Bidirectional Multi-Device Daisy Chain Communication Architecture and Verification Based on UART Serial Communication Protocol

Xun Sun *

Maynooth International Engineering College, Fuzhou University, Fuzhou, 350108, China

* Corresponding Author Email: 832101210@fzu.edu.cn

Abstract. This paper proposes an innovative bidirectional multi-device daisy chain communication framework grounded in the Universal Asynchronous Receiver-Transmitter (UART) serial communication protocol. This intricate architecture employs four distinct physical Input/Output (IO) interfaces to facilitate dynamic serial communication amongst multiple devices. These devices can be conjoined in a sequential daisy chain or in a closed-loop ring configuration. To further enhance the functionality, our system incorporates a unique mechanism that leverages randomized device codes for rapid address allocation. This not only facilitates directional communication but also supports broad-spectrum broadcast transmissions. An exhaustive simulation of the said protocol was carried out utilizing Proteus software in tandem with a Microcontroller Unit (MCU) model. Critical parameters such as communication response delay, the cumulative failure rate of transmissions, and the isolated failure rate of individual data packets between nodes were meticulously evaluated. Preliminary findings underscore the protocol's potential in fostering cost-effective and dependable communication amongst a plethora of devices. The practicality and versatility of this system manifest prominently in arenas like cluster robot command, robotic automation, and holistic environmental monitoring.

Keywords: UART protocol; daisy-chain communication; address allocation; simulation.

1. Introduction

With the rapid development of the Internet of Things, the demand for multi-device communication is growing. Traditional network communication protocols often require a large number of physical IO ports and complex configurations, which make it difficult to meet the requirements of resource consumption, power consumption, and stability indicators, making the cost, energy consumption, and reliability of multi-device communication challenging [1].

UART (Universal Asynchronous Receiver/Transmitter) protocol, as the earliest application of the general computer communication protocol, realizes reliable asynchronous full-duplex communication between two devices through three signal lines TX, RX, and GND [2]. It is a preferable protocol choice when bandwidth, cost, and speed requirements are relatively low, along with certain requirements for transmission distance.

The development of electronic engineering technology and the widespread use of serial port protocols in industrial production parameter detection, robot cluster control, real-time data collection and other applications has created a need for a new upper-layer communication protocol [3-6]. This protocol should have low interface consumption, support automatic system expansion without human assistance and address allocation, and be decentralized while being robust.

This paper extends the UART protocol to an upper-layer implementation. Each terminal in the system uses two pairs of serial ports to realize a daisy-chain dynamic multi-device full-duplex communication protocol. The terminals in the link can be linked in chain or ring shape, which are hot-swappable with a high upper limit of the volume of expansion devices. Simulation software is also used to verify the feasibility of this transmission protocol.

This paper first introduces the overall structure of the system and then introduces the mutual recognition of interconnected devices, the identification of the link structure state, the automatic refresh and allocation logic of link addresses, and the communication protocol between devices in

chain and ring structures under the protocol. It then introduces the overall running logic and configuration method of the terminal program from the aspect of an individual device. Finally, this paper introduces the process and results of performance testing verification of the protocol using the Proteus simulation software using STM32F103ZET6 MCU model of STMicroelectronics.

2. System Structure Overview

As shown in Figure 1, the devices in this system can use the serial protocol to link together in a chain or ring formation, using either physical wired connections or wireless UART data relay methods. When linked in chain or ring formation, devices located in the middle of the link structure use two pairs of IO ports for UART communication. They read and relay information from both ends and also receive information sent to themselves or broadcast information. Devices located at the ends of the link chain only use one pair of IO interfaces for communication with devices on one side. Different encryption methods such as AES encryption and physical protocols including RS323, TTL, and RS485 standards could be chosen in order to meet the requirements of different applications and transfer distances [7, 8].

