Examensarbete

Automated Telephony Testing

Examensarbete utfört i Kommunikationssystem
vid Tekniska högskolan i Linköping
av

Anders Hansén & Jacob Svensson

LiTH-ISY-EX-ET--08/0343--SE
Linköping 2008
Automated Telephony Testing

Examensarbete utfört i Kommunikationssystem vid Tekniska högskolan i Linköping
av

Anders Hansén & Jacob Svensson

LiTH-ISY-EX-ET--08/0343--SE

Handledare: Karl Fogdegård
Ericsson

Examinator: Mikael Olofsson
isy, Linköpings universitet

Linköping, 19 August, 2008
Automated Telephony Testing

Testing is a crucial part when developing electronics. One way to lower the costs and increase the efficiency is to avoid human interaction. This can be done with automated testing. The purpose with this thesis is to look into the possibility of automating telephony and facsimile tests with "off the shelf components" such as modems and standard computers. The proposed solution was put together and the needed software was developed, using Java. When testing electronics it is of most importance that the hardware and software carrying out the tests are reliable. To be able to ensure this, the software has been thoroughly tested, and the different error sources discussed. The biggest cause to the problems found was that the modems weren’t reliable. A general work around is presented, implemented and tested.
Abstract

Testing is a crucial part when developing electronics. One way to lower the costs and increase the efficiency is to avoid human interaction. This can be done with automated testing. The purpose with this thesis is to look into the possibility of automating telephony and facsimile tests with "off the shelf components" such as modems and standard computers. The proposed solution was put together and the needed software was developed, using Java. When testing electronics it is of most importance that the hardware and software carrying out the tests are reliable. To be able to ensure this, the software has been thoroughly tested, and the different error sources discussed. The biggest cause to the problems found was that the modems weren’t reliable. A general work around is presented, implemented and tested.
Acknowledgments

*If you wish to make an apple pie from scratch, you must first invent the universe.*

Carl Sagan

With that said, we would like to thank Ericsson for giving us the opportunity to carry out our thesis work on an interesting subject in an inspiring environment. We would especially want to thank our supervisors Karl Fogdegård and Hanna Karlström and all the other friendly and helpful peoples at the CPE PS department at Ericsson in Linköping. Thanks for your help, and for answering all our questions again and again. We also want to thank our examiner Mikael Olofsson at ISY for your help with the report and advices during the work. Last, but not least, we would like to thank our friends and families for your support, proofreading and for listening to endless technical monologs over and over again.

Anders Hansén & Jacob Svensson
Linköping, Juni 2008
# Contents

1 Introduction  
1.1 Background ................................................... 1  
1.2 Purpose .................................................... 1  
1.3 Methodology ................................................ 2  

2 Technical Background  
2.1 Telephony .................................................... 3  
  2.1.1 Public Switched Telephone Network ..................... 3  
  2.1.2 Dual-Tone Multi-Frequency ............................... 4  
  2.1.3 Caller Identification ................................... 4  
2.2 Facsimile ..................................................... 5  
  2.2.1 The Transmission Protocol -T.30 ........................... 5  
2.3 Modem ......................................................... 7  
  2.3.1 Hayes Command Set ...................................... 7  
  2.3.2 AT Commands ............................................. 7  
  2.3.3 Modes ................................................... 7  
2.4 Reverse Engineering ......................................... 8
### Contents

