A Graphical User Interface for an Underwater Reconfigurable Modem

by

Kevin Rice

May 18, 2006

For fullfillment of the Advanced Undergraduate Project for the Department of Electrical Engineering and Computer Science

Completed through the MIT Sea Grant

Project Supervisors: Chryssostomos Chryssostomidis and Ethem Mutlu Sozer

Academic Advisor: Arthur Smith
# Table of Contents

1 Introduction ......................................................................................................................... 3
2 Design Specifications ............................................................................................................... 3
   2.1 Threads ............................................................................................................................. 3
   2.2 Running Commands ......................................................................................................... 4
   2.3 Receiving Responses ....................................................................................................... 4
   2.4 Command Logging ........................................................................................................... 4
   2.5 Terminal Logging ............................................................................................................. 4
   2.6 Batch Runs ....................................................................................................................... 4
   2.7 Receiving Files ............................................................................................................... 4
   2.8 Sending Files .................................................................................................................... 5
   2.9 Text Messages ................................................................................................................. 5
   2.10 Physical Layer Status ..................................................................................................... 5
3 Implementation .................................................................................................................... 5
   3.1 Functionality .................................................................................................................. 5
      3.1.1 State Register .......................................................................................................... 6
      3.1.2 Packet Headers, Subheaders and Trailers ............................................................... 6
      3.1.3 Sending Data – Text, Files, Pictures ...................................................................... 6
      3.1.4 Plotting ................................................................................................................... 7
      3.1.5 Receiving Files and Pictures ................................................................................... 7
      3.1.6 Raw Packets to GUI Display .................................................................................... 7
      3.1.7 Command Logging ................................................................................................ 8
      3.1.8 Terminal Logging ................................................................................................... 8
      3.1.9 Byte Conversions .................................................................................................. 8
   3.2 GUI Components ............................................................................................................. 8
      3.2.1 Terminal Window .................................................................................................... 8
      3.2.2 Byte Input Window ................................................................................................ 8
      3.2.3 Packet Structure Fields ......................................................................................... 9
      3.2.4 Batch Mode Button ............................................................................................... 9
      3.2.5 Send File Button ................................................................................................... 9
      3.2.6 Update Button ....................................................................................................... 9
      3.2.7 Text Message Window ........................................................................................... 9
      3.2.8 Packet Information Fields ..................................................................................... 10
   3.3 Java Class Structure ....................................................................................................... 10
      3.3.1 Threads .................................................................................................................. 10
      3.3.2 Java Classes ........................................................................................................... 11
4 Commands .......................................................................................................................... 13
   4.1 Sending Text Messages, Files and Pictures ................................................................... 13
   4.2 Verbose Level .................................................................................................................. 13
   4.3 rModem State ............................................................................................................... 14
   4.4 Default Recipient ID .................................................................................................... 14
   4.5 rModem Packet Size .................................................................................................... 14
   4.6 rModem NodeID ......................................................................................................... 14
   4.7 Clock Tick Value ......................................................................................................... 14
   4.8 Clock Tick Frequency .................................................................................................. 14
   4.9 Scatter Plot Mode ....................................................................................................... 15
   4.10 Received Samples Mode ............................................................................................. 15
   4.11 Channel Estimation Mode .......................................................................................... 15
5 Conclusion .......................................................................................................................... 15
6 References .......................................................................................................................... 17
1 Introduction

Underwater research is predicated on the use of acoustic modem technology to provide real-time analysis of data collected by various platforms such as autonomous underwater vehicles (AUVs). Without acoustic modems, data can only be processed once an AUV makes it back to a base station. This makes experiments much less efficient because they cannot be performed in real time. Research on new communication and networking algorithms is required to enable real-time high-rate data communications.

The Reconfigurable Modem (rModem) is being developed as a rapid prototyping environment for underwater acoustic communications and networking algorithms. The rModem is very flexible and modular enabling researchers to run their algorithms in real-time in real environments. This makes for rapid development of algorithms, which is important in the development of the newest and best underwater networking and data collection protocols [1].

This paper focuses on discussing a graphical user interface (GUI) for the rModem. Before this graphical user interface was developed, experiments performed with the rModem had to be carried out through a terminal. This made experiments very tedious and any mistake in the command input would send incorrect commands to the modem. Furthermore, it was impossible to send and receive files through the terminal. The GUI also acts as a simple transport layer (layer 4) where large data are divided into multiple packets.

