HIGH THROUGHPUT VIA LASER DIODE

By

Muhammad Farhan Shabir

Muhammad Shayan

Usama Hussain

Ejaz Ahmed

Supervisor: Dr. Alina Mirza

Submitted to Faculty of Department of Electrical Engineering,

Military College of Signals, National University of Sciences and Technology, Islamabad

in partial fulfilment for the requirements of a B.E Degree in

Electrical (Telecom) Engineering

JUNE 2019

بسم الله الرحمن الرحيم

In the name of Allah, the Most Beneficent, the most Merciful

CERTIFICATE FOR CORRECTNESS AND APPROVAL

It is certified that work contained in the thesis – High Throughput Via LASER Diode carried out by Muhammad Farhan Shabir, Muhammad Shayan, Usama Hussain and Ejaz Ahmed under the supervision of Dr. Alina Mirza for partial fulfilment of Degree of Bachelor of Electrical (Telecom) Engineering is correct and approved. The plagiarism is 12 % including references.

Approved By

Dr. Alina Mirza

Department of EE, MCS

Dated: 14th June, 2019 ____________

ABSTRACT

HIGH THROUGHPUT VIA LASER DIODE

The project basically revolves around the concept of Visible light communication (VLC) which was put forward by Professor H. Hass in Japan. Visible light communication is a subset of optical wireless communication. Visible light communication uses light wave's frequency spectrum which has the range of 300KHz to 900GHz.

The project basically aims to make the hardware and software combination for the secure transmission and reception of data from one point to another point remotely using a LASER diode. Real time secure transmission and reception of data will be done through a microcontroller and a Graphical User Interface. The data we are transmitting includes audio file, real time voice signal, text message, text file and image. Graphical User Interface will be running on Visual Basic while the coding of microcontroller will be done through Arduino IDE editor. AES 128 bit encryption technique will be used to encrypt the data. Encryption will be done in microcontroller for security purpose. The laptop or computer connected to hardware will transmit data via a LASER diode to the other laptop or computer which will also be connected to a hardware. Graphical User Interface will be running on both the laptops which will allow the transmission and reception of text message, text file and image, while the transmission and reception of audio file and real time voice signal will be done in hardware portion.

DECLARATION

No portion of the work presented in this thesis has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

To our parents, teachers and all those individuals who have helped us accomplishing our goals and have greatly contributed to betterment of our life and who are of great inspiration and motivation to us.

Table of Contents:

List of Figures:

List of Tables:

Appendix ‘A

List of Abbreviations

References

Bibliography

Chapter 1: Introduction

Chapter 1: Introduction

1.1 Overview:

Visible light communication is another subset for end to end communication. Visible light communication includes a transmitter, channel and a receiver [1]. LASER diodes have high modulation bandwidth, efficiency and beam convergence hence they are rated above LEDs for data transmission. Data is basically transmitted and received in the form of bits. The targeted market of LASER diode communication can vary from institutions to corporate sectors. We are determined to achieve high data rates by using LASER diode [2-3].

1.2 Problem Statement:

Now a days normally radio frequencies are used for transmission of data from one point to another point. The bandwidth of radio frequencies ranges between 3 KHz to 300MHz which is very less as the number of users are increasing day by day. Moreover the RF communication suffers from interference and security issues [4]. In this modern era everyone needs a secure and a good speed data transmission and reception.

1.3 Approach:

Our project will contain both hardware and software portion. In the hardware portion, the LASER diode will be attached to the transmitter, which will be connected via Arduino to the computer. On the receiving end a photodiode will be used for receiving bits coming from LASER diode and then conversion of that bits to original signals through Arduino and output will be shown on the receiving computer. We will also transmit audio file including real time voice signal and receive it through a speaker at the receiver end. As the medium used is visible light so we will be using convex lens for amplification of LASER beam. To tackle any corners and obstacles, we will be using prisms of different angles to make our communication successful.

In Software portion, we will be working on GUI. We will make two GUIs for both transmitter and receiver computers. The language of GUI will be Visual Basic. As we are dealing with security of our data also so we will be encrypting our data with the latest encryption technique i.e. AES 128 bit. The encryption will be done in Arduino after the data is converted from its original form to bits.

There are two block diagrams for our project which are given below:

Case-1 Straight Line:

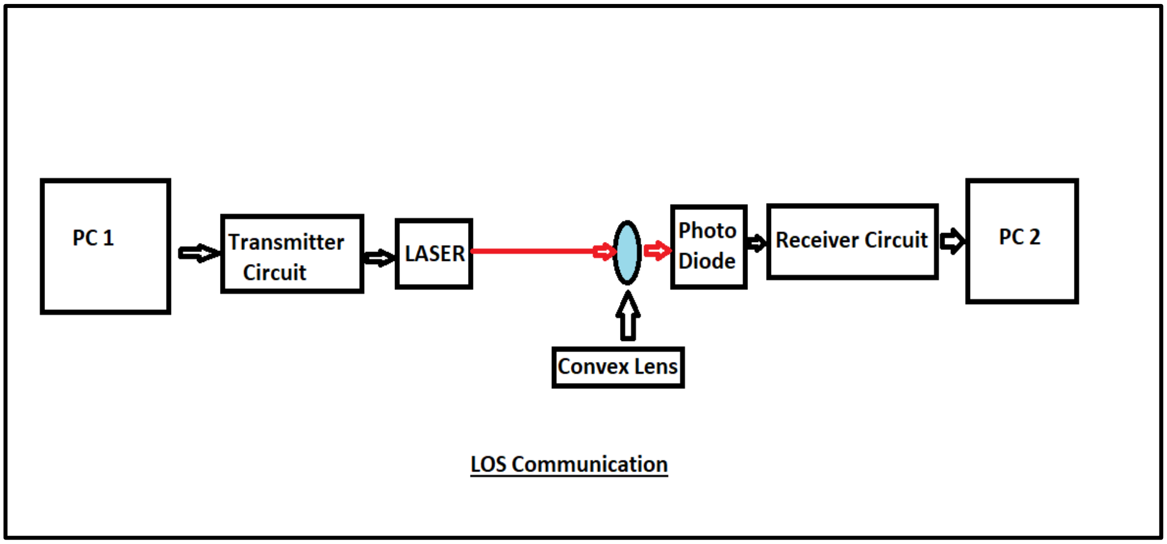


Figure 1-1: Block Diagram Straight Line

Case-2 Different Angles:

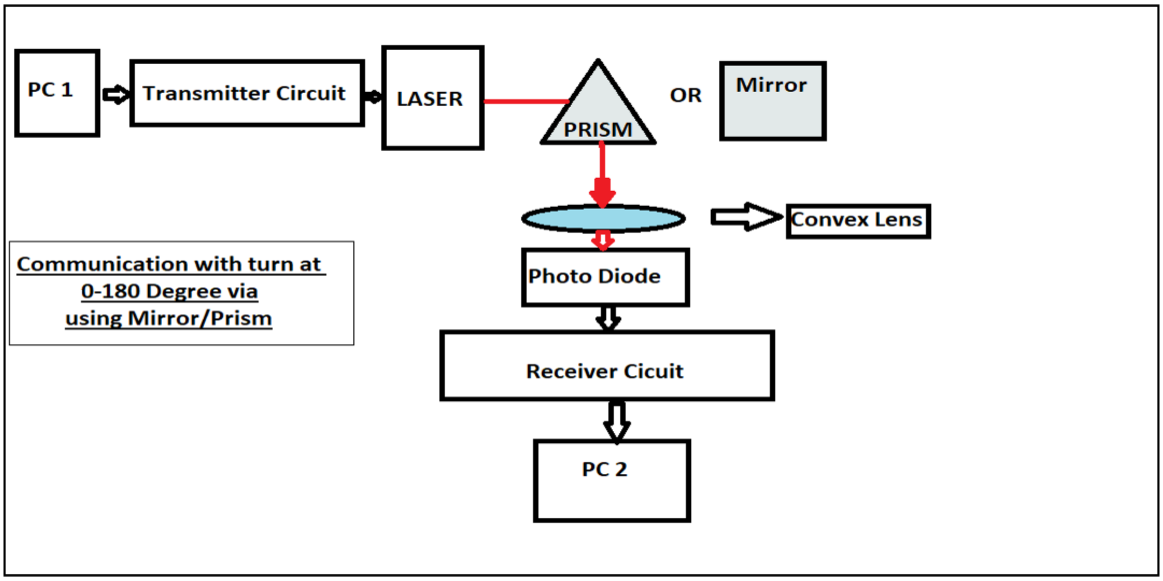


Figure 1-2: Block Diagram at Different Angles

1.4 Objectives:

The key objectives of our work are:

- We intend to make up a portable setup for secure data communication between two nodes/PCs.
- To make a user friendly GUI.
- To help the users to transmit and receive data where line of sight may be the issue (to tackle any corners or obstacles) [1].
- We are using AES 128 bit encryption technique to secure our transferred text.

1.5 Deliverables:

Table 1-1: Deliverables

SR	Tasks	Deliverables
1	Literature Review	Literature Survey and Feasibility Analysis
2	Requirements Specification	Software/Hardware Requirements Specification document (SRS)
3	Detailed Design	Software/Hardware Design Specification document
4	Implementation	Project demonstration
5	Testing	Evaluation plan and test document

1.6 Overview of Document:

1.6.1 Purpose:

This document covers detailed review of all major steps involved in development of a setup which will be used for data communication via LASER diode. These all involved steps acted as guide to the development team and now shall provide insight to the reader that how prototype idea was formulated and then how hardware integration took place, how software was designed and finally tested.

1.6.2 Organization of Document:

- In chapter 1 an Introduction to document and system is provided.
- Chapter 2 covers the requirement specifications part and covers Functional, Non-Functional Parts Requirements, resources required, and constraints involved
- Chapter 3 covers the Design Specifications which provide an in-depth view of how the systems is developed and how the functionalities are distributed.
- Chapter 4 discusses the hardware implementation of our project.
- Chapter 5 gives detail about software implementation.
- Chapter 6 is all about conclusion and future work.

Chapter 2: Literature Review

Chapter 2: Literature Review

2.1 Preamble:

Now a days normally radio frequencies are used for transmission of data from one point to another point. The bandwidth of radio frequencies ranges between 3 KHz to 300MHz. Moreover the radio frequencies communication suffers from interference, security and health issues [4]. In this modern era everyone needs a high speed data transmission and reception without any interference, health and security issues.

LASER light communication system will communicate data using high speed [2-3] and encryption. The LASER diode will be attached to the transmitter, which will be connected via Arduino to the computer. On the receiving end a photodiode will be used for receiving LASER and conversion to electrical signals which will then be processed through Arduino and output will be given on the receiving computer. Audio signal will also be transmitted which have a MIC at transmitter and a speaker at receiver. As the medium used is visible light so we will be using optics for amplification and redirection of LASER beam.

Research on such Visible Light Communications (VLC) started in Japan, where the Visible Light Communications Consortium (VLCC) has been in presence for quite a while. Intrigue is presently developing quickly, both in Asia and Europe, where the Remote World Research Forum has worked here [3-4]. The appearance of Visible light communication opens up another probability for future communication between any sort of gadgets [8]. Visible light range transmission capacity, which ranges from 430 THz to 750 THz is a lot bigger than the radio frequency range data transmission going from 3 KHz to 300 GHz. Because of this bigger data transfer capacity, it is conceivable to accomplish an a lot higher rate of information exchange and furthermore a vast number of clients can be suited [4-5].

Following are the few excerpts from the research papers related to the project:

In [1], author discussed the maintenance of line of sight for a visible light communication system and key issues of line of sight. A basic line of sight channel model if required to characterize the path loss and received power is needed to optimize the receiver. In this paper author also showed a practical design of visible light communication system while maintaining line of sight.

In [2], author describes that in recent experiments 10 Gb/s of data has been transferred through LEDs but LEDs have some disadvantage that LEDs have tradeoff between optical efficiency and bandwidth. So LASER diodes can be a best alternative for better implementation of visible light communication system. In this paper author also describes capabilities of LASER diodes in many situations.

In [3], author describes some properties of red LASER light. In Visible light communication system slow white LEDs must be replaced by LASER diodes, most preferably red in color. Red LASER can improve the speed of data rate coverage and light quality.

In [4], authors described that the Radio Frequency (RF) communication experiences obstruction and high idleness issues. Alongside this, RF communication requires a different setup for transmission and receiving of RF waves. Over-coming the above impediments, Visible Light Communication (VLC) is a favored communication system as a result of its high data transmission and resistance to obstruction from electro-magnetic sources. In this paper there is also a review of the potential applications, design, adjustment procedures, institutionalization and research difficulties in VLC.

In [5], authors have discussed that wireless optical communication has been improved with high proficiency, vast data transfer capacity, and high uniform white light source turns out to be more and increasingly significant. . In this paper author described that a white light source produced by red, green, and blue laser diodes (RGB LDs) was synthesized as indicated by the determined power ratio of RGB LDs in view of the chromaticity hypothesis. The high rationality of the lasers typically prompts the non-uniform white light and it demonstrates that the transfer speed was more than 1 GHz, which is restricted by the photo detector cut-off recurrence.

In [6], author discussed the properties of LED used for visible light communication. The advantages of visible light communication are also discussed. Author has also discussed the influence of interference and reflection including path loss using numerical analysis which concludes that the system for visible light communication using white light LEDs is only for indoor communication.

In [7], author describes about modulation and dimming schemes of visible light communication systems. Visible light communication alludes to short range optical wireless communication utilizing visible light range from 380 to 780 nm. Empowered by ongoing advances in LED innovation, IEEE 802.15.7 backings high-information rate visible light communication up to 96 Mb/s by quick tweak of optical light sources which might be dimmed during their activity. IEEE 802.15.7 gives diminishing versatile systems to flash free high-information rate visible light communication.

In [8], author described and designed one of the application of visible light communication, which is road to vehicle communication. Designed proposed by author is that the LED traffic light of car is a transmitter and a camera is a receiver. This system enables multi channeling. LED transmitters organized in the state of a plane are balanced independently, and a camera is utilized as a collector for demodulating the sign by utilizing image processing techniques.

In [9], author described experimentally under water visible light communication. Data has been transmitted up to 2.5 meter underwater using different bit rates and modulation schemes. According to author bit error rate was also calculated in several hours. The system proposed by the author include to low cost LEDs at transmitter side and an photodiode at receiver side.

In [10], author described that LEDs are conventionally used for visible light communication, but the need of data transfer has been increased to gigabytes. To achieve this high data rate LEDs can be replaced by LASER diodes which will provide more focused and high speed data transfer. LASER diodes also have hight modulation beam width, efficiency and beam convergence. This paper also compares the unique characteristics of LASER diodes and LEDs.

In [11], author described and demonstrate high speed visible light communication using a blue color of LASER combined with remote phosphorous. This strategy both generates white light and supports multi Gb/s communications. Data rates of up to 5.62 Gb/s and 6.52 Gb/s were accomplished by utilizing OFDM with fixed-rate and adaptive loading approaches, individually. An all-out information rate of 10 Gb/s was observed to be attainable by utilizing a two channel imaging framework.

In [12], author has discussed, visible light communication as an alternative way of wireless communication. VLC can replace radio frequency communication with high data rate downlink communication for indoor places like offices, schools, organizations etc. author also discussed that in large scale commercialization of visible light communication will depend upon the fast engineering solutions and high processing speed.

In [13], author has discussed the achievements and trends in visible light communication systems. A bidirectional system has been implemented using LEDs achieving a high data rate. Technical challenges for visible light communication are also discussed by author while implementing a system. VLC can replace radio frequency systems using high engineering skills with great processing speed.

2.2 Conclusion:

Visible light communication includes a transmitter, channel, and a receiver. By analyzing different research papers and journals from IEEE and other technical references we have concluded that LASER diodes have high modulation bandwidth, efficiency and beam convergence hence they are rated above LEDs for data transmission. Data is basically transmitted and received in the form of bits and these bits are then converted in to common language.

Chapter 3: Design Requirements

Chapter 3: Design Requirements

3.1 Introduction:

The system requirement and specification for High throughput via LASER diode are covered in this chapter. This chapter is meant to outline the features and requirements of our project, to serve as a guide to the concerned people on one hand and a software validation document for the prospective client/stakeholders on the other.

3.2 Overall Description:

The idea of this project is to transmit and receive encrypted data from one point to another point through a LASER diode. The data will be in the form of message, image, text file, audio file and real time voice signal. To transmit and receive file, image and message, we have also made two GUIs in Visual Basic language for our project.

3.3 Product Features:

The key features of LASER diode communication are as follows:

1. Encrypted data will be transmitted and received via LASER diode.
2. Resolving the main issue of bandwidth by using visible light spectrum.
3. A graphical user interface for the users for easy transmission and reception of their files.
4. Use of convex lens and prisms/mirror for amplification and redirection of LASER beam.

3.4 Operating Environment:

The sub-sections below give a brief description of environment, hardware & software-based requirements for the operation of our project.

3.4.1 Hardware:

Our project will operate with the following hardware:

Arduino Microcontroller: It will convert the analogue signal to digital signal coming from LASER diode and vice versa at receiver end. We are using ATmega328 for this purpose. Two microcontrollers will be used for transmission and reception of data.

USB Power Cable: It is used to power the microcontroller from PC.

Printed Circuit Board: We have implemented the whole circuit diagram on printed circuit board. Two printed circuit boards have been used as a transmitter and receiver.

LASER Diode and Photo Diode: LASER diode will be used for throwing data bits from transmitter side to receiver side where Photo diode will sense that bits and then transmit it to microcontroller for further process.

Convex Lens: We have used convex lens for converging our light beam at a single focusing point and working as an amplifier for our project. It will reduce the noise factor. Basically when we transmit our data through LASER diode, the light diverges from its path due to presence of white light in surroundings and expands due to long distance. Intensity of light can get change at receiver end. So we need to have a good focused point of LASER light at the receiver end. For this purpose we are using convex lens in front of our photodiode so we can converge our light at a single focus point.

In our project we have used a transparent color, spherical convex lens which has a diameter of 50mm and focal length of 3.5 cm. Without convex lens we have got noise in our results as the light gets diverged due to the presence of white light in surroundings. When we have attached convex lens in front of photodiode our data reception became more accurate and with less noise or distortion.

Figure 3-1: Convex Lens [14]

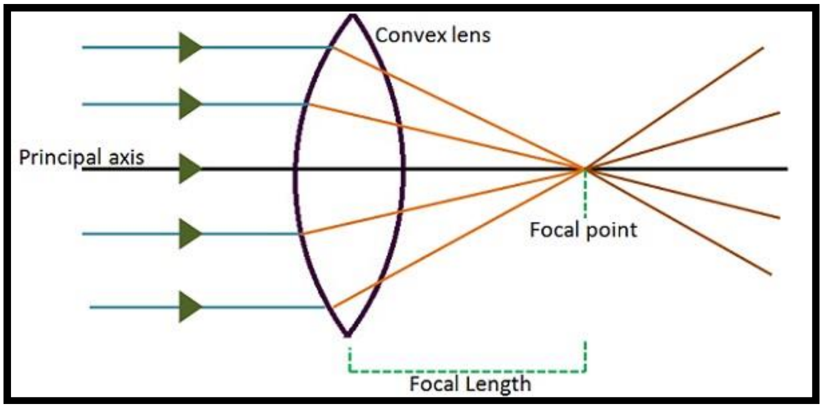


Figure 3-2: Convex Lens Working Diagram [15]

Prism: Prism is used for tackling the corners and obstacles for maintaining line of sight. When someone desires to get his data transmitted to another room or corridor, prism is used to maintain line of sight and get the data transmitted successfully. In our project we have use right angle prism with size of 5.0*5.0 mm and 2 arcmin 90 degree angle tolerance with 1 arcmin pyramidal tolerance. This prism produce inverted or reverted left handed light coming from LASER diode, depending on the orientation of the prism.

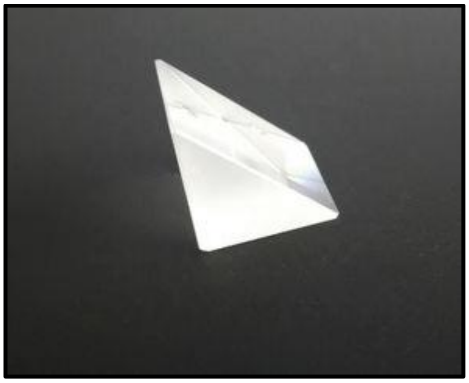


Figure 3-3: Right Angle Prism [16]

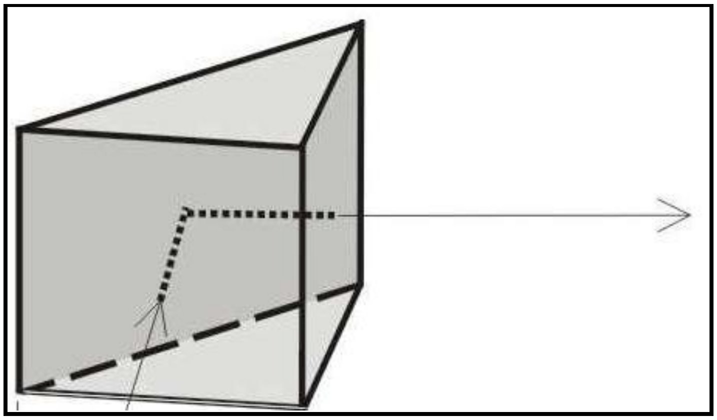


Figure 3-4: Right Angle Prism Working Diagram [17]

Mirror: Mirror is a reflecting surface, typically made of glass in which one side is shined and other is dull. Mirror has the ability to reflect light at different angles. In our project, we have used mirror to cover the different angles of light from 30 degrees to 180 degrees. When the light comes from LASER diode at these angles, mirror is used to reflect the light at convex lens which is attached in front of photodiode.

Figure 3-5: Mirror [18]

Block Diagram at Different Angles: In order to cover edges and hurdles we have used prism and mirror. Prism diverts light at an angle of 90 degrees and we used mirror for diverting light at 0 degrees to 180 degrees.

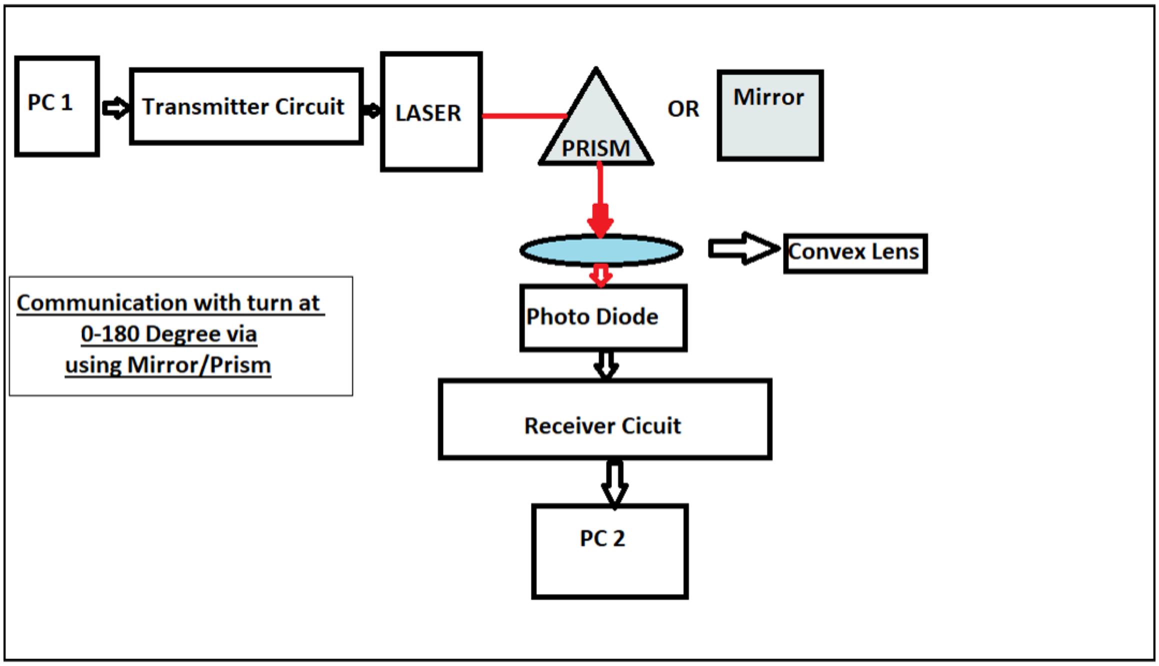


Figure 3-6: Mirror, Convex Lens and Prism Working Diagram

3.4.2 Software:

- **Visual Basic:** Visual Basic is a third generation programming language in which we have made our GUIs. We have implemented VB6 to make our GUIs for project. All the code written for GUIs are given in Appendix 'A.
- **Arduino IDE:** It is an open source software to write and upload the code to a microcontroller. It can be operate able in Windows, Mac and Linux. We have used Arduino IDE to write code for our transmitter and receiver portion to convert the data into bits and vice versa. Also we have included our AES 128 bit encryption and decryption in Arduino IDE coding. All the code is given in Appendix 'A for both transmitter and receiver microcontroller.
- **Diptrace:** All the schematics design of our circuit diagrams have been done in Diptrace. It is a tool to design your circuit according to your own wish. You can get PCB layout for all of your designed circuits in Diptrace.
- **CH341:** This is a tool for recognition of port in the laptop. When we connect our hardware with laptop we need to identify the port number in GUI. With this

software installed in our laptop we can simply go to device manager of our laptop and can easily get the port number on which the usb of our hardware is connected.

3.4.3 Design and Implementation Limitations:

- Line of sight is mandatory for completing the communication process at both ends.
- Gain can be the issue for different environment due to different light intensity at different places.
- Serial communication is limited to 9600 baud rate.
- Our project is half duplex means that it can't transmit and receive data at the same time.

Chapter 4: Hardware Implementation

Chapter 4: Hardware Implementation

4.1 Introduction:

This chapter describes the hardware design of project ‘High Throughput Via LASER Diode’. The chapter is meant to detail the design of features and requirements of our project, to serve as a guide to the users on one hand and a software validation document for the prospective client on the other. It also includes detailed descriptions, sequence diagrams and various other figures.

4.2 Components Used:

Following are the components used and their description:

4.2.1 Arduino Microcontroller ATmega328:

ATmega328 is a microcontroller which will be used in this project. Following are the specifications of ATmega328:

- 28 pins
- 8 bit AVR CPU
- 32kb memory
- 10 ADC bits
- 8 ADC channels
- 6 PWM pins
- 23 I/O pins
- 20Mhz Oscillator
- Flash memory type
- 3.3V-5.5V operating voltage

Figure 4-1: ATmega328 Microcontroller [19]

4.2.2 LASER Diode:

This is the main component of our project. We have used a red light LASER diode for transmitting our data to the receiver end. The specification of our LASER diode is 650nm and 6mm.

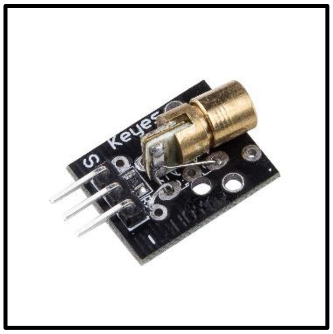


Figure 4-2: LASER Diode [20]

4.2.3 Photodiode:

A photodiode is used at the receiver end for sensing the bits coming from LASER diode. The bits after sensing will be transmitted to the receiver microcontroller from photodiode, from where it will be again converted into our data.

Figure 4-3: Photodiode [21]

4.2.4 LM-386:

LM-386 is the main component for transmitting and receiving of audio signal. Detail of LM-386 is given below in next topic.

Figure 4-4: LM-386 [22]

4.2.5 Arduino UNO: It contains

- A Microcontroller Board
- 14 digital I/O Pins
- 6 Analog I/P
- 16 MHz Quartz Crystal
- A USB Connection Port
- A Power Jack
- A Reset Button

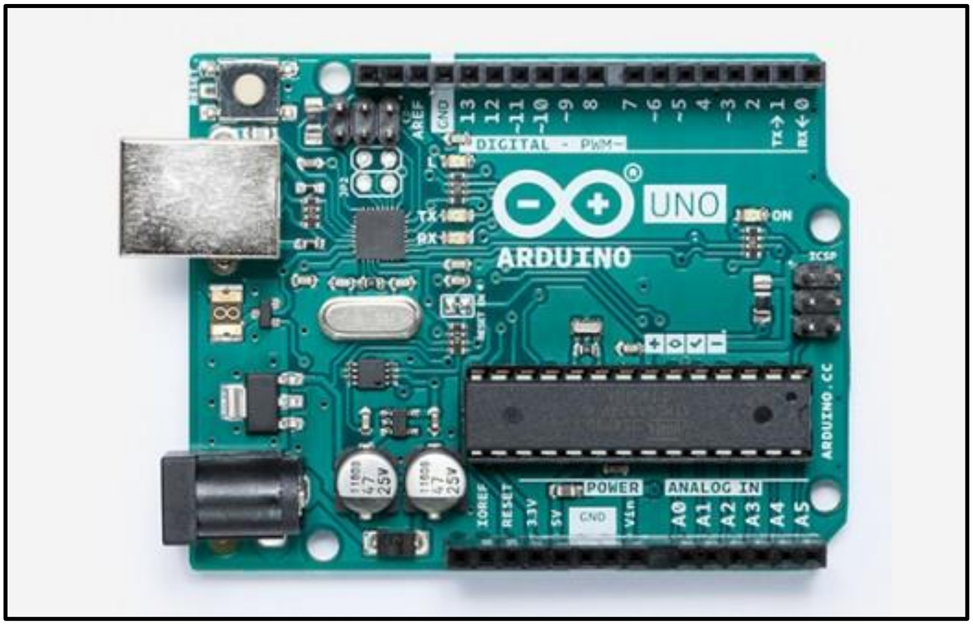


Figure 4-5: Arduino UNO [23]

4.2.6 Speaker:

We have used 8-ohm 1 Watt speaker for our audio signal output at receiver end. The speaker have also a gain of 10k variable resistor which will control the output sound of audio file.