3 Implementation 9

3.1 Current Test Environment 9

3.2 Proposed Test Setup 10

3.2.1 Modem 10

3.2.2 Platform 11

3.2.3 RXTX 11

3.3 Automated Phone Call 11

3.3.1 Common Tasks for Transmitter and Receiver 12

3.3.2 Transmitter 12

3.3.3 Receiver 13

3.4 Transferring and Verifying a Fax Document 14

3.4.1 Transmitting and Receiving 15

3.4.2 Verifying the PSTN Connection 16

3.5 Configuration 16

3.6 Errors 16

4 Discussion 19

4.1 Conclusion 19

4.2 Design Decisions 20

4.3 Future Development 20

5 Users Guide 21

5.1 Setting up the Program 21

5.2 Configuring the Program 21

5.3 Executing the Program 22

5.4 Results 23

5.4.1 Phone Test Output 23

5.4.2 Facsimile Test Output 24

Bibliography 27
List of Figures

2.1 Overview of the T.30 transmission protocol .................. 5

3.1 The system environment .................................. 10

3.2 A block diagram describing the common tasks .............. 12

3.3 A block diagram describing the transmitter ................. 13

3.4 A block diagram describing the receiver ...................... 13

3.5 A block diagram describing the fax implementation ........ 15
List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTR</td>
<td>Confirmation to Receive</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>DCS</td>
<td>Digital Command Signal</td>
</tr>
<tr>
<td>DIS</td>
<td>Digital Identification Signal</td>
</tr>
<tr>
<td>DTC</td>
<td>Digital Transmit Command</td>
</tr>
<tr>
<td>DTMF</td>
<td>Dual-tone multi-frequency</td>
</tr>
<tr>
<td>DTR</td>
<td>Data Terminal Ready</td>
</tr>
<tr>
<td>EOP</td>
<td>End of Procedure</td>
</tr>
<tr>
<td>FSK</td>
<td>Frequency-shift keying</td>
</tr>
<tr>
<td>FTT</td>
<td>Failure to Train</td>
</tr>
<tr>
<td>FWT</td>
<td>Fixed Wireless Terminal</td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunication Union</td>
</tr>
<tr>
<td>MCF</td>
<td>Message Confirmation</td>
</tr>
<tr>
<td>PSTN</td>
<td>Public Switched Telephone Network</td>
</tr>
<tr>
<td>RTC</td>
<td>Return to Control</td>
</tr>
<tr>
<td>RTS</td>
<td>Request to Send</td>
</tr>
<tr>
<td>TCF</td>
<td>Training Check Facilities</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background

Testing is a crucial part when developing electronics. Companies such as Ericsson spend huge amount of resources on testing. An important way to lower the costs and increase the efficiency is to avoid human interaction. This makes automatization a very interesting area and a common task in testing. Ericsson has developed a product named W25. It is a fixed wireless terminal that provides broadband, voice and fax in a convenient way through 3G networks. The target groups of the product are ordinary homes and small offices. One of the main benefits with these kinds of products is that they are able to provide internet access and voice communication without the need of fixed infrastructure. The background for this thesis is that Ericsson, during the development of their fixed wireless terminals, has automated big parts of the tests that are carried out regularly. However, the telephony and the fax is still tested outside the automatic test environment. Their main reason for this is that the currently available equipment used for testing telephony is expensive and therefore, among other things, not suited for the automated test environment used at Ericsson.

1.2 Purpose

The objective of this thesis is to develop and implement an automated telephony test tool to a low cost. To achieve this, the automated telephony test environment will be based on "off the shelf components“, such as Data/Fax/Voice modems and a standard computer. On the software side, programs will be written in Sun’s Java and executed in Linux, this to fit the existing test framework at Ericsson.
1.3 Methodology

The goal of this thesis is to reach the objective stated in the purpose. To reach this goal, a short feasibility study will be carried out to be able to outline the problem. This part will also be used to decide which hardware to use, and to assemble the test bench. The main part of this work will be to write the software needed to execute the tests, and analyze the results.
Chapter 2

Technical Background

This chapter will give the reader a brief view of the terminology and technology’s discussed later in this thesis. It will give a short background to a number of areas, covering different aspects of telephony, facsimiles and modems. It will also give a short explanation to the term reverse engineering.

2.1 Telephony

Few things have had such big impact on the way people communicate with each other as the telephone. Things this widespread will eventually evolve a variety of standards such as the public switched telephone network, Dual-tone multifrequency and Caller identification.

2.1.1 Public Switched Telephone Network

To give a simplified picture, a telephone call is a connection over a telephone network between two parts, the calling part and the called part. The end parts can for example be two telephones. In the simplest case, the phone only consists of a speaker, a microphone, and a switch called hook switch. The switch is used for connecting and disconnecting the phone from the telephone network. A modern phone almost always has a touch-tone keypad and frequency generator but this is actually not really needed to make a call. Almost all telephone switches still recognize pulse dialing, which means that you can dial simply by tapping the hook switch [2].
There are different standards around the globe, but in general it works as simple as that one click dials a 1, two clicks a 2 and so on. If a phone from the early twentieth century is connected to the telephone network, it will actually work just fine. The standard way to make a call today is tone dialing, which uses Dual-tone multi-frequency (DTMF) to signal what number to call. For two parts to be able to call each other there is need for some kind of network in between. The biggest network is the world wide network called the Public Switched Telephone Network, or PSTN. It’s a network built up by alot of smaller connected networks, and makes it possible to make a phone call over long distances. As well as being big and spanning across the globe, a telephone network can be very small and cover just a few phones, for example an internal network at a company. To make a connection between two nodes, a telephone number is needed. The number format is specified in the ITU-T (International Telecommunication Union) recommendation E.164 [5]. The recommendation says that a number can have a maximum of 15 digits, including for example country code, identification code and subscriber number.

### 2.1.2 Dual-Tone Multi-Frequency

Dual-tone multi-frequency is used for sending information over telephony lines or radio connections. It was developed by Bell Labs around 1960, but has its roots in a technique called Multi-Frequency, that was used by AT&T in the 1950s to direct call using in-band signaling. DTMF is two tones that are sent simultaneously, and then detected on the other side. The reason for using two tones instead of one is the increased fail tolerance. The frequencies are chosen so that no frequency is a multiple of another and a sum of frequencies does not equal other frequencies. The frequencies used in Sweden are described in figure 2.1 [4].