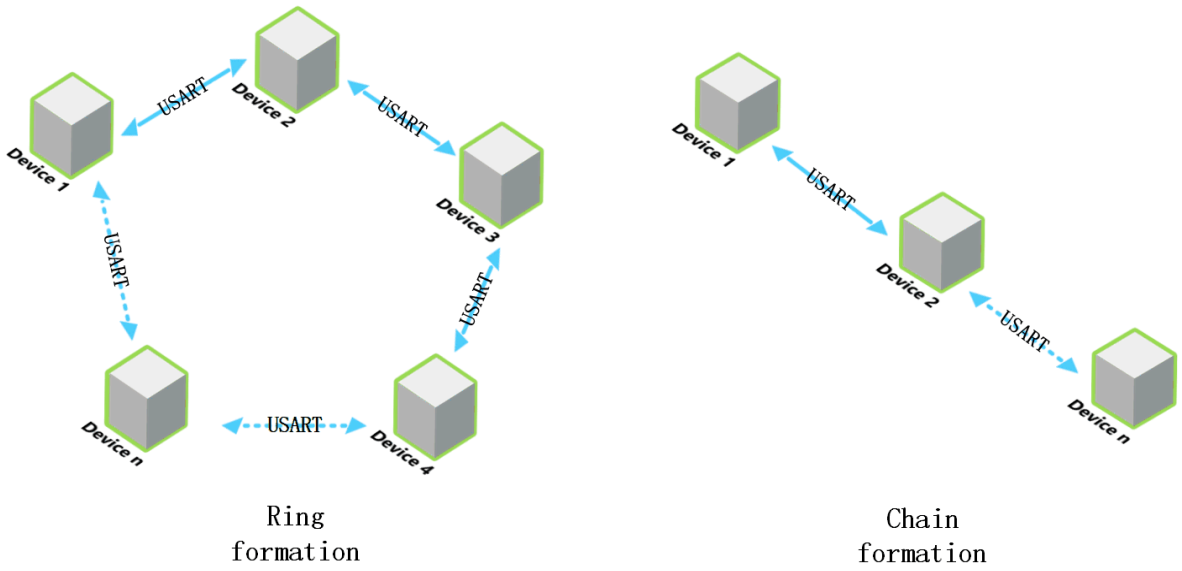

Fig. 1 Structure of the system (Photo/Picture credit: Original)

In ring formation, all devices in the communication link use two pairs of IO interfaces. In the link protocol, a pair of devices that have successfully connected to each other are defined as adjacent devices. If the physical connection between two adjacent devices in the link fails or a software error causes the communication to fail, the system logic can automatically switch all devices in the link to chain communication mode, ignoring the link error and continuing to communicate. Enhancing the error tolerance of the system.

In all serial communication processes, the standard parameters for using the serial protocol are followed. Each serial data packet is one byte in size. 1 start bit, 8 data bits, and 1 stop bit are used. For the checksum, the system uses 0 bits of parity because an upper-layer data packet-level parity check has been implemented, the calculation of the checksum could be implemented using both hardware or software methods [9]. The communication baud rate can be set by the user while the selection of the baud rate might have an impact on the stability of transmission and the transmission speed. For the error handling mode, since there is an upper-layer implementation for data packet loss retransmission, the lost data frames in the communication process are discarded [10]. To avoid an inappropriate baud rate chosen, the baud rate of the system could be configured to be continuously adjustable throughout the process if possible [11]. Finally, to achieve the design goal of reducing the number of IO ports used, the protocol does not use additional IO to implement software or hardware flow control.

3. System Implementation

3.1. Device connection

Logic diagram of connection process is shown in Figure 2. In the disconnected state, the two serial ports of the device continuously send data packets with the value of 0x00ff as the handshake initialization signal. When the handshake signal is received from the corresponding RX, the device enters the waiting address allocation mode. The corresponding serial port TX continuously sends data with the value of 0x00ee, indicating that the device has confirmed the connection with the adjacent device. After the connection between the two sides is established, the serial port communication is terminated, and the address allocation is awaited.

