Con las indicaciones desarrolladas en este artículo y utilizando la Interfaz serie-paralelo LX.1127 con la Tarjeta experimental LX.1128 se pueden realizar eficaces y sencillos programas en JAVA para controlar el puerto serie de un ordenador.

Programar en JAVA

JAVA Y EL PUERTO SERIE

No cabe la menor duda de que el lenguaje de programación **Java** ha tenido, tiene, y seguramente tendrá, una **difusión enorme**. Basta pensar, por ejemplo, en los **teléfonos móviles**, para los que se han desarrollado centenares de **juegos** y **aplicaciones**.

El éxito de Java está basado en que permite realizar **aplicaciones independientes** del **hardware** y que tiene la potencia de la **programación orientada a objetos**, especialmente útil para realizar **interfaces gráficas**.

Mediante este artículo queremos mostrar lo fácil, e incluso divertido, que resulta la programación en **Java** para **controlar** el **puerto serie** de un ordenador utilizando como ejemplo nuestra interfaz serie-paralelo **LX.1127** y la tarjeta experimental **LX.1128**.

INTRODUCCIÓN

El lenguaje **Java** lo creó **James Gosling**, junto a su equipo, en los laboratorios de **Sun Microsystems**. Su sintaxis deriva del lenguaje **C++**. Fue liberado oficialmente en **1995**.

Una de sus características más importantes es que se trata de un **lenguaje multiplataforma**, esto es, una vez escrito un programa puede ser ejecutado en **cualquier máquina**, independientemente de su **sistema operativo**: Windows, Mac OS X, Linux, Windows CE, Palm OS, etc.

Seguramente la primera pregunta que surge es cómo se desarrolla, por ejemplo, un programa en un entorno **Windows** y se puede ejecutar en un **teléfono móvil** o en un PC con **Linux**.

La respuesta reside en la **J**ava **V**irtual **M**achine

(**JVM**), esto es, la **máquina virtual Java**. Una vez realizado un **programa fuente** en **Java** se crea un archivo .**class**, que **no** es **ejecutable directamente** ya que no está realizado en el lenguaje de ninguna máquina concreta.

Esto hace a **Java** completamente **independiente** de la **máquina**, ya que el programa es el mismo para **cualquier dispositivo**, como por ejemplo un PC con su sistema operativo.

Para ejecutar un archivo .**class** independiente del hardware hace falta una **JVM** que **interprete** el **código** del **archivo** y **genere** el **código ejecutable** para la máquina donde reside.

Otra peculiaridad de **Java** es que se ha desarrollado para utilizar la moderna técnica de **programación orientada a objetos**. Detallar está técnica llevaría decenas de páginas y conllevaría muchas horas. Veamos la idea principal con un sencillo ejemplo.

La programación orientada a los objetos representa entidades definidas: Los **objetos**.

Por ejemplo, cada ser humano tiene **propiedades** representadas por **valores** (color de ojos, número de dedos, etc.) y **cualidades** (carácter, pensamiento, etc.). Una agrupación de **propiedades** y **cualidades** definen un **objeto**, en nuestro ejemplo el hombre.

La programación orientada a objetos consiste en **definir** y **administrar** las **propiedades** y las **cualidades** de los **objetos** para desarrollar un **proceso**. Aunque puede que esto resulte muy abstracto con la lectura del artículo quedará muy claro.

Concluimos esta breve introducción con una pequeña curiosidad. **Java** es, además, un tipo de **café** de la homónima **isla indonesia**. El nombre del lenguaje fue elegido por **Gosling** y **Van Hoof**, un colaborador suyo, porque consumían mucho este café ... el **logotipo oficial** es una taza de café humeante.