<table>
<thead>
<tr>
<th></th>
<th>1209 Hz</th>
<th>1336 Hz</th>
<th>1477 Hz</th>
<th>1633 Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>697 Hz</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>770 Hz</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>B</td>
</tr>
<tr>
<td>852 Hz</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>C</td>
</tr>
<tr>
<td>941 Hz</td>
<td>*</td>
<td>0</td>
<td>#</td>
<td>D</td>
</tr>
</tbody>
</table>

**Table 2.1.** The frequencies used in the keypad of a phone

### 2.1.3 Caller Identification

Calling number identification is a service in the telephony networks, used to transmit the callers telephone number to the answering side. It can be sent in different ways and at different times, depending on localization. The most common standard used both in USA and China, is the 1200 baud Bell 202 tone modulation
and is sent out between the first and second ring [1]. One drawback is that if the
call is answered before the second ring, the callers identification will not be sent.
The standard used in Sweden, and at Ericsson, uses DTMF tones to transmitt the
phone number. The tones are transmitted just before the first ring.

2.2 Facsimile

Facsimile or Fax is a technology used for sending documents over the telephone
line. To send a facsimile a fax machine is needed. This can e.g. be a modem
that supports fax. Although the fax protocol is out-of-date and the internet-based
alternatives are taking over, the fax protocol is still, in large scale, being used.
Especially when sending sensitive documents.

When sending and receiving fax via modems the developer can choose from
several different command sets e.g. Service Class 1 and Class 2. Class 1 provides
low level compatibility but is fairly complicated and the user is expected to have high
knowledge about the fax protocols [8]. The other protocol (Class 2) lets the modem
take care of T.30/T.4 protocol procedures, where T.4 is the compression method
and T.30 is the necessary transmission protocol procedures for fax transmission
over the telephone line [9]. T.30 is a recommendation created by ITU.

2.2.1 The Transmission Protocol -T.30

The T.30 protocol is divided into five different phases. As illustrated in figure 2.1
each phase represents a stage that has to be completed if the transmission is to be
seen as successful.

![Figure 2.1. Overview of the T.30 transmission protocol](image)

**Phase A - Establishing Call:** In this phase the caller (transmitter) goes
off-hook and tries to detect the dial tone, if it’s detected the caller dials a num-
ber free of choice and waits for the receiver to answer. If the call is answered the
transmitter instantly sends a fax announce tone. The receiver awaits the incoming
call and if so, answers it with a fax answer tone. The announce tone is a generated
1.1 kHz tone with $\frac{1}{2}$ second duration that is repeated every three seconds. The
answers tone is a 2.1 kHz tone with the duration of three seconds.
Phase B - Pre-Message Procedure: In Phase B the transmitter and the receiver carries out a handshake procedure. This is done in 300bps and in FSK (Frequency-shift keying), which is a frequency modulation scheme. The receiver starts the handshake when it sends a DIS (Digital Identification Signal), which is a signal containing the receiver’s capabilities. The transmitter answers with DCS (Digital Command Signal), with the desired capabilities.

Phase C - Message Transmission: To prevent errors and image corruption Phase C starts with a training session. The session begins with the transmitter sending a TCF (Training Check Facilities) data pattern. It’s sent with a data rate between 2.4kbps and 14.4kbps. If an error occurs the receiver will return FTT (Failure to Train), which will restart the training. If the training went smooth and without problems, the receiver sends back CFR (Confirmation to Receive) and the real data image will be transmitted.

Phase D - Post-message Procedure: When the entire data image has been transferred the transmitter sends a RTC (Return to Control) pattern to the receiver. When doing this both the caller and the called will switch back to 300bps, and End of Procedure (EOP) is sent from the transmitting side. The receiver confirms by sending Message Confirmation (MCF) pattern.

Phase E - Call Release: The transmitter sends a Disconnect (DNC) signal to the other terminal, and both terminals hang up.
2.3 Modem

Modem, whose names originate from modulator-demodulator, is a device used for modulating an analog signal to digital information and vice versa. A typical area of use is to send and receive digital information over the ordinary telephone network.

2.3.1 Hayes Command Set

Since some time in the 80s the standard way to communicate with modems is AT Command Set, introduced by the modem manufacture Hayes 1977 in their product, the Smartmodem [3]. This standard was later widely adapted by other manufactures, and in the second half of the 80s, nearly all consumer modems where shipped with some kind of AT commands support. Years later, 1992, it becomes a Telecommunications Industry Association standard named TIA/EIA-602.

2.3.2 AT Commands

The modem is controlled by commands sent to it via the serial port when it’s in command mode. Those are called AT commands, where AT stands for Attention. See the reference guide for the modems for more information about this [7].