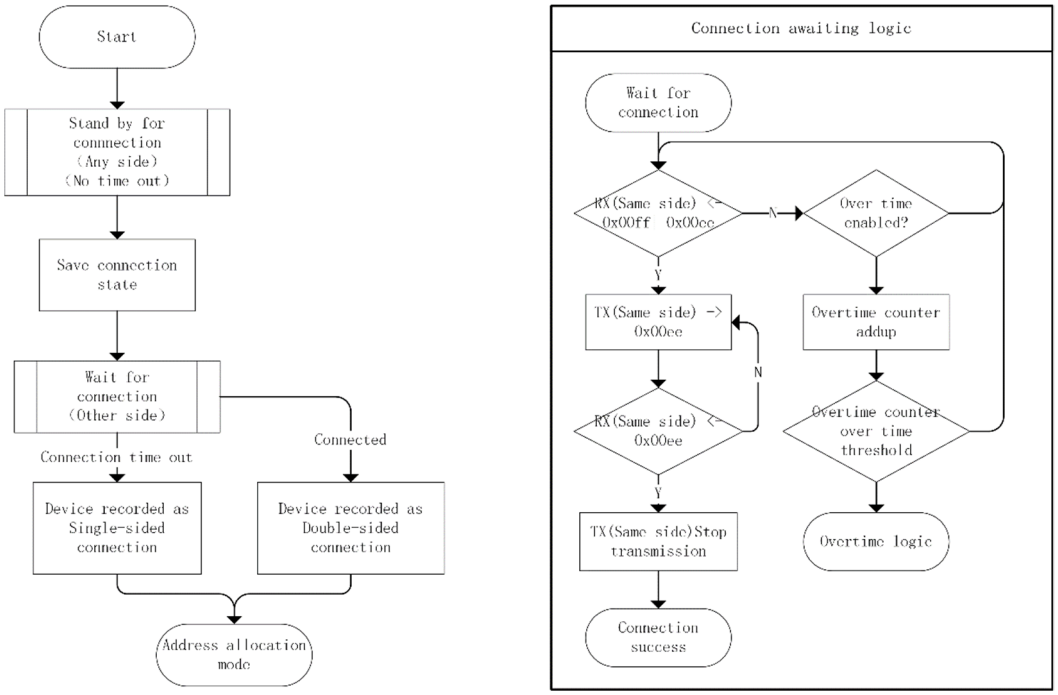

Fig. 2 Logic diagram of connection process (Photo/Picture credit: Original)

The connection process based on serial port protocol is shown figure 3. There are two correct connection methods for devices with the outside world, namely two-sided connection, and single-sided connection. Two-sided connection refers to the device using two pairs of serial port IO to connect with the previous and next two devices in a daisy chain structure. Single-sided connection refers to the device being located at the ends of the chain structure.

During the initial handshake connection, the device will record the serial port that successfully completes the handshake. If it is a two-sided connection, the device will record that it is in double-sided mode. If it is a single-sided connection, the device will record that it is in single-sided mode. This information will be used in the master-slave allocation logic in the next step of address allocation.

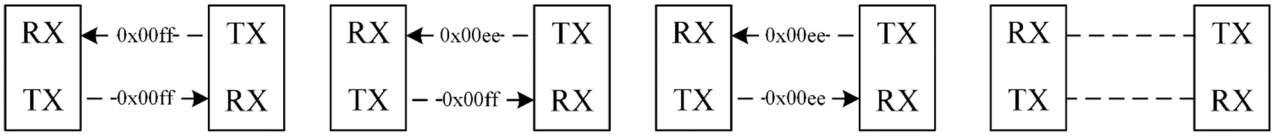

Fig. 3 Connection process based on serial port protocol (Photo/Picture credit: Original)

3.2. Address Allocation

3.2.1 System Operation

To allocate addresses, each terminal needs to generate a long random number, which will be held by the device during the address allocation process as its unique identifier. Since the maximum device capacity of a single system is designed to be 256, it is decided to use a 64-bit random number as the

device's unique ID. When all the numbers generated are evenly distributed, the probability of ID duplication is less than 1e-17, so the ID generated by the device can be considered unique. The generated ID will be sent to the two sides of the connected side of the device according to the device's connection status. The encoding structure and generation logic of the data packet are shown in the corresponding parts of Figure 4.

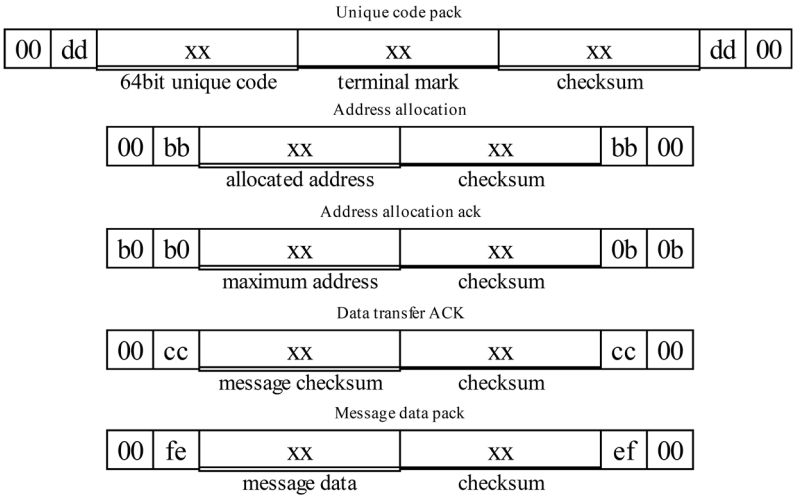

Fig. 4 Different data pack formats (Photo/Picture credit: Original)

The address allocation process is shown in Figure 5. In the case of a chain connection, in order to reduce the complexity of the system, the devices at the two ends of the chain will include the terminal mark in the data packet that sends the ID. The random data packet with the terminal mark will be directly related to the next device by the non-header-end devices in the chain until it is sent to the device at the other end of the chain.

