ISSN 2616-5775 Vol. 9, Issue 4: 96-102, DOI: 10.25236/AJCIS.2026.090412

Design and Implementation of Etherne Communication System Based on STM32F407

Cuiping Zhang[1,2], Jiazu Xie[1,2,*]

[1]Tianjin Key Laboratory of Wireless Mobile Communications and Power Transmission, Tianjin Normal University, Tianjin, 300387, China
[2]College of Electronic and Communication Engineering, Tianjin Normal University, Tianjin, 300387, China
[*]Corresponding Author: veryaz@tjnu.edu.cn

Abstract: *With the rapid development of the Internet of Things (IoT) and industrial control networks, the demand for embedded devices to access Ethernet is increasing day by day. This article designs and implements an Ethernet communication system based on STMicroelectronics' high-performance ARM Cortex-M4 core microcontroller STM32F407ZGT6. The system adopts a modular design concept, with STM32F407 as the core hardware and LAN8720A physical layer transceiver (PHY) chip integrated to build an Ethernet interface; The lightweight TCP/IP protocol stack (LWIP) was ported to the software to implement network protocol processing, and a Socket based application layer communication program was developed. This paper provides a detailed explanation of hardware circuit design, software architecture construction, the porting and configuration of the LWIP protocol stack, application layer communication protocol design, and the system testing plan. The test results show that the system can stably implement TCP and UDP communication, meeting the basic needs of embedded devices accessing local area networks for data exchange. It has the advantages of low cost, stable performance, and easy development, laying the foundation for more complex network applications such as web servers and remote control in the future.*

Keywords: *STM32F407; Ethernet communication; LWIP; TCP/IP; Embedded systems; LAN8720A; Socket*

1. Introduction

Driven by the Internet of Things (IoT) and Industry 4.0, embedded devices need to have efficient network communication capabilities to achieve data interconnection. STM32F407, as a high-performance microcontroller launched by STMicroelectronics, has rich peripheral interfaces (such as SPI, USART, GPIO, etc.) and powerful computing capabilities (Cortex-M4 core, with a maximum frequency of 168MHz), making it suitable as the core controller for embedded network communication systems [1]. Traditional embedded network solutions often rely on dedicated network chips or expensive processors, while solutions based on STM32F407 can achieve stable Ethernet communication at low cost and have broad application prospects in fields such as intelligent sensors, remote monitoring, and industrial control[2].

This article aims to design an Ethernet communication system based on STM32F407, build hardware circuits, and achieve interface connection between STM32F407 and Ethernet controllers; Transplant LwIP lightweight TCP/IP protocol stack to implement protocol functions such as IP, ARP, ICMP, TCP, UDP, etc; Develop application layer programs to verify the communication stability and data transmission performance of the system. With the popularization of industrial automation and smart home systems, embedded devices have put forward higher requirements for real-time, reliability, and compatibility of network communication. This system not only meets basic network access requirements, but also provides hardware and software foundation for advanced functions such as remote configuration, OTA upgrade, and multi device networking for subsequent expansion. Through modular design and protocol stack optimization, the system ensures performance while also considering operational efficiency in resource constrained environments, making it highly practical in engineering applications.

ISSN 2616-5775 Vol. 9, Issue 4: 96-102, DOI: 10.25236/AJCIS.2026.090412

2. System design scheme

2.1 Hardware Architecture

The hardware architecture is shown in Figure 1, with the core processor being the STM32F407ZGT6 microcontroller, with a clock frequency of 168MHz and Ethernet MAC. The PHY chip uses LAN8720A and the network interface uses HR911105A, with an internal transformer. The clock circuit uses a 25MHz crystal oscillator to provide the PHY clock, and the STM32 internal PLL provides the MAC clock. Other auxiliary circuits include power supply circuit, reset circuit, debugging interface (SWD/JTAG), and indicator LED.