2.3.3 Modes

The smart thing Hayes did was to introduce two different modes in the modem, data mode and command mode. Before this, a separate dialer was needed to give the modem directions, e.g. which number to call. Now, all that’s needed is to switch between data and command mode. In data mode, the modem handles everything as data. Everything passed from the computer is forwarded to a remote modem, and everything received from the remote modem is forwarded to the computer.

In command mode, data from the local computer is interpreted as commands to the modem. To switch from command to data mode is trivial, just issue a command. The other way around, however, is a bit trickier. The original way is to send +++ (three plus) in the data stream, followed by a second of silence. This puts the modem in online command mode, meaning command mode but without loosing the connection. An other rather common way to change mode is using the DTR (Data Terminal Ready) pin on the serial port. In addition to those two, there are at least two other, more or less standardized modes which are going to
be used in this thesis. The first of those is voice mode, used for transmitting and receiving voice recordings over the phone line. To use this mode a voice modem has to be used. This means that it has built-in capability of handling audio. The second mode used is fax mode, which is used for sending and receiving facsimiles. [7]

### 2.4 Reverse Engineering

One important way of understanding the technological principle and working of a device or software is to use reverse engineering to some extend. It can be described as 'going backwards through the development cycle' [10]. For this thesis, basic reverse engineering has been used to examine the communication between the modem and the computer. For this, a Microsoft software named Portmon\(^1\) has been utilized. The straightforward function of this program is that it has the ability to display and log all activity on a specific serial port. This ability has been exploited by using an existing software to make a call or to send a facsimile, and examine the communication between the computer and the modem at the time. The software used for making the calls and sending the facsimiles are mainly components from Microsoft Windows 2000, as e.g. the built in fax service. From the data received this way, the specific port settings and AT commands are being extracted and examined, to get an understanding of the proceedings. These understandings are later used when in the implementation phase, writing a set of software that achieve the same things. These softwares are written from scratch and the purpose of the reverse engineering is to gain understanding.

Chapter 3

Implementation

In this chapter the hardware and software implementation will be presented in detail. The hardware consists of “off the shelf components” such as modems and an ordinary computer, and the software is developed in Java. The implementation consists of two parts, telephony and facsimile. Both parts are based on the same basic steps, send, receive and compare the sent with the received. This chapter also gives a brief explanation of how the tests are carried out today.

3.1 Current Test Environment

The short story is that there is no current test environment. The long story is that there is some kind of modem based implementation, but it is very undeveloped and shaky, and therefore hardly used. This means that the tests are carried out either manually or not at all. Lots of tests are also carried out with a commercial generic telephony test system. Those tests include stress testing, sound and line quality and more. The problem with this solution, as stated before, is that it is way too expensive and complex to use for regressions tests and alike.
3.2 Proposed Test Setup

The environment consists of an ordinary standalone PC that’s connected to several modems. The server is also accessible via telnet from any terminal, so that the test can be executed from a remote connection. The interface between the computer and modem is a standard serial (RS-232) connection. In turn, one of the modems is connected directly to the FWT through a phone cord, and the other through the PSTN and mobile network. Figure 3.1 gives a basic picture of how the system is designed.

3.2.1 Modem

The key component in this project is the MT5656ZDXV, a 56 Kbps voice and fax modem manufactured by MultiTech. The modems are used for sending and receiving both calls and facsimiles, creating, sending, receiving and recognizing signals and data. Those modems are designed for use over ordinary fixed telephone...
3.3 Automated Phone Call

lines and not for 3G and wireless. Even if this may sound like a problem or drawback, this is just what we want. If it is possible to successfully establish a connection, and transfer sound or facsimiles, using only modems, one will know that the unit under test works as supposed. There are however some problems with those modems. They aren’t totally reliable, which make error identification and handling of outermost importance.

3.2.2 Platform

As platform for the test setup, an ordinary PC is used. The lowest hardware requirements aren’t tested, but a computer on 500 MHz and 512 Mb ram is more than enough even when a graphical environment is running. When an USB to RS232 converter is used for connecting the modems to the computer, support for high speed USB (USB 2.0) is highly recommended.

As operating system, most platforms capable of running java and the rxtx library will work, e.g. Windows XP, Windows Vista, Linux, Mac OSX and more. That said, only Linux will be used and discussed in this thesis. The main reason for choosing Linux is the ability to control everything remote via a terminal and for simple integration in the existing test framework. Ubuntu¹ is chosen as distribution. It’s very easy to setup, it makes java and rxtx available with a simple command, it has good documentation and it’s already used at Ericsson to some extend. If another distribution is preferred, using that instead should be pretty straight forward.

3.2.3 RXTX

RXTX is a java library that provides, among other things, serial communication. It’s based on, and more or less compatible with, Sun’s Java Communications API. It has support for Linux, Windows, Solaris, and Mac OSX and is released under the GNU Lesser General Public License. [6]

3.3 Automated Phone Call

To ensure that the phone line is fully functional, a phone call is carried out by the modems. The flow graphs, combined with the following sections of text are meant to give a description on how such a phone call is conducted.