Figure 4-6: Speaker [24]

4.2.7 LCD Display:

LCD display is used for displaying the transmission and reception mode of our data i.e. audio mode and simple mode.

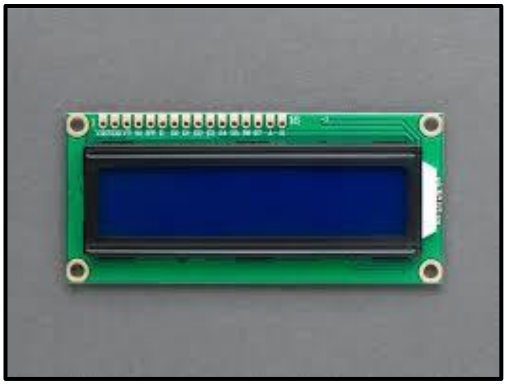


Figure 4-7: LCD Display [25]

4.3 Implementation Methodology:

Following methodology is used while implementing and integrating the hardware of High Throughput Via LASER Diode.

4.3.1 Hardware design and its implementation:

We have designed two modules in hardware side; one is transmitter module which sends data received from GUI and audio through LASER. Second is receiver module which receives data sent from transmitter side.

4.3.1.1 Transmitter Hardware design and implementation:

Transmitter module sends data received from GUI and audio through LASER. LASER is connected to Arduino which is a micro controller. Arduino receives data from GUI and then forward it and operates the LASER.

Transmitter module basically consists of two sub modules one for sending audio and second for receiving data from GUI and then forward it to LASER.

4.3.1.2 Audio sub module:

Audio sub module is again in two parts; one for transmitter and second for receiver. If we want to send audio we have made a switch which will change the function of transmitter. The main component in the audio modules is LM-386 IC.

4.3.1.3 LM-386:

LM-386 is the main and important component in our audio sub modules. Pin Configuration is as following.

Table 4-1: LM-386 Pin Configuration

Pins	Configuration
1	Gain
2	Inverting I/P
3	Non Inverting I/P
4	Ground
5	Output
6	VCC
7	Bypass
8	Gain

4.3.1.4 Transmitter Audio sub Module:

The input of audio is given by audio jack which is connected to PC or mobile phone. Variable resistor at pin 3 of LM-386 is for control of input signal. Capacitor and variable resistor at pin 1 and 8 of LM-386 is for gain control between 20-200. At pin 7 there is a bypass capacitor. At the end at pin 5 there is the output circuit through which LASER is connected giving output converting electrical signals to light signals. The Proteus design for transmitter audio module is in fig 4.8.

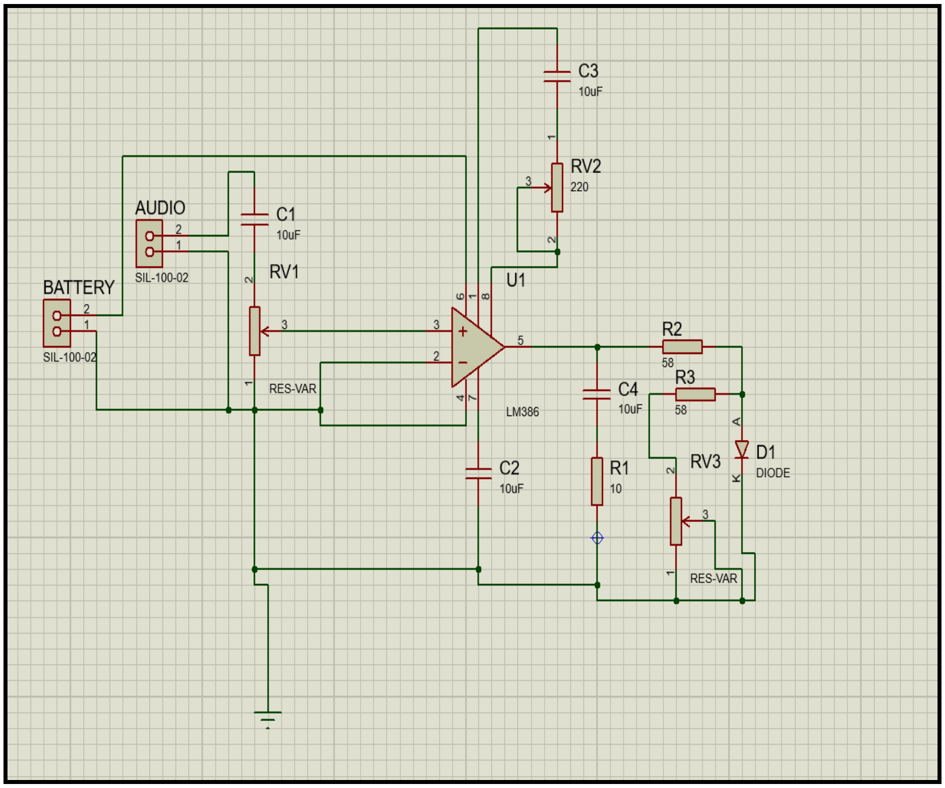


Figure 4-8: Schematic Diagram for Transmitter Audio Sub Module

4.3.1.5 Data Sub Module:

Data sub module is also in two parts; one is for transmission and second is for reception. Data sub module at transmitter side basically takes input from GUI and then send the output through LASER. Data sub module at receiver basically takes input from photo diode which and then sends received data to GUI, where received data will be displayed.

4.3.1.6 Transmitter Data Sub Module:

The main components in transmitter data sub module are Arduino and LASER. Data which is sent on COM port from GUI is transferred to Arduino. The USB port is connected to pin 2 and 3 of Arduino for reception and transmission respectively. Pin 9 and 10 of Arduino is connected with local oscillator. Pin 15, 16, 17 and 19 is connected to LCD display. The LASER is connected to pin 2 of the Arduino which is the receiver pin. LASER is also operated by Arduino.

4.3.1.7 Basic Working:

When the data is sent from GUI to USB port, Arduino is connected to USB port via pin 2 and 3, from where the data is received by Arduino which is converted in the form of bits. LASER is connected to pin 2 of Arduino which takes input from Arduino in the form of bits and goes in on state when bit 1 is sent and off state when bit 0 is sent. The LCD displays shows the information we already stored in Arduino.

4.3.1.8 Integration of Data Sub Module with Audio Sub Module at Transmitter Side:

The audio sub module is integrated with data sub module. The whole circuit of audio sub module of transmitter side is connected to pin 13 of the Arduino. We used a switch at pin 1 which will be used for switching between data and audio transmission.

4.3.1.9 Schematic Design of Transmitter Module as a Whole:

The overall implemented transmitter design on printed circuit board is depicted in fig 4.9

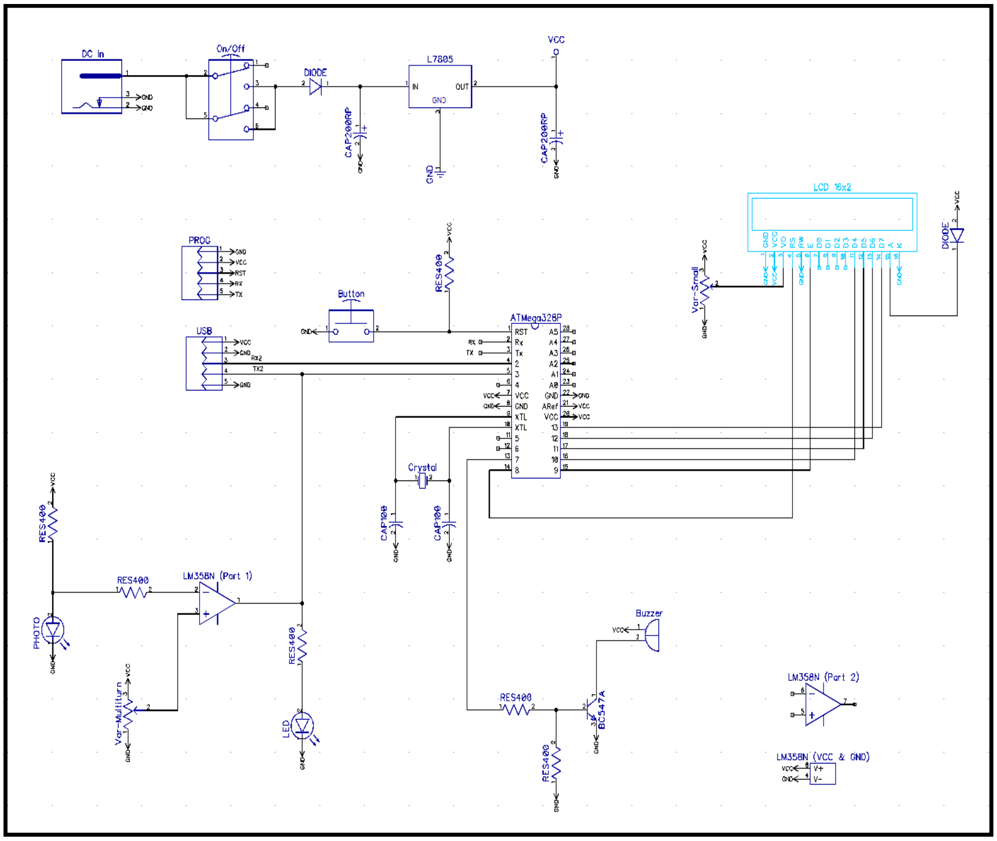


Figure 4-9: Schematic Diagram for Transmitter as A Whole

4.3.2 Receiver Data Sub Module:

The main components in receiver data sub module are Arduino and photo diode. Data which is sent from laser is transferred to Arduino by photo diode. Basically photo diode is sensing the data from LASER and Arduino is converting data from digital to analogue form.

4.3.2.1 Receiver Audio Sub Module:

At the receiver side, light of LASER is being detected by photo-cell which is converting light signals to electrical signals. At pin 3 of the LM-386 input signal is coming, similarly at pin 1 and 8 the capacitor is for maintaining gain and at pin 5 the output is taken on speaker (8 ohm). Speaker is converting electrical energy in form of voice. The Proteus design is as following:

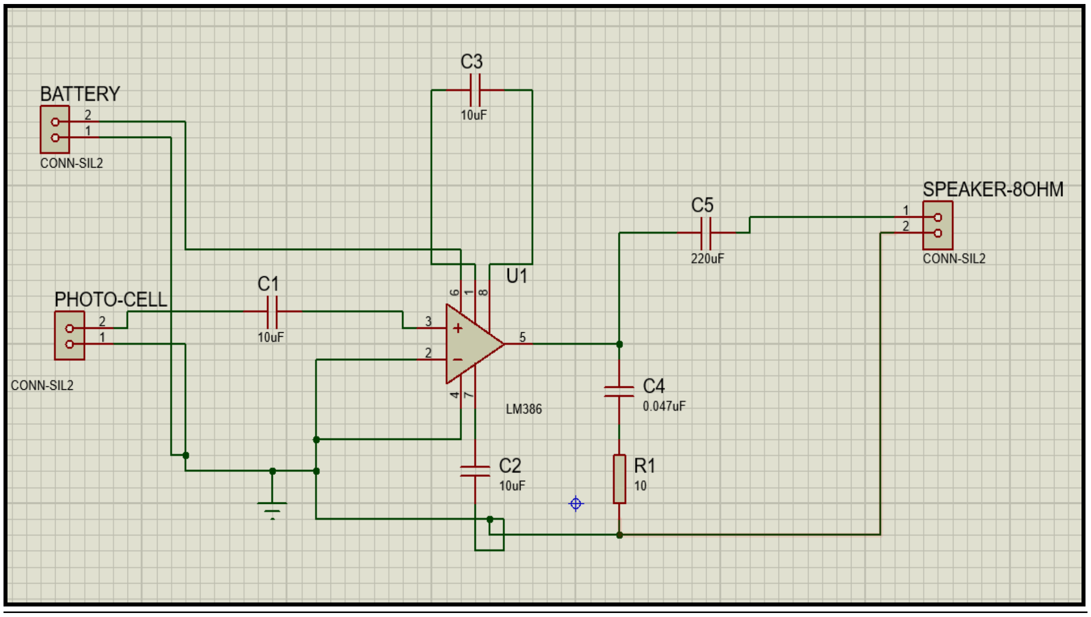


Figure 4-10: Schematic Diagram for Receiver Audio Sub Module

The both audio sub modules are then integrated with data sub module. We used switch between both modules to change the mode of transmission and reception.

4.3.2.2 Basic working:

In the receiver sub module we have a USB port, Arduino microcontroller, speaker and a LCD display. Pin 2 of Arduino is used for reception of Data. Pin 7 is providing VCC, Pin 8 and 22 are grounded, Pin 4 and 5 are connected to the USB for reception and transmission of Data respectively. Pin 18 is connected to the crystal oscillator whereas 14, 15, 16, 17, 18 and 19 are connected to LCD display. Different resistors and capacitors are used to maintain the gain in control.

4.3.2.3 Integration of Data sub module with audio sub module at receiver side:

For audio reception we have used LM-386 IC which is also connected to photo diode. The IC is connected to pin 5 of Arduino microcontroller. A switch is used at pin 1 which will be used for switching between data and audio reception. A variable resistor is connected to the speaker which is used to control the output of audio signal.

4.3.2.4 Schematic Diagram of Receiver Module as a Whole:

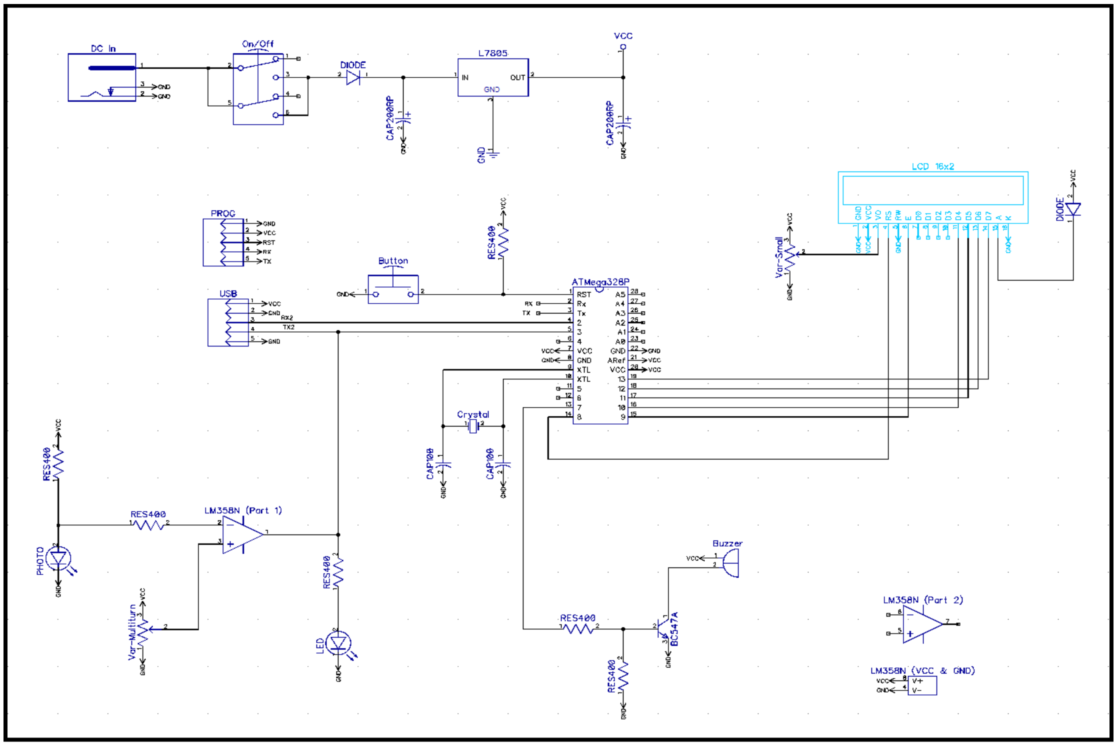


Figure 4-11: Schematic Diagram for Receiver As A Whole

4.3.3 ATmega328 Pin Configuration:

Table 4-2: ATmega328 Pin Configuration

Pin	Config	Pin	Config	Pin	Config	Pin	Config
1	Reset	8	Gnd	15	LCD	22	Gnd
2	Rx	9	Crystal	16	LCD	23	-
3	Tx	10	Crystal	17	LCD	24	-
4	USB Rx	11	-	18	LCD	25	-
5	USB Tx	12	-	19	LCD	26	-
6	-	13	RES	20	VCC	27	-
7	VCC	14	LCD	21	VCC	28	-

Chapter 5: Software Implementation

Chapter 5: Software Implementation

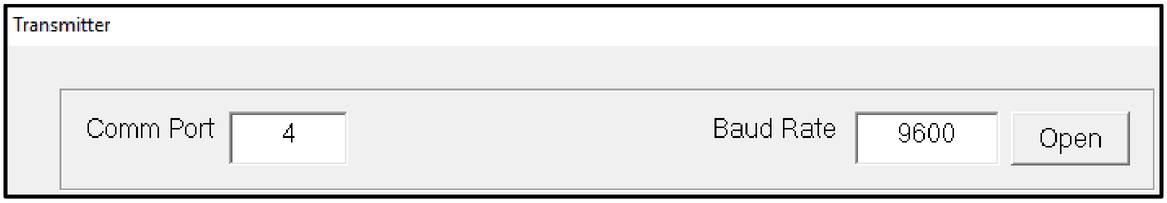
5.1 Basics of Software in LASER communication:

To start off we needed a Graphical User interface which helps the user navigate through the system and send/receive data. Our project also includes AES-128 bit encryption, which was to be implemented via software also. Since we also have 2 Arduinos working in our hardware so they were also meant to be programmed. Encryption/decryption takes place in Arduino's code. All of this leads to 2 GUIs, namely Transmitter and Receiver. Both the GUIs were made on Visual Basic 6 and programmed in visual basic.

5.2 Transmitter GUI:

- ### 5.2.1 COMM port and BAUD rate selection:

The transmitter GUI has multiple functionalities. First it makes the user set the communication port and decide the Baud Rate for his communication. Baud rate has been set to 9600 as default. Once the user has set the baud rate and is done with communication port selection, he will click the OPEN button. This will initiate the communication with the serial port leading to Arduino. If the user wrongly selects the comm port a dialog box will open saying "error opening comm port" and the program won't function.



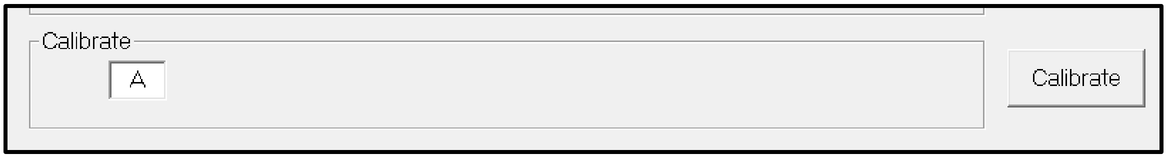
Comm Port 4

Baud Rate 9600

Figure 5-1: Comm Port Selection

- ### 5.2.2 Calibration:

Once the user sets the system he/she needs to calibrate the connection so that he/she is sure that the original data that is sent, safely reaches the receiver. To calibrate, user will send a random bit i.e. A, B x, y etc. and check on the receiver if that is received. To allow this to happen a calibrate button is made along with a character box. User needs to enter data in that box and then click the calibrate button.



Calibrate: A

Figure 5-2: Calibration

- ### 5.2.3 Message Transmission:

After the user has calibrated the connection, he/she can now send the data that is message/file/image. To send a message, the user will his message in the char box along the message option and then click send. To clear the existing message the user will click the clear button.

Note: This will only work once you have calibrated the connection.

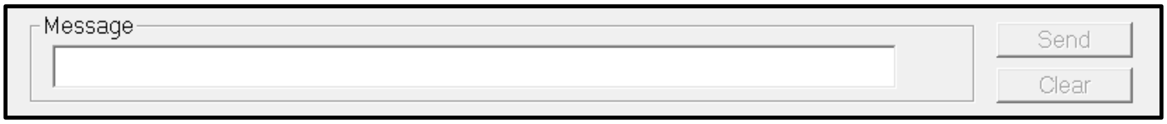
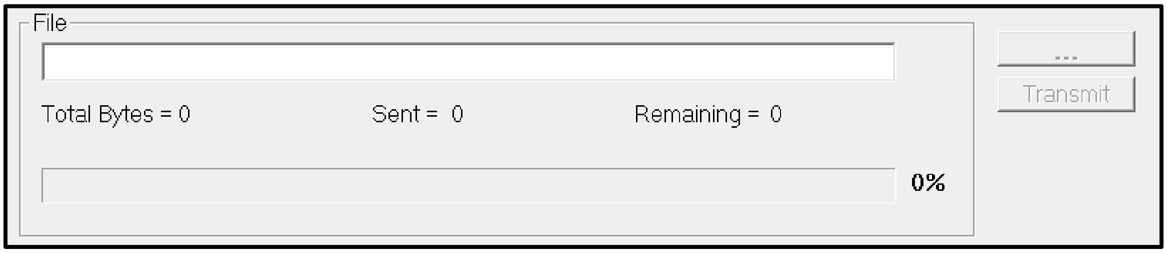


Figure 5-3: Message Transmission

- ### 5.2.4 File/Image Transmission:

To transmit file/image, the user will select the file by clicking on "…" button. This will make the user browse through his computer and find the file/image and send it. By clicking on the "Transmit" button the file will be transmitted. When the user transmits the data a bar graph below shows how much data has been sent. It shows the number of bytes being sent and the number of remaining bytes and total bytes.



Total Bytes = 0 Sent = 0 Remaining = 0

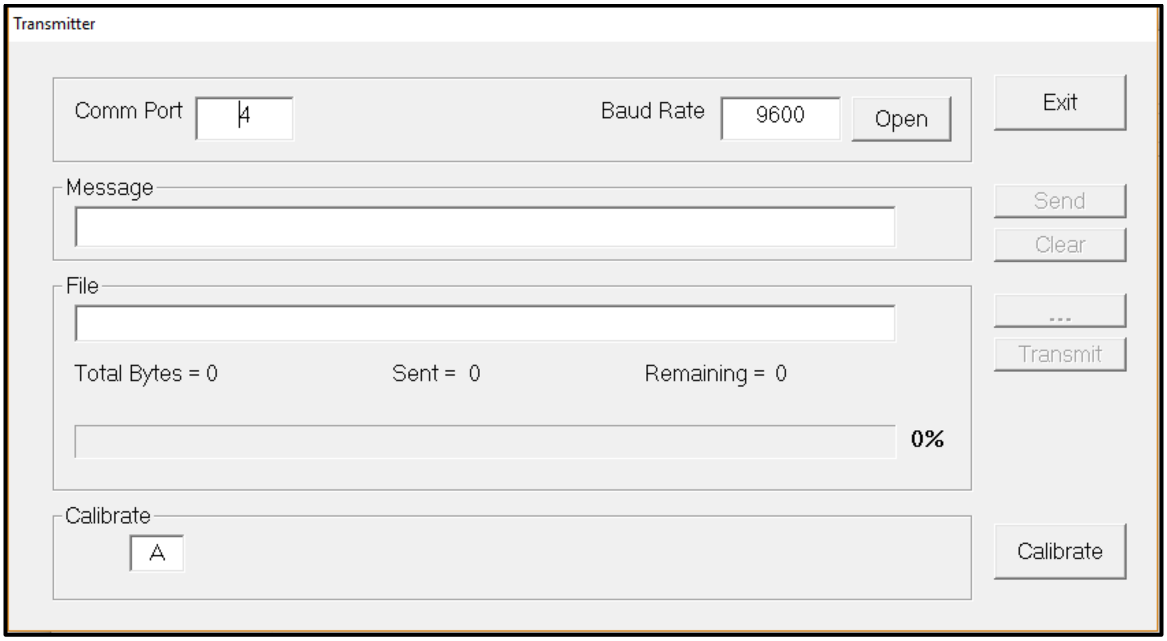
0%

Figure 5-4: File/Image Transmission

- ### 5.2.5 Exit Button:

  To end the program and exit the transmitter mode there is an "Exit" button. User can click that to exit the GUI.

Here is how the transmitter looks.



Comm Port 4

Baud Rate 9600

Total Bytes = 0

Sent = 0

Remaining = 0

0%

A

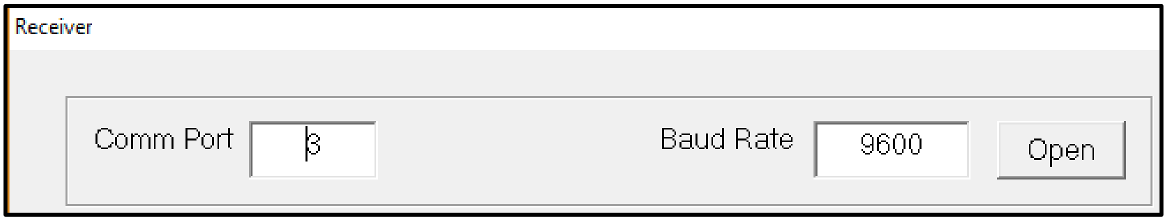
Figure 5-5: Transmitter GUI

5.3 Receiver GUI:

The receiver GUI is responsible to show the user the Data that is received on the receiver Arduino. This GUI is similar to transmitter. First the user has to set the baud rate and select the comm port and then calibrate the connection. After this the user will be able to receive data.

- ### 5.3.1 COMM port and Baud Rate:

The first thing the user need to do after opening the GUI is to select the comm port and select baud rate. Since the baud rate on Arduino has been set to 9600, hence the baud rate in GUI will also be selected to 9600. The comm port can be confirmed by going to your computer's device manager and selecting "ports" option. It will tell you which port has been connected serially to Arduino circuitry. After you are done with the baud rate and comm port selection, click OPEN and the char boxes will turn green. Otherwise a dialogue box will appear showing "Error opening comm port".



Comm Port 3

Baud Rate 9600

Figure 5-6: Comm Port Selection

- ### 5.3.2 Calibration:

To calibrate the user just needs to click the calibrate button and see if he is receiving the same bit as the one that is being sent.

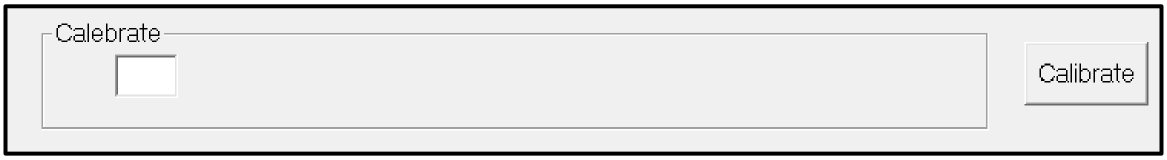


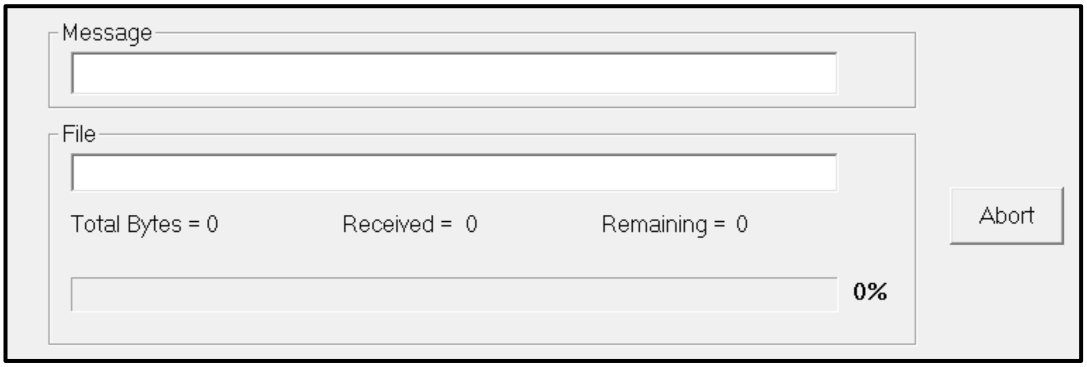
Figure 5-7: Calibration

- ### 5.3.3 Receiving Message/File/Image:

If the user has calibrated the connection then as soon as the sender sends the message he will receive that immediately on his message Char Box.

Files/images are received in a folder at named “received” in the receivers computer. The path of this folder has been set In the code. The bar graph below the file button shows the progress of file being received. It tell about how many bytes have been received and how many are left. If the user can abort the receiving by clicking the “abort” button. This will halt the file/image being received.

Note: The supported file format is .txt and Image format is jpg.



