3 Client GUI Design

3.1 Introduction

The user interface of a remote laboratory should be as realistic as possible to give users an impression that they are actually operating on the physically existing instruments. Also, a user-friendly interface will enable the user to conduct the experiments conveniently and efficiently. The design and creation of vivid control and parameter adjustment components such as buttons, knobs, and cables are thus essential in any Web-based laboratory.

Typically, these movable components are positioned on top of an instrument panel or an appropriate laboratory setup or background. On the panel, buttons can be pressed or released, knobs can be turned, and plugs and cables can be plugged in or dragged to connect or disconnect relevant input or output terminals in the laboratory experiment. As examples, some instrument panels for the frequency modulation and oscilloscope experiments are illustrated in Figures 3.1 and 3.2, respectively.

The client graphical user interface (GUI), which includes the instrument panel and other information such as experimental procedures and real-time experimental data, may be implemented using Java, JavaScript, and HTML technologies, most of which are supported by popular Web browsers such as Microsoft Internet Explorer and Netscape Navigator.

Java applets embedded in HTML pages can be used to construct the main interface of the experiment. Java is a natural choice on the client side because of its flexibility in GUI design, convenient network programming capability, and platform independence. Platform independence, the last feature, is most significant since it allows the same applet program to run on client machines with different platforms. To use Java, the codes must be compiled into applets and embedded into the HTML file.

Figure 3.1. Instrument panels for the frequency modulation experiment.

Figure 3.2. Instrument panel for the oscilloscope experiment.

Unlike Java, JavaScript is a scripting language, which is a subset of programming languages. JavaScript code can be included directly in HTML documents and can be easily modified. Typically, scripting languages are software extensions or operating system extensions, unlike true programming languages that can create compiled self-contained applications. PERL, VBScript, and JavaScript are examples of scripting languages, while Java, C, C++, and Pascal are examples of true programming languages. Scripting languages are in general less powerful but also easier to learn. A higher-level language such as Java, for instance, has a very steep learning curve and requires substantial involvement to master. JavaScript is actually descended from Java, but it was developed by Netscape as a means of bringing the simplicity of scripting language to a wider audience.

To create and realize a realistic instrument panel, it is necessary to mix codes in Java, JavaScript, and HTML together. Although the processes or functions written in JavaScript can always be rewritten in Java, the former may be preferred in some places for the sake of simplicity and efficiency in development.

In the following sections and the next two chapters, we will describe in detail, step-by-step, how a user-friendly and vivid panel can be created from scratch. In our discussion, we will use the GUIs of the frequency modulation and oscilloscope experiments as examples.

3.2 Frequency Modulation Experiment

Before we proceed, we will present some background knowledge on the instruments in the frequency modulation experiment that will be relevant to the design of the GUI and the instrument panels on the client. Essentially, the experiment makes use of a spectrum analyzer and some other instruments so that the user can investigate the spectrum of a frequency-modulated signal generated from a circuit board.

3.2.1 Circuit Board

Figure 3.3 shows the circuit diagram of the circuit board used for generating the frequency-modulated signal in the experiment. The circuit to the left of the dashed line is the frequency modulator, while that to the right corresponds to a second-order low-pass filter.

On the client GUI under the Internet Explorer browser, the circuit board is realized as shown in Figure 3.4. Note that a picture of the actual circuit board in the laboratory is used. The important terminals or components, including the input and output terminals, a test point, and a variable resistor, are marked by dashed circles in the figure.

