Project Title: A 1200 Baud Phase Modulated Telephone Modem

Team Members: Saad Z. Khan

Aman P. Dasson

Faculty Advisor: Elizabeth A. Thompson, Ph.D.

Date: December 8th 2008
### TABLE OF CONTENTS

**TABLE OF FIGURES** ...........................................................................................................3

**ACKNOWLEDGEMENTS** .................................................................................................4

**ABSTRACT/SUMMARY** ...................................................................................................5

**SECTION I: PROBLEM STATEMENT** .............................................................................6

1.1 Background Information .........................................................................................6

1.2 Requirements and Specifications ............................................................................6

1.3 Given Parameters and Quantities ............................................................................7

1.4 Design Variables ......................................................................................................7

1.5 Limitations and Constraints .....................................................................................9

1.6 Additional Considerations ......................................................................................9

**SECTION II: SELECTED CONCEPTUAL DESIGN** ......................................................10

2.1 Hardware ..................................................................................................................10

2.1.1 Transmitter ..........................................................................................................10

2.1.2 Receiver ...............................................................................................................10

2.2 Software ..................................................................................................................11

2.3 Advantages, Disadvantages and Considerations ....................................................11

**SECTION III: FINAL DESIGN** ....................................................................................13

3.1 Hardware Design .....................................................................................................14

3.1.1 Hardware Components .......................................................................................14

3.1.2 Printed Circuit Board (PCB) Design & Fabrication ...........................................20

3.1.3 Hardware Process Flow .......................................................................................23

3.2 Software Design .....................................................................................................23

3.2.1 Transmit & Modulation ......................................................................................23

3.2.2 Receive & Demodulation ....................................................................................27

3.2.3 Acknowledgement & Verification ......................................................................30

**SECTION IV: TESTING AND EVALUATION** ...............................................................32

4.1 Hardware ..................................................................................................................32

4.2 Software ..................................................................................................................32

4.2.1 Modifications to Software during Build Time ...................................................33

4.2.2 Post-Finalization Testing and Results .................................................................34

**CONCLUSION** .............................................................................................................37

**REFERENCES** ..............................................................................................................38

**APPENDIX** ................................................................................................................39
TABLE OF FIGURES

Figure 2.1.1 Basic transmitter circuit diagram ................................................................. 10
Figure 2.1.2 Basic receiver circuit diagram ................................................................. 11
Figure 3 Overall process and data flow for the design ............................................. 13
Figure 3.1.1(a) Diagram of transmitter circuit ............................................................. 14
Figure 3.1.1(b) Quad OP-Amp LM324 ................................................................. 16
Figure 3.1.1(c) Non-inverting OP-Amp circuit .......................................................... 16
Figure 3.1.1(d) Voltage-follower OP-Amp circuit ..................................................... 17
Figure 3.1.1(e) Top view of completed circuit ........................................................... 17
Figure 3.1.1(f) Left view of completed circuit ........................................................... 17
Figure 3.1.1(g) Right view of completed circuit ......................................................... 18
Figure 3.1.1(h) Back view of completed circuit ......................................................... 18
Figure 3.1.1(i) Diagram of receiver circuit ................................................................. 18
Figure 3.1.1(j) Left view of completed circuit ........................................................... 19
Figure 3.1.1(k) Left view of completed circuit ........................................................... 19
Figure 3.1.1(l) Left view of completed circuit ........................................................... 19
Figure 3.1.2(a) Ultiboard receiver circuit layout ....................................................... 20
Figure 3.1.2(b) Ultiboard transmitter circuit layout .................................................. 20
Figure 3.1.2(c) Receiver PCB top view ................................................................. 21
Figure 3.1.2(d) Receiver PCB bottom view ............................................................... 21
Figure 3.1.2(e) Transmitter PCB top view ............................................................... 21
Figure 3.1.2(f) Transmitter PCB bottom view .......................................................... 21
Figure 3.1.2(g) Completed PCB receiver top view .................................................... 22
Figure 3.1.2(h) Completed PCB receiver bottom view ............................................ 22
Figure 3.1.2(i) Completed PCB transmitter top view ............................................... 22
Figure 3.1.2(j) Completed PCB transmitter bottom view ....................................... 22
Table 3.2.1(a) Table of DTMF Frequencies ............................................................... 24
Figure 3.2.1(a) Screenshot of the DTMF Dial Tone Generator program .................. 24
Figure 3.2.1(b) Table of dibit pattern and corresponding phase angle change ...... 25
Figure 3.2.1(c) Phase shift of carrier wave according to dibit pattern ...................... 25
Figure 3.2.1(d) Arrangement of dibit / phase angle sequence ................................ 26
Figure 3.2.1(e) Plots of the original unmodulated carrier wave and the modulated analog wave .............................................................. 27
Figure 3.2.2(a) Plot of the raw analog data captured by the receiver ..................... 28
Figure 3.2.2(b) Plot of the phase angle difference between the baud intervals .... 29
Figure 3.2.3 Flowchart of overall software modulation/demodulation process ...... 31
Table 4.2.2(a) Measurements for transmission duration of signal, the preamble and the data ..................................................................................... 34
Table 4.2.2(b) Packet error rate measurement ......................................................... 35
Figure 4.2.2(a) Power spectrum of DTMF tones at 12000 samples/sec ............... 36
Figure 4.2.2(b) Power spectrum of DTMF tones at 48000 samples/sec ............... 36
Table 4.2.2(c) Measurement of audio levels through the circuits ......................... 36
ACKNOWLEDGMENT

We would like to extend our gratitude to our faculty advisor Dr. Elizabeth Thompson for her continuing support and dedicated guidance in the development and formulation of this project. We would also like to thank Dr. Timothy Loos of Raytheon Corporation for his feedback and recommendations. Without their contribution, this project would have never been fully realized. Dr. Jim Isaacs is another honorable mention for all his timely and valuable input to our many questions. And finally, a thank you to the entire Department of Engineering faculty at IPFW for giving us the opportunity to earn and apply the prerequisite knowledge and education that was put to work during this project.
Data transmission via the application of telecommunications is an essential part of today’s technological world. We propose to demonstrate the design and implementation of a moderately inexpensive 1200 baud phase modulated telephone modem that is capable of operating over a plain-old-telephone-system (POTS) interface by dialing out DTMF touch tones and establishing connection with another terminal. It should utilize some form of modulation protocol and error detection and correction method to successfully transmit and receive a small data file over the telephone network.

The students went through several steps to come to a design which they consider appropriate for the job in accordance with the requirements of the project. For the first half of the project, the students worked on the design process and analysis by going through a number of steps. First, the problem was stated. Then, analysis and research was done on the different conceptual designs. A final design was selected by evaluation and screening. Finally, detailed design analysis and cost estimation of the chosen design was done. For the second half of the project, the final selected design was actually built and programmed in stages and then rigorous testing and evaluation was done to ensure that it worked as close to specifications as was originally stated.

It should be noted that the goal of this project is not to make a pioneering product or an enhanced version of an existing technology. It is to gain knowledge. Knowledge of how a modem is designed, built and made to function. The students will seek to get a thorough understanding of the design process and implementation and ultimately recognize what is meant by ‘to engineer’.
A telephone modem is a device that sends digital data over a phone line. It modulates an analog carrier signal to encode digital information. A modem on the receiver end demodulates this received carrier signal to decode the transmitted information. Modems are generally classified by the amount of data they can send in a given time, normally measured in bit rate and/or baud rate. Modems work on one of two modes of operation, either full duplex or half duplex[^3]. We will be using the half duplex mode. Modulation in telecommunications is the process of varying the shape of a signal, typically a periodic sinusoid waveform which would be the carrier signal[^4]. Amplitude, phase and frequency can all be altered to attain the modulated signal. There are several modulation techniques that can be used to achieve this. For our purposes, we will be using a four phase modulation technique. Error detection and correction are also essential in ensuring data integrity while transmitting or receiving across noisy channels[^5]. There are several schemes to both error detection and correction of which we shall be using a simple checksum to correctly tally the transmitted data bits. Digital Tone Multi Frequency (DTMF)[^6] is a system of telephone signaling that is used over the line to instruct a telephone switching system of the telephone number to be dialed, or to issue commands to other related telephony equipment such as the modem in this case. The modem will use these touch tones to dial out to the receiving party and establish a connection.

