North American Railroad Signal Aspects and Indications Demonstration Graphical User Interface (GUI) Code

30 August 2018

Kyle Dick, Alex Christmas

2,610 Lines of Java Code

```java
/**
 * Communicates with the Signal Controller on the Arduino DUE
 *
 * @author Kyle Dick
 * @copyright 2018
 *
 * To import the serial communicator into Eclipse:
 *
 * 1. Right click on project file
 * 2. Go to Build Path -> Configure Build Paths
 * 3. Click on Add External JARs
 * 4. Find the jSerialComm-1.3.11.jar file and add it
 * 5. Click Apply and Close
 *
 * Only do the above steps if the import com.fazecast.jSerialComm.* line shows a not found warning
 *
 * NOTE: Anything that is printed to the Java Console is not visible to the end user
 */

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSlider;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import com.fazecast.jSerialComm.*;

public class Controller extends JFrame {
	private static final long serialVersionUID = -903844250172327266L;	// used to suppress warning
	JFrame window = new JFrame();						// frame for GUI window
	String[] rulebook = {							// list of the rulebooks for the drop down menu
			"CSX System Standard / Seaboard Color-Light",	// CSX-SBD
			"CSX Chessie System / C&O Color-Light",		// CSX-CS
			"CSX Conrail Color-Light",			// CSX-CR
```

```
             "CN Route Color-Light",                         // CN-R
             "CN Speed Color-Light",                         // CN-S
             "CN Bessemer & Lake Erie Color-Light",          // CN-BLE
             "Lake Superior and Ishpeming Color-Light",      // LS&I
             "BNSF Railway Color-Light",                     // BNSF
             "Amtrak Michigan Line Color-Light",             // AMTK-AML
             "NS Conrail Territory Color-Light",             // NS-CR
             "NS N&W Territory Color-Light",                 // NS-NW
             "NS Southern Territory Color-Light",            // NS-SOU
             "New York Central Color-Light",                 // NYC
             "Union Pacific Color-Light",                    // UPRR
             "Canadian Railroad Operating Rules Color-Light", // CROR
             "NORAC Color-Light",                            // NORAC
             "ROTATE: Cycle through all indications every XX seconds",
             "RANDOM: Cycle through indications at random"
};

JList<String> rules = new JList<String>(rulebook);                      // creates the list for the rulebooks
JList<String> aspectList = null;                                                           // creates the list for the aspects

JScrollPane aspectSP = null;                                                                     // turns the aspect list into a scrollable pane

JSlider duration = new JSlider(JSlider.HORIZONTAL, 0, 99, 10);                 // creates a slider for the duration selector

JButton cycleKill = new JButton("Click to terminate cycle");                   // creates a button to terminate the rotate cycle
JButton timeOK = new JButton("OK");                                                              // creates a button to confirm the duration

boolean rotate = false;                                                 // tells the program if the user would like to rotate through indications
boolean portOpen = false;                                               // used to tell the program if the port has been successfully opened
boolean aspectVisible = false;                                          // used to tell the program if the aspect list is visible to the end user
boolean durVisible = false;                                             // used to tell the program if the duration text field is visible to the end user
boolean debugGUI = false;                                               // used to tell the program that the programmer is only interested in debugging the GUI
boolean loading = false;

String indicationNum = "";                                              // used to store the indication number if the program is rotating through indications
String aspectSelected = "";

InputStream in = null;                                                  // used to receive serial input from the Arduino
OutputStream out = null;                                                // used to send serial data to the Arduino

JPanel header = new JPanel();                                           // Used to setup easy locations in the window to add components
JPanel west = new JPanel();
JPanel center = new JPanel();
JPanel east = new JPanel();
JPanel footer = new JPanel();

JLabel rulHdr = new JLabel("Rulebook");                                                    // used to display text for the rulebook header
JLabel aspHdr = new JLabel("Please select rule");                                    // used to display text for the aspect header
JLabel desHdr = new JLabel("Description");                                                 // used to display text for the description header
JLabel footerTxt = new JLabel("Please select options");                              // used to display text in the footer of the program

JLabel desc01 = new JLabel("");                                                                  // used to display the description of the aspect
JLabel desc02 = new JLabel("");
JLabel desc03 = new JLabel("");
JLabel desc04 = new JLabel("");
JLabel desc05 = new JLabel("");
JLabel desc06 = new JLabel("Please select aspect");
JLabel desc07 = new JLabel("");
JLabel desc08 = new JLabel("");
JLabel desc09 = new JLabel("");
JLabel desc10 = new JLabel("");
JLabel desc11 = new JLabel("");
JLabel desc12 = new JLabel("");

int i = 0;

/**
 * Looks for the serial port the Arduino is connected to
 *
 * @param comPortIn - the variable that will store the serial port
 * @return comPortIn - the serial port the Arduino is connected to
 */
SerialPort findP(SerialPort comPortIn) {
      // tries to find the port that the Arduino is on
      SerialPort[] coms = SerialPort.getCommPorts();
      for (int i = 0; i < coms.length; i++) {
            String portName = coms[i].getDescriptivePortName();
            if (portName.length() > 27 && portName.substring(0, 28).equals("Arduino Due Programming Port")) {
                  comPortIn = coms[i];
            }
      }
      // finds the port the Arduino is connected to
      // if successful, the program will be told that the port is found
      if (comPortIn == null) {               // if the port fails to be found, print out a message stating so
            System.out.println("Port has failed to open");    // print message to the Java Console
            JFrame frame = new JFrame();                      // creates a frame for the dialog box
            Object[] options = {"Retry", "Cancel"};           // creates the buttons for the dialog box
            int n = JOptionPane.showOptionDialog(frame,       // creates the dialog box itself and sets the result to an integer
                        "Port failed to open. Check to make sure that the Arduino is plugged in and nothing else is using it.",
                        "Port Error",
                        JOptionPane.YES_NO_OPTION,
                        JOptionPane.WARNING_MESSAGE,
                        null,
                        options,
                        options[1]);
            if (n == JOptionPane.YES_OPTION) {                // if the "Retry" button was clicked on
                  coms = SerialPort.getCommPorts();
                  for (int i = 0; i < coms.length; i++) {
                        if (coms[i].getDescriptivePortName().substring(0,  28).equals("Arduino Due Programming Port")) {
                              comPortIn = coms[i];
                        }
                  }
                  if (comPortIn == null) {
                        System.out.println("Port is still failing to open");
                  } else {
                        portOpen = true;
                  }
            } else if (n == JOptionPane.NO_OPTION || n == JOptionPane.CLOSED_OPTION) { // if the "Cancel" button or the red 'X' was clicked on
                  JFrame frame2 = new JFrame();
                  Object[] options2 = {"Yes", "No"};                          // sets up a dialog box asking the programmer if they're debugging the GUI
```

```
				int o = JOptionPane.showOptionDialog(frame2,
						"Test GUI?",
						"Debug GUI?",
						JOptionPane.YES_NO_OPTION,
						JOptionPane.QUESTION_MESSAGE,
						null,
						options2,
						options2[1]);
				if (o == JOptionPane.YES_OPTION) {				// if so, enable GUI debugging
					debugGUI = true;
					System.out.println("GUI debug mode enabled. No data will be sent.");
				} else if (o == JOptionPane.NO_OPTION) {		// if not, terminate the program
					System.out.println("User exited program due to Port Error.");
					System.exit(0);
				}
			}
		} else if (comPortIn != null) {
			portOpen = true;
		}

		while (true) {							// This block will hang the program in order to re-attempt to find the port the Arduino is connected to
			if (portOpen || debugGUI) {				// If the port is successfully opened from the first/second attempt, or if debug mode is enabled:
				break;							// Get out of this while loop
			}
			while (!portOpen) {						// While the port fails to be found:
				JFrame frame = new JFrame();				// Keep showing the dialog box until either the port is found, or the cancel/red 'X' button is pressed
				Object[] options = {"Retry", "Cancel"};
				int n = JOptionPane.showOptionDialog(frame,
						"Port failed to open. Check to make sure that the Arduino is plugged in and nothing else is using it.",
						"Port Error",
						JOptionPane.YES_NO_OPTION,
						JOptionPane.WARNING_MESSAGE,
						null,
						options,
						options[1]);
				if (n == JOptionPane.YES_OPTION) {
					coms = SerialPort.getCommPorts();
					for (int i = 0; i < coms.length; i++) {
						if (coms[i].getDescriptivePortName().substring(0,  28).equals("Arduino Due Programming Port")) {
							comPortIn = coms[i];
						}
					}
					if (comPortIn == null) {
						System.out.println("Port is still failing to open");
					} else {
						portOpen = true;
					}
				} else if (n == JOptionPane.NO_OPTION || n == JOptionPane.CLOSED_OPTION) {
					System.out.println("User exited program due to Port Error");
					System.exit(0);
				}
				System.out.flush();			// Flushes the serial input to keep the communication up with the Arduino so that the program doesn't hang prematurely
			}
			System.out.flush();
			break;						// Exits this block if the port is found
```

```
        }
        return comPortIn;                              // returns the port that the Arduino is connected to
}

/**
 * Finds the header from the serial input
 *
 * @param comPortIn - the serial port the Arduino is connected to
 */
void findHeader(SerialPort comPortIn) {
        if (!debugGUI) {
                comPortIn.openPort();                                                      // opens the port to communication
                comPortIn.setComPortTimeouts(SerialPort.TIMEOUT_READ_SEMI_BLOCKING, 0, 0); // avoids glitch in Windows
                in = comPortIn.getInputStream();                                       // initializes the input stream from the Arduino
                out = comPortIn.getOutputStream();                                     // initializes the output stream to the Arduino
                char header[] = new char[16];                                             // used to store the header of the program to verify the starting point
                char clipped[] = new char[15];                                            // used to compare the header

                /**
                 * This section will advance the buffer to the proper starting point of the program by looking for the "SignalController" text
                 * The Arduino sometimes lags to where the text input seems scrambled sometimes. This will flush the input until it's caught up
                 */
                for (int j = 0; j < 16; j++) {                              // reads in the first 16 characters sent in from the Arduino
                        try {
                                char a = (char) in.read();
                                header[j] = a;
                        } catch (NullPointerException e) {               // if another instance of this program is running, or if another program is using the serial port, display error message
                                JFrame frame = new JFrame();
                                JOptionPane.showMessageDialog(frame, "Error: NullPointerException. This can be caused by multiple instances running\n"
                                                + "or another program using the Arduino. Make sure all instances of the SignalController are closed\n"
                                                + "and ensure that no other programs are using the Arduino, then try again.", "Serial Error", JOptionPane.ERROR_MESSAGE);
                                System.out.println("Serial Error: Duplicate Instance. The program has been terminated.");
                                System.exit(0);
                        } catch (IOException ee) { }
                }
                while (!(new String(header)).equals("SignalController")) {  // while the program hasn't found "SignalController" (if the Arduino hasn't sent "SignalController" yet)
                        char a = 0;
                        try { a = (char) in.read(); } catch (IOException e) { } // read in the next character from the Arduino
                        for (int i = 0; i < 15; i++) {                          // delete the first character from the header array and shift the rest over
                                clipped[i] = header[i + 1];
                                header[i] = clipped[i];
                        }
                        header[15] = a;                                         // add the next character to the end of the header
                }
                System.out.print(header);                                  // print out the header once "SignalController" has been found
        }
}

/**
 * Initializes the GUI window
 */
void initGUI() {
        window.setTitle("Signal Controller");                          // adds a title to the GUI window
        window.setLayout(new BorderLayout());                          // allows objects to be added in by specifying general direction
        window.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE); // tells the window not to close out when the 'X' is clicked on, allows the confirmation box to work properly
window.addWindowListener(new confirmExit());                 // adds the 'Confirm exit on 'X' click' function to the GUI window

rules.addMouseListener(new ruleMouse());                     // adds the function to tell the program that the rule has been selected
footer.setLayout(new FlowLayout());                                              // allows objects to be added in from left to right
window.add(west, BorderLayout.WEST);                         // adds in blank spots to the GUI to allow for easier layout management
window.add(center, BorderLayout.CENTER);
window.add(east, BorderLayout.EAST);
window.add(footer, BorderLayout.SOUTH);
west.setLayout(new BoxLayout(west, BoxLayout.Y_AXIS));       // allows objects to be added in via a stack method
center.setLayout(new BoxLayout(center, BoxLayout.Y_AXIS));
east.setLayout(new BoxLayout(east, BoxLayout.Y_AXIS));

duration.setMajorTickSpacing(10);                                                // adds tick marks to the duration slider
duration.setMinorTickSpacing(5);
duration.setPaintTicks(true);                                                    // makes the tick marks visible