Besides giving the user the capability to send and receive files, the rModem GUI makes running experiments much easier and quicker. The GUI creates two logs, one that logs the commands sent to the modem so experiments can be duplicated as a batch run, and one that logs the commands and response of the modem so results can be analyzed after the experiment. The GUI also supports accessing the lower level layers of the modem. The GUI can ask the modem for the channel estimate, I-Q scatter plot or raw channel samples. This data can be plotted in real time using MATLAB for initial analysis and saved to disk for further analysis using MATLAB or other software. The GUI has other features, all of which are described below that make it easier and possible to run complex experiments using the rModem.

2 Design Specifications

The rModem GUI will consist of a terminal displaying commands and responses as well as one displaying outgoing and incoming text messages. Other sections of the GUI will display relevant information to the user.

2.1 Threads

There are multiple threads running in the application. Without separate threads the user would not be able to input information while bytes were being accepted into a buffer.
or packets were being processed. The simplest thread constantly monitors the serial connection of incoming bytes and adds them to a buffer. Another thread processes packets from the buffer and deletes them after they are processed and passed to the user interface. The highest level thread accepts input from the user to send commands to the modem. In certain instances other threads need to be spawned to run command files and send files.

2.2 Running Commands

Most basic commands to the rModem will be inputted in a terminal-like window.

2.3 Receiving Responses

The responses to basic commands will be displayed in the terminal window.

2.4 Command Logging

An important feature of this GUI that a regular terminal does not have is the ability to log commands. Every time the application is opened a new command log is created. This log notes each command, when it was run, and how long it was run after the last command. This file can be later used as a batch run file or the time line of the experiment.

2.5 Terminal Logging

The terminal log, logs everything that appears in the terminal window. This includes commands and results. This can be used to analyze what happened in a command sequence.

2.6 Batch Runs

It is possible for the user of the GUI to create batch files that send commands automatically, just like if the user was typing them in himself. He has the option of running batch files several times or just once. A batch file must contain the time duration between each command and each command itself. Command logs can be used as batch files to recreate experiments that a user performed manually.

2.7 Receiving Files
One capability that was almost impossible to do with a simple terminal is the ability to receive files. The rModem will send files to the GUI by splitting the file up into multiple packets and sending each one with information preceding the data to tell the GUI information like which node sent the file to the modem, the size of the file, and the packet number. Once all the packets are received the GUI will piece this file together and save it on the disk. If the file is a picture it will be immediately displayed.

2.8 Sending Files

The GUI will be able to send files as well as receive them. The method for sending a file is the same as receiving one, only the roles are reversed, except for one exception. If the GUI sends too many packets too fast the modem may become busy, in which case the GUI will wait a specified amount of time and continue sending packets.

2.9 Text Messages

Text messages can be sent back and forth from the rModem to the GUI. These messages will have a separate terminal window that will function like an instant messenger client window.

2.10 Physical Layer Status

It may be useful for a user to view physical layer information such as channel estimate and scatter plot. The rModem will provide options to send channel estimate and/or scatter plot for each received packet. The scatter plot will be obtained from I-Q values sent by the modem. The channel response will be a time series graph obtained at the output of the preamble correlator. The data in these values will be saved to disk for later graphing and immediately displayed for the user using MATLAB’s graphing capabilities. In addition, the rModem can send raw channel samples for low level off-line processing.

3 Implementation

This project was implemented in Java version 1.5.0 because of its ease of user interface development. The Eclipse integrated development environment was also used in this project.

3.1 Functionality
3.1.1 State Register

The state register is returned as a response to the \texttt{rmr} command to describe the state of the rModem. Currently the setup of the state register is:


diagram

The information of the state register can be extracted for use by \texttt{Sender.extract()}.

3.1.2 Packet Headers, Subheaders and Trailers

Incoming packets are always preceded by a header and a trailer. This makes it so the Sender class can extract packets easily. Following the trailer for most packets is a three or four letter name for the result being sent from the rModem. Most of the time these subheaders are the same name as the command that prompted the rModem to send that specific response. After the subheader there is usually information following giving some information that the GUI can use. The information that is returned for most commands is described in the commands section.

3.1.3 Sending Data – Text, Files, Pictures

Data Packets are packets that contain text, pieces of a file, or pieces of a picture. These packets are special because most of the time files and pictures will be longer than one packet themselves. These packets are the ones that actually get sent around to different destinations in the network. The other packets are just commands that go from the GUI to the modem.

All data packets have the same headers and trailers as the other packets. Their subheader is the keyword \texttt{rms}. A two digit destination node of the packet follows the \texttt{rms} keyword. This specifies to the rModem what node the packet will be sent to on the network.

Following this is the data packet structure header and the data itself. The packet structure plus the data is what constitutes the packet size as specified by the \texttt{rmp} command. The packet structure conveys information of the following in this exact order, session number, packet number, data type, data size, user parameter 1, and user parameter 2. The remaining bytes are used for the actual data.

