“物理层”，也叫“PHY 层”，最上面的一层叫做“应用层”，中间的三层（自下而上）分别是“链路层”，也叫“MAC 层”、“网络层”和“传输层”。越下面的层，越靠近硬件；越上面的层，越靠近用户。

1.2 层与协议

每一层都有其各自的功能。为了实现这些功能，就需要大家都遵守一个共同的规则。

那么这个共同的规则，就叫做“协议”（Protocol）。以太网的每一层都定义了很多协议。这些协议的总称就叫做“互联网协议”（Internet Protocol Suite）。它们是互联网的核心，下面介绍每一层的功能，及其中的主要协议。

第 2 章　以太网分层概述

下面对以太网的五层结构模型进行详细解释，让大家对网络的通信过程、每层的具体定义和功能、数据收发机制以及要遵守的协议进行理解。首先，大家要知道在不同层由于封包机制不同，数据的叫法也不同，这样有利于大家更好的理解下面的内容。传输层叫数据段（Segment），网络层叫数据报（Datagram），链路层叫数据帧（Frame）。

2.1 物理层

我们从最底下的一层开始。

物理层也叫“PHY 层”，它负责将上层所要发送的信息编码成电流脉冲或其它信号用于网上传输。

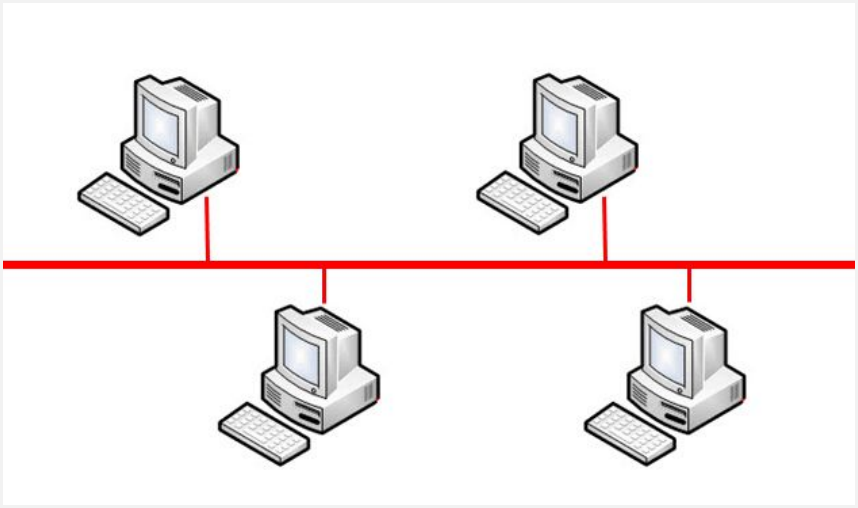


图 2-1-1 计算机的网络连接

物理层由计算机和网络介质之间的实际界面组成，可定义电气信号、符号、线的状态和时钟要求、数据编码和数据传输用的连接器。如最常用的 RS-232 规范、10BASE-T 的曼彻斯特编码以及 RJ-45 就属于这一层。所有比物理层高的层都通过事先定义好的接口而与它通话。

2.2 链路层

2.2.1 定义

数据链路层通过物理网络链路提供可靠的数据传输。不同的数据链路层定义了不同的网络和协议特征，其中包括物理编址、网络拓扑结构、错误校验、帧序列以及流控。

物理编址（相对应的是网络编址）定义了设备在数据链路层的编址方式；网络拓扑结构定义了设备的物理连接方式，如总线拓扑结构和环拓扑结构；错误校验向发生传输错误的上层协议告警；数据帧序列重新整理并传输除序列以外的帧；流控可能延缓数据的传输，以使接收设备不会因为在某一时刻接收到超过其处理能力的信息流而崩溃。

2.2.2 以太网协议

早期的时候，每家公司都有自己的电信号分组方式，后来逐渐形成了以“以太网（Ethernet）”为主的一整套协议。

以太网规定，一组电信号构成一个数据包，叫做“帧”（Frame）。每一帧分成三个部分：以太网首部、数据及以太网尾部。

图 2-2-1 以太网数据帧结构

“以太网首部”包含数据帧的一些说明项，比如发送者、接收者、数据类型等等；“数据”部分则是数据的具体内容；“以太网尾部”则是 CRC 校验码。

“以太网首部”的长度固定为 14 字节。“数据”的长度，最短为 46 字节，最长为 1500 字节。“以太网尾部”的长度固定位 4 字节。因此，整个“帧”最短为 64 字节，最长为 1518 字节。如果数据很长，就必须分割成多个帧进行发送。

2.2.3 MAC 地址

上面提到，以太网数据帧的“首部”，包含了发送者和接收者的信息。那么，发送者和接受者是如何标识呢？

以数据链路层实际上由两个独立的部分组成，介质存取控制（Media Access Control,MAC）和逻辑链路控制层（Logical Link Control,LLC）。MAC 描述在共享介质环境中如何进行站的调度、发生和接收数据。MAC 确保信息跨链路的可靠传输，对数据传输进行同步，识别错误和控制数据的流向。一般地讲，MAC 只在共享介质环境中才是重要的，只有在共享介质环境中多个节点才能连接到同一传输介质上。IEEE MAC 规则定义了地址，也就是 MAC 地址，以标识数据链路层中的多个设备，因此链路层也叫“MAC 层”。

每块网卡出厂的时候，都有一个全世界独一无二的 MAC 地址，长度是 48 个二进制位，通常用 12 个十六进制数表示。

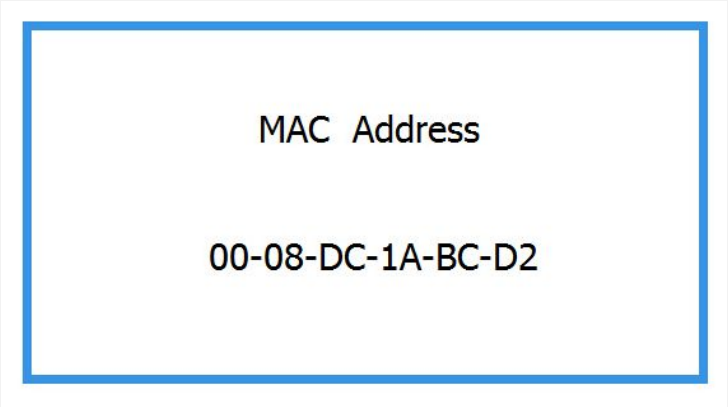


图 2-2-2 MAC 地址

前 6 个十六进制数是厂商编号，后 6 个是该厂商的网卡流水号。有了 MAC 地址，就可以定位网卡和数据包的路径了。

2.2.4 广播

定义地址只是第一步， 那么一块网卡怎么会知道另一块网卡的 MAC 地址？ARP 协议可以解决这个问题。这个留到后面介绍，这里只需要知道，以太网数据帧必须知道接收方的 MAC 地址，然后才能发送。

其次，就算有了 MAC 地址，系统怎样才能把数据帧准确送到接收方？

其实，以太网采用了一种很“原始”的方式，它不是把数据帧准确送到接收方，而是向本网络内所有计算机发送，让每台计算机自己判断，是否为接收方。它们读取这个帧的“首部”，找到接收方的 MAC 地址，然后与自身的 MAC 地址相比较，如果两者相同，就接受这个帧，做进一步处理，否则就丢弃这一帧。这种发送方式就叫做“广播”（broadcasting）。

有了数据帧的定义、网卡的 MAC 地址、广播的发送方式，“链路层”就可以在多台计算机之间传送数据了。

2.3 网络层

2.3.1 定义

网络层负责在源和终点之间建立连接。它一般包括网络寻径，还可能包括流量控制、错误检查等。相同 MAC 标准的不同网段之间的数据传输一般只涉及到数据链路层，而不同的 MAC 标准之间的数据传输都涉及到网络层。例如 IP 路由器工作在网络层，因而可以实现多种网络间的互联。

2.3.2 IP 协议

规定网络地址的协议，叫做 IP 协议。它所定义的地址，就被称为 IP 地址。目前，广泛采用的是 IP 协议第四版，简称 IPv4。这个版本规定，网络地址由 32 个二进制位组成。

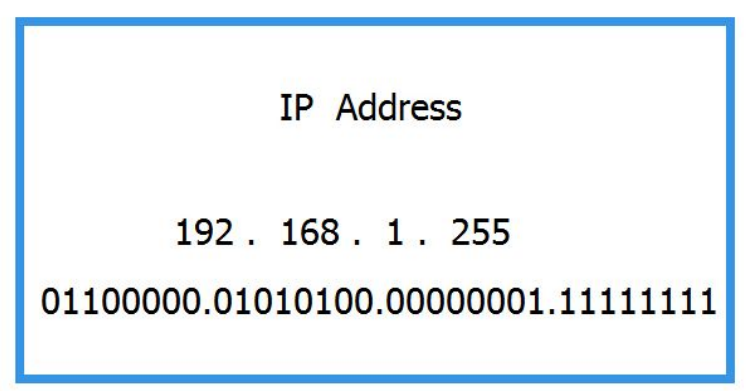


图 2-3-1 IP 地址

习惯上，我们用分成四段的十进制数表示 IP 地址，从 0.0.0.0 一直到 255.255.255.255。

互联网上的每一台计算机，都会分配到一个 IP 地址。这个地址分成两个部分，前一部分代表网络，后一部分代表主机。比如，IP 地址 172.16.254.1，这是一个 32 位的地址，假定它的网络部分是前 24 位（172.16.254），那么主机部分就是后 8 位（最后的那个 1）。处于同一个子网络的电脑，它们 IP 地址的网络部分必定是相同的，也就是说 172.16.254.2 应该与 172.16.254.1 处在同一个子网络。

2.3.3 IP 数据报

根据 IP 协议发送的数据，就叫做 IP 数据报。不难想象，其中必定包括 IP 地址信息。

但是前面说过，以太网数据帧只包含 MAC 地址，并没有 IP 地址的信息。那么是否需要修改数据定义，再添加 IP 地址信息呢？

答案是不需要，我们可以把 IP 数据报直接放进以太网数据帧的"数据"部分，因此完全不用修改以太网的规格。这就是互联网分层结构的好处：上层的变动完全不涉及下层的结构。

具体来说，IP 数据报分为“标头”和“数据”两个部分。

2-3-2 IP 数据报结构

“标头”部分主要包括版本、长度、IP 地址等信息，“数据”部分则是 IP 数据报的具体内容。

IP 数据报的“标头”部分长度为 20 到 60 字节，整个数据报的总长度最大为 65,535 字节。因此理论上，一个 IP 数据报的“数据”部分，最长为 65,515 字节。前面说过，以太网数据帧的“数据”部分，最长只有 1500 字节。因此，如果 IP 数据报超过了 1500 字节，它就需要分割成几个以太网数据帧，分开发送了。

2.4 传输层

2.4.1 定义

传输层向高层提供可靠的端到端的网络数据流服务。传输层的功能一般包括流控、多路传输、虚电路管理及差错校验和恢复。流控管理设备之间的数据传输，确保传输设备不发送比接收设备处理能力大的数据；多路传输使得多个应用程序的数据可以传输到一个物理链路上；虚电路由传输层建立、维护和终止；差错校验包括为检测传输错误而建立的各种不同结构；而差错恢复包括所采取的行动（如请求数据重发），以便解决发生的任何错误。

2.4.2 UDP 协议

我们必须在数据包中加入端口信息，这就需要新的协议。最简单的实现叫做 UDP 协议，UDP 数据段也是由“标头”和“数据”两部分组成。

2-4-1 UDP 数据段结构

“标头”部分主要定义了发出端口和接收端口，“数据”部分就是具体的内容。然后，把整个 UDP 数据段放入 IP 数据报的“数据”部分，而 IP 数据报又是放在以太网数据帧之中的。

UDP 数据段非常简单，“标头”部分一共只有 8 个字节，总长度不超过 65,535 字节，正好放进一个 IP 数据报。

2.4.3 TCP 协议

为了提高网络可靠性，诞生了 TCP 协议。这个协议非常复杂，但可以近似认为，它就是有确认机制的 UDP 协议，每发出一个数据都要求确认。如果有一个数据遗失，就收不到确认，发出方就知道有必要重发这个数据了。

TCP 协议能够确保数据不会遗失，缺点是过程复杂、实现困难、消耗较多的

资源。TCP 数据段和 UDP 数据段一样，都是内嵌在 IP 数据报的“数据”部分。TCP 数据段没有长度限制，理论上可以无限长，但是为了保证网络的效率，通常 TCP 数据段的长度不会超过 IP 数据报的长度，以确保单个 TCP 数据段不必再分割。

2.5 应用层

2.5.1 定义

应用层是最接近终端用户的第一层，这就意味着应用层与用户之间是通过应用软件直接相互作用的。注意，应用层并非由计算机上运行的实际应用软件组成，而是由向应用程序提供访问网络资源的 API（应用程序接口）组成。应用层的功能一般包括标识通信伙伴、定义资源的可用性和同步通信。因为可能丢失通信伙伴，应用层必须为传输数据的应用子程序定义通信伙伴的标识和可用性。定义资源可用性时，应用层为了请求通信而必须判定是否有足够的网络资源。在同步通信中，所有应用程序之间的通信都需要应用层的协同操作。

2.5.2 应用层协议

应用程序收到“传输层”的数据，接下来就要进行解读。由于互联网是开放架构，数据来源五花八门，必须事先规定好格式，否则根本无法解读。“应用层”的作用，就是规定应用程序的数据格式。

应用层的 HTTP（超文本传输）协议、DNS（域名解析）协议、FTP（文件传送）协议、SMTP（简单邮件管理）协议等。

举例来说，TCP 协议可以为各种各样的程序传递数据，比如发 Email 用的 SMTP（简单邮件管理）协议、网上冲浪用到的 HTTP（超文本传输）协议、下载资料用到的 FTP（文件传送）协议等等，这些应用程序协议就构成了“应用层”。

这是最高的一层，直接面对用户。它的数据就放在 TCP 数据段的“数据”部分。因此，现在的以太网的数据帧就变成下图这样。

图 2-5-1 以太网数据帧整体结构

至此，整个以太网的五层结构就介绍完毕。包括计算机和单片机在内的任何设备需要联网，就必须搭建这五层物理连接以及处理层内和层与层之间的 TCP/IP 协议方能实现网络应用。下面第 3 章就介绍如何通过这几层的衔接来实现单片机网络连接的两种不同方案。

第 3 章　以太网接入方案

第 1 章和第 2 章我们介绍了以太网的五层结构模型及各层所要实现的功能，按照这一模型诞生出了各式各样的单片机网络连接方案来满足客户的不同要求。单片机的种类繁多，从低端到高端，有以 51 单片机为代表的 8 位单片机和以 ARM 为代表的 32 位单片机，不同档次的单片机实现网络接口的方法不同。对于像 ARM 等高端处理器一般都可以运行嵌入式操作系统，例如嵌入式 Linux。对于无操作系统要求的单片机如何实现网络接入，我下面将这些方案按 TCP/IP 协议栈的不同归结为两大类：第一类是传统的软件 TCP/IP 协议栈方案；第二类是最新的硬件 TCP/IP 协议栈方案。下面我就这两类方案的实现方式进行分析。