Antes de iniciar la descripción de los procedimientos aclaramos que nosotros hemos elegido como **plataforma de desarrollo** un ordenador personal con sistema operativo **Windows**. Para poder utilizar los programas aquí expuestos en otras plataformas hay que seguir los procedimientos descritos detalladamente en los correspondientes sitios oficiales.

el PUERTO SERIE

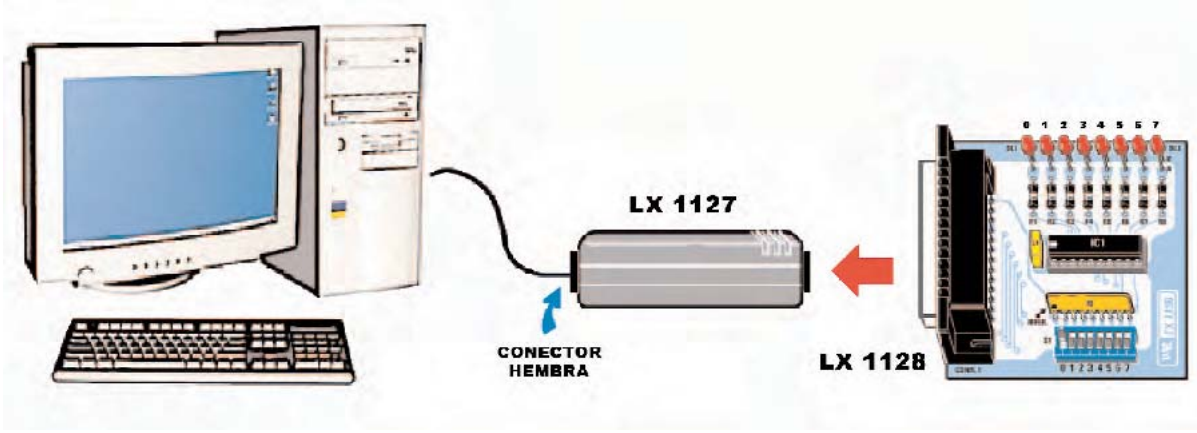

Fig.1 Conexión de la tarjeta experimental LX.1128 al puerto serie de un ordenador a través de la interfaz serie/paralelo LX.1127. Quienes no dispongan en su ordenador de un puerto serie pueden utilizar un conversor estándar USB-Serie.

INSTALACIÓN DE JAVA Y RXTX

Para poder programar en **Java** y utilizar el **puerto serie** es necesario instalar algunos programas.

Comencemos por el **entorno de desarrollo** y la **JVM**. A fecha de escritura de este artículo **Sun Microsystems** tiene disponible la versión **J2SE 6.0 update 4**. Es importante que la versión sea **J2SE 6.0**, si el número de actualización no es la 4 no supone ningún problema.

Para descargar el entorno de desarrollo completo hay que conectarse a la **Web oficial** de **Java** (**http://www.java.sun.com**). Una vez abierta la página principal hay que acceder al menú **Download** y, a continuación, seleccionar **Java SE (JDK) 6 update 4**.

Al hacer click en **Download** se abrirá una ventana de selección de **plataforma** (**Linux**, **Solaris**, **Windows**). Hay que seleccionar el sistema operativo, **Windows** en nuestro caso, y marcar la casilla de aceptación de **licencia**.

Una vez terminada la descarga hay que **instalar** el programa, operación que se realiza de forma muy sencilla siguiendo los **pasos** indicados en **pantalla**.

El entorno de desarrollo se instala de forma predeterminada en la carpeta **C:\Archivos de programa\Java**. Dentro de esta carpeta hay dos **subcarpetas**:

C:\Archivos de programa\Java\jdk1.6.0_04
C:\Archivos de programa\Java\jre1.6.0_04

La carpeta **JDK** se utiliza para el **desarrollo** del **código fuente** de los programas mientras que la carpeta **JRE** se utiliza para **ejecutarlos**. En efecto, sus acrónimos corresponden a **J**ava **D**evelopment **K**it (**JDK**) y **J**ava **R**untime **E**nvironment (**JRE**). Es muy importante no confundir estas carpetas.

De ahora en adelante cuando escribimos de forma genérica **C:\path_java** nos referiremos a **C:\Archivos de programa\Java\jdk1.6.0_04**.

En segundo lugar hay que instalar **RXTX**. Este aplicativo permite gestionar la **comunicación** del **puerto serie**.

Para facilitar esta tarea hemos puesto los archivos necesarios en la sección **DESCARGAS** de nuestra **página Web** (**www.nuevaelectronica.com**).