¹Url: http://www.ubuntu.com/
3.3.1 Common Tasks for Transmitter and Receiver

There are some parts that are common for both the receiver and the transmitter. This includes the initiation in the beginning, and the cleanup in the end. As seen in figure 3.2, both the beginning and the end are common.

![Figure 3.2. A block diagram describing the common tasks](image)

The first thing both instances have to do is to initialize the serial port. To decide what port to use, a configuration file, which is discussed more later on, is used. The chosen port is initialized to work at a speed of 115200 bit/s, a data size of 8 bits, one stop bit, no parity and RTS/CTS (Request to Send / Clear to Send) as flow control.

The second thing common to both instances is the modem initialization. During the initialization, a lot of parameters are set. The different settings that are made can be divided into categories. Firstly, various ”cosmetic“ changes are made, like turning off the speaker. Secondly, things that effect the way the modem communicates with the computer is done. This include turning off echoing of messages, formatting of result codes, formatting of caller id and similar. The third category is settings that affect the communications between modems. This includes loading factory settings for the correct region, data carrier settings and flow control.

At last, after the call is completed and the specific tasks in the module are executed, hang up the line and close the serial port.

3.3.2 Transmitter

The transmitter is the part making the call, and sending the tones to be recognized. The process, as described later, can be seen in figure 3.3.

The first thing that happens is that the modem is set to voice mode, it then dials out and waits for the second part to answer. When the other side answers, the recording process in the modem is started. The recoding process is needed to be able to fetch sound from the phone line. When the connection is established, the transmitter acts as receiver for short while, and listens for the DTMF tone 'one' that the answering side is supposed to send out. The main reason for this handshake procedure is to eliminate the error that can occur when the modem some times, wrongfully, believes that someone answered when that’s not the case. This is most likely related to that the modem misunderstands what’s happening
when no one answers for a long time, and the ringback tone disappears. It is also a simple way to make sure the right number is dialed.

After a successful handshake and when the called part is ready to receive, it’s time to start the playback process. This is done to be able to send out sounds. When the modem is ready, the task of sending the predefined DTMF tones takes place. Tones can be randomly chosen, but due to some problems with detecting the length of a tone, the best result is achieved by avoiding that two subsequent tones are alike. When the tones are sent, the connection is closed.

### 3.3.3 Receiver

The receiver is the part answering the call, conducting the handshake and then listening for the tones. The process, more deeply described later, can be seen in figure 3.4.

When everything is initiated as described before, the receiver idles and waits for an incoming call. This is the standard mode for the program, if it isn’t getting any specific directions, e.g. make a call.
Upon incoming call, the receiver answers it in voice mode. To properly connect and verify that the call is established the DTMF tone for the digit "one" should be transmitted. The main reason for this is for the caller to be able to confirm that the connection is correct, but it also has the positive bieffekt that it confirms that the communication works both ways. When the handshake is done and the recording process is started the program idles and waits for events to occur. The events are identified, and if needed, the appropriate action is taken. The only events that are interesting at this moment are the incoming DTMF tones, and the event signaling that the conversation is over and its time to compute the result. If the event is a DTMF tone, it is stored in the designated list until later. When the event signaling that the conversation is over, the received DTMF tones are compared with the expected ones. If they are the same, the test passed, otherwise it failed.

### 3.4 Transferring and Verifying a Fax Document

The facsimile software is in many ways similar to the phone implementation. The transmission can be divided into three major categories, the common, the transmitting and the receiving. The each category and its tasks are specified in figure 3.1. Most of the transmitter and receiver tasks are based on the T.30 protocol.

<table>
<thead>
<tr>
<th>The Common</th>
<th>Open Serial port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initiate modem</td>
</tr>
<tr>
<td></td>
<td>Clean up</td>
</tr>
<tr>
<td>The transmitter</td>
<td>Establish Connection</td>
</tr>
<tr>
<td></td>
<td>Send message</td>
</tr>
<tr>
<td></td>
<td>Close Connection</td>
</tr>
<tr>
<td>The receiver</td>
<td>Establish Connection</td>
</tr>
<tr>
<td></td>
<td>Receive Message</td>
</tr>
<tr>
<td></td>
<td>Close connection</td>
</tr>
<tr>
<td></td>
<td>Verify message</td>
</tr>
</tbody>
</table>

**Table 3.1.** The three categories and their related tasks.

Both the transmitter and receiver categories contain establish and close connection. These two tasks recalls each other, but they are executed with different commands and that’s why their not in the common section.

A few of the similarities with the phone call implementation is how initializing of the serial port and modem is carried out. Even the clean up procedure is reminiscent of how the phone part is implemented. The biggest difference is that
the modem handles, when sending class 2 fax, e.g. the handshake procedure instead of leaving it to the developer.