3.1 MAC+PHY 方案

所谓的 TCP/IP 协议栈是一系列网络协议的统称，不仅包括我们熟知的 TCP 协议和 IP 协议，还有网络层的 ICMP（Internet 控制报文）协议、IGMP（Internet 组管理）协议、ARP（地址解析）协议，传输层的 UDP（用户数据包）协议，应用层的 HTTP（超文本传输）协议、DNS（域名解析）协议、FTP（文件传送）协议、SMTP（简单邮件管理）协议等等。

传统的以太网接入方案如下图，由 MCU+MAC+PHY 再加入网络接口实现以太网的物理连接，通过在主控芯片中植入 TCP/IP 协议代码实现通信及上层应用。

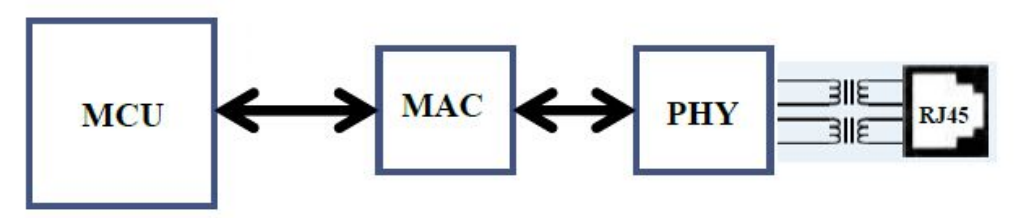


图 3-1-1 MAC+PHY 以太网方案

应用这种软件 TCP/IP 协议栈方式实现的比较成熟方案有 ENC28J60，CS8900A，DM9000，当然也有像 STM32F107 这类（内部自带 MAC）+PHY 等方

案。

由于软件协议栈操作需要主控 MCU 不断地响应中断，这在很大程度上占用了 MCU 的运算/时钟资源。经过测试发现，单线程操作的情况下，MCU 的运行速度和数据的处理速度仅能满足需要，但随着线程增多，MCU 的工作效率直线下降，会严重影响通信质量。

代码量方面，即便是采用轻量级的 TCP/IP 协议栈 LWIP 协议，也会为主控芯片带来超过 40KB 的代码量，这对于本身内存资源匮乏的单片机来说负荷过重。

再从安全性的角度，设备并入互联网之后必须考虑网络安全问题，这种软件协议栈的方式系统一旦受到复杂的恶意攻击，单片机很有可能瘫痪掉，这对系统就是致命性打击，虽然目前网络技术不断发展，各类新的加密技术试图让通信变得更加安全，但是还会出现各种各样的漏洞。

3.2 硬件协议栈芯片方案

硬件协议栈芯片方案如下图所示。由 MCU+硬件协议栈芯片（内含 MAC 和 PHY）直接加网络接口，便可方便的实现单片机联网，所有的处理 TCP/IP 协议的工作都是通过这位 MCU 的“小秘书”——硬件协议栈芯片来完成。

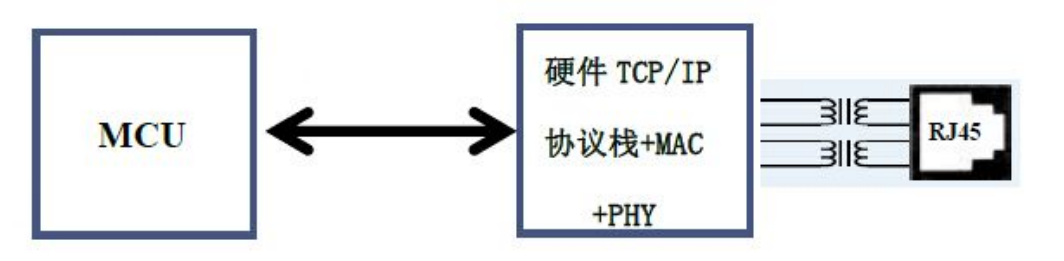


图 3-2-1 硬件协议栈芯片方案

这套方案是由 WIZnet 首次提出，并成功推出以太网系列芯片：W5100、W5200、W5300 和 W5500。

所谓硬件协议栈是指通过将传统的软件 TCP/IP 协议栈用硬件化的逻辑门电路来实现，如下图所示。

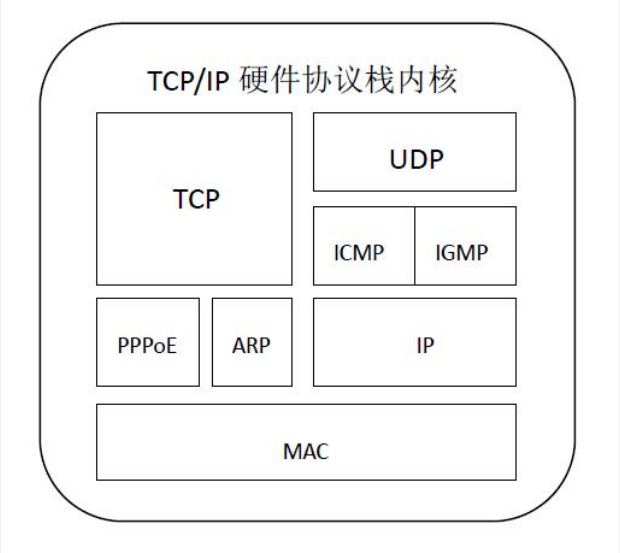


图 3-2-2 TCP/IP 硬件协议栈内核原理简图

以太网芯片的内核由传输层的 TCP、UDP、ICMP、IGMP 等协议、网络层的 IP、ARP、PPPoE 等协议以及链路层的 MAC 构成，再加上物理层的 PHY 和外围的寄存器、内存、SPI 接口组成了这一整套硬件化的以太网解决方案。

这套硬件 TCP/IP 协议栈代替了以往的 MCU 来处理这些中断请求，即 MCU 只需要处理面向用户的应用层数据即可，传输层、网络层、链路层及物理层全部由外围 WIZnet 的芯片完成。这套方案从硬件开销和软件开发两个方面来简化前面所述的五层网络模型，简化产品开发方案。这样一来，工程师们就不必再面对繁琐的通信协议代码，只需要了解简单的寄存器功能以及 Socket 编程便能完成产品开发工作的的网络功能开发部分。

由于硬件协议栈的加入协助单片机处理了几乎所有的 TCP/IP 协议工作，不仅极大地减少了单片机的中断次数，让单片机腾出更多资源去完成其他工作，而且硬件化的电路处理协议会更加快速、稳定。经试验测试，单线程下，该方案的通信速度是软件协议方案的 10 倍左右；随着线程的增加，因为硬件协议栈是通过独立的 Socket 进行通信，因而通信速度实现累加，而且单片机工作效率仍然会维持在高位。

代码量方面，因为这套方案主要是完成对 Socket 的编程以及寄存器的调用，因此仅有 10K 左右的代码量，远小于软件协议方案，对 51 以及 STM32 等内存很有限的单片机来说非常适用。

从成本角度来讲，硬件协议栈芯片的价格跟用 MAC+PHY 比起来基本差不多。而前者简单易用，用很短时间便能完成产品的开发过程。另外，官方例程库及上位机程序丰富，也缩短了测试过程，后期基本免于维护。

最后安全性方面，硬件化的逻辑门电路来处理 TCP/IP 协议是不可攻击的，也就是说网络攻击和病毒对它无效，这也充分弥补了网络协议安全性不足的短板。也正是因为这一优势，硬件协议栈技术在未来物联网以及智能家居领域有着广泛的发展前景，让人们尽情享受现代科技带来的乐趣的同时，免受安全问题的困扰。

当然，不可避免的硬件化的协议栈相对来说失去了软件协议栈那样的灵活性。目前只支持 4 个/8 个 Socket，不能随时开启更多 Socket。但是，在嵌入式应用中 8 个 Socket 已经足够应对超过大部分的应用。

第 4 章　STM32+W5500 方案

第 3 章给大家介绍了一种简单易用的硬件化以太网接入方案，后面两章我们将通过野火官方开发板 ISO STM32 搭配野火 W5500 网络适配板实现以太网接入，并提供了上层应用例程的讲解部分，帮助大家更加深入地了解以太网硬件协议栈芯片 W5500，以及如何通过 STM32+W5500 轻松实现各种常见的应用协议。

4.1 实验描述

野火开发板 ISO STM32 如图 4-1-1 所示，它搭载 STM32F103ZET6 单片机，所有 IO 均以总线的方式引出，板载外设都有跳帽与 CPU 隔离，使用起来非常方便，是单片机初学者的最佳选择。

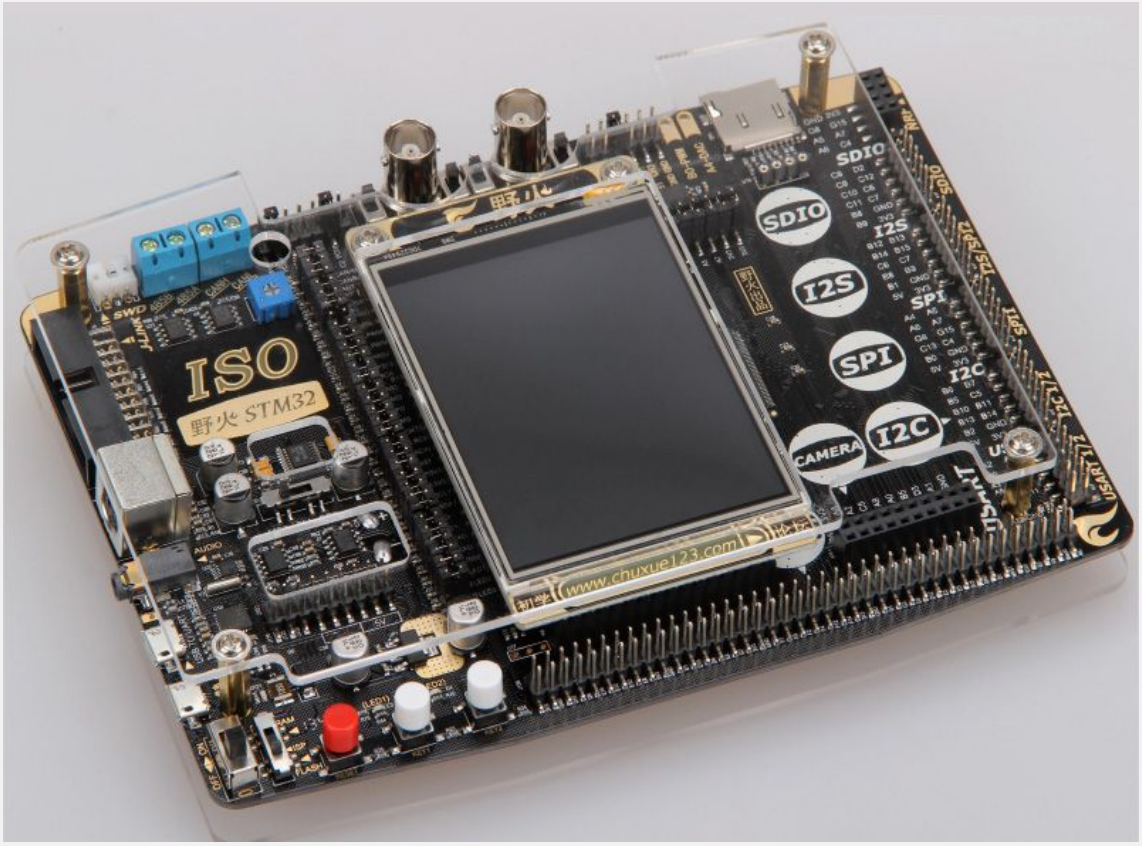


图 4-1-1 野火开发板 ISO STM32

我们选用野火开发板 ISO STM32 的 SPI1 接口与野火 W5500 网络适配板进行连接。野火开发板的 SPI1 接口电路图如图 4-1-2 所示。其中 1、2、3、4 脚则对

应 W5500 网络适配板的 SCSn, SCLK, MOSI, MISO 4 路信号。5 脚对应 W5500 的中断引脚，如果大家想通过中断方式实现联网功能，就需要定义次引脚。6 脚对应 W5500 的系统复位引脚。7、8、9 引脚尚未用到，所以 W5500 模块硬件设计为空脚，10、12 脚为 W5500 供电引脚。

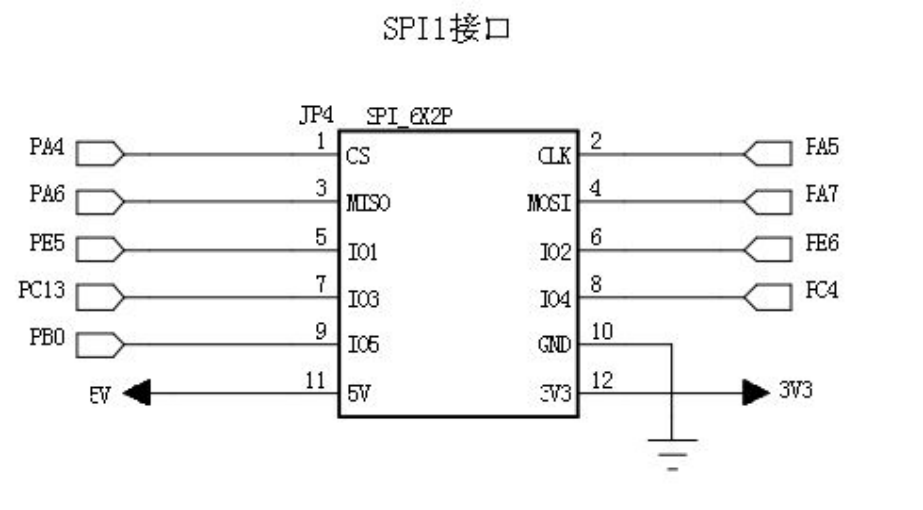


图 4-1-2 野火开发板 SPI1 接口

W5500 有两种工作模式，分别是可变数据长度模式和固定数据长度模式。在可变数据长度模式中，W5500 可以与其他 SPI 设备共用 SPI 接口。但是一旦将 SPI 接口指定给 W5500 之后，也就是 1 脚 CS 片选信号如果选中了这个 W5500，则不能再与其他 SPI 设备共用。在固定数据长度模式，SPI 将指定给 W5500，不能与其他 SPI 设备共享。因为 1 脚 CS 片选信号将直接接地为低电平。