Total Bytes = 0

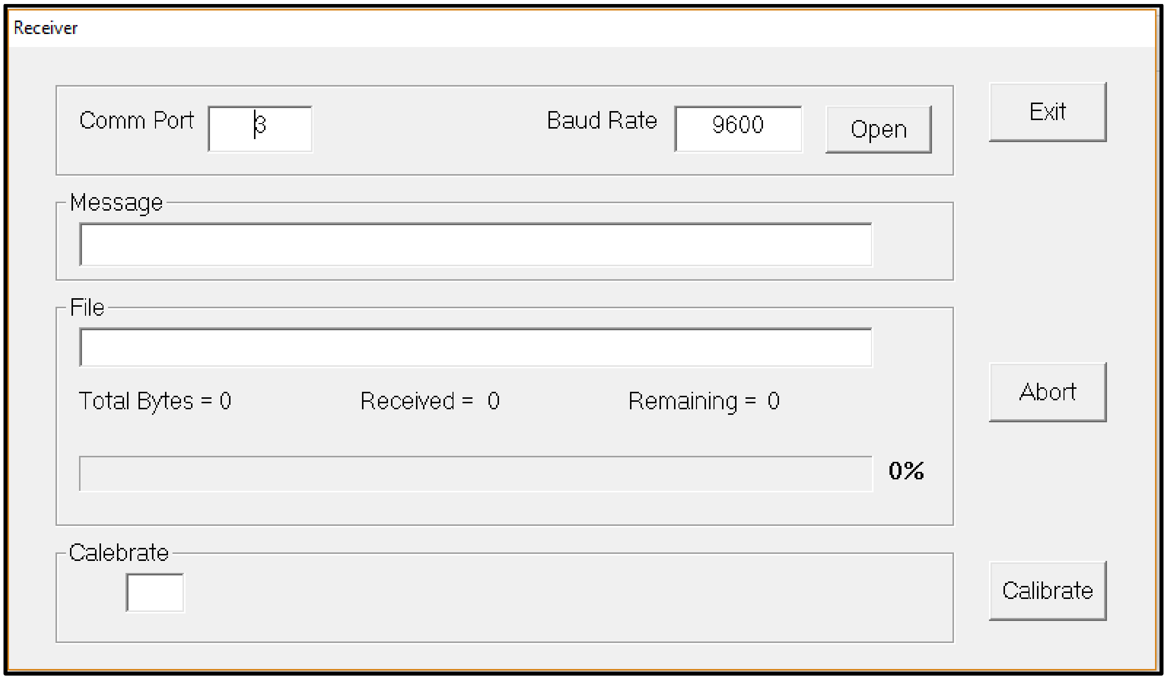
Received = 0

Remaining = 0

0%

Figure 5-8: Receiving File/Image/Message

Here is how the receiver looks.



Comm Port 3

Baud Rate 9600

Total Bytes = 0

Received = 0

Remaining = 0

0%

Figure 5-9: Receiver GUI

5.4 Arduino's role in LASER communication:

Arduino has a major role in this project. The data sent by the transmitter is gathered and sent bit by bit to LASER. It controls the LASER in terms of voltage. For every Bit that is 1 it turns on the LASER by giving it voltage and vice versa. Moreover it also encrypts and decrypts the data that is being sent. The encryption scheme used is 128-bit AES. The built-in AES library is used for encryption and decryption in Arduino. The Arduino is also connected to an LCD display which shows some basic information in both circuits. It tells about the mode the circuit is working in I.e. Audio mode, video mode.

- ### 5.4.1 Arduino Transmitter:

The transmitter takes the data from sender, encrypts it and sends it bit by bit to LASER by turning the LASER on and off. It turns on the buzzer also when the data is being sent. This is to notify the user. The Buzzer also turns on and off when audio mode is turned on or off respectively.

- ### 5.4.2 Arduino’s Receiver:

The receiver receives the data in form of bits, decrypts it and converts it to characters that are then shown on the GUI. The receiver Arduino also controls the buzzer on the receiver. This buzzer is turned on every time the data is received. The LCD display connected to Arduino tells about the mode in which the circuit is currently working on.

Chapter 6: Conclusion and Future Work

Chapter 6: Conclusion and Future Scope

6.1 Conclusion:

We were able to make a setup which has the capability to communicate remotely between two nodes or PCs using a LASER diode. We have successfully transmitted files/images/messages and audio files including real time voice signal as form of data. The messages to be sent are encrypted to make them secure. The encryption technique we have used is AES 128 bit. We used mirrors and prism to maintain the line of sight around corners and edges and convex lens to focus the LASER on photo diode. We have successfully achieved the range of up to 500 meters in night time and up to 60 meters in day time providing the line of sight is not disturbed. The difference in day and night time is due to the fact that at day time we have a lot of white light which acts as noise to our photo diode at receiver end.

We were aimed at creating a communication system without using the regular radio frequency spectrum and working on the concept of visible light communication in which the user is provided an ease to communicate with other people without using any internet, Bluetooth, USB or any other wire connections.

6.2 Future Work:

Following can be done in order to make this setup more convenient and easy to use for users.

- **Full Duplex:**

Our project was aimed at transmitting and receiving data in Simplex form. This project can also be implemented in full duplex form. To obtain that LASERs and Photo diodes will have to be implemented on both terminals i.e. transmitter, receiver. The circuit will be much complex than this one and it may reduce rate of transmission. Both the GUIs will have to be reprogrammed accordingly.

- **File/Image Formats:**

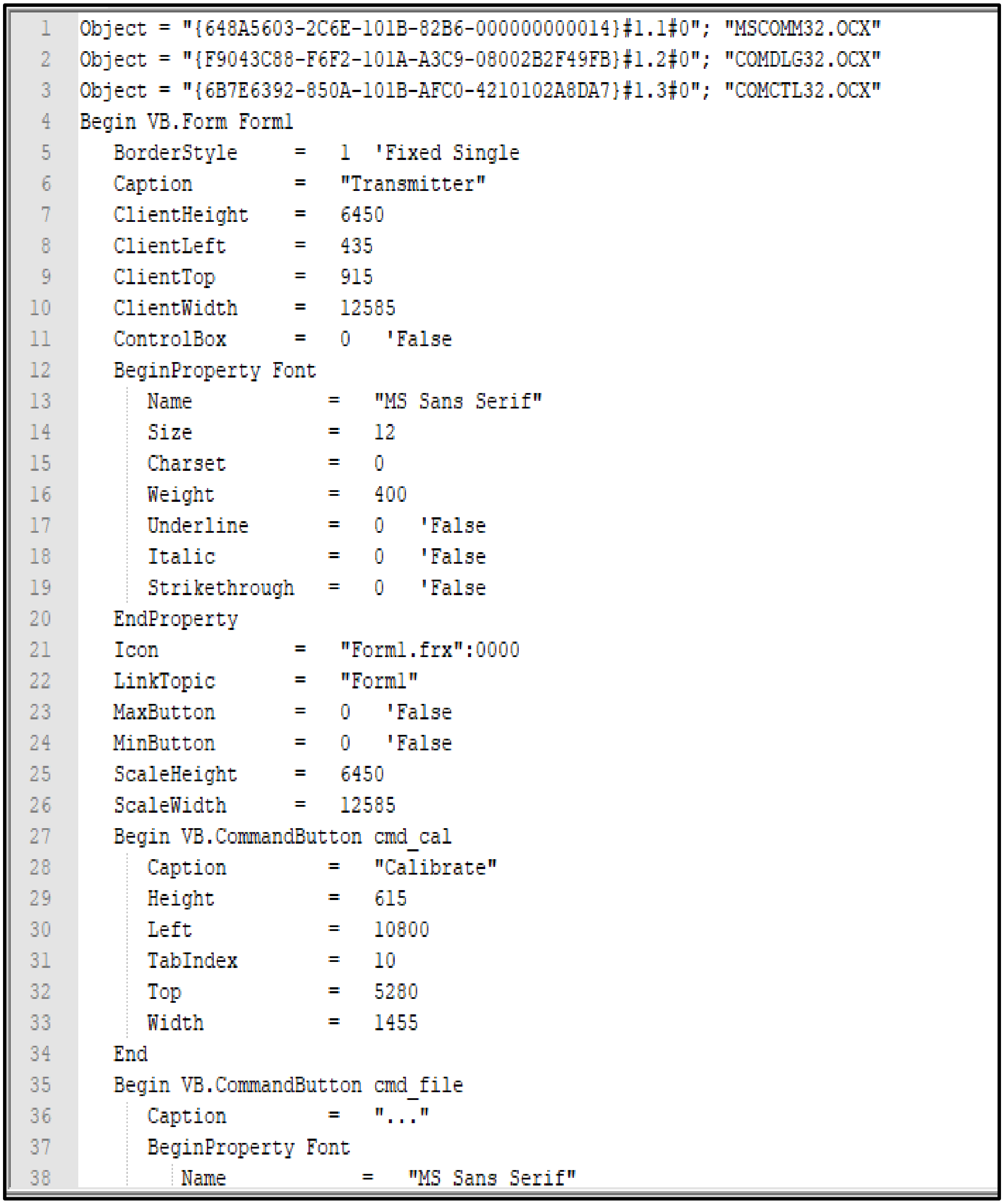
Our project successfully transmitted .txt file and jpg image. This can be increased to other file/image formats including .exe .PNG .doc .pdf etc. Moreover any kind of video format is also not supported in this project. To accomplish this, it will take a lot of processing power and AT Mega 328P (Arduino chip) may have to be replaced with some higher ranked microcontroller chip.

- **Baud rate:**

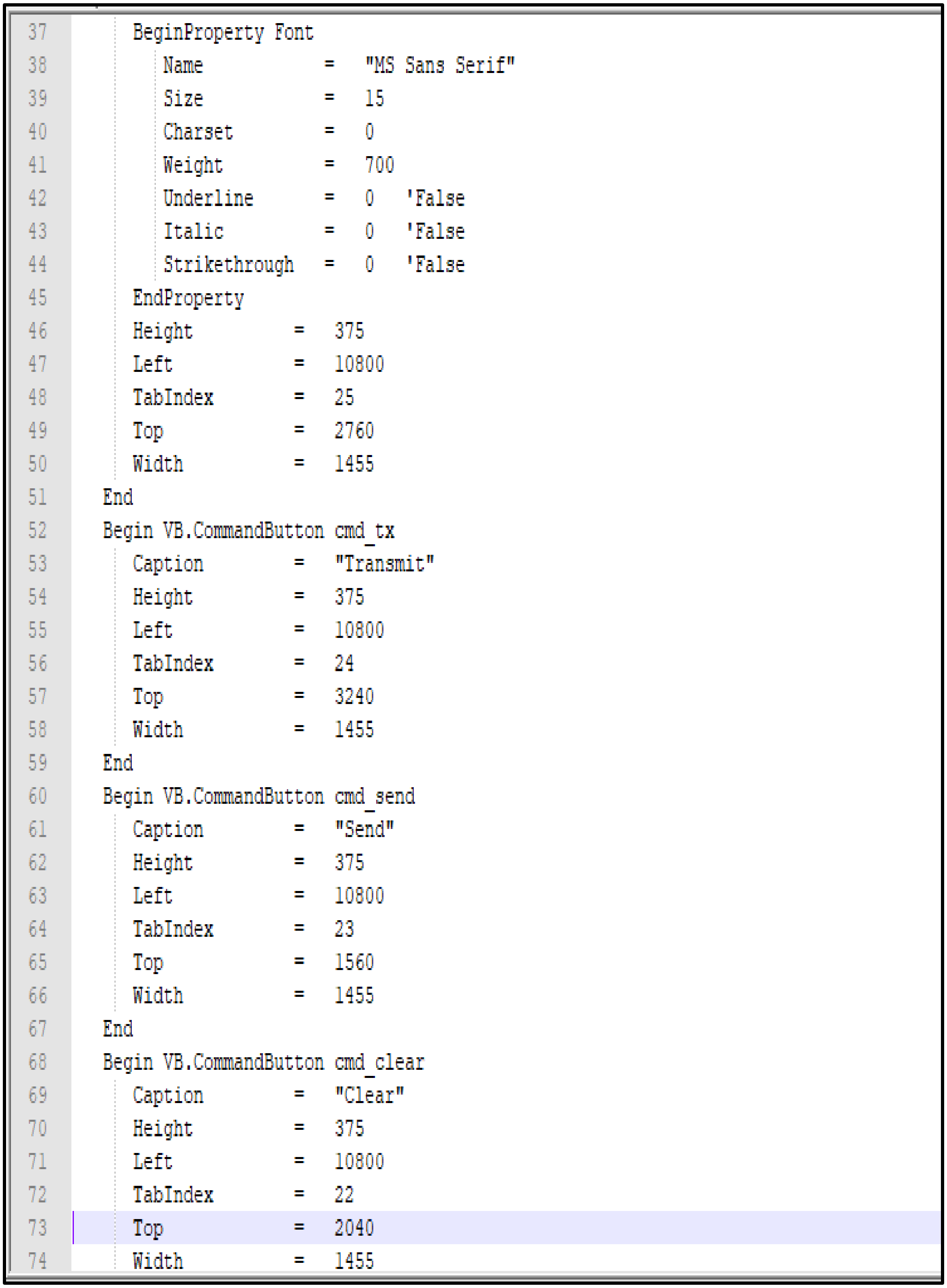
Our baud rate was 9600 fixed. We can increase this baud rate to much higher values but it comes with data loss and noise. But if we high serial communicating microcontrollers this baud rate can be increased.

APPENDIX ‘A

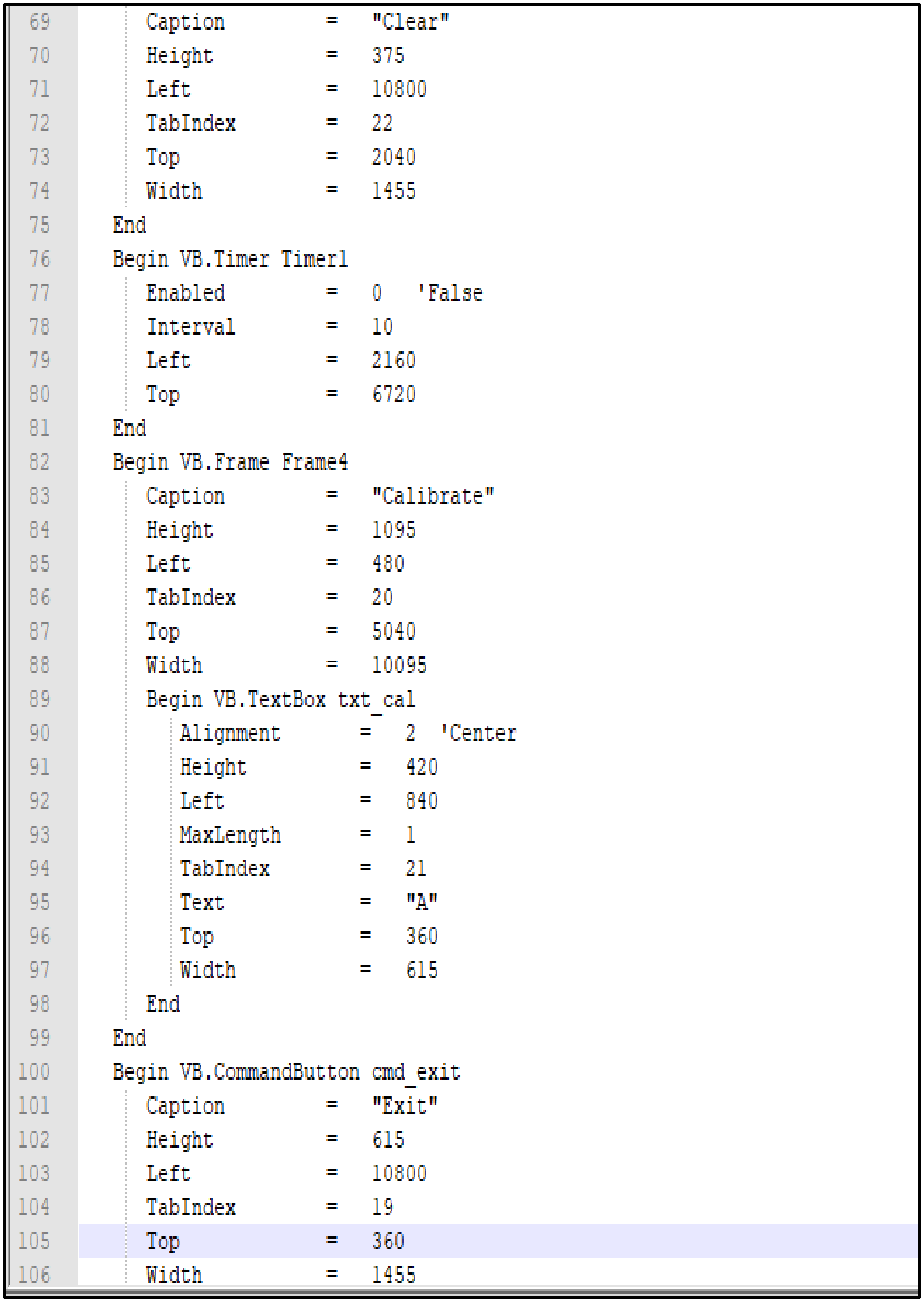
Transmitter GUI Code



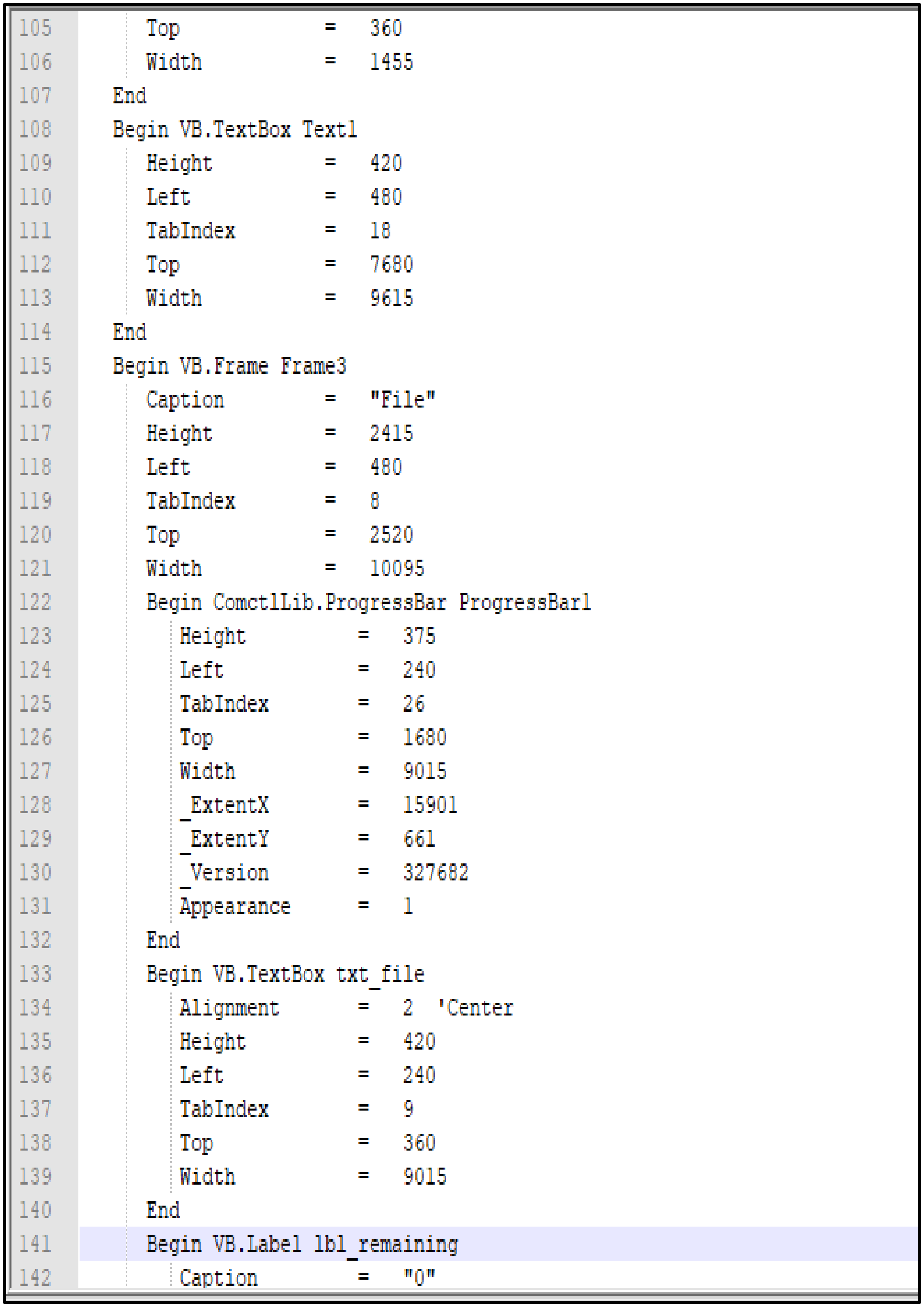
```
Object = "{648A5603-2C6E-101B-82B6-000000000014}#1.1#0"; "MSCOMM32.OCX"
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
Object = "{6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.3#0"; "COMCTL32.OCX"
Begin VB.Form Form1
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Transmitter"
   ClientHeight    =   6450
   ClientLeft      =   435
   ClientTop       =   915
   ClientWidth     =   12585
   ControlBox      =   0   'False
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   12
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Icon            =   "Form1.frx":0000
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   6450
   ScaleWidth      =   12585
   Begin VB.CommandButton cmd_cal
      Caption         =   "Calibrate"
      Height          =   615
      Left            =   10800
      TabIndex        =   10
      Top             =   5280
      Width           =   1455
   End
   Begin VB.CommandButton cmd_file
      Caption         =   "..."
      BeginProperty Font
         Name            =   "MS Sans Serif"
```



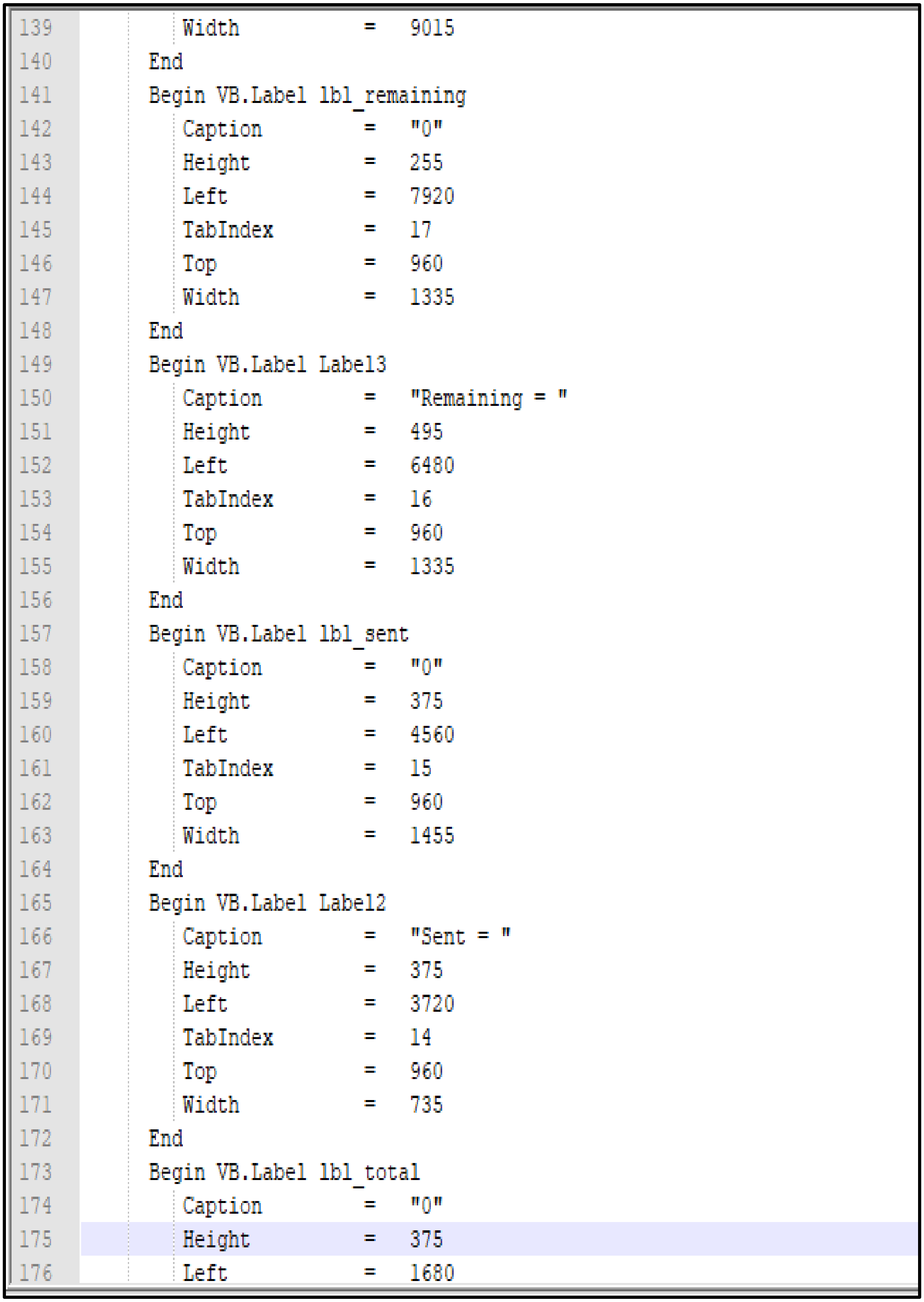
```
       BeginProperty Font
          Name            =   "MS Sans Serif"
          Size            =   15
          Charset         =   0
          Weight          =   700
          Underline       =   0   'False
          Italic          =   0   'False
          Strikethrough   =   0   'False
       EndProperty
       Height          =   375
       Left            =   10800
       TabIndex        =   25
       Top             =   2760
       Width           =   1455
    End
    Begin VB.CommandButton cmd_tx
       Caption         =   "Transmit"
       Height          =   375
       Left            =   10800
       TabIndex        =   24
       Top             =   3240
       Width           =   1455
    End
    Begin VB.CommandButton cmd_send
       Caption         =   "Send"
       Height          =   375
       Left            =   10800
       TabIndex        =   23
       Top             =   1560
       Width           =   1455
    End
    Begin VB.CommandButton cmd_clear
       Caption         =   "Clear"
       Height          =   375
       Left            =   10800
       TabIndex        =   22
       Top             =   2040
       Width           =   1455
```



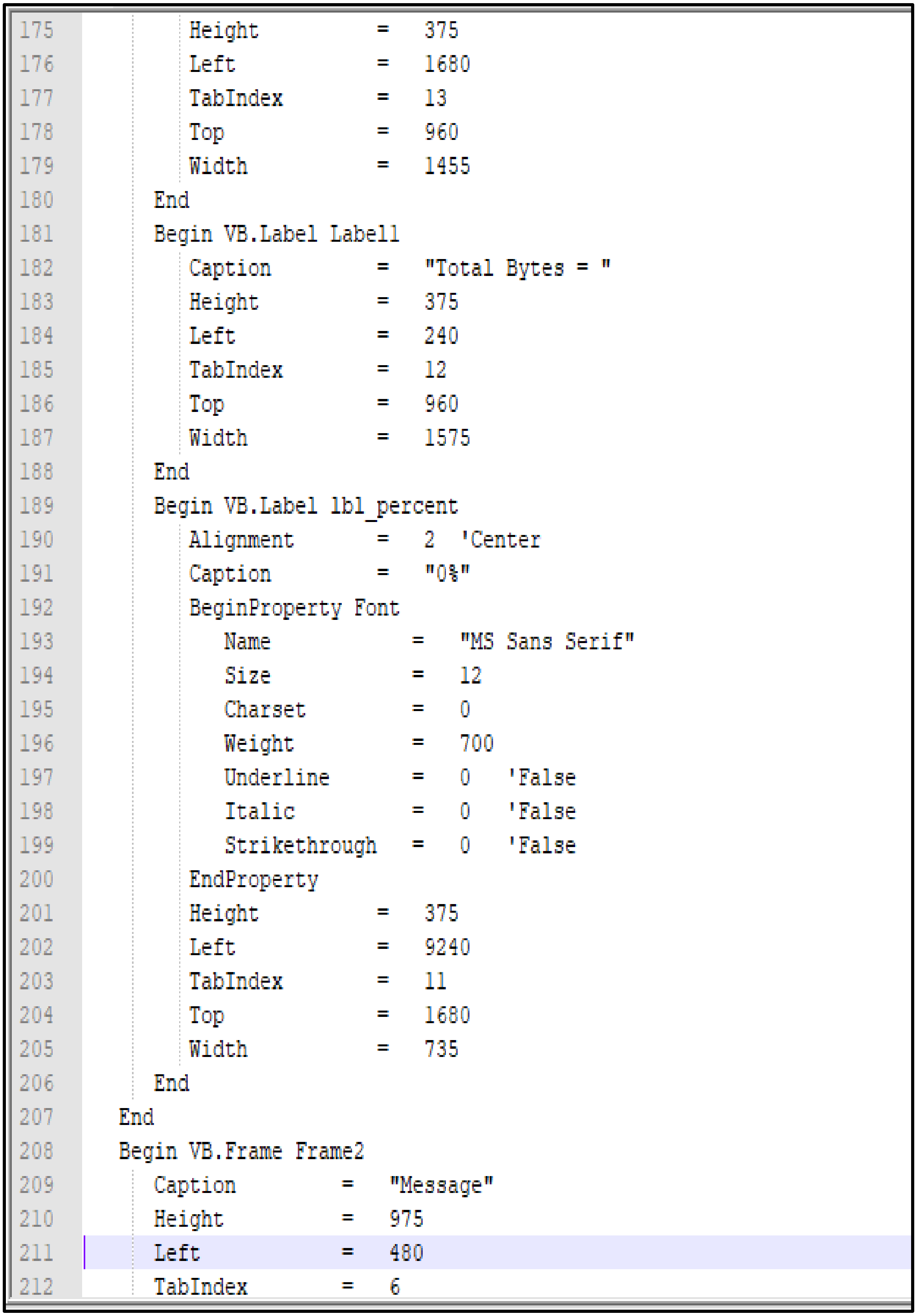
```
      Caption         =   "Clear"
      Height          =   375
      Left            =   10800
      TabIndex        =   22
      Top             =   2040
      Width           =   1455
   End
   Begin VB.Timer Timer1
      Enabled         =   0   'False
      Interval        =   10
      Left            =   2160
      Top             =   6720
   End
   Begin VB.Frame Frame4
      Caption         =   "Calibrate"
      Height          =   1095
      Left            =   480
      TabIndex        =   20
      Top             =   5040
      Width           =   10095
      Begin VB.TextBox txt_cal
         Alignment       =   2  'Center
         Height          =   420
         Left            =   840
         MaxLength       =   1
         TabIndex        =   21
         Text            =   "A"
         Top             =   360
         Width           =   615
      End
   End
   Begin VB.CommandButton cmd_exit
      Caption         =   "Exit"
      Height          =   615
      Left            =   10800
      TabIndex        =   19
      Top             =   360
      Width           =   1455
```