1.2 - Requirements and Specifications

The project requires certain design criteria to be met which include the following:

- **Telephone interface** - The modem is to operate over a regular telephone system with a two-wire telephone interface consisting of TIP and Ring line. The full bandwidth of 3 kHz for the line is to be used.

- **Baud rate** – The system should have a minimum baud rate of 600 baud/sec with a corresponding bit rate of around 1200 bps.

- **Duplex system** - A half duplex system is to be designed in which calls will be initiated by sending touch tone frequencies for a time which is more than 60ms.

- **Error correction** – The device has to incorporate some form of error correction with 99% probability of correct data transmission. A message error probability can be calculated by running repeated tests and checking to see how many times an error message is generated if there is indeed corruption in the data.
A 1200 Baud Phase Modulated Telephone Modem

- **File Size** – The text file to be transmitted should not exceed 2 KB (This is a practical limitation due to the MATLAB processing.)

- **Audio I/O** – The transmitter must be able to generate output audio at -9 dBm and the receiver should receive at least 12 dB down from transmit, or -21 dBm.

- **Dual Tone Multi Frequency (DTMF)** – The system should be able to send out touch tones for dialing. The touch tones are generated when a single number on the keypad is pressed which creates a two tone pair. Recording actual terms via the PC microphone maybe useful in debugging the system. MATLAB or any other software could be used to generate the tones at a later stage.

- **Isolation Transformer and Surge Protection** – An isolation transformer for coupling is needed that is capable of providing up to 1500V isolation. The transformer needs to protect both the sound card with a surge protection of ~5V and allow a ring voltage of 70~120V. For additional circuit protection, an isolation amplifier and resistors should be used between the audio out from the sound card and the transformer. Zener diodes and metal oxide varisters or neon bulbs can be used to protect the circuit.

### 1.3 - Given Parameters or Quantities

The following are fixed parameters for this design:

- **Compatibility** – Should be compatible with POTS two-wire telephone interface with standard RJ -11 phone jacks and be able to interface with any current computers that have a sound card.

### 1.4 - Design Variables

The design variables are the quantities of variation in the system that are needed so as to comply with the given requirements and specifications. The hardware involved in the system design will have at the most, just a few minor variations, implemented as per needed by the requirements. Software variations will most probably take up the bulk of the design variables.

**Hardware**

- **Ring Detection** – There are a number of options available to the designers to ensure that the modem detects the ring in a phone line. These are needed to be investigated and a suitable solution chosen.

- **Off / On Hook** – One of the issues is whether the off/on hook should be electronically controlled. If it is to be manually controlled, then all that needs to be done is that a switch needs to be closed by hand. One way of electronically controlling the off/on hook is may be to use the audio channels of the sound card and control a photo metal oxide
semiconductor (MOS) relay to regulate the off/on hook. This and other possibilities need to be explored.

**Software**

- **Data Rate** – The data rate (or bit rate) can be varied. As one increases the data rate, the bit error rate also increases proportionally. Therefore, the designers have to decide on a trade-off that will deliver the required results yet minimize the receive error rate.

- **Low Pass Filter** – For the sampling rate of the signals, if the sound card’s maximum sample rate of 48 KHz is used, there will be a significant unused bandwidth since the audio signal can only range from 200 Hz to 3.2 KHz. Therefore, the implementation of a low pass filter in the software will be used to eliminate noise above these frequencies.

- **Error Correction & Detection** – If error protection is added to the design, the transmission time increases. However, since one of the requirements of the project is to have 99% validity of the data, the designers have to implement one of several available types of error correction. These include resending blocks of data, redundancy, CRC – cyclical redundancy checking, Hamming code, parity bit, checksum, hash functions etc.

- **Communications Protocol & Modulation** – Similarly, there are a number of different modulations that can be used for modulating the carrier signals. Amongst the most routinely used ones are ASK, PSK, QPSK, FSK and QAM, a combination of ASK and PSK. ACK and NACK are parts of a communications protocol that are useful for indicating acknowledgment of received data packets.

- **Determination of bit start/end** – One of the software issues is determining where the first bit starts and also when the transmitter has stopped sending bits. A transmit preamble protocol could be employed to detect the first bit start.

- **Synchronous/Asynchronous Data Transmission** – There are two ways to transmit data: Synchronous transmissions are synchronized by an external clock, while asynchronous transmissions do not occur at specific rate. Synchronous data transmission will be used.

**System Conditions**

- **Environment** – The entire design will be implemented indoors, at standard room temperature and pressure. The two computers will be connected through their sound cards with proper isolation to the external hardware and regular phone lines that will serve as the data transmission lines.
1.5 - Limitations & Constraints

- **Phone Interface** – As previously stated, the phone has to operate over a standard POTS phone line with an RJ-11 jack. A custom PBX or other digital telephone systems will not be supported.

- **Cost** – The total cost of the project excluding the two computers and phones should not exceed $50.

- **Size** – The prototype will be designed and built on a standard breadboard; therefore, the components and structure of the system must be taken into consideration.

1.6 - Additional Considerations

There are other restraining factors that need to be analyzed and recognized which will play into the design of the prototype. These parameters include the following:

- **Time** – The timeframe provided for the completion of this project is limited to that set forth by the school of ETCS and the Department of Engineering; therefore, the design must be manageable within the allotted time.

- **Safety** – The system must be safe to use in any standard indoor environment with the prerequisite phone connections and meet all safety requirements set forth by the appropriate regulatory bodies. This may include surge protection, power consumption etc. FCC regulations (shown in the appendix) will be followed in all applicable areas.

- **Existing Technology** – The hardware components should utilize existing technology that is readily available and relatively inexpensive. They should be compatible with each other and also with the connected computers and phone lines.

- **Tools** – Standard laboratory tools will be used to implement the hardware. For the software aspect of the design, MATLAB or any other available and suitable software will be utilized.
SECTION II: SELECTED CONCEPTUAL DESIGN

2.1 – Hardware

2.1.1 - Transmitter

The selected conceptual design that was chosen gave a fair picture of what hardware components and software features are to be used in the design of the modem. A brief overview of that conceptual design is given here. Firstly, the audio output port of the sound card connected to the modem is utilized to send out the DTMF touch tone signals generated in MATLAB or any other suitable software. The audio input port of the sound card will be used for acknowledgement by the transmitter when the receiver side relays back the information that the data has been correctly sent. The transmitter circuit will be isolated from the phone line via a 1:1 (600Ω:600Ω) isolation transformer that will decouple the two circuits. It allows an AC signal or power to be taken from one device and fed into another without electrically connecting the two circuits and block transmission of DC signals while allowing AC signals to pass. The isolation transformer will also block interference caused by ground loops. This is to protect the components in case there is a voltage surge coming in from the phone line. An isolation amplifier and resistor will be used between the audio output port of the soundcard and the transformer to amplify the signal. In addition, back to back Zener diodes will be used to protect the circuit. A RJ-11 phone jack completes the transmitter circuit for the modem. Preliminary circuit figure of the transmitter is shown below:

![Transmitter Circuit Diagram](image_url)

Figure 2.1.1 – Basic Transmitter circuit diagram

2.1.2 – Receiver

The receiver circuit, which is essentially a replica of the transmitter circuit, will make use of the audio input port of the soundcard to receive the data while the audio output port will be used to send the acknowledgement to the transmitter circuit. Here too, a 1:1 isolation transformer will be used along with Zener diodes to protect the components. In
addition, the other end of the transformer will be connected to a LED for the ring tone detection and that will then be interfaced to a RJ-11 phone jack. Preliminary circuit figure of the receiver is shown below:

![Basic Receiver Circuit Diagram](image)

**Figure 2.1.2 – Basic Receiver circuit diagram**

### 2.2 – Software

DTMF touch tones will be generated and decoded using one or a combination of the softwares listed above. A modulation scheme and error protection protocol will also be implemented through the software interface. Another software aspect of the design is the use of a bandpass filter to eliminate noise above the 400Hz to 3.4 kHz frequencies and filter out the 60 Hz hum.

### 2.3 – Advantages, Disadvantages and Considerations

**Advantages:**

- Relatively cheap to implement since it utilizes existing technology and the components required are commercially available.

- The circuit hardware is self contained and therefore compact enough to be built on a standard breadboard.

- There is plenty of room for improvisation and future upgrade options.