W5500 的初始化过程主要用到以下几个函数：

W5500_conf.c 主要配置 W5500 的 MAC、IP 地址，W5500 基本的数据读写过程，复位设置函数等。

Socket.c 函数主要介绍了 W5500 的 SOCKET 相关配置函数，比如 SOCKET 的打开、关闭，以及接收数据、发送数据等等。

Utility.c 函数主要介绍了基本的延时函数，还有数据格式转化函数。

w5500.c 主要介绍 W5500 的寄存器读写过程。

4.2 W5500 的 Socket 初始化

接下来就详细介绍一下 W5500 具体的初始化过程。首先介绍基本设置，W5500 的操作需要设置以下寄存器的参数：

1，模式寄存器（MR）

2，中断屏蔽寄存器（IMR）

3，重发时间寄存器（RTR）

4，重发计数寄存器（RCR）

接下来就是设置网络信息，下面的寄存器是关于网络的基本配置，需要根据网络环境来进行设置。

1，网关地址寄存器（GAR）

2，本机物理地址寄存器（SHAR）

3，子网掩码寄存器（SUBR）

4，本机 IP 地址寄存器（SIPR）

本机物理地址寄存器（SHAR）的地址是 MAC 层的硬件地址，这是生产商指定使用的地址。MAC 地址可由 IEEE 指定。

最后介绍设置端口存储信息，这一步设置端口 TX/RX 存储信息，每个端口的基地址和屏蔽地址在这里确定并保存。W5500 有 1 个通用寄存器，8 个 Socket 寄存器区，以及对应每个 Socket 的收/发缓存区。每一个 Socket 的发送缓存区都在一个 16KB 的物理发送内存中，初始化分配为 2KB。每一个 Socket 的接收缓存区都在一个 16KB 的物理接收内存中，初始化分配为 2KB。无论给每个 Socket 分配多大的收/发缓存，都必须在 16 位的偏移地址范围内（从 0x0000 到 0xFFFF）。W5500 有一个 16KB 的发送内存用于 Socket n 的发送缓存区，以及一个 16KB 的接收内存用于 Socket n 的接收缓存区。

16KB 的发送内存初始化被分配为每个 Socket 2KB 发送缓存区（2KB X 8 = 16KB）。初始化分配的 2KB Socket 发送缓存，可以通过使用 Socket 发送缓存大小

寄存器（Sn_TXBUF_SIZE）重新分配。一旦所有的 Socket 发送缓存大小寄存器（Sn_TXBUF_SIZE）配置完成，16KB 的发送内存就会按照配置分配给每个 Socket 的发送缓存，并按照从 Socket 0 到 7 顺序分配。16KB 物理内存的地址是可以自增的。但是，为了避免数据传输错误，需要避免发送缓存大小寄存器（Sn_TXBUF_SIZE）的和超过 16。

16KB 的发送内存中分配了对应 Socket n 的发送缓存区，用于为来自主机传输的数据做缓存。Socket n 发送缓存区的 16 位偏移地址支持 64KB 的寻址范围（从 0x000 到 0xFFFF），关于他的配置请参考‘Socket n 发送写指针寄存器(Sn_TX_WR)’以及 Socket n 发送读指针寄存器(Sn_RX_WR)。然而，这 16 位偏移地址会自动转化为指定的 16KB 发送内存的物理地址。

16KB 的读取内存的分派方式与 16KB 的发送内存一样。16KB 的接收内存初始化被分配为每个 Socket 2KB 接收缓存区（2KB X 8 = 16KB）。初始化分配的 2KB Socket 接收缓存，可以通过使用 Socket 接收缓存大小寄存器（Sn_XBUF_SIZE）重新分配。一旦所有的 Socket 发缓存大小寄存器（Sn_TXBUF_SIZE）配置完成，16KB 的发送内存就会按照配置分配给每个 Socket 的发送缓存，并按照从 Socket 0 到 7 顺序分配。16KB 物理内存的地址是可以自增的。但是，为了避免数据传输错误，需要避免发送缓存大小寄存器（Sn_TXBUF_SIZE）的和超过 16。

16KB 的接收内存中分配了对应 Socket n 的接收缓存区，用于为来自网络传输的数据做缓存。Socket n 接收缓存区的 16 位偏移地址支持 64KB 的寻址范围（从 0x000 到 0xFFFF），关于他的配置请参考‘Socket n 接受读指针寄存器(Sn_RX_RD)’以及 Socket n 接受写指针寄存器(Sn_RX_WR)。然而，这 16 位偏移地址会自动转化为指定的 16KB 接收内存的物理地址。代码清单 4-1 就主要定义每一个 SOCKET 接收和发送缓存的大小。

代码清单 4-1 SOCKET 接收和发送缓存的大小配置文件

```
1. void socket_buf_init( uint8 * tx_size, uint8 * rx_size  )
2. {
3.   int16 i;
4.   int16 ssum=0,rsum=0;
5.   for (i = 0 ; i < MAX_SOCK_NUM; i++)
8.   {
9.      IINCHIP_WRITE( (Sn_TXMEM_SIZE(i)), tx_size[i]);
10.     IINCHIP_WRITE( (Sn_RXMEM_SIZE(i)), rx_size[i]);
11.     #ifdef __DEF_IINCHIP_DBG__
12.     printf("tx_size[%d]: %d, Sn_TXMEM_SIZE = %d\r\n",i,
13.             tx_size[i], IINCHIP_READ(Sn_TXMEM_SIZE(i)));
14.     printf("rx_size[%d]: %d, Sn_RXMEM_SIZE = %d\r\n",i,
15.             rx_size[i], IINCHIP_READ(Sn_RXMEM_SIZE(i)));
16.     #endif
17.             SSIZE[i] = (int16)(0);
18.             RSIZE[i] = (int16)(0);
19.             if (ssum <= 16384)
20.             {
21.                     SSIZE[i] = (int16)tx_size[i]*(1024);
22.             }
23.             if (rsum <= 16384)
24.             {
25.                     RSIZE[i]=(int16)rx_size[i]*(1024);
26.             }
27.             ssum += SSIZE[i];
28.             rsum += RSIZE[i];
29.     }
30. }
```

4.3 应用层协议开发

以太网的应用层包括支撑协议和应用协议两部分。

支撑协议：域名服务系统（DNS），简单网络管理协议（SNMP）等，典型应用包过 Web 浏览、电子邮件、文件传输访问、远程登录等。

应用协议：超文本传输协议（HTTP），简单邮件传输协议（SMTP），文件传输协议（FTP），简单文件传输协议（TFTP）和远程登录（Telnet）。

图 4-3-1 为 TCP/IP 协议族及上层的应用协议，前面介绍了硬件协议栈实现了整个 TCP/IP 的功能，因此我们距离具体的用户界面的应用实现只差一步了。那么下一章我们将带领大家运用野火官方开发板 ISO STM32+野火 W5500 网络适配板实现这一过程。

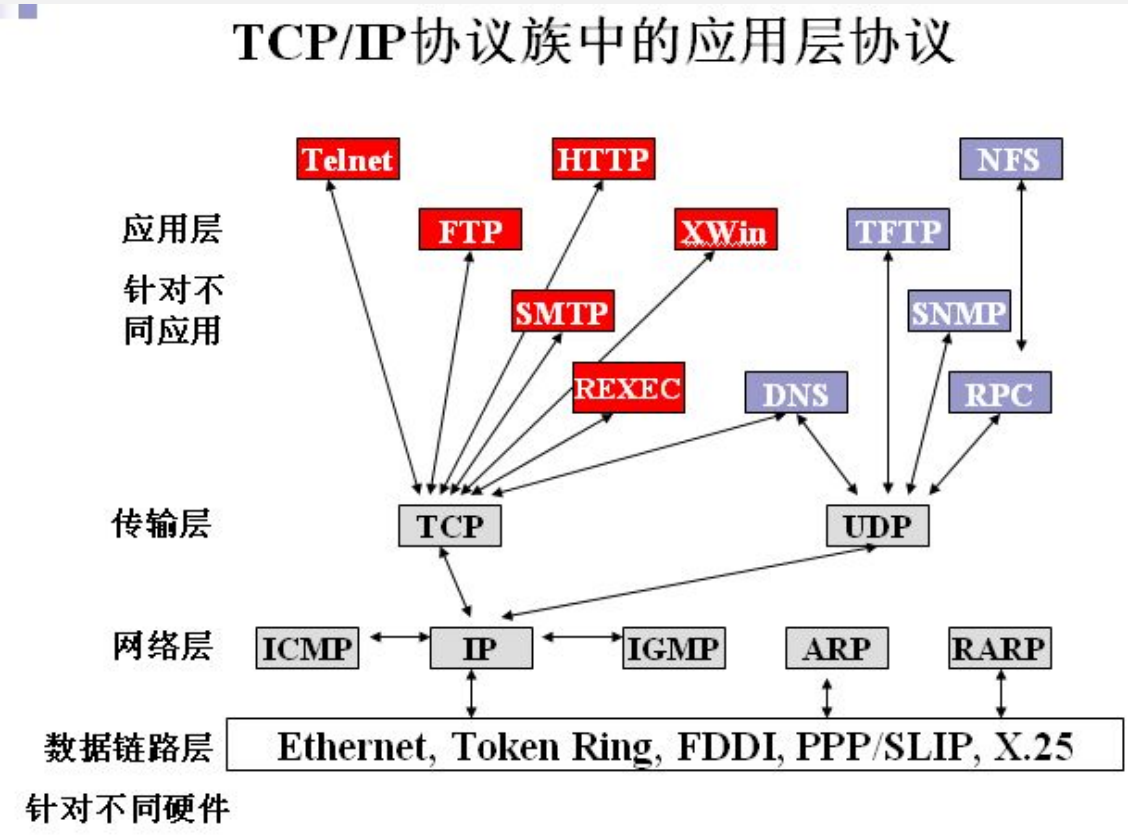


图 4-3-1 TCP/IP 协议族

第 5 章 例程讲解

以太网控制芯片 W5500 在内部利用硬件实现了 TCP/IP 协议栈，即内部结构包含了物理层、数据链路层、网络层和传输层。全硬件 TCP/IP 协议栈完全独立于主控芯片，可以降低主芯片负载且无需移植繁琐的 TCP/IP 协议栈，便于产品实现网络化更新。接下来就对基于 W5500 的应用层协议例程进行讲解，需要注意的是，下面的例程均在 Keil 4.72 的开发环境下进行。

5.1 TCP Server/Client

TCP 是整个协议簇的核心协议之一，是 TCP/IP 体系中面向连接的传输层协议，它使用 IP 作为网络层，提供全双工的和可靠交付的服务。TCP 建立通信的两端一端称为服务器端，另一端为客户端。服务器端，指网络中能为用户提供服务的计算机系统，是为客户端服务的；客户端是指与服务器相对应，它是接受服务的一段，为客户提供本地服务的程序。

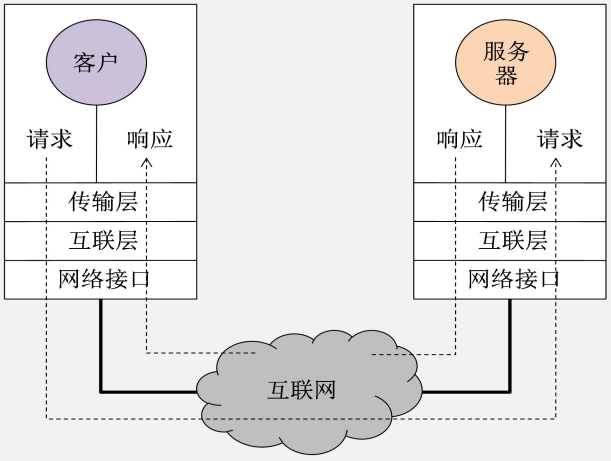


图 5-1-1 客户/服务器交互模型

如图 5-1-1 所示为一个通过互联网进行交互的 C/S 模型，C/S 模型是一个最典型和最常用的通讯结构。此时服务器处于守候状态，并侦听客户端的请求。客户端发出请求，并请求经互联网送给服务器。一旦服务器接收到这个请求，就可以执行请求所制定的任务，并将执行的结果经互联网回送给客户。

TCP 协议通过三个报文段完成连接的建立，这个过程称为三次握手（three-way handshake）。TCP 连接建立过程如图 5-1-2 所示。

1，第一次握手：建立连接时，客户端发送 SYN 包(seq=j)到服务器，并进入 SYN_SEND 状态，等待服务器确认。

2，第二次握手：服务器收到 SYN 包，必须确认客户的 SYN（ack=j+1），同时自己也发送一个 SYN 包（seq=k），即 SYN+ACK 包，此时服务器进入 SYN_RECV 状态。

3，第三次握手：客户端收到服务器的 SYN＋ACK 包，向服务器发送确认包 ACK(ack=k+1)，此包发送完毕，客户端和服务器进入 ESTABLISHED 状态，完成三次握手。

建立一个连接需要三次握手，而终止一个连接要经过四次挥手，这是由 TCP 的半关闭（half-close）造成的，如图 5-1-2 所示。

1，第一次挥手：主动方发出设置了 FIN 位的报文，表示主动终止从本地到远端的单向连接；此时，主动方进入 FIN_WAIT1 状态，意思就是它在等待远端的 FIN 报文。

2，第二次挥手：远端收到 FIN 后，会立即发送 ACK；主动方收到 ACK 后，进入 FIN_WAIT2 状态，所以 FIN_WAIT1 状态持续的时间非常短；此时远端进入 CLOSE-WAIT 状态，一条单向连接终止了，但另一条还没有，处于 HALF-CLOSE 连接状态。

3，第三次挥手：当远端进行了必要的数据发送后，它发送 FIN，表示从它出发的单向连接也要关闭；同时它进入 LAST ACK 状态。

4，第四次挥手：主动方收到 FIN 后，回应一个 ACK；远端就此进入 CLOSED 状态，连接关闭；主动方进入 TIME WAIT 状态；确保最后一个 ACK 没有丢失，防止新连接占用刚刚关闭的主动方的地址端口，使得网络中流浪的老连接的分组被误认为新连接的分组。