El fichero descargado autodescomprimible **JavaNE.EXE** contiene **tres archivos**:

- Los archivos **rxtxParallel.dll** y **rxtxSerial.dll** deben copiarse en el directorio **C:\Archivos de programa\java\jdk1.6.0_04\jre\bin**

- El archivo **RXTXcomm.jar** debe copiarse en el directorio **C:\Archivos de programa\ java\ jdk1.6.0_04\jre\lib\ext**

Para los sistemas **Linux** y **Mac OS X** el procedimiento es ligeramente diferente. Quienes trabajen en estos entornos pueden encontrar toda la documentación necesaria en el **sitio oficial**.

Aquí nos limitamos a señalar que el sitio oficial es **http://www.rxtx.org** y que se ha de descargar el archivo comprimido **rxtx2.1-7-bins-r2.zip** para trabajar con los sistemas **Linux** y **Mac OS X**.

Con estos pocos y sencillos pasos ya se pueden realizar programas que interaccionen con el puerto serie.

Los **ejemplos** que proponemos en este artículo han sido verificados, tanto en plataformas **Windows** como en plataformas **Linux**. Garantizamos su funcionamiento.

NOTA Para una mejor comprensión del código fuente con el kit **LX.1127** es aconsejable leer el artículo correspondiente a este kit (**revista Nº118**).

Principios de PROGRAMACIÓN JAVA

El código fuente de los ejemplos que aquí presentamos debe **copiarse literalmente** utilizando un **editor de texto** sencillo, como por ejemplo el **Bloc de notas** o el **Wordpad**.

Hay que tener presente que **cada línea** tiene que acabar con el carácter "**;**" (punto y coma) y que los **números** situados a la izquierda **no deben copiarse** (los utilizamos como **referencia** para localizar de forma precisa las diferentes líneas en la descripción de los programas).

Los **comentarios** están **precedidos** por los caracteres "**//**". Un comentario válido puede ser, por ejemplo, "**// MI COMENTARIO**".

Las **subrutinas** han de estar delimitadas por llaves "**{ }**".

También hay que tener presente que **Java diferencia** entre **letras mayúsculas** y **minúsculas**. Para Java la palabra "Casa" es diferente a "CASA", "casa", "Casa", etc.

Cada programa Java tiene que hacer corresponder el **nombre** de la **clase principal** con el **nombre del archivo** para que no se produzca un error durante la compilación. Por ejemplo:

```
import java.io.*;

public class MiPrograma {
// CÓDIGO
// CÓDIGO

}
```

El archivo que contiene este programa en código fuente tiene que llamarse obligatoriamente "**MiPrograma.java**", respetando los caracteres en mayúsculas y minúsculas.

Por último, para **mostrar mensajes** en **pantalla** se utiliza la instrucción Java:

"**System.out.println ("TEXTO");**" o bien "**System.out.print ("TEXTO");**"

Después de estas consideraciones sobre el lenguaje exponemos los programas ejemplo.

PROGRAMA 1: "SendData"

Las instrucciones correspondientes a las **líneas 1** a **6** le indican a la **JVM** las **librerías a utilizar** para ejecutar el programa. A continuación, en la **línea 7**, se declara **la clase principal** "**SendData**", que corresponde al nombre del archivo.

En la **línea 8** se encuentra la **declaración** "**main**": Todos los programas Java **inician** su ejecución en este punto. Las dos instrucciones siguientes indican que será utilizada la **lectura por consola**, en la práctica se definen las condiciones para realizar **entradas** desde **teclado**.

La **línea 13** es muy importante ya que determina el **puerto serie a abrir**. En el ejemplo hemos utilizado **COM5**.Esta línea ha de adaptarse al hardware utilizado, así quienes utilicen COM2 o COM3 tienen que ajustar este parámetro con los valores "COM2" o "COM3".

ATENCIÓN A causa del auge de **USB** los ordenadores de última generación tienen solo **un puerto serie** o, Incluso, **ninguno**. Existen **adaptadores USB-Serie estándar** que se pueden utilizar para conectar la interfaz **LX.1127** a un puerto USB del ordenador.

En caso de utilizar adaptadores USB-Serie la **numeración del puerto** suele depender del número de puertos USB presentes en el PC. Por ejemplo si se dispone de **4 puertos USB** el puerto serie será identificado como **COM5**.