Figure 3.5 provides an overall picture of how the transfer of facsimile is implemented. During transmission the modems are set to operate in service class 2. As mentioned earlier in 2.2 the class 2 command set is a further developed version of service class 1. The main difference is that class 2 let’s the modem handle most of the T.30 procedures.

### 3.4.1 Transmitting and Receiving

Both the transmitter and receiver categories are implemented in the same library but as separate modules. Each module contains several functions for sending, receiving, converting and verifying facsimile. With some minor modifications, both modules can be executed as a standalone and without dependences from the other modules.

During modem initialization, the user sets the local identification strings, which is exchanged during Phase B. These can be used to verify that the incoming call comes from the right machine. After the connection has been established, a sequence with hexadecimal digits\(^2\) is embedded into a pre-existing fax image and then sent over the channel as bytes. When the remote connection receives the image, all the bytes have been reversed by the modem. Figure 3.2 gives two simple examples of how the bytes are reversed.

<table>
<thead>
<tr>
<th>Sent</th>
<th>Received</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111 0000</td>
<td>0000 1111</td>
</tr>
<tr>
<td>0001 0001</td>
<td>1000 1000</td>
</tr>
</tbody>
</table>

**Table 3.2.** Example of how the bytes are reversed.

This means that the recipient must reverse all bytes before they can be converted back to hexadecimal digits and stored.

When the entire fax image is sent the transmitter signals the receiver and the connection is terminated.

\(^2\)The software converts the digits from hexadecimal to binary before sending.
3.4.2 Verifying the PSTN Connection

To verify that the PSTN connection is functional for transmitting facsimile and that the correct message has been received, the sent and received messages are compared. This is done by scanning through the received fax image and comparing it with the original sequence contained within the sent image. If the sequence is found the transfer was successful and the connection is functional.

3.5 Configuration

One very important aspect of a good software is the ability to configure it. To achieve this there are at least three different approaches. For someone used to a graphical environment, like Windows, changing the settings in graphical interface may be the most natural way. However, as this software is supposed to run on a headless server, with just a telnet connection, and without manual interference, a graphical user interface is not the way to go. A second way is to configure everything in a file before running the program. This approach is good for programs that need initial configuration, that doesn’t change very often, and for settings and conditions who are the same from time to time. A third way of configuration is by giving parameters to the program when starting it. This is often done via a command prompt, and is suited for programs started from a remote location.

For the program written during this thesis, a combination between option 2, configuration file, and 3, parameters, is chosen. A configuration file is used for the more permanent options, giving aliases to modems, which modem is connected to which serial port, and which modem has which telephone number. This information is saved as ordinary text, in a format that is simple to edit for the user.

For options that are changing between tests, arguments via the command prompt are used. Things that are set this way include the choice to send a fax or make a phone call, to decide from who and to who to call, the number of tests to run and the amount of output that is desired.

3.6 Errors

One big but important problem when engineering test tools is that there are two different kinds of errors, which should be handled totally different. The first kind is those that always are present when developing software, bugs. Those bugs can be present in both the software and the hardware. This is unwanted errors, that can be very hard to find and identify. The second kind of errors are those introduced
when the unit under test fails. The big problem is that it sometimes is far from trivial to distinguish between the two.

To give a concrete example from this thesis, the transmitter calls the receiver, and the receiver never answers. The possible cause can be a plethora of reasons, spanning from failing modems and problems with the telephone connection to bugs in the calling or answering procedures. On the other hand, it can as well be the unit under test malfunctioning.

This makes it crucial that the hardware and software is reliable. To achieve this, the software must be properly and thoroughly tested. Nevertheless, all advanced systems will contain bugs. Because of this good error handling is of the utmost importance.

In addition to the software, there are several sources for errors. Among them, the modems, the phone lines as well as the FWT, the unit under test. The software is implemented in such a way that by repeating the test several times and logging the results, the user will get a good picture of the outcome even if some iteration failed out of unexpected reasons. The software passes on the result, including the possible reasons in case of that error occurred, to the user. This gives the one writing the test the possibility to decide if the result is satisfying or not.

At a most basic level, even if only one out of several tests iterations succeeded, the test can be considered successfully. Even if only one of the test iteration is successful, it implies that the unit under test has worked as supposed at least one time. Of course, a higher success rate may indicate a more stable environment, and perhaps a more stable unit under test, but this is not necessarily the case.
Chapter 4

Discussion

4.1 Conclusion

The scope for this thesis was to examine if "off the shelf components" and own developed software could be used to create a test bench for Fixed Wireless Terminals. We have successfully showed that this is the case, but that there are a number of problems that affect the final result. The biggest cause of problems where that the modems didn’t behaved reliably. Typical errors that occurred could be that the modems didn’t hang up properly, that DTMF tones weren’t recognized correctly or that the modem initiation failed unexpectedly. We never really managed to figure out the underlying problem, but we believe that a possible contributing factor could be that we used the modems in a way that was not originally intended.