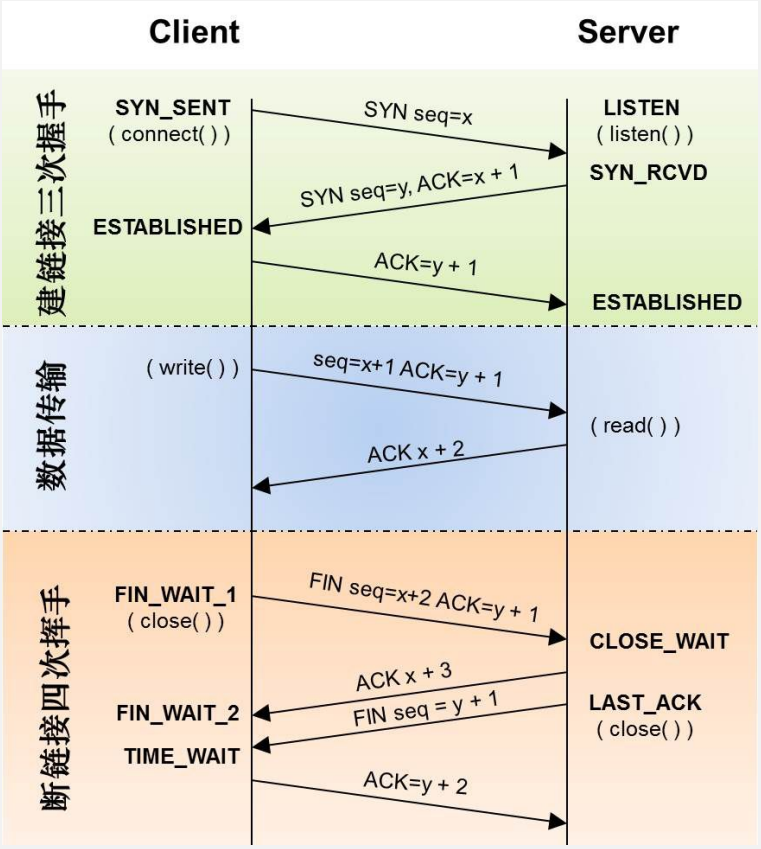


图 5-1-2 TCP 的连接和断开过程

5.1.1 TCP 服务器

下面介绍 W5500 作为 TCP 服务器的基本操作过程。当 W5500 初始化完成后，程序进入主循环，此时读取该 Socket 的状态值，并选择进入哪种模式。当 Socket 处于关闭状态时，在进行通信之前，我们先将该 Socket 初始化。这个 Socket 作为服务器端，端口号要固定为要侦听的端口。当 socket 将处于初始化完成状态即 SOCK_INIT 状态，此时，作为 TCP 服务器就要执行 listen()函数来侦听端口。由于 W5500 内嵌了 TCP/IP 协议，连接过程是不需要单片机干预的。如果连接过程中出错造成超时，该 Socket 将会被关闭，重新进入 SOCK_CLOSE 状态。待 TCP 连接的 3 次握手完成后，socket 的状态将会转变为连接建立状态，即代码中定义的 SOCK_ESTABLISHED 状态。在进入 SOCK_ESTABLISHED 状态后，便可进行数据收发。数据通信完毕之后即执行 disconnect()函数，在收到对方 FIN 数据包之前，该 Socket 将进入 SOCK_CLOSE_WAIT 状态。TCP 服务器模

式流程图如下：

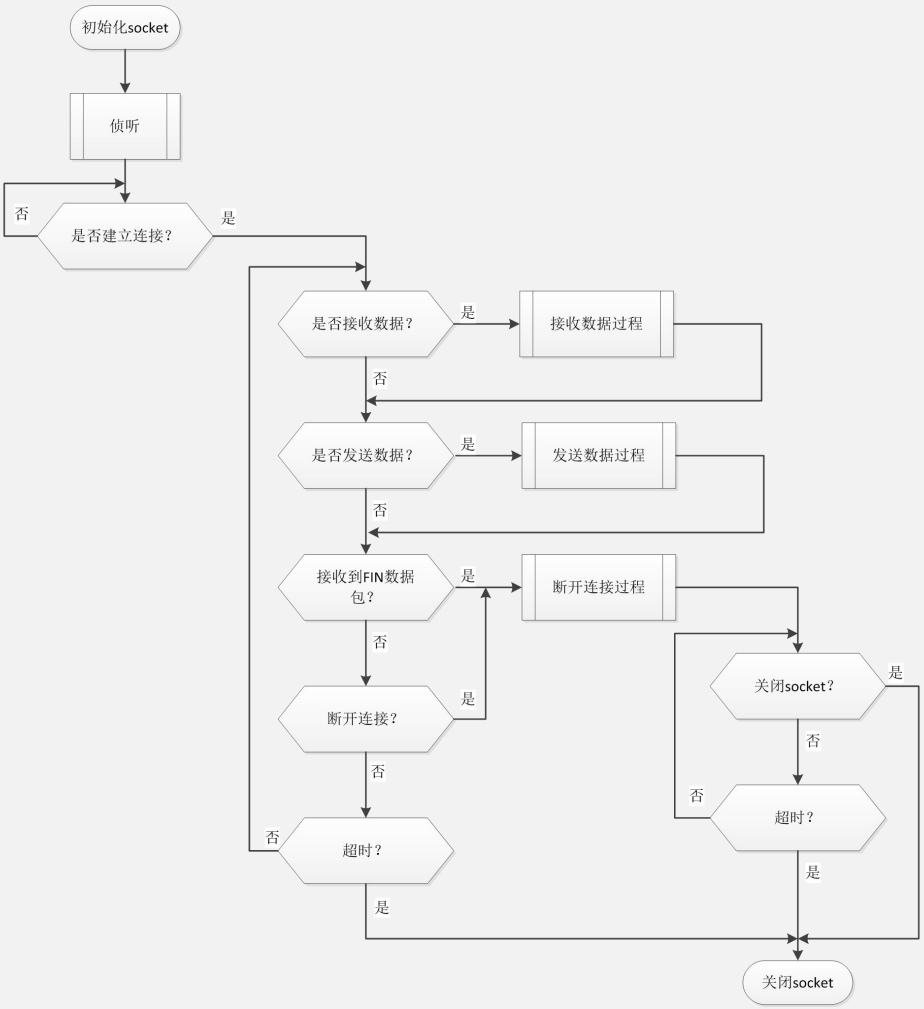


图 5-1-2 TCP Server 流程图

接下来我们对照代码讲解具体的测试步骤：

1，TCP 服务器采用默认的 IP 信息，所以在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DEFINE。

2，对代码进行编译，之后将程序烧录到野火开发板。

3，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输出结果可以得到图 5-1-3 设置信息。

4，在测试之前要注意两个问题，第一个就是建议关闭 PC 的防火墙；第二个问题，如果 W5500 模块与 PC 直接通过网线连接，需要修改 PC 的 IP 地址为静态 IP，且保持与 W5500 的 IP 在同一个网段，如果直接连接路由器，可以不用修改 PC 的 IP 地址。

图 5-1-3 W5500 基本配置信息

5，W5500 作为服务器，PC 作为客户端连接服务器。如图 5-1-3 所示，从串口打印信息可以得到 W5500 服务器的 IP 地址为 192.168.1.88，端口号为 5000。添加无误以后点击 connect 连接服务器，如果成功会显示本地主机的 IP 信息。

6，然后通过发送数据可以验证是否成功连接，如图 5-1-4 所示，说明 W5500 作为服务器建立成功。

图 5-1-4 通信测试

5.1.2 TCP 客户端

下面介绍 W5500 作为 TCP 服务器的基本操作过程，下图简单地描述了数据收发的过程。当 W5500 初始化完成后，程序进入主循环，可以调用 getSn_SR (Socket 号)来读取该 Socket 的状态值。当 Socket 处于关闭状态时，在进行通信之前，我们需要先将该 Socket 初始化，这里通信协议这里我们将配置成 TCP，即 Sn_MR_TCP。当程序成功执行 socket(...)函数后，socket0 将处于 SOCK_INIT 状态。此时，作为 TCP 客户端，就要调用 connect(...)函数连接远程服务器。待 TCP 连接的 3 次握手完成后，socket0 的状态将会转变为 SOCK_ESTABLISHED 状态。在进入 SOCK_ESTABLISHED 状态后，便可进行数据收发。

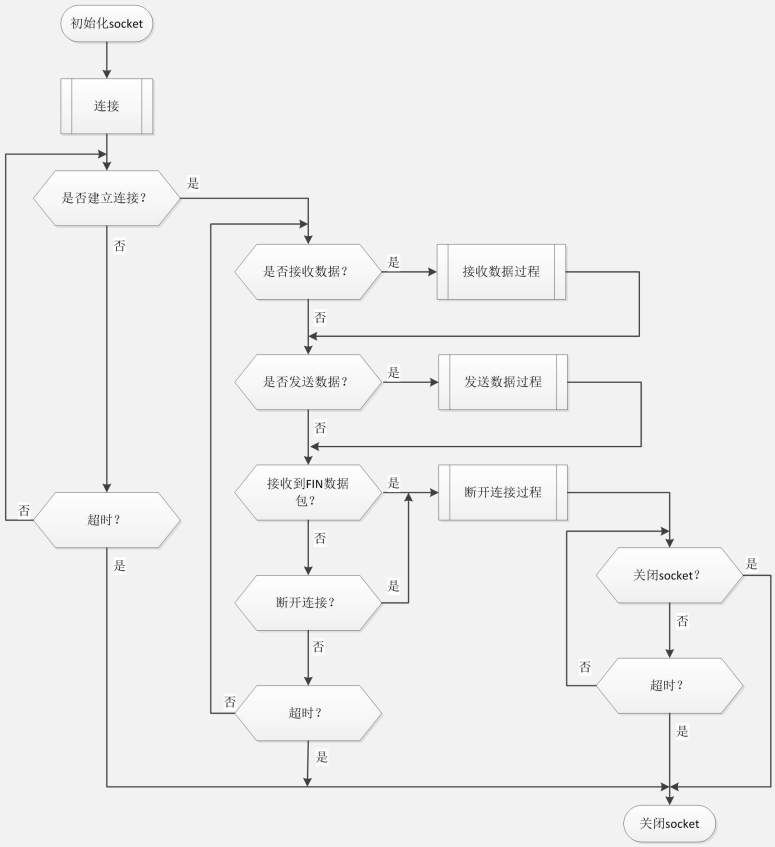


图 5-1-5 TCP Client 流程图

接下来我们对照代码讲解具体的测试步骤：

1，在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DEFINE。

2，在 Windows 下的具体操作是，开始→运行→（键入）cmd，输入 ipconfig →回车，将电脑 IP 地址赋给 w5500_conf.c 文件中的 remote_ip 变量。

3，对代码进行编译，之后将程序烧录到野火开发板。

4，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输出结果可以得到图 5-1-6 设置信息。

图 5-1-6 TCP 客户端配置信息

5，PC 作为服务器，W5500 作为客户端连接服务器。如图 5-1-7 所示，PC 作为服务器的监听端口号为 5000。添加无误以后点击 Listen 连接监听。

6，然后通过发送数据可以验证是否成功连接，观察图 5-1-7 中的信息，说明服务器建立成功。

图 5-1-7 通信测试

5.2 UDP

在 TCP/IP 协议栈的传输层中，TCP 面向连接，而接下来要演示的 UDP 协议则是面向非连接。下面介绍 W5500 作为 UDP 服务器的基本操作过程。如图 5-2-1 为 UDP 通信流程图。

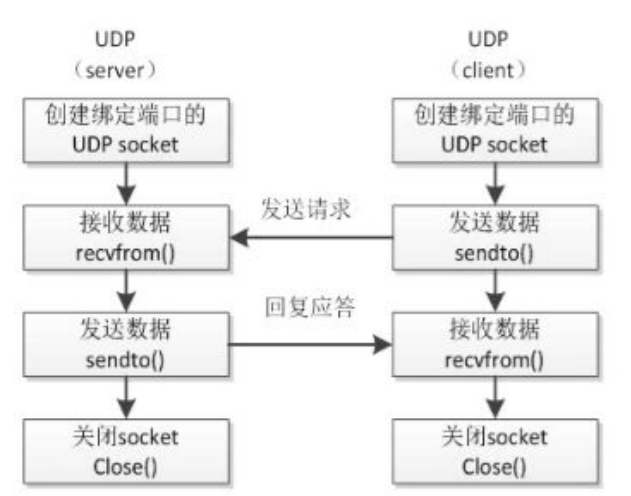


图 5-2-1 UDP 通信流程图

W5500 的 UDP 建立过程也是非常方便的，通过简单的读写寄存器便可以轻松实现。程序初始化完成以后，进入主循环函数。当 Socket 处于关闭状态时，在进行通信之前，我们先将该 UDP 模式的 Socket 端口初始化。当 socket 将处于初始化完成状态即 SOCK_UDP 状态，此时就可以通过广播方式发送数据了。

接下来使用 W5500 演示如何使用 UDP 发送和接收数据。在测试之前要注意两个问题，首先建议关闭 PC 的防火墙；其次，若 W5500 模块与 PC 直接通过网线连接，需要修改 PC 的 IP 地址为静态 IP，且保持与 W5500 的 IP 在同一个网段；如果直接连接路由器，可以不用修改 PC 的 IP 地址。具体的测试步骤为：

1，在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DEFINE。

2，在 Windows 下的具体操作是，开始→运行→（键入）cmd，输入 ipconfig →回车。将电脑 IP 地址赋给 w5500_conf.c 文件中的 remote_ip 变量。

3，对代码进行编译，之后将程序烧录到野火开发板。

4，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输出结果可以得到图 5-2-2 设置信息。

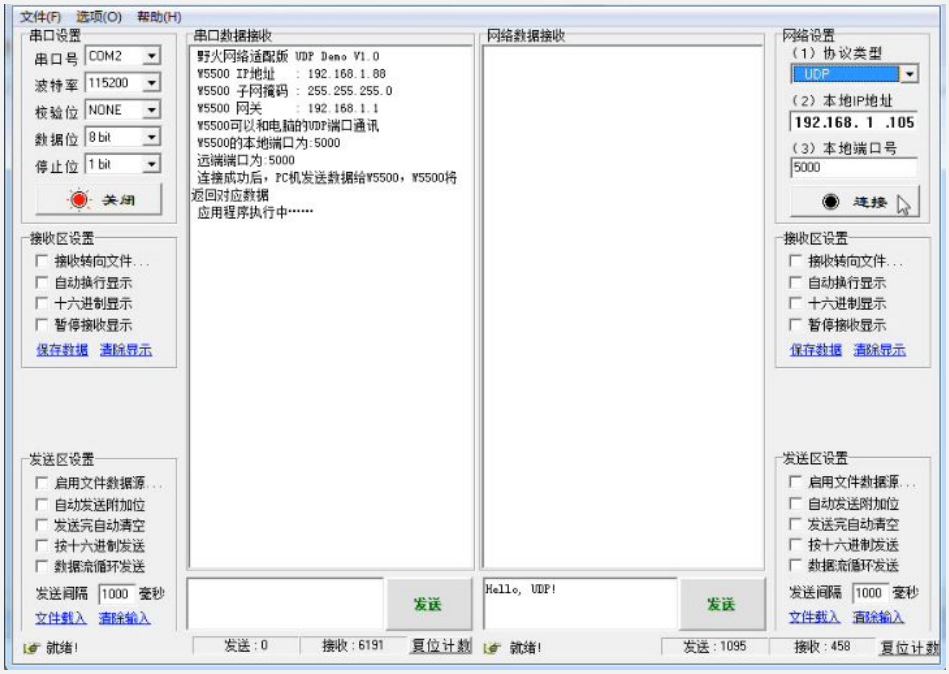


图 5-2-2 W5500 的 UDP 初始化信息

5，如图 5-2-2 所示，设置 Module IP 为单片机的 IP：192.168.1.88，Port 为 5000；Local port 设置为计算机的端口 5000，点击“连接”。