NOTA En sistemas **Linux** el **puerto serie 1** (**COM1**) generalmente se identifica como **/dev/ttyS0**, el **puerto serie 2** (**COM2**) como **/dev/ttyS1**, etc.

Una vez definido el puerto serie el programa **controla** que **no** esté siendo utilizado por **otros programas**. Este control es fundamental para evitar sorpresas desagradables o un funcionamiento incorrecto del **LX.1127**.

Si se detecta que el **puerto** está libre se procede a **abrirlo** mediante la instrucción de la **línea 21** (**Open** devuelve un **identificador** para el puerto recién abierto). Mediante la instrucción de la **línea 22** se controla si ha sido abierto un **puerto serie**, ya que por error podríamos haber abierto una puerto de otro tipo.

Una vez verificado que estamos utilizando un puerto serie vamos a proceder a **definir** los **parámetros** necesarios para la **comunicación**: **Velocidad de transmisión**, número de **bits de datos**, **bits de parada** y **bits de paridad**. En nuestro caso, como se indica en el artículo de la revista **Nº118**, los datos son: **2400 baudios**, **8 bits de datos**, **1 bit de parada** y **ningún bit de paridad** (**línea 26**).

Una vez abierto un canal de comunicación nos preparamos para ejecutar un **ciclo** en el que se requiere introducir un **valor** incluido entre **0** y **255**, correspondiente al **peso** de los **diodos LED** a **encender**.

Programa SendData

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import gnu.io.CommPort;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
public class SendData {
	public static void main(String[] args) throws Exception {
		InputStreamReader in = new InputStreamReader(System.in);
		BufferedReader input = new BufferedReader(in);

		// Identifica el puerto serie a abrir
		  CommPortIdentifier portIdentifier =
CommPortIdentifier.getPortIdentifier("COM5");

		// Controla si el puerto está en uso
		if (portIdentifier.isCurrentlyOwned()) {
			System.out.println("Error: El puerto está en uso.");
		} else {

			// Abre el puerto serie
			CommPort commPort = portIdentifier.open("SendData", 2000);
			if (commPort instanceof SerialPort) {
				SerialPort serialPort = (SerialPort) commPort;

				// Ajusta los parámetros de transmisión
				serialPort.setSerialPortParams(2400,
SerialPort.DATABITS_8, SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
				OutputStream out = serialPort.getOutputStream();
				int peso = 255;

				// Ciclo envío de datos
				while (peso > 0) {
					System.out.print("Introducir el valor para
enviar a los LED(-1 para salir):");
					peso = Integer.parseInt(input.readLine());
					if (peso > 255) {
						System.out.println("Valor demasiado
grande.");
						System.out.println("Introducir un número
entre 0 y 255.");
					} else {
						// Selecciona el puerto
						out.write((byte) 0);
						// Selecciona TX
						out.write((byte) 255);
						// Transmisión al puerto
						out.write((byte) 4);
						// Dato a transmitir
						out.write((byte) peso);
					}
				}
				System.out.println("¡¡Adiós!!");
			} else {
				System.out.println("Error: Solo se aceptan puertos
serie.");
			}
		}
		  System.exit(1);