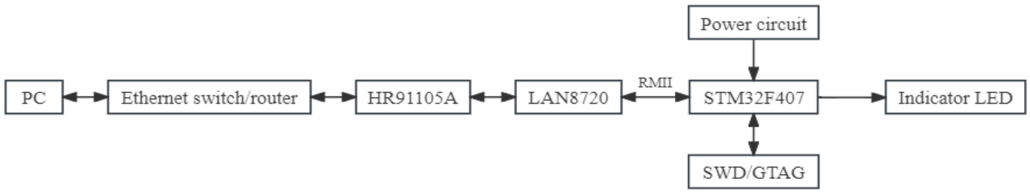

Fig. 1 Hardware Architecture

In hardware design, STM32F407 is connected to LAN8720A through RMII interface to achieve high-speed data exchange between MAC and PHY [3,4]. The system is powered by 3.3V and provides stable voltage through the LD1117 voltage regulator chip, ensuring the stable operation of the PHY chip and microcontroller in complex electromagnetic environments. In addition, the system also features status indicator LEDs for visually displaying network connection and data transmission status, facilitating debugging and status monitoring. The hardware layout fully considers signal integrity and power integrity, adopting a four layer PCB design to effectively reduce high-frequency signal interference, enhance system anti-interference ability and communication reliability. The specific peripheral characteristics and register configuration of STM32F407 can be found in its data manual [5].

2.2 Software Architecture

The underlying driver layer includes the STM32 HAL library (or standard peripheral library) to provide MAC drivers GPIO, Basic support such as clock and interrupt [6]. The protocol stack layer uses LWIP lightweight TCP/IP protocol stack, which includes core protocols such as ARP, IP, ICMP, UDP, TCP, DHCP, DNS, etc[2]. The application layer is a communication program developed based on Socket API, including TCP and UDP, which enables data exchange with the upper computer.

In software architecture design, the idea of layering and modularization is adopted to facilitate subsequent functional expansion and maintenance. The underlying driver layer implements unified management of hardware resources through the HAL library, improving code portability. The protocol stack layer retains only the required protocol modules of the system through customized configuration, reducing memory usage [2]. The application layer abstracts network communication through Socket interface, supports reliable transmission of TCP and real-time broadcasting of UDP, and meets communication needs in different scenarios. The system also integrates a watchdog mechanism and exception handling process to ensure stable operation in complex network environments.

3. Hardware design

3.1 Core Controller

STM32F407 is a 32-bit microcontroller chip based on ARM Cortex-M4 core, with 1024KB FLASH and 192KB SRAM. The main frequency is up to 168MHz, and the computing power meets the processing requirements of the protocol stack; Built in 10/100M Ethernet MAC, SPI, DMA and other peripherals, can efficiently connect to LAN8720.

3.2 Ethernet Interface Circuit Design

The network communication module adopts the LAN8720 chip, which is an integrated Ethernet physical layer (PHY) chip from Microchip. LAN8720 supports communication with Ethernet MAC layer through RMII interface, with built-in 10-BASE-T/100BASE-TX full duplex transmission module, supporting 10Mbps and 100Mbps. It supports IEEE 802.3 standard Ethernet communication and can achieve high-speed and stable data transmission. The LAN8720 also features a built-in adaptive equalizer and clock recovery function, providing high-quality signal processing to ensure the stability and reliability of data transmission. It also supports automatic MDI/MDIX functionality, enabling direct connection to switches or routers and simplifying network cabling and connectivity. The network port adopts HR911105A, which integrates a network transformer internally, as shown in figure 2.

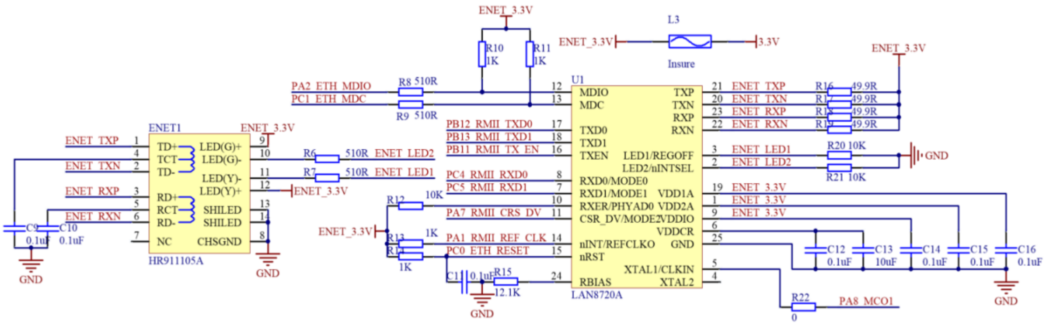

Fig. 2 Schematic diagram of network chip LAN8720

In circuit design, the wiring of RMII interfaces follows the principles of equal length and impedance matching to ensure signal transmission quality [3,4]. The reset and configuration pins of the PHY chip are controlled through the GPIO of STM32, supporting software reset and mode switching. The power supply adopts multiple LDOs to supply analog and digital parts respectively, effectively reducing noise interference. In addition, the system reserves SPI Flash interface for future expansion of network boot or configuration file storage functions, enhancing system flexibility.