6，此时就可以通过发送数据进行通信测试了，通信效果如图 5-2-3 所示。这就说明 W5500 已经迅速通过 UDP 实现了数据的收发。

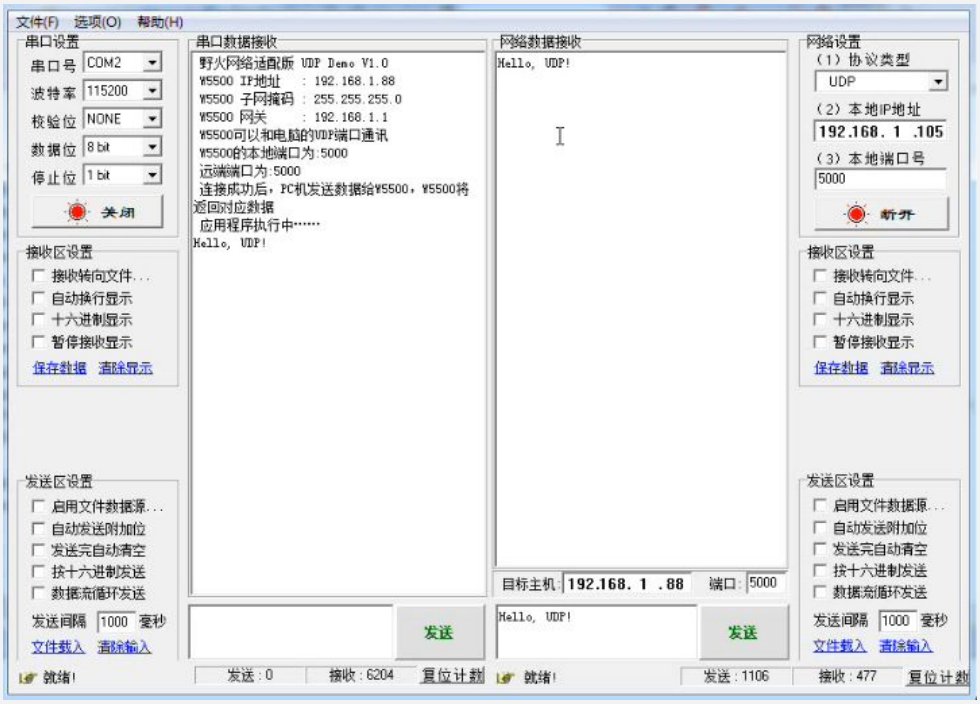


图 5-2-3 通信测试

5.3 DHCP

本节中 W5500 作为 DHCP 客户端，路由器作为 DHCP 服务器端。在 DHCP 请求的过程中，包括 4 个主要的阶段：发现阶段、提供阶段、选择阶段以及确认阶段。下面来说一下整个 DHCP 请求流程。

首先 W5500 客户端发送 DHCPDISCOVER 消息（IP 地址租用申请），这个消息通过广播方式发出，所有网络中的 DHCP 服务器都将接收到这个消息。 随后，网络中的 DHCP 服务器会回应一个 DHCPOFFER 消息（IP 地址租用提供），由于这个时候客户端还没有网络地址，所以 DHCPOFFER 也是通过广播的方式发送出去的。然后，向该服务器发送 DHCPREQUEST 消息。在 DHCPREQUEST 消息中将包含客户端申请的 IP 地址。 最后，DHCP 服务器将回送 DHCPACK 的响应消息来通知客户端可以使用该 IP 地址，该确认里面包含了分配的 IP 地址和该地址的一个稳定期限的租约（默认是 8 天），并同时更新 DHCP 数据库。

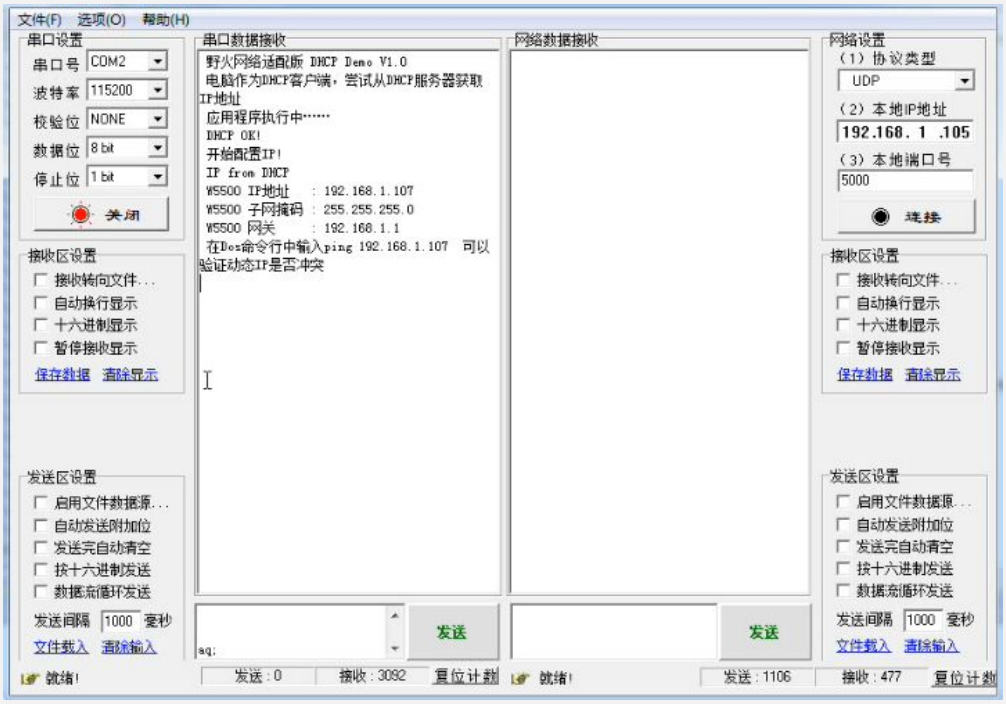
以上是 W5500 作为 DHCP 客户端向 DHCP 服务器申请 IP 地址的一个过程，下面对照 W5500 的 DHCP 执行函数来看一下具体操作过程。W5500 发送广播封包并接收路由器的 IP 地址分配，完成了完整的 DHCP 工作过程。DHCP 初始化完成以后，通过获取 DHCP_SOCKET 的状态来获取动态 IP 地址。第一个状态 DHCP_RET_NONE 就是获取不成功，第二个状态 DHCP_RET_TIMEOUT 是获取 IP 地址超时，都不符合条件。第三个状态 DHCP_RET_UPDATE 就是获取动态 IP 地址成功，此时将得到的 IP 地址通过 SPI 写入 W5500 的寄存器。如果是 DHCP_RET_CONFLICT 冲突状态，就要返回 DHCP 服务重新获取。

主程序除了初始化 W5500 之外，在主循环里面主要完成了 DHCP 状态机。具体的程序测试步骤如下：

1，DHCP 例程将动态获取的 IP 信息配置给 W5500，所以在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DHCP。

2，对代码进行编译，之后将程序烧录到野火开发板。

3，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输出结果可以得到图 5-3-1 设置信息。



5-3-1 DHCP 动态获取信息

4，在 Windows 下的具体操作是，开始→运行→（键入）cmd，输入 ping 192.168.1.109→回车，看到回复信息如图 5-3-2 所示。

图 5-3-2 ping DHCP 获取 IP 地址

可以成功 ping 到通过 DHCP 动态获取分配给 W5500 的 IP 地址，可以看到整个过程是如此的简单。

5.4 DNS

DNS 是域名服务器的简称，用于域名解析。DNS 的出现就是为了用户在访问网页时，不用再输入网站的 IP 地址，而是用一串便于记忆的字母就可以访问。

首先来看一下 DNS 解析的过程的原理示意图，如图 5-4-1：

图 5-4-1 DNS 域名解析原理示意图

通过 W5500 来实现 DNS 域名解析也是非常方便的。本节将解析 www.baidu.com 对应的 IP 地址。客户端初始化完毕之后，如果 DNS 运行标志位是 1 或者 DNS 发送次数超过 3 次就直接返回。如果不是上面的情况，并且 DNS Server 的 IP 不是 0.0.0.0 时，进入 switch 函数。当处于 DNS 域名解析成功状态时，DNS_OK 置 1，发送请求报文次数置 0，并且把得到的域名对应的 IP 地址 copy 到 ConfigMsg.rip，然后我们把解析到的 www.baidu.com 的 IP 地址通过串口打印出来。当处于 DNS 解析域名失败时，标志位置 0，请求报文次数加 1，然后跳出循环。如果以上情况都不是，打印出无效的 DNS Server 地址。以上是简单介绍了通过 W5500 解析百度 IP 地址的过程。过程其实是很简单的，就是通过不断读取寄存器状态来判断 W5500 的状态，最终成功解析到 IP 地址。

具体测试步骤如下：

1，DNS 例程采用默认的 IP 信息，所以在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DEFINE。

2，对代码进行编译，之后将程序烧录到野火开发板。

3，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输出结果可以得到图 5-4-1 设置信息。按照此方法，把程序中“www.baidu.com”域名换成其它的域名，同样可以解析成功。

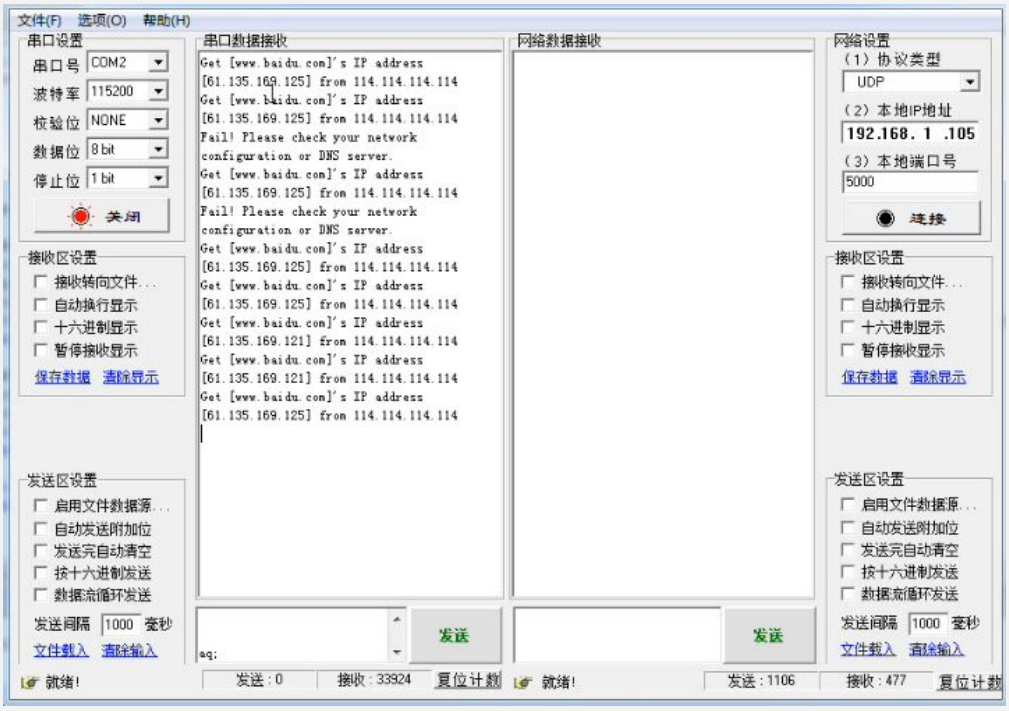


图 5-4-1 DNS 解析信息

5.5 SMTP

SMTP 即简单邮件传输协议，它是一组用于由源地址到目的地址传送邮件的规则，由它来控制信件的中转方式。实际发送邮件的过程，可以如图 5-5-1 所示：

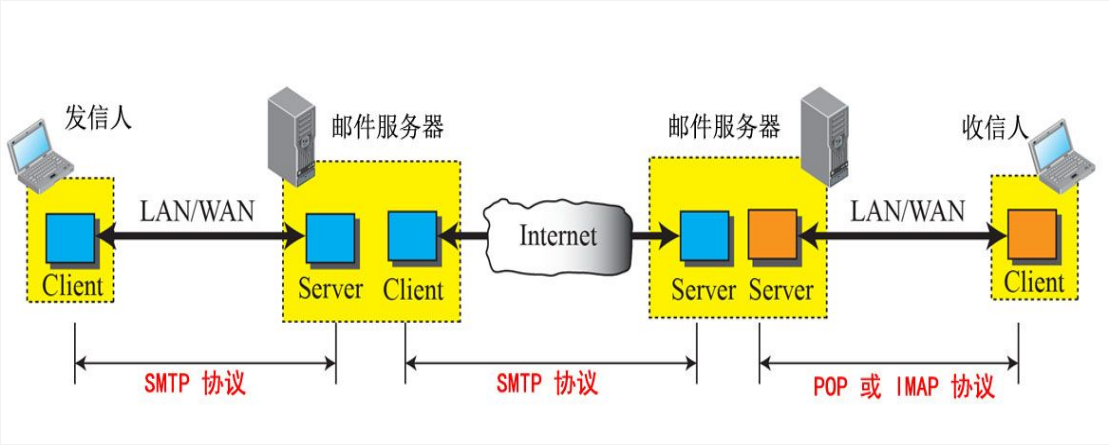


图 5-5-1 邮件发送过程示意图

我们可以看到 SMTP 协议是发送过程中所使用的协议，我们这次给大家模拟演示的仅是图中第一部分，发信人向邮件服务器发送请求的过程。

SMTP 在 TCP 协议 25 号端口监听连续请求。SMTP 连接和发送过程：

1，建立 TCP 连接。

2，客户端发送 HELO 命令以标识发件人自己的身份，然后客户端发送 MAIL 命令；服务器端正希望以 OK 作为响应，表明准备接收。

3，客户端发送 RCPT 命令，以标识该电子邮件的计划接收人，可以有多个 RCPT 行；服务器端则表示是否愿意为收件人接收邮件。

4，协商结束，发送邮件，用命令 DATA 发送。

5，以“.”号表示结束并将内容一起发出去，结束此次发送，用 QUIT 命令退出。

由于资源受限，在没有操作系统的支持下，通过单片机发送邮件与传统的电脑操作将有很大的不同。这里用 W5500与126邮箱通信为例来具体分析邮件的发送过程。在本示例代码中，发件人邮箱名为：wiznet2013@126.com，邮箱密码为：hello123。收件人邮箱地址为：3196855541@qq.com，邮件内容为：Hello!WIZnet!。如果想用别的邮箱做测试的话，请修改代码中收件人和发件人的邮箱名和密码。