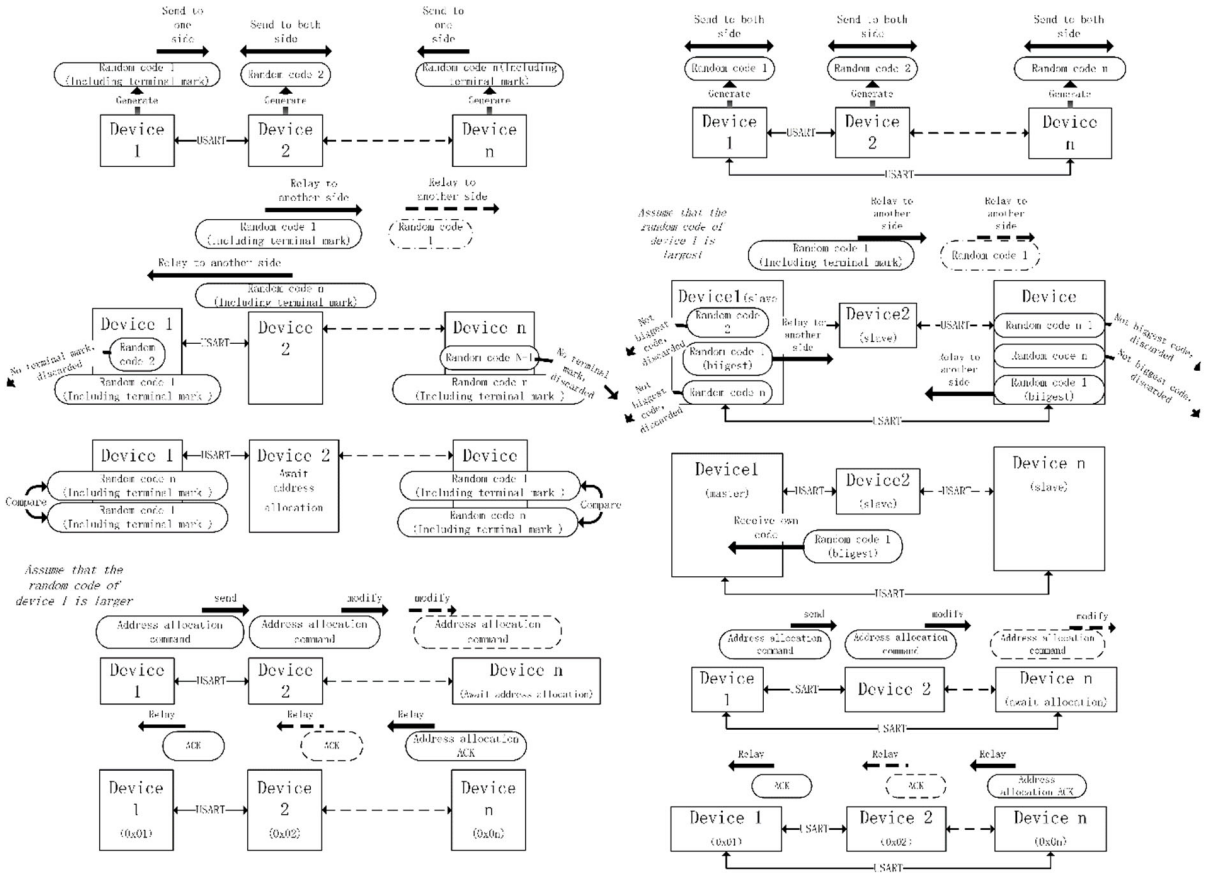

Fig. 5 Address allocation process (Photo/Picture credit: Original)

After the two-end devices get the unique code packet of the other end, they will compare their unique code. The device with the larger random code will act as the address allocation end. The address allocation end (which has address 0x01 by default) will send an address allocation instruction to the device connected to it, the format of which is shown in the corresponding part of Figure 4. The allocated address is its own address plus one. The devices in the chain that receive the address allocation instruction will send a new address allocation instruction to the next device, with the address value increased by one each time. Finally, after the terminal device receives the address allocation instruction, it will return a return data packet indicating that the connection is complete. The return packet also contains the current system address range information, that is, the effective maximum address of the system. The format of the return packet is shown in the corresponding part of Figure 4.

In the case of a ring connection, as shown in the relevant part of Figure 6, the unique code of each terminal in the link will be transmitted along the link structure in the link in both the clockwise and counterclockwise directions. Each terminal will compare the received unique code and its own unique code and take the larger unique code to the other end. Until the device with the largest unique code receives data packets with the same unique code from both sides, it will be the master device for address allocation.