4. Software design [8]

4.1 Development Environment

The system development environment adopts Keil MDK5, the software support package is STM32F4 standard peripheral library v1.8.0, and the protocol stack version adopts LwIP (V2.1.2).

4.2 LwIP protocol stack porting and configuration

4.2.1 Protocol Stack Trimming

Get the latest stable version source code (V 2.1.2 version), the source code structure mainly includes:

(1) Src/: Core code of protocol stack (TCP, UDP, IP, ICMP, etc.);

(2) Contrib/: contributed code (including some driver examples and OS adaptation templates);

(3) Doc/: Documentation (Migration Guide, API Description) [1,7]

Tailoring LwIP functionality according to actual needs, retaining core protocols: IPv4 (static IP configuration), ARP (address resolution); ICMP (supports ping command); TCP (Reliable Transport), UDP (connectionless transport).

4.2.2 Interface adaptation

Implement interface function adaptation between LwIP protocol stack and hardware:

(1)Low_1evel_init(): Initialize the network card hardware (reset, PHY configuration, MAC address setting, etc.) and return the initialization result.

ISSN 2616-5775 Vol. 9, Issue 4: 96-102, DOI: 10.25236/AJCIS.2026.090412

```
static err_t low_level_init(struct netif *netif) {
  netif->mtu = 1500;
  netif->flags = NETIF_FLAG_BROADCAST | NETIF_FLAG_ETHARP;
  return ERR_OK;
}
```

(2)Low_1evel_output(): Send LwIP packets (pbuf structure) to the physical link through the network card.

```
static err_t low_level_output(struct netif *netif, struct pbuf *p) {
  struct pbuf *q;
  for (q = p; q != NULL; q = q->next) {
    memcpy(tx_buf + offset, q->payload, q->len);
    offset += q->len;
  }
  ETH_StartTransmit();
  return ERR_OK;
}
```

(3) Low_1evel_input(): Pass the received packet into the LwIP protocol stack

```
static struct pbuf *low_level_input(struct netif *netif) {
  struct pbuf *p = NULL;
  uint16_t len = 0;

  if (ETH_CheckReceivedFrame()) {
    len = ETH_GetReceivedFrameLength();

    p = pbuf_alloc(PBUF_RAW, len, PBUF_POOL);
    if (p != NULL) {

      ETH_ReadReceivedData(p->payload, len);
    }

    ETH_ClearReceivedFrame();
  }
  return p;
}
```

During the protocol stack porting process, the focus was on optimizing the memory management strategy, adopting dynamic memory pooling and customized memory allocation functions to avoid fragmentation issues caused by frequent memory allocation. At the same time, to improve real-time performance, a semaphore mechanism is used in the interrupt service function to notify the LwIP main thread to process data packets, reducing protocol stack processing latency.

4.3 Application layer program design

4.3.1 TCP Server Implementation

Design a TCP based server program, bind ports (such as 8080) and listen for connection requests; After establishing a connection with the client, the loop receives data and returns it (echo function); If there is no data after timeout, the connection will be automatically disconnected. The core code is shown in Figure 3 below:

ISSN 2616-5775 Vol. 9, Issue 4: 96-102, DOI: 10.25236/AJCIS.2026.090412

```
static void tcp_server_thread(void *arg) {
  struct tcp_pcb *pcb = tcp_new();
  tcp_bind(pcb, IP_ADDR_ANY, 8080);
  struct tcp_pcb *listen_pcb = tcp_listen(pcb);
  tcp_accept(listen_pcb, tcp_accept_callback);
  while(1) {
    sys_check_timeouts();
    osDelay(10);
  }
}
```

Fig. 3 TCP Core Code

4.3.2 UDP Communication Implementation

Design UDP broadcast function to periodically send device status. The core program is shown in Figure 4:

```
static void udp_broadcast_thread(void *arg) {
  struct udp_pcb *pcb = udp_new();
  ip_addr_t broadcast_addr;
  IP4_ADDR(&broadcast_addr, 255, 255, 255, 255);
  while(1) {
    char data[50];
    sprintf(data, "Temp: %d, Hum: %d", temp, hum);
    udp_sendto(pcb, data, strlen(data), &broadcast_addr, 8081);
    osDelay(1000);
  }
}
```

Fig. 4 UDP Core Code

In the application layer design, in addition to basic communication functions, a simple protocol command parsing mechanism is also implemented, which supports remote querying of device status and configuration parameters. The system manages the TCP connection lifecycle through a state machine, supports multi client connection management and data packet processing, and enhances the robustness and compatibility of the system.