具体过程很简单，先解析 126 邮箱的服务器域名 smtp.126.com，成功以后就执行邮件发送函数，邮件发送成功以后就跳出循环或者等待。让程序进入了一个死循环，这样程序将不再跳到主循环，避免重复发送相同的邮件，这样使得在 W5500 的运行模式下，按一下 Reset 键或者上电一次，只发送一封邮件。

具体测试步骤如下：

1，SMTP 例程采用默认的 IP 信息，所以在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DEFINE。

2，对代码进行编译，之后将程序烧录到野火开发板。

3，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输出结果可以得到图 5-5-2 设置信息。

4，首先解析到 smtp.126.com 邮件服务器的 IP，然后发送邮件，如果发送成功，会得到 mail send OK 的信息。

5，登陆串口提示的账号，查看邮件的收件箱可以找到刚才发送的邮件。

图 5-5-2 邮件成功发送打印信息

5.6 ICMP

ICMP 是 Internet 控制报文协议，用于在 IP 主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。我们在网络中经常会使用到 ICMP 协议，比如我们经常使用的用于检查网络通不通的 Ping 命令（Linux 和 Windows 中均有），这个“Ping”的过程实际上就是 ICMP 协议工作的过程。Ping 命令利用 ICMP 回射请求报文和回射应答报文来测试目标系统是否可达。

执行 ping 后，首先向目标服务器发出回送请求报文。计算机送出的回送请求到达目标服务器后，服务器回答这一请求，向送信方发送回送请求。这个 ICMP 回送回答报文在 IP 层来看，与被送来的回送请求报文基本上一样。不同的只是，源和目标 IP 地址字段被交换了，类型字段里填入了表示回送回答的 0，这两点。也就是，从送信方来看，自己送出的 ICMP 报文从目标服务器那里象鹦鹉学舌那样原样返回了。

具体的 ping 请求函数和 ping 回复函数就不在一一讲解，具体实现过程请参考例程。

接下来介绍一下具体的测试过程，测试步骤如下：

1，ping 例程采用默认的 IP 信息，所以在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DEFINE。

2，具体要 ping 的 IP 地址，在 w5500_conf.c 中通过定义 remote_ip 变量来表示，大家可以以根据自己的要求修改

3，对代码进行编译，之后将程序烧录到野火开发板。

4，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输出结果可以得到图 5-6-1 设置信息。

从打印信息可以看到，发送一次 ping 请求，之后得到一次 ping 回复，如果请求和回复的次数保持相同，则说明 ping 一直正常。

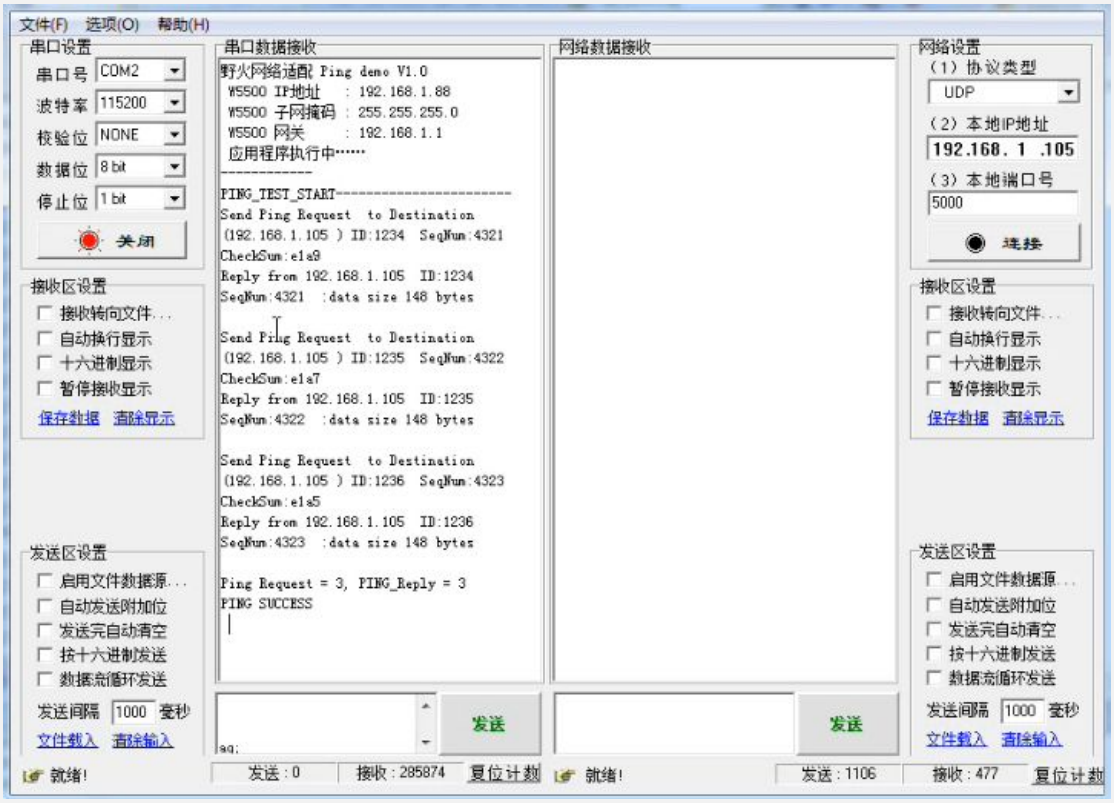


图 5-6-1 ping 回复信息

5.7 HTTP Server

HTTP 服务器也就是平时我们所说网页服务器，其实这种嵌入式设备内嵌的网页服务器在我们日常生活中十分常见，例如我们天天使用的无线路由器。我们在配置无线路由器时，都会使用浏览器打开其配置页面进行配置。这里我们使用的就是无线路由器里面内嵌的网页服务器功能。

图 5-7-1 HTTP Server 实现原理图

本节我们就用 W5500 建立自己的网页服务器，你只在浏览器地址栏里键入 W5500 的 IP 地址，就成功在你的网页上显示出来了。HTTP Server 实现原理图见图 5-7-1。

首先在 main 主函数中，我们完成对 W5500 的初始化，同时调用 init_http_server()函数实现对 HTTP Server 的初始化。在这个初始化函数中，配置 W5500 的 IP 地址，MAC 地址等基本网络参数，然后在主循环中调用 do_http()函数实现 HTTP 服务器。单片机作为 HTTP 服务器的具体工作过程在 void proc_http(SOCKET s, uint8 * buf)中有详细过程，解析 http 请求报文并发送 http 响应报文。请参考例程注释了解详细过程。

具体测试步骤如下：

1，HTTP 服务器采用默认的 IP 信息，所以在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DEFINE。

2，对代码进行编译，之后将程序烧录到野火开发板。

3，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输

出结果可以得到图 5-7-2 设置信息。

图 5-7-2 串口打印信息

4，打开浏览器，键入 192.168.1.89，如图 5-7-3 所示，在浏览器里面成功地显示出来了。

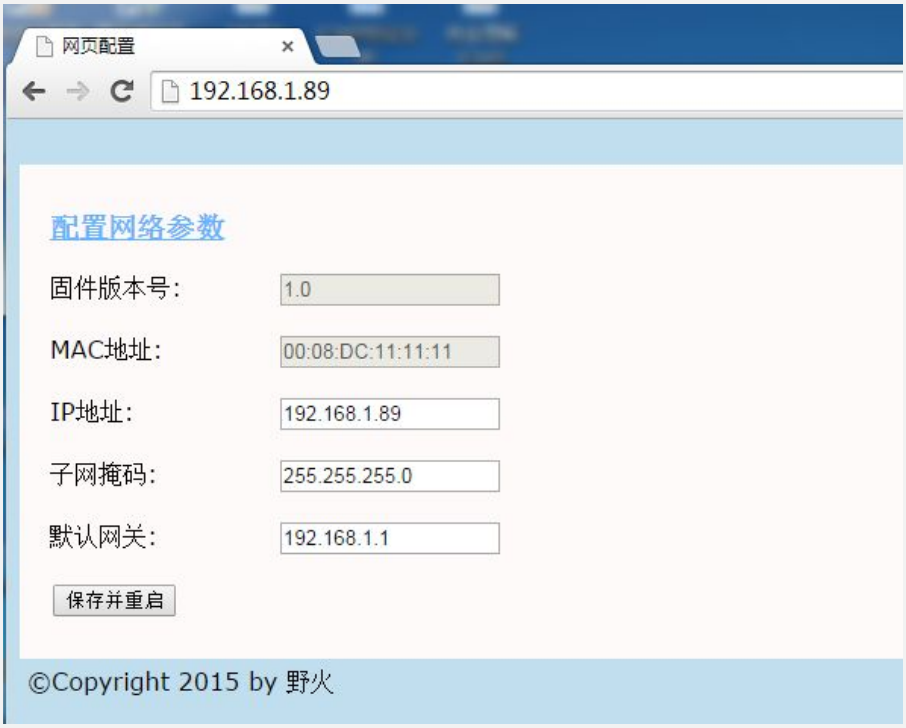


图 5-7-3 浏览器显示效果

5.8 HTTP Client

本节我们将在 W5500 上面实现一个 HTTP Client 程序去连接网络服务器 Yeelink 平台。野火开发板预留有温湿度传感器 DHT11 的接口，因此我们将通过 W5500 将采集到温度和湿度信息上传 Yeelink 平台，并且在 Yeelink 平台实时观察温湿度变化情况。图 5-8-1 为 W5500 通过 Yeelink 平台实现 HTTP Client 模型。

图 5-8-1 W5500 通过 Yeelink 平台实现 HTTP Client 模型

Yeelink 是一个免费的云平台，旨在利用无线网络、开源硬件和软件，智能手机和 App 共同打造一个家庭智能中心打开。打开 Yeelink 平台服务器网页，点击“快速开始”，我们可以注册一个账号，登陆后，可以添加自己的设备和传感器。它将复杂的传感器以极简的方式组到同一个网络内，可满足智能家居的各种需求。通过 Yeelink 提供的数据接口，用户可以把自己的传感器通过互联网接入 Yeelink 云平台，从而实现随时随地获取传感器数据，为一些智能家居设备接入互联网提供了云平台支持。（http://www.yeelink.net）

具体的设备添加过程，Yeelink 平台有详细的介绍，在此就不多解释。在申请账号时要注意生成的 API KEY，添加设备时会生成一个 URL 请求，里面提供了设备 ID device/xxxx，传感器 ID sensor/xxxx。这几个信息与程序开头建立的请求报文中信息是一一对应的，因此应该把程序中这些信息修改为我们自己申请账号和添加设备是得到的信息。同大多数开源平台一样，Yeelink 提供的 API 也是基于 HTTP 协议提交和接收数据。有关于 API 文档的详细介绍请参考 Yeelink 网站。图 5-8-2 是添加好的设备界面信息。

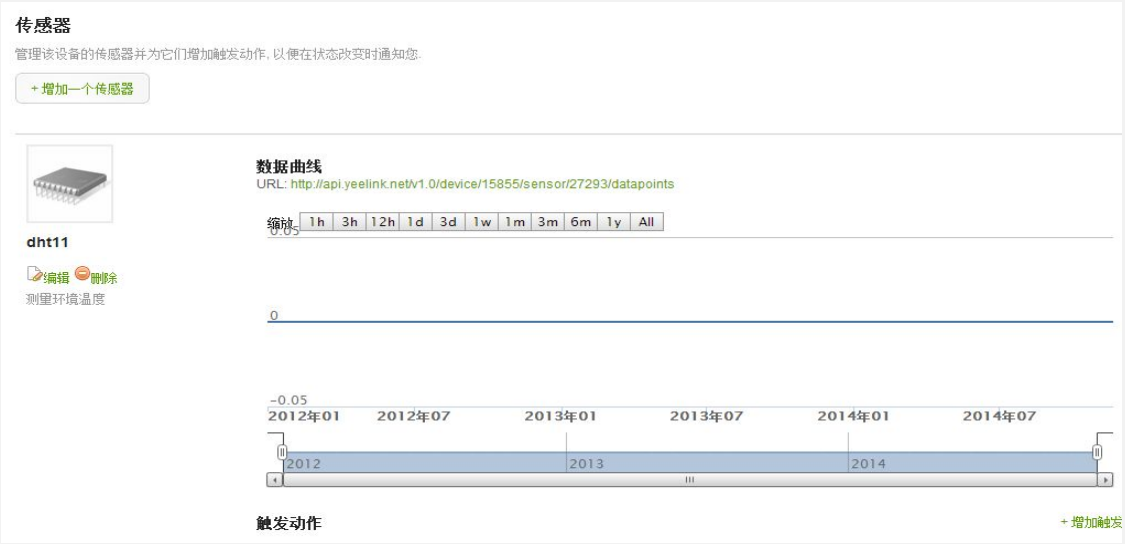


图 5-8-2 设备添加界面信息

接下来简单分析一下程序，main.c 主文件实现 W5500 上传数据的主流程，bsp_dht11.c 文件实现对温湿度数据的采集，http_client 文件实现 W5500 与服务器的的连接以及数据上传。初始化函数没有具体的变化，在此就不多讲解。

为了使程序能方便地组建 Yeelink 所要求的 JSON 数据格式，我们把采集到数据直接替代这个字符串中的 xx 即可。device/ID/sensor/ID 这两处 ID 已经在 Yeelink 网站上注册得到，一个是设备 ID，一个是设备上的传感器的 ID，同时，一个设备可以有若干个传感器。U-ApiKey 则是你的身份识别码，只有正确提交 U-ApiKey 才可以在 Yeelink 上面更新数据。Content-Length:后面的 12，指的就是 {"value":xx} 的长度，更具体的格式可以翻阅 Yeelink 的 API 文档。需要注意的是，在 C 语言中，双引号 “”需要使用转义字符。

接下来介绍一下具体的测试过程，测试步骤如下：

1，HTTP 客户端例程采用默认的 IP 信息，所以在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DEFINE。