- Suitable for modification, visualization and demonstration of various system components.

**Disadvantages:**

- Because of the vast software aspect of the design, it will be challenging to program and configure.
Considerations:

- Owing to the in-depth software aspect of this design, it would be beneficial to have some kind of graphical user interface that would clearly and effectively allow the user to configure and demonstrate the functioning of the modem.
SECTION III: FINAL DESIGN

The figure below is a representation of the general process of the hardware and software interfacing that will take place in this design:

Figure 3 – Overall process and data flow for the design
3.1 – Hardware Design

As previously stated, the hardware components used in the system are to a certain degree, the same except for a few modifications as per required.

3.1.1 – Hardware Components

Below is the circuit diagram of the final hardware design for the transmitter circuit:

Figure 3.1.1(a) – Diagram of transmitter circuit

i. Sound Card

A standard sound card with built in audio input and output jacks is used. It is the only connection between the computer and the modem. The digital to analog conversion (D/A) and analog to digital conversion (A/D) will take place on the sound card itself therefore bypassing the need for extra hardware. The dynamic range of the sound card is at least 16 bit and is capable of a sampling rate between 3 kHz - 48 kHz.
ii. Resistor R1

Resistor R1 must be selected to draw enough DC voltage so that it drops from 48 V to 6 V at nodes 1 and 2. The DC voltage across the telephone line is typically 48 V but it can vary from 42 V – 54 V. The typical value for this resistor is around 180 Ω.

iii. Double-Pole Double-Throw (DPDT) Switch

This switch (RS #275-403) serves to make the transmitter circuit go off hook when it is moved to the closed position.

iv. Capacitor C2

The capacitor C2 (RS #272-1055) blocks the DC voltage from the transformer. C2 is a non-polar type since the polarity of the DC voltage cannot be guaranteed and often times reverses with different operating modes. The voltage rating must be high enough to withstand the usual DC voltage (and variations) plus the AC ring voltage; a value of 250 V working voltage is recommended. The value of this capacitor is 1 µF.

v. Isolation Transformer

The 1:1 (600Ω:600Ω) isolation transformer (RS #273-1374) isolates the circuit from the phone line and decouples the two circuits. It allows an AC signal or power to be taken from one device and fed into another without electrically connecting the two circuits and blocks transmission of DC signals while allowing AC signals to pass. The isolation transformer will also block interference caused by ground loops. This is to protect the components in case there is a voltage surge coming in from the phone line.

vi. Back-to-Back Zener Diodes

The IN4733A 5.1 V back-to-back zener diodes (RS #276-0565) provide a secondary clamp to any high voltage that gets through the transformer and protects the downstream equipment. A wide range of values is accepted. The IN433A limits the output to 5.1 V peak-to-peak and has a current rating of 49 mA with a maximum power dissipation of 1 W.

vii. Quad-Operational Amplifier

The 14-pin LM324 Quad OP AMP IC (RS #276-1711) amplifies the signal and provides isolation to the sound card. It will be powered by a 10 V DC power source. One of the four internal op-amps is connected to the audio output of the soundcard and provides a gain of 11 to the generated analog signal. The figure below is a schematic of the Quad OP AMP LM324:
A 1200 Baud Phase Modulated Telephone Modem

Figure 3.1.1(b) – Quad OP-Amp LM324

The figure below shows the non-inverting op-amp circuit:

Figure 3.1.1(c) – Non-inverting OP-Amp circuit

\[
Av(\text{gain}) = \frac{vo}{v1} = 1 + \frac{R2}{RI}
\]

Another op-amp is connected to the audio input of the soundcard and has a voltage-follower configuration. The figure below shows the voltage-follower op-amp circuit:
A 1200 Baud Phase Modulated Telephone Modem

![Voltage-follower OP-Amp circuit](image)

Figure 3.1.1(d) – *Voltage-follower OP-Amp circuit*

\[ Av(\text{gain}) = \frac{v_o}{v_l} = 1 \]

viii. 3.5mm Audio Input/Output Jacks

These 1/8” stereo audio jacks (RS #274-0249) are used to connect the soundcard to the circuitry.

ix. RJ-11 Phone Jacks

These are generic phone jacks that connect the circuit to the external telephone line.

Shown below are images of the completed transmitter circuit:

![Figure 3.1.1(e) – Top View](image)

![Figure 3.1.1(f) – Left View](image)
Below is the circuit diagram of the final hardware design for the receiver circuit:

![Diagram of receiver circuit](image)

The receiver circuit is almost identical to the transmitter circuit except for the inclusion of ring detection circuitry.
x. Light Emitting Diode (LED)

A green LED (RS #276-0304) rated at 2.1 V with a current rating of 30 mA is used for ring detection in the circuit. It is connected in series across the telephone line with a resistive network equivalent to 1.39 kΩ.

Shown below are images of the completed receiver circuit:

Figure 3.1.1(j) – Top View

Figure 3.1.1(k) – Right View

Figure 3.1.1(l) – Back View
3.1.2 – Printed Circuit Board (PCB) Design & Fabrication

We decided that PCB design and fabrication would promote a more professional design prototype. We were also able to design in a smaller scale for size constraints. This decision came with a cost in that we lost time by drawing up the schematic in MultiSim and then designing the board layout in Ultiboard. This included searching for proper footprints for all the components involved in the design. However, the finished product worked just as well and had more aesthetic appeal.

The figures shown below are the Ultiboard board layout schematics for both the receiver and the transmitter circuit:

Figure 3.1.2(a) – Ultiboard receiver circuit layout

Figure 3.1.2(b) – Ultiboard transmitter circuit layout
Shown below are the images of the receiver PCB board after fabrication:

Figure 3.1.2(c) – Receiver PCB Top View
Figure 3.1.2(d) – Receiver PCB Bottom View

Shown below are the images of the transmitter PCB board after fabrication:

Figure 3.1.2(e) – Transmitter PCB Top View
Figure 3.1.2(f) – Transmitter PCB Bottom View
Shown below are the images of the finished receiver PCB with components installed:

Figure 3.1.2(g) – Receiver Top View  
Figure 3.1.2(h) – Receiver Bottom View

Shown below are the images of the finished transmitter PCB with components installed:

Figure 3.1.2(i) – Transmitter Top View  
Figure 3.1.2(j) – Transmitter Bottom View
3.1.3 – Hardware Process Flow

i. The switch on the transmitter circuit is manually moved to the closed position and that enables resistor R1 to be connected across the telephone line which drops the DC voltage from 48 V to approximately 6 V. Once the voltage has been dropped, the telephone exchange is aware that the line is off-hook and ready for communication and a dial tone is sent out.

ii. DTMF tones are sent out from the transmitter circuit over the line to the telephone exchange where they are decoded and an AC signal is sent out over the line to the destination.

iii. The LED on the receiver circuit starts to flash once the AC signal is detected. The switch is then manually moved to the closed position which drops the DC voltage from 48 V to approximately 6 V.

iv. Once the DC voltage has dropped, the telephone exchange recognizes that the line is off-hook and ready for communication. The telephone exchange then connects the two lines and handshaking takes place. Now, data transmission can be initiated.

3.2 – Software Design

The bulk of this project’s design and implementation lay in the software aspect of it. There were numerous additions and revisions to the code that enabled the modem to successfully transmit, receive and process the data to yield the final result. The major sections of the software code for the transmission and the receiving functions will be discussed here to gain an understanding of how the data was processed.

