PROYECTO INTEGRADOR

Multiherramienta para electrónica

PROGRAMACIÓN ORIENTADA A OBJETOS

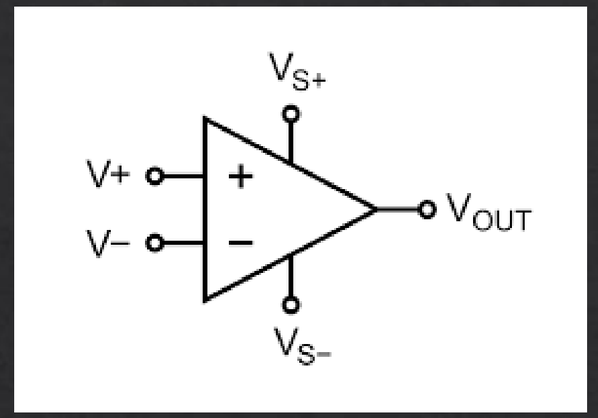
KENNETH EDUARDO CASTILLO GARCÍA
MARCO HEIMDAL CASTRO MAGAÑA

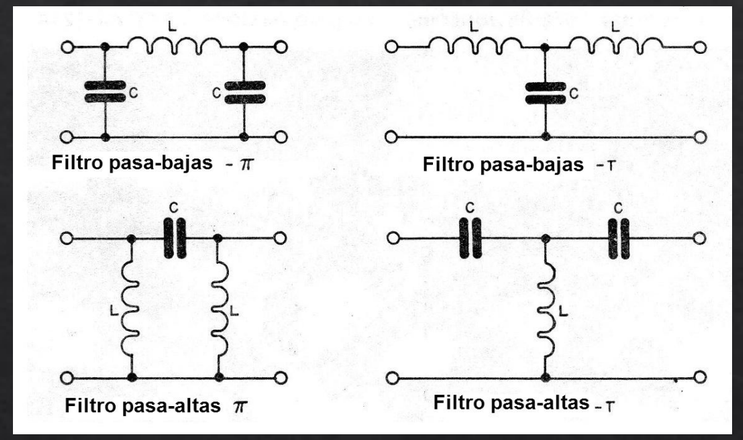
´¿Qué es?

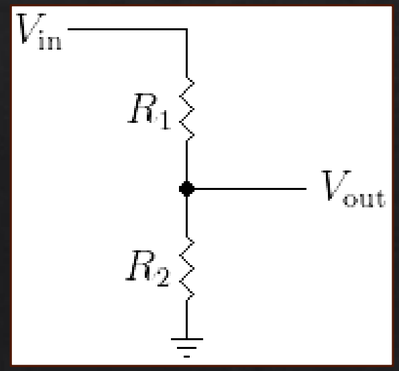
Es un programa que cuenta con diferentes utilidades o herramientas para el desarrollo de proyectos o prácticas afines a la electrónica.

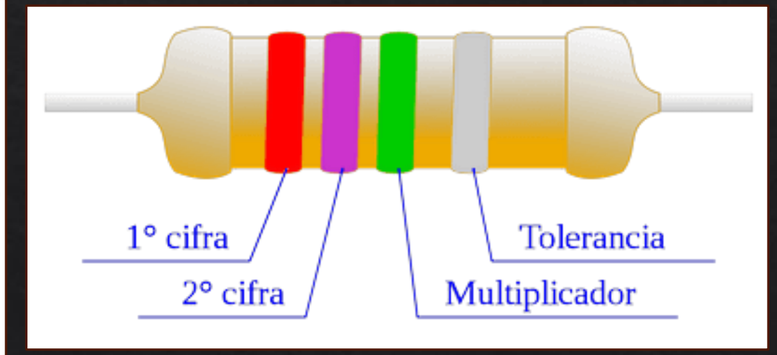
Cálculo de valores para:

- Resistencias: colores – valores, valores – colores
- Filtros pasivos: pasa-bajas y pasa-altas
- Filtros activos: pasa-bajas y pasa-altas
- Divisores de tensión
- Amplificadores diferenciales









Conocimientos de la materia

Nuestros cursos y talleres permiten conocer desde niños la robótica a través de actividades lúdicas que desarrollan la imaginación, creatividad y el pensamiento lógico.

Manejo de excepciones

Se define una excepción personalizada llamada ExcepcionCalculo, que extiende la clase Exception. Esta se lanza si los valores calculados son inválidos (como resistencia o capacitancia negativas). Además, se manejan otras excepciones como IllegalArgumentException para validar entradas desde el constructor, lo cual evita que el sistema procese configuraciones físicas imposibles.