The sizes of the packet structure are fully customizable by the user with the GUI. The parameters in the packet structure can be of any length including zero. For example, in the test setup the session number is an 8-bit field, the packet number is an 8-bit field,
the data type is a 4-bit field and the data size is a 16-bit field. The method Sender.combine() is used to make a packet structure header and Sender.extract() gets the values out of a packet structure header. Every time the values are changed, they are written to file and subsequent runs of the application use those values.

These configurable parameters are sizes, not values. They specify, for example, that the field in the packet structure header is 8-bits. The backend of the GUI takes care of what that value is. A session number is the number of distinct sessions the GUI has had with the rModem. A file that was 100 packets counts as a session and so does a one packet command of rmr, for example. The packet number is the packet in that session, using zero based indexing. So if a file is being sent and it is the 46th packet, 45 will be put in the field for packet number. The data type field tells whether the data is a text message (value of 1), file (value of 3), or picture (value of 5). The data size tells how many bits are coming in the data. This is how the receiver of data can tell when all the packets have been received. The user parameters are currently not in use, and have sizes of zero.

Sometimes the rModem cannot process all the packets for a text message, file, or picture. In this event the modem sends a busy signal to the GUI specifying the last packet it successfully received. When this happens the GUI loops sending packets starting at the next one the rModem needs through the last packet. When all the packets have been received by the rModem, the rModem sends an OK signal and the GUI stops looping through sending the packets.

3.1.4 Plotting

Plotting channel response is an option available to users that have MATLAB. When the rModem is set to one of the three plotting modes, the plots are saved to disk so they can be analyzed and graphed. A separate Java program must be launched to plots these automatically. This program must be launched in MATLAB. This is accomplished by using a class called MatlabControl.

3.1.5 Receiving Files and Pictures

Files and pictures are received in packets and saved to the disk with information about their session number and individual packet number. They are not kept in memory because if there’s a crash during a transfer the packets that made it through will be safely stored on the disk. Also, the packets could get out of order, so packets are temporarily stored in the hard disk. After all the packets for a file are received, a thread is spawned to piece it together and save it.

3.1.6 Raw Packets to GUI Display

When packets are extracted by the Sender class, they are placed in a queue for analysis. Each packet is analyzed in order and actions are taken with it. In the case of a
file, picture or plot, the data is written to disk as explained earlier. For other commands variables are changed in the instance of the Sender class. Some things only contain information that is useful to the user. In any case, when packets come in, information needs to be displayed in the terminal window in the GUI. A List of display Strings are kept in Sender and whenever a packet comes in, the RModemInterface is notified so it can dequeue those Strings in the terminal window. Also, with each notification, RModemInterface updates information on the GUI of the state register, the default recipient ID, the node ID and the packet structure.

3.1.7 Command Logging

See CommandLogger in section 3.3.2, Java Classes.

3.1.8 Terminal Logging

See TerminalLogger in section 3.3.2, Java Classes.

3.1.9 Byte Conversions

When receiving data from the rModem on clock ticks, clock frequency, and all three of the plots it is necessary to convert the incoming byte stream into 32-bit floats and integers. I used a free software package that performed these operations [2].

3.2 GUI Components

3.2.1 Terminal Window

The terminal window displays commands that were run by the user and the results that are received. Under the terminal window is a command window where the commands are typed and executed upon pressing the return key. The terminal window displays the exact command that was typed into the command window, messages when a file is being sent, and the results the GUI gets form the rModem. The raw data that comes from the modem is discussed in the next section.

3.2.2 Byte Input Window

Above the terminal window, there is a byte input window. This window displays every single byte that is received by the GUI. These bytes can be displayed in ASCII or hexadecimal form. The user can select the form by checking or unchecking a text box on
the GUI. This window is good for debugging purposes because even if input from the rModem is not processed for some reason, the bytes that came in will be displayed on the window. Right clicking this window inserts a marker at the end of the bytes received at that time so the user can easily tell which bytes are new.

3.2.3 Packet Structure Fields

The user of the rModem can specify how he wants the packet structure of the data he sends to look. He can change this by changing the values in the packet structure entry fields on the bottom of the GUI and pressing the Update button. The update button also sends the command to rModem to check what size packet it expects. It compares this expectation to the value inputted by the user and displays an alert if they do not match.

3.2.4 Batch Mode Button

To run a batch file the user can click on the batch file button. He will be presented with a file chooser where he can select a file from the computer. After the selection of a file, he will be prompted on how many times he wants to run the batch file. A window will open showing the commands being run with a stop button, so the user can stop the batch run.