3.2.1 – Transmit and Modulation

Modulation is performed at a rate of 1200 baud using the carrier frequency of 1800 Hz and a sampling frequency of $f_s=12000$ samples/sec which is within the range of the sound card’s sampling capabilities. The number of samples in a baud interval are calculated by

$$N = \frac{f_s}{\text{baud}}.$$

i. Dual-Tone Multi-Frequency (DTMF)

To send out the touch tones for dialing, a freely available software program known as DTMF Encoder/Decoder Dial Tone Generator (Ham Radio Software by Norbert Pieper) was used. This simple program uses a signal that is encoded as a pair of sinusoidal (sine wave) tones from the table below which are mixed with each other. DTMF is used by most PSTN (public switched telephone networks) systems for number dialing, and is also used for voice-response systems such as telephone banking and sometimes over private radio networks to provide signaling and transferring of small amounts of data.
This software is very easy to use. The telephone number to be dialed is input from the graphical keypad or by using the computer keyboard and the ‘play’ button is then pressed. This sends out the dual tone frequencies through the sound card and into the circuit which then sends the sinusoidal signal over the telephone line. Provided below is a snapshot of the program’s interface:

Table 3.2.1(a) - Table of DTMF frequencies

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Tone B [Hz]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1209</td>
</tr>
<tr>
<td><strong>Tone A [Hz]</strong></td>
<td></td>
</tr>
<tr>
<td>697</td>
<td>1</td>
</tr>
<tr>
<td>770</td>
<td>4</td>
</tr>
<tr>
<td>852</td>
<td>7</td>
</tr>
<tr>
<td>941</td>
<td>*</td>
</tr>
</tbody>
</table>

Figure 3.2.1(a) – Screenshot of the DTMF Dial Tone Generator program

ii. Conversion of ASCII to Binary Data

The contents of the text file that has to be transmitted first have to be converted into binary form for digital modulation. This is achieved by using a code written in MATLAB titled ‘encode.m’ that reads in the ASCII characters from the text file and then accesses a lookup table in ‘code.m’ that lists the equivalent 8-bit binary representations of these ASCII characters. Those binary representations are then written into a new text file that is read into MATLAB and stored into an array in a dibit form (i.e. 2 x N array, where N is the number of required rows).
iii. Addition of Preamble

A preamble is employed to enable the demodulator to window align the baud interval when it receives the analog signal. For this, a significantly long string of binary 1’s is used with the starting value at zero phase and each subsequent dibit ‘11’ changing the phase angle of the previous one by $\frac{5\pi}{4}$. This will ensure that the preamble stands out from the data stream and can be easily picked up by the demodulator.

iv. Start Characters

The function of the start characters is to signify that the data is going to immediately follow after the predefined start character sequence bits end. A pattern of twelve repeating ‘01’ dibits were chosen that continuously change the phase angle by $\frac{3\pi}{4}$ after each subsequent dibit.

v. Data Stream Modulation

Next is the actual data itself that is stored in the original array. As before with the preamble and start characters, each of the dibits are read in and encoded as a phase change of the 1800 Hz carrier relative to the phase of the carrier during transmission of the immediately preceding dibit as indicated in the table below:

<table>
<thead>
<tr>
<th>DIBIT</th>
<th>PHASE CHANGE (DEGREES)</th>
<th>PHASE CHANGE (RADIANS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>+ 45°</td>
<td>+ $\pi/4$</td>
</tr>
<tr>
<td>01</td>
<td>+ 135°</td>
<td>+ $3\pi/4$</td>
</tr>
<tr>
<td>11</td>
<td>+ 225°</td>
<td>+ $5\pi/4$</td>
</tr>
<tr>
<td>10</td>
<td>+ 315°</td>
<td>+ $7\pi/4$</td>
</tr>
</tbody>
</table>

Table 3.2.1(b) – Dibit pattern and corresponding phase angle change

The phase shift is the actual phase shift in the transition region from the end of one signaling element to the beginning of the following signaling element as illustrated in the figure below:

Figure 3.2.1(b) – Phase shift of the carrier wave according to dibit pattern
vi. Checksum

A checksum of a message is an arithmetic sum of message code words of a certain word length, for example byte values, and their carry value. The sum is negated by means of ones-complement, and stored or transferred as an extra code word extending the message. The purpose of a checksum is for the receiver side to detect any errors or missing bits in the received data. The data packet’s checksum is calculated by the receiver and compared to the one received from the sender at the end of the packet. If the two match, the receiver sends an <ACK> message back to the sender. If there is a problem with the checksum, the receiver instead sends a <NAK>. If a <NAK> was received, the sender would re-send the packet. In the modulation code, the function ‘c’ that points to ‘count_ascii_values.m’ calculates the decimal sum of all the ASCII values in the original text file and then converts that sum into a binary representation. This binary sum is stored as a string of characters which is then converted to doubles. Then a new array is formed with zeros appended before the original checksum to make sure that there are at least eighteen binary digits in total (this is because the largest file that we will try to transmit will be of a size of 3 KB whose checksum does not exceed the eighteen bit binary representation). This is then reordered as a dibit array and the same procedure is implemented to convert those to phase angle values and stored in a separate array.

vii. Stop Characters

Finally, stop characters are used to signify the end of the data stream. A pattern of five repeating ASCII ‘@’ symbols were chosen as the sequence and their binary representations in an array were added to the end of the data stream.

viii. Transmission of Analog Signal

The phase angle values of the preamble, start characters, data stream, checksum and stop characters are concatenated into one matrix and replicated N= 10 times which are the number of samples in each baud interval. The figure below shows the arrangement of the dibit and corresponding phase angle sequences prior to transmission:

<table>
<thead>
<tr>
<th>Preamble bits</th>
<th>Start character bits</th>
<th>Data Stream</th>
<th>Checksum</th>
<th>Stop Character bits</th>
</tr>
</thead>
</table>

Figure 3.2.1(c) – *Arrangement of the dibit / phase angles sequence*

The array containing the phase angle values is incorporated as the phase angle of a sine wave with carrier frequency 1800 Hz. Thus, the modulated wave consists of a sine wave whose phase remains constant for the 10 samples of the baud interval, but changes at the baud boundaries. In this way, the data stream, in combination with the preamble, start characters, checksum, and stop characters, is encoded into the phase angle of the modulated sine wave, and each phase change
represents one of the dibit patterns depicted in figure 3.2.1(b). This modulated wave is then transmitted out of the computer as sound via the audio out port of the soundcard over the telephone line to the receiving party.

![Original unmodulated waveform](image1)

![Modulated waveform](image2)

**Figure 3.2.1(d) – Plots of the original unmodulated carrier wave and the modulated analog wave**

### 3.2.2 – Receive and Demodulation

Similar to modulation, the demodulation is also performed at a rate of 1200 baud using the carrier frequency of 1800 Hz and a sampling frequency of $f_s=12000$ samples/sec which is within the range of the sound card’s sampling capabilities. The number of samples in a baud interval are calculated by $N = \frac{f_s}{\text{baud}}$.

i. **Sound Capture**

The receiving computer is setup to automatically begin data acquisition when it detects data coming through the sound card. It should be noted that MATLAB normalizes the audio signal, regardless of the precision of the A/D. This is achieved by initiating the capture device’s properties on the soundcard and setting a trigger on the receiving channel that will begin data acquisition the instant it detects a value greater than 0.3 on the rising edge of the sinusoidal waveform. The actual signal coming in from the line is usually around the level of around $\approx 0.5$. 
**A 1200 Baud Phase Modulated Telephone Modem**

Anything below 0.3 is usually noise and so the triggering automatically takes care of any noise that is being transmit with the signal and starts acquisition from where the preamble starts. This acquired data is stored into a variable.

![Captured Data](image)

**Figure 3.2.2(a) – Plot of the raw analog data captured by the receiver**

**ii. Pre-signal Noise Elimination**

As a precaution against pre-signal noise being acquired before the start of the preamble, the input data is compared to check where the level is above 0.3 and the index of the data is accordingly shifted to make that as the starting point of the actual data stream (this includes the preamble, start characters, etc.).

**iii. Window Alignment using Preamble**

The next step involves aligning the window so that all the baud intervals in the signal are properly lined up for the demodulator. The number of baud intervals is calculated by diving the remaining length of the received signal (after removing pre-signal noise) by the fixed number of samples in a given band interval, N. Two carrier waveforms are generated with a carrier frequency of 1800 Hz and length same as that of the received signal. One of the carriers is a cosine and the other a sine wave. The cosine waveform is multiplied by the received signal and the summation is taken according to the following equation:
A 1200 Baud Phase Modulated Telephone Modem

\[ \cos(A) \cdot \cos(B) = \frac{1}{2} \cdot [\cos(A + B) + \cos(A - B)] \]

When the summation is done, this automatically does low pass filtering and eliminates the second cosine term. The remaining fraction of the summation is known as the ‘Inphase’ component of the signal. Then the sine waveform is multiplied by the received signal and again the summation is taken which forms the ‘Quadrature’ component. Following that, the magnitude of the inphase and quadrature components is determined for every baud interval in the signal and finally the total sum of all these magnitudes is obtained. A new index is generated from the received signal at the value where the maximum sum occurs and this then is the baud interval alignment.