At this point, the master device will send an address allocation instruction to one of the devices. The address allocation mode is the same as that of chain connection. After receiving the address allocation instruction from the other side, the master device will send a return packet with the same format as the chain mode in the opposite direction to test the integrity of the communication link.

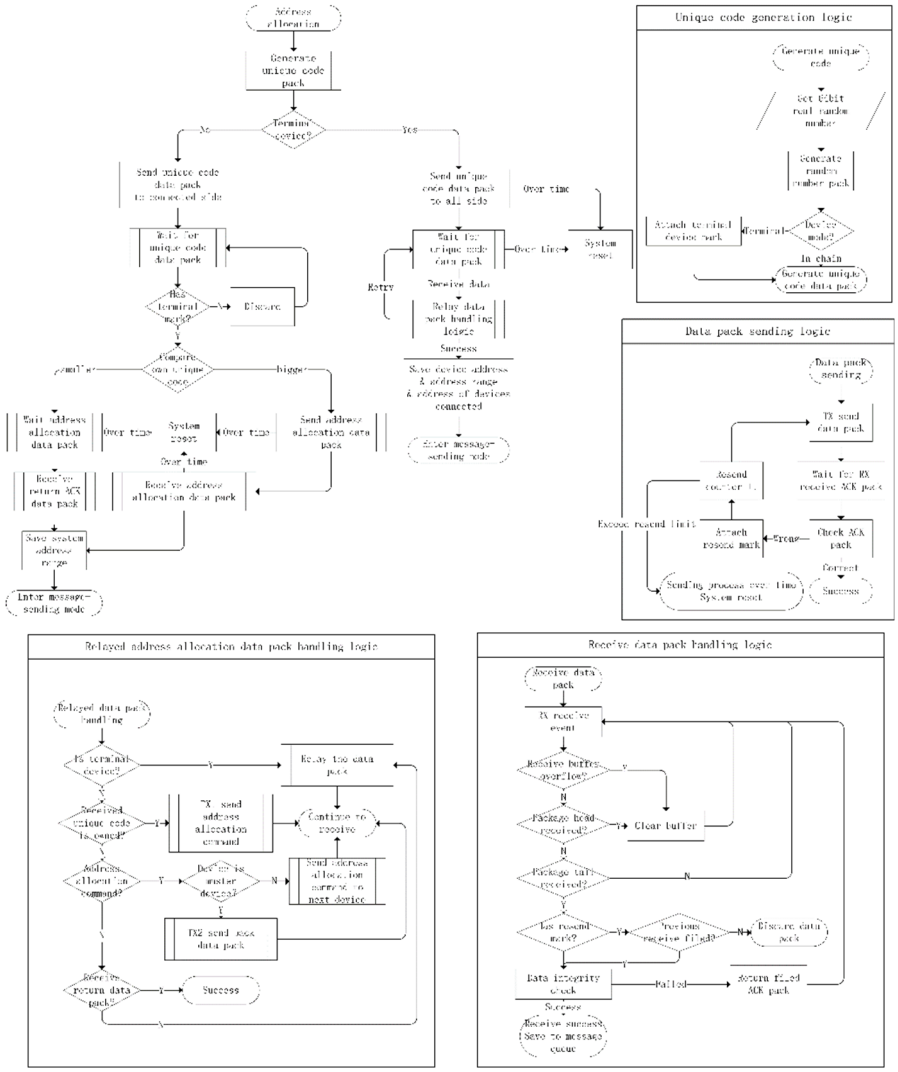

Fig. 6 Logic diagram of address allocation (Photo/Picture credit: Original)

3.2.2 Single device operation

For an individual device, its operation will start with identifying whether it is in the end position. If the device is in the end position, it will relay the unique code to the other end device through the intermediate device in the link. After receiving the unique code of the other device, it will set the master-slave mode of address allocation by comparing the unique codes of both sides. It will wait for or issue an address allocation instruction to perform address allocation. After the address allocation process is completed, the end device in the slave state will send the return code through the intermediate device. At the same time as confirming the validity of the physical connection, all the devices in the link will exit the address allocation mode and enter the message-sending mode.

If the device is not in the end position, it is necessary to confirm whether the system is in ring connection mode. If the device receives a unique code packet with a terminal mark, it proves that the entire system is in a chain link mode. The device itself only needs to pass the unique code of two terminal devices, wait for the address allocation instruction and ACK return data, and then enter the message-sending mode.