2，对代码进行编译，之后将程序烧录到野火开发板。

3，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输出结果可以得到图 5-8-3 打印信息。

图 5-8-3 串口打印信息

4，我们根据串口提示的信息登录 Yeelink 账号，打开我的设备→管理设备，就可以看到具体的温度和湿度信息，我们可以看到如图 5-8-4 的实时温湿度信息。

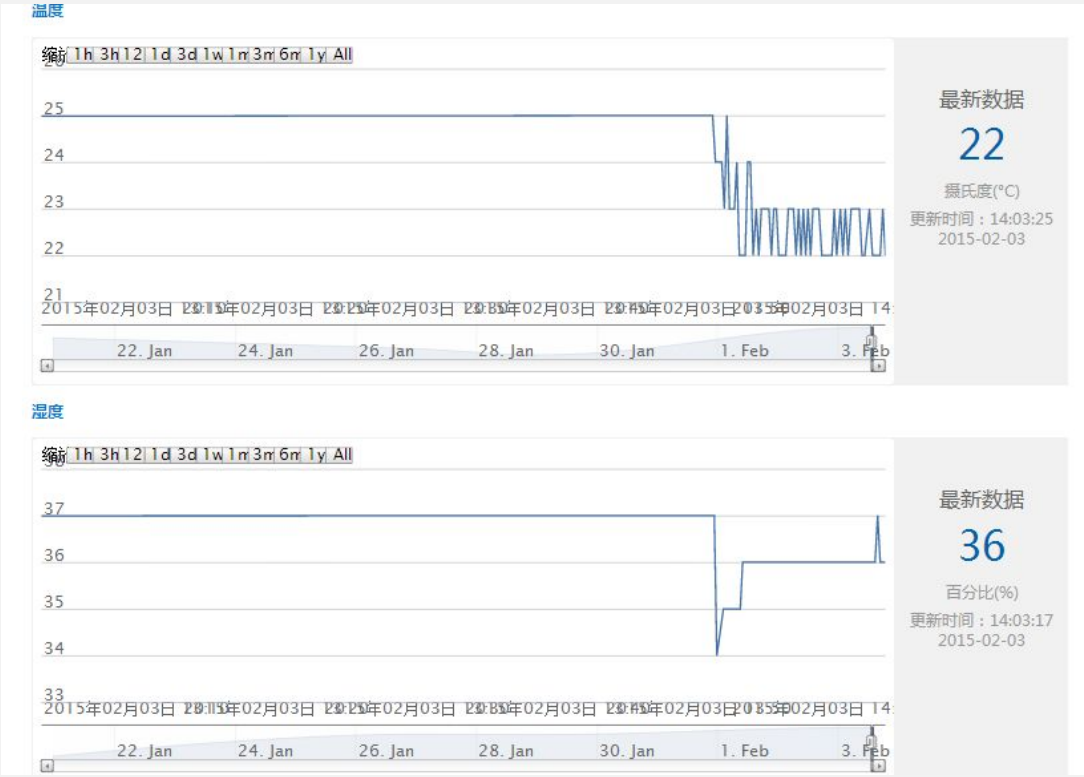


图 5-8-4 实时的温湿度读取信息

5.9 NetBIOS

NetBIOS 协议是一种在局域网上的程序可以使用的应用程序编程接口（API），为程序提供了请求低级服务的统一的命令集，作用是为了给局域网提供网络以及其他特殊功能，几乎所有的局域网都是在 NetBIOS 协议的基础上工作的。NetBIOS 协议，简单来说就是通过访问设备名称就可以实现对 IP 地址的访问。NetBIOS 实现过程示意图如 5-9-1 所示。

图 5-9-1 NetBIOS 实现过程示意图

在 Windows 操作系统中，默认情况下在安装 TCP/IP 协议后会自动安装 NetBIOS 协议。NetBIOS 的报文类型较多、结构复杂，在不同的网络环境和不同的用途中会使用不同的报文，可用端口进行区分， NetBIOS 数据报报文使用 UDP 138 端口，NetBIOS 会话报文使用 TCP 139 端口。

本节主要通过 NetBIOS 协议实现把 IP 地址解析为对应的一个名字，就比如 180.97.33.107 是百度服务器的 IP 地址，对应的名字是“www.baidu.com”。这样我们就不用记住很多的 IP 地址，通过一个简单的名字就可搜索到相应的信息。本例程首先通过 DHCP 获取一个动态 IP 地址分配给 W5500，然后通过 NetBIOS 协议来解析 W5500 这个名字和 IP 地址对应，然后添加 HTTP 服务器功能，在网页中输入 W5500 就可以进入相应的网页信息，在 windows 下 ping W5500 也可以得到对应的 IP 地址信息。

程序的 Main 很简单，主要是网络初始化和芯片的初始化。由于本程序通过

DHCP 获取动态 IP 以后，再通过 NetBIOS 实现名字解析的，所以要在 w5500_conf.c 文件下定义 ip_from 为 IP_FROM_DHCP。DHCP 程序在 5.3 节已经详细介绍，在此不再多解释。NetBIOS 的处理主要在 NetBIOS.c 中。在 do_netbios 函数中对 UDP 广播查询做了解析，并且回应。这就完成了 b 节点的 NetBIOS 名称服务。在 do_netbios 中一个 Socket 在指定的端口一直侦听，如果收到数据包就判断格式，看是不是 NetBIOS 查询包。如果是，调用 netbios_name_decoding 函数解包。之后按照 NetBIOS 格式组包回复。

具体实现过程请参考例程。接下来介绍一下具体的测试过程，测试步骤如下：

1，本例程通过 DHCP 动态获取 IP 信息来配置 W5500 的 IP 信息，所以在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DHCP。

2，对代码进行编译，之后将程序烧录到野火开发板。

3，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输出结果可以得到图 5-9-2 设置信息。

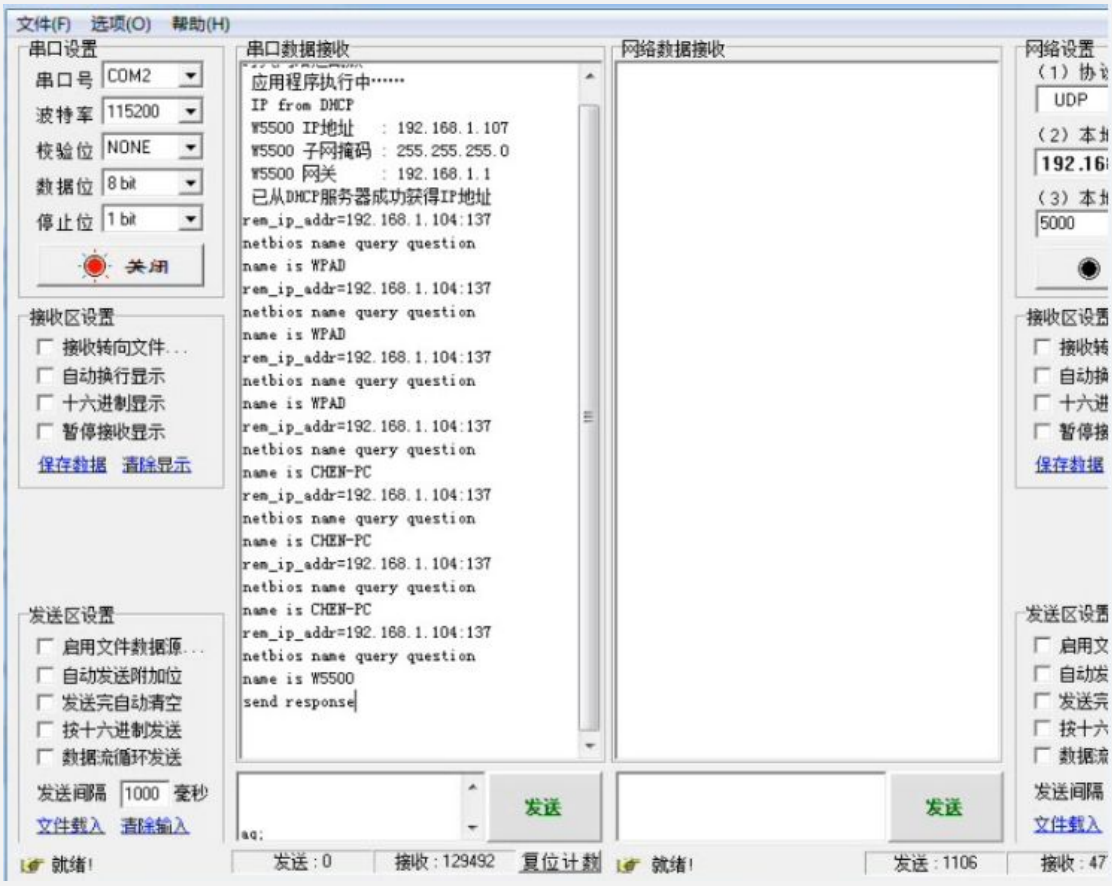


图 5-9-2 串口打印信息

4，在 Windows 下的具体操作是，开始→运行→ 键入 cmd，输入 ping W5500 →回车。看到回复信息如图 5-9-3 所示。W5500 对应的 IP 地址就是 192.168.1.109，说明 NetBIOS 名称解析成功。

图 5-9-3 ping W5500 回复信息

5，在 IE 浏览器中输入 W5500 并点击回车，看到网页信息如图 5-9-4 所示。W5500 对应的信息也在网页中正确显示，说明 NetBIOS 名称解析成功。

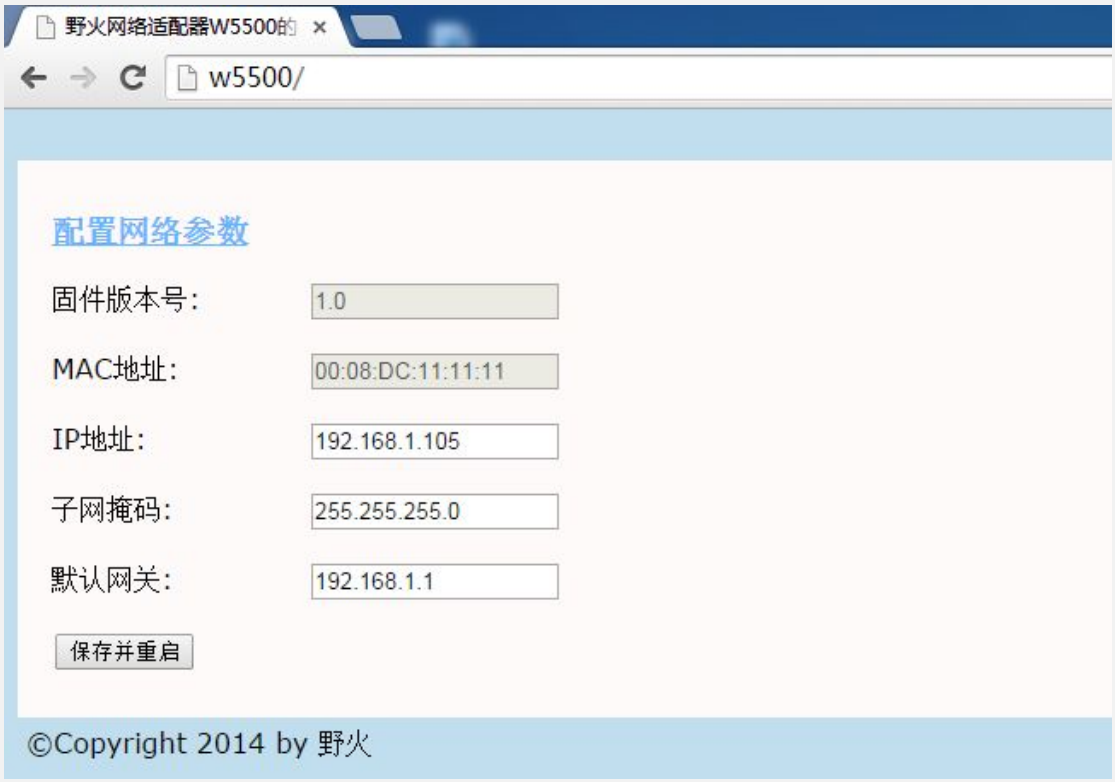


图 5-9-4 网页登陆信息

5.10 NTP

NTP 是网络时间协议，是用来使设备时间同步化的一种协议，在一些应用场合里，时间同步是十分重要的，特别是随着设备运行时间云长，时间误差就越来越大，因为设备中晶振自身会产生误差。那么 NTP 协议就可以解决这个问题。首先，我们来看 NTP 校时过程中的原理示意图，如图 5-10-1 所示。其中在我们的演示中，W5500 为图中 LS_A 端，NTP 服务器（国家授时中心）为 LS_B 端。详细校时过程的讲解，请参考相关教学视频。

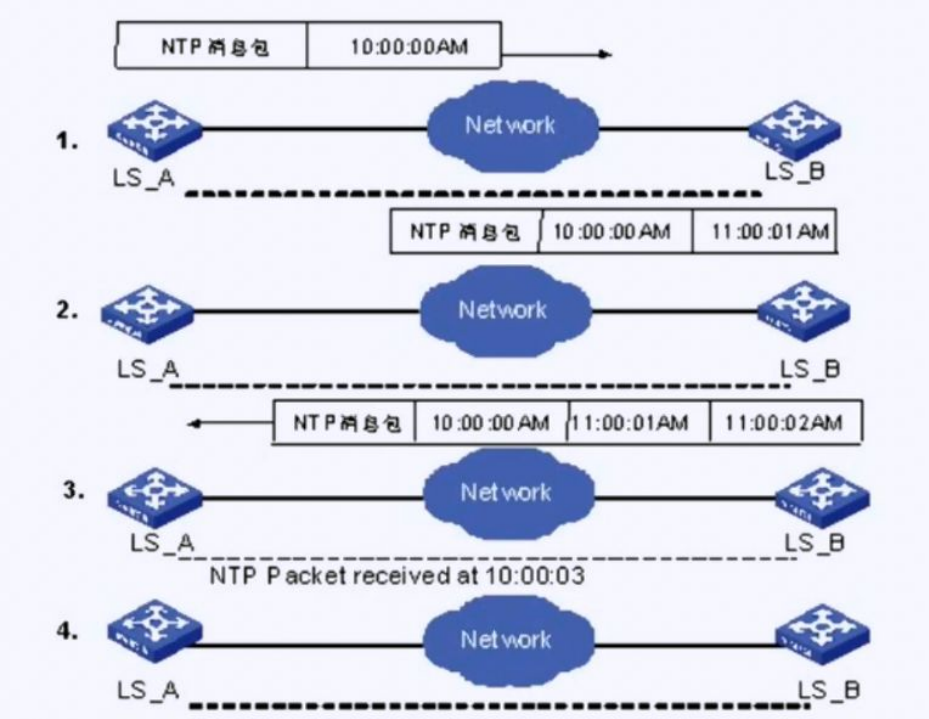


图 5-10-1 NTP 校时过程原理图

下面我们就结合 W5500 介绍一下如何使用 W5500 实现 NTP 协议，即从一个远程 NTP 服务器获取标准网络时间，通过换算成北京时间（东八区），然后通过串口把时间打印出来。相信通过本次讲解，一定会使你对 NTP 有更清晰的认识。

主函数中初始化单片机以及进行网络配置的步骤与前面章节相同，这里就不再赘述。主函数中重要的是调用 ntpclient_init()和 do_ntp_client()两个函数。前者初始化 NTP 报文，后者完成与 NTP 服务器的交互过程。

由于本程序只是实现从服务器获取时间，并未涉及时钟同步的问题，所以后

面的字段都不需要用到，全部初始化为 0，为了简化程序，NTP_Message 中也仅仅包含 flag 中的内容。NTP 服务器的 IP 地址在全局变量 NTP_Server_IP 中定义，NTP 服务器的默认端口号是 123。