Now that the baud intervals are aligned, this whole process is again repeated for the received signal from the proper index and the arc tangent values of theta are calculated and stored into an array which contains the phase angle differences between the baud intervals. These phase angle differences are mapped back into their binary counterparts using the same four-quadrant-phase modulation criteria that was previously established and stored into a new array. This new array now contains the demodulated binary preamble, start characters, data stream, checksum, stop characters and whatever noise that was included in the data acquisition after the trailing edge of the actual signal.

![Figure 3.2.2(b) – Plot of the phase angle differences between the baud intervals](image-url)
iv. Detection of Start Characters

The newly obtained array is then searched for the previously defined repeating twelve dibit sequence of ‘01’. Once the program finds these dibits, the ‘data_start_index’ is assigned to the value of the dibit where the actual data stream starts. The preamble and start characters are thus discarded.

v. Detection of Stop Characters

The new array containing the data stream, checksum, stop bits and any remaining noise is then searched for three out of the five stop bits (ASCII ‘@’) that were originally transmitted along with the signal. Once the program finds this dibit pattern, it truncates the signal by readjusting the index to the end of the data stream and checksum.

vi. Extract Checksum Value

The resulting array is then split into two such that the last eighteen bits (or equivalently 9 dibits) are separated. These eighteen bits are the binary representation of the checksum. The binary values in this array are doubles and those are then converted back to characters which are consequently converted into their decimal equivalent. This decimal value of the checksum is stored in the variable ‘y3’.

vii. Converting Binary Data to ASCII

The remaining array from the previous step now contains only the original binary data values and are those are converted to ASCII using ‘decode.m’ and then written to a text file. The contents of this text file should be ‘exactly’ the same as the original text file that was transmitted over the telephone line!

3.2.3 – Acknowledgement & Verification

It has to be verified that the received data is indeed an exact replica of the transmitted source data. To do this, the decimal sum of the ASCII characters in the newly formed text file is calculated using ‘count_ASCII_values’ and stored into the variable ‘y’. This value is then compared to ‘y3’, the decimal value of the checksum previously calculated, and if there is a match then an acknowledgement is sent out by the receiver in the form of a simple sine wave from the soundcard. If the values do not match, the receiver goes back into polling mode and waits for retransmission.

The transmitter in the meantime enters into polling mode for any incoming data once it is done transmitting. If it obtains the acknowledgement from the receiver, then it prints out a confirmation of correct data transfer and the program stops running. In the event that the transmitter does not receive an acknowledgement from the receiver within
approximately 30 seconds, it will display an error message and retransmit the data. Shown below is the flowchart of overall software modulation/demodulation process:

![Flowchart of overall software modulation/demodulation process](image-url)

Figure 3.2.3 – Flowchart of overall software modulation/demodulation process
SECTION IV: TESTING AND EVALUATION

The designed modem was thoroughly tested both during build time and even after the final completion of the hardware circuit and of the software aspect of the design. The project went through several major and minor modifications that were implemented into the original design depending on the requirements and specifications. These will be documented and explained in this section.

4.1 – Hardware

The only significant change that took place in the hardware design of the modems was that the gain on the op-amps was adjusted to suit our needs. Initially both op-amps were without any applied gain; therefore they were in a voltage-follower configuration. However, when the signal level was checked at the Zener diodes (node 13) on the transmitter circuit, it was too low. As a result we could not see any signal at the receiver circuit.

The gain was then increased to 11 (using $Av(gain) = \frac{vo}{vl} = 1 + \frac{R2}{R1}$) by changing R4 from 150Ω to 10kΩ, R5 from 1.5MΩ to 10kΩ. After doing this, it was discovered that the op-amp was being overdriven. This was due to the output volume on the computer being set too high. We readjusted the volume level to half the original volume level. We could then see a proper signal without any distortion across the Zener diodes on the receiver circuit. However, the op-amp gain going into the soundcard for the receiver circuit was also set at 11 therefore it was also amplifying the signal and any noise components excessively therefore resulting in a severely distorted signal. The op-amp connected to the input of the soundcard was then changed to a voltage-follower configuration and this resolved the problem.

In our final configuration, the op-amps of the transmitter and receiver circuits connected to the outputs of the soundcard each have a gain of 11 whereas the op-amps connected to the inputs of the soundcard have unity gain.

4.2 - Software

The general method of assessing whether correct modulation and demodulation was being done and the right result being obtained was tested in three ways:

i. Modulating and demodulating the signal on the same computer.

ii. Next, both computers were connected to each other directly through their soundcards and the signal would then be modulated and transmitted on one while the other received and demodulated the analog signal.

iii. Finally, the same process was repeated by testing over the phone line and modem circuitry.

The software codes were revised continuously as we applied these three methods. This also served as a way to crosscheck our results across different platforms in case the results were not in agreement with what we expected them to be.
4.2.1 – Modifications to Software during Build Time

Originally when we started work on the code, we managed to get the signal to modulate and demodulate on the same computer with success. The signal began at a zero phase in order to signify the start of the data stream. Next we tried the second method of transmitting between two computers directly but that did not work for the obvious reason that there was no synchronization involved and therefore the data acquisition was inaccurate or at times completely off the mark. We figured that we would need to use a preamble and some start characters to precede the data stream before transmitting. This eliminates the need for using an initial zero phase since the phase changes accordingly with the next successive dibit pattern for the entire duration of the signal. The demodulator could then window align the baud interval and recognize the point where the data stream starts. This would get rid of the pre-signal noise, preamble, and the start characters leaving only the data plus whatever noise trailed it. This managed to solve the problem involved with transmitting between the two computers.

Following that, we attempted to transmit over the line but that failed to work. The reason for that happening was that when we attempted to transmit over the line, the op-amp gain would amplify the noise in the signal as well. The demodulator would assume that the start of the data is then somewhere between the pre-signal noise and hence compute the result using the incorrect data start index of the signal. To counter this we analyzed the plots of the acquired data and noted the average maximum value of the noise in the signal which was approximately 0.1-0.2 (normalized MATLAB values). We used a value of 0.3 as a threshold value and the demodulator would now only consider that part of the signal to be actual data where and when the signal went over 0.3. Even after fixing this issue with the pre-signal noise, we were still unable to yield the correct result after demodulation.

We then reduced the baud rate to 400 baud so that we could get more samples per baud interval. This gave us the correct result. The baud rate was then increased to 600 baud and that also proved to be successful. When we tried transmitting at the normal 1200 baud however, the results went astray. We analyzed the plots of the phase angle differences for the 600 and 1200 baud samples. From those plots, we were able to ascertain that there was some kind of distortion in the phase angles when we were using a baud rate of 1200. We thought that perhaps the trailing noise after the data stream was responsible for the corruption in the data. Stop characters were added to the code which would signify the end of the data stream and automatically eliminate the post-signal noise. This still did not rectify the problem.

Next we analyzed the demodulated binary data in the arrays for both the 600 and 1200 baud samples and found out that the phase angle tolerances would have to be increased for the four quadrants. This finally managed to resolve the problem and we were successfully able to demodulate the signal to get back our original data.

To further simplify the data acquisition process, we added a polling function to the receiving side which would start logging data once a trigger was initiated. We set the
A 1200 Baud Phase Modulated Telephone Modem

trigger condition to start data acquisition once the rising edge of the signal went above 0.3. As an unintended bonus, the data start index would automatically shift right to the start of the preamble because that is where the signal reached over 0.3 and so pre-signal noise also got eliminated from right from the onset.

Since one of the project requirements is error protection, we decided to use a simple checksum at the end of the data stream. This checksum is verified at the receiver’s end and if it matches, then an acknowledgement is sent back to the transmitter (which enters polling mode after sending out the signal) to let it know that the data has been correctly received. The transmitter waits for 30 seconds to get a reply and in the event that it does not receive an acknowledgment, it sends the data again.