duration.setPaintLabels(true);                                                   // puts labels on the major tick marks
duration.addChangeListener(new durSlider());                      // adds a change listener to tell the program what the slider is set to

timeOK.addActionListener(new durClick());                               // adds the Action Listener to the button to tell the program when its been clicked

window.setResizable(false);                                                                    // prevents the user from resizing the window

west.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));   // sets a border around each panel that will hold the lists for the rules aspects, and indication description
center.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));
east.setBorder(BorderFactory.createEmptyBorder(0, 10, 10, 10));

west.add(Box.createVerticalStrut(10));                            // puts in space between the window border and the header
west.add(rulHdr);                                                                              // adds in the rulebook header
west.add(Box.createVerticalStrut(10));                                            // puts in space between the rulebook header and the scroll list border
west.add(new JScrollPane(rules));                                        // adds the list of rulebooks to the GUI in as a scrollable list

west.setPreferredSize(new Dimension(356, 246));                                   // sets the sizes of the panels
center.setPreferredSize(new Dimension(356, 246));
east.setPreferredSize(new Dimension(356, 246));

center.add(Box.createVerticalStrut(10));
center.add(aspHdr);
center.add(Box.createVerticalStrut(10));
east.add(Box.createVerticalStrut(10));
east.add(desHdr);
east.add(Box.createVerticalStrut(10));
east.add(Box.createVerticalGlue());
east.add(desc01);
east.add(desc02);
east.add(desc03);
east.add(desc04);
east.add(desc05);
east.add(desc06);
east.add(desc07);
east.add(desc08);
east.add(desc09);
east.add(desc10);
east.add(desc11);
east.add(desc12);
east.add(Box.createVerticalGlue());
```

```
        aspHdr.setAlignmentX(JLabel.CENTER_ALIGNMENT);   // sets the headers to center alignment
        rulHdr.setAlignmentX(JLabel.CENTER_ALIGNMENT);
        footer.add(footerTxt);                           // adds the status text to the bottom of the GUI
        window.pack();                                   // packs in the elements to the window
        window.setLocationRelativeTo(null);              // centers the window to the desktop screen
        window.setVisible(true);                         // makes the window visible to the end user
}

/**
 * Sends data to the Arduino
 *
 * @param send - String to send to the Arduino
 * @throws IOException if an I/O error occurs
 */
void sendArduino(String send) throws IOException {
        if (!debugGUI) {                        // if debug GUI mode isn't enabled:
                System.out.println(send);        // print out what is being sent to the Arduino on the Java Console
                byte inByte[] = send.getBytes(); // translate the String being sent into bytes
                out.write(inByte);               // send out the String
        }
}

/**
 * Reads in serial input from Arduino
 *
 * @throws IOException if an I/O error occurs
 */
char readArduino() throws IOException {
        if (loading) {
                footerTxt.setText("Loading, please wait");
                window.pack();
        }
//              char[] fiveIn = new char[5];
//              int i = 0;
        char oneIn = 0;                          // stores one character at a time from the serial input
        while ((int) oneIn != 6) {               // prints the serial output until it hits the ASCII Acknowledge code (used to denote end of output)
                oneIn = (char) in.read();         // reads in a character from the Arduino
//                      fiveIn[i++ % 5] = oneIn;
//                      System.out.println(" " + new String(fiveIn));
//                      if (new String(fiveIn).equals("ERROR")) {
//                              System.out.println("*NE*");
//                      }
                if ((int) oneIn != 6) {           // if the read in character doesn't equal the terminate character:
                        return oneIn;               // print out one character at a time received from the Arduino
                }
                if (rotate && (int) oneIn != 6) { // if the program is rotating through indications:
                        indicationNum += oneIn;    // store the number
                }
        }
        if (oneIn == 6 && loading) {
                footerTxt.setText("Currently displaying: " + aspectSelected);
                aspectList.setEnabled(true);
                window.pack();
                loading = false;
        }
        return 0;
    }

    public static void main(String[] args) throws IOException, InterruptedException {
        Controller ctrl = new Controller();                                    // creates instance of this program
        SerialPort comPort = null;                                             // used to retrieve the serial port the Arduino is hooked up to
        comPort = ctrl.findP(comPort);                                         // finds the serial port the Arduino is connected to
        ctrl.findHeader(comPort);                                                                  // finds the "SignalController" text from the serial input
        ctrl.initGUI();                                                        // initializes the GUI

        if (!ctrl.debugGUI) {
            /**
             * This block will print out the serial input as it comes through
             */
            String out = "";
            while (true) {                                                      // program will hang here, reading in input from the Arduino
                char arduOut = ctrl.readArduino();
                System.out.print(arduOut);                                 // reads and prints input from the Arduino
                out += arduOut;
                if (arduOut == '\n') {
                    out = "";
                }
                if (out.equals("Enter a rule number: ")) {
                    ctrl.durVisible = false;
                }
                if (ctrl.rotate) {                                                  // if the program is rotating through indications:
                    ctrl.footerTxt.setText("Displaying " + ctrl.indicationNum); // display indication number
                    ctrl.window.repaint();                                        // refresh window
                    ctrl.indicationNum = "";                                      // reset indication number string
                }
            }
        }
    }

    /**
     *  Window Listener for exit confirm dialogue box
     */
    class confirmExit implements WindowListener {

        @Override
        public void windowClosing(WindowEvent e) {
            int confirmed = JOptionPane.showConfirmDialog(null,
                    "Are you sure you want to exit the program?", "Exit Program",
                    JOptionPane.YES_NO_OPTION);

            if (confirmed == JOptionPane.YES_OPTION) {
                window.dispose();
                System.out.println("\nProgram 'SignalController' has been terminated");
                System.exit(0);
            }
        }

        @Override
        public void windowActivated(WindowEvent e) { }
        @Override
        public void windowClosed(WindowEvent e) { }

        @Override
        public void windowDeactivated(WindowEvent e) { }

        @Override
        public void windowDeiconified(WindowEvent e) { }

        @Override
        public void windowIconified(WindowEvent e) { }

        @Override
        public void windowOpened(WindowEvent e) { }
}

/**
 * Action listener for the cycle kill button
 */
class cycleKillClick implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
                try {
                        sendArduino("KILL");
                        footer.remove(cycleKill);
                        center.remove(duration);
                        center.remove(timeOK);
                        footerTxt.setText("Please select options");
                        aspHdr.setText("Please select rule");
                        desc06.setText("Please select aspect");
                        window.repaint();
                        rotate = false;

                } catch (IOException e1) {

                }
        }

}

/**
 * Mouse Listener for the rule list
 */
class ruleMouse implements MouseListener {

        @Override
        public void mouseClicked(MouseEvent arg0) {
                if (arg0.getClickCount() == 2) {
                        if (aspectVisible) {
                                center.remove(aspectSP);
                                window.repaint();
                                aspectVisible = false;
                        }
```

```
if (durVisible) {
      try { sendArduino("CNL"); } catch (IOException e) { }
      center.remove(duration);
      center.remove(timeOK);
      window.repaint();
      durVisible = false;
}
aspHdr.setText("Aspect");
String ruleSelected = (String) rules.getSelectedValue();
if (ruleSelected.equals(rulebook[0])) { // CSX-SBD
      String[] aspectSet = {
                  "CSX 1281  - Clear",
                  "CSX 1281B - Approach Limited",
                  "CSX 1281C - Limited Clear",
                  "CSX 1281D - Limited Approach",
                  "CSX 1282  - Approach Medium",
                  "CSX 1282A - Advance Approach",
                  "CSX 1283  - Medium Clear",
                  "CSX 1283A - Medium Approach Medium",
                  "CSX 1283B - Medium Approach Slow",
                  "CSX 1283C - Medium Advance Approach",
                  "CSX 1284  - Approach Slow",
                  "CSX 1285  - Approach",
                  "CSX 1286  - Medium Approach",
                  "CSX 1287  - Slow Clear",
                  "CSX 1288  - Slow Approach",
                  "CSX 1290  - Restricting",
                  "CSX 1291  - Restricted Proceed",
                  "CSX 1292  - Stop",
                  "CSX 1293  - Stop and Check",
                  "CSX 1294  - Stop and Open Switch",
                  "CSX 1298  - Grade",};
      aspectList = new JList<String> (aspectSet);

} else if (ruleSelected.equals(rulebook[1])) { // CSX-CS
      String[] aspectSet = {
                  "CSX C1281  - Clear",
                  "CSX C1281B - Approach Limited",
                  "CSX C1281C - Limited Clear",
                  "CSX C1281D - Limited Approach",
                  "CSX C1282  - Approach Medium",
                  "CSX C1283  - Medium Clear",
                  "CSX C1283A - Medium Approach Medium",
                  "CSX C1283B - Medium Approach Slow",
                  "CSX C1284  - Approach Slow",
                  "CSX C1285  - Approach",
                  "CSX C1286  - Medium Approach",
                  "CSX C1287  - Slow Clear",
                  "CSX C1288  - Slow Approach",
                  "CSX C1290  - Restricting",
                  "CSX C1291  - Restricted Proceed",
                  "CSX C1292  - Stop",
                  "CSX C1298  - Grade",};
      aspectList = new JList<String> (aspectSet);
```

```
} else if (ruleSelected.equals(rulebook[2])) { // CSX-CR
      String[] aspectSet = {
                   "CSX CR1281  - Clear",
                   "CSX CR1281A - Cab Speed",
                   "CSX CR1281B - Approach Limited",
                   "CSX CR1281C - Limited Clear",
                   "CSX CR1282  - Approach Medium",
                   "CSX CR1282A - Advanced Approach",
                   "CSX CR1283  - Medium Clear",
                   "CSX CR1283A - Medium Approach Medium",
                   "CSX CR1284  - Approach Slow",
                   "CSX CR1285  - Approach",
                   "CSX CR1286  - Medium Approach",
                   "CSX CR1287  - Slow Clear",
                   "CSX CR1288  - Slow Approach",
                   "CSX CR1290  - Restricting",
                   "CSX CR1291  - Restricted Proceed",
                   "CSX CR1292  - Stop",};
      aspectList = new JList<String> (aspectSet);

} else if (ruleSelected.equals(rulebook[3])) { // CN-R
      String[] aspectSet = {
                   "CN 803 - Clear",
                   "CN 804 - Advance Approach",
                   "CN 805 - Approach",
                   "CN 806 - Approach Restricting",
                   "CN 807 - Advance Approach Diverging",
                   "CN 808 - Approach Diverging",
                   "CN 809 - Diverging Clear",
                   "CN 810 - Diverging Clear Approach Diverging",
                   "CN 811 - Diverging Advance Approach",
                   "CN 812 - Diverging Approach",
                   "CN 813 - Diverging Approach Restricting",
                   "CN 814 - Restricting",
                   "CN 815 - Restricted Proceed",
      "CN 816 - Stop"};
      aspectList = new JList<String> (aspectSet);

} else if (ruleSelected.equals(rulebook[4])) { // CN-S
      String[] aspectSet = {
                   "CN 817   - Clear",
                   "CN 818   - Advance Approach",
                   "CN 818.1 - Medium Advance Approach",
                   "CN 818.2 - Limited Advance Approach",
                   "CN 819   - Approach",
                   "CN 820   - Approach Limited",
                   "CN 821   - Approach Medium",
                   "CN 822   - Approach Slow",
                   "CN 823   - Limited Clear",
                   "CN 824   - Limited Clear Limited",
                   "CN 825   - Limited Approach",
                   "CN 826   - Medium Clear",
                   "CN 827   - Medium Clear Medium",
                   "CN 828   - Medium Approach",
                   "CN 829   - Slow Clear",
                   "CN 830   - Slow Approach",
                   "CN 831   - Restricting",
      "CN 832   - Stop"};
      aspectList = new JList<String> (aspectSet);

  } else if (ruleSelected.equals(rulebook[5])) { // CN-BLE
      String[] aspectSet = {
                   "CN 833 - Clear",
                   "CN 834 - Approach Limited",
                   "CN 835 - Limited Clear",
                   "CN 836 - Approach Medium",
                   "CN 837 - Medium Clear",
                   "CN 838 - Medium Approach Medium",
                   "CN 839 - Approach",
                   "CN 840 - Medium Approach",
                   "CN 841 - Slow Clear",
                   "CN 842 - Slow Approach",
                   "CN 843 - Restricting",
      "CN 844 - Stop"};
      aspectList = new JList<String> (aspectSet);

  } else if (ruleSelected.equals(rulebook[6])) { // LS&I
      String[] aspectSet = {
                   "LS&I 230 - Clear",
                   "LS&I 231 - Approach",
                   "LS&I 232 - Diverging Clear",
                   "LS&I 233 - Diverging Approach",
                   "LS&I 234 - Restricting",
      "LS&I 235 - Stop"};
      aspectList = new JList<String> (aspectSet);

  } else if (ruleSelected.equals(rulebook[7])) { // BNSF
      String[] aspectSet = {
                   "BNSF 9.1.3  - Clear",
                   "BNSF 9.1.4  - Approach Limited",
                   "BNSF 9.1.5  - Advance Approach",
                   "BNSF 9.1.6  - Approach Medium",
                   "BNSF 9.1.7  - Approach Restricting",
                   "BNSF 9.1.8  - Approach",
                   "BNSF 9.1.9  - Diverging Clear",
                   "BNSF 9.1.10 - Diverging Approach Diverging",
                   "BNSF 9.1.11 - Diverging Approach Medium",
                   "BNSF 9.1.12 - Diverging Approach",
                   "BNSF 9.1.13 - Restricting",
                   "BNSF 9.1.14 - Restricting",
      "BNSF 9.1.15 - Stop"};