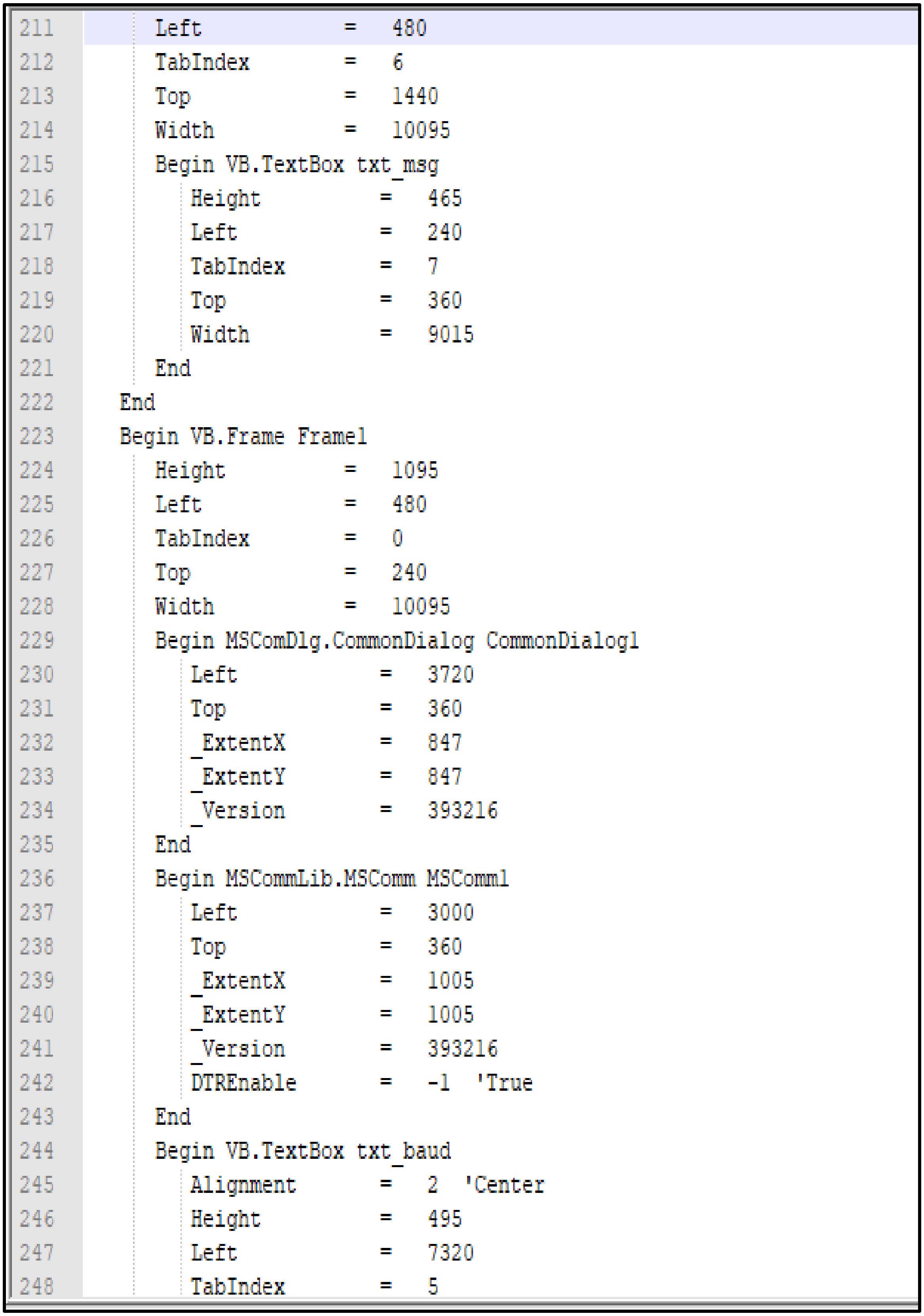
```
         Top             =   360
         Width           =   1455
      End
      Begin VB.TextBox Text1
         Height          =   420
         Left            =   480
         TabIndex        =   18
         Top             =   7680
         Width           =   9615
      End
      Begin VB.Frame Frame3
         Caption         =   "File"
         Height          =   2415
         Left            =   480
         TabIndex        =   8
         Top             =   2520
         Width           =   10095
         Begin ComctlLib.ProgressBar ProgressBar1
            Height          =   375
            Left            =   240
            TabIndex        =   26
            Top             =   1680
            Width           =   9015
            _ExtentX        =   15901
            _ExtentY        =   661
            _Version        =   327682
            Appearance      =   1
         End
         Begin VB.TextBox txt_file
            Alignment       =   2  'Center
            Height          =   420
            Left            =   240
            TabIndex        =   9
            Top             =   360
            Width           =   9015
         End
         Begin VB.Label lbl_remaining
            Caption         =   "0"
```



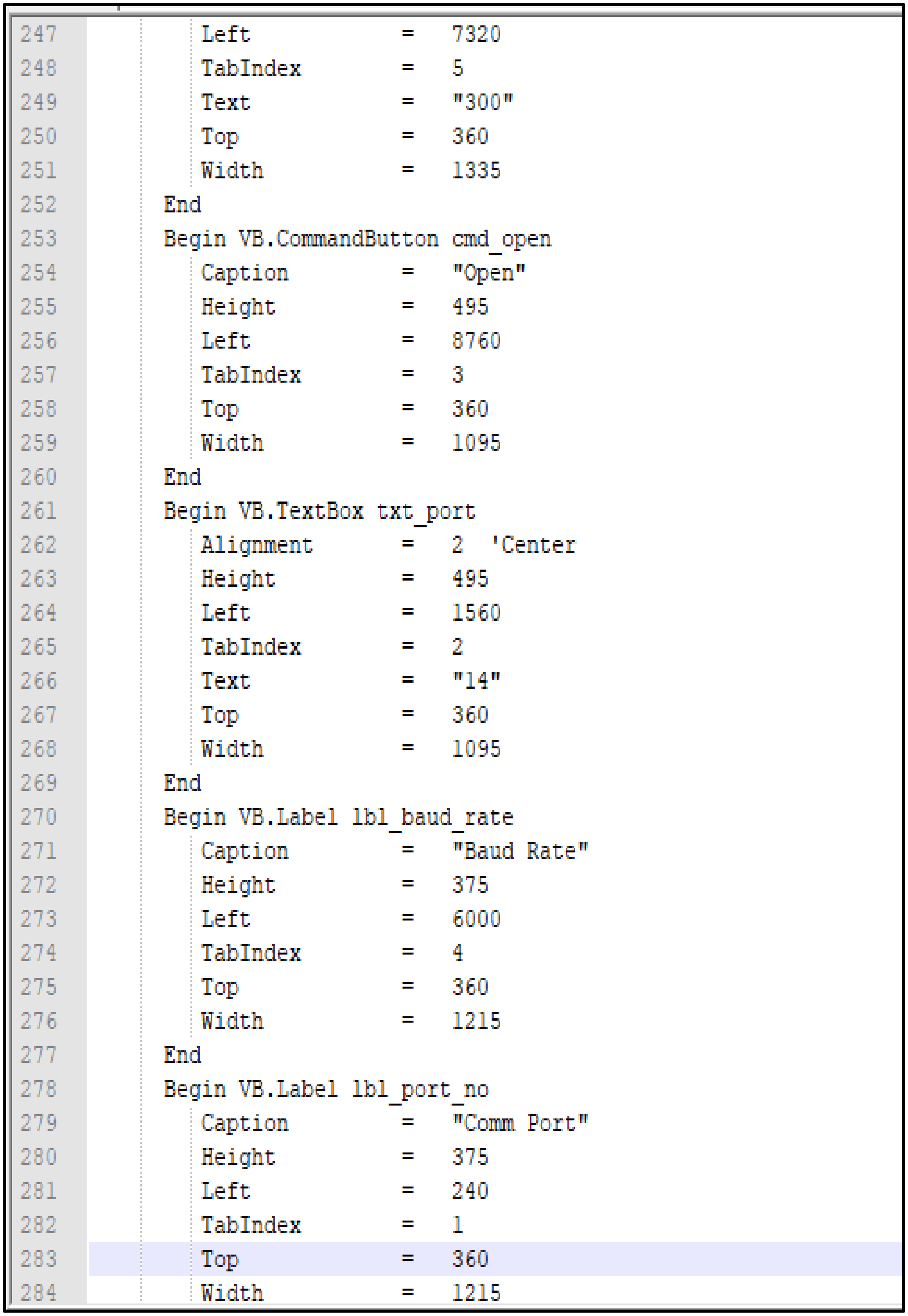
```
      Width           =   9015
   End
   Begin VB.Label lbl_remaining
      Caption         =   "0"
      Height          =   255
      Left            =   7920
      TabIndex        =   17
      Top             =   960
      Width           =   1335
   End
   Begin VB.Label Label3
      Caption         =   "Remaining = "
      Height          =   495
      Left            =   6480
      TabIndex        =   16
      Top             =   960
      Width           =   1335
   End
   Begin VB.Label lbl_sent
      Caption         =   "0"
      Height          =   375
      Left            =   4560
      TabIndex        =   15
      Top             =   960
      Width           =   1455
   End
   Begin VB.Label Label2
      Caption         =   "Sent = "
      Height          =   375
      Left            =   3720
      TabIndex        =   14
      Top             =   960
      Width           =   735
   End
   Begin VB.Label lbl_total
      Caption         =   "0"
      Height          =   375
      Left            =   1680
```



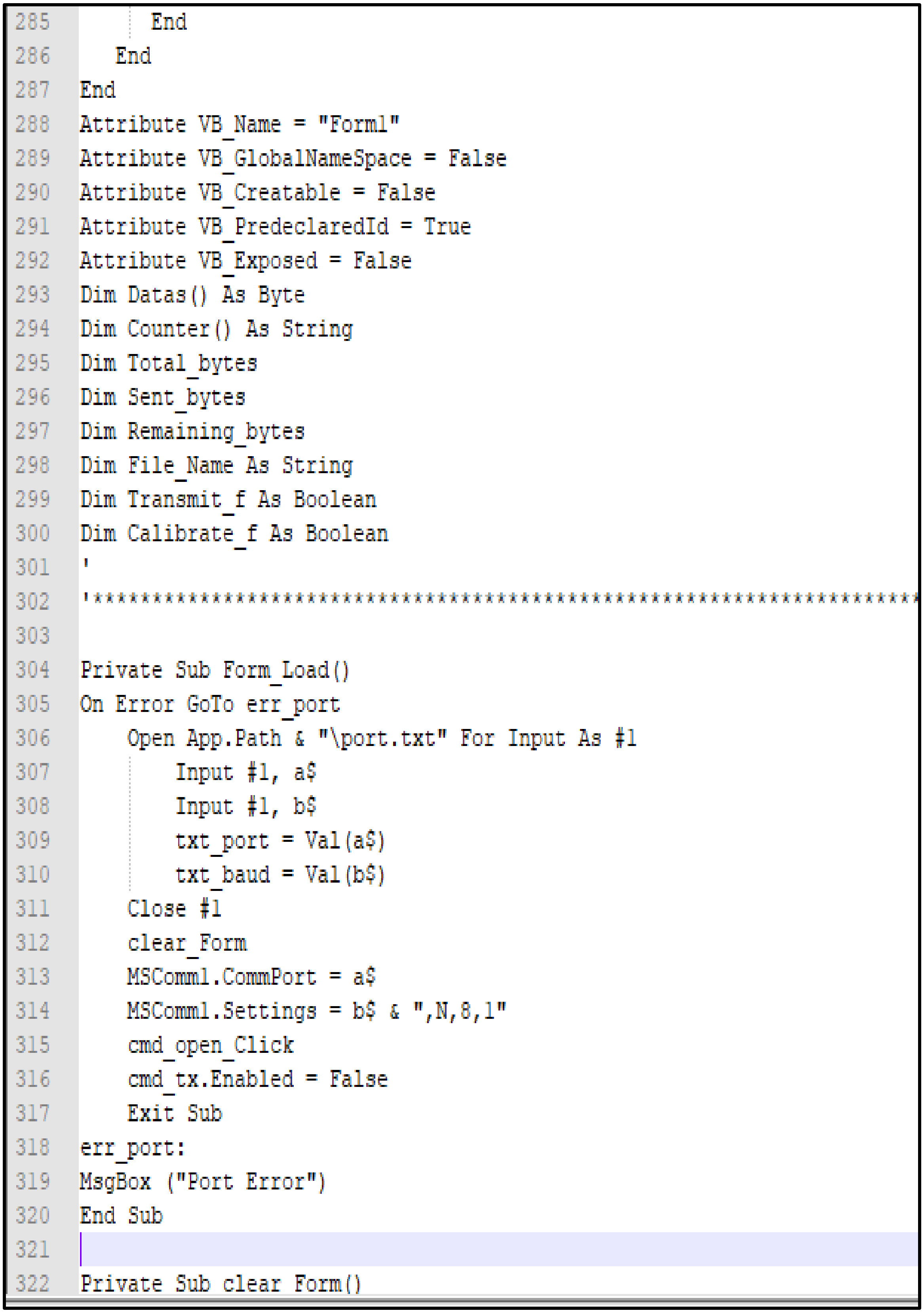
```
         Height          =   375
         Left            =   1680
         TabIndex        =   13
         Top             =   960
         Width           =   1455
      End
      Begin VB.Label Label1
         Caption         =   "Total Bytes = "
         Height          =   375
         Left            =   240
         TabIndex        =   12
         Top             =   960
         Width           =   1575
      End
      Begin VB.Label lbl_percent
         Alignment       =   2  'Center
         Caption         =   "0%"
         BeginProperty Font
            Name            =   "MS Sans Serif"
            Size            =   12
            Charset         =   0
            Weight          =   700
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
         EndProperty
         Height          =   375
         Left            =   9240
         TabIndex        =   11
         Top             =   1680
         Width           =   735
      End
   End
   Begin VB.Frame Frame2
      Caption         =   "Message"
      Height          =   975
      Left            =   480
      TabIndex        =   6
```



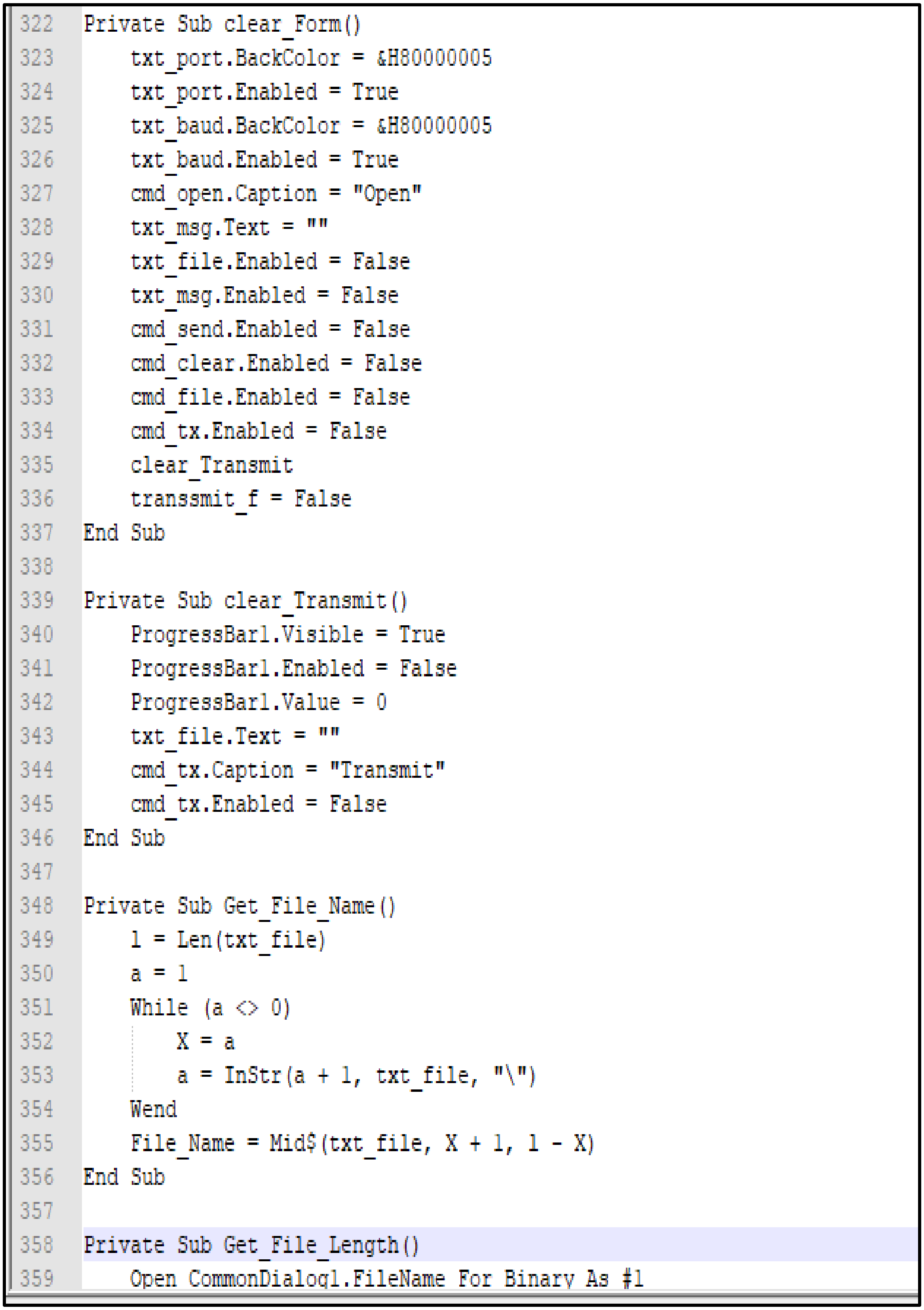
```
      Left            =   480
      TabIndex        =   6
      Top             =   1440
      Width           =   10095
      Begin VB.TextBox txt_msg
         Height          =   465
         Left            =   240
         TabIndex        =   7
         Top             =   360
         Width           =   9015
      End
   End
   Begin VB.Frame Frame1
      Height          =   1095
      Left            =   480
      TabIndex        =   0
      Top             =   240
      Width           =   10095
      Begin MSComDlg.CommonDialog CommonDialog1
         Left            =   3720
         Top             =   360
         _ExtentX        =   847
         _ExtentY        =   847
         _Version        =   393216
      End
      Begin MSCommLib.MSComm MSComm1
         Left            =   3000
         Top             =   360
         _ExtentX        =   1005
         _ExtentY        =   1005
         _Version        =   393216
         DTREnable       =   -1  'True
      End
      Begin VB.TextBox txt_baud
         Alignment       =   2  'Center
         Height          =   495
         Left            =   7320
         TabIndex        =   5
```



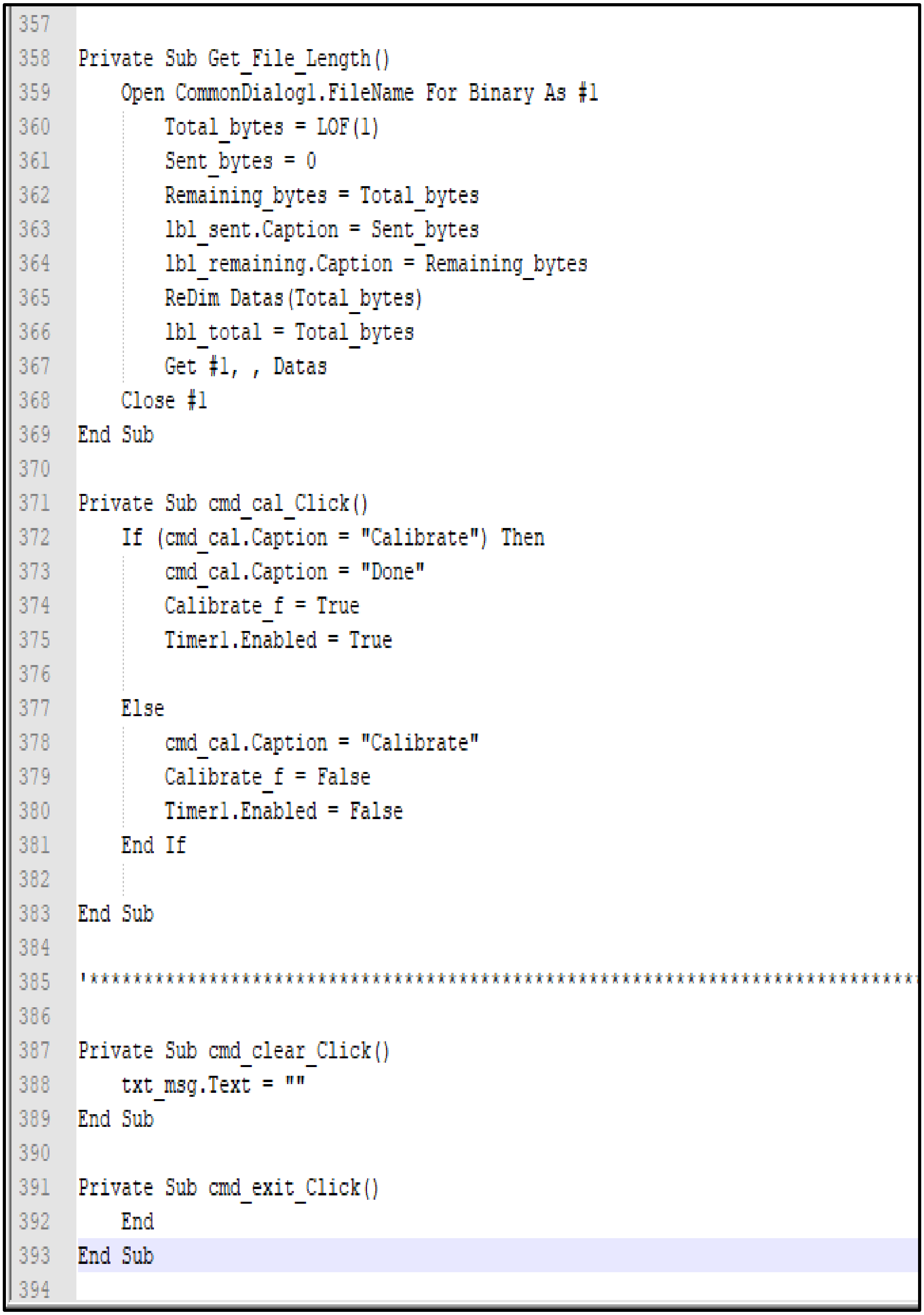
```
      Left            =   7320
      TabIndex        =   5
      Text            =   "300"
      Top             =   360
      Width           =   1335
   End
   Begin VB.CommandButton cmd_open
      Caption         =   "Open"
      Height          =   495
      Left            =   8760
      TabIndex        =   3
      Top             =   360
      Width           =   1095
   End
   Begin VB.TextBox txt_port
      Alignment       =   2  'Center
      Height          =   495
      Left            =   1560
      TabIndex        =   2
      Text            =   "14"
      Top             =   360
      Width           =   1095
   End
   Begin VB.Label lbl_baud_rate
      Caption         =   "Baud Rate"
      Height          =   375
      Left            =   6000
      TabIndex        =   4
      Top             =   360
      Width           =   1215
   End
   Begin VB.Label lbl_port_no
      Caption         =   "Comm Port"
      Height          =   375
      Left            =   240
      TabIndex        =   1
      Top             =   360
      Width           =   1215
```



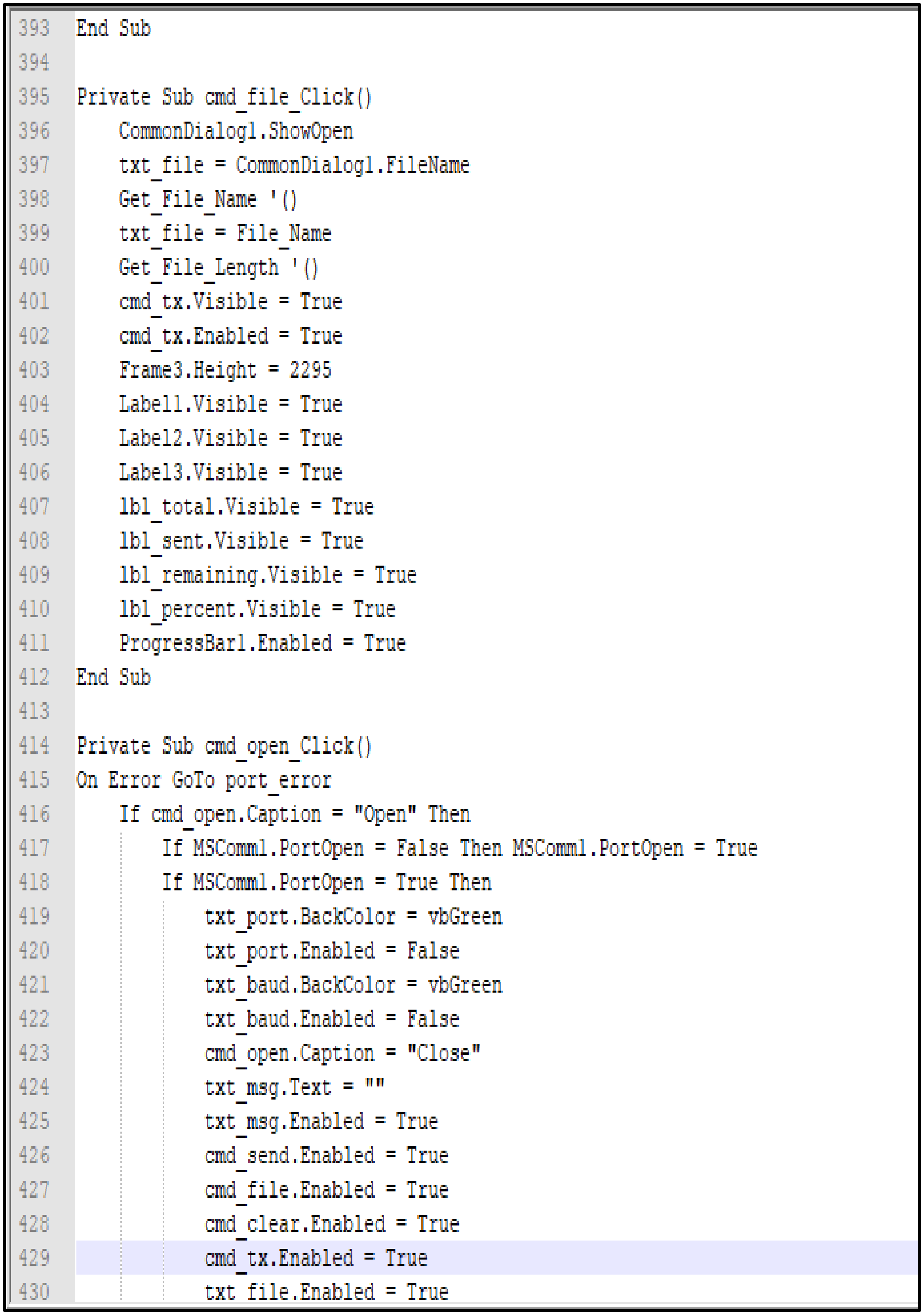
```
        End
    End
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Dim Datas() As Byte
Dim Counter() As String
Dim Total_bytes
Dim Sent_bytes
Dim Remaining_bytes
Dim File_Name As String
Dim Transmit_f As Boolean
Dim Calibrate_f As Boolean
'
'**********************************************************************

Private Sub Form_Load()
On Error GoTo err_port
    Open App.Path & "\port.txt" For Input As #1
        Input #1, a$
        Input #1, b$
        txt_port = Val(a$)
        txt_baud = Val(b$)
    Close #1
    clear_Form
    MSComm1.CommPort = a$
    MSComm1.Settings = b$ & ",N,8,1"
    cmd_open_Click
    cmd_tx.Enabled = False
    Exit Sub
err_port:
MsgBox ("Port Error")
End Sub

Private Sub clear_Form()
```