4.2.2 – Post-Finalization Testing and Results

In order to meet the requirements and specifications of the project, we had to carry out several tests after our design was finalized to measure certain parameters and quantities and verify the proper functionality of the modem.

i. Transmission Time of Data and Preamble

The following table shows the measurements of the duration of the transmission of the entire signal, and the duration of the preamble and data by themselves:

<table>
<thead>
<tr>
<th># of trials</th>
<th>Time duration of transmission of entire signal (s)</th>
<th>Time duration of preamble (s)</th>
<th>Time duration of data (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.2204</td>
<td>10.1966</td>
<td>0.0238</td>
</tr>
<tr>
<td>2</td>
<td>10.2389</td>
<td>10.1969</td>
<td>0.042</td>
</tr>
<tr>
<td>3</td>
<td>10.2268</td>
<td>10.198</td>
<td>0.0288</td>
</tr>
<tr>
<td>4</td>
<td>10.2358</td>
<td>10.1985</td>
<td>0.0373</td>
</tr>
<tr>
<td>5</td>
<td>10.2347</td>
<td>10.1973</td>
<td>0.0374</td>
</tr>
</tbody>
</table>

| Average Value | 10.23132 | 10.19746 | 0.03386 |

Table 4.2.2(a) – Measurements for transmission duration of signal, the preamble and the data

ii. Packet Error Rate Measurement

The data transmission was done for a total of 100 trials and a running tally was kept of the confirmations for data received when polled, no error detected (checksum value matches), and of the acknowledgement sent (sine wave received at the transmitter).
Table 4.2.2(b) – Packet error rate measurement

<table>
<thead>
<tr>
<th># of Trials</th>
<th>Data Received</th>
<th>No Error Detected</th>
<th>Acknowledgement Sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>41</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>42</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>43</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>44</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>45</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>47</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>48</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>49</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The results were very promising as all 100 trials were successful for the three types of confirmations and thus achieved a 100% success rate.
iii. DTMF Tone Duration, Frequency and Power Spectrum

The following plots show the power spectrum of the signal sampled first at 12000 samples/sec and then at 48000 samples/sec. It can be inferred from the plots that the frequencies of the DTMF tones are in the region of the standard frequency assignments for the particular symbols as shown previously in Table 3.2.1(a).

![Power Spectrum of DTMF tones at 12000 samples/sec](image)

**Figure 4.2.2(a) – Power spectrum of DTMF tones at 12000 samples/sec**

![Power Spectrum of DTMF tones at 48000 samples/sec](image)

**Figure 4.2.2(b) – Power Spectrum of DTMF tones at 48000 samples/sec**

iv. Audio Input/Output Levels (RMS)

The following table shows the $V_{\text{RMS}}$ and $V_{\text{P-P}}$ measurements that were taken at various points in both the transmitter and receiver circuits:

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Location</th>
<th>Audio Level (RMS Voltage (mV))</th>
<th>Audio Level (P-P Voltage (mV))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitter</td>
<td>Output of soundcard</td>
<td>104</td>
<td>375</td>
</tr>
<tr>
<td></td>
<td>OP-Amp output</td>
<td>250</td>
<td>850</td>
</tr>
<tr>
<td>Receiver</td>
<td>Zener Diodes</td>
<td>25</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>OP-Amp output</td>
<td>25</td>
<td>150</td>
</tr>
</tbody>
</table>

**Table 4.2.2(c) – Measurement of audio levels through the circuits**
The ‘1200 Baud Phase Modulated Modem’ was meant to be a demonstrative project that would showcase how the hardware components and the software program would interface to perform data transmission over a communication medium such as the telephone line in this case. The modem was able to successfully perform all the above tasks and met all the requirements and specifications that were outlined in the problem statement. According to the test results, this modem is entirely capable of performing the standard tasks such as handshaking, modulation, demodulation, error correction and acknowledgement that are required of an actual modem.

If we had more time, we would have liked to achieve higher data transfer rates from the modem and make use of advanced features such as adaptive equalization and also set up a graphical user interface that would allow the user to select and program the modem to perform different modulation/demodulation schemes, employ various error protection protocols and be able to display or output statistical data pertaining to data rate measurements, frequency spectrum plots, etc.

In conclusion, we thought this to be a great learning experience for us as it simulates a real world problem and solution case in engineering. It gave us a thorough understanding of how to approach an engineering problem by way of going through ordered and well thought out stages of planning, formulation, development, testing, fabrication, diagnosing and implementation. All in all, we managed to do the one fundamental thing that we had set out to do during the course of our education and that is ‘engineer.’
SECTION VI: REFERENCES


<http://www.epanorama.net/circuits/teleinterface.html#usenet>.


<http://www.techbooksforfree.com/intro_to_data_com/page88.html#88>

<http://en.wikipedia.org/wiki/Error_detection_and_correction>

<http://www.techbooksforfree.com/intro_to_data_com/page51.html#Hardware>


MATLAB code for transmit & modulation:

*Final_Modulate.m*

```matlab
% ECE406 - Capstone Senior Design II
% Design of a 1200 Baud Phase Modulated Telephone Modem
% Indiana University-Purdue University Fort Wayne
% Fall 2008
% Team: Saad Z. Khan & Aman P. Dasson
% Faculty Advisor: Elizabeth A. Thompson, Ph.D.

% This code modulates a carrier signal with the phase angle changes formed
% by the digital data content of the text file that is to be
% transmitted. After the modulation is done, the analog signal is
% transmitted out of the soundcard of the computer.

clear all
clc

% Converting ASCII text in file to binary form
encode('test.txt','test.txt');

% Reading binary data from file
fid = fopen('test1.txt','r');  % test1.txt is the binary version of test.txt
A = fread(fid,[2,inf],'*char')';
fclose(fid);

y=count_ascii_values('test.txt');  %Assumed max length is 18 for a maximum
file size of 3 KB
d=dec2bin(y);   % Convert decimal values to binary

baud =1200;  % Baud rate
fc = 1800;   % Carrier frequency (Hz)
fs = 12000;  % Sampling frequency (samples/sec)
T = 1/fs;    % Time period (sec)
N=fs/baud;   % Number of samples in baud interval

% Define preamble for baud interval alignment
preamble = ones(2,500*N)';
[r1,c1] = size(preamble);
phi3 = zeros(1,r1);

% Define phi3(1)
phi3(1)=5*pi/4; % Assumes preamble(1,1) is '11'

for i=2:r1
    if (preamble(i,1)==1 & preamble(i,2)==1)
        phi3(i)=phi3(i-1)+5*pi/4;
    else
        a = 10 %test for failure
    end
end
```

```matlab
% Transmit the phase value
```
end

% Define pseudo-noise (PN)/start characters to define start of data
pn = [0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1];
[r2,c2] = size(pn);
phi4 = zeros(1,r2);
phi4(1) = phi3(r1)+3*pi/4;

for i=2:r2
    if (pn(i,1)==0 && pn(i,2)==1)
        phi4(i)=phi4(i-1)+3*pi/4;
    else
        b = 10 %test for failure
    end
end

% Define phi (phases corresponding to dibit stream)
% Each dibit encoded as a phase change of the 1800 Hz carrier relative to
% the phase of the carrier during transmission of the immediately preceding
% dibit
% if dibit is 00 add pi/4 to phi
% if dibit is 01 add 3*pi/4 to phi
% if dibit is 11 add 5*pi/4 to phi
% if dibit is 10 add 7*pi/4 to phi

[r,c] = size(A);
phi = zeros(1,r);
% Define phi(1) based on value of A(1,:)& last value of start characters
if (A(1,:) == '00')
    phi(1)=phi4(r2)+pi/4;
elseif (A(1,:) == '01')
    phi(1)=phi4(r2)+3*pi/4;
elseif (A(1,:) == '10')
    phi(1)=phi4(r2)+7*pi/4;
else
    phi(1)=phi4(r2)+5*pi/4;
end

for i=2:r
    if (A(i,:) == '00')
        phi(i)=phi(i-1)+pi/4;
    elseif (A(i,:) == '01')
        phi(i)=phi(i-1)+3*pi/4;
    elseif (A(i,:) == '10')
        phi(i)=phi(i-1)+7*pi/4;
    else
        phi(i)=phi(i-1)+5*pi/4;
    end
end