	}
}
```

En las instrucciones de las **líneas 39-41-43-45** se puede observar claramente como se mandan los datos al puerto serie. Según las indicaciones de la interfaz **LX.1127** elegimos el **puerto** del **kit** a utilizar (**39**), seleccionamos función de **transmisión** (**41**), decidimos **dónde transmitir** (**43**) y mandamos los **datos** (**45**).

El resto del programa son simples **mensajes informativos** o de **error**.

Una vez escrito el **código fuente** mediante un **editor** y **guardado** con el nombre adecuado, que tiene que ser idéntico al nombre de la **clase principal** (**SendData**) con **extensión .java**, hay que **compilarlo**.

Se puede guardar en cualquier carpeta, por ejemplo en una nueva carpeta denominada **C:\ProgramasJava**, así la ruta y el nombre serían **C:\ProgramasJava\SendData.java**.

Para **compilarlo** hay que abrir un **intérprete de comandos MS-DOS**.

Para lanzar un **intérprete de comandos DOS** en sistemas **Windows** hay que pulsar en **Inicio**, **Programas**, **Accesorios**, **Símbolo del sistema**. Se abrirá una ventana con contenido texto con una indicación similar a esta:

C:\WINDOWS>

Para cambiar de directorio hay que teclear:
CD "\Archivos de programa \java \jdk1.6.0_04\bin\"

En pantalla aparecerá: **C:\Archivos de programa \java \jdk1.6.0_04\bin\:>**

Hay que teclear:
Javac C:\ProgramasJava\SendData.java

El compilador crea el archivo **SendData.class** que corresponde al **archivo ejecutable**.

Una vez **compilado** el programa se puede lanzar su **ejecución**. Hay que posicionarse en el directorio **bin** del entorno tecleando:
CD "\Archivos de programa \java\ jdk1.6.0_04\bin\"

En pantalla aparecerá: **C:\Archivos de programa\java\jdk1.6.0_04\bin\:>**

Ahora hay que teclear:
Java C:\ProgramasJava\SendData

NOTA En Java para ejecutar los programas **no** es **necesario** añadir la extensión **.class**.

PROGRAMA2: "ReceiveData"

Como se puede deducir el segundo programa Java que presentamos permite **leer** los **datos** mandados hacia el ordenador por la interfaz **LX.1127** mediante el **puerto serie**.

No vamos a detallar las **primeras líneas** ya que son **casi idénticas** a las del programa **SendData**. La parte específica de este programa corresponde a las **líneas 32** a **41**.

Una vez seleccionado el **puerto** (**línea 32**), seleccionada **recepción** (**línea 35**) y el **puerto del kit** (**línea 38**), nos preparamos para recibir datos. La **lectura** se realiza mediante la instrucción **read** de la **línea 41**.

Llegado este punto **convertimos** el valor capturado en **formato legible** (**línea 44**) y **mostramos** el contenido (**líneas 47-48**).

Para **compilar** y **ejecutar** el programa cambiamos al directorio **bin** y tecleamos las órdenes:

CD "\Archivos de programa\ java\ jdk1.6.0_04\bin\"

Javac C:\ProgramasJava\ReceiveData.java

Java C:\ProgramasJava\ReceiveData

Para **compilar** el programa SendData con un único comando teclear:

C:\Archivos de programa\java\jdk1.6.0_04\bin\javac.exe C:\ProgramasJava\SendData.java

Para **ejecutar** el programa SendData con un único comando teclear:

C:\Archivos de programa\java\jdk1.6.0_04\bin\java.exe C:\ProgramasJava\SendData

Programa ReceiveData

```
import gnu.io.CommPort;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import java.io.InputStream;
import java.io.OutputStream;

public class ReceiveData {

	public static void main(String[] args) throws Exception {

		// Identifica el puerto serie a abrir
		  CommPortIdentifier portIdentifier =
CommPortIdentifier.getPortIdentifier("COM5");

		// Controla si el puerto está en uso
		if (portIdentifier.isCurrentlyOwned()) {
			System.out.println("Error: El puerto está en uso.");
		} else {
			// Abre el puerto serie
			CommPort commPort = portIdentifier.open("ReceiveData", 2000);

			if (commPort instanceof SerialPort) {
				byte[] buffer = new byte[1];
				int c=0;
				SerialPort serialPort = (SerialPort) commPort;

				// Ajusta los parámetros de recepción
				serialPort.setSerialPortParams(2400,
SerialPort.DATABITS_8, SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
				OutputStream out = serialPort.getOutputStream();
				InputStream dipSwitch = serialPort.getInputStream();

				// Selecciona el puerto
				out.write((byte)1);

				// Selecciona RX
				out.write((byte)0);

				// Recepción del puerto
				out.write((byte)3);

				// Recibe el dato
				dipSwitch.read(buffer);

				// Convierte el dato a decimal
				c=255-(0xFF&((char)buffer[0]));

				// Imprime el dato recibido
				System.out.println("Peso ="+c);
				System.out.println("¡¡Adiós!!");
			} else {
				System.out.println("Error: Solo se aceptan puertos
serie.");
			}
		}
		  System.exit(1);