      aspectList = new JList<String> (aspectSet);

  } else if (ruleSelected.equals(rulebook[8])) { // AMTK-AML
      String[] aspectSet = {
                   "AMTK 9.1.1  - Clear",
                   "AMTK 9.1.2  - Advance Approach",
                   "AMTK 9.1.3  - Approach Slow",
                   "AMTK 9.1.4  - Approach",
                   "AMTK 9.1.5  - Slow Clear",
                   "AMTK 9.1.6  - Slow Approach",
                   "AMTK 9.1.7  - Restricting",
                   "AMTK 9.1.8  - Stop and Proceed",
                   "AMTK 9.1.9  - Stop",
                   "AMTK 9.1.10 - Approach Medium",
                   "AMTK 9.1.11 - Approach Limited",
                   "AMTK 9.1.12 - Medium Clear",
                   "AMTK 9.1.13 - Limited Clear",
       "AMTK 9.1.14 - Medium Approach"};
       aspectList = new JList<String> (aspectSet);

} else if (ruleSelected.equals(rulebook[9])) { // NS-CR
       String[] aspectSet = {
                   "NS 306 - Clear",
                   "NS 307 - Approach Limited",
                   "NS 308 - Limited Clear",
                   "NS 309 - Approach Medium",
                   "NS 310 - Advance Approach",
                   "NS 311 - Medium Clear",
                   "NS 312 - Medium Approach Medium",
                   "NS 313 - Approach Slow",
                   "NS 314 - Approach",
                   "NS 315 - Medium Approach",
                   "NS 316 - Slow Clear",
                   "NS 317 - Slow Approach",
                   "NS 318 - Restricting",
       "NS 319 - Stop"};
       aspectList = new JList<String> (aspectSet);

} else if (ruleSelected.equals(rulebook[10])) { // NS-NW
       String[] aspectSet = {
                   "NS 326 - Clear",
                   "NS 327 - Approach Diverging",
                   "NS 328 - Advance Approach",
                   "NS 329 - Diverging Clear",
                   "NS 330 - Diverging Approach Diverging",
                   "NS 331 - Approach",
                   "NS 333 - Diverging Approach",
                   "NS 334 - Slow Clear",
                   "NS 335 - Slow Approach",
                   "NS 336 - Restricting",
       "NS 337 - Stop"};
       aspectList = new JList<String> (aspectSet);

} else if (ruleSelected.equals(rulebook[11])) { // NS-SOU
       String[] aspectSet = {
                   "NS 340 - Clear",
                   "NS 341 - Approach Diverging",
                   "NS 342 - Advance Approach",
                   "NS 343 - Diverging Clear",
                   "NS 344 - Approach Restricted",
                   "NS 345 - Diverging Approach Restricted",
                   "NS 346 - Approach",
                   "NS 347 - Diverging Approach",
                   "NS 348 - Restricting",
      "NS 349 - Stop"};
      aspectList = new JList<String> (aspectSet);

} else if (ruleSelected.equals(rulebook[12])) { // NYC
      String[] aspectSet = {
                   "NYC 281  - Clear",
                   "NYC 281A - Advance Approach Medium",
                   "NYC 281B - Approach Limited",
                   "NYC 281C - Limited Clear",
                   "NYC 281D - Limited Approach",
                   "NYC 282  - Approach Medium",
                   "NYC 282A - Advance Approach",
                   "NYC 283  - Medium Clear",
                   "NYC 283A - Medium Advance Approach",
                   "NYC 283B - Medium Approach Slow",
                   "NYC 284  - Approach Slow",
                   "NYC 285  - Approach",
                   "NYC 286  - Medium Approach",
                   "NYC 287  - Slow Clear",
                   "NYC 288  - Slow Approach",
                   "NYC 290  - Restricting",
                   "NYC 291  - Stop and Proceed",
      "NYC 292  - Stop"};
      aspectList = new JList<String> (aspectSet);

} else if (ruleSelected.equals(rulebook[13])) { // UPRR
      String[] aspectSet = {
                   "UP 9.2.1  - Clear",
                   "UP 9.2.2  - Approach Clear Sixty",
                   "UP 9.2.3  - Approach Clear Fifty",
                   "UP 9.2.4  - Advance Approach",
                   "UP 9.2.5  - Approach Diverging",
                   "UP 9.2.6  - Approach",
                   "UP 9.2.7  - Approach Restricting",
                   "UP 9.2.8  - Diverging Clear Limited",
                   "UP 9.2.9  - Diverging Clear",
                   "UP 9.2.10 - Diverging Advance Approach",
                   "UP 9.2.11 - Diverging Approach",
                   "UP 9.2.12 - Diverging Approach Diverging",
                   "UP 9.2.13 - Restricting",
                   "UP 9.2.14 - Restricted Proceed",
                   "UP 9.2.15 - Stop",
      "UP 9.2.16 - Diverging Approach Clear Fifty"};
      aspectList = new JList<String> (aspectSet);

} else if (ruleSelected.equals(rulebook[14])) { // CROR
      String[] aspectSet = {
                   "CROR 405 - Clear",
                   "CROR 406 - Clear to Limited",
                   "CROR 407 - Clear to Medium",
                   "CROR 408 - Clear to Diverging",
                   "CROR 409 - Clear to Slow",
                   "CROR 410 - Clear to Restricting",
                   "CROR 411 - Clear to Stop",
                   "CROR 412 - Advance Clear to Limited",
                    "CROR 413 - Advance Clear to Medium",
                    "CROR 414 - Advance Clear to Slow",
                    "CROR 415 - Advance Clear to Stop",
                    "CROR 416 - Limited to Clear",
                    "CROR 417 - Limited to Limited",
                    "CROR 418 - Limited to Medium",
                    "CROR 419 - Limited to Slow",
                    "CROR 420 - Limited to Restricting",
                    "CROR 421 - Limited to Stop",
                    "CROR 422 - Medium to Clear",
                    "CROR 423 - Medium to Limited",
                    "CROR 424 - Medium to Medium",
                    "CROR 425 - Medium to Slow",
                    "CROR 426 - Medium to Restricting",
                    "CROR 427 - Medium to Stop",
                    "CROR 428 - Diverging to Clear",
                    "CROR 429 - Diverging to Stop",
                    "CROR 430 - Diverging",
                    "CROR 431 - Slow to Clear",
                    "CROR 432 - Slow to Limited",
                    "CROR 433 - Slow to Medium",
                    "CROR 434 - Slow to Slow",
                    "CROR 435 - Slow to Stop",
                    "CROR 436 - Restricting",
                    "CROR 437 - Stop and Proceed",
                    "CROR 438 - Take/Leave Siding",
      "CROR 439 - Stop"};
      aspectList = new JList<String> (aspectSet);

} else if (ruleSelected.equals(rulebook[15])) { // NORAC
      String[] aspectSet = {
                    "NORAC 281  - Clear",
                    "NORAC 281A - Cab Speed",
                    "NORAC 281B - Approach Limited",
                    "NORAC 281C - Limited Clear",
                    "NORAC 282  - Approach Medium",
                    "NORAC 282A - Advanced Approach",
                    "NORAC 283  - Medium Clear",
                    "NORAC 284  - Approach Slow",
                    "NORAC 285  - Approach",
                    "NORAC 286  - Medium Approach",
                    "NORAC 287  - Slow Clear",
                    "NORAC 288  - Slow Approach",
                    "NORAC 290  - Restricting",
                    "NORAC 291  - Stop and Proceed",
      "NORAC 292  - Stop"};
      aspectList = new JList<String> (aspectSet);

} else if (ruleSelected.equals(rulebook[16]) || ruleSelected.equals(rulebook[17])) { // ROTATE
      try {
             if (ruleSelected.equals(rulebook[16])) { sendArduino("ROTATE"); }
             else if (ruleSelected.equals(rulebook[17])) { sendArduino("RANDOM"); }
             desc01.setText("");
             desc02.setText("");
             desc03.setText("");
```

```java
                    desc04.setText("");
                    desc05.setText("");
                    desc06.setText("Random indications");
                    desc07.setText("");
                    desc08.setText("");
                    desc09.setText("");
                    desc10.setText("");
                    desc11.setText("");
                    desc12.setText("");
                } catch (IOException e) { }
                durVisible = true;
                aspHdr.setText("Duration");
                timeOK.setAlignmentX(JButton.CENTER_ALIGNMENT);
                center.add(duration);
                center.add(timeOK);
                footerTxt.setText("Please enter in a duration");
            }
            if (!ruleSelected.equals(rulebook[16]) && !ruleSelected.equals(rulebook[17])) {
                if (!aspectVisible) {
                    aspectVisible = true;
                    aspHdr.setAlignmentX(JLabel.CENTER_ALIGNMENT);
                    aspectList.addMouseListener(new aspectMouse());
                    aspectSP = new JScrollPane(aspectList);
                    center.add(aspectSP);
                }
            }
            if (loading) {
                aspectList.setEnabled(false);
            }
            window.pack();
        }
    }

    @Override
    public void mouseEntered(MouseEvent arg0) { }

    @Override
    public void mouseExited(MouseEvent arg0) { }

    @Override
    public void mousePressed(MouseEvent arg0) { }

    @Override
    public void mouseReleased(MouseEvent arg0) { }

}

/**
 * Mouse Listener for aspect list
 */
class aspectMouse implements MouseListener {

    public void thruSwitchCSX(String indication, int restrict, String exceed) {
        desc05.setText(indication + " through turnouts,");
        desc06.setText("crossovers, siding, and over power");
        if (restrict == 0) {
              desc07.setText("operated switches; then proceed at");
              desc08.setText("posted speed.");
        } else if (restrict == 1) {
              desc07.setText("operated switches; then proceed,");
              desc08.setText("prepared to stop at the next signal.");
        } else if (restrict == 2) {
              desc06.setText("operated switches; then proceed,");
              desc07.setText("approaching next signal not");
              desc08.setText("exceeding " + exceed + ".");
        } else if (restrict == 3) {
              desc07.setText("operated switches; then proceed,");
              desc08.setText("prepared to stop at the second signal.");
        }
  }

  public void proceedCN(int option, int exced1, int exced2) {
        if (option == 1 || option == 2 || option == 3) {
              desc05.setText("Proceed not exceeding " + exced1 + " MPH");
              desc06.setText("through turnouts, then proceed");
        }

        if (option == 1) { desc07.setText("prepared to stop at second signal."); }
        else if (option == 2) {
              desc07.setText("approaching next signal not");
              desc08.setText("exceeding " + exced2 + " MPH.");
        }
        else if (option == 3) { desc07.setText("prepared to stop at next signal."); }
        else if (option == 4) {
              desc05.setText("Proceed approaching next signal not");
              desc06.setText("exceeding " + exced1 + " MPH.");
        }
        else if (option == 5) {
              desc05.setText("Proceed, not exceeding " + exced1 + " MPH");
              desc06.setText("through turnouts.");
        }
  }

  public void longDescUP(int option, int exced, int reduce, int exced2, int trnOut) {

        if (option == 1) {
              desc01.setText("Proceed. Freight trains exceeding");
              desc02.setText(exced + " MPH must immediately reduce to");
              desc03.setText(reduce + " MPH. Passenger train may");
              desc04.setText("proceed, but be prepared to pass");
              desc05.setText("the next signal not exceeding " + exced2);
              desc06.setText("MPH. When signal governs the");
              desc07.setText("approach to a control point with a");
              desc08.setText(trnOut + " MPH turnout speed, be prepared to");
              desc09.setText("advance on diverging route.");
        } else if (option == 2 || option == 3) {
              desc01.setText("Proceed, prepared to stop at second");
              desc02.setText("signal. Freight trains exceeding " + exced);
              desc03.setText("MPH must immediately reduce to " + reduce);
```

```
			desc04.setText("MPH. Passenger trains may proceed,");
			desc05.setText("but must be prepared to pass the");
			desc06.setText("next signal not exceeding " + exced2 + " MPH.");
		}
		if (option == 3) {
			desc07.setText("When signal governs the approach to");
			desc08.setText("a control point with a " + trnOut + " MPH");
			desc09.setText("turnout speed, be prepared to");
			desc10.setText("advance on normal or diverging route.");
		}

	}

	public void proceedCROR(int option, String speed) {

		if (option == 1) {
			desc05.setText("Proceed, approaching next signal at");
			desc06.setText(speed);
		} else if (option == 2) {
			desc05.setText("Proceed, approaching second signal");
			desc06.setText("at " + speed);
		}
	}

	public void turnOutCROR(String speed1, String speed2) {

		desc01.setText("Proceed at " + speed1 + " past");
		desc02.setText("signal and through turnouts,");
		desc03.setText("approaching next signal at " + speed2 + ".");

	}

	public void pastSignalCROR(int option, String speed, String speed2) {

		desc05.setText("Proceed at " + speed + " past signal");

		if (option == 1) {
			desc06.setText("and through turnouts, preparing to");
			desc07.setText("stop at the next signal.");
		}

		if (option == 2) {
			desc06.setText("and through turnouts.");
		}

		if (option == 3) {
			desc06.setText("and through turnouts, approaching");
			desc07.setText("next signal at " + speed2);
		}
	}

	@Override
	public void mouseClicked(MouseEvent e) {
		if (e.getClickCount() == 2) {
			footerTxt.setText("Loading, please wait");
```

```
loading = true;
aspectList.setEnabled(false);
desc01.setText("");
desc02.setText("");
desc03.setText("");
desc04.setText("");
desc05.setText("");
desc06.setText("");
desc07.setText("");
desc08.setText("");
desc09.setText("");
desc10.setText("");
desc11.setText("");
desc12.setText("");
try {
      aspectSelected = (String) aspectList.getSelectedValue();
      if (aspectSelected.equals("CSX 1281  - Clear")) {
            sendArduino("CSX 1281");
            desc06.setText("Proceed.");
            desc07.setText("No restrictions.");
      }
      else if (aspectSelected.equals("CSX 1281B - Approach Limited")) {
            sendArduino("CSX 1281B");
            desc06.setText("Proceed, approaching next signal not");
            desc07.setText("exceeding Limited Speed.");
      }
      else if (aspectSelected.equals("CSX 1281C - Limited Clear")) {
            sendArduino("CSX 1281C");
            thruSwitchCSX("Limited Speed", 0, null);
      }
      else if (aspectSelected.equals("CSX 1281D - Limited Approach")) {
            sendArduino("CSX 1281D");
            thruSwitchCSX("Limited Speed", 1, null); // CORRECTION
      }
      else if (aspectSelected.equals("CSX 1282  - Approach Medium")) {
            sendArduino("CSX 1282");
            desc06.setText("Proceed, approaching next signal not");
            desc07.setText("exceeding Medium Speed.");
      }
      else if (aspectSelected.equals("CSX 1282A - Advance Approach")) {
            sendArduino("CSX 1282A");
            desc06.setText("Proceed, prepared to stop at second signal.");
      }
      else if (aspectSelected.equals("CSX 1283  - Medium Clear")) {
            sendArduino("CSX 1283");
            thruSwitchCSX("Medium Speed", 0, null);
      }
      else if (aspectSelected.equals("CSX 1283A - Medium Approach Medium")) {
            sendArduino("CSX 1283A");
            thruSwitchCSX("Medium Speed", 2, "Medium Speed"); //?