5. System testing and result analysis

5.1 Testing Environment

Hardware: STM32F407 development board PC(Windows 10), Ethernet switch;

Tools: Network Debugging Assistant (testing TCP/UDP), Wireshark (packet capture analysis).

5.2 Test Cases and Results

5.2.1 Connectivity Testing

Ping test: PC ping STM32 IP address (such as 192.168.1.100), continuously sending 100 data packets with a packet loss rate of 0 and an average response time of 1.2ms.

5.2.2 TCP Communication Test

After establishing a TCP connection, the PC sends a 1000 byte packet to STM32, and STM32 returns the data to verify:

Data accuracy: 100% (no errors or missing packets);

Maximum transmission rate: approximately 800KB/s (limited by SPI interface speed).

5.2.3 UDP Communication Test

STM32 sends 100 byte broadcast packets every second, and after receiving them on the PC, statistics show:

Reception success rate: 99.5% (5 packets lost in 1000 transmissions due to network interference);

Delay: Average 0.8ms.

5.3 Result Analysis

The system has implemented stable Ethernet communication function, TCP protocol is suitable for scenarios with high reliability requirements (such as control instruction transmission), and UDP protocol is suitable for scenarios with high real-time requirements (such as sensor data broadcasting). Due to interface speed limitations, the communication speed is slightly lower than the theoretical value of 100BASE-T (100Mbps), but it can meet the needs of most embedded applications. Through Wireshark packet capture analysis, the system performed normally during protocol interactions such as ARP request response, ICMP response, TCP handshake, and disconnection, without experiencing protocol stack crashes or memory leaks. In the long-term running test (continuous operation for 72 hours), the system remained stable without disconnection or data abnormalities, verifying the reliability and robustness of the software and hardware design.

6. Conclusion and Prospect

The network communication system based on STM32F407 designed in this article achieves stable communication of TCP/UDP protocol through the collaborative design of hardware and software, verifying the feasibility of STM32F407 in the field of embedded networks. The system has the advantages of low cost, high integration, and stable operation, and is suitable for various scenarios such as industrial control, smart homes, and remote monitoring. In the future, optimization can be carried out in the following directions: adopting higher performance Ethernet controllers that support hardware TCP/IP acceleration; Add WiFi modules (such as ESP8266) to achieve wired/wireless dual-mode communication; Transplant TLS protocol to enhance the security of data transmission [9].

In addition, application layer protocols such as MQTT and HTTP can be further integrated to achieve support for cloud platforms [9]; Optimize protocol stack memory management and task scheduling strategies to enhance multi connection concurrent processing capabilities; Combining RTOS to achieve multitasking collaboration and meet the needs of more complex real-time network applications. This system provides a complete and extensible solution for embedded devices to access the Internet, and has good engineering application prospects.

Acknowledgements

The authors gratefully acknowledge the financial support from the Tianjin Municipal Education Commission Research program 2024KJ061 and School level educational reform project JG01223093 funds.

References

[1] Wang Tianmiao, Wei Hongxing. Embedded System Design and Example Development [M]. Beijing: Tsinghua University Press, 2020.
[2] Dunkels A. Design and Implementation of the lwIP TCP/IP Stack [J]. Swedish Institute of Computer Science, 2001, 2(1): 1-47.
[3] STMicroelectronics. RM0090 Reference manual - STM32F407xx advanced ARM-based 32-bit MCUs [EB/OL]. (2019-10-01) .
[4] Microchip. LAN8720A Datasheet - Small Footprint RMII 10/100 Ethernet Transceiver with HP Auto-MDIX and flexPWR® Technology [DS]. 2021.
[5] STMicroelectronics. STM32F407IGH6 Datasheet [Z]. 2019.
[6] STMicroelectronics. Description of STM32F4 HAL and low-layer drivers: UM1725 [EB/OL]. (2021-01-01) .
[7] lwIP Contributors. lwIP - A Lightweight TCP/IP stack [EB/OL]. [2024-11-20].

ISSN 2616-5775 Vol. 9, Issue 4: 96-102, DOI: 10.25236/AJCIS.2026.090412

[8] Huang Keya. ARM Cortex-M4 Embedded System Principles and Applications: Based on STM32F407 Microcontroller HAL Library Development [M]. Beijing: Tsinghua University Press, 2024.
[9] OASIS. MQTT Version 5.0: OASIS Standard [S/OL]. (2019-03-07) .