% Insert checksum after data and before stop characters
% Convert characters in d to numbers
Len=length(d);
for i=1:Len
    if(d(i)=='1')
        d1(i)=1;
    end

else
d1(i)=0;
end
if Len<18
    num=18-Len;
d1=[zeros(1,num) d1];
end
d2=reshape(d1,2,9)';
[r4,c4] = size(d2);
phi7 = zeros(1,r4);
if (d2(1,:) == 0 & d2(1,2)==0)
    phi7(1)=phi(r)+pi/4;
elseif (d2(1,:) == 0 & d2(1,2)==1)
    phi7(1)=phi(r)+3*pi/4;
elseif (d2(1,:) == 1 & d2(1,2)==0)
    phi7(1)=phi(r)+7*pi/4;
else
    phi7(1)=phi(r)+5*pi/4;
end
for i=2:r4
    if (d2(i,1) == 0 & d2(i,2)==0)
        phi7(i)=phi7(i-1)+pi/4;
    elseif (d2(i,1) == 0 & d2(i,2)==1)
        phi7(i)=phi7(i-1)+3*pi/4;
    elseif (d2(i,1) == 1 & d2(i,2)==0)
        phi7(i)=phi7(i-1)+7*pi/4;
    else
        phi7(i)=phi7(i-1)+5*pi/4;
    end
end
np = [0 1; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0];
[r3,c3] = size(np);
phi6 = zeros(1,r3);
phi6(1) = phi7(r4)+3*pi/4;
for i=2:r3
    if (np(i,1)==0 & np(i,2)==1)
        phi6(i)=phi6(i-1)+3*pi/4;
    elseif (np(i,1)==0 & np(i,2)==0)
        phi6(i)=phi6(i-1)+pi/4;
    else
        phi6(i)=phi6(i-1)+pi/4;
    end
end
b = 10
% Define stop characters to define stop of data
% Concatenate preamble, pn, data stream, checksum, and stop characters
% Replicate each phi value N times to create phi
A 1200 Baud Phase Modulated Telephone Modem

\[
\phi_2 = \phi_5(i) \times \text{ones}(1,N);
\]
\[
\phi_1 = [\phi_1, \phi_2];
\]
\[
t = 0:T:((\text{length}((\phi_1)) - 1) \times T);
\]
\[
s = \sin(2\pi f_c \times t + \phi_1); \quad \% \text{Modulated waveform}
\]
\[
ex = \sin(2\pi f_c \times t + \phi_1(1)); \quad \% \text{Original unmodulated sinusoidal waveform}
\]

\text{sound}(s, fs) \% \text{Plays the sound from the soundcard}

\text{figure}(1)
\text{subplot}(2,1,1)
\text{plot}(t(1:200), ex(1:200))
\text{title}('\text{Original unmodulated waveform}')
\text{subplot}(2,1,2)
\text{plot}(t(1:200), s(1:200))
\text{title}('\text{Modulated waveform}')
\text{transmit} \% \text{After transmission, enter polling mode for acknowledgement response}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% \text{THE END} %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\textbf{Transmit.m}

\% \text{This is used in conjunction with Final_Modulate for ACK/NACK}

Fs = 12000;
\text{AI} = \text{analoginput}('\text{winsound}');
ch = \text{addchannel}((\text{AI},1));
duration = 20;
\text{set}((\text{AI}, '\text{SampleRate}'), Fs)
\text{set}((\text{AI}, '\text{SamplesPerTrigger}'), Fs \times \text{duration})
\text{set}((\text{AI}, '\text{TriggerChannel}'), ch(1))
\text{set}((\text{AI}, '\text{TriggerType}'), '\text{Software}')
\text{set}((\text{AI}, '\text{TriggerCondition}'), '\text{Rising}')
\text{set}((\text{AI}, '\text{TriggerConditionValue}'), 0.3)
\text{start}((\text{AI}))
\text{wait}((\text{AI}), \text{duration} + 10)
z = \text{getdata}((\text{AI}));
\text{figure}(2)
\text{plot}(z);
\text{title}('\text{Acknowledgement from receiver}')
\text{xlabel}('\text{Samples}')
\text{ylabel}('\text{Signal Level (Normalized)}')
\text{grid on}
delete(\text{AI})
clear \text{AI}
\text{for} \text{i} = 1: \text{length}(z)
    \text{if} \ z(i) > 0.3
        \text{clc}
        \text{fprintf}(''Data received with no errors. Ending session...'')
    \text{else}
        \text{clc}
        \text{wait}((\text{AI}), 30)
\end{if}

fprintf('(No data received or data received with errors. Retransmitting data...')
    Final_Modulate
end
end

MATLAB code for receive & demodulation:

Final_Demodulate.m:

% ECE406 - Capstone Senior Design II
% Design of a 1200 Baud Phase Modulated Telephone Modem
% Indiana University-Purdue University Fort Wayne
% Fall 2008
% Team: Saad Z. Khan & Aman P. Dasson
% Faculty Advisor: Elizabeth A. Thompson, Ph.D.

% This code demodulates the incoming modulated analog signal and outputs a text file containing the original source data.

Z=z.';     % Acquired signal
fc = 1800; % Carrier frequency in Hz
fs=12000;  % Sampling frequency--must be same as that of modulation
T=1/fs;    % Time period
t=0:T:(length(Z)-1)*T;
baud=1200; % Baud rate
N=fs/baud; % Number of samples in baud interval.
C=length(Z)/N;  % Number of baud intervals

% Eliminate noise at start of sequence prior to actual signal
for i=1:length(Z)
    if Z(i)> 0.3
        z_start_index=i;
        fprintf('Data acquisition started...
')
        fprintf('Starting index of data: %6.2f
',z_start_index);
        break;
    end
end

Z1=Z(z_start_index:end);     % Should eliminate noise by readjusting the index
t1=0:T:(length(Z1)-1)*T;
C1=length(Z1)/N;  % Number of baud intervals

% Determine optimum window alignment using preamble
D1=cos(2*pi*fc.*t1);
E1=sin(2*pi*fc.*t1);
for k=1:N
    for i=1:ceil(C1)-N % In case C is not an integer
        X1=Z1(start_index+k-1:end_index+k-1);  % shift by k
        X2=D1(start_index:end_index);
        X3=E1(start_index:end_index);
A 1200 Baud Phase Modulated Telephone Modem

X4=X1.*X2;  %Multiply by carrier waveform
% Note that cos(A)cos(B)=1/2*[cos(A+B)+cos(A-B)]
InPhase=sum(X4);  %Performs low pass filtering to eliminate 2nd cos term
X5=X1.*X3;
QPhase=sum(X5);  % Quadrature component
Magn(i)=sqrt(InPhase^2+QPhase^2);
end  % End of for loop
Sum_magn(k)=sum(Magn);
end

% index_max should be the index of Z1 at which maximum window alignment occurs. This is the baud interval alignment.
[max_mag,index_max]=max(Sum_magn);
fprintf('Maximum sum for window alignment: %6.2f\n',max_mag);
fprintf('New starting index after window alignment: %6.2f\n',index_max);
Z2=Z(z_start_index-1+index_max:end);
t2=0:T:(length(Z2)-1)*T;
D2=cos(2*pi*fc.*t2);
E2=sin(2*pi*fc.*t2);
C2=length(Z2)/N;  % Number of baud intervals

% Demodulate remainder of signal, which should consist of the remaining preamble, start characters, data stream, checksum and stop characters.
for i=1:floor(C2)  % In case C is not an integer
  % Extract N samples within the baud interval
  start_index=N*i-N+1;
  end_index=i*N;
  X1=Z2(start_index:end_index);
  X2=D2(start_index:end_index);
  X3=E2(start_index:end_index);
  X4=X1.*X2;  %Multiply by carrier waveform
  % Note that cos(A)cos(B)=1/2*[cos(A+B)+cos(A-B)]
  X=sum(X4);  %Performs low pass filtering to eliminate 2nd cos term
  X5=X1.*X3;
  Y=sum(X5);
  theta(i)=-atan2(Y,X);
end  % End of for loop

for i=2:length(theta)
  diff(i)=theta(i)-theta(i-1);  % Phase differences between baud intervals
end

B=ones(length(theta)-1,2);
for i=2:length(theta)
  if ((diff(i)>(3*pi/2)) & (diff(i)<(2*pi)))
    B(i-1,1)=1;
    B(i-1,2)=0;
  elseif ((diff(i)>(pi)) & (diff(i)<(3*pi/2)))
    B(i-1,1)=1;
    B(i-1,2)=1;
  elseif ((diff(i)>(pi/2)) & (diff(i)<(pi)))
    B(i-1,1)=0;
    B(i-1,2)=1;
  elseif ((diff(i)>(0)) & (diff(i)<(pi/2)))
    B(i-1,1)=0;
    B(i-1,2)=0;
  else
    B(i-1,1)=0;
    B(i-1,2)=0;
  end
end

44
A 1200 Baud Phase Modulated Telephone Modem

\[
B(i-1,1)=0;
B(i-1,2)=0;
\]
\[
\text{elseif } ((\text{diff}(i)>(3\pi/2-2\pi)) \& (\text{diff}(i)<(2\pi-2\pi)))
\]
\[
B(i-1,1)=1;
B(i-1,2)=0;
\]
\[
\text{elseif } ((\text{diff}(i)>(\pi-2\pi)) \& (\text{diff}(i)<(3\pi/2-2\pi)))
\]
\[
B(i-1,1)=1;
B(i-1,2)=1;
\]
\[
\text{elseif } ((\text{diff}(i)>(\pi/2-2\pi)) \& (\text{diff}(i)<(\pi-2\pi)))
\]
\[
B(i-1,1)=0;
B(i-1,2)=1;
\]
\[
\text{else}
\]
\[
B(i-1,1)=0;
B(i-1,2)=0;
\]
end
end

% B now contains the demodulated binary preamble, start characters, data
% stream, checksum and stop characters.