If there is no terminal device in the system, it can be considered that the system is in a ring link mode. It is necessary to determine the only host to perform address allocation, and at the same time hold the address 0x01. Therefore, all devices with two-sided links will send their unique codes to the two devices on both sides. The two devices will compare the unique codes and pass the larger unique code to the next level. Until the unique code value of the device with the largest unique code is transmitted back to itself along the ring in a clockwise and counterclockwise direction, it will set the device as the address allocation host. Other devices are always in slave mode, waiting for the address allocation instruction.

After completing the address allocation, the address allocation host will also send the return code in the opposite direction of the address allocation. The device itself receives the verification of the loop integrity and makes the devices in the slave mode enter the message-sending mode.

3.3. Message Sending

In the message-sending mode, the system uses the encoding format corresponding to part of Figure 4, and the logic is shown in Figure 7.

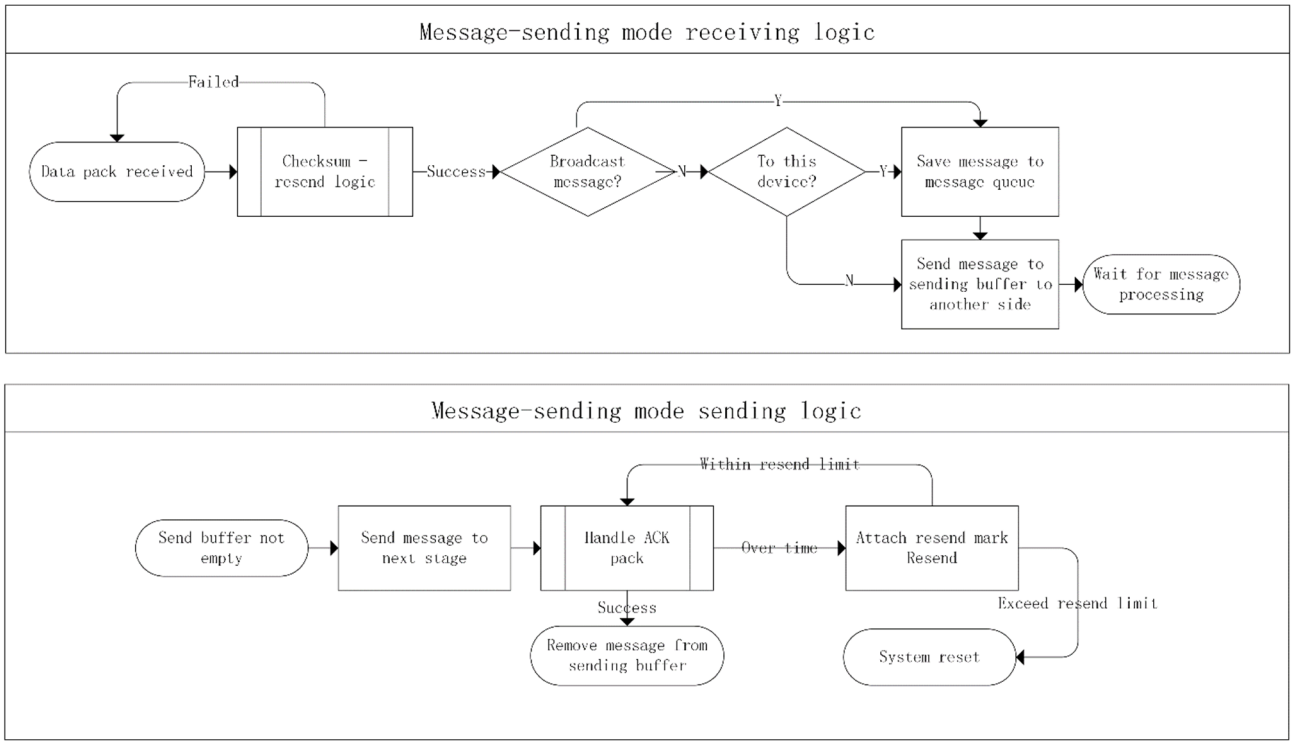

Fig. 7 Logic diagram of message-sending mode (Photo/Picture credit: Original)

After the system receives a message, it will first check the ACK return packet. If the verification is successful, it will judge whether the message received belongs to its own terminal or a broadcast message. If it meets any of the above conditions, it will save the message to the message data cache, waiting for other system processes to read it. It will also call the message receive callback interface. Users could implement the callback interface if they require message processing using such a method.