The input terminal and socket in circle 1 corresponds to the input or point 1 in Figure 3.3. The knob in circle 2 is the variable resistor R_v in Figure 3.3. The terminal in circle 3 is a test point and corresponds to point 3 of Figure 3.3. The terminal in circle 4 is the ground terminal. By measuring the voltage difference between terminals 3 and 4, the voltage at pin 5 of the voltage-controlled oscillator LM566 can be

found. The sockets under both circles 5 and 6 correspond to point 5 of Figure 3.3. Through socket 5, the output signal can be fed to a frequency counter to be measured. Through socket 6, the same signal can be sent directly to a spectrum analyzer to be analyzed.
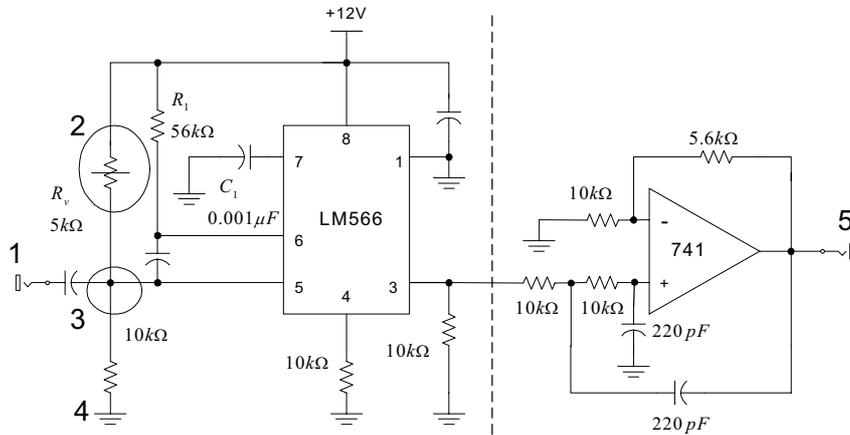

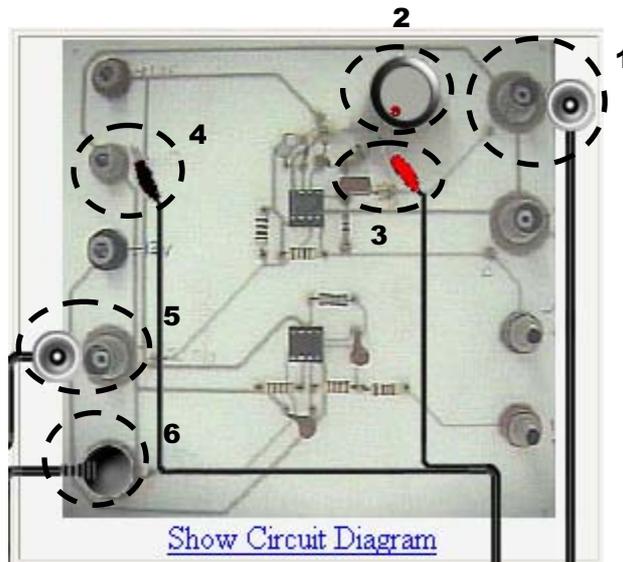**Figure 3.3.** Circuit diagram for the frequency modulation experiment.

Figure 3.4. Instrument panel for the circuit board.

3.2.2 Spectrum Analyzer

Figure 3.5 shows the HP 8590L spectrum analyzer [28] used in the frequency modulation experiment. It is a full-featured analyzer that can operate from 9 kHz to 1.8 GHz with an amplitude range of −115 dB to +30 dB. Regardless of whether it is manually or remotely operated, the analyzer has more than 200 functions, including marker types such as marker delta, marker peak search, and up to four on-screen markers. With the option of a GPIB interface, the instrument can be controlled by a computer.