Encapsulamiento

Se utilizó para proteger los atributos internos de cada clase (FiltroPasivo, FiltroActivo1erOrd, etc.). Las variables como R, C, frecCort son private y sólo se accede a ellas mediante métodos get, lo cual evita modificaciones externas no deseadas y mantiene la integridad de los datos.

Herencia

Aunque no hay herencia directa entre clases concretas, sí se emplea herencia a través de interfaces. Todas las clases que representan componentes electrónicos (FiltroPasivo, FiltroActivo1erOrd, AmpDif) implementan la interfaz Calculador, que define el método calcular(). Esto permite tratar a todos los objetos de forma uniforme, aunque internamente hagan cálculos distintos.

Polimorfismo

Gracias a que las clases implementan la misma interfaz (Calculador), el método calcular() se puede invocar de forma polimórfica, es decir, el mismo llamado se comporta diferente dependiendo del tipo de objeto (pasivo, activo o amplificador). Por ejemplo, en el main, podrías tener una lista de Calculador y llamar calcular() sin preocuparte por el tipo exacto del componente.

Implementación Gráfica y Funcional

Se desarrolló una interfaz gráfica interactiva utilizando Java Swing y AWT, con imágenes representativas de los circuitos electrónicos. Cada herramienta visual (filtro pasivo, filtro activo y amplificador diferencial) permite ingresar parámetros, realizar cálculos y visualizar los resultados directamente sobre la imagen del circuito.

Java Swing

Creación de ventanas, botones, menús y campos de texto

Java AWT

Dibujo de resistencias y capacitancias calculadas sobre imágenes

Java IO + ImageIO

Carga de imágenes desde archivos y recursos

JComboBox y JPanel

Para selección dinámica del tipo de filtro y actualización visual del circuito

Código fuente

```java
import javax.swing.*;

public class ResistorPanel extends JPanel {
    private Image backgroundImage;
    private boolean imageLoaded;
    private Color[] bandColors = new Color[4];

    private static final int IMAGE_WIDTH = 800;
    private static final int IMAGE_HEIGHT = 180;

    public ResistorPanel(String imagePath) {
        setPreferredSize(new Dimension(IMAGE_WIDTH, IMAGE_HEIGHT));
        for (int i = 0; i < 4; i++) {
            bandColors[i] = Color.BLACK;
        }
        try {
            InputStream imageFile = ResourceLoader.load(imagePath);
            if (imageFile == null) {
                throw new IOException("El archivo de imagen no existe: " + imagePath);
            }
            backgroundImage = ImageIO.read(imageFile);
            imageLoaded = true;
        } catch (IOException e) {
            imageLoaded = false;
            JOptionPane.showMessageDialog(this,
                    "Error al cargar la imagen: " + e.getMessage(),
                    "Error de Imagen",
                    JOptionPane.ERROR_MESSAGE);
        }
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        int panelWidth = getWidth();
        int panelHeight = getHeight();
        int xImage = (panelWidth - IMAGE_WIDTH) / 2;
        int yImage = (panelHeight - IMAGE_HEIGHT) / 2;

        if (imageLoaded && backgroundImage != null) {
            g.drawImage(backgroundImage, xImage, yImage, IMAGE_WIDTH, IMAGE_HEIGHT, this);
        } else {
            g.setColor(Color.LIGHT_GRAY);
            g.fillRect(0, 0, panelWidth, panelHeight);
        }

        int bandWidth = 40;
        int bandHeight = 100;
        int startX = xImage + 250;
        int y = yImage + (IMAGE_HEIGHT - bandHeight) / 2;
        int spacing = 30;

        for (int i = 0; i < 4; i++) {
            if (i == 3) {
                g.setColor(bandColors[i]);
                int x = startX + i * (bandWidth + spacing + 20);
                g.fillRect(x, y, bandWidth, bandHeight);
            } else {
                g.setColor(bandColors[i]);
                int x = startX + i * (bandWidth + spacing);
                g.fillRect(x, y, bandWidth, bandHeight);
            }
        }
    }

    public void setBandColor(int bandIndex, Color color) {
        if (bandIndex >= 0 && bandIndex < 4) {
            bandColors[bandIndex] = color;
            repaint();
        }
    }
}
```

```java
package electool;

public enum TipoFiltroActivo {
    L_P, H_P
}
```

```java
package electool;

public enum TipoFiltroPasivo {
    L_P, H_P
}
```

```java
package electool;

public class FiltroActivo1erOrd implements Calculador {
    private final TipoFiltroActivo tipo;
    private final double           frecCort;
    private final double           gan;
    private Double R1;
    private Double C;
    private Double R2;

    public FiltroActivo1erOrd() {
        // