```c


#include "UART_Bluetooth.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include "Command_Handler.h"
#include "nrf24.h"
#include <stdlib.h>
#include <string.h>

uint8_t* uartBufferPos;
uint8_t* uartTxMessageEnd;
bool commandAvailable;

void initBluetoothUart(){
    // UART Initialization : 8-bit : No parity bit : 1 stop bit
    UBRR0H = (BRC >> 8); UBRR0L =  BRC;             // UART BAUDRATE
    UCSR0A |= (1 << U2X0);                          // DOUBLE UART SPEED
    UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00);        // 8-BIT CHARACTER SIZE

    // Setup UART buffer
    initliazeMemory();
    uartBufferPos = command_buffer;
}

void transmitMessage(uint8_t* message, uint8_t length){
    while (!(UCSR0A & (1<<UDRE0)));
    uartBufferPos = command_buffer;
    uartTxMessageEnd = (command_buffer+length);
    memcpy(command_buffer, message, length);
    UCSR0A |= (1<<TXC0) | (1<<RXC0);
    UCSR0B |= (1<<TXEN0) | (1<<TXCIE0);
    UCSR0B &=~(1<<RXEN0) &~(1<<RXCIE0);

    uartBufferPos++;
    UDR0 = *(command_buffer);
}

void transmitMessageSync(uint8_t* message, uint8_t length){
    while (!(UCSR0A & (1<<UDRE0)));
    uartBufferPos = command_buffer;
    uartTxMessageEnd = (command_buffer+length);
    memcpy(command_buffer, message, length);
    UCSR0A |= (1<<TXC0) | (1<<RXC0);
    UCSR0B |= (1<<TXEN0) | (1<<TXCIE0);
    UCSR0B &=~(1<<RXEN0) &~(1<<RXCIE0);
    sei();

    uartBufferPos++;
    UDR0 = *(command_buffer);

    while (transmissionState());
```

```c

}

bool transmissionState(){
    // True : Currently transmitting | False : Transmission finished
    if (uartBufferPos!=uartTxMessageEnd)
    {
        return true;
    }
    else
    {
        return false;
    }
}


void setupReceiveMode(){
    while (!(UCSR0A & (1<<UDRE0)));
    uartBufferPos = command_buffer;

    UCSR0A |= (1<<RXC0) | (1<<TXC0);
    UCSR0B &=~(1<<TXEN0) &~(1<<TXCIE0);
    UCSR0B |= (1<<RXEN0) | (1<<RXCIE0);
    sei();
}

bool catchModuleReply(){
    nrf24_initRF_SAFE((lastTargetModuleID-1), RECEIVE); // CONNECTION TO MODULE: GENERAL RF CHANNEL 112 (lastTargetModuleID-1) offset 1
    uint8_t targetModuleID = lastTargetModuleID;
    uint8_t RF_TIME_OUT;
    while(RF_TIME_OUT!=0xFF)
    {
        if(nrf24_dataReady()){
            nrf24_getData(command_buffer);
            CommandStatus status = DecomposeMessageFromBuffer();
            if (status==SUCCESFUL_DECOMPOSITION&&lastTargetModuleID==targetModuleID) {
              transmitMessageSync(command_buffer, 32);
              return true;
            }
        }
        RF_TIME_OUT++; _delay_ms(2);
    }
    return false;
}

void processReceivedLine(){
    commandAvailable = false;

    CommandStatus status = DecomposeMessageFromBuffer();
    if(status==SUCCESFUL_DECOMPOSITION) {
        if (lastTargetModuleID==MAIN_MODULE){
            //Executed by main module
            HandleAvailableCommand();
        } else {
            //Retransmitted to other module

            RF_TransmissionStatus RF_Status = RetransmissionToModule();

            //Catch module reply

            //bool didModuleRelpy = catchModuleReply();

            // Send RF STATUS
            switch (RF_Status) {
                case RF_UNREACHEABLE_MODULE:
                writeParameterValue(0, &(uint8_t){RETRANSMISSION_FAILED}, 1);
                break;
                case RF_ACKNOWLEDGE_FAILED:
                writeParameterValue(0, &(uint8_t){RETRANSMISSION_FAILED}, 1);
                break;
                case RF_SUCCESFUL_TRANSMISSION:
                writeParameterValue(0, &(uint8_t){SUCCESFUL_RETRANSMISSION}, 1);
                break;
            }
            ComposeMessageToBuffer(MESSAGE_STATUS_ID, 1, PHONE_MODULE);
            transmitMessageSync(command_buffer, 32);


        }
    }else {
}


}

void disableUART(){
    UCSR0B &=~(1<<TXEN0) &~(1<<TXCIE0);
    UCSR0B &=~(1<<RXEN0) &~(1<<RXCIE0);
}

ISR(USART_TX_vect){
    if (uartBufferPos!=uartTxMessageEnd){
        UDR0 = *uartBufferPos;
        uartBufferPos++;
    }
}

ISR(USART_RX_vect){
    if(uartBufferPos!=(command_buffer+uartBufferSize)) {
        *uartBufferPos=UDR0;
        if ((*uartBufferPos==ETB)&&(DecomposeMessageFromBuffer()
          ==SUCCESFUL_DECOMPOSITION)) {
            disableUART(); commandAvailable = true;
        }
        else if(*uartBufferPos==uartCarriageReturnChar) {

            bool hasToReturnCarriage = true;
            uint8_t* savedUartBufferPos = uartBufferPos+1;

            for (uint8_t x = 1; x < 4; x++) {
                if ((uartBufferPos-x)<command_buffer) uartBufferPos = command_buffer+(uartBufferSize-1);
                if (*(uartBufferPos-x)!=uartCarriageReturnChar) { hasToReturnCarriage = false; break; }
            }
            if (hasToReturnCarriage) {
                 uartBufferPos = command_buffer;

            } else {
                uartBufferPos = savedUartBufferPos;
            }

        } else {
            uartBufferPos++;
        }

    } else {
        uartBufferPos = command_buffer;
        *uartBufferPos=UDR0;
    }
}
```