```c


#include "UART_Bluetooth.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include "Command_Handler.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);

    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);
}

void processReceivedLine(){
    commandAvailable = false;

    commandType currentCommand;
    bool success = decomposeCommand(command_buffer, &currentCommand, parameter);
    if(success) currentCommand.handlerFunction();
}

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!=uartEndMsgChar) {
            if(*uartBufferPos!=uartCarriageReturnChar) {uartBufferPos++;} else
              { uartBufferPos = command_buffer; }
        }
        else { disableUART(); commandAvailable = true; }
    } else {uartBufferPos = command_buffer;}
}
```