For the sending process, the emptying process of the message cache uses interval polling. Once the timer triggers the message cache check, if the message cache contains data, the sending process will send a limited amount of data that could be customed in the code. Specifically, once the message cache is not empty, the sending logic will judge the receiver address and send the message to the next terminal in the direction of the network distance. At the same time, it will wait for the ACK check packet from the receiving terminal. If the check packet is not received, the data will be attached with a retransmission mark and retransmission will be attempted. The message cache will be released by the sending process after the information in it is sent, realizing the circular storage of information. When the message cache is more than three-quarters full, the sending process will change the three-quarters full flag to prompt the system to increase the polling rate.

3.4. Error Handling

The system is designed to handle link failures during operation. The system confirms the link with the next device by receiving a return verification packet after sending a data packet. If the system does not receive the verification packet after multiple retransmissions, the device is considered to be faulty. The system then sends a reset command to both devices on its sides. The devices that receive the reset command also send a reset command to the other device. After a delay, the system enters pairing mode and repeats the address allocation and connection creation process.

The connection of new devices also triggers a system reset and address re-allocation. However, data stored in the message queue will continue to be sent after the connection is reestablished.

4. Tests and Results

4.1. Testing Method

This protocol was simulated using the stm32f103zet6 MCU module in the Proteus simulation platform with additional auxiliary circuitry. The simulation experiment tested the address allocation, point-to-point communication, and broadcast communication functions of the system, and also performed communication performance indicator tests based on these functions, which included Device communication response delay, final transmission failure rate, and single-transmission failure rate of data packets between nodes.

Device communication response delay refers to the time it takes for a communication data packet to travel from one end of a link to the other. It is affected by the cache delay, address delay, and system response delay in the system. In the test, a microcontroller located at one end of the link sends a data packet (64 bits in total, including a 2-byte random number) at a specified interval. After sending, the GPIO pin is pulled high. The microcontroller at the other end of the link receives the data packet and returns a checksum. After the checksum is successful, the receiving end pulls the same pin low. The length of the high level is measured using an oscilloscope in Proteus simulation software to reflect the response time of the device receiving the information.

The final transmission failure rate and single-transmission failure rate of data packets between nodes refer to the proportions of discarded data packets and checksum failure data packets transmitted between all nodes during the transmission of the two devices with the farthest distance in the network. In the test, the sending end has a sending counter, a checksum failure error counter, and a data packet discard counter. The counter values are then calculated against the total number of data packets sent. This process is repeated multiple times to obtain the final transmission failure rate and single transmission failure rate, which are used to reflect the reliability of the communication process.

In actual operation, the difference in test parameters and the selection of devices may lead to significant differences. This study selected certain UART data transmission parameters for testing. The relevant parameters are as follows in table 1.

Table. 1 Simulation parameters

Parameters	Value
Data send interval	500ms
Data pack amount	100
Data pack size	64bit
Terminal amount in the chain	3~20
Baud rate	115200Baud/s
MCU main clock frequency	72MHz
Highest GPIO flipping rate	50MHz

4.2. Test Results and Analysis

As the communication and addressing methods in ring connection and chain connection modes are similar, the performance measured in the farthest connection case is similar. Therefore, the test results show the average of the results obtained by conducting the same multiple tests using ring connection and chain connection.