      }
      else if (aspectSelected.equals("CSX 1283B - Medium Approach Slow")) {
            sendArduino("CSX 1283B");
            thruSwitchCSX("Medium Speed", 2, "Medium Speed"); //?
      }
```

```java
else if (aspectSelected.equals("CSX 1283C - Medium Advance Approach")) {
	sendArduino("CSX 1283C");
	thruSwitchCSX("Medium Speed", 3, null);
}
else if (aspectSelected.equals("CSX 1284  - Approach Slow")) {
	sendArduino("CSX 1284");
	desc06.setText("Proceed, approaching next signal not");
	desc07.setText("exceeding Slow Speed.");
}
else if (aspectSelected.equals("CSX 1285  - Approach")) {
	sendArduino("CSX 1285");
	desc04.setText("Proceed prepared to stop at the next");
	desc05.setText("signal. Trains exceeding Medium");
	desc06.setText("Speed must begin reduction to Medium");
	desc07.setText("Speed as soon as the engine passes");
	desc08.setText("the signal.");
}
else if (aspectSelected.equals("CSX 1286  - Medium Approach")) {
	sendArduino("CSX 1286");
	thruSwitchCSX("Medium Speed", 1, null);
}
else if (aspectSelected.equals("CSX 1287  - Slow Clear")) {
	sendArduino("CSX 1287");
	thruSwitchCSX("Slow Speed", 0, null);
}
else if (aspectSelected.equals("CSX 1288  - Slow Approach")) {
	sendArduino("CSX 1288");
	thruSwitchCSX("Slow Speed", 1, null);
}
else if (aspectSelected.equals("CSX 1290  - Restricting")) {
	sendArduino("CSX 1290");
	desc06.setText("Proceed at Restricted Speed.");
}
else if (aspectSelected.equals("CSX 1291  - Restricted Proceed")) {
	sendArduino("CSX 1291");
	desc06.setText("With number plate or 'P' plate: Proceed at"); // CORRECTION
	desc07.setText("Restricted Speed.");
}
else if (aspectSelected.equals("CSX 1292  - Stop")) {
	sendArduino("CSX 1292");
	desc06.setText("Without any plates: Stop.");
}
else if (aspectSelected.equals("CSX 1293  - Stop and Check")) {
	sendArduino("CSX 1293");
	desc03.setText("With “C” Plate: Stop and check position");
	desc04.setText("of drawbridge, spring switch,");
	desc05.setText("derails or gates protecting railroad");
	desc06.setText("crossings. If way is clear and");
	desc07.setText("drawbridge, spring switch, derails");
	desc08.setText("or gate are in proper position,");
	desc09.setText("proceed at restricted speed."); // CORRECION
}
else if (aspectSelected.equals("CSX 1294  - Stop and Open Switch")) {
	sendArduino("CSX 1294");
	desc06.setText("With 'S' Marker Illuminated: Stop and");
```

```
        desc07.setText("open hand-operated switch.");
}
else if (aspectSelected.equals("CSX 1298  - Grade")) {
        sendArduino("CSX 1298");
        desc06.setText("With 'G' plate: Proceed at Restricted Speed.");
}


else if (aspectSelected.equals("CSX C1281  - Clear")) {
        sendArduino("CSX C1281");
        desc05.setText("Proceed.");
        desc06.setText(" ");
        desc07.setText("Dwarf signal requires number plate."); // CORRECTION
}
else if (aspectSelected.equals("CSX C1281B - Approach Limited")) {
        sendArduino("CSX C1281B");
        desc06.setText("Proceed, approaching next signal not");
        desc07.setText("exceeding Limited Speed.");
}
else if (aspectSelected.equals("CSX C1281C - Limited Clear")) {
        sendArduino("CSX C1281C");
        thruSwitchCSX("Limited Speed", 0, null);
}
else if (aspectSelected.equals("CSX C1281D - Limited Approach")) {
        sendArduino("CSX C1281D");
        thruSwitchCSX("Limited Speed", 1, null);
}
else if (aspectSelected.equals("CSX C1282  - Approach Medium")) {
        sendArduino("CSX C1282");
        desc06.setText("Proceed, approaching next signal not");
        desc07.setText("exceeding Medium Speed.");
}
else if (aspectSelected.equals("CSX C1283  - Medium Clear")) {
        sendArduino("CSX C1283");
        thruSwitchCSX("Medium Speed", 0, null);
}
else if (aspectSelected.equals("CSX C1283A - Medium Approach Medium")) {
        sendArduino("CSX C1283A");
        thruSwitchCSX("Medium Speed", 2, "Medium Speed");
}
else if (aspectSelected.equals("CSX C1283B - Medium Approach Slow")) {
        sendArduino("CSX C1283B");
        thruSwitchCSX("Medium Speed", 2, "Slow Speed");
}
else if (aspectSelected.equals("CSX C1284  - Approach Slow")) {
        sendArduino("CSX C1284");
        desc06.setText("Proceed, approaching next signal not");
        desc07.setText("exceeding Slow Speed.");
}
else if (aspectSelected.equals("CSX C1285  - Approach")) {
        sendArduino("CSX C1285");
        desc03.setText("Proceed prepared to stop at the next");
        desc04.setText("signal. Trains exceeding Medium");
        desc05.setText("Speed must begin reduction to Medium");
        desc06.setText("Speed as soon as the engine passes");
```

```
		desc07.setText("the signal.");
		desc08.setText(" ");
		desc09.setText("This aspect requires a number plate."); // CORRECTION
	}
	else if (aspectSelected.equals("CSX C1286  - Medium Approach")) {
		sendArduino("CSX C1286");
		thruSwitchCSX("Medium Speed", 1, null);
	}
	else if (aspectSelected.equals("CSX C1287  - Slow Clear")) {
		sendArduino("CSX C1287");
		thruSwitchCSX("Slow Speed", 0, null);
	}
	else if (aspectSelected.equals("CSX C1288  - Slow Approach")) {
		sendArduino("CSX C1288");
		thruSwitchCSX("Slow Speed", 1, null);
	}
	else if (aspectSelected.equals("CSX C1290  - Restricting")) {
		sendArduino("CSX 1290");
		desc06.setText("Proceed at Restricted Speed.");
	}
	else if (aspectSelected.equals("CSX C1291  - Restricted Proceed")) {
		sendArduino("CSX C1291");
		desc06.setText("With number plate or 'P' plate: Proceed at"); // CORRECTION
		desc07.setText("Restricted Speed");
	}
	else if (aspectSelected.equals("CSX C1292  - Stop")) {
		sendArduino("CSX C1292");
		desc06.setText("Without number plate or sign: Stop."); // CORRECTION
	}
	else if (aspectSelected.equals("CSX C1298  - Grade")) {
		sendArduino("CSX C1298");
		desc06.setText("With 'G' plate: Proceed at Restricted");
		desc07.setText("Speed.");
	}


	else if (aspectSelected.equals("CSX CR1281  - Clear")) {
		sendArduino("CSX CR1281");
		desc06.setText("Proceed");
	}
	else if (aspectSelected.equals("CSX CR1281A - Cab Speed")) {
		sendArduino("CSX CR1281A");
		desc04.setText("Proceed in accordance with cab");
		desc05.setText("signal indication. Reduce speed to");
		desc06.setText("not exceeding 60 mph if Cab Speed");
		desc07.setText("cab signal is displayed without a");
		desc08.setText("signal speed, or if cab signals are");
		desc09.setText("not operative."); // CORRECTION
	}
	else if (aspectSelected.equals("CSX CR1281B - Approach Limited")) {
		sendArduino("CSX CR1281B");
		desc06.setText("Proceed, approaching next signal not");
		desc07.setText("exceeding Limited Speed.");
	}
	else if (aspectSelected.equals("CSX CR1281C - Limited Clear")) {
		sendArduino("CSX CR1281C");
		thruSwitchCSX("Limited Speed", 1, null);
	}
	else if (aspectSelected.equals("CSX CR1282  - Approach Medium")) {
		sendArduino("CSX CR1282");
		desc06.setText("Proceed, approaching next signal not");
		desc07.setText("exceeding Medium Speed.");
	}
	else if (aspectSelected.equals("CSX CR1282A - Advanced Approach")) {
		sendArduino("CSX CR1282A");
		desc04.setText("Proceed, prepared to stop at the");
		desc05.setText("second signal. Trains exceeding");
		desc06.setText("Limited Speed must begin reduction");
		desc07.setText("to Limited Speed as soon as");
		desc08.setText("locomotive passes the signal.");
	}
	else if (aspectSelected.equals("CSX CR1283  - Medium Clear")) {
		sendArduino("CSX CR1283");
		thruSwitchCSX("Medium Speed", 0, null);
	}
	else if (aspectSelected.equals("CSX CR1283A - Medium Approach Medium")) {
		sendArduino("CSX CR1283A");
		thruSwitchCSX("Medium Speed", 2, "Medium Speed");
	}
	else if (aspectSelected.equals("CSX CR1284  - Approach Slow")) {
		sendArduino("CSX CR1284");
		desc04.setText("Proceed, approaching next signal not");
		desc05.setText("exceeding Slow Speed. Trains");
		desc06.setText("exceeding Medium Speed must begin");
		desc07.setText("reduction to Medium Speed as soon as");
		desc08.setText("the locomotive passes the signal.");
	}
	else if (aspectSelected.equals("CSX CR1285  - Approach")) {
		sendArduino("CSX CR1285");
		desc04.setText("Proceed prepared to stop at the next");
		desc05.setText("signal. Trains exceeding Medium");
		desc06.setText("Speed must begin reduction to Medium");
		desc07.setText("Speed as soon as the engine passes");
		desc08.setText("the signal.");
	}
	else if (aspectSelected.equals("CSX CR1286  - Medium Approach")) {
		sendArduino("CSX CR1286");
		thruSwitchCSX("Medium Speed", 1, null);
	}
	else if (aspectSelected.equals("CSX CR1287  - Slow Clear")) {
		sendArduino("CSX CR1287");
		thruSwitchCSX("Slow Speed", 0, null);
	}
	else if (aspectSelected.equals("CSX CR1288  - Slow Approach")) {
		sendArduino("CSX CR1288");
		thruSwitchCSX("Slow Speed", 1, null);
	}
	else if (aspectSelected.equals("CSX CR1290  - Restricting")) {
		sendArduino("CSX CR1290");
		desc04.setText("Proceed at Restricted Speed until");
```

```
            desc05.setText("the entire train has cleared all");
            desc06.setText("switches (if signal is a CP signal)");
            desc07.setText("and the leading wheels have passed a");
            desc08.setText("more favorable signal, or entered");
            desc09.setText("non-signaled DCS territory.");
      }
      else if (aspectSelected.equals("CSX CR1291  - Restricted Proceed")) {
            sendArduino("CSX CR1291");
            desc06.setText("With number plate: Proceed at");
            desc07.setText("Restricted Speed.");
      }
      else if (aspectSelected.equals("CSX CR1292  - Stop")) {
            sendArduino("CSX CR1292");
            desc06.setText("Without number plate or sign: Stop.");
      }


      else if (aspectSelected.equals("CN 803 - Clear")) {
            sendArduino("CN 803");
            desc06.setText("Proceed.");
      }
      else if (aspectSelected.equals("CN 804 - Advance Approach")) {
            sendArduino("CN 804");
            desc06.setText("Proceed, prepared to stop at second"); // CORRECTION
            desc07.setText("signal.");
      }
      else if (aspectSelected.equals("CN 805 - Approach")) {
            sendArduino("CN 805");
            desc06.setText("Proceed, prepared to stop at next"); // CORRECTION
            desc07.setText("signal.");
      }
      else if (aspectSelected.equals("CN 806 - Approach Restricting")) {
            sendArduino("CN 806");
            desc06.setText("Proceed, prepared to pass next signal");
            desc07.setText("at restricted speed.");
      }
      else if (aspectSelected.equals("CN 807 - Advance Approach Diverging")) {
            sendArduino("CN 807");
            desc06.setText("Proceed, prepated to enter diverging");
            desc07.setText("at second signal at prescribed speed.");
      }
      else if (aspectSelected.equals("CN 808 - Approach Diverging")) {
            sendArduino("CN 808");
            desc05.setText("Proceed, prepared to enter diverging");
            desc06.setText("route at next signal at prescribed");
            desc07.setText("speed. Proceed prepared to stop at");
            desc08.setText("second signal.");
      }
      else if (aspectSelected.equals("CN 809 - Diverging Clear")) {
            sendArduino("CN 809");
            desc06.setText("Proceed on diverging route at");
            desc07.setText("prescribed speed.");
      }
      else if (aspectSelected.equals("CN 810 - Diverging Clear Approach Diverging")) {
            sendArduino("CN 810");
```

```
        desc04.setText("Proceed on diverging route at");
        desc05.setText("prescribed speed prepared to enter");
        desc06.setText("diverging route at next signal at");
        desc07.setText("prescribed speed. Proceed prepared");
        desc08.setText("to stop at second signal.");
}
else if (aspectSelected.equals("CN 811 - Diverging Advance Approach")) {
        sendArduino("CN 811");
        desc06.setText("Proceed on diverging route at");
        desc07.setText("prescribed speed prepared to stop at");
        desc08.setText("second signal.");
}
else if (aspectSelected.equals("CN 812 - Diverging Approach")) {
        sendArduino("CN 812");
        desc06.setText("Proceed on diverging route at");
        desc07.setText("prescribed speed prepared to stop at");
        desc08.setText("next signal.");
}
else if (aspectSelected.equals("CN 813 - Diverging Approach Restricting")) {
        sendArduino("CN 813");
        desc06.setText("Proceed on diverging route at");
        desc07.setText("prescribed speed prepared to pass");
        desc08.setText("next signal at restricted speed.");
}
else if (aspectSelected.equals("CN 814 - Restricting")) {
        sendArduino("CN 814");
        desc06.setText("Proceed at restricted speed.");
}
else if (aspectSelected.equals("CN 815 - Restricted Proceed")) {
        sendArduino("CN 815");
        desc06.setText("With number plate: Proceed at");
        desc07.setText("restricted speed.");
}
else if (aspectSelected.equals("CN 816 - Stop")) {
        sendArduino("CN 816");
        desc06.setText("Without number plate: Stop.");