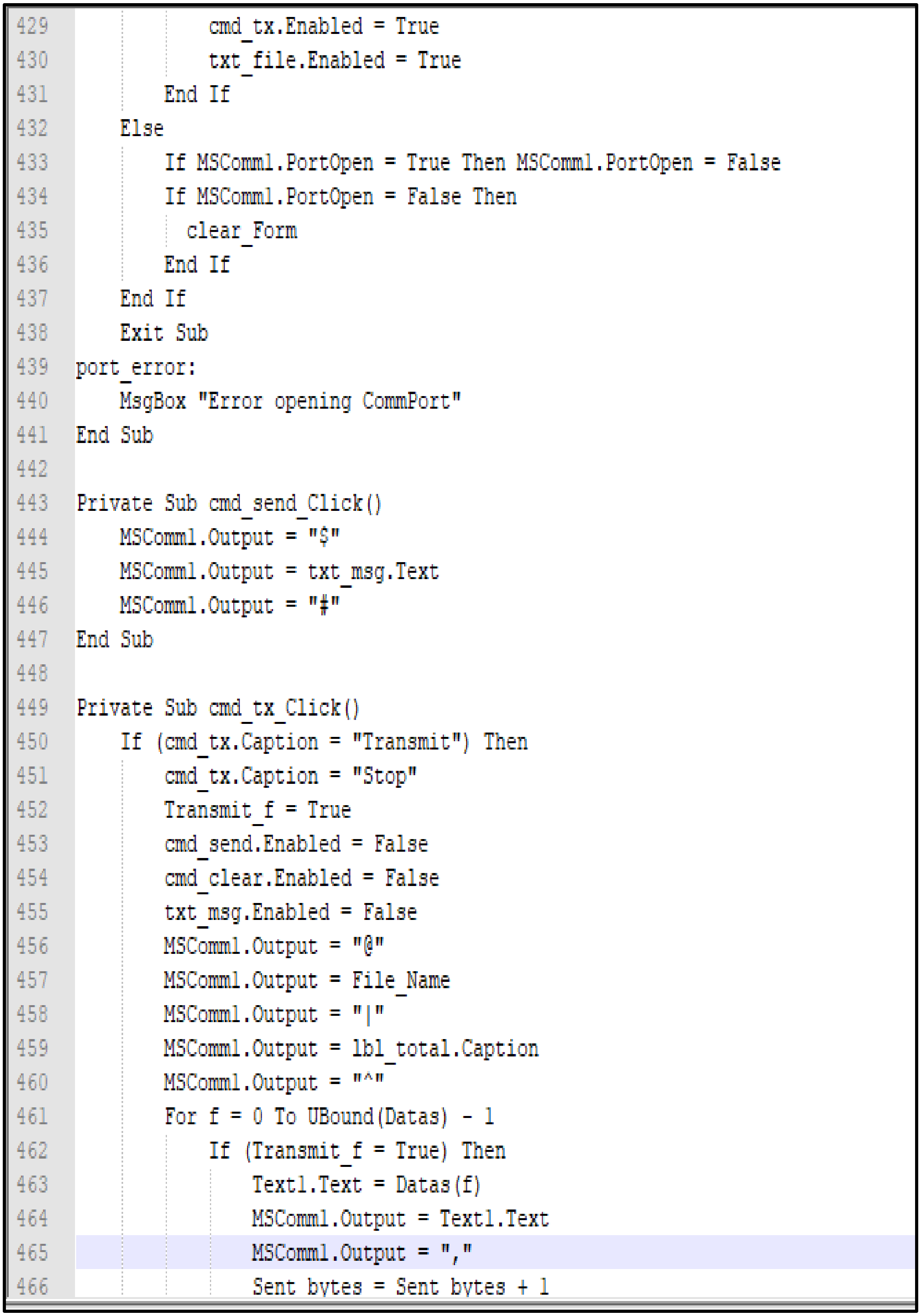
```
Private Sub clear_Form()
    txt_port.BackColor = &H80000005
    txt_port.Enabled = True
    txt_baud.BackColor = &H80000005
    txt_baud.Enabled = True
    cmd_open.Caption = "Open"
    txt_msg.Text = ""
    txt_file.Enabled = False
    txt_msg.Enabled = False
    cmd_send.Enabled = False
    cmd_clear.Enabled = False
    cmd_file.Enabled = False
    cmd_tx.Enabled = False
    clear_Transmit
    transsmit_f = False
End Sub

Private Sub clear_Transmit()
    ProgressBar1.Visible = True
    ProgressBar1.Enabled = False
    ProgressBar1.Value = 0
    txt_file.Text = ""
    cmd_tx.Caption = "Transmit"
    cmd_tx.Enabled = False
End Sub

Private Sub Get_File_Name()
    l = Len(txt_file)
    a = 1
    While (a <> 0)
        X = a
        a = InStr(a + 1, txt_file, "\")
    Wend
    File_Name = Mid$(txt_file, X + 1, l - X)
End Sub

Private Sub Get_File_Length()
    Open CommonDialog1.FileName For Binary As #1
```



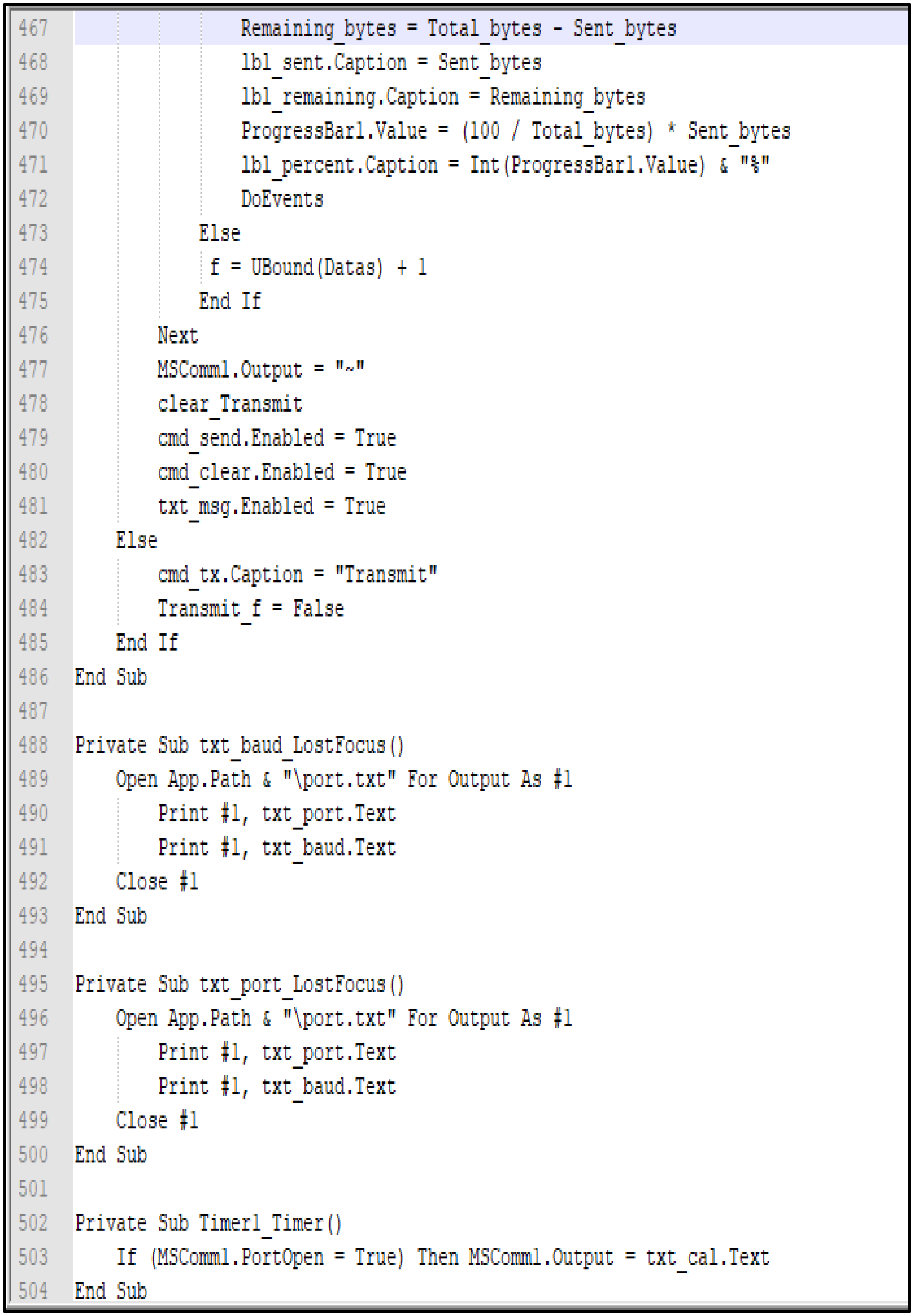
```
Private Sub Get_File_Length()
    Open CommonDialog1.FileName For Binary As #1
        Total_bytes = LOF(1)
        Sent_bytes = 0
        Remaining_bytes = Total_bytes
        lbl_sent.Caption = Sent_bytes
        lbl_remaining.Caption = Remaining_bytes
        ReDim Datas(Total_bytes)
        lbl_total = Total_bytes
        Get #1, , Datas
    Close #1
End Sub

Private Sub cmd_cal_Click()
    If (cmd_cal.Caption = "Calibrate") Then
        cmd_cal.Caption = "Done"
        Calibrate_f = True
        Timer1.Enabled = True

    Else
        cmd_cal.Caption = "Calibrate"
        Calibrate_f = False
        Timer1.Enabled = False
    End If

End Sub

'****************************************************************************

Private Sub cmd_clear_Click()
    txt_msg.Text = ""
End Sub

Private Sub cmd_exit_Click()
    End
End Sub
```



```
End Sub

Private Sub cmd_file_Click()
    CommonDialog1.ShowOpen
    txt_file = CommonDialog1.FileName
    Get_File_Name '()
    txt_file = File_Name
    Get_File_Length '()
    cmd_tx.Visible = True
    cmd_tx.Enabled = True
    Frame3.Height = 2295
    Label1.Visible = True
    Label2.Visible = True
    Label3.Visible = True
    lbl_total.Visible = True
    lbl_sent.Visible = True
    lbl_remaining.Visible = True
    lbl_percent.Visible = True
    ProgressBar1.Enabled = True
End Sub

Private Sub cmd_open_Click()
On Error GoTo port_error
    If cmd_open.Caption = "Open" Then
        If MSComm1.PortOpen = False Then MSComm1.PortOpen = True
        If MSComm1.PortOpen = True Then
            txt_port.BackColor = vbGreen
            txt_port.Enabled = False
            txt_baud.BackColor = vbGreen
            txt_baud.Enabled = False
            cmd_open.Caption = "Close"
            txt_msg.Text = ""
            txt_msg.Enabled = True
            cmd_send.Enabled = True
            cmd_file.Enabled = True
            cmd_clear.Enabled = True
            cmd_tx.Enabled = True
            txt_file.Enabled = True
```

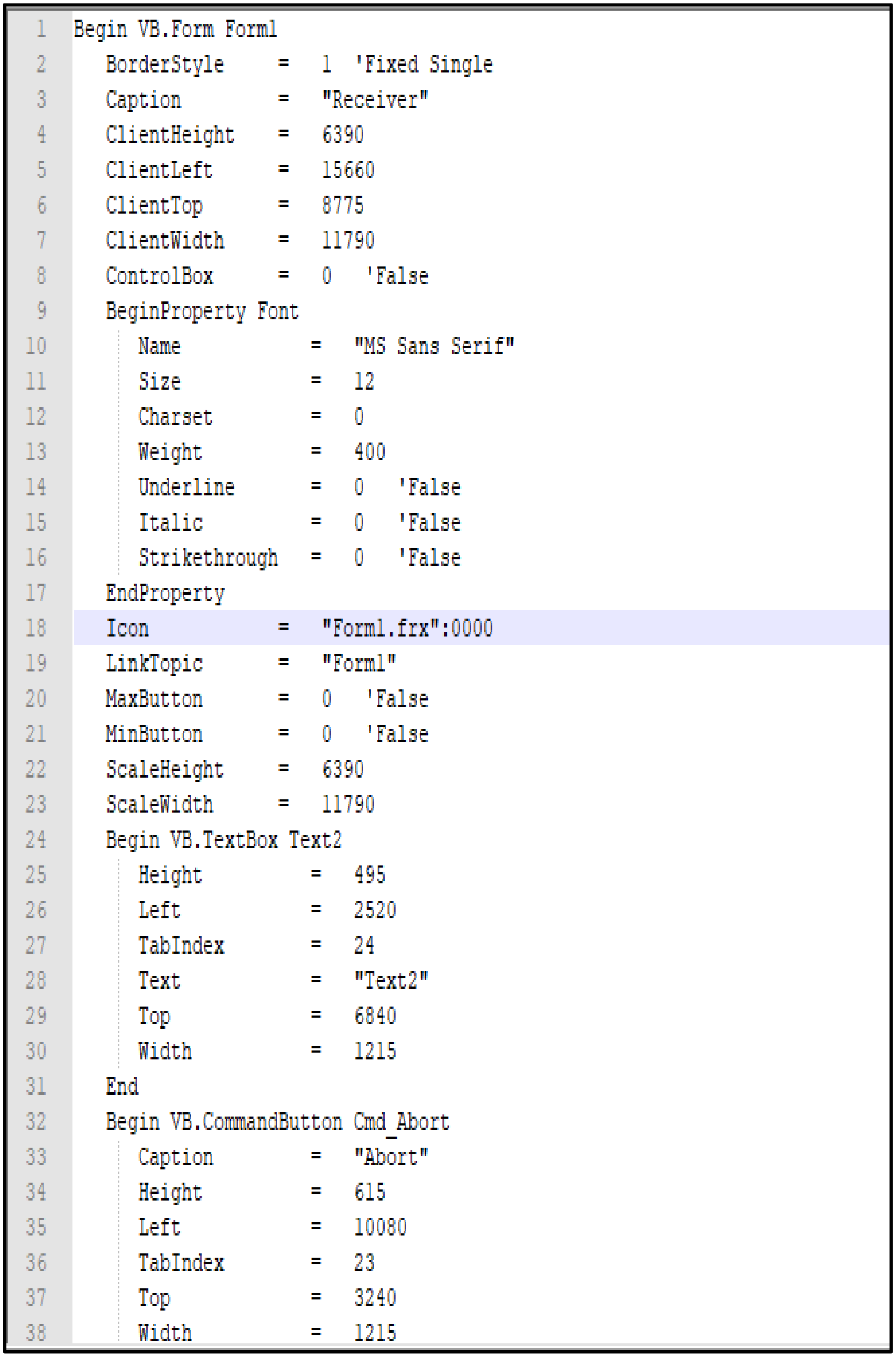


```
            cmd_tx.Enabled = True
            txt_file.Enabled = True
        End If
    Else
        If MSComm1.PortOpen = True Then MSComm1.PortOpen = False
        If MSComm1.PortOpen = False Then
          clear_Form
        End If
    End If
    Exit Sub
port_error:
    MsgBox "Error opening CommPort"
End Sub

Private Sub cmd_send_Click()
    MSComm1.Output = "$"
    MSComm1.Output = txt_msg.Text
    MSComm1.Output = "#"
End Sub

Private Sub cmd_tx_Click()
    If (cmd_tx.Caption = "Transmit") Then
        cmd_tx.Caption = "Stop"
        Transmit_f = True
        cmd_send.Enabled = False
        cmd_clear.Enabled = False
        txt_msg.Enabled = False
        MSComm1.Output = "@"
        MSComm1.Output = File_Name
        MSComm1.Output = "|"
        MSComm1.Output = lbl_total.Caption
        MSComm1.Output = "^"
        For f = 0 To UBound(Datas) - 1
            If (Transmit_f = True) Then
                Text1.Text = Datas(f)
                MSComm1.Output = Text1.Text
                MSComm1.Output = ","
                Sent_bytes = Sent_bytes + 1
```

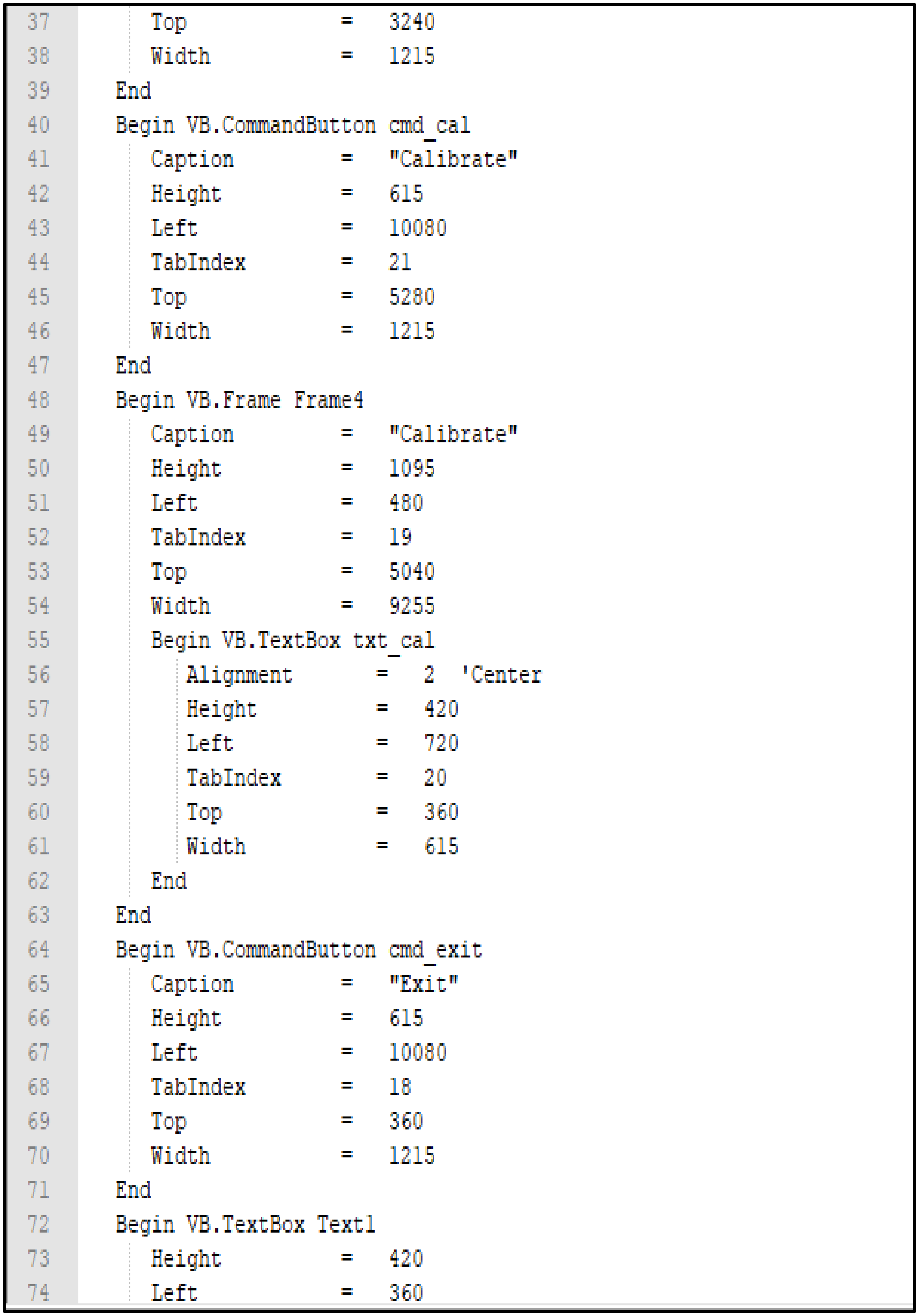


```
                Remaining_bytes = Total_bytes - Sent_bytes
                lbl_sent.Caption = Sent_bytes
                lbl_remaining.Caption = Remaining_bytes
                ProgressBar1.Value = (100 / Total_bytes) * Sent_bytes
                lbl_percent.Caption = Int(ProgressBar1.Value) & "%"
                DoEvents
            Else
             f = UBound(Datas) + 1
            End If
        Next
        MSComm1.Output = "~"
        clear_Transmit
        cmd_send.Enabled = True
        cmd_clear.Enabled = True
        txt_msg.Enabled = True
    Else
        cmd_tx.Caption = "Transmit"
        Transmit_f = False
    End If
End Sub

Private Sub txt_baud_LostFocus()
    Open App.Path & "\port.txt" For Output As #1
        Print #1, txt_port.Text
        Print #1, txt_baud.Text
    Close #1
End Sub

Private Sub txt_port_LostFocus()
    Open App.Path & "\port.txt" For Output As #1
        Print #1, txt_port.Text
        Print #1, txt_baud.Text
    Close #1
End Sub

Private Sub Timer1_Timer()
    If (MSComm1.PortOpen = True) Then MSComm1.Output = txt_cal.Text
End Sub
```

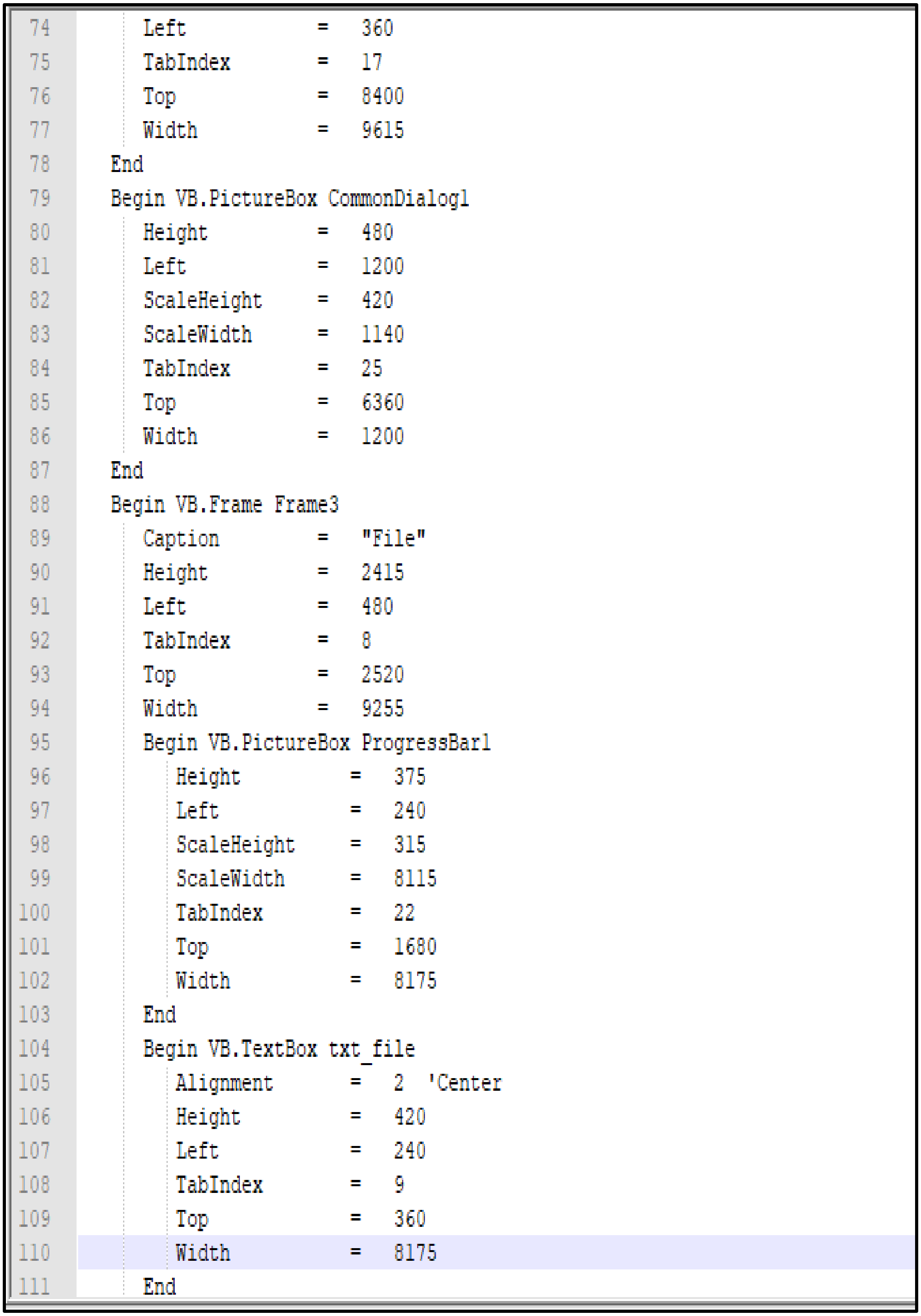
Receiver GUI Code:



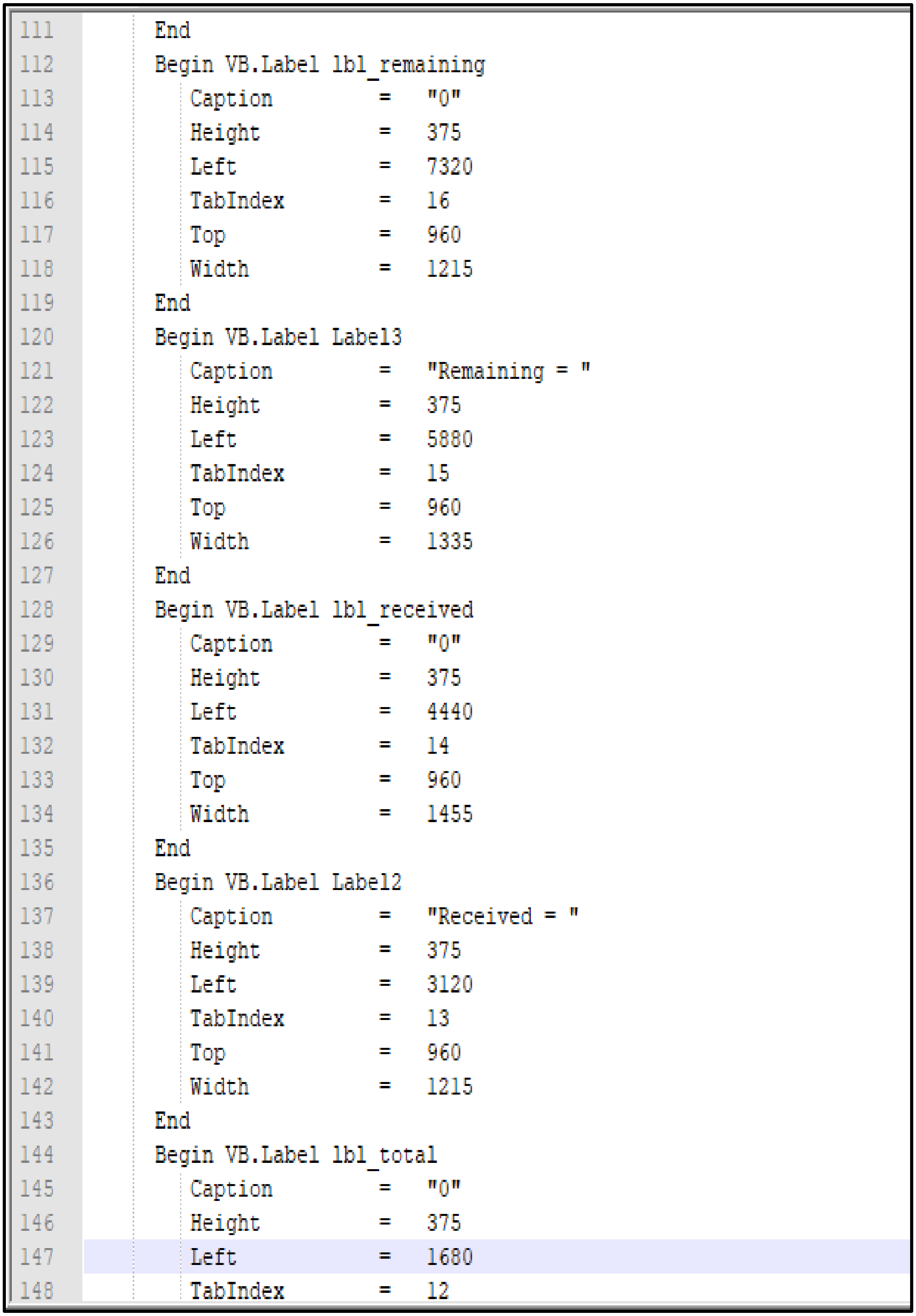
```
Begin VB.Form Form1
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Receiver"
   ClientHeight    =   6390
   ClientLeft      =   15660
   ClientTop       =   8775
   ClientWidth     =   11790
   ControlBox      =   0   'False
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   12
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Icon            =   "Form1.frx":0000
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   6390
   ScaleWidth      =   11790
   Begin VB.TextBox Text2
      Height          =   495
      Left            =   2520
      TabIndex        =   24
      Text            =   "Text2"
      Top             =   6840
      Width           =   1215
   End
   Begin VB.CommandButton Cmd_Abort
      Caption         =   "Abort"
      Height          =   615
      Left            =   10080
      TabIndex        =   23
      Top             =   3240
      Width           =   1215
```