On the Internet Explorer GUI, the spectrum analyzer is realized as shown in Figure 3.6. Since the actual instrument has many features but only a few important ones are relevant for the experiment, a simplified front panel is provided in the GUI. Another reason for having such a design is that the users or students will not be overwhelmed by the complexity of the instrument and yet will be able to get a good, realistic feeling of using a real, expensive spectrum analyzer. In terms of feasibility, it is in fact quite straightforward for the Web-based instrument to have exactly the same function as the real equipment if necessary. The controls and components on the instrument panel of the spectrum analyzer are described below [28].
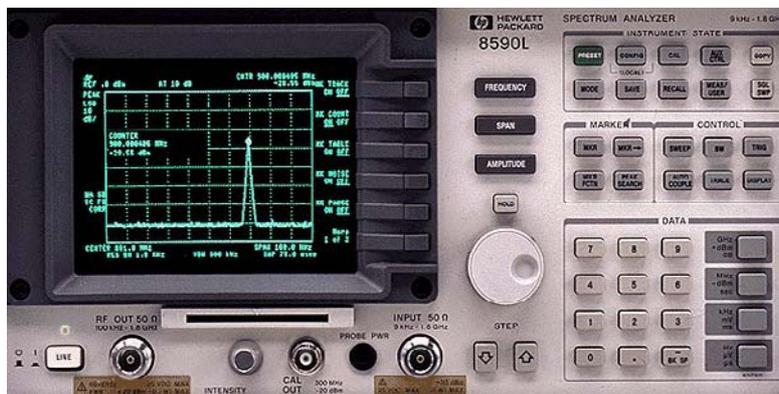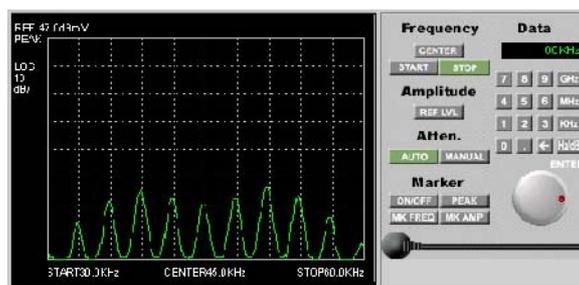

Figure 3.5. HP 8590L spectrum analyzer.

Figure 3.6. Instrument panel for the spectrum analyzer.

- **Buttons**

  There are five groups of buttons on the panel:

  o   Frequency Buttons

CENTER	Activate the center-frequency function to set the frequency at the center of the display.
START	Set the start frequency or that corresponding to the leftmost position of the display.
STOP	Set the stop frequency or that corresponding to the rightmost position of the display.

  o   Amplitude Button

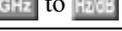 | | |
  |---|---|
  | REF LVL | Set the reference level, which corresponds to the power in dBm at the top topmost position of the display. |

  o   Attenuation Buttons

AUTO	Couple the attenuation to the reference level. The spectrum analyzer will attenuate the input signal power accordingly.
MANUAL	Set the input attenuation in 10 dB increments.

  o   Marker Buttons

ON/OFF	Turn the marker on or off.
PEAK	Position the marker at the peak of the spectrum.
MK FREQ	Specify the frequency of the active marker.
MK AMP	Get the spectrum value at the marker position.
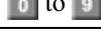
  o   Data Buttons

0 to 9	Number keys.
.	Decimal point.
⇐	Backspace or negative sign.
GHz to Hz/dB	Unit keys.

The data buttons allow the entry of exact values, which may include a decimal point, for many of the spectrum analyzer functions. All numerical entries must be terminated with a unit key, which will activate the instrument to accept the values and make corresponding changes in the operation of the instrument.

- **Knob**

  The knob allows changes to be made to parameters such as the center frequency, reference level, and marker position in an incremental continuous or discrete manner. Clockwise rotation of this control corresponds to an increase in value. To change values in a continuous manner, the extent of the change or resolution is determined by the range of the function selected. The speed at which the knob is turned, however, does not affect the rate at which values are changed. Through the GUI on the client side, the turning of the knob is achieved by dragging the mouse pointer when it is positioned next to the red dot of the control.

3.2.3 Frequency Counter

Figures 3.7 and 3.8 show the frequency counter used in the experiment and the corresponding GUI version under the Internet Explorer browser, respectively. The instrument counter is capable of measuring frequencies up to 225 MHz [29]. Frequency and time-interval resolutions are 10 digits in one second and 500 picoseconds, respectively. Programmable control is performed via GPIB, and this provides the user with a speed of 200 measurements per second.

In the experiment, this instrument is used to measure the frequency of the carrier signal for the purpose of calibrating the voltage-controlled oscillator (VCO) before the latter is used for frequency modulating an input message signal.
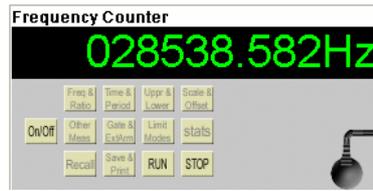

Figure 3.7. HP 53131A universal counter.