}


else if (aspectSelected.equals("CN 817   - Clear")) {
        sendArduino("CN 817");
        desc06.setText("Proceed");
}
else if (aspectSelected.equals("CN 818   - Advance Approach")) {
        sendArduino("CN 818");
        desc06.setText("Proceed, prepared to stop at second signal.");
}
else if (aspectSelected.equals("CN 818.1 - Medium Advance Approach")) {
        sendArduino("CN 818.1");
        proceedCN(1, 25, 0);
}
else if (aspectSelected.equals("CN 818.2 - Limited Advance Approach")) {
        sendArduino("CN 818.2");
        proceedCN(1, 40, 0);
}
```

```
else if (aspectSelected.equals("CN 819   - Approach")) {
      sendArduino("CN 819");
      desc06.setText("Proceed, prepared to stop at next");
      desc07.setText("signal.");
}
else if (aspectSelected.equals("CN 820   - Approach Limited")) {
      sendArduino("CN 820");
      proceedCN(4, 40, 0);
}
else if (aspectSelected.equals("CN 821   - Approach Medium")) {
      sendArduino("CN 821");
      proceedCN(4, 25, 0);
}
else if (aspectSelected.equals("CN 822   - Approach Slow")) {
      sendArduino("CN 822");
      proceedCN(4, 15, 0);
}
else if (aspectSelected.equals("CN 823   - Limited Clear")) {
      sendArduino("CN 823");
      proceedCN(5, 40, 0);
}
else if (aspectSelected.equals("CN 824   - Limited Clear Limited")) {
      sendArduino("CN 824");
      proceedCN(2, 40, 40);
}
else if (aspectSelected.equals("CN 825   - Limited Approach")) {
      sendArduino("CN 825");
      proceedCN(3, 40, 0);
}
else if (aspectSelected.equals("CN 826   - Medium Clear")) {
      sendArduino("CN 826");
      proceedCN(5, 25, 0);
}
else if (aspectSelected.equals("CN 827   - Medium Clear Medium")) {
      sendArduino("CN 827");
      proceedCN(2, 25, 25);
}
else if (aspectSelected.equals("CN 828   - Medium Approach")) {
      sendArduino("CN 828");
      proceedCN(3, 25, 0);
}
else if (aspectSelected.equals("CN 829   - Slow Clear")) {
      sendArduino("CN 829");
      proceedCN(5, 15, 0);
}
else if (aspectSelected.equals("CN 830   - Slow Approach")) {
      sendArduino("CN 830");
      proceedCN(3, 15, 0);
}
else if (aspectSelected.equals("CN 831   - Restricting")) {
      sendArduino("CN 831");
      desc06.setText("Proceed at restricting speed.");
      desc07.setText(" ");
      desc08.setText("This indication requires a number plate.");
}
```

```
else if (aspectSelected.equals("CN 832   - Stop")) {
	sendArduino("CN 832");
	desc06.setText("Stop");
}


else if (aspectSelected.equals("CN 833 - Clear")) {
	sendArduino("CN 833");
	desc06.setText("Proceed");
}
else if (aspectSelected.equals("CN 834 - Approach Limited")) {
	sendArduino("CN 834");
	proceedCN(4, 35, 0); // CORRECTION
}
else if (aspectSelected.equals("CN 835 - Limited Clear")) {
	sendArduino("CN 835");
	proceedCN(5, 35, 0);
}
else if (aspectSelected.equals("CN 836 - Approach Medium")) {
	sendArduino("CN 836");
	proceedCN(4, 30, 0);
}
else if (aspectSelected.equals("CN 837 - Medium Clear")) {
	sendArduino("CN 837");
	proceedCN(5, 30, 0);
}
else if (aspectSelected.equals("CN 838 - Medium Approach Medium")) {
	sendArduino("CN 838");
	desc05.setText("Proceed through turnouts not");
	desc06.setText("exceeding 30 MPH, approaching next");
	desc07.setText("signal not exceeding 30 MPH.");
}
else if (aspectSelected.equals("CN 839 - Approach")) {
	sendArduino("CN 839");
	desc06.setText("Proceed prepared to stop at next signal.");
}
else if (aspectSelected.equals("CN 840 - Medium Approach")) {
	sendArduino("CN 840");
	desc05.setText("Proceed through turnouts not");
	desc06.setText("exceeding 30 MPH, prepared to stop");
	desc07.setText("at next signal.");
}
else if (aspectSelected.equals("CN 841 - Slow Clear")) {
	sendArduino("CN 841");
	desc06.setText("Proceed through turnouts not");
	desc07.setText("exceeding 20 MPH.");
}
else if (aspectSelected.equals("CN 842 - Slow Approach")) {
	sendArduino("CN 842");
	desc05.setText("Proceed through turnouts not");
	desc06.setText("exceeding 20 MPH, prepared to stop");
	desc07.setText("at next signal");
}
else if (aspectSelected.equals("CN 843 - Restricting")) {
	sendArduino("CN 843");
```

```
		desc06.setText("Proceed at restricted speed.");
	}
	else if (aspectSelected.equals("CN 844 - Stop")) {
		sendArduino("CN 844");
		desc06.setText("Stop");
	}


	else if (aspectSelected.equals("LS&I 230 - Clear")) {
		sendArduino("LS&I 230");
		desc06.setText("Proceed");
	}
	else if (aspectSelected.equals("LS&I 231 - Approach")) {
		sendArduino("LS&I 231");
		desc06.setText("Proceed prepared to stop at next signal.");
	}
	else if (aspectSelected.equals("LS&I 232 - Diverging Clear")) {
		sendArduino("LS&I 232");
		desc06.setText("Proceed on diverging route at");
		desc07.setText("prescribed speed.");
	}
	else if (aspectSelected.equals("LS&I 233 - Diverging Approach")) {
		sendArduino("LS&I 233");
		desc05.setText("Proceed on diverging route at");
		desc06.setText("prescribed speed prepared to stop");
		desc07.setText("at next signal.");
	}
	else if (aspectSelected.equals("LS&I 234 - Restricting")) {
		sendArduino("LS&I 234");
		desc06.setText("Proceed at restricted speed.");
	}
	else if (aspectSelected.equals("LS&I 235 - Stop")) {
		sendArduino("LS&I 235");
		desc06.setText("Stop");
	}


	else if (aspectSelected.equals("BNSF 9.1.3  - Clear")) {
		sendArduino("BNSF 9.1.3");
		desc06.setText("Proceed");
	}
	else if (aspectSelected.equals("BNSF 9.1.4  - Approach Limited")) {
		sendArduino("BNSF 9.1.4");
		desc05.setText("Proceed prepared to pass the next");
		desc06.setText("signal not exceeding 60 MPH and be");
		desc07.setText("prepared to end diverging route at");
		desc08.setText("prescribed speed.");
	}
	else if (aspectSelected.equals("BNSF 9.1.5  - Advance Approach")) {
		sendArduino("BNSF 9.1.5");
		desc05.setText("Proceed prepared to pass next");
		desc06.setText("signal not exceeding 50 MPH and be");
		desc07.setText("prepared to enter diverging route");
		desc08.setText("at prescribed speed.");
	}
```

```
else if (aspectSelected.equals("BNSF 9.1.6  - Approach Medium")) {
      sendArduino("BNSF 9.1.6");
      desc05.setText("Proceed prepared to pass next");
      desc06.setText("signal not exceeding 40 MPH and be");
      desc07.setText("prepared to enter diverging route");
      desc08.setText("at prescribed speed.");
}
else if (aspectSelected.equals("BNSF 9.1.7  - Approach Restricting")) {
      sendArduino("BNSF 9.1.7");
      desc06.setText("Proceed prepared to pass next");
      desc07.setText("signal at restricted speed.");
}
else if (aspectSelected.equals("BNSF 9.1.8  - Approach")) {
      sendArduino("BNSF 9.1.8");
      desc06.setText("Proceed prepared to stop at next");
      desc07.setText("signal, trains exceeding 30 MPH");
      desc08.setText("must immediately reduce to that speed.");
}
else if (aspectSelected.equals("BNSF 9.1.9  - Diverging Clear")) {
      sendArduino("BNSF 9.1.9");
      desc06.setText("Proceed on diverging route not");
      desc07.setText("exceeding prescribed speed through");
      desc08.setText("turnout(s).");
}
else if (aspectSelected.equals("BNSF 9.1.10 - Diverging Approach Diverging")) {
      sendArduino("BNSF 9.1.10");
      desc04.setText("Proceed on diverging route not");
      desc05.setText("exceeding prescribed speed through");
      desc06.setText("turnout, prepared to advance on");
      desc07.setText("diverging route at the next signal,");
      desc08.setText("not exceeding prescribed speed");
      desc09.setText("through turnout.");
}
else if (aspectSelected.equals("BNSF 9.1.11 - Diverging Approach Medium")) {
      sendArduino("BNSF 9.1.11");
      desc05.setText("Proceed on diverging route not");
      desc06.setText("exceeding prescribed speed through");
      desc07.setText("turnout, prepared to pass next");
      desc08.setText("signal not exceeding 35 MPH.");
}
else if (aspectSelected.equals("BNSF 9.1.12 - Diverging Approach")) {
      sendArduino("BNSF 9.1.12");
      desc03.setText("Proceed on diverging route not");
      desc04.setText("exceeding prescribed speed through");
      desc05.setText("turnout, and approach next signal");
      desc06.setText("preparing to stop. If exceeding 30");
      desc07.setText("MPH immediately reduce to that");
      desc08.setText("speed. Note: Speed is 40 MPH for"); // CORRECTION
      desc09.setText("Passenger trains.");
}
else if (aspectSelected.equals("BNSF 9.1.13 - Restricting")) {
      sendArduino("BNSF 9.1.13");
      desc06.setText("Proceed at Restrcited Speed");
      desc07.setText(" ");
      desc08.setText("This indication requires a number or 'G' plate.");
```

```
}
else if (aspectSelected.equals("BNSF 9.1.14 - Restricting")) {
      sendArduino("BNSF 9.1.14");
      desc06.setText("Proceed at Restricted Speed.");
}
else if (aspectSelected.equals("BNSF 9.1.15 - Stop")) {
      sendArduino("BNSF 9.1.15");
      desc06.setText("Stop");
}


else if (aspectSelected.equals("AMTK 9.1.1  - Clear")) {
      sendArduino("AMTK 9.1.1");
      desc06.setText("Proceed");
}
else if (aspectSelected.equals("AMTK 9.1.2  - Advance Approach")) {
      sendArduino("AMTK 9.1.2");
      desc04.setText("Proceed prepared to stop at the");
      desc05.setText("second signal. Trains exceeding");
      desc06.setText("Limited Speed must begin reduction");
      desc07.setText("to Limited Speed as soon as the");
      desc08.setText("engine passes the Advance Approach signal.");
}
else if (aspectSelected.equals("AMTK 9.1.3  - Approach Slow")) {
      sendArduino("AMTK 9.1.3");
      desc04.setText("Proceed appraoching the next signal");
      desc05.setText("at Slow Speed. Trains exceeding");
      desc06.setText("Medium Speed must begin reduction");
      desc07.setText("to Medium Speed as soon as the");
      desc08.setText("engine passes the Approach Slow signal.");
}
else if (aspectSelected.equals("AMTK 9.1.4  - Approach")) {
      sendArduino("AMTK 9.1.4");
      desc04.setText("Proceed prepared to stop at the");
      desc05.setText("next signal. Trains exceeding");
      desc06.setText("Medium Speed must begin reduction");
      desc07.setText("to Medium Speed as soon as the");
      desc08.setText("engine passes the Approach Signal.");
}
else if (aspectSelected.equals("AMTK 9.1.5  - Slow Clear")) {
      sendArduino("AMTK 9.1.5");
      desc05.setText("Proceed at Slow Speed until the");
      desc06.setText("entire train clears all");
      desc07.setText("interlocking or spring switches,");
      desc08.setText("then proceed at maximum authorized speed.");
}
else if (aspectSelected.equals("AMTK 9.1.6  - Slow Approach")) {
      sendArduino("AMTK 9.1.6");
      desc04.setText("Proceed prepared to stop at next");
      desc05.setText("signal. Slow speed applies until");
      desc06.setText("entire train clears all");
      desc07.setText("interlocking or spring switches,");
      desc08.setText("then Medium Speed applies.");
}
else if (aspectSelected.equals("AMTK 9.1.7  - Restricting")) {
      sendArduino("AMTK 9.1.7");
      desc04.setText("Proceed at Restricted Speed until");
      desc05.setText("the entire train has cleared all");
      desc06.setText("interlocking and spring switches");
      desc07.setText("and the leading wheels have passed");
      desc08.setText("a more favorable fixed signal or");
      desc09.setText("entered non-signaled territory.");
}
else if (aspectSelected.equals("AMTK 9.1.8  - Stop and Proceed")) {
      sendArduino("AMTK 9.1.8");
      desc03.setText("Stop, then proceed at restricted");
      desc04.setText("speed until the entire train has");
      desc05.setText("cleared all interlocking and spring");
      desc06.setText("switches and the leading wheels");
      desc07.setText("have passed a more favorable fixed");
      desc08.setText("signal or entered non-signaled territory.");
      desc09.setText(" ");
      desc10.setText("This indication requires a number plate.");
}
else if (aspectSelected.equals("AMTK 9.1.9  - Stop")) {
      sendArduino("AMTK 9.1.9");
      desc06.setText("Stop");
}
else if (aspectSelected.equals("AMTK 9.1.10 - Approach Medium")) {
      sendArduino("AMTK 9.1.10");
      desc06.setText("Proceed, approaching the next signal"); // CORRECTION
      desc07.setText("at Medium Speed.");
}
else if (aspectSelected.equals("AMTK 9.1.11 - Approach Limited")) {
      sendArduino("AMTK 9.1.11");
      desc06.setText("Proceed, approaching the next signal"); // CORRECTION
      desc07.setText("at Limited Speed.");
}
else if (aspectSelected.equals("AMTK 9.1.12 - Medium Clear")) {
      sendArduino("AMTK 9.1.12");
      desc05.setText("Proceed at Medium Speed until");
      desc06.setText("entire train clears all");
      desc07.setText("interlocking or spring switches,");
      desc08.setText("then proceed at maximum authorized speed.");
}
else if (aspectSelected.equals("AMTK 9.1.13 - Limited Clear")) {
      sendArduino("AMTK 9.1.13");
      desc05.setText("Proceed at Limited Speed until");
      desc06.setText("entire train clears all");
      desc07.setText("interlocking or spring switches,");