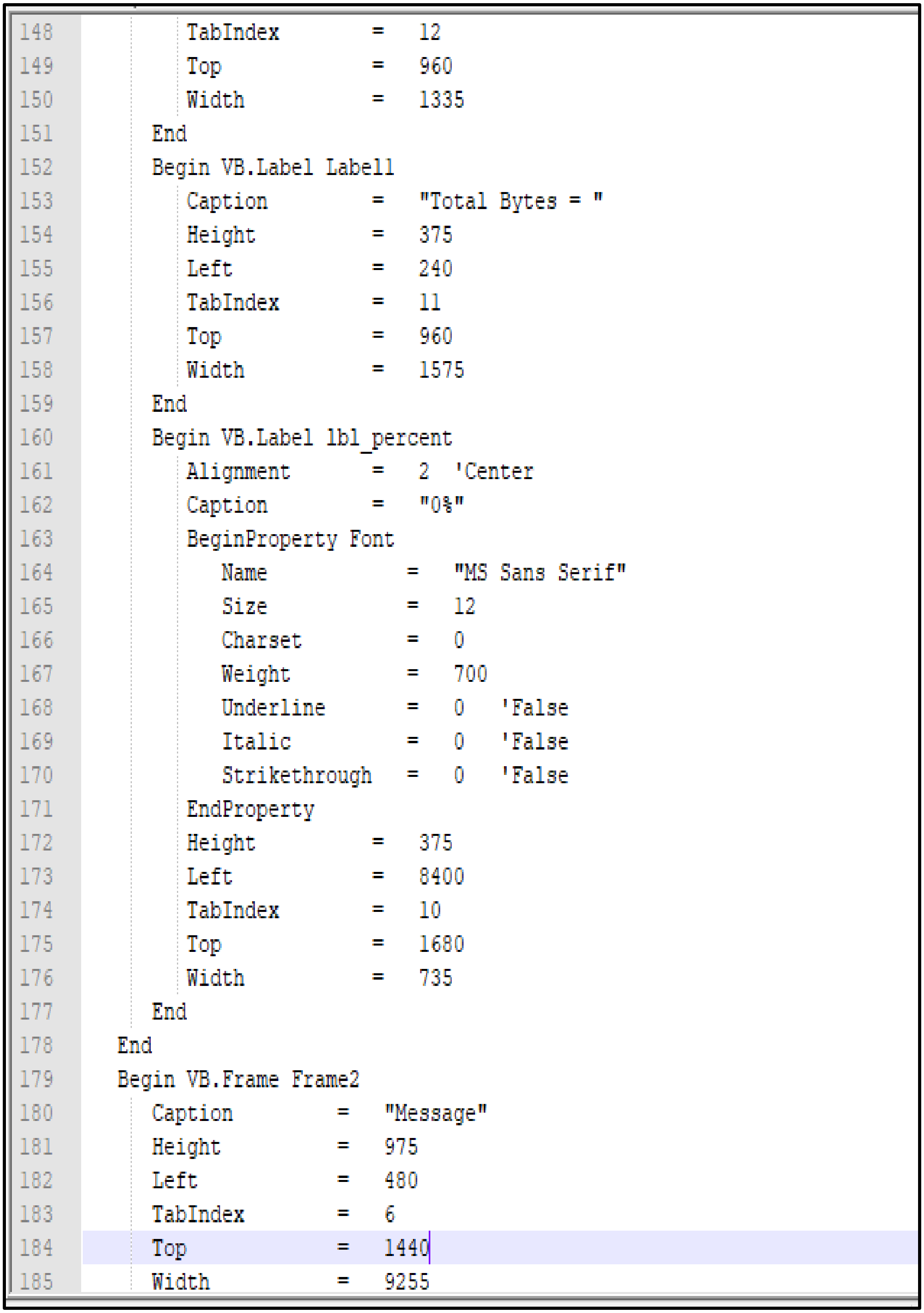
```
      Top             =   3240
      Width           =   1215
   End
   Begin VB.CommandButton cmd_cal
      Caption         =   "Calibrate"
      Height          =   615
      Left            =   10080
      TabIndex        =   21
      Top             =   5280
      Width           =   1215
   End
   Begin VB.Frame Frame4
      Caption         =   "Calibrate"
      Height          =   1095
      Left            =   480
      TabIndex        =   19
      Top             =   5040
      Width           =   9255
      Begin VB.TextBox txt_cal
         Alignment       =   2  'Center
         Height          =   420
         Left            =   720
         TabIndex        =   20
         Top             =   360
         Width           =   615
      End
   End
   Begin VB.CommandButton cmd_exit
      Caption         =   "Exit"
      Height          =   615
      Left            =   10080
      TabIndex        =   18
      Top             =   360
      Width           =   1215
   End
   Begin VB.TextBox Text1
      Height          =   420
      Left            =   360
```



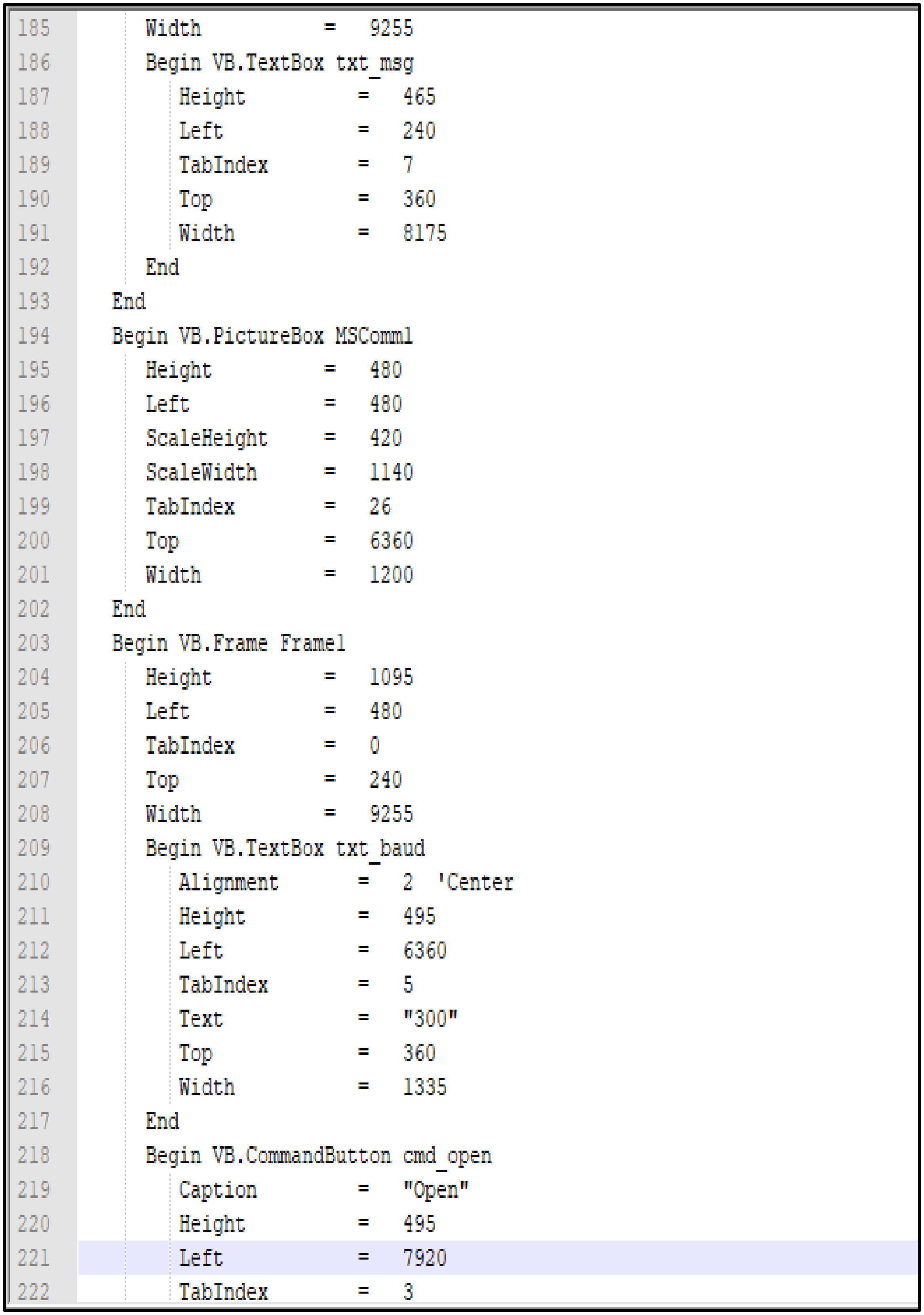
```
      Left            =   360
      TabIndex        =   17
      Top             =   8400
      Width           =   9615
   End
   Begin VB.PictureBox CommonDialog1
      Height          =   480
      Left            =   1200
      ScaleHeight     =   420
      ScaleWidth      =   1140
      TabIndex        =   25
      Top             =   6360
      Width           =   1200
   End
   Begin VB.Frame Frame3
      Caption         =   "File"
      Height          =   2415
      Left            =   480
      TabIndex        =   8
      Top             =   2520
      Width           =   9255
      Begin VB.PictureBox ProgressBar1
         Height          =   375
         Left            =   240
         ScaleHeight     =   315
         ScaleWidth      =   8115
         TabIndex        =   22
         Top             =   1680
         Width           =   8175
      End
      Begin VB.TextBox txt_file
         Alignment       =   2  'Center
         Height          =   420
         Left            =   240
         TabIndex        =   9
         Top             =   360
         Width           =   8175
      End
```



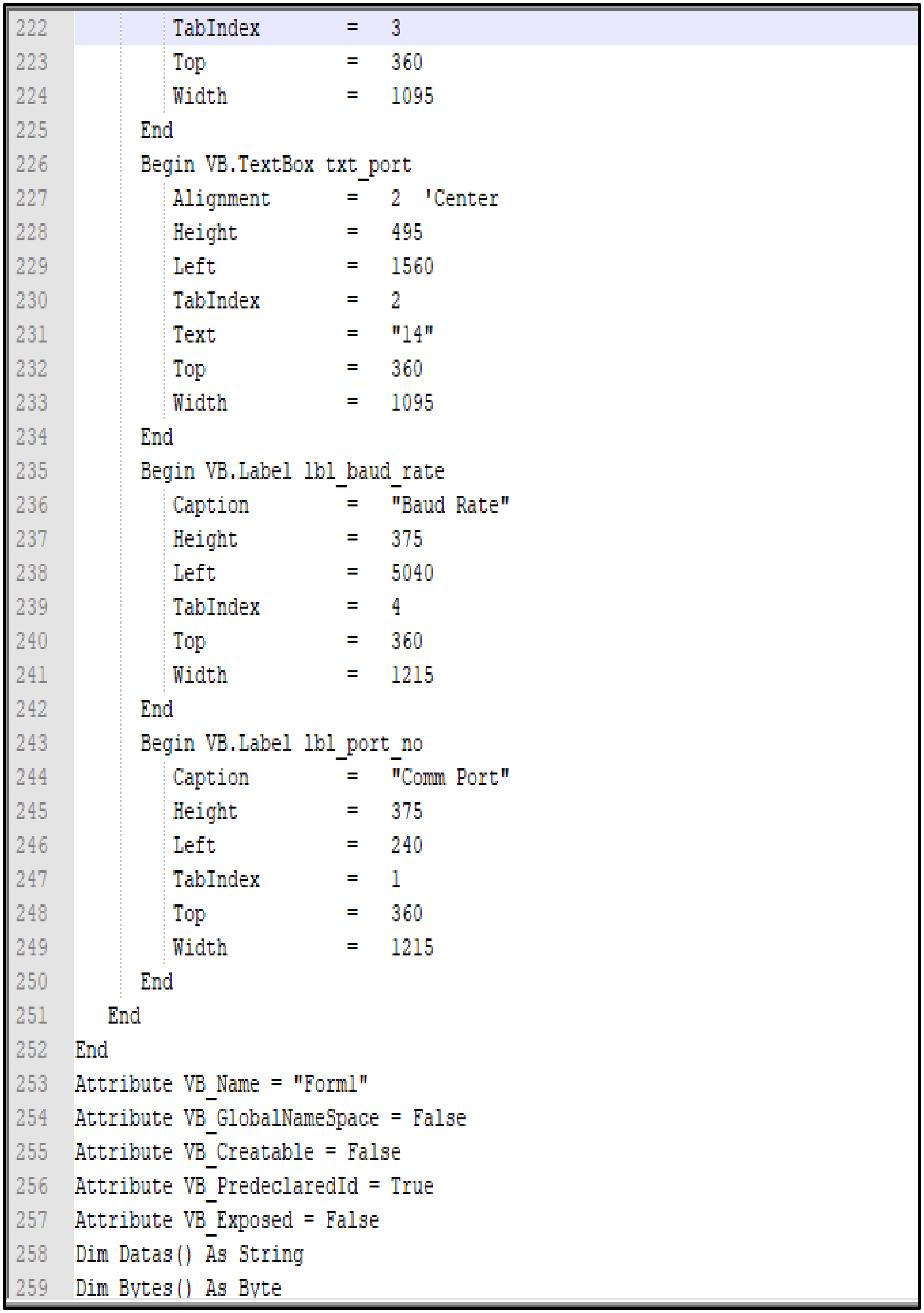
```
    End
    Begin VB.Label lbl_remaining
       Caption         =   "0"
       Height          =   375
       Left            =   7320
       TabIndex        =   16
       Top             =   960
       Width           =   1215
    End
    Begin VB.Label Label3
       Caption         =   "Remaining = "
       Height          =   375
       Left            =   5880
       TabIndex        =   15
       Top             =   960
       Width           =   1335
    End
    Begin VB.Label lbl_received
       Caption         =   "0"
       Height          =   375
       Left            =   4440
       TabIndex        =   14
       Top             =   960
       Width           =   1455
    End
    Begin VB.Label Label2
       Caption         =   "Received = "
       Height          =   375
       Left            =   3120
       TabIndex        =   13
       Top             =   960
       Width           =   1215
    End
    Begin VB.Label lbl_total
       Caption         =   "0"
       Height          =   375
       Left            =   1680
       TabIndex        =   12
```



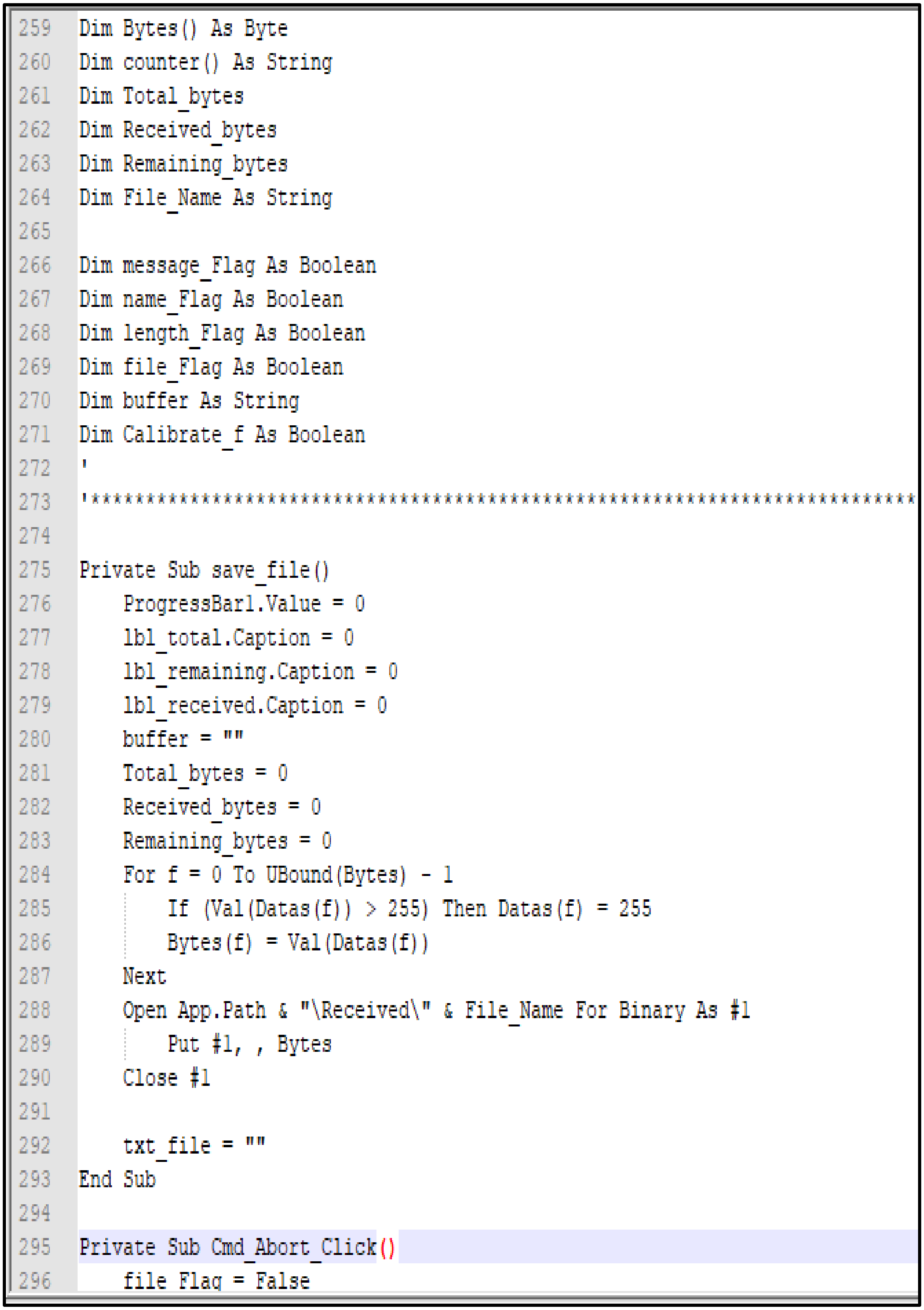
```
            TabIndex        =   12
            Top             =   960
            Width           =   1335
        End
        Begin VB.Label Label1
            Caption         =   "Total Bytes = "
            Height          =   375
            Left            =   240
            TabIndex        =   11
            Top             =   960
            Width           =   1575
        End
        Begin VB.Label lbl_percent
            Alignment       =   2  'Center
            Caption         =   "0%"
            BeginProperty Font
                Name            =   "MS Sans Serif"
                Size            =   12
                Charset         =   0
                Weight          =   700
                Underline       =   0   'False
                Italic          =   0   'False
                Strikethrough   =   0   'False
            EndProperty
            Height          =   375
            Left            =   8400
            TabIndex        =   10
            Top             =   1680
            Width           =   735
        End
    End
    Begin VB.Frame Frame2
        Caption         =   "Message"
        Height          =   975
        Left            =   480
        TabIndex        =   6
        Top             =   1440
        Width           =   9255
```



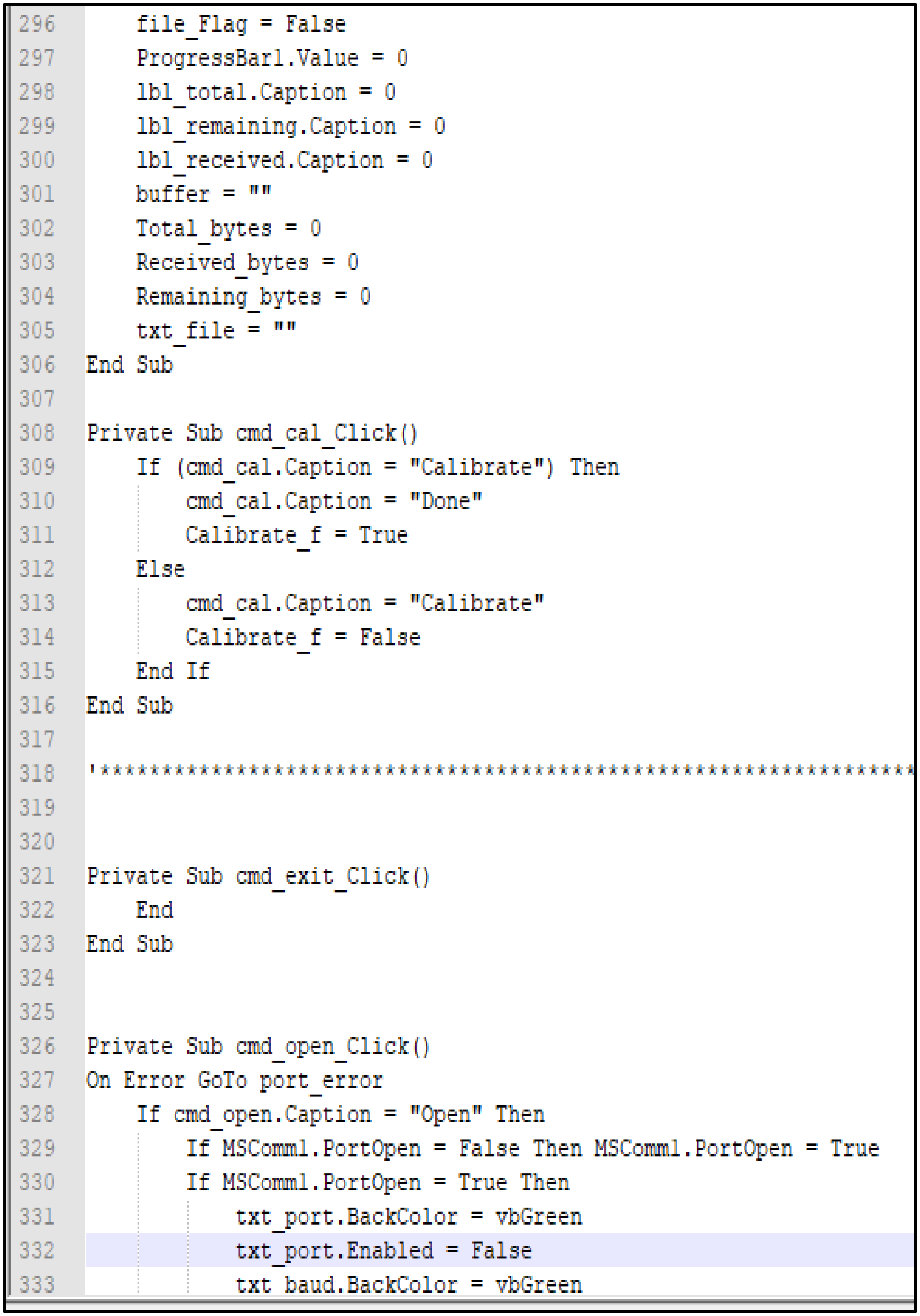
```
        Width           =   9255
        Begin VB.TextBox txt_msg
            Height          =   465
            Left            =   240
            TabIndex        =   7
            Top             =   360
            Width           =   8175
        End
    End
    Begin VB.PictureBox MSComm1
        Height          =   480
        Left            =   480
        ScaleHeight     =   420
        ScaleWidth      =   1140
        TabIndex        =   26
        Top             =   6360
        Width           =   1200
    End
    Begin VB.Frame Frame1
        Height          =   1095
        Left            =   480
        TabIndex        =   0
        Top             =   240
        Width           =   9255
        Begin VB.TextBox txt_baud
            Alignment       =   2  'Center
            Height          =   495
            Left            =   6360
            TabIndex        =   5
            Text            =   "300"
            Top             =   360
            Width           =   1335
        End
        Begin VB.CommandButton cmd_open
            Caption         =   "Open"
            Height          =   495
            Left            =   7920
            TabIndex        =   3
```



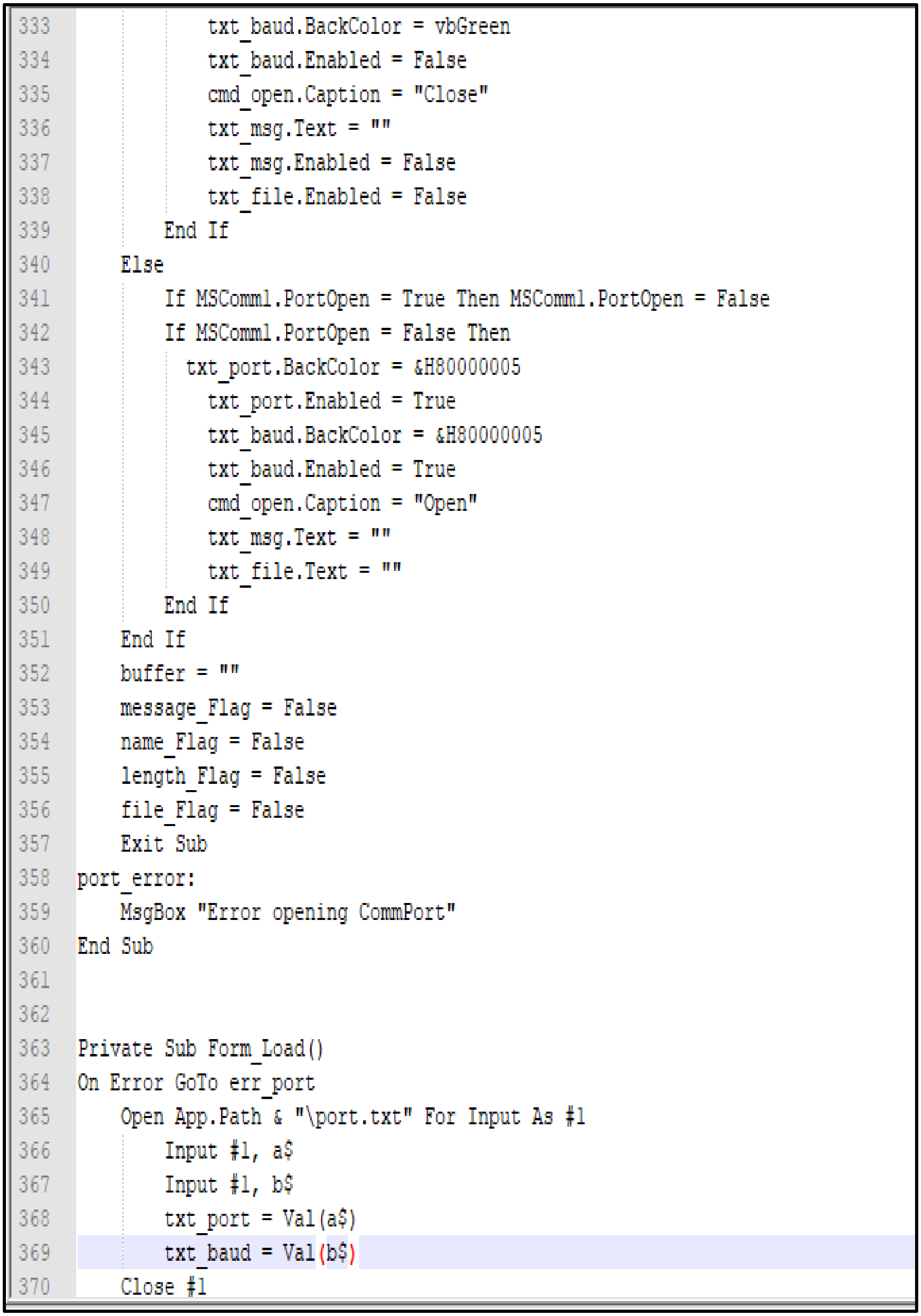
```
         TabIndex        =   3
         Top             =   360
         Width           =   1095
      End
      Begin VB.TextBox txt_port
         Alignment       =   2  'Center
         Height          =   495
         Left            =   1560
         TabIndex        =   2
         Text            =   "14"
         Top             =   360
         Width           =   1095
      End
      Begin VB.Label lbl_baud_rate
         Caption         =   "Baud Rate"
         Height          =   375
         Left            =   5040
         TabIndex        =   4
         Top             =   360
         Width           =   1215
      End
      Begin VB.Label lbl_port_no
         Caption         =   "Comm Port"
         Height          =   375
         Left            =   240
         TabIndex        =   1
         Top             =   360
         Width           =   1215
      End
   End
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Dim Datas() As String
Dim Bytes() As Byte
```



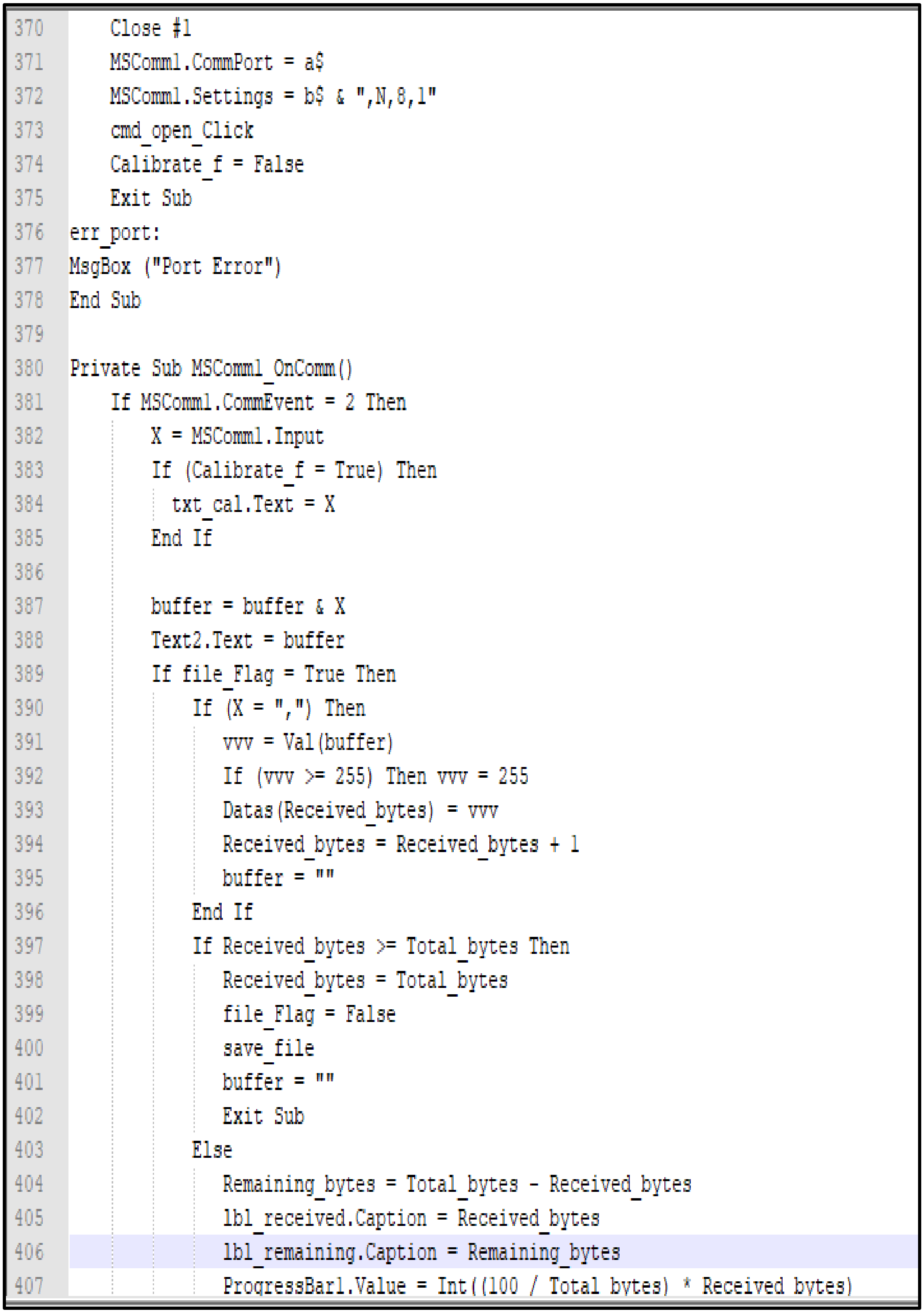
```
Dim Bytes() As Byte
Dim counter() As String
Dim Total_bytes
Dim Received_bytes
Dim Remaining_bytes
Dim File_Name As String

Dim message_Flag As Boolean
Dim name_Flag As Boolean
Dim length_Flag As Boolean
Dim file_Flag As Boolean
Dim buffer As String
Dim Calibrate_f As Boolean
'
'***************************************************************************

Private Sub save_file()
    ProgressBar1.Value = 0
    lbl_total.Caption = 0
    lbl_remaining.Caption = 0
    lbl_received.Caption = 0
    buffer = ""
    Total_bytes = 0
    Received_bytes = 0
    Remaining_bytes = 0
    For f = 0 To UBound(Bytes) - 1
        If (Val(Datas(f)) > 255) Then Datas(f) = 255
        Bytes(f) = Val(Datas(f))
    Next
    Open App.Path & "\Received\" & File_Name For Binary As #1
        Put #1, , Bytes
    Close #1

    txt_file = ""
End Sub

Private Sub Cmd_Abort_Click()
    file_Flag = False
```