具体测试步骤如下：

1，本例程通过 DHCP 动态获取 IP 信息来配置 W5500 的 IP 信息，所以在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DHCP。

2，对代码进行编译，之后将程序烧录到野火开发板。

3，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输出结果可以得到图 5-10-2 设置信息。

4，可以看到同步 NTP 服务器时间为 2015-02-03 16:24:57，在观察自己电脑的时间，基本没有误差。

图 5-10-2 串口打印信息

5.11 网页远程配置设备

本节将实现用浏览器直接控制和访问单片机，来配置 W5500 的 IP 地址，子网掩码，默认网关等信息。这里使用 EEPROM 来存储配置信息，野火开发板中有 2k 字节的 EEPROM 来保存配置信息。和 Flash 类似，EEPROM 也是一种掉电后数据不丢失的存储芯片。但与 Flash 不同的是，EEPROM 是一个外部芯片，容量一般比较小，专门用来给用户存储掉电需要保存的配置信息。使用者可以自行对比这两种方法的优劣，根据实际需求在自己的程序中选择配置信息的存储介质。本程序中通过 cgi_ipconfig()函数来将配置信息写入 EEPROM 中。

源程序分三部分来解读。main.c 主文件实现程序运行的主流程，httputil.c 文件用于接收 HTTP 请求报文和发送 http 响应报文，httpd.c 用于实现对请求报文的解析，需要调用的其它函数在其他应用程序中声明。

主程序的初始化过程基本一样，唯一不同的是把 W5500_conf.c 文件下的 use_eeprom 赋值为 1。因为本函数在配置网络信息的过程中要从 EEPROM 中读取信息，将复制 EEPROM 配置信息到配置结构体。While 主循环运行 HTTP Server 主函数，如果标志位是 1 的话，重新启动程序。不难看出，该程序具有掉电记忆功能，因为我们把配置信息都写进了 EEPROM 中。

void do_http( )函数和 void proc_http( )函数在 5.7 章节中已经详细叙述过了， W5500 程序中使用一个比较大字符数组来保存网页的程序，当浏览器访问 W5500 的 IP 地址时，W5500 将网页程序发送给浏览器显示。

代码清单 5-1 是 void proc_http(SOCKET s, uint8 * buf)函数中状态机的一种模式。第 1 行，当 HTTP Request 方法是 POST 时，第 2 行，获取 HTTP Request 的文件名，第 3 行，如果 req_name 是 config.cgi 时，第 5 行把 http_request 中的 IP 地址等信息写进闪存中，替换以前的默认设置，第 6、7 行生成 http_reponse 的文本部分，然后以 HTTP 响应的报文格式存入 http_response 数组中。第 8 行发送 http_response，第 9 行，断开 socket 连接，第 10 行把标志位置 1。通过这段程序就实现了通过浏览器修改单片机 IP 地址的功能。

代码清单 5-1　SOCKET 状态机函数

```
case METHOD_POST:
      mid(http_request->URI, "/", " ", req_name);
      if(strcmp(req_name,"config.cgi")==0)
     {
              cgi_ipconfig(http_request);
              make_cgi_response(5,(int8*)ConfigMsg.lip,tx_buf);
              sprintf((char*)http_response,"HTTP/1.1200OK\r\nContent-
              Type: text/html\r\nContent-Length: %d\r\n\r\n%s",
              strlen(tx_buf), tx_buf);
       send(s, (u_char *)http_response, strlen((char *) http_response));
      disconnect(s);
       reboot_flag=1;
       return;
       }
break;
```

程序还是 5.7 节用到的 http_server 程序，编译无误后下载到野火开发板，打开串口工具并复位开发板，串口打印信息如图 5-11-1 所示，在 WEB 输入串口提示的 IP 地址，回车后出现 Web 界面。

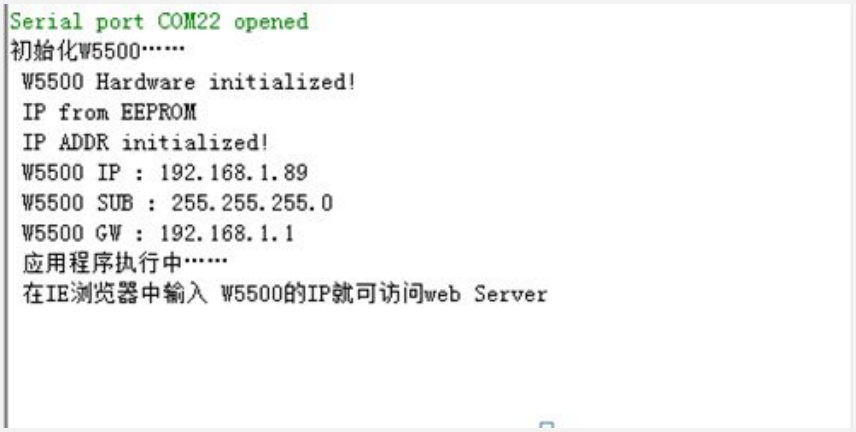


图 5-11-1 串口打印初始化信息

如图 5-11-2 可见，Device Settings 中依次列出 W5500 的硬件版本号、MAC 地址、IP 地址、子网掩码以及默认网关。

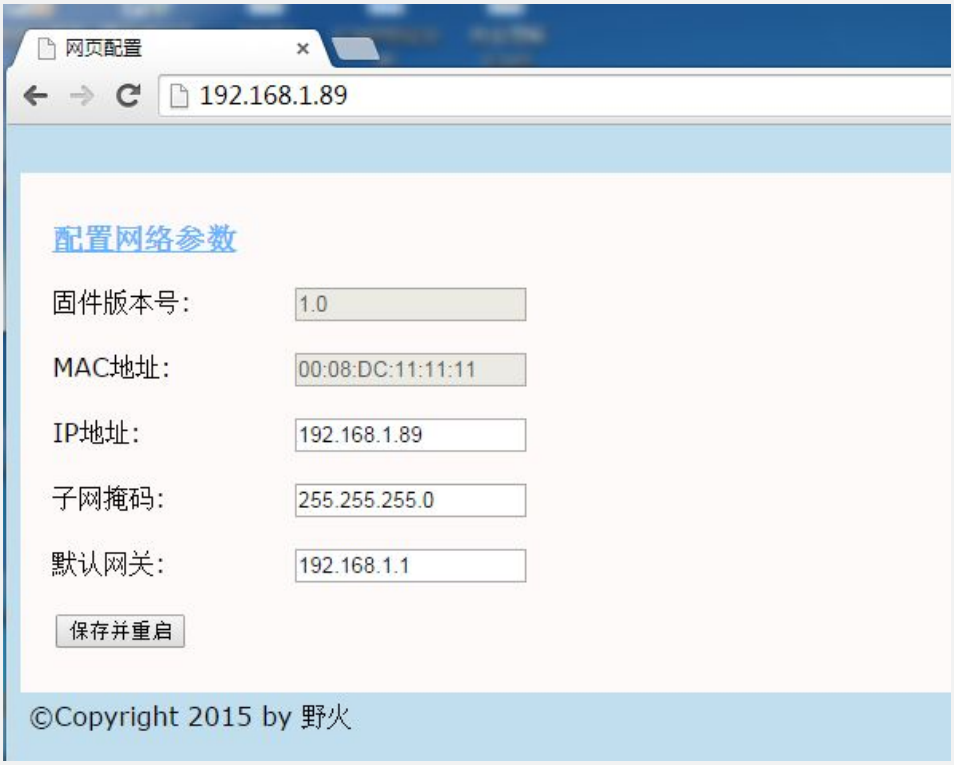


图 5-11-2 浏览器显示效果

用户可以配置其他 IP 地址，子网掩码及网关，并点击“保存并重启”，设置并重启生效。比如修改 IP 地址为 192.168.1.100，网页刷新后的信息如图 5-11-3 所示，IP 地址确实变了。

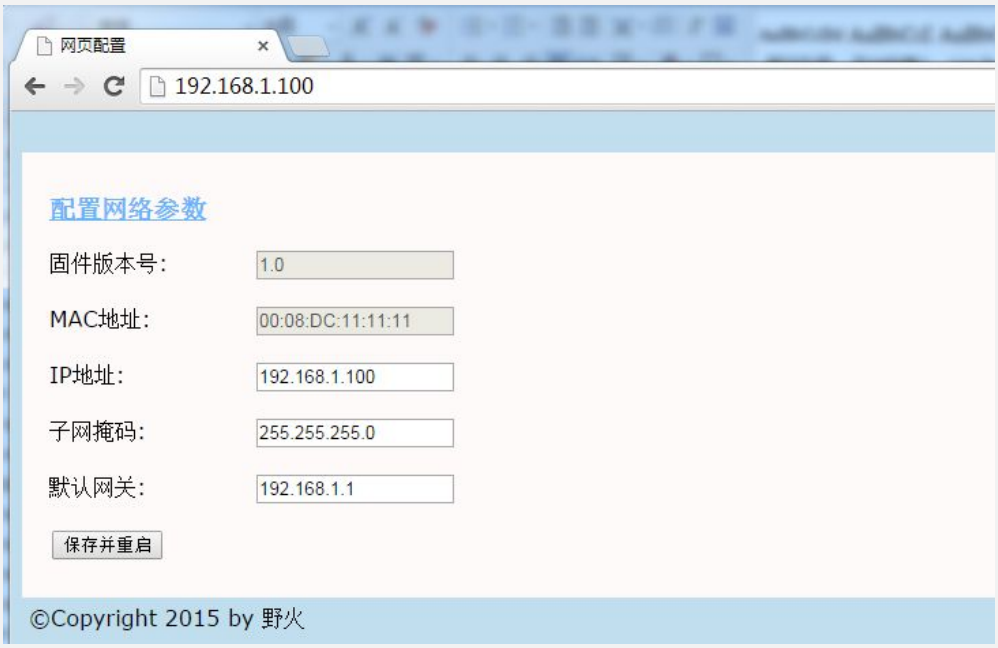


图 5-11-3 更新后的网页显示信息

同时可以观察串口的打印信息也发生了变化，如图 5-11-4 所示。串口的打印信息是直接从 W5500 的内部寄存器读取的，说明信息配置成功。

```
Serial port COM22 opened
初始化W5500……
 W5500 Hardware initialized!
 IP from EEPROM
 IP ADDR initialized!
 W5500 IP : 192.168.1.89
 W5500 SUB : 255.255.255.0
 W5500 GW : 192.168.1.1
 应用程序执行中……
 在IE浏览器中输入 W5500的IP就可访问web Server
 系统重启后，请重新选择应贸绦?
 初始化W5500……
 W5500 Hardware initialized!
 IP from EEPROM
 IP ADDR initialized!
 W5500 IP : 192.168.1.100
 W5500 SUB : 255.255.255.0
 W5500 GW : 192.168.1.1
 应用程序执行中……
 在IE浏览器中输入 W5500的IP就可访问web Server
```

图 5-11-4 串口更新信息

5.12 发微博

单片机加上 W5500 就可以发微博？是的，按照下面的介绍和操作你将会让你的单片机实现这一功能。

本程序的设计思想是让 W5500 通过一个简单的代理服务器与新浪微博开放平台的接口通信完成发微博的操作。为什么需要使用代理服务器而不是直接与新浪微博服务器通信呢？因为刚开始时，新浪微博有一个基于 Basic Auth 的用户认证开放接口，但基于安全方面的考虑关闭了这个接口。对现有第三方应用，只能使用 OAuth 2.0 这个接口来开发应用。然而基于 SSL 的 OAuth 2.0 虽然安全性得到了提高，但是复杂性也有所增加，通过单片机来实现变得更加困难。因此，本程序采用增加代理服务器的方式，由代理服务器来做 OAuth 2.0 授权和认证，单片机只需把自己的微博账号和密码，以及要发的微博内容提交给代理服务器即可。

接下来介绍一下具体的测试过程，测试步骤如下：

1，本例程把默认 IP 信息来配置 W5500 的 IP 信息，所以在 w5500_conf.c 文件中设置 ip_from 为 IP_FROM_DEFINE。

2，对代码进行编译，之后将程序烧录到野火开发板。

3，连接好网线，USB 串口线。打开串口调试工具，复位野火开发板，从输出结果可以得到图 5-12-1 设置信息。

```
Serial port COM22 opened
初始化W5500……
 W5500 Hardware initialized!
 IP ADDR initialized!
 W5500 IP : 192.168.1.88
 W5500 SUB : 255.255.255.0
 W5500 GW : 192.168.1.1
 我的微博ID为:3196855541@qq.com
 我的微博PWD为:wildfire123
 看串口调试信息：rev255: ok 说明上传成功，登陆以上微博账号就可看到发送
微博信息
 应用程序执行中……
Connect with Weibo server.
Connected with Weibo server.
rev255: ok
```

图 5-12-1 串口打印信息

4，如图 5-13-1 所示，看串口调试信息：255: ok。说明上传成功。

5，根据串口提示微博账号登陆登陆到微博看看，刚才写进程序里面的那句话果然出现在了微博上面，如图 5-13-2 所示。可以看到下方标注：“来自 iWIZnet”，正是我们与新浪微博平台对接的网站应用，点击之后可链接到指定网站。

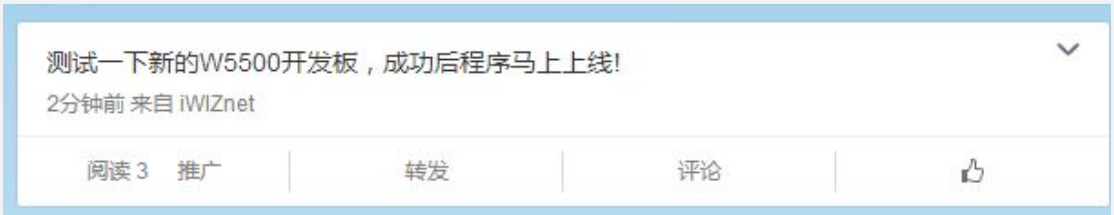


图 5-12-2 登录微博信息

5.13 小结

至此，我们应用野火官方开发板 ISO STM32 搭配野火 W5500 网络适配板实现的上层应用例程介绍完毕，而这些应用基本涵盖了单片机接入以太网后所需要实现的所有功能。当然，这只是结合 STM32 做出的一些应用代码和演示，如果主控是其他型号的，这套代码也可以非常方便地移植到新的平台上，只需要简单的修改一些参数即可。

通过这个简单的教程我们可以看到，用硬件协议栈方案实现单片机接入以太网的整个操作过程如此简单，我们甚至不需要研究所谓的 TCP/IP 协议族等底层结构就能实现想要的功能。

在网络无处不在的今天，随着物联网、智能家居等概念的推广和普及，金融、交通、电力、日常生活等行业的智能化控制将会产生大量的需求，简单、安全、稳定的网络接入方案将使单片机焕发出新的强大生命力，推动智能化时代的到来。