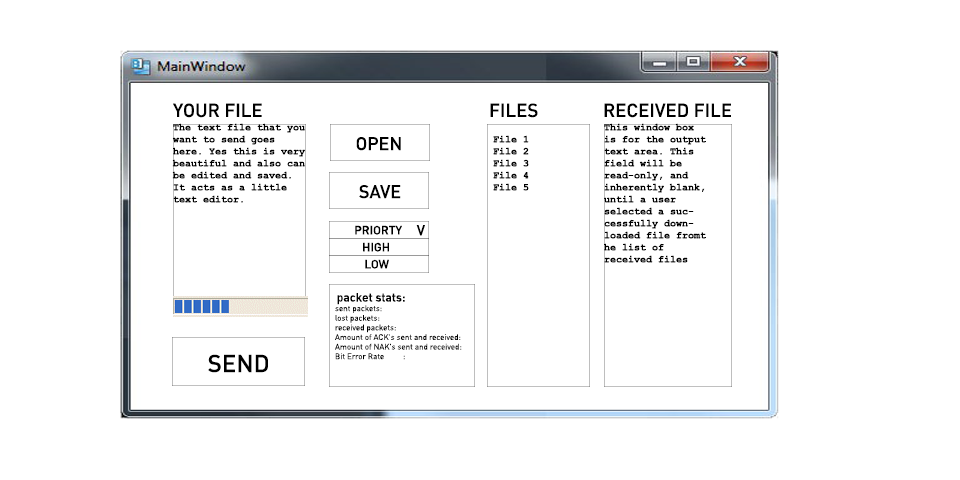


Team oscar

COMP 3980 Wireless Protocol





Jerry Jia

Jaeger SARAUEr

Oscar Kwan

Alvin Man

Contents

[**Driver Pseudocode** 2](#_Toc435563061)

[**BuildWindow – WinMain** 2](#_Toc435563062)

[**Enter Comm Param** 2](#_Toc435563063)

[**Command Mode – WndProc** 2](#_Toc435563064)

[**File Loader** 2](#_Toc435563065)

[**Packetize Data** 3](#_Toc435563066)

[**Engine Write Thread Start** 3](#_Toc435563067)

[**Engine Read Thread Start** 3](#_Toc435563068)

[**Idle** 3](#_Toc435563069)

[**Read Thread – Forever loop** 4](#_Toc435563070)

[**Acknowledge Line** 4](#_Toc435563071)

[**Wait for data** 4](#_Toc435563072)

[**Packet Check** 4](#_Toc435563073)

[**Acknowledge Package** 5](#_Toc435563074)

[**Receiver side: Check Priority State** 5](#_Toc435563075)

[5](#_Toc435563076)

[**Send Thread – Ends when file transfer complete** 5](#_Toc435563077)

[**Confirm Line** 5](#_Toc435563078)

[**Wait for ACK** 5](#_Toc435563079)

[**Send Data** 5](#_Toc435563080)

[**Wait for Packet ACK** 5](#_Toc435563081)

[**Sender side: Check Priority State** 6](#_Toc435563082)

[**GUI Implementation** 7](#_Toc435563083)

### **Driver Pseudocode**

### **BuildWindow – WinMain**

Initialize window

Set window dimensions

Populate window with the several items:

File loader open button

Send button

Save Button

Priority Hierarchy Checkbox

Editable text areas for files to be sent

Non-editable text areas for files received

List view box that displays files received

Progress bar to keep track of the files received

Non-editable text area for statistics of each file

Define file opener structure

Assign file opener structure a parent window

**Transition** to **Enter Comm Param**

### **Enter Comm Param**

Bring up dialog box to allow user to set communication parameters

When user clicks OK

Get the parameters from the comm dialog

Set the parameters for the wireless modem

if read thread not started

**transition** to **Engine Read Thread Start**

else

**transition** to **Command Mode**

### **Command Mode – WndProc**

Waits for user inputs, such as button presses

If user selects high priority on PriorityCheckBox

Set **weHavePriority** to true

Else

Set **weHavePriority** to false

If user presses **Open** button

**transition** to **File Loader**

If user presses **Send**

if user has selected a file to send

**transition** to **Engine Write Thread Start**

else

prompt user to select a file first

### **File Loader**

User can select a file to send

if user selects and file and presses OK

Create a file loader thread

Parse through the file and put each lines onto the screen

update and display content in GUI

**transition** to **Command Mode**

### **Packetize Data**

Check the size of the file that was passed in

Set buffer size to 516 bytes

**Loop:** Iterate through file to packetize:

Set first byte in buffer to **SOH**(**0x01**)

Set second byte in buffer to **0 (0x30)** or **1 (0x31)** (alternating

sync bit)

Set 2-byte checksum to third and fourth byte in buffer

if remaining file data is larger than 512 bytes

loop from buffer index 0+4 to 516

add each char to buffer

else

loop from buffer index 0+4 to size of data

add each char to buffer

add **EOT** (**0x04**) after the last char in buffer

store packet into buffer

**Transition** to **Engine Write Thread Start**

### **Engine Write Thread Start**

*Note:* ***Packet array*** *is stored as a global variable*

Create thread handle

Initialize the write thread

Inside Write Thread:

**Loop:** Iterate through ***packet array***, pass in each packet to the

protocol engine, until no packets are left

**Transition** to **Command Mode**

### **Engine Read Thread Start**

Create thread handle

Initialize the read thread

Inside Read Thread:

Handles **Reading** state in the protocol engine until program terminates

**Transition** to **Command Mode** to allow user input

### 

### **Idle**

Set communication mask to EV\_RXCHAR

**Loop**:

Wait for event

If ENQ or DC2 is received

**transition to Acknowledge Line**

If **Send** button pressed

**transition to confirm line**

**Read Thread – Forever loop**

### **Acknowledge Line**

*Note:* **senderHasPriority** is stored as a global

If DC2 received

set **senderHasPriority** to true

if ENQ is received

set **senderHasPriority** to false

if **weHavePriority** is true

transmit DC1

if **weHavePriority** is false

transmit ACK1

set **reading** to true

**transition** to **Wait For Data** state

### **Wait for data**

Set communication mask to EV\_RXCHAR

Initialize a **total** variable to keep track characters successfully read

Wait for the event to occur on the serial port

If timeout is greater than defined timeout constant

Break

While **total** < **516**

Read from serial port

append character read into a buffer

If total is >= 516 or **EOT** detected

Return the buffer that stores all the characters read

**Transition** to **Packet Check** state

### **Packet Check**

*Note: Previous sync byte* is stored as a global

Check the **previous sync byte**

if sync byte is different

perform **checksum** and if **checksum** passes

loop through the packet from 4 to end (message portion)

append each character onto a global buffer

if an **EOT** has been found

print out the global buffer

break

**transition** to **Acknowledge Packet,** passing in the

packet as a parameter

else sync byte is the same

drop packet

**Transition** to **Wait for Data**

### **Acknowledge Packet**

Receives packet from **Packet Check**

if **weHavePriority** is true

transmit DC1

if **weHavePriority** is false

transmit ACK1

loop through the packet to check for **EOT**

if **EOT** found

set **reading** to false (specifies File Transfer Complete)

transition to **Receiver Side: Check Priority State**

### **Receiver side: Check Priority State**

If **senderHasPriority** is true AND **weHavePriority** is false

**transition** to **wait** state

else

**transition** to **idle** state

# **Send Thread – Ends when file transfer complete**

### **Confirm Line**

Note: close the read thread, make sure only 1 thread is operating on the serial at one time

if **number of tries >= 5**

**transition** to **Wait**

if **weHavePriority** is true

transmit **DC2**

if **weHavePriority** is false

transmit **ENQ**

**transition** to **Wait for ACK**

### **Wait for ACK**

*Note:* ***Number of tries*** *is a counter*

Set communication event for EV\_RXCHAR

Wait for Event

Set **timeout** to 5ms

if **timeout** > 5ms

increment **number of tries**

**transition** to **Confirm Line**

else read from serial port

if **get ACK** or **DC1**

**transition** to **Send Data**

### **Send Data**

Lock the thread using mutex

Send the packet through serial port

If the send failed

If overlapped object is busy

Wait for the operation to complete

Increment send packets

Release the thread

**transition** to **Wait for Packet ACK**

### **Wait for Packet ACK**

*Note: timeouts are globally set*

set listener for ACK and DC1

Wait for Event

if **timeout** (elapsed time since packet sent)> 5ms

increment **number of tries**

**transition** to **Send Data**

if **get ACK** or **DC1** (packet transfer was successful)

flip header sync bit (1->0, or 0->1)

**transition** to **Sender side: Check Priority State**

### **Sender side: Check Priority State**

if **weHavePriority** is **true** and **theyHavePriority** is **false**

**transition** to **Idle** state

else

**transition** to **Wait** state

### **GUI Implementation**

**Open button:**

This button opens a file opener dialog which allows the user to select a file from their computer by easily browsing through the filesystem. (Upon triggering this button within WndProc, a new thread will be created, constructing the window.)

**Send Button:**

This button will send the loaded information on the left pane of the window (our loaded file), to receiving end.

**Save Button:**

This button will save the text on the right to a new file. This file will be specified by user input through a file opener dialog.

**Priority Hierarchy Checkbox:**

This checkbox will tell the receiver that it is requesting a higher priority over the transmission than the receiver. If they are both requesting higher priority, they will share the line.

**Input text area:**

The input text area will display a blank area, which may be loaded with an existing file via the open button. The user also has the option to click on the field and edit it themselves.

**Output text area:**

This field will allow the user to see a received file from their sender. This field will be read-only, and inherently blank, until a user selected a successfully downloaded file from the list of received files.

**Received File ComboBox:**

This ComboBox will be a list of all received files. When a user clicks on the ComboBox, it will display the loaded file into the output text area window.

**Information about the transmissions will also be displayed as:**

* Sending progress bar

The sending progress bar will display the progress on how much of the current file has been sent, and how much left there is to send.

* Statistics area

The statistics area will include several text fields which display information about the transmissions. The information will include:

* + - 1. Sent packets
      2. Lost packets
      3. Received Packets
      4. Corrupted Packets Received
      5. Amount of ACK's sent and received
      6. Amount of NAK's sent and received
      7. Bit Error Rate