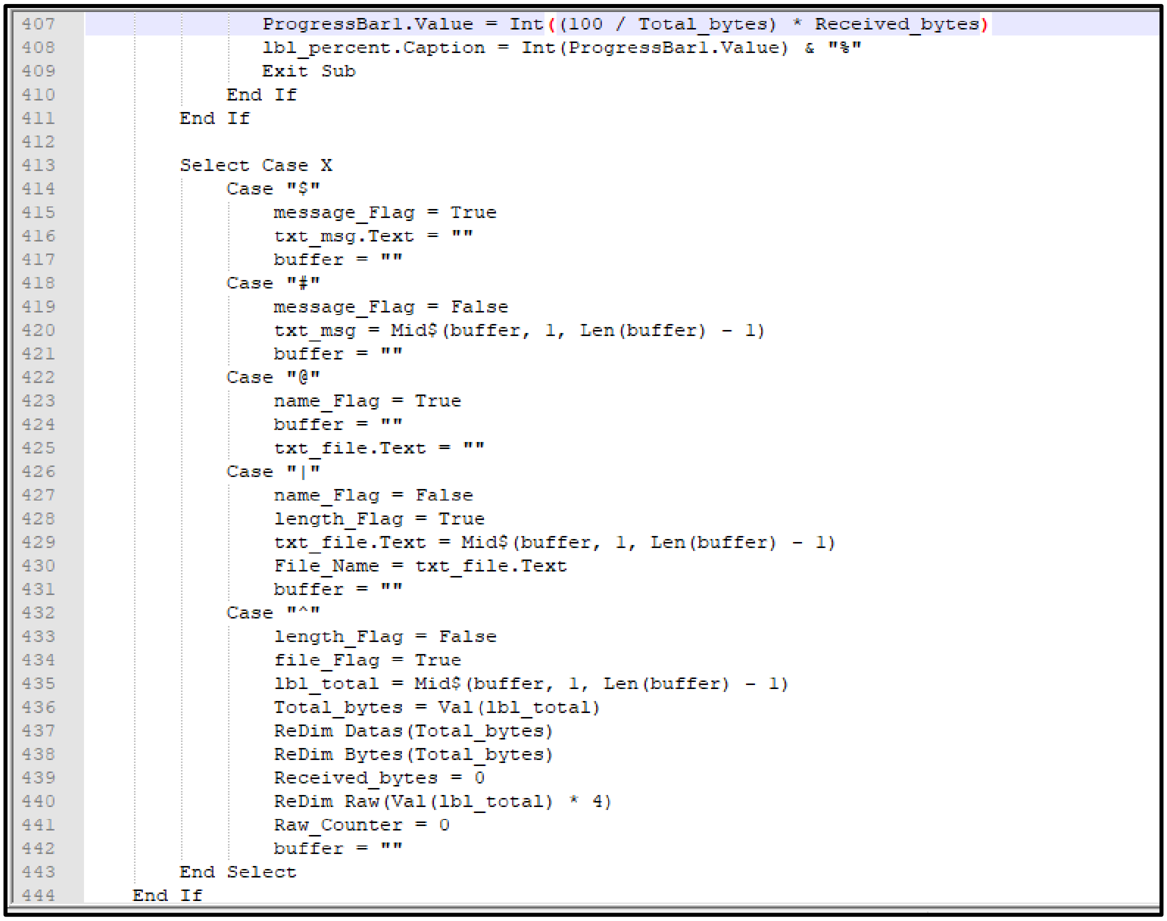
```
    file_Flag = False
    ProgressBar1.Value = 0
    lbl_total.Caption = 0
    lbl_remaining.Caption = 0
    lbl_received.Caption = 0
    buffer = ""
    Total_bytes = 0
    Received_bytes = 0
    Remaining_bytes = 0
    txt_file = ""
End Sub

Private Sub cmd_cal_Click()
    If (cmd_cal.Caption = "Calibrate") Then
        cmd_cal.Caption = "Done"
        Calibrate_f = True
    Else
        cmd_cal.Caption = "Calibrate"
        Calibrate_f = False
    End If
End Sub

'******************************************************************


Private Sub cmd_exit_Click()
    End
End Sub


Private Sub cmd_open_Click()
On Error GoTo port_error
    If cmd_open.Caption = "Open" Then
        If MSComm1.PortOpen = False Then MSComm1.PortOpen = True
        If MSComm1.PortOpen = True Then
            txt_port.BackColor = vbGreen
            txt_port.Enabled = False
            txt_baud.BackColor = vbGreen
```



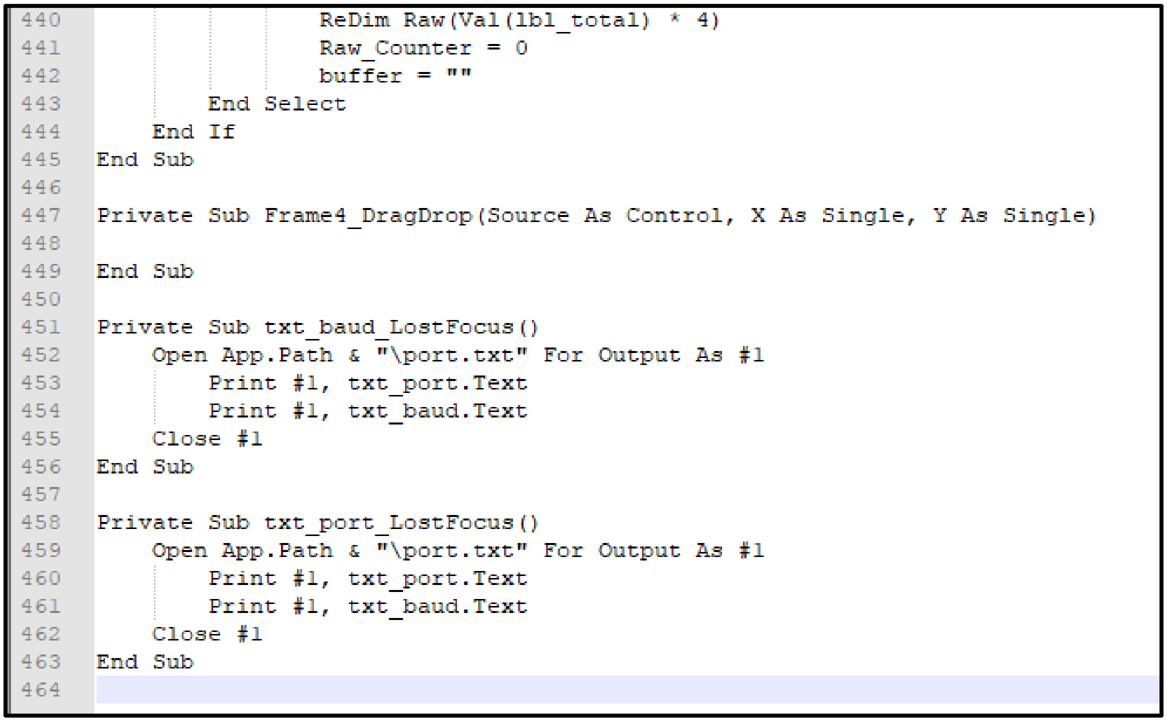
```
            txt_baud.BackColor = vbGreen
            txt_baud.Enabled = False
            cmd_open.Caption = "Close"
            txt_msg.Text = ""
            txt_msg.Enabled = False
            txt_file.Enabled = False
        End If
    Else
        If MSComm1.PortOpen = True Then MSComm1.PortOpen = False
        If MSComm1.PortOpen = False Then
          txt_port.BackColor = &H80000005
            txt_port.Enabled = True
            txt_baud.BackColor = &H80000005
            txt_baud.Enabled = True
            cmd_open.Caption = "Open"
            txt_msg.Text = ""
            txt_file.Text = ""
        End If
    End If
    buffer = ""
    message_Flag = False
    name_Flag = False
    length_Flag = False
    file_Flag = False
    Exit Sub
port_error:
    MsgBox "Error opening CommPort"
End Sub


Private Sub Form_Load()
On Error GoTo err_port
    Open App.Path & "\port.txt" For Input As #1
        Input #1, a$
        Input #1, b$
        txt_port = Val(a$)
        txt_baud = Val(b$)
    Close #1
```



```
    Close #1
    MSComm1.CommPort = a$
    MSComm1.Settings = b$ & ",N,8,1"
    cmd_open_Click
    Calibrate_f = False
    Exit Sub
err_port:
MsgBox ("Port Error")
End Sub

Private Sub MSComm1_OnComm()
    If MSComm1.CommEvent = 2 Then
        X = MSComm1.Input
        If (Calibrate_f = True) Then
          txt_cal.Text = X
        End If

        buffer = buffer & X
        Text2.Text = buffer
        If file_Flag = True Then
            If (X = ",") Then
               vvv = Val(buffer)
               If (vvv >= 255) Then vvv = 255
               Datas(Received_bytes) = vvv
               Received_bytes = Received_bytes + 1
               buffer = ""
            End If
            If Received_bytes >= Total_bytes Then
               Received_bytes = Total_bytes
               file_Flag = False
               save_file
               buffer = ""
               Exit Sub
            Else
               Remaining_bytes = Total_bytes - Received_bytes
               lbl_received.Caption = Received_bytes
               lbl_remaining.Caption = Remaining_bytes
               ProgressBar1.Value = Int((100 / Total_bytes) * Received_bytes)
```

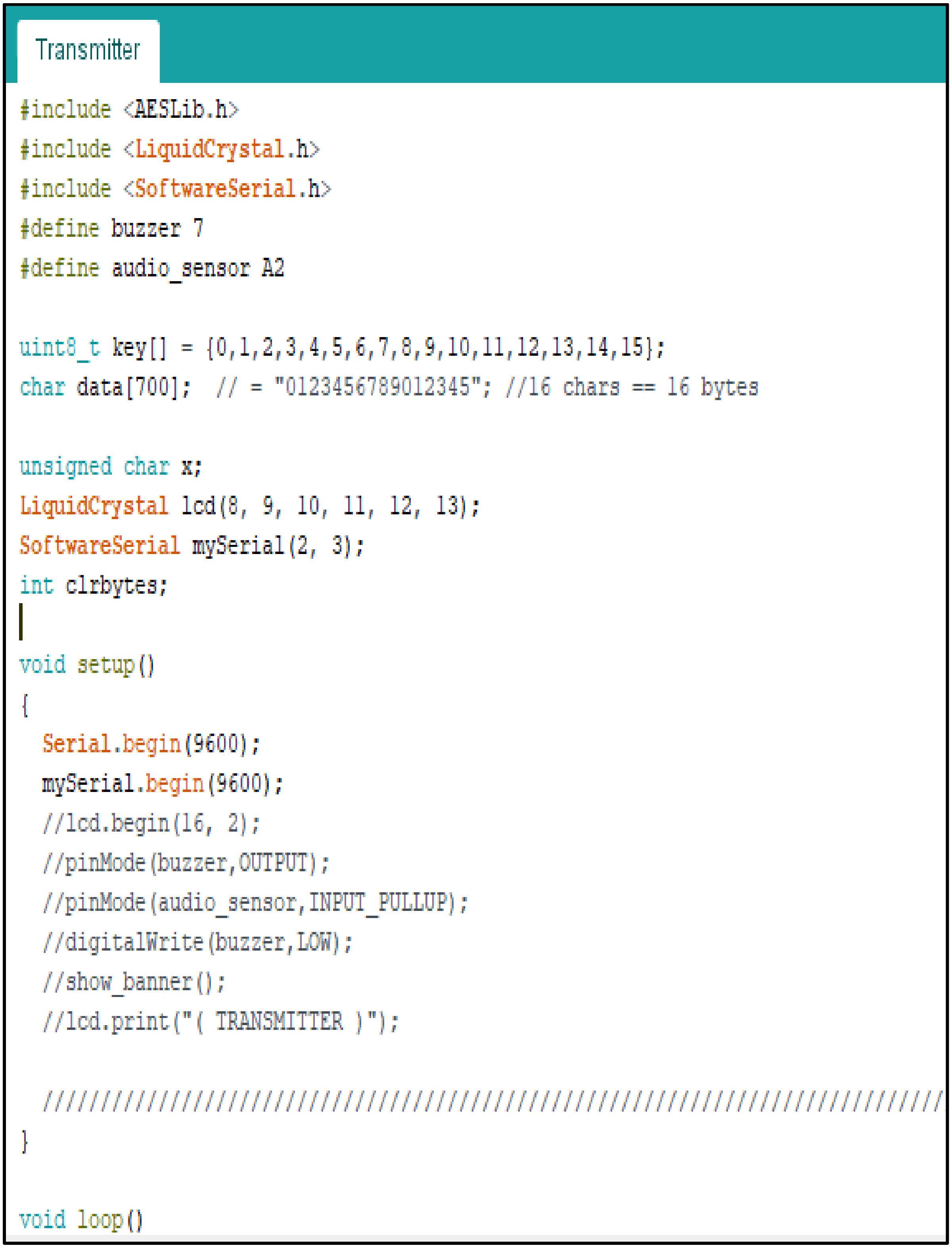


```
               ProgressBar1.Value = Int((100 / Total_bytes) * Received_bytes)
               lbl_percent.Caption = Int(ProgressBar1.Value) & "%"
               Exit Sub
            End If
        End If

        Select Case X
            Case "$"
                message_Flag = True
                txt_msg.Text = ""
                buffer = ""
            Case "#"
                message_Flag = False
                txt_msg = Mid$(buffer, 1, Len(buffer) - 1)
                buffer = ""
            Case "@"
                name_Flag = True
                buffer = ""
                txt_file.Text = ""
            Case "|"
                name_Flag = False
                length_Flag = True
                txt_file.Text = Mid$(buffer, 1, Len(buffer) - 1)
                File_Name = txt_file.Text
                buffer = ""
            Case "^"
                length_Flag = False
                file_Flag = True
                lbl_total = Mid$(buffer, 1, Len(buffer) - 1)
                Total_bytes = Val(lbl_total)
                ReDim Datas(Total_bytes)
                ReDim Bytes(Total_bytes)
                Received_bytes = 0
                ReDim Raw(Val(lbl_total) * 4)
                Raw_Counter = 0
                buffer = ""
        End Select
    End If
```

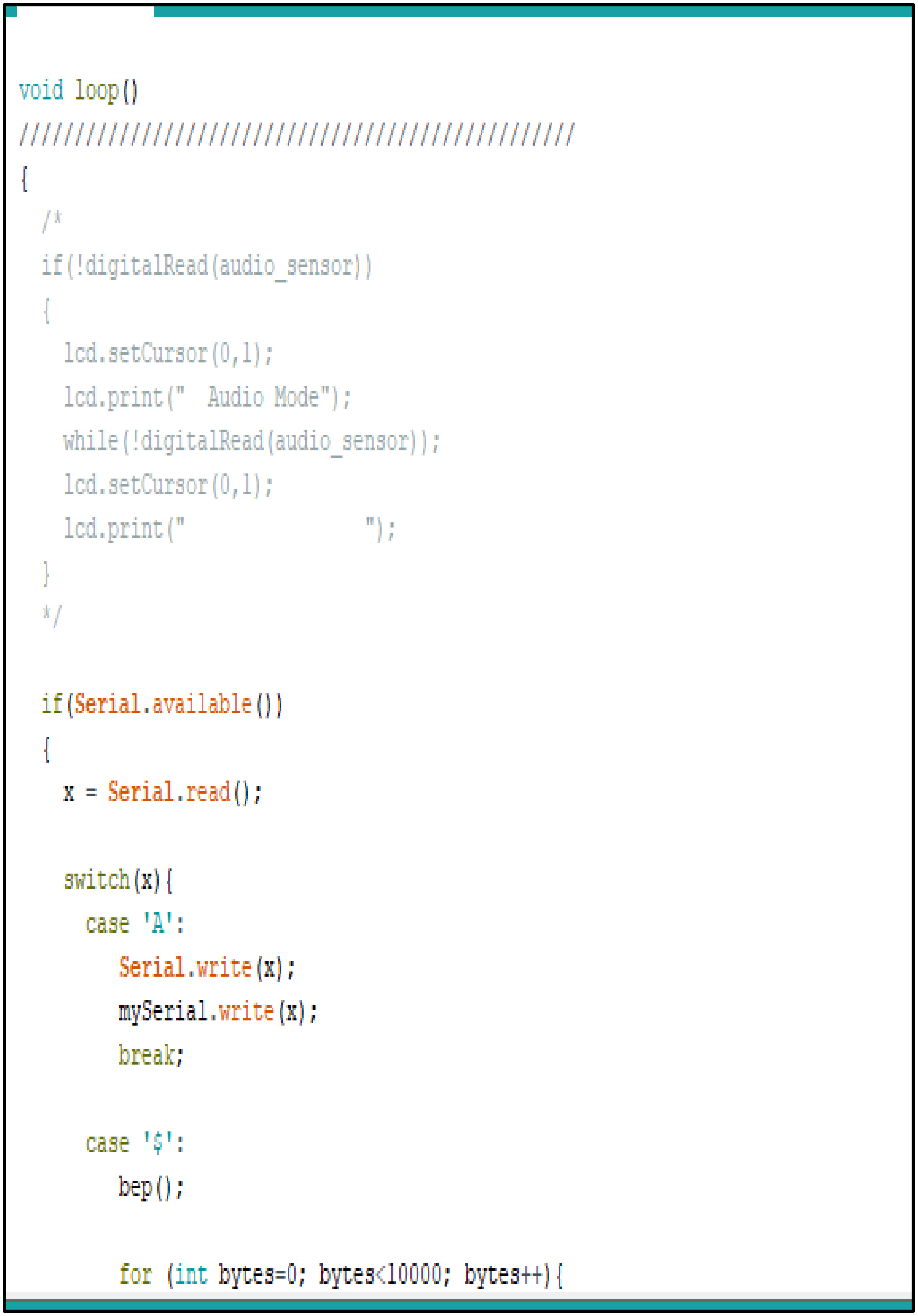


```
                ReDim Raw(Val(lbl_total) * 4)
                Raw_Counter = 0
                buffer = ""
        End Select
    End If
End Sub

Private Sub Frame4_DragDrop(Source As Control, X As Single, Y As Single)

End Sub

Private Sub txt_baud_LostFocus()
    Open App.Path & "\port.txt" For Output As #1
        Print #1, txt_port.Text
        Print #1, txt_baud.Text
    Close #1
End Sub

Private Sub txt_port_LostFocus()
    Open App.Path & "\port.txt" For Output As #1
        Print #1, txt_port.Text
        Print #1, txt_baud.Text
    Close #1
End Sub
```

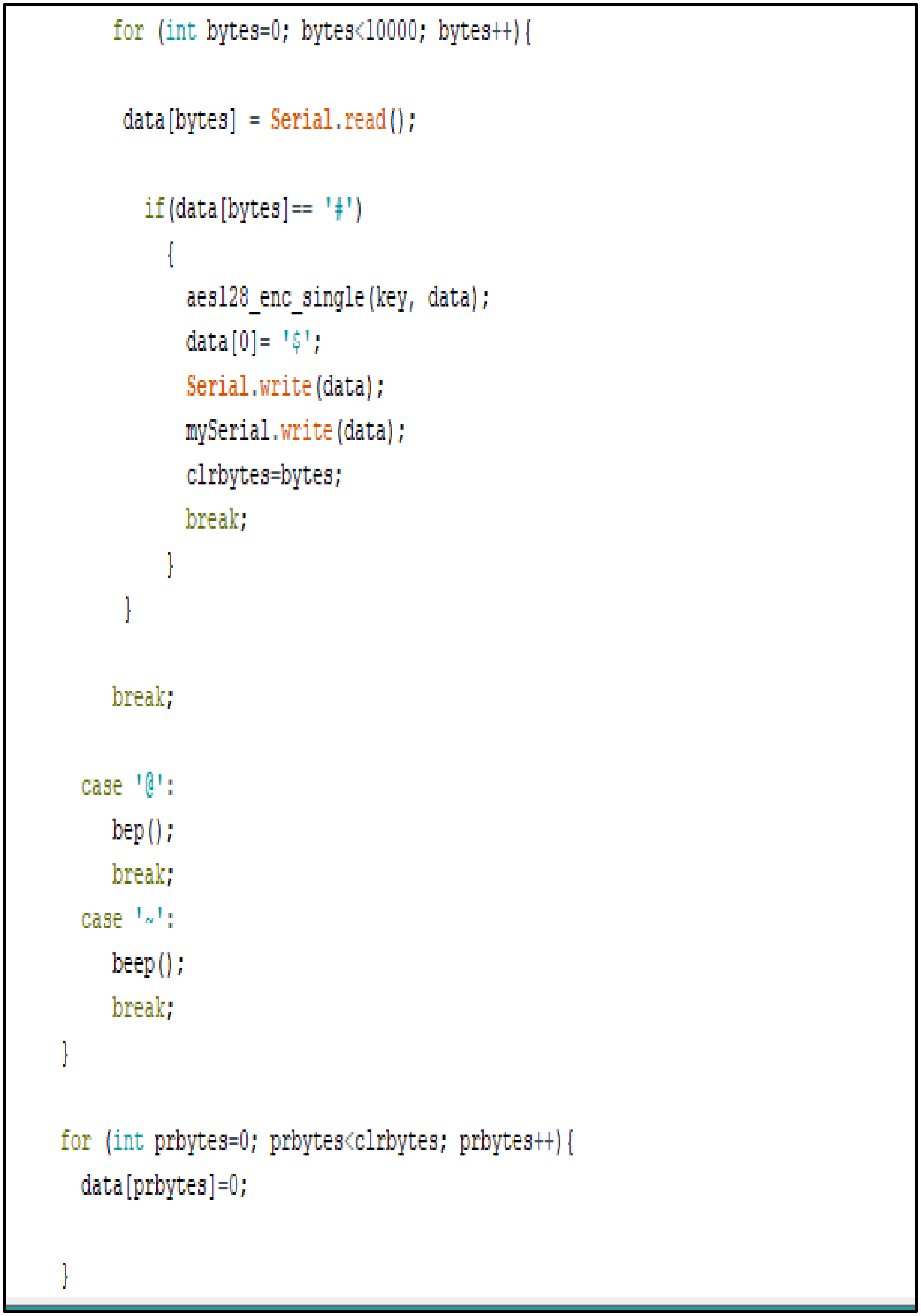
Code For Arduino Transmitter:



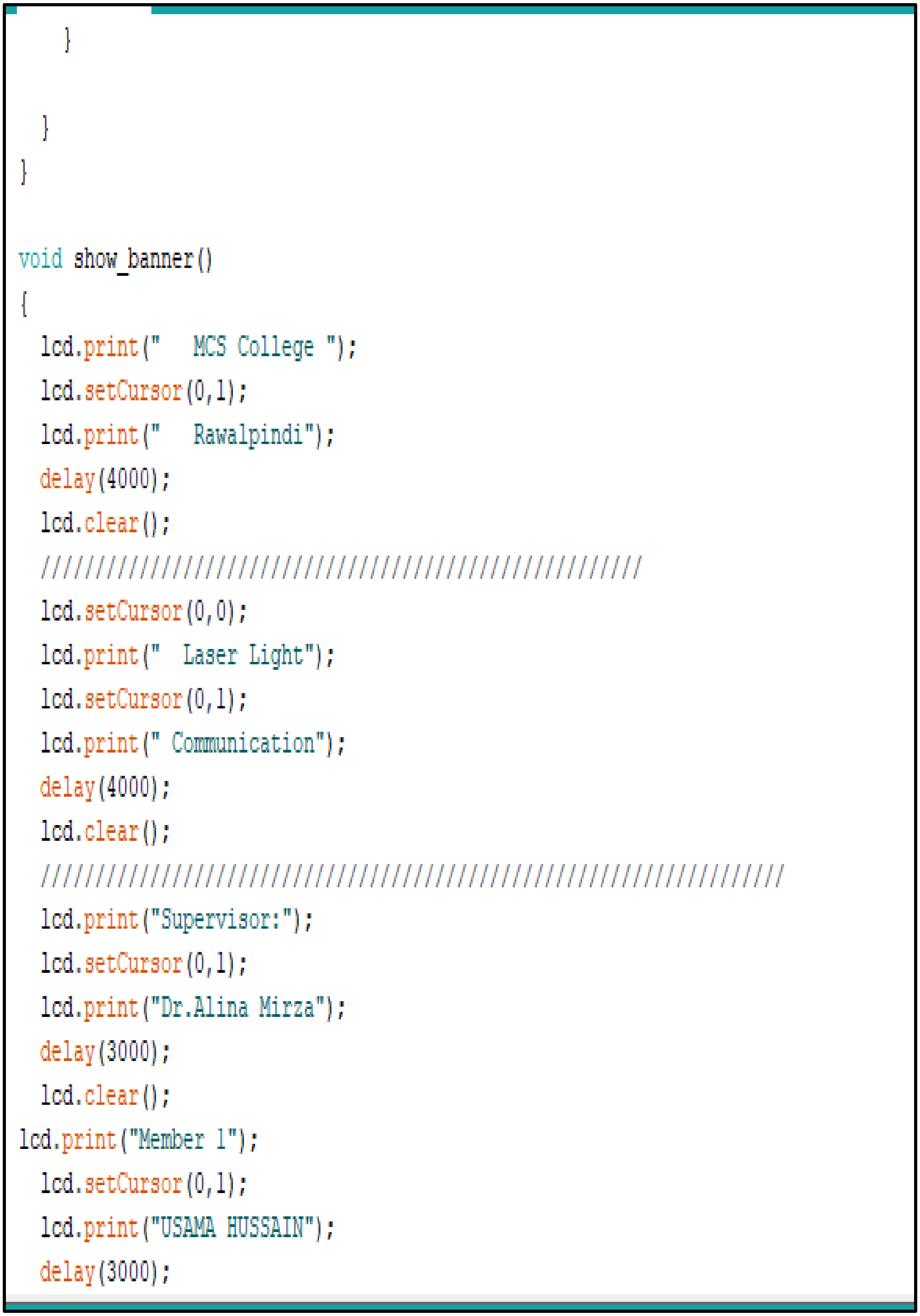
```
#include <AESLib.h>
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
#define buzzer 7
#define audio_sensor A2

uint8_t key[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
char data[700];  // = "0123456789012345"; //16 chars == 16 bytes

unsigned char x;
LiquidCrystal lcd(8, 9, 10, 11, 12, 13);
SoftwareSerial mySerial(2, 3);
int clrbytes;

void setup()
{
  Serial.begin(9600);
  mySerial.begin(9600);
  //lcd.begin(16, 2);
  //pinMode(buzzer,OUTPUT);
  //pinMode(audio_sensor,INPUT_PULLUP);
  //digitalWrite(buzzer,LOW);
  //show_banner();
  //lcd.print("( TRANSMITTER )");

  //////////////////////////////////////////////////////////////////////////////
}

void loop()
```