% Look for start characters pattern
count_start=0;
for k=1:length(B)
  if (B(k,1)==0 & B(k,2)==1)  % Start dibit characters
    count_start=count_start+1;
    if count_start == 12
      data_start_index=k+1;
      break;
    end
  end
end

% Discard preamble and start characters to obtain data, checksum and stop
% characters
B1=B(data_start_index:length(B),:);

% Search B1 for stop characters in data + noise following the data
count_stop=0;
for k=1:length(B1)-3
  if (B1(k,1)==0 & B1(k,2)==1 & B1(k+1,1)==0 & B1(k+1,2)==0 & B1(k+2,1)==0 & B1(k+2,2)==0 & B1(k+3,1)==0 & B1(k+3,2)==0)  % start character
    count_stop=count_stop+1;
    if count_stop == 3
      data_stop_index=k-9;
      break;
    end
  end
end

% Discard stop characters + noise following the data
B2=B1(1:data_stop_index,:);

% Extract checksum value
[r3,c3]=size(B2);
B3=B2(1:r3-9,:).'  % subtracting 9 rows->equiv to 18 bits
B4=B2(r3-8:r3,:);  % checksum value
y1=reshape(B4',1,18);
y2=num2str(y1);
y3=bin2dec(y2); % Should be restored checksum value in decimal

% Write binary data in B3 to a text file
fid = fopen('test2.txt','wt');
if fid < 0
    error('Failed to open data file for write')
end
count=fprintf(fid,'%i',B3);
fclose(fid);

% Convert from binary to ASCII
decode('test2.txt','test3.txt');
y = count_ascii_values('test3.txt'); %Assumed max length is 18 for a 3KB file
if y3==y
    % Send acknowledgement
    fprintf('Checksum matched. No errors detected. Sending acknowledgement...');
    t=0:T:200000*T;
y4=sin(2*pi*fc*t);
sound(y4,fs)
else
    record % If checksum does not match, resume polling for data
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Record.m

% This is used to poll for incoming data, capture, demodulate and ACK/NACK

clear all
clc
Fs = 12000;
AI = analoginput('winsound');
ch = addchannel(AI,1);
duration = 15;
set(AI,'SampleRate',Fs)
set(AI,'SamplesPerTrigger',Fs*duration)
set(AI,'TriggerChannel',ch(1))
set(AI,'TriggerType','Software')
set(AI,'TriggerCondition','Rising')
set(AI,'TriggerConditionValue',0.3)
start(AI)
wait(AI, duration+5)
z = getdata(AI);
figure(1)
plot(z);
title('Captured data')
xlabel('Samples')
ylabel('Signal Level (Normalized)')
grid on
A 1200 Baud Phase Modulated Telephone Modem

delete(AI)
clear AI
Final_Demodulate
figure(2)
plot(diff,'.');
title('Phase angle differences between baud intervals')
xlabel('Index')
ylabel('Radians')
y% = 20*log10(abs(fft(z(10000:20000))));

Function codes:

Code.m

spc 00100000
ent 00001010
! 00100001
# 00100011
$ 00100100
% 00100101
& 00100110
' 00100111
( 00101000
) 00101001
* 00101010
+ 00101011
, 00101100
- 00101101
. 00101110
/ 00101111
0 00110000
1 00110001
2 00110010
3 00110011
4 00110100
5 00110101
6 00110110
7 00110111
8 00111000
9 00111001
: 00111010
; 00111011
< 00111100
= 00111101
> 00111110
? 00111111
@ 01000000
A 01000001
B 01000010
C 01000011
D 01000100
E 01000101
F 01000110
G 01000111
H 01001000
I 01001001
A 1200 Baud Phase Modulated Telephone Modem

```
J   01001010
K   01001011
L   01001100
M   01001101
N   01001110
O   01001111
P   01010000
Q   01010001
R   01010010
S   01010011
T   01010100
U   01010101
V   01010110
W   01010111
X   01011000
Y   01011001
Z   01011010
[   01011011
\   01011100
]   01011101
^   01011110
_   01011111
`   01100000
a   01100001
b   01100010
c   01100011
d   01100100
e   01100101
f   01100110
g   01100111
h   01101000
i   01101001
j   01101010
k   01101011
l   01101100
m   01101101
n   01101110
o   01101111
p   01110000
q   01110001
r   01110010
s   01110011
t   01110100
u   01110101
v   01110110
w   01110111
x   01111000
y   01111001
z   01111010
{   01111011
|   01111100
}   01111101
~   01111110

Encode.m
```
function encode(File,File2)
% ENCODE Converting ASCII text to Binary numbers
% DECODE(FILE,FILE2) where FILE is an input text file consisting of ASCII text and
% FILE2 is the output file.
% The code key file (code.m) should be included in the same directory in order
% for this program to run.

wpl=0;
[spc_ent,C0]=textread('code.m','%s %s',2);
[L,C]=textread('code.m','%s %s','headerlines',2);
L=char(L);
FID = fopen(File,'r');
OUT = fopen(File2,'w');
while 1
tline = fgetl(FID);
  if ~ischar(tline), break, end
  for i=1:size(tline,2)
    x=find(L==tline(i));
    if isempty(x)==1
      fprintf(OUT,'%s',char(C0(1)));
      wpl=wpl+1;
    else
      fprintf(OUT,'%s',char(C(x)));
      wpl=wpl+1;
    end
    if wpl==10, fprintf(OUT,'
');, wpl=0;, end
  end
  fprintf(OUT,char(C0(2)));
  wpl=wpl+1;
end
fclose('all');

Decode.m

function decode(File,File2)
% DECODE Converting Binary numbers to ASCII text
% DECODE(FILE,FILE2) where FILE is an input text file consisting of binary numbers
% and FILE2 is the output ASCII text file.
% The code key file (code.m) should be included in the same directory in order
% for this program to run.

t=0;
[spc_ent,C0]=textread('code.m','%s %s',2);
[L,C]=textread('code.m','%s %s','headerlines',2);
L=char(L);
bit = length(char(C(1)));
FID = fopen(File,'r');
OUT = fopen(File2,'w');
while 1
tline = fgetl(FID);
  if ~ischar(tline), break, end
  for i=1:size(tline,2)/bit
A 1200 Baud Phase Modulated Telephone Modem

```matlab
for j=1:93
    x=isequal(char(C(j,:)),tline(8*i-7:8*i));
    if x==1
        t=j;
    end
end
if t~=0
    fprintf(OUT,'%s',L(t));
elseif tline(8*i-7:8*i)==char(C0(2))
    fprintf(OUT,'
');
elseif tline(8*i-7:8*i)==char(C0(1))
    fprintf(OUT,' ');
end
end
fclose('all');
```

`Count_ASCII_values.m`

```matlab
function c=count_ascii_values(File)
    % Revised from encode.m
    %Counting ASCII sum value
    FID = fopen(File,'r');
    x=0;
    while 1
        tline = fgetl(FID);
        if ~ischar(tline), break, end
        [r1,c1]=size(tline);
        for i=1:c1
            x=x+double(tline(i));  % x should contain the decimal sum of all ASCII values
        end
    end
    c=x;  % decimal count
    fclose('all');
```