Figure 3.8. Instrument panel for the frequency counter.

3.2.4 Signal Generator

The signal generator for producing the input message signal and its corresponding Internet Explorer GUI version are shown in Figures 3.9 and 3.10, respectively. The instrument is a versatile arbitrary waveform generator, can operate over the frequency range from 1 mHz to 10 MHz, and is capable of giving a 1 mV to 10 V peak-to-peak output into a 50 Ω load [30]. In the experiment, the instrument is remotely operated via the IEEE 488.2 bus and is SCPI-compatible.
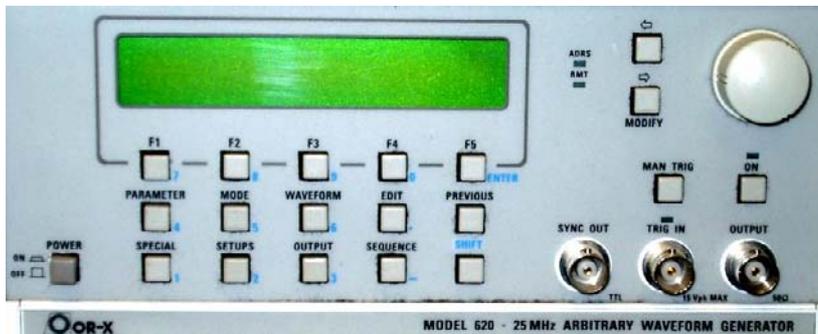

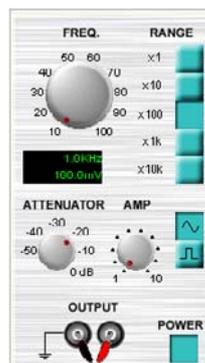**Figure 3.9.** OR-X Model 620 arbitrary waveform generator.

Figure 3.10. Instrument panel for the signal generator.

3.2.5 Voltmeter

The last instrument used in the experiment is a standard remote-controlled voltmeter for measuring the voltage of the VCO on the circuit board. The instrument is controlled and its reading is captured through the IEEE 488.2 bus. In the GUI under Internet Explorer, it is simply realized as shown in Figure 3.11. Basically, once the power button is on, the voltage applied to its input will be continuously measured and displayed.
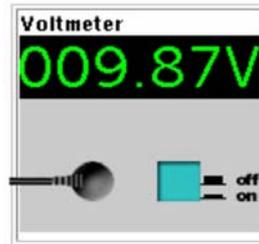

Figure 3.11. Instrument panel for the voltmeter.

3.3 Java and OOP

3.3.1 Java Applet

The instrument panels described above are created using Java applets. In general, Java can be used to create either a stand-alone application or an applet [31]. The former does not need to be embedded in an HTML document and can be run without using any browser.

In our Web-based laboratories, most of the Java programs are for the creation of applets. Put simply, an applet is a part of a Web page, just like an image or a line of text in a document. Thus, similar to how a browser looks after the display of an image in an HTML document, a Java-enabled browser is needed to locate and run an applet. When a Java-capable Web browser loads an HTML document, the Java applet is also loaded and executed. Using applets, one can perform tasks such as having animated graphics, games, or utilities executed from a Web page downloaded from the Internet.

The development of Web-based laboratories will involve many languages, such as Java, JavaScript, VBScript, Perl, MATLAB, and LabVIEW. However, Java is perhaps the most important in dealing with situations, especially the user interface, on the Internet.

To create a Java program, a text editor can be used to create a Java language source code file. After saving the source code with a `.java` file extension, the saved file can then be compiled into its byte-code format, creating another file with

a `.class` extension. It is the `.class` file that the interpreter loads and executes. Because the byte-code files are fully portable between operating systems, they can be executed on any system that has a Java interpreter.