3.2.5 Send File Button

To send a file from the GUI the Send File button is pressed and the user is presented with a file chooser where he can specify which file he wants to send. The user can click the stop sending button to stop a file transfer.

3.2.6 Update Button

The Update Button sends three commands instructing the rModem to return the state register, the default recipient ID and the node ID. These commands could be typed in manually in the command input window, but a quick update button executing all three makes it more convenient. The recipient ID and node ID are shown in a field on the bottom left corner of the GUI.

3.2.7 Text Message Window

The text message window functions like the terminal window except it displays outgoing and incoming text messages. There is a small text input window right below it where text messages are entered to be sent.
3.2.8 Packet Information Fields

On the top of the GUI there are six fields specifying the status of sending and receiving text messages, files, and pictures. The three fields on the left specify information about sessions sent from the GUI to the rModem. When a session is being sent a field is updated displaying the total number of packets in the file that is being sent plus the current packet that is being sent. When a file is being sent to the rModem, the Stop Sending button becomes enabled, and pressing it will cease transmission of the file.

The fields on the right serve the same purpose, but for incoming files. The session number that is reported by the rModem in the packet structure is displayed in the first field on the right. The other two fields display the number of total packets for the current transmission as well as the last packet received by the GUI from the rModem.

![Figure 1: This is the layout of the GUI. The text message window is on the right. The byte input window is on the upper left and the terminal window in on lower left.](image)

3.3 Java Class Structure

3.3.1 Threads

The rModem interface is made up of three main threads. The simplest Thread is named Recv. This thread is always watching the serial port for incoming bytes, adding each one to a buffer (specifically a Java StringBuffer). The Sender class checks the buffer of the Recv class frequently to determine if a new packet was added. The RModemInterface class observes the Recv class and on at each change to the buffer, adds
the new byte to the incoming byte window. The RModemInterface class accepts input from the user to be processed by the Sender class. It also observes the Sender class for new packets that can be displayed in the terminal window.

### 3.3.2 Java Classes

**Recv**

This class extends the Java class Observable and implements the Java interface Runnable. It is meant to be spawned as a thread. When the thread is spawned it waits for incoming bytes on the channel and when receiving one adds it to a StringBuffer. It then notifies its observers of the change.

**Sender**

This class is the main backend of the rModem interface. It implements the Java Runnable interface and is meant to be spawned as a thread. As a thread, it is constantly looking for packets to be processed by checking for the header and trailer of a packet. This class does all the processing of incoming packets. It processes the packets and notifies its observer, RModemInterface, of its changes. Input from RModemInterface is sent to Sender and processed by it. Commands that are entered into the GUI are sent to the Sender class from the RModemInterface class to be sent over the channel to the modem. Section 3.1, Functionality, discusses the backend of the system that is handled by the Sender class.

**RModemInterface**

This class inherits from the Java JFrame class. It observes both the Recv and Sender class for incoming data sent from the rModem over the channel. This class makes up the GUI. Section 3.2, GUI Components, describes all the components of the GUI. The Sender and Recv classes are oblivious to the RModemInterface class as consistent with a model-view-controller structure.

**CommandLogger**

This class creates a RandomAccessFile that will be the command log. Its only method is append, where it takes in a String that is the command and appends it with a timestamp and a delay from the time the user entered the last command.
TerminalLogger

This class, like CommandLogger, creates a RandomAccessFile when instantiated and its method append appends the input command Strings to it. It does not use timestamps and looks just like the terminal window.

MatlabPlottingWrapper

MatlabPlottingWrapper has one method, called start(), that spawns a plotter thread. MatlabPlottingWrapper.start() is meant to be called from MATLAB. To start this, it must be added to the Java path in MATLAB and called in MATLAB. An example of how to use this is as follows:

> mpw = MatlabPlottingWrapper
> mpw.start

After this, a small Java window will be opened that when closed will exit MATLAB and the MatlabPlottingWrapper program. More information on how to use Java in MATLAB can be found in the sources [3] and [4] in this References section of this document.

Plotter

This class does the work of plotting the graphs. This method implements Runnable and is meant to be spawned as a thread. This thread watches the directory where plots are placed. It stores the contents of the last time it looked at the directory and compares it to the new contents. If it finds a new file, it will graph it using MATLAB commands. Plotter is instantiated and called in the RMPlotter method start().

FileConstructor

FileConstructor is instantiated with parameters specifying the name of the file it will be constructing. It is also given how many packets make up the file and whether the file is a picture. File constructor implements Runnable and is meant to be spawned as a thread. This thread reads in every individual packet into a buffer. Up until a special three character sequence in the buffer is the title of the file. The thread writes a file with that title containing the file data, which is the rest of the buffer. If the file is specified to be a picture, it opens a new ImagePane displaying the picture.