	}

}
```

PROGRAMA 3: "GuiLX1127"

Hasta ahora hemos presentado dos sencillos programas que se ejecutan desde un **prompt** y muestran las salidas como **valores** en **pantalla**. Se puede mejorar su forma de utilización y su aspecto interaccionando con el ratón, botones, ventanas, etc., es decir trabajar con una **interfaz gráfica**.

Precisamente una de las características más apreciadas de **Java** es la **facilidad** de construir **G**raphical **U**ser **I**nterfaces (**GUI**), esto es, ventanas, botones, etiquetas y todo lo necesario para crear **interfaces gráficas**.

Además, gracias a ser un entorno **multiplataforma**, una vez construida la GUI se puede procesar con **diferentes sistemas operativos**.

En las imágenes de las Figs.2-4 se muestra el aspecto del programa que vamos a describir en entornos **Windows**, **Mac OS X** y **Linux**.

NOTA El código fuente de **GuiLX1127**, debido al espacio asignado al artículo, se limita a la creación de una **ventana** con **8 botones** y **etiquetas descriptivas**. Quienes deseen profundizar pueden visitar el sitio oficial de Java y consultar los ejemplos propuestos.

La **interfaz gráfica** se **define** mediante la función "**createAndShowGUI ()**" (**línea 104**).

En la **línea 94** se encuentra la **declaración** de la función **main**. Vamos a analizarla.

En primer lugar definimos un **frame** (**ventana**) donde se alojarán nuestros componentes (**botones** y **etiquetas**).

Algunas operaciones son **estándar**, como el **cierre** (**línea 97**) y el **dibujo final** (**líneas 101-102**). En la **línea 99** se llama al método "**addComponentsToPane**" que **dibuja** la **GUI**.

AddComponentsToPane se **define** en la **línea 44**. En nuestro caso cada ventana puede imaginarse como una **rejilla** construida por líneas y columnas dentro de la cual insertaremos nuestros **elementos**.

La primera rejilla que definimos tiene **3 líneas** y **0 columnas**, es la rejilla principal (**líneas 48-49**). Luego creamos una segunda rejilla de **0 líneas** y **8 columnas** utilizada para alojar los 8

Fig.2 El lenguaje de programación Java permite construir con mucha facilidad interfaces gráficas de usuario (GUI - Graphical User Interface). Aquí se muestra el aspecto de la interfaz definida en el programa GuiLX1127 bajo sistema operativo Windows.

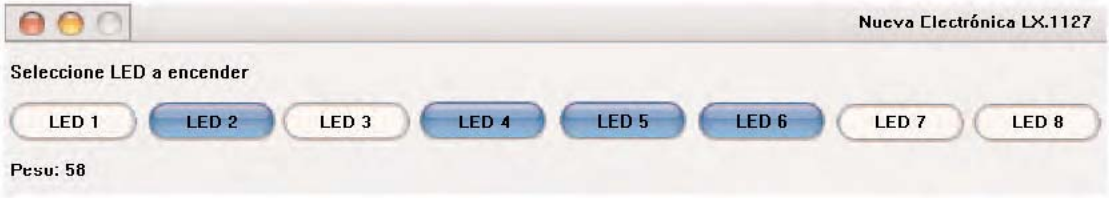

Fig.3 Aspecto de la interfaz gráfica definida en el programa GuiLX1127 en un entorno MAC OS X.

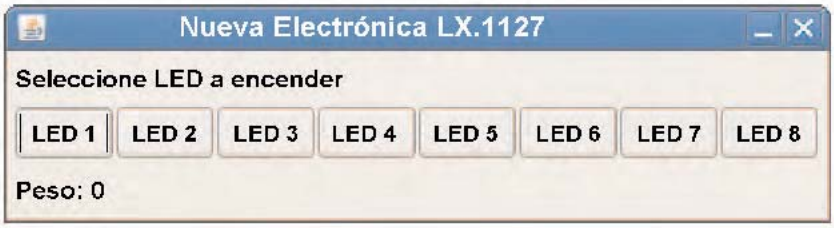

Fig.4 En un sistema operativo Linux la ejecución del programa GuiLX1127 es idéntica mostrando este aspecto. En todos los sistemas se muestran los botones y las etiquetas de forma análoga, con la estética propia de cada entorno.

Programa GuiLX1127