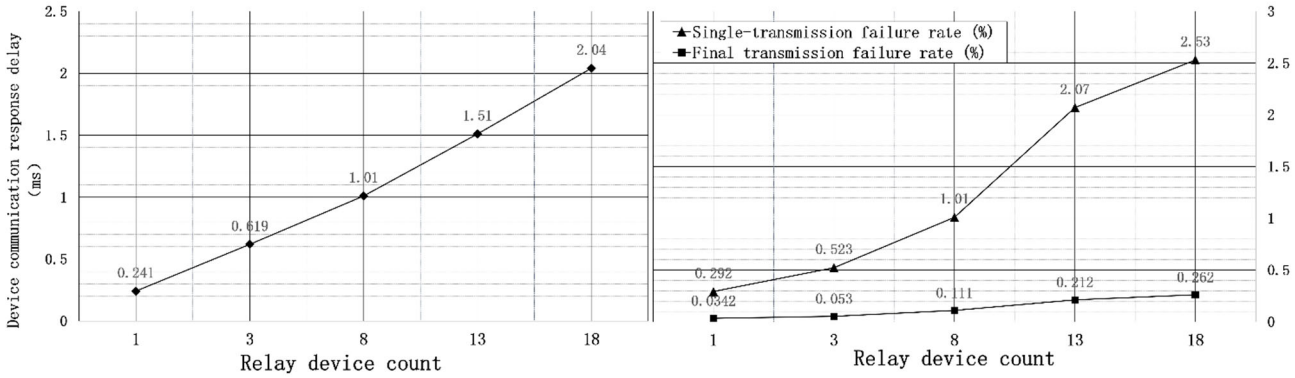

Fig. 8 Statistic data of test

As shown in Figure 8, when the link length is within a reasonable range, the measured data delay and data packet loss situation increase steadily with the increase in the number of relay terminals, and the data transmission efficiency shows a downward trend. Within the range of relay numbers used in the test, the data transmission loss rate and delay are within the range of conventional non-high instantaneous application requirements. With the increase in the number of relay devices, the failure rate of a single transmission and the communication delay have a certain increase. It should also be noted that the performance of this protocol may depend largely on the hardware performance and the parameters used [12]. On better hardware architectures, this protocol can also provide more reliable transmission due to the improvement of the underlying serial port communication protocol. Therefore, in this protocol, serial link communication of a large number of devices may lead to a relatively high data packet retransmission rate, which may lead to a certain delay fluctuation and increase.

In summary, the system successfully achieved the goal of using fewer physical ports to realize reliable multi-device dynamic serial communication. It can perform efficient address allocation, and directional and broadcast communication, which has relatively stable communication performance indicators.

5. Conclusion

This protocol is based on the UART communication protocol and implements multi-device daisy-chain links on its upper layer. It consumes four physical IO interfaces, and the system uses a random

device code comparison mechanism to implement a fast address allocation mechanism after establishing a connection. It supports ring and chain link connection methods and can achieve low-cost and reliable communication between multiple devices.

This research can be widely applied to the rapidly developing fields of cluster robot control, automation, and environmental monitoring. It can be combined with LoRa or other wireless connection protocols to achieve low-cost, low-overhead, and fast bidirectional interconnection between multiple devices. The established link can be used to implement data transmission sharing from device to device or broadcast to the entire system in the formed communication structure.

Further development of this protocol can also improve the effective data ratio, realize the function of multi-loop interconnection, and improve the protocol scalability. It can provide an excellent and convenient communication implementation solution in most scenarios.

References

[1] Murali A, Kakarla H K, Anitha Priyadarshini G M. Improved design debugging architecture using low power serial communication protocols for signal processing applications. International Journal of Speech Technology, 2021, 24: 291-302.

[2] Harutyunyan S, Kaplanyan T, Kirakosyan A, et al. Design and verification of autoconfigurable UART controller. 2020 IEEE 40th International Conference on Electronics and Nanotechnology (ELNANO). IEEE, 2020: 347-350.

[3] ZHI Chaoqun. Research on Communication Network Deployment and Path Planning Strategy for Water Quality Monitoring Robot Swarm. North University of China, 2023.

[4] LIU Zijun. Research on Flight Monitoring and Control System Design for Pigeon Robot Swarm. Zhengzhou University, 2022.

[5] Sudhapriya K, Amudha A, Divyapriya S, et al. Wireless vehicle control with speed adjustment. 2022 6th International Conference on Computing Methodologies and Communication (ICCMC). IEEE, 2022: 583-588.

[6] Elangovan Y, Saraf M N, Satyanarayana B, et al. A Compact and Cost-Effective Data Acquisition Module (C-DAQ) for Particle Physics Instrumentation. International workshop on Advanced Radiation Detector and Instrumentation in Nuclear and Particle Physics. Cham: Springer International Publishing, 2021: 109-114.

[7] Liang Xiaoxue. Design and Implementation of Encrypted UART IP Core Based on AES Algorithm. Xidian University, 2022.

[8] Xu Zhenzhong, Zhang Jiamin, Zhou Jie et al. Research and verification of embedded device interconnection method based on UART. Information Recording Materials, 2022, 23(11): 28-31.

[9] Rathore K, Khosla M, Raman A. A New Approach to Improve Reliability in UART Using Checksum Algorithm. Proceedings of Fourth International Conference on Computer and Communication Technologies: IC3T 2022. Singapore: Springer Nature Singapore, 2023: 243-252.

[10] Singh A, Gupta K, Mala J. Determination of UART Receiver Baud Rate Tolerance. Proceedings of the Sixth International Conference on Computer and Communication Technology 2015. 2015: 265-270.

[11] Wu Zhiyong, Guo Yuanxin, Liu Yuqin. Continuous Adjustable Baud Rate UART Interface Design Based on FPGA. Communication Technology, 2018, 51(01): 252-256.

[12] Zhang Y, Yu Z, Yang X. Design and Implementation of Configurable UART with Self-Test Function. 2023 5th International Conference on Electronic Engineering and Informatics (EEI). IEEE, 2023: 222-226.