3.3.2 Class

The creation of classes in the context of OOP (object-oriented programming) is crucial in the development of Web-based laboratories. However, before we describe in detail how this can be done for the client GUI in the next two chapters, we will first present here some basic concepts and principles on classes.

Under OOP, a class is essentially a template for an object. An example is the common data type integer, or `int`, which is predefined in Java and practically all other programming languages. When the need arises for a new class to be created, however, the characteristics of the class must be specified. Specifically, a class associated with the display of a button can be defined by using the `class` keyword along with an appropriate class name, as follows:

```
class imgButton
  {
  }
```

Despite having an empty body, these few lines actually create a complete class. If they are saved in a file called `imgButton.java`, compilation to a `.class` file can be carried out and execution will be possible, even though nothing useful will result.

Nevertheless, one can still create an object `powerButtt` from the defined example class. To do this, we can use either

```
ImgButton powerButtt = new imgButton();
```

or

```
ImgButton powerButtt;
```

The `ImgButton` is empty and useless. Both data fields and methods will have to be specified for the class if it is to serve any purpose. The declaration of four integer fields for storing the size of any button created from the class `imgButton` can be achieved as shown in the following example:

```
class imgButton
  {
     int left, top, width, height;
  }
```

As specified, the data fields are by default accessible only by methods in the same class. However, these can be changed by using the `public`, `protected`, and `private` keywords. A public data field can be accessed by any part of the

program, inside or outside of the class in which the field is defined. A protected data field can only be accessed from within the class or from within a derived class or subclass. A private data field is not accessible even for a derived class.

In addition to adding relevant data fields, appropriate methods or functions must also be provided to operate on the fields if the defined class is to serve any specific purpose. Of all the methods, the most special one is the constructor. This is a method with the same name as the class and enables an object to initialize itself as it is created.

Figure 3.11 shows the constructor of the `imgButton` class. The constructor assigns input values to the data fields `left` and `top` and initiates the button image, including its pushed-down and released-up status. As shown, the constructor starts with the `public` keyword. This is necessary and important so that an object belonging to this class can be created anywhere in the program. When an object is created, its constructor is actually called. After the `public` keyword and the name of the constructor or class, the constructor's arguments are specified in parentheses. When an `imgButton` object is created, these arguments must be supplied. For example, a new button may be created by using

```
nButt = new imgButton(ButtUpName,ButtDownName,140,290);
```

which will pass all four arguments to the constructor for processing.

The creation of other methods is similar to that for the constructor. Also, as for data fields, an appropriate type of access must be provided. Specifically, methods callable from outside the class should be defined as `public`, methods callable only from the class and its derived classes should be defined as `protected`, and methods callable only from within the class should be declared as `private`.

ImgButton: Constructor

```
public imgButton(String s_upImg, String s_downImg, int x, int y)
 {
     left = x;
     top = y;

     try{
         InitImage(new URL(s_upImg), new URL(s_downImg));
        }
     catch(MalformedURLException e)
        {
         System.out.println("imgButton.class: MalformedURL");
        }
 }
```

Figure 3.11. Constructor of the `imgButton` class.

3.3.3 Class Creation

Like other OOP languages, one can create a new class from an existing class in Java through the principle of inheritance. As an example, using the `extends` keyword, we can create a new `imgButton` subclass from the `Component` superclass by using the following:

```
public class imgButton extends Component
```

The `Component` class is one that has been created in Java for the convenience of programmers. There are numerous basic classes such as this in Java, a common feature of which is that they contain all the basic functionality one may need for various purposes.

The creation of a new subclass enables one to inherit all the data fields and methods that have been created for the superclass without the need to develop new codes. In addition, one can also specify new data fields and methods for the new class that will be special or new or not supplied by the superclass. Also, it is possible to override or specify a new version of a method that forms part of the superclass.

http://www.springer.com/978-1-85233-837-4

Creating Web-based Laboratories
Ko, C.C.; Chen, B.M.; Chen, J.
2004, XVI, 300 p., Hardcover
ISBN: 978-1-85233-837-4