A second source of problem with the modems was the voice mode. As this isn’t a commonly used functionality in the modem, there is reason to believe that it isn’t as thoroughly tested as the rest. This manifests itself by the modems not acting consistently to commands, depending on earlier events. This kind of errors has been dealt with in the same manner as the one discussed earlier, by repeated test runs. This forced us to abandon the solution where we tried to handle and resolve each error separately. This solution was based on the idea that we could continue the test run even though an error had occurred. The solution we finally settled for in our attempts to handle the various problems was the one where we used multiple test runs to produce reliable results. This solution weren’t without drawbacks. Stressing the modems with multiple continuous test runs introduced new errors. However, the amounts of new errors where still few enough for the solution to be viable. A part intended to be included in the test environment but was left out is the different level 5 services in the telephone network. This is due to incompatibility between our modems and the telephone switches at Ericsson. Even so, large portions of the necessary code needed is written and implemented, but for obvious reasons, not tested.
4.2 Design Decisions

A central part of this thesis has been the possibility to generate and to recognize sounds. In the beginning, the idea was to generate and recognize tones manually, and to send and receive it from the modems by using sound cards. The sound card idea was discarded at a very early stage in the feasibility study, to benefit of the sound codec in the modems. The basic idea of generating and recognizing tones wasn’t disregarded, but the means to reach it where changed when we started to realize how much extra effort that where needed to develop the tools needed. Instead we chose the path where we used the modems ability to generate and recognize DTMF tones. This saved us a lot of work, and made it possible to solve the task at the intended time. The solution has two major drawbacks. The first is quite obvious; we loose the ability to measure the sound quality. The second problem where much more insidious, and a problem we found out about at a point where the implementation couldn’t be redone to reflect it. When sending DTMF tones via the FWT it detects them and converts the sound into data packets, which is then sent over the mobile network. This means that when you use the FWT to make the call, you don’t really send any sound at all; you just check that the data connection is functional.

4.3 Future Development

A proposal for future development is to abandon the use DTMF tones, and implement functions for generating and detecting tones. This solves two of the big problems mentioned in 4.2; it removes the problem with the FWT not sending the information as sound, and gives a good platform for implementing routines for measuring the sound quality.
Chapter 5

Users Guide

The guide assumes that a PC with a Linux environment and Sun’s Java already installed. The software is developed for Java SE 5, but should work in any later versions as well. The PC also needs two or more serial ports with MultiTech MT5656ZDXV modems connected.

5.1 Setting up the Program

The program relays on the RXTX library (Version 2.1-7r2), which can be downloaded from http://www.rxtx.org. The jar archive, named fwtTest.jar, can be placed anywhere but preferably in a dedicated folder. In this folder a subfolder lib must be created. Extract and copy the files RXTXcom.jar and /rxtx-2.1-7-bins-r2/Linux/i686-unknown-linux-gnu/librxtxSerial.so from rxtx package rxtx-2.1-7-bins-r2.zip. Place them in the subfolder lib.

File hierarchy:

/lib/RXTXcomm.jar
/lib/librxtxSerial.so
/fwttTest.jar
/fwttTest.conf

5.2 Configuring the Program

The program uses a configuration file named fwttTest.conf for its settings. This file must exist in the folder as the fwtTest.jar otherwise the program won’t start.
The syntax for \texttt{fwtTest.conf} is:

\begin{verbatim}
Identifier\hspace{1em}serial port\hspace{1em}phone number
Identifier\hspace{1em}serial port\hspace{1em}phone number
\end{verbatim}

Example,

\begin{verbatim}
A  /dev/ttyS0 70001
B  /dev/ttyUSB0 70002
\end{verbatim}

5.3 Executing the Program

The program is controlled by different command prompt arguments.

- \texttt{-fax}
  Sets the program in fax mode. Can’t be used together with \texttt{-phone}.

- \texttt{-phone},
  Sets the program in phone mode. Can’t be used together with \texttt{-fax}.

- \texttt{-from identifier},
  Selects which id to call from.

- \texttt{-to identifier},
  Selects which id to be called.

- \texttt{-i iterations},
  Number of iterations to run. The default is 1.

- \texttt{-debug level},
  The level of output.

  The phone has four levels,
  0, print only the results.
  1, print the result and possible errors.
  2, normal output, print AT commands and such
  3, prints all available information, for debug purposes.
5.4 Results

The fax has two levels, 0, print results and possible errors. 1, prints all available information, for debug purposes.

-h, Help section.

Example, 
# java -jar fwtTest.jar -phone -from A -to B -i 1 -debug 0

5.4 Results

The program output when executing in fax mode and phone mode are similar, but the error messages are different. Therefore, the output is presented in separate sections.

5.4.1 Phone Test Output

Possible outputs:

No error, No errors, the test was successful.