ImagePane
ImagePane is instantiated with a byte array of data that will be a picture. ImagePane displays that picture.

PacketStructureField

PacketStructureField is a class to facilitate easy manipulation of packet structure input fields on the GUI.

RModemInterface Inner Classes

All the Java Swing listeners are inner classes of RModemInterface. This makes for less files and gives the inner classes direct use of RModemInterfaces’ methods and fields.

4 Commands

4.1 Sending Text Messages, Files and Pictures

Text messages, files and pictures are all transmitted by the same means. The command \texttt{rmsnn}, where \texttt{nn} is the default destination node, followed by a series of packet information bits and then the data is the actual data being sent to the modem. Since typing the information bits is not possible to enter in an ASCII terminal, and sending multiple packets would be tedious, a user of the GUI should never have to type the command letters \texttt{rms}.

Text messages are sent by typing a message into the text message input window. This message will be routed to the default recipient ID node, which can be changed as described below.

Files and pictures are transmitted exactly the same to the rModem. The “send file” button opens a file chooser that lets the user select which file he wants to send. The rest is taken care of by the GUI. The backend of the interface splits the file into packets and sends them using the \texttt{rms} command to the default recipient node. The interface takes care of setting the session number, packet number, parameter size and the section of the data for each particular packet. It also takes care of the busy signals sent by the modem and the OK signal sent after the modem has received all of the packets.

4.2 Verbose Level

The verbose level specifies how results are sent from the rModem to the GUI.

\texttt{rmvn}, where \texttt{n} is the verbose level. The current valid values of \texttt{n} are:
0: quiet mode
1: terminal mode
2: GUI mode

4.3 rModem State

Entering this command returns the state register of the rModem. The state register gives information on the state and the values of parts of the rModem. It returns the default recipient ID, the operation mode, and the verbose level of the rModem.

```
rmr
```
returns the state register of the rModem.

4.4 Default Recipient ID

This command sets or returns, depending on its usage, the recipient ID that text messages, files and pictures will be sent to.

```
rmd
rmdnn, where nn is the destination node number, sets the modem’s recipient ID.
```

4.5 rModem Packet Size

```
rmp
```
returns the expected packet size of the rModem.

4.6 rModem NodeID

This command sets or returns, depending on its usage, the node ID of the modem.

```
rmd
rmdnn, where nn is the destination node number, sets the modem’s recipient ID.
```

4.7 Clock Tick Value

```
rmct
```
returns a 32-bit integer of the number of clock ticks since the rModem was powered on.

4.8 Clock Tick Frequency
4.9 Scatter Plot Mode

When set, before every received packet, the rModem will send a packet with 32-bit complex floating point numbers of the symbols at the receiver (after the equalizer, before the decoder) to the rModem GUI.

rmxn where n is 1 to turn the mode on, 0 to turn the mode off

4.10 Received Samples Mode

When set, before every received packet, the rModem will send a packet with 32-bit complex floating point numbers of the received samples to the rModem GUI.

rmqn where n is 1 to turn the mode on, 0 to turn the mode off

4.11 Channel Estimation Mode

When set, before every received packet, the rModem will send a packet with 32-bit floating point numbers of the channel estimate based on the preamble processing to the host.

rmfn where n is 1 to turn the mode on, 0 to turn the mode off

5 Conclusion

Much of the difficulty of this project arose from the use of many concurrent threads. These threads, while necessary, often made the code hard to debug. As features were added, sometimes the original thread structure was not the best choice considering the implementation of new features. Sending and receiving files correctly was the most difficult part in both the design and implementation of the interface. With the incoming buffer stored as a StringBuffer, only certain Java classes and methods would correctly account for non-printable characters that are necessary in the transfer of binary data. This became very frustrating at points and made for many bugs in the implementation. The protocol that was agreed on for sending files with the possibility that the rModem becomes busy is significantly better than the other possible protocols we looked at.

This rModem graphical user interface provides much of the functionality that will let people carry out experiments with the rModem. After half of the current functionality was completed, potential users of the GUI provided input on what else they would like to see implemented in the GUI. This led to the plotting of the channel response and other
plots being implemented as well as the text messaging interface. More can be done, however, to make experiments done with the rModem GUI even more effective. Currently the interface can receive and display pictures in real time. In the future it may be helpful to have streaming video, instead of static pictures, so that experiments can be even more real time than they are now. Also, the interface could be extended to be able to access lower level data in the modem as well as change its low level parameters.
6 References