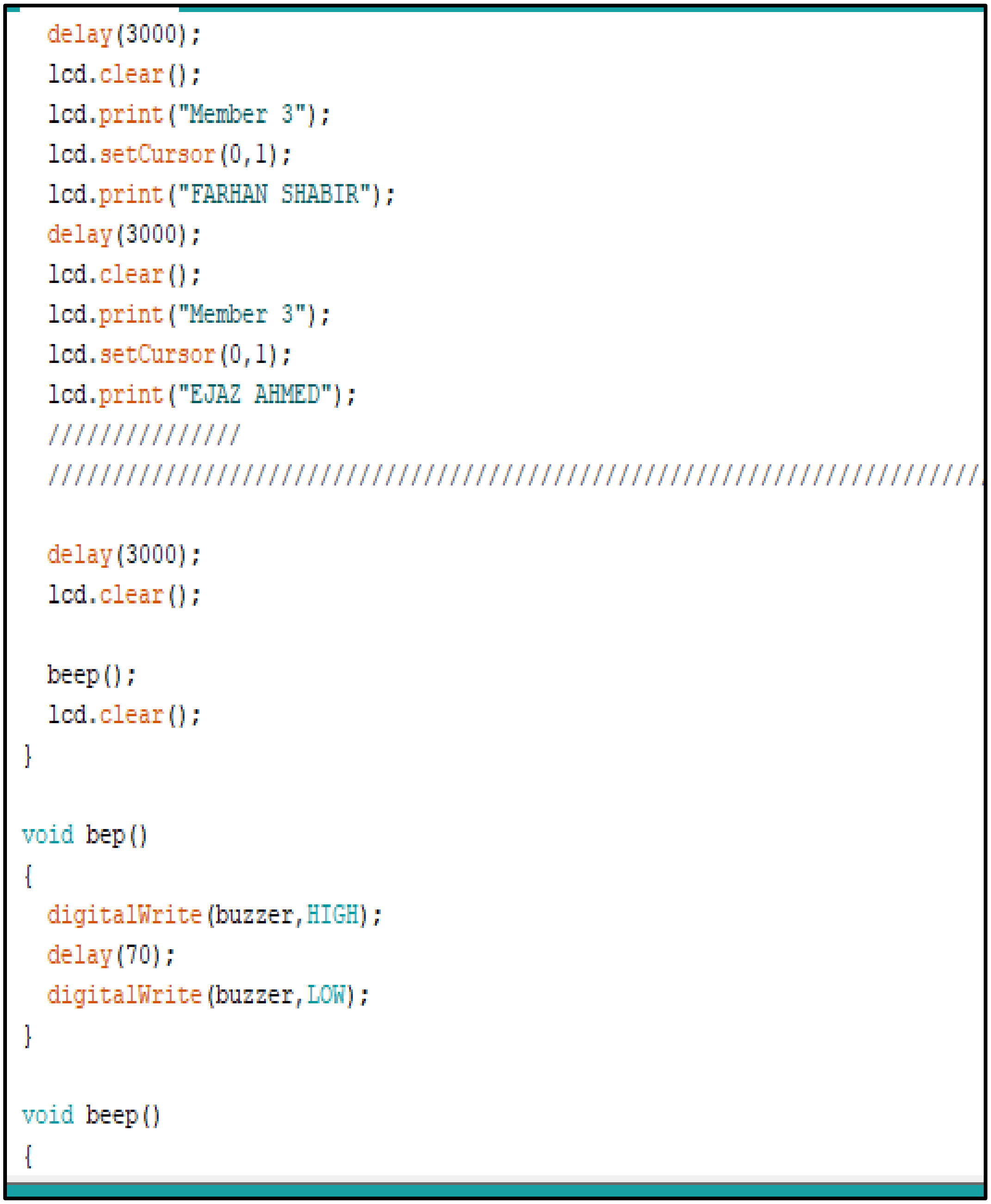
```
void loop()
//////////////////////////////////////////////////
{
  /*
  if(!digitalRead(audio_sensor))
  {
    lcd.setCursor(0,1);
    lcd.print("  Audio Mode");
    while(!digitalRead(audio_sensor));
    lcd.setCursor(0,1);
    lcd.print("                ");
  }
  */

  if(Serial.available())
  {
    x = Serial.read();

    switch(x){
      case 'A':
        Serial.write(x);
        mySerial.write(x);
        break;

      case '$':
        bep();

        for (int bytes=0; bytes<10000; bytes++){
```



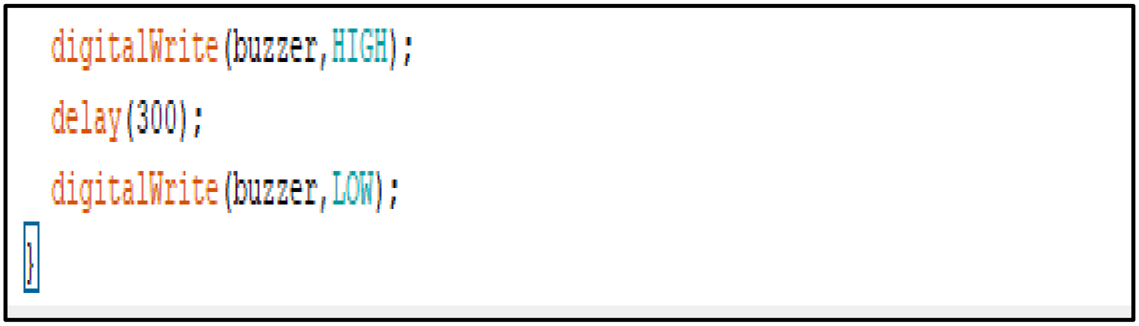
```
     for (int bytes=0; bytes<10000; bytes++){

      data[bytes] = Serial.read();

        if(data[bytes]== '#')
          {
            aes128_enc_single(key, data);
            data[0]= '$';
            Serial.write(data);
            mySerial.write(data);
            clrbytes=bytes;
            break;
          }
      }

     break;

  case '@':
     bep();
     break;
  case '~':
     beep();
     break;
}

for (int prbytes=0; prbytes<clrbytes; prbytes++){
  data[prbytes]=0;

}
```



```
    }


  }
}

void show_banner()
{
  lcd.print("   MCS College ");
  lcd.setCursor(0,1);
  lcd.print("   Rawalpindi");
  delay(4000);
  lcd.clear();
  ////////////////////////////////////////////////////////
  lcd.setCursor(0,0);
  lcd.print("  Laser Light");
  lcd.setCursor(0,1);
  lcd.print(" Communication");
  delay(4000);
  lcd.clear();
  /////////////////////////////////////////////////////////////////////
  lcd.print("Supervisor:");
  lcd.setCursor(0,1);
  lcd.print("Dr.Alina Mirza");
  delay(3000);
  lcd.clear();
lcd.print("Member 1");
  lcd.setCursor(0,1);
  lcd.print("USAMA HUSSAIN");
  delay(3000);
```

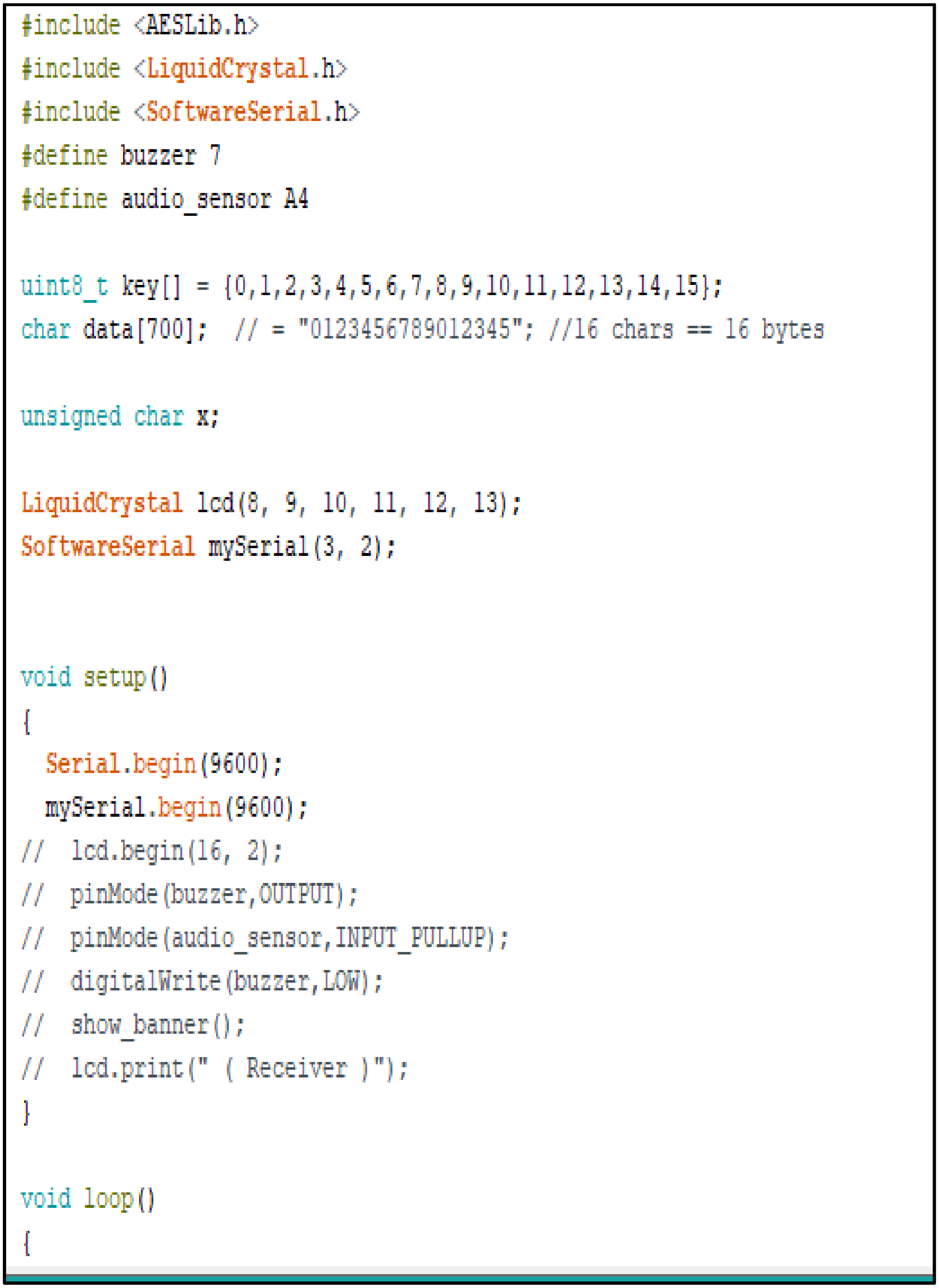


```
  delay(3000);
  lcd.clear();
  lcd.print("Member 3");
  lcd.setCursor(0,1);
  lcd.print("FARHAN SHABIR");
  delay(3000);
  lcd.clear();
  lcd.print("Member 3");
  lcd.setCursor(0,1);
  lcd.print("EJAZ AHMED");
  ///////////////
  /////////////////////////////////////////////////////////////////////////

  delay(3000);
  lcd.clear();

  beep();
  lcd.clear();
}

void bep()
{
  digitalWrite(buzzer,HIGH);
  delay(70);
  digitalWrite(buzzer,LOW);
}

void beep()
{
```

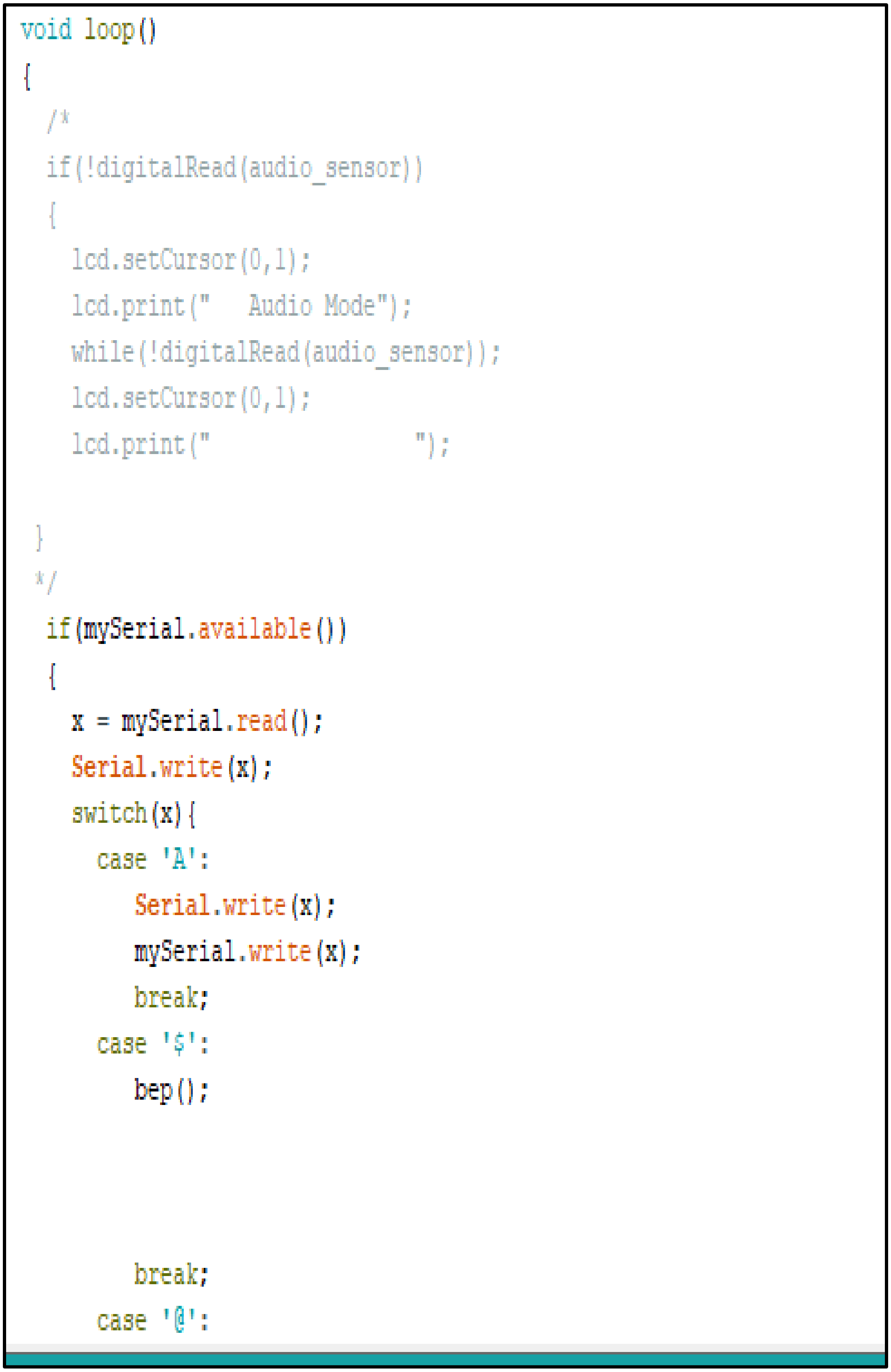


```
  digitalWrite(buzzer,HIGH);
  delay(300);
  digitalWrite(buzzer,LOW);
}
```

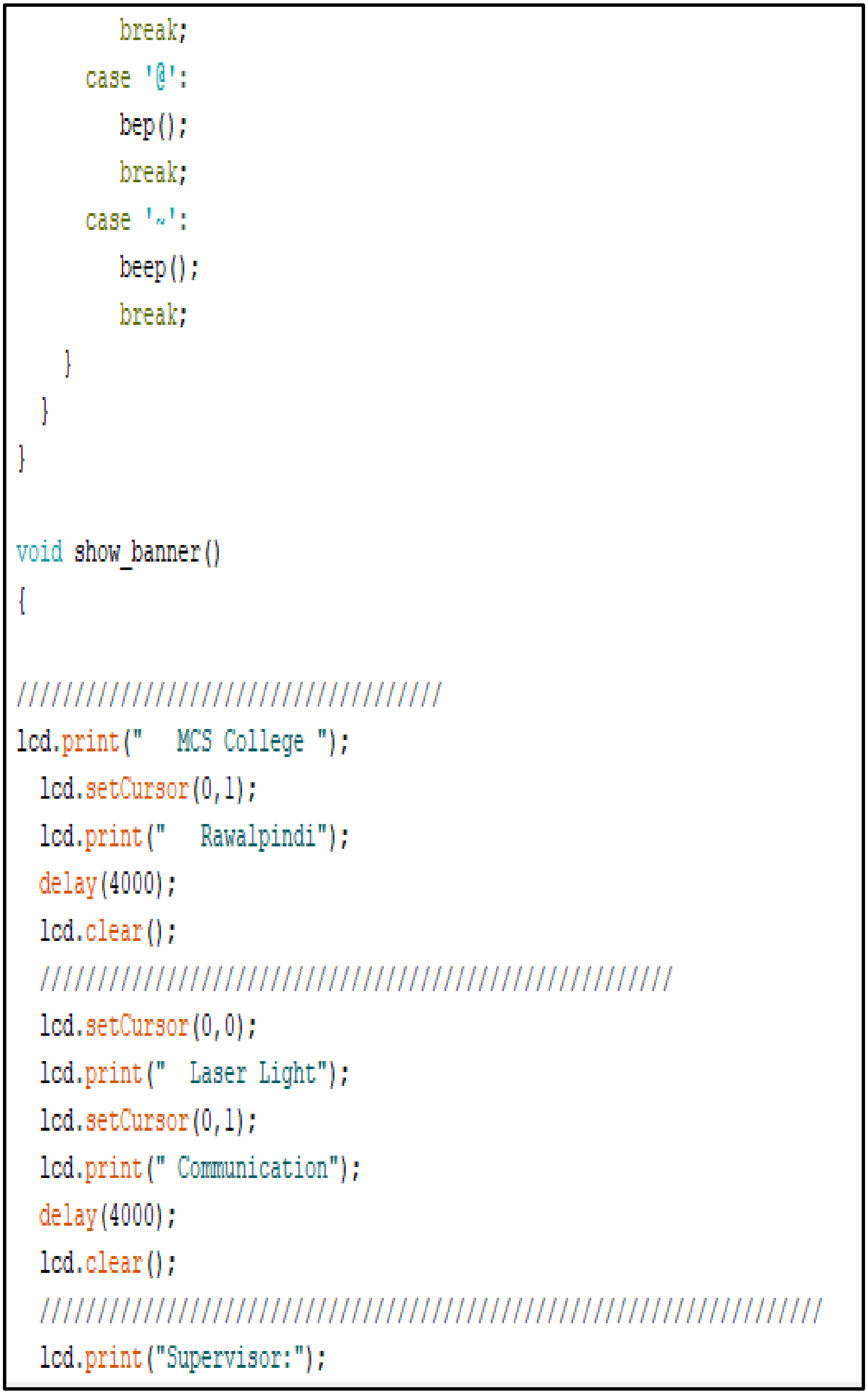
Code for Arduino Receiver:



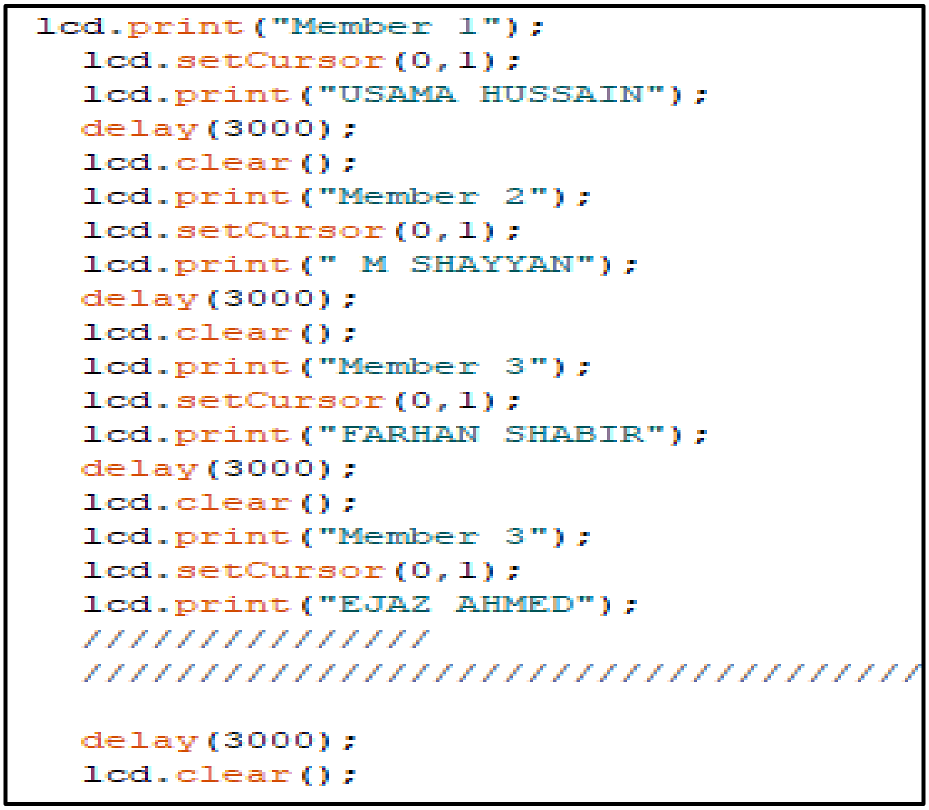
```
#include <AESLib.h>
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
#define buzzer 7
#define audio_sensor A4

uint8_t key[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
char data[700];  // = "0123456789012345"; //16 chars == 16 bytes

unsigned char x;

LiquidCrystal lcd(8, 9, 10, 11, 12, 13);
SoftwareSerial mySerial(3, 2);



void setup()
{
  Serial.begin(9600);
  mySerial.begin(9600);
//  lcd.begin(16, 2);
//  pinMode(buzzer,OUTPUT);
//  pinMode(audio_sensor,INPUT_PULLUP);
//  digitalWrite(buzzer,LOW);
//  show_banner();
//  lcd.print(" ( Receiver )");
}

void loop()
{
```



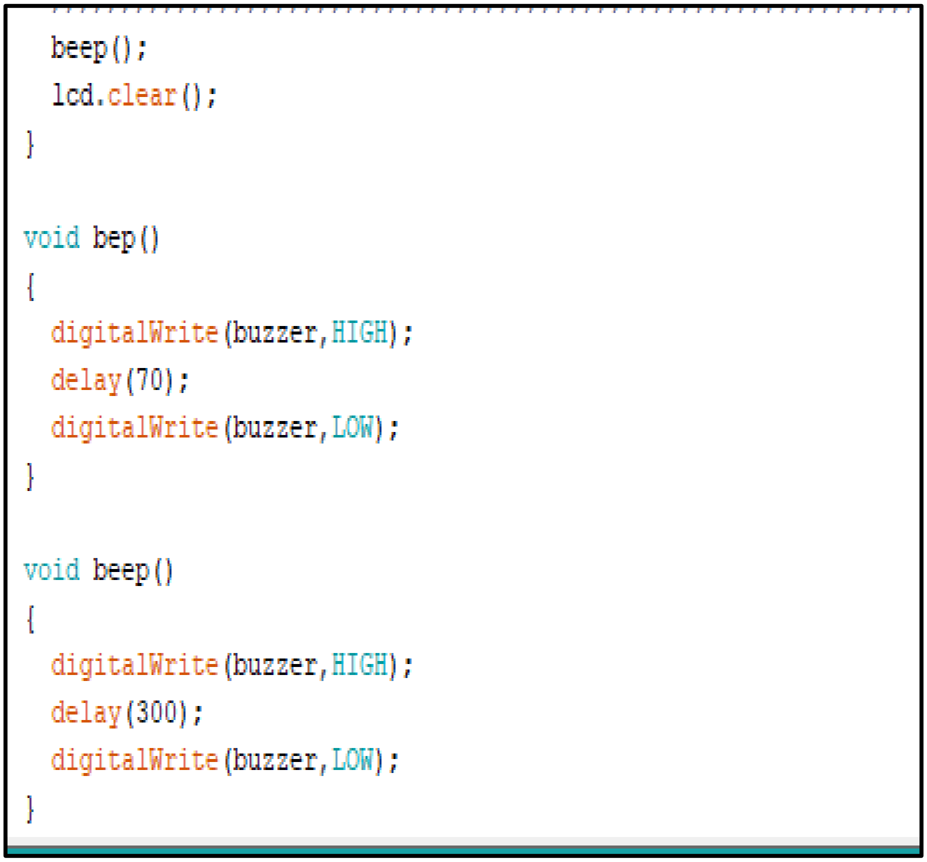
```
void loop()
{
  /*
  if(!digitalRead(audio_sensor))
  {
    lcd.setCursor(0,1);
    lcd.print("   Audio Mode");
    while(!digitalRead(audio_sensor));
    lcd.setCursor(0,1);
    lcd.print("                ");

 }
 */
  if(mySerial.available())
  {
    x = mySerial.read();
    Serial.write(x);
    switch(x){
      case 'A':
         Serial.write(x);
         mySerial.write(x);
         break;
      case '$':
         bep();



         break;
      case '@':
```



```
         break;
      case '@':
         bep();
         break;
      case '~':
         beep();
         break;
    }
  }
}

void show_banner()
{

/////////////////////////////////////
lcd.print("  MCS College ");
  lcd.setCursor(0,1);
  lcd.print("   Rawalpindi");
  delay(4000);
  lcd.clear();
  ///////////////////////////////////////////////////////
  lcd.setCursor(0,0);
  lcd.print("  Laser Light");
  lcd.setCursor(0,1);
  lcd.print(" Communication");
  delay(4000);
  lcd.clear();
  ////////////////////////////////////////////////////////////////////
  lcd.print("Supervisor:");
```



```
lcd.print("Member 1");
  lcd.setCursor(0,1);
  lcd.print("USAMA HUSSAIN");
  delay(3000);
  lcd.clear();
  lcd.print("Member 2");
  lcd.setCursor(0,1);
  lcd.print(" M SHAYYAN");
  delay(3000);
  lcd.clear();
  lcd.print("Member 3");
  lcd.setCursor(0,1);
  lcd.print("FARHAN SHABIR");
  delay(3000);
  lcd.clear();
  lcd.print("Member 3");
  lcd.setCursor(0,1);
  lcd.print("EJAZ AHMED");
  ////////////////
  /////////////////////////////////////

  delay(3000);
  lcd.clear();
```



```
  beep();
  lcd.clear();
}

void bep()
{
  digitalWrite(buzzer,HIGH);
  delay(70);
  digitalWrite(buzzer,LOW);
}

void beep()
{
  digitalWrite(buzzer,HIGH);
  delay(300);
  digitalWrite(buzzer,LOW);
}
```

LIST OF ABBREVATIONS:

VLC	Visible Light Communication
RF	Radio Frequency
IDE	Integrated Development Environment
LED	Light Emitting Diode
LASER	Light Amplification by Stimulated Emission of Radiation
GUI	Graphical User Interface
VB	Visual Basic
PCB	Printed Circuit Board
VLCC	Visible Light Communication Consotium
IEEE	Institute of Electrical and Electronics Engineers
OFDMA	Orthogonal Frequency-Division Multiple Access

REFERENCES

[1] Cui, Kaiyun, et al. "Line-of-sight visible light communication system design and demonstration." *2010 7th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP 2010)*. IEEE, 2010.

[2] Tsonev, Dobroslav, Stefan Videv, and Harald Haas. "Towards a 100 Gb/s visible light wireless access network." *Optics express* 23.2 (2015): 1627-1637.

[3] Borogovac, Tarik, and Thomas DC Little. "Laser visible light communications." *2012 IEEE Photonics Society Summer Topical Meeting Series*. IEEE, 2012.

[4] Yang, Jie, et al. "Highly uniform white light-based visible light communication using red, green, and blue laser diodes." *IEEE Photonics Journal* 10.2 (2018): 1-8.

[5] Khan, Latif Ullah. "Visible light communication: Applications, architecture, standardization and research challenges." *Digital Communications and Networks* 3.2 (2017): 78-88.

[6] Komine, Toshihiko, and Masao Nakagawa. "Fundamental analysis for visible-light communication system using LED lights." *IEEE transactions on Consumer Electronics* 50.1 (2004): 100-107.

[7] Rajagopal, Sridhar, Richard D. Roberts, and Sang-Kyu Lim. "IEEE 802.15. 7 visible light communication: modulation schemes and dimming support." *IEEE Communications Magazine* 50.3 (2012): 72-82.

[8] Wada-M, Yendo-T, Fujii-T, and Tanimoto-M: 'Road-to-vehicle communication using LED traffic light'. Proc. 2005 IEEE Intelligent Vehicles Symposium Proceedings. Las Vegas, NV, USA. IEEE Intelligent Transportation Syst. Soc. 6 8 June 2005., pp.

[9] Cossu, G., et al. "Experimental demonstration of high speed underwater visible light communications." *2013 2nd International Workshop on Optical Wireless Communications (IWOW)*. IEEE, 2013.

[10] Zafar, Fahad, Masuduzzaman Bakaul, and Rajendran Parthiban. "Laser-diode-based visible light communication: Toward gigabit class communication." *IEEE Communications Magazine* 55.2 (2017): 144-151.

[11] Chun, Hyunchae, et al. "Visible light communication using laser diode based remote phosphor technique." *2015 IEEE International Conference on Communication Workshop (ICCW)*. IEEE, 2015.

[12] Jovicic, Aleksandar, Junyi Li, and Tom Richardson. "Visible light communication: opportunities, challenges and the path to market." *IEEE Communications Magazine* 51.12 (2013): 26-32.

[13] Grobe, Liane, et al. "High-speed visible light communication systems." *IEEE communications magazine* 51.12 (2013): 60-66.

Bibliography

[14] https://bitlylink.com/Adysb

[15] https://bitlylink.com/5nXUQ

[16] https://bitlylink.com/wFHYK

[17] https://bitlylink.com/LO8OG

[18] https://bitlylink.com/VpkTw

[19] https://bitlylink.com/nGXNL

[20] https://bitlylink.com/ghvFd

[21] https://bitlylink.com/FAFV1

[22] https://bitlylink.com/iKCpi

[23] https://bitlylink.com/BVxGJ

[24] https://bitlylink.com/kcZN6

[25] https://bitlylink.com/3McNN