      desc08.setText("then proceed at maximum authorized speed.");
}
else if (aspectSelected.equals("AMTK 9.1.14 - Medium Approach")) {
      sendArduino("AMTK 9.1.14");
      desc04.setText("Proceed prepared to stop at the");
      desc05.setText("next signal. Trains exceeding");
      desc06.setText("Medium Speed must begin reduction");
      desc07.setText("to Medium Speed as soon as the");
      desc08.setText("Medium Approach Medium signal is");
      desc09.setText("clearly visible.");
```

```
}

else if (aspectSelected.equals("NS 306 - Clear")) {
      sendArduino("NS 306");
      desc06.setText("Proceed at authorized speed.");
}
else if (aspectSelected.equals("NS 307 - Approach Limited")) {
      sendArduino("NS 307");
      desc06.setText("Proceed approaching the next signal");
      desc07.setText("not exceeding limited speed.");
}
else if (aspectSelected.equals("NS 308 - Limited Clear")) {
      sendArduino("NS 308");
      desc04.setText("Proceed at limited speed until");
      desc05.setText("entire train clears all");
      desc06.setText("interlocking, controlled point, or");
      desc07.setText("spring switches, then proceed at");
      desc08.setText("authorized speed.");
}
else if (aspectSelected.equals("NS 309 - Approach Medium")) {
      sendArduino("NS 309");
      desc06.setText("Proceed approaching the next signal");
      desc07.setText("at Medium Speed.");
}
else if (aspectSelected.equals("NS 310 - Advance Approach")) {
      sendArduino("NS 310");
      desc04.setText("Proceed prepared to stop at the");
      desc05.setText("second signal. Trains exceeding");
      desc06.setText("limited speed must begin reduction");
      desc07.setText("to limited speed as soon as the engine"); // CORRECTION
      desc08.setText("passes the signal.");
}
else if (aspectSelected.equals("NS 311 - Medium Clear")) {
      sendArduino("NS 311");
      desc04.setText("Proceed at medium speed until");
      desc05.setText("the entire train clears all"); // CORRECTION
      desc06.setText("interlocking, controlled point, or");
      desc07.setText("spring switches, then proceed at");
      desc08.setText("authorized speed.");
}
else if (aspectSelected.equals("NS 312 - Medium Approach Medium")) {
      sendArduino("NS 312");
      desc02.setText("Proceed at medium speed until");
      desc03.setText("the entire train clears all"); // CORRECTION
      desc04.setText("interlocking, controlled point, or");
      desc05.setText("spring switches, then approach the");
      desc06.setText("next signal at medium speed. Trains");
      desc07.setText("exceeding medium speed must begin");
      desc08.setText("reuction to medium speed as soon");
      desc09.setText("as the medium approach medium");
      desc10.setText("signal is clearly visible.");
}
else if (aspectSelected.equals("NS 313 - Approach Slow")) {
      sendArduino("NS 313");
```

```
        desc05.setText("Proceed approaching the next signal");
        desc06.setText("at slow speed. Trains exceeding");
        desc07.setText("medium speed must, at once, reduce to"); // CORRECTION
        desc08.setText("that speed.");
}
else if (aspectSelected.equals("NS 314 - Approach")) {
        sendArduino("NS 314");
        desc05.setText("Proceed prepared to stop at the");
        desc06.setText("second signal. Trains exceeding");
        desc07.setText("medium speed must, at once, reduce to"); // CORRECTION
        desc08.setText("that speed.");
}
else if (aspectSelected.equals("NS 315 - Medium Approach")) {
        sendArduino("NS 315");
        desc04.setText("Proceed prepared to stop at the");
        desc05.setText("next signal. Trains exceeding");
        desc06.setText("medium speed must begin reduction");
        desc07.setText("to medium speed as soon as the");
        desc08.setText("medium approach signal is clearly visible.");
}
else if (aspectSelected.equals("NS 316 - Slow Clear")) {
        sendArduino("NS 316");
        desc04.setText("Proceed at slow speed until entire");
        desc05.setText("train clears all interlocking,");
        desc06.setText("controlled point, or spring");
        desc07.setText("switches, then proceed at");
        desc08.setText("authorized speed.");
}
else if (aspectSelected.equals("NS 317 - Slow Approach")) {
        sendArduino("NS 317");
        desc04.setText("Proceed prepared to stop at next");
        desc05.setText("signal. Slow speed applies until");
        desc06.setText("entire train clears all");
        desc07.setText("interlocking, controlled point, or");
        desc08.setText("spring switches, then medium speed applies.");
}
else if (aspectSelected.equals("NS 318 - Restricting")) {
        sendArduino("NS 318");
        desc02.setText("Proceed at restricting speed until");
        desc03.setText("entire train clears all");
        desc04.setText("interlocking, controlled point, or");
        desc05.setText("spring switches, and the leading");
        desc06.setText("end has either passed a more");
        desc07.setText("favorable fixed signal, or entered");
        desc08.setText("rule 171 (non-signaled) territory.");
        desc09.setText(" ");
        desc10.setText("This indication requires a number plate.");
}
else if (aspectSelected.equals("NS 319 - Stop")) {
        sendArduino("NS 319");
        desc06.setText("Stop");
}


else if (aspectSelected.equals("NS 326 - Clear")) {
		sendArduino("NS 326");
		desc05.setText("Proceed at authorized speed.");
		desc06.setText(" ");
		desc07.setText("This indication requires a number plate.");
	}
	else if (aspectSelected.equals("NS 327 - Approach Diverging")) {
		sendArduino("NS 327");
		desc05.setText("Proceed preparing to take diverging");
		desc06.setText("route beyond next signal at");
		desc07.setText("authorized speed.");
	}
	else if (aspectSelected.equals("NS 328 - Advance Approach")) {
		sendArduino("NS 328");
		desc06.setText("Proceed preparing to stop at second signal.");
	}
	else if (aspectSelected.equals("NS 329 - Diverging Clear")) {
		sendArduino("NS 329");
		desc05.setText("Proceed through diverging route,");
		desc06.setText("observing authorized speed through");
		desc07.setText("turnout(s) or crossover(s).");
	}
	else if (aspectSelected.equals("NS 330 - Diverging Approach Diverging")) {
		sendArduino("NS 330");
		desc04.setText("Proceed through turnout(s) or");
		desc05.setText("crossover(s) at authorized speed");
		desc06.setText("preparing to take diverging route");
		desc07.setText("beyond next signal at authorized speed.");
	}
	else if (aspectSelected.equals("NS 331 - Approach")) {
		sendArduino("NS 331");
		desc05.setText("Proceed preparing to stop at");
		desc06.setText("next signal. Trains exceeding medium"); // CORRECTION
		desc07.setText("speed must, at once, reduce to that speed."); // CORRECTION
		desc08.setText(" ");
		desc09.setText("This indication requires a number plate.");
	}
	else if (aspectSelected.equals("NS 333 - Diverging Approach")) {
		sendArduino("NS 333");
		desc03.setText("Proceed through diverging route,");
		desc04.setText("observing authorized speed through");
		desc05.setText("turnout(s) and crossover(s)");
		desc06.setText("preparing to stop at next signal.");
		desc07.setText("Trians exceeding medium speed must,"); // CORRECTION
		desc08.setText("at once, begin reduction to that speed."); // CORRECTION
	}
	else if (aspectSelected.equals("NS 334 - Slow Clear")) {
		sendArduino("NS 334");
		desc05.setText("Procced at Slow speed within"); // CORRECTION
		desc06.setText("controlled point/interlocking");
		desc07.setText("limits or through turnout(s) or crossover(s).");
	}
	else if (aspectSelected.equals("NS 335 - Slow Approach")) {
		sendArduino("NS 335");
		desc05.setText("Proceed preparing to stop at next");
		desc06.setText("signal; Slow speed within");
		desc07.setText("controlled point/interlocking");
		desc08.setText("limits or through turnout(s) or crossover(s).");
	}
	else if (aspectSelected.equals("NS 336 - Restricting")) {
		sendArduino("NS 336");
		desc05.setText("Proceed at restricted speed, until");
		desc06.setText("the leading end either passes a");
		desc07.setText("more favorable fixed signal, or");
		desc08.setText("enters non-signaled track.");
	}
	else if (aspectSelected.equals("NS 337 - Stop")) {
		sendArduino("NS 337");
		desc06.setText("Stop");
	}


	else if (aspectSelected.equals("NS 340 - Clear")) {
		sendArduino("NS 340");
		desc05.setText("Proceed at authorized speed.");
		desc06.setText(" ");
		desc07.setText("This indication requires a number plate.");
	}
	else if (aspectSelected.equals("NS 341 - Approach Diverging")) {
		sendArduino("NS 341");
		desc05.setText("Proceed, preparing to take diverging"); // CORRECTION
		desc06.setText("route at next signal at authorized speed.");
	}
	else if (aspectSelected.equals("NS 342 - Advance Approach")) {
		sendArduino("NS 342");
		desc06.setText("Proceed, preparing to stop at second signal."); // CORRECTION
	}
	else if (aspectSelected.equals("NS 343 - Diverging Clear")) {
		sendArduino("NS 343");
		desc05.setText("Proceed through diverging route,");
		desc06.setText("observing authorized speed through");
		desc07.setText("turnout(s) and crossover(s).");
	}
	else if (aspectSelected.equals("NS 344 - Approach Restricted")) {
		sendArduino("NS 344");
		desc03.setText("Proceed, approaching next signal at");
		desc04.setText("restricted speed, not exceeding 15");
		desc05.setText("MPH. Trains exceeding medium speed"); // CORRECT
		desc06.setText("must at once reduce to that speed.");
		desc07.setText(" ");
		desc08.setText("This indication requires a number plate.");
	}
	else if (aspectSelected.equals("NS 345 - Diverging Approach Restricted")) {
		sendArduino("NS 345");
		desc02.setText("Proceed through diverging route,");
		desc03.setText("observing authorized speed through");
		desc04.setText("turnout(s) and crossover(s),");
		desc05.setText("approaching next signal at");
		desc06.setText("Restricted Speed, not exceeding 15");
		desc07.setText("MPH. Trains exceeding medium speed");
		desc08.setText("must at once begin reduction to");
```

```
        desc09.setText("that speed.");
}
else if (aspectSelected.equals("NS 346 - Approach")) {
        sendArduino("NS 346");
        desc04.setText("Proceed, preparing to stop at next"); // CORRECTION
        desc05.setText("signal. Trains exceeding medium");
        desc06.setText("speed must at once reduce to that speed.");
        desc07.setText(" ");
        desc08.setText("This indication requires a number plate.");
}
else if (aspectSelected.equals("NS 347 - Diverging Approach")) {
        sendArduino("NS 347");
        desc03.setText("Proceed onto diverging route,");
        desc04.setText("observing authorized speed through");
        desc05.setText("turnout(s) and crossover(s),");
        desc06.setText("preparing to stop at the next");
        desc07.setText("signal. Trains exceeding medium");
        desc08.setText("speed must at once reduce to that speed.");
}
else if (aspectSelected.equals("NS 348 - Restricting")) {
        sendArduino("NS 348");
        desc03.setText("Proceed at Restricted Speed.");
        desc04.setText("Restricted speed must be observed");
        desc05.setText("until the leading wheels reach a");
        desc06.setText("more favorable fixed signal, or");
        desc07.setText("enter non-signaled territory.");
        desc08.setText(" ");
        desc09.setText("This indication requires a number plate.");
}
else if (aspectSelected.equals("NS 349 - Stop")) {
        sendArduino("NS 349");
        desc06.setText("Stop");
}


else if (aspectSelected.equals("NYC 281  - Clear")) {
        sendArduino("NYC 281");
        desc06.setText("Clear");
}
else if (aspectSelected.equals("NYC 281A - Advance Approach Medium")) {
        sendArduino("NYC 281A");
        desc06.setText("Proceed approaching second signal");
        desc07.setText("at medium speed.");
}
else if (aspectSelected.equals("NYC 281B - Approach Limited")) {
        sendArduino("NYC 281B");
        desc03.setText("Proceed approaching the next signal");
        desc04.setText("at Limited Speed. Trains exceeding");
        desc05.setText("60 MPH must at once reduce to that");
        desc06.setText("speed. Reduction to 60 MPH must");
        desc07.setText("commence before passing signal, and");
        desc08.setText("must be completed before accepting");
        desc09.setText("a more favorable indication.");
}
else if (aspectSelected.equals("NYC 281C - Limited Clear")) {
	sendArduino("NYC 281C");
	desc06.setText("Procced. Limited Speed applies");
	desc07.setText("through interlocking limits.");
}
else if (aspectSelected.equals("NYC 281D - Limited Approach")) {
	sendArduino("NYC 281D");
	desc06.setText("Proceed at limited speed prepared");
	desc07.setText("to stop at the next signal."); // CORRECTION
}
else if (aspectSelected.equals("NYC 282  - Approach Medium")) {
	sendArduino("NYC 282");
	desc02.setText("Proceed, approaching next signal at");