Config file missing, The configuration file, fwtTest.conf, is missing.

Serialport initialization failed, The serial ports failed to initialize.

Modem initialization failed, The modem failed to initialize.

Timeout, Something took too long time, most possibly an answer from the modem.

No answer, The caller got no answer.

No connect, The caller got an answer, but the connection failed.

Handshake failed,
The caller got an answer, and the call was connected, but the handshake failed.

**No dialtone,**
The caller has no dial tone.

**No carrier,**
There are no carrier on the line.

**Line busy,**
The line is busy. A usual cause is that the caller is calling itself.

**Error from modem,**
The modem unexpectedly answered with an error.

**Wrong DTMF tones received,**
The receiver received the wrong DTMF tones. One or more tones were interpreted wrong.

**Never got a call,**
The receiver part timed out because no call was received for some time.

**Interrupted,**
The thread got interrupted.
Report bugs.

**Exception thrown,**
Some unhandled exception was thrown.
Report bugs.

**Unknown error,**
An unknown error.
Report bugs.

Example output,

### 5.4.2 Facsimile Test Output

**No error,**
The test was successful.

**No Carrier,**
There is no carrier on the line.
5.4 Results

<table>
<thead>
<tr>
<th>Errors:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitter: No error</td>
<td>Receiver: No error</td>
</tr>
<tr>
<td>A: No error</td>
<td>B: No error</td>
</tr>
<tr>
<td>CallId: Unknown</td>
<td></td>
</tr>
<tr>
<td>Transmitter: No error</td>
<td>Receiver: No error</td>
</tr>
<tr>
<td>A: No error</td>
<td>B: No error</td>
</tr>
<tr>
<td>CallId: Unknown</td>
<td></td>
</tr>
<tr>
<td>Transmitter: No error</td>
<td>Receiver: No error</td>
</tr>
<tr>
<td>A: No error</td>
<td>B: No error</td>
</tr>
<tr>
<td>CallId: Unknown</td>
<td></td>
</tr>
<tr>
<td>Transmitter: Handshake failed</td>
<td>Receiver: Wrong DTMF tones received</td>
</tr>
<tr>
<td>A: No error</td>
<td>B: No error</td>
</tr>
<tr>
<td>CallId: Unknown</td>
<td></td>
</tr>
<tr>
<td>Transmitter: No error</td>
<td>Receiver: No error</td>
</tr>
<tr>
<td>A: No error</td>
<td>B: No error</td>
</tr>
<tr>
<td>CallId: Unknown</td>
<td></td>
</tr>
<tr>
<td>Transmitter: No error</td>
<td>Receiver: No error</td>
</tr>
<tr>
<td>A: No error</td>
<td>B: No error</td>
</tr>
<tr>
<td>CallId: Unknown</td>
<td></td>
</tr>
<tr>
<td>Transmitter: No error</td>
<td>Receiver: Wrong DTMF tones received</td>
</tr>
<tr>
<td>A: No error</td>
<td>B: No error</td>
</tr>
<tr>
<td>CallId: Unknown</td>
<td></td>
</tr>
<tr>
<td>Transmitter: No error</td>
<td>Receiver: No error</td>
</tr>
<tr>
<td>A: No error</td>
<td>B: No error</td>
</tr>
<tr>
<td>CallId: Unknown</td>
<td></td>
</tr>
<tr>
<td><strong>Ratio: 8/10 = 80.0%</strong></td>
<td></td>
</tr>
</tbody>
</table>

Busy,
Phone line is busy.

Error,
The modem returned an error.

Timeout,
A task (AT command) timed out, most possibly an answer from the modem

Example facsimile output,

<table>
<thead>
<tr>
<th>Errors:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitter: No errors</td>
<td>Receiver: No errors</td>
</tr>
<tr>
<td>Transmitter: No errors</td>
<td>Receiver: No errors</td>
</tr>
<tr>
<td>Transmitter: No errors</td>
<td>Receiver: No errors</td>
</tr>
<tr>
<td>Transmitter: No errors</td>
<td>Receiver: No errors</td>
</tr>
<tr>
<td>Transmitter: No errors</td>
<td>Receiver: No errors</td>
</tr>
<tr>
<td>Transmitter: No errors</td>
<td>Receiver: No errors</td>
</tr>
<tr>
<td>Transmitter: AT+FDT timeout, 20 secs</td>
<td>Receiver: No errors</td>
</tr>
<tr>
<td>Transmitter: No errors</td>
<td>Receiver: No errors</td>
</tr>
<tr>
<td>Transmitter: No errors</td>
<td>Receiver: No errors</td>
</tr>
<tr>
<td>Transmitter: No errors</td>
<td>Receiver: No errors</td>
</tr>
<tr>
<td><strong>Ratio: 9/10 = 90.0%</strong></td>
<td></td>
</tr>
</tbody>
</table>
Bibliography