```
1  import gnu.io.CommPort;
2  import gnu.io.CommPortIdentifier;
3  import gnu.io.SerialPort;
4  import java.awt.*;
5  import java.awt.event.*;
6  import java.io.OutputStream;
7  import javax.swing.*;
8  public class GuiLX1127 extends JFrame implements ActionListener {
9
10         private static final long serialVersionUID = 1L;
11         private int peso = 0;
12         private boolean[] boolButton = { false, false, false, false, false, false,
   false, false };
13         private JLabel pesoLabel=null;
14         private OutputStream out = null;
15
16         public GuiLX1127(String name) throws Exception {
17                 super(name);
18                 setResizable(false);
19
20                 CommPortIdentifier portIdentifier =
   CommPortIdentifier.getPortIdentifier("COM1");
21
22                 if (portIdentifier.isCurrentlyOwned()) {
23                         System.out.println("Error: El puerto está en uso.");
24                 } else {
25                         CommPort commPort = portIdentifier.open("GuiLX1127", 2000);
26                         if (commPort instanceof SerialPort) {
27                                 SerialPort serialPort = (SerialPort) commPort;
28                                 serialPort.setSerialPortParams(2400,
   SerialPort.DATABITS_8, SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
29                                 out = serialPort.getOutputStream();
30                                 sendData();
31                         } else {
32                                 System.out.println("Error: Solo se aceptan puertos
   serie.");
33                         }
34                 }
35
36         }
37         private void sendData() throws Exception {
38                 out.write((byte) 0);
39                 out.write((byte) 255);
40                 out.write((byte) 4);
41                 out.write((byte) peso);
42         }
43
44         public void addComponentsToPane(final Container pane) {
45
46                 // Crea un panel principal y uno
47                 // con los botones
48                 final JPanel mainPanel = new JPanel();
49                 mainPanel.setLayout(new GridLayout(3, 0));
50                 JPanel controls = new JPanel();
51                 controls.setLayout(new GridLayout(0, 8));
52
53                 // Crea 8 botones
54                 for (int b = 1; b < 9; b++) {
55                         JToggleButton button = new JToggleButton("LED " + b);
56                         button.addActionListener(this);
57                         controls.add(button);
58                 }
```

