LoRa Transceiver with Error Detection and Retransmission

Overview

This Arduino program implements a simple LoRa transceiver using the LoRa library by Sandeep Mistry. The transceiver sends and receives messages over the LoRa protocol and implements basic error detection and retransmission functionality.

Code Structure

The code is structured into several parts:

1. **Global Variables:** These variables store the state of the program, including the last message sent, whether the program is waiting for an acknowledgement, the time the last message was sent, and the number of attempts to send the current message.
2. **Setup Function:** This function runs once when the program starts. It initializes the serial communication for debugging and the LoRa library for communication.
3. **Main Loop:** This function runs repeatedly as long as the program is running. It checks for incoming packets and handles sending new messages and resending old ones.
4. **Packet Checking Function:** This function checks for incoming packets, verifies their checksums, and sends an acknowledgement or negative acknowledgement as appropriate.
5. **Send Function:** This function sends a message with a checksum appended to the end.

How It Works

When the program starts, it initializes the LoRa library and begins waiting for incoming packets. If it receives a packet, it calculates the checksum of the received data and compares it to the received checksum. If the checksums match, it prints the received message to the serial monitor and sends an "ACK" (acknowledgement) back to the sender. If the checksums don't match, it sends a "NACK" (negative acknowledgement) back to the sender.

The program also periodically sends a message ("Hello, world!" in this example). After sending a message, it waits for an "ACK" from the receiver. If it receives an "ACK", it sends the next message. If it receives a "NACK", or if it doesn't receive an acknowledgement within a certain timeout period, it resends the message. It will attempt to resend a message up to a certain limit (5 times in this example), after which it gives up and moves on to the next message.

Error Detection and Retransmission

The error detection is implemented using a simple XOR checksum. Each byte of the message is XORed together to create the checksum, which is then appended to the end of the message. When a message is received, the receiver calculates the checksum of the received data and compares it to the received checksum. If they match, the message is considered valid; if they don't match, the message is considered corrupted.

The retransmission is implemented using acknowledgements and a timeout. After sending a message, the sender waits for an acknowledgement from the receiver. If it receives a positive acknowledgement

("ACK"), it knows the message was received successfully and moves on to the next message. If it receives a negative acknowledgement ("NACK"), or if it doesn't receive an acknowledgement within the timeout period, it resends the message. It will attempt to resend a message up to a certain limit, after which it gives up and moves on to the next message.

Limitations and Possible Improvements

This program is a simple example and has several limitations:

- It doesn't handle packet loss, other than by resending messages when a "NACK" is received, or the acknowledgement timeout expires. A more robust program might implement a more advanced retransmission strategy, such as exponential backoff.
- The error detection is very basic and may not catch all errors. A more robust program might use a more advanced error detection method, such as a CRC.