	desc03.setText("Medium Speed. Trains exceeding");
	desc04.setText("limited speed must at once reduce");
	desc05.setText("to that speed. Reduction to limited");
	desc06.setText("speed must commence before passing");
	desc07.setText("signal, and must be completed");
	desc08.setText("before accepting a more-favorable");
	desc09.setText("indication.");
}
else if (aspectSelected.equals("NYC 282A - Advance Approach")) {
	sendArduino("NYC 282A");
	desc02.setText("Procced, preparing to stop at second"); // CORRECTION
	desc03.setText("signal. Trains exceeding Limited");
	desc04.setText("Speed must, at once, reduce to that"); // CORRECTION
	desc05.setText("speed. Reduction to limited speed");
	desc06.setText("must commence before passing");
	desc07.setText("signal, and must be completed");
	desc08.setText("before accepting a more-favorable");
	desc09.setText("indication.");
}
else if (aspectSelected.equals("NYC 283  - Medium Clear")) {
	sendArduino("NYC 283");
	desc06.setText("Proceed. Medium Speed within"); // CORRECTION
	desc07.setText("interlocking limits.");
}
else if (aspectSelected.equals("NYC 283A - Medium Advance Approach")) {
	sendArduino("NYC 283A");
	desc05.setText("Procced, preparing to stop at second"); // CORRECTION
	desc06.setText("signal. Medium Speed within"); // CORRECTION
	desc07.setText("interlocking limits.");
}
else if (aspectSelected.equals("NYC 283B - Medium Approach Slow")) {
	sendArduino("NYC 283B");
	desc06.setText("Procced at medium speed,");
	desc07.setText("approaching next signal at slow speed.");
}
else if (aspectSelected.equals("NYC 284  - Approach Slow")) {
	sendArduino("NYC 284");
	desc05.setText("Proceed, approaching next signal at"); // CORRECTION
	desc06.setText("Slow Speed. Trains exceeding Medium"); // CORRECTION
	desc07.setText("Speed must at once reduce to that speed."); // CORRECTION
}
else if (aspectSelected.equals("NYC 285  - Approach")) {
	sendArduino("NYC 285");
```

```
        desc02.setText("Proceed, prepared to stop at the"); // CORRECTION
        desc03.setText("next signal. Trains exceeding");
        desc04.setText("Medium Speed must at once reduce to"); // CORRECTION
        desc05.setText("that speed. Reduction to Medium"); // CORRECTION
        desc06.setText("Speed must comment before passing");
        desc07.setText("signal, and must be complete before");
        desc08.setText("accepting a more-favorable signal.");
}
else if (aspectSelected.equals("NYC 286  - Medium Approach")) {
        sendArduino("NYC 286");
        desc06.setText("Proceed at Medium Speed prepared to"); // CORRECTION
        desc07.setText("stop at the next signal.");
}
else if (aspectSelected.equals("NYC 287  - Slow Clear")) {
        sendArduino("NYC 287");
        desc06.setText("Proceed. Slow Speed within"); // CORRECTION
        desc07.setText("interlocking limits.");
}
else if (aspectSelected.equals("NYC 288  - Slow Approach")) {
        sendArduino("NYC 288");
        desc05.setText("Proceed, prepared to stop at the"); // CORRECTION
        desc06.setText("next signal. Slow Speed within"); // CORRECTION
        desc07.setText("interlocking limits.");
}
else if (aspectSelected.equals("NYC 290  - Restricting")) {
        sendArduino("NYC 290");
        desc06.setText("Proceed at Restricted Speed."); // CORRECTION
}
else if (aspectSelected.equals("NYC 291  - Stop and Proceed")) {
        sendArduino("NYC 291");
        desc04.setText("Stop before passing signal. Then");
        desc05.setText("proceed at Restricted Speed."); // CORRECTION
        desc06.setText(" ");
        desc07.setText("This indication requires offset heads and/or a number plate.");
}
else if (aspectSelected.equals("NYC 292  - Stop")) {
        sendArduino("NYC 292");
        desc06.setText("Stop before passing signal.");
}


else if (aspectSelected.equals("UP 9.2.1  - Clear")) {
        sendArduino("UP 9.2.1");
        desc06.setText("Proceed");
}
else if (aspectSelected.equals("UP 9.2.2  - Approach Clear Sixty")) {
        sendArduino("UP 9.2.2");
        longDescUP(1, 60, 60, 60, 60);
}
else if (aspectSelected.equals("UP 9.2.3  - Approach Clear Fifty")) {
        sendArduino("UP 9.2.3");
        longDescUP(1, 50, 50, 50, 50);
}
else if (aspectSelected.equals("UP 9.2.4  - Advance Approach")) {
        sendArduino("UP 9.2.4");
```

```
		longDescUP(3, 40, 40, 40, 40);
	}
	else if (aspectSelected.equals("UP 9.2.4P - Advance Approach Passenger")) {
		sendArduino("UP 9.2.4P");
		longDescUP(2, 40, 40, 60, 0);
		desc07.setText(" ");
		desc08.setText("This indication requires a 'C' Commuter Plate.");
	}
	else if (aspectSelected.equals("UP 9.2.5  - Approach Diverging")) {
		sendArduino("UP 9.2.5");
		desc05.setText("Proceed, prepared to advance on"); // CORRECTION
		desc06.setText("diverging route at next signal at");
		desc07.setText("prescribed speed through turnout.");
	}
	else if (aspectSelected.equals("UP 9.2.6  - Approach")) {
		sendArduino("UP 9.2.6");
		desc03.setText("Proceed prepared to stop before any");
		desc04.setText("part of the train passes the next");
		desc05.setText("signal. Freight trains exceeding 30");
		desc06.setText("MPH must immediately reduce to 30");
		desc07.setText("MPH. Passenger trains exceeding 40");
		desc08.setText("MPH must immediately reduce to 40 MPH.");
	}
	else if (aspectSelected.equals("UP 9.2.7  - Approach Restricting")) {
		sendArduino("UP 9.2.7");
		desc02.setText("Proceed prepared to pass next");
		desc03.setText("signal at Restricted Speed, but not");
		desc04.setText("exceeding 15 MPH. When the next");
		desc05.setText("signal is seen to display a proceed");
		desc06.setText("indication, the requirement to pass");
		desc07.setText("the next signal at restricted speed");
		desc08.setText("no longer applies. Speed may be");
		desc09.setText("resumed once leading wheels of the"); // CORRECTION
		desc10.setText("train have passed the next signal"); // CORRECTION
	}
	else if (aspectSelected.equals("UP 9.2.8  - Diverging Clear Limited")) {
		sendArduino("UP 9.2.8");
		desc05.setText("Proceed on diverging route. Speed");
		desc06.setText("through turnout must not exceed 40 MPH.");
	}
	else if (aspectSelected.equals("UP 9.2.9  - Diverging Clear")) {
		sendArduino("UP 9.2.9");
		desc05.setText("Proceed on diverging route not");
		desc06.setText("exceeding prescribed speed through turnout.");
	}
	else if (aspectSelected.equals("UP 9.2.10 - Diverging Advance Approach")) {
		sendArduino("UP 9.2.10");
		desc01.setText("Proceed on diverging route not");
		desc02.setText("exceeding prescribed speed through");
		desc03.setText("turnout and be prepared to stop at");
		desc04.setText("second signal. Freight trains");
		desc05.setText("exceeding 40 MPH must immediately");
		desc06.setText("reduce to 40 MPH. Passenger trains");
		desc07.setText("may proceed, but must be prepared to");
		desc08.setText("pass next signal not exceeding 40");
		desc09.setText("MPH. When signal governs the");
		desc10.setText("approach to a control point with a");
		desc11.setText("40 MPH turnout speed be prepared to");
		desc12.setText("advance on normal or diverging route.");
	}
	else if (aspectSelected.equals("UP 9.2.10P- Diverging Advance Approach Passenger")) {
		sendArduino("UP 9.2.10P");
		desc02.setText("Proceed on diverging route at");
		desc03.setText("prescribed speed through turnout");
		desc04.setText("prepared to stop at second signal.");
		desc05.setText("Freight trains exceeding 40 MPH");
		desc06.setText("must immediately reduce to 40 MPH.");
		desc07.setText("Passenger trains exceeding 60 MPH");
		desc08.setText("must immediately reduce to 60 MPH");
		desc09.setText(" ");
		desc10.setText("This indication requries a 'C' Commuter Plate.");
	}
	else if (aspectSelected.equals("UP 9.2.11 - Diverging Approach")) {
		sendArduino("UP 9.2.11");
		desc02.setText("N diverging route at prescribed"); //?
		desc03.setText("speed through turnout prepared to");
		desc04.setText("stop before any part of train or");
		desc05.setText("engine passes the next signal.");
		desc06.setText("Freight trains exceeding 30 MPH");
		desc07.setText("must immediately reduce to 30 MPH.");
		desc08.setText("Passenger trains exceeding 40 MPH");
		desc09.setText("must immediately reduce to 40 MPH.");
	}
	else if (aspectSelected.equals("UP 9.2.12 - Diverging Approach Diverging")) {
		sendArduino("UP 9.2.12");
		desc04.setText("Proceed on diverging route not");
		desc05.setText("exceeding prescribed speed through");
		desc06.setText("turnout prepared to advance on");
		desc07.setText("diverging route at the next signal");
		desc08.setText("at prescribed speed through the turnout.");
	}
	else if (aspectSelected.equals("UP 9.2.13 - Restricting")) {
		sendArduino("UP 9.2.13");
		desc04.setText("Proceed at restricted speed, not");
		desc05.setText("exceed prescribed speed through");
		desc06.setText("turnout when applicable.");
		desc07.setText(" ");
		desc08.setText("This indication requires a number and 'G' plate.");
	}
	else if (aspectSelected.equals("UP 9.2.14 - Restricted Proceed")) {
		sendArduino("UP 9.2.14");
		desc06.setText("Proceed at restricted speed.");
		desc07.setText(" ");
		desc08.setText("This indication requires a number plate.");
	}
	else if (aspectSelected.equals("UP 9.2.15 - Stop")) {
		sendArduino("UP 9.2.15");
		desc06.setText("Stop before any part of the train"); // CORRECTION
		desc07.setText("passes the signal.");
	}
```

```
else if (aspectSelected.equals("UP 9.2.16 - Diverging Approach Clear Fifty")) {
      sendArduino("UP 9.2.16");
      desc01.setText("Proceed on diverging route at");
      desc02.setText("prescribed speed through turnout.");
      desc03.setText("Freight trains exceeding 50 MPH");
      desc04.setText("must immediately reduce to 50 MPH.");
      desc05.setText("Passenger trains may proceed, but");
      desc06.setText("must be prepared to pass the next");
      desc07.setText("signal not exceeding 50 MPH. When");
      desc08.setText("signal governs the approach to a");
      desc09.setText("control point with a 50 MPH turnout");
      desc10.setText("speed, be prepared to advance on");
      desc11.setText("diverging route.");
}


else if (aspectSelected.equals("CROR 405 - Clear")) {
      sendArduino("CROR 405");
      desc06.setText("Proceed");
}
else if (aspectSelected.equals("CROR 406 - Clear to Limited")) {
      sendArduino("CROR 406");
      proceedCROR(1, "Limited Speed");
      desc07.setText(" ");
      desc08.setText("This indication requires an 'L' plate.");
}
else if (aspectSelected.equals("CROR 407 - Clear to Medium")) {
      sendArduino("CROR 407");
      proceedCROR(1, "Medium Speed");
}
else if (aspectSelected.equals("CROR 408 - Clear to Diverging")) {
      sendArduino("CROR 408");
      proceedCROR(1, "Diverging Speed");
      desc07.setText(" ");
      desc08.setText("This indication requires a 'DV' plate.");
}
else if (aspectSelected.equals("CROR 409 - Clear to Slow")) {
      sendArduino("CROR 409");
      proceedCROR(1, "Slow Speed");
}
else if (aspectSelected.equals("CROR 410 - Clear to Restricting")) {
      sendArduino("CROR 410");
      desc06.setText("Proceed, next signal is displaying");
      desc07.setText("a Restricting indication."); // CORRECTION
}
else if (aspectSelected.equals("CROR 411 - Clear to Stop")) {
      sendArduino("CROR 411");
      desc06.setText("Proceed, preparing to stop at next signal.");
}
else if (aspectSelected.equals("CROR 412 - Advance Clear to Limited")) {
      sendArduino("CROR 412");
      proceedCROR(2, "Limited Speed");
}
else if (aspectSelected.equals("CROR 413 - Advance Clear to Medium")) {
      sendArduino("CROR 413");
```

```
        proceedCROR(2, "Medium Speed");
}
else if (aspectSelected.equals("CROR 414 - Advance Clear to Slow")) {
        sendArduino("CROR 414");
        proceedCROR(2, "Slow Speed");
        desc07.setText(" ");
        desc08.setText("This indication requires a 'DV' plate.");
}
else if (aspectSelected.equals("CROR 414A - Advance Clear to Diverging")) {
        sendArduino("CROR 414A");
        proceedCROR(2, "Diverging Speed");
}
else if (aspectSelected.equals("CROR 415 - Advance Clear to Stop")) {
        sendArduino("CROR 415");
        desc05.setText("Proceed, next signal is displaying");
        desc06.setText("the Clear to Stop indication, be prepared"); // CORRECTION
        desc07.setText("to stop at the second signal."); // CORRECTION
}
else if (aspectSelected.equals("CROR 416 - Limited to Clear")) {
        sendArduino("CROR 416");
        desc05.setText("Proceed at Limited Speed past");
        desc06.setText("signal and through turnouts.");
        desc07.setText(" ");
        desc08.setText("This indication requires an 'L' plate.");
}

else if (aspectSelected.equals("CROR 417 - Limited to Limited")) {
        sendArduino("CROR 417");
        turnOutCROR("Limited Speed", "Limited Speed");
}
else if (aspectSelected.equals("CROR 418 - Limited to Medium")) {
        sendArduino("CROR 418");
        turnOutCROR("Limited Speed", "Medium Speed");
}
else if (aspectSelected.equals("CROR 419 - Limited to Slow")) {