```
59              // Asigna etiqueta y botones al panel principal
60              mainPanel.add(new JLabel("Seleccione LED a encender:"));
61              mainPanel.add(controls);
62              pesoLabel=new JLabel("Peso: " + peso);
63              mainPanel.add(pesoLabel);
64              pane.add(mainPanel);
65      }
66
67      public void actionPerformed(ActionEvent e) {
68              // Ciclo para controlar qué botón
69              // está pulsado
70              for (int b = 1; b < 9; b++) {
71                      if (("LED " + b).equals(e.getActionCommand())) {
72                              // Control para ver si el botón
73                              // ya estaba pulsado
74                              if (boolButton[b - 1]) {
75                                      peso = peso - (int) (Math.pow(2, (b 1)));-
76                              } else {
77                                      peso = peso + (int) (Math.pow(2, (b - 1)));
78                              }
79                              boolButton[b - 1] = !boolButton[b - 1];
80                              b = 9;
81                      }
82              }
83
84              // Asigna etiqueta con el nuevo peso
85              pesoLabel.setText("Peso: " + peso);
86
87              // Transmite el dato al puerto
88              try {
89                      sendData();
90              } catch (Exception e1) {
91                      e1.printStackTrace();
92              }
93      }
94      private static void createAndShowGUI() throws Exception {
95              // Crea la ventana principal
96              GuiLX1127 frame = new GuiLX1127("Nueva Electrónica LX.1127");
97              frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
98              // Asigna el contenido de la ventana
99              frame.addComponentsToPane(frame.getContentPane());
100             // Visualiza la ventana
101             frame.pack();
102             frame.setVisible(true);
103     }
104     public static void main(String[] args) {
105             //Usa el tema del sistema operativo
106             try {
107     UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
108             } catch (Exception ex) {
109                     ex.printStackTrace();
110             }
111             javax.swing.SwingUtilities.invokeLater(new Runnable() {
112                     public void run() {
113                             try {
114                                     createAndShowGUI();
115                             } catch (Exception e) {
116                                     e.printStackTrace();
117                             }
118                     }
119             });
120     }
121 }
```

botones en sentido horizontal (ver Fig.2).

En la imagen se pueden observar las **3 líneas principales**. La primera y la última contienen solamente **etiquetas** (**label**), mientras que la segunda incluye los **botones**.

Ahora hay que crear los **8 botones**. La instrucción que **crea** un botón está en la **línea 55**, mientras que la instrucción de la **línea 57 coloca** el botón recién creado en la rejilla.

La **línea 56** merece mención aparte. Hasta ahora nos hemos preocupado solo de dibujar un botón y colocarlo en una posición concreta, pero no le hemos asignado ninguna **función** cuando se **hace click** sobre él. La **línea 56** se ocupa de esta operación.

Acabado el ciclo hay que colocar todo en la ventana. Añadimos al panel principal la **etiqueta** descriptiva ("**Seleccionar LED a encender**") en la **línea 60**, **añadimos** los **botones** recién creados y **añadimos** la **etiqueta** que indica la **suma** de los **pesos** ("**Peso: 0**").

Una vez finalizado el panel principal lo adjuntamos a la **ventana** creada en la **línea 96**. Llegado este punto nuestra ventana está lista.

La última función por analizar es "**actionPerformed ()**" presente en la **línea 67**, función que se ocupa de **interaccionar** con los **botones** y la **etiqueta** de los **pesos**. Cada vez que se **pulsa un botón** se determina **cuál** se ha accionado. Una vez determinado se controla si hay que **incrementar** (**línea 77**) o **decrementar** (**línea 75**) **su peso**.
A continuación se **actualiza** la variable global "**peso**" y se **visualiza su valor** mediante la instrucción de la **línea 85**. Por último se invoca **sendData** para **encender** los **diodos LED**.

El resto de las instrucciones, como se puede percibir fácilmente, son similares a las del programa "**SendData**".

Las **líneas 38-41** mandan los **datos** al **puerto serie** mientras que las **líneas 20-34 programan** el puerto COM según las directivas descritas anteriormente. El resto son **declaraciones** de **variables globales**.

Como habíamos adelantado la creación de un programa con una **ventana gráfica** que permite **gestionar** los datos de un **puerto serie** ha sido **muy simple**.

Para la **compilación** y la **ejecución** procedemos de forma similar a la expuesta en los casos anteriores:

CD "\Archivos de programa\ java\ jdk1.6.0_04\bin\"

Javac C:\ProgramasJava\GuiLX1127.java

Java C:\ProgramasJava\GuiLX1127

EJERCICIO

En la Fig.5 se expone una **GUI** que muestra las **posiciones** (**valor binario**) del **dipswitch**.

Para adquirir experiencia os proponemos realizarla mediante unos sencillos ejercicios utilizando el código del ejemplo anterior:

1. Cambiar la etiqueta "**Seleccionar LED a encender**" por la etiqueta "**Presionar para conocer los valores**".

2. Sustituir los **8 botones** por **uno** solo con el texto "**Leer dipswitch**".

3. Una vez accionado el botón hay que **leer** los **datos** del **puerto serie** y **visualizarlos** en la etiqueta "**Peso: xxx**".

Todo lo necesario para su realización se encuentra expuesto en los **3 programas** presentados en el artículo.

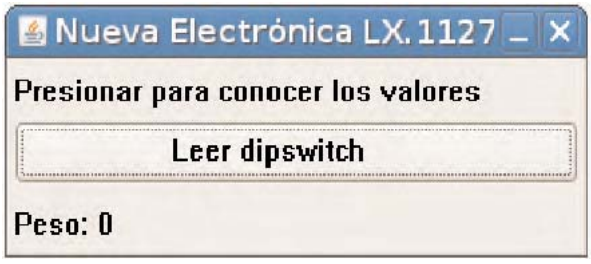

Fig.5 Utilizando el código fuente de los programas descritos en el artículo se puede crear fácilmente una GUI similar a esta.