        sendArduino("CROR 419");
        turnOutCROR("Limited Speed", "Slow Speed");
}
else if (aspectSelected.equals("CROR 419A - Limited to Diverging")) {
        sendArduino("CROR 419A");
        turnOutCROR("Limited Speed", "Diverging Speed");
        desc04.setText(" ");
        desc05.setText("This indication requires a 'DV' plate.");
}
else if (aspectSelected.equals("CROR 420 - Limited to Restricting")) {
        sendArduino("CROR 420");
        desc05.setText("Proceed at Limited Speed past");
        desc06.setText("the signal and through turnouts, next"); // CORRECTION
        desc07.setText("signal is displaying Restricting.");
        desc08.setText(" ");
        desc09.setText("This indication requires a 'DV' plate.");
}
else if (aspectSelected.equals("CROR 421 - Limited to Stop")) {
        sendArduino("CROR 421");
        desc05.setText("Proceed at Limited Speed past");
```

```
		desc06.setText("the signal and through turnouts,"); // CORRECTION
		desc07.setText("preparing to stop at the next signal."); // CORRECTION
	}
	else if (aspectSelected.equals("CROR 422 - Medium to Clear")) {
		sendArduino("CROR 422");
		desc06.setText("Proceed at Medium Speed past signal");
		desc07.setText("and through turnouts.");
	}
	else if (aspectSelected.equals("CROR 423 - Medium to Limited")) {
		sendArduino("CROR 423");
		turnOutCROR("Medium Speed", "Limited Speed");
	}
	else if (aspectSelected.equals("CROR 424 - Medium to Medium")) {
		sendArduino("CROR 424");
		turnOutCROR("Medium Speed", "Medium Speed");
	}
	else if (aspectSelected.equals("CROR 425 - Medium to Slow")) {
		sendArduino("CROR 425");
		turnOutCROR("Medium Speed", "Slow Speed");
	}
	else if (aspectSelected.equals("CROR 425A - Medium to Diverging")) {
		sendArduino("CROR 425A");
		turnOutCROR("Medium Speed", "Diverging Speed");
	}
	else if (aspectSelected.equals("CROR 426 - Medium to Restricting")) {
		sendArduino("CROR 426");
		desc05.setText("Proceed at Medium Speed past signal");
		desc06.setText("and through turnouts, next signal");
		desc07.setText("is displaying Restricting,");
	}
	else if (aspectSelected.equals("CROR 427 - Medium to Stop")) {
		sendArduino("CROR 427");
		pastSignalCROR(1, "Medium Speed", "");
	}
	else if (aspectSelected.equals("CROR 428 - Diverging to Clear")) {
		sendArduino("CROR 428");
		pastSignalCROR(2, "Diverging Speed", "");
		desc07.setText(" ");
		desc08.setText("This indication requires a 'DV' plate.");
	}
	else if (aspectSelected.equals("CROR 429 - Diverging to Stop")) {
		sendArduino("CROR 429");
		pastSignalCROR(1, "Diverging Speed", "");
		desc08.setText(" ");
		desc09.setText("This indication requires a 'DV' plate.");
	}
	else if (aspectSelected.equals("CROR 430 - Diverging")) {
		sendArduino("CROR 430");
		desc05.setText("Proceed at Reduced Speed, not");
		desc06.setText("exceeding Diverging Speed past");
		desc07.setText("signal and through turnouts.");
		desc08.setText(" ");
		desc09.setText("This indication requires a 'DV' plate.");
	}
	else if (aspectSelected.equals("CROR 431 - Slow to Clear")) {
        sendArduino("CROR 431");
        pastSignalCROR(2, "Slow Speed", "");
}
else if (aspectSelected.equals("CROR 432 - Slow to Limited")) {
        sendArduino("CROR 432");
        pastSignalCROR(3, "Slow Speed", "Limited Speed");
        desc08.setText(" ");
        desc09.setText("This indication requires a 'DV' plate.");
}
else if (aspectSelected.equals("CROR 432A - Diverging to Limited")) {
        sendArduino("CROR 432A");
        pastSignalCROR(3, "Diverging Speed", "Limited Speed");
        desc08.setText(" ");
        desc09.setText("This indication requires a 'DV' plate.");
}
else if (aspectSelected.equals("CROR 433 - Slow to Medium")) {
        sendArduino("CROR 433");
        pastSignalCROR(3, "Slow Speed", "Medium Speed");
}
else if (aspectSelected.equals("CROR 433A - Diverging to Medium")) {
        sendArduino("CROR 433A");
        pastSignalCROR(3, "Diverging Speed", "Medium Speed");
        desc08.setText(" ");
        desc09.setText("This indication requires a 'DV' plate.");
}
else if (aspectSelected.equals("CROR 434 - Slow to Slow")) {
        sendArduino("CROR 434");
        pastSignalCROR(3, "Slow Speed", "Slow Speed");
}
else if (aspectSelected.equals("CROR 434A - Diverging to Diverging")) {
        sendArduino("CROR 434A");
        pastSignalCROR(3, "Diverging Speed", "Diverging Speed");
        desc08.setText(" ");
        desc09.setText("This indication requires a 'DV' plate.");
}
else if (aspectSelected.equals("CROR 435 - Slow to Stop")) {
        sendArduino("CROR 435");
        pastSignalCROR(1, "Slow Speed", "");
}
else if (aspectSelected.equals("CROR 436 - Restricting")) {
        sendArduino("CROR 436");
        desc05.setText("Proceed at Restricted Speed.");
        desc06.setText(" ");
        desc07.setText("This indication requires an 'R' plate.");
}
else if (aspectSelected.equals("CROR 437 - Stop and Proceed")) {
        sendArduino("CROR 437");
        desc05.setText("Stop, then proceed at Restricted Speed.");
        desc06.setText(" ");
        desc07.setText("This indication requires that the signal heads be offset.");
}
else if (aspectSelected.equals("CROR 438 - Take/Leave Siding")) {
        sendArduino("CROR 438");
        desc05.setText("Indications will be specific in the");
        desc06.setText("Special Instructions for each");
```

```
		desc07.setText("specific application of this signal.");
	}
	else if (aspectSelected.equals("CROR 439 - Stop")) {
		sendArduino("CROR 439");
		desc05.setText("Stop");
		desc06.setText(" ");
		desc07.setText("This indication requires an 'A' plate.");
	}


	else if (aspectSelected.equals("NORAC 281  - Clear")) {
		sendArduino("NORAC 281");
		desc06.setText("Proceed not exceeding Normal Speed.");
	}
	else if (aspectSelected.equals("NORAC 281A - Cab Speed")) {
		sendArduino("NORAC 281A");
		desc04.setText("Proceed in accordance with cab");
		desc05.setText("signal indication. Reduce speed to");
		desc06.setText("not exceeding 60 MPH if Cab Speed"); // CORRECTION
		desc07.setText("cab signal is displayed without a");
		desc08.setText("signal speed, or if cab signal is"); // CORRECTION
		desc09.setText("not operative.");
	}
	else if (aspectSelected.equals("NORAC 281B - Approach Limited")) {
		sendArduino("NORAC 281B");
		desc06.setText("Proceed appraoching next signal not");
		desc07.setText("exceeding Limited Speed.");
	}
	else if (aspectSelected.equals("NORAC 281C - Limited Clear")) {
		sendArduino("NORAC 281C");
		desc05.setText("Proceed at Limited Speed until");
		desc06.setText("entire train clears all");
		desc07.setText("interlocking or spring switches,");
		desc08.setText("then proceed at Normal Speed.");
	}
	else if (aspectSelected.equals("NORAC 282  - Approach Medium")) {
		sendArduino("NORAC 282");
		desc06.setText("Proceed approaching the next signal");
		desc07.setText("at Medium Speed.");
	}
	else if (aspectSelected.equals("NORAC 282A - Advanced Approach")) {
		sendArduino("NORAC 282A");
		desc05.setText("Proceed prepared to stop at the");
		desc06.setText("second signal. Trains exceeding");
		desc07.setText("Limited Speed must begin reduction");
		desc08.setText("to Limited Speed as soon as the engine"); // CORRECTION
		desc09.setText("passes the signal.");
	}
	else if (aspectSelected.equals("NORAC 283  - Medium Clear")) {
		sendArduino("NORAC 283");
		desc05.setText("Proceed at Medium Speed until");
		desc06.setText("the entire train clears all"); // CORRECTION
		desc07.setText("interlocking or spring switches,");
		desc08.setText("then proceed at Normal Speed.");
	}
```

```java
else if (aspectSelected.equals("NORAC 283A - Medium Approach Medium")) {
	sendArduino("NORAC 283A");
	desc03.setText("Proceed at Medium Speed until");
	desc04.setText("the entire trains clears all"); // CORRECTION
	desc05.setText("interlocking or spring switches,");
	desc06.setText("then approach the next signal at");
	desc07.setText("Medium Speed. Trains exceeding");
	desc08.setText("Medium Speed must begin reduction");
	desc09.setText("to Medium Speed as soon as the");
	desc10.setText("signal is clearly visible.");
}
else if (aspectSelected.equals("NORAC 284  - Approach Slow")) {
	sendArduino("NORAC 284");
	desc05.setText("Proceed approaching the next signal");
	desc06.setText("at Slow Speed. Trains exceeing");
	desc07.setText("Medium Speed must begin reduction");
	desc08.setText("to Medoum Speed as soon as the");
	desc09.setText("engine passes the signal.");
}
else if (aspectSelected.equals("NORAC 285  - Approach")) {
	sendArduino("NORAC 285");
	desc05.setText("Proceed prepared to stop at the"); // CORRECTION
	desc06.setText("next signal. Trains exceeding");
	desc07.setText("Medium Speed must begin reduction");
	desc08.setText("to Medium Speed as soon as the");
	desc09.setText("engine passses the signal.");
}
else if (aspectSelected.equals("NORAC 286  - Medium Approach")) {
	sendArduino("NORAC 286");
	desc05.setText("Proceed prepared to stop at the");
	desc06.setText("next signal. Trains exceeding");
	desc07.setText("Medium Speed must begin reduction");
	desc08.setText("to Medium Speed as soon as the");
	desc09.setText("Medium Approach signal is clearly visible.");
}
else if (aspectSelected.equals("NORAC 287  - Slow Clear")) {
	sendArduino("NORAC 287");
	desc05.setText("Proceed at Slow Speed until the entire"); // CORRECTION
	desc06.setText("train clears all interlocking or");
	desc07.setText("spring switches, then proceed at");
	desc08.setText("Normal Speed.");
}
else if (aspectSelected.equals("NORAC 288  - Slow Approach")) {
	sendArduino("NORAC 288");
	desc05.setText("Proceed prepared to stop at the next");
	desc06.setText("signal. Slow Speed applies until the"); // CORRECTION
	desc07.setText("entire train clears all");
	desc08.setText("interlocking or spring switches,");
	desc09.setText("then Medium Speed applies.");
}
else if (aspectSelected.equals("NORAC 290  - Restricting")) {
	sendArduino("NORAC 290");
	desc03.setText("Proceed at Restricted Speed until the"); // CORRECTION
	desc04.setText("entire train has cleared all");
	desc05.setText("interlocking and spring switches");
```

```
                                desc06.setText("(if the signal is an interlocking");
                                desc07.setText("or controlled point signal) and the");
                                desc08.setText("leading wheels have either passed a");
                                desc09.setText("more favorable fixed signal, or");
                                desc10.setText("enterned non-signaled territory.");
                        }
                        else if (aspectSelected.equals("NORAC 291  - Stop and Proceed")) {
                                sendArduino("NORAC 291");
                                desc02.setText("Stop, then proceed at Restricted");
                                desc03.setText("Speed until the entire train has");
                                desc04.setText("cleared all interlocking or spring");
                                desc05.setText("switches (if the signal is an");
                                desc06.setText("interlocking or controlled point signal)");
                                desc07.setText("and the leading wheels have either");
                                desc08.setText("passed a more favorable fixed");
                                desc09.setText("signal, or entered non-signaled territory.");
                                desc10.setText(" ");
                                desc11.setText("This indication requires a number plate.");
                        }
                        else if (aspectSelected.equals("NORAC 292  - Stop")) {
                                sendArduino("NORAC 292");
                                desc06.setText("Stop");
                        }

                        window.pack();
                } catch (IOException ee) { }
        }
    }

    @Override
    public void mouseEntered(MouseEvent e) {}
    @Override
    public void mouseExited(MouseEvent e) {}
    @Override
    public void mousePressed(MouseEvent e) {}
    @Override
    public void mouseReleased(MouseEvent e) {}
}

/**
 * Change Listener for the duration slider
 */
class durSlider implements ChangeListener {

    @Override
    public void stateChanged(ChangeEvent arg0) {
        JSlider source = (JSlider) arg0.getSource();
        int value = source.getValue();
        aspHdr.setText("Duration: " + value + " seconds");
    }

}

/**
 * Action Listener for the duration OK button
```

```
	 */
	class durClick implements ActionListener {

		@Override
		public void actionPerformed(ActionEvent arg0) {
			rotate = true;
			try {
				sendArduino(Integer.toString(duration.getValue()));
				cycleKill.addActionListener(new cycleKillClick());
				//for (MouseListener ml : rules.getMouseListeners()) {
				//	rules.removeMouseListener(ml);
				//}
				footer.add(cycleKill);
				window.repaint();
			} catch (IOException e) { }
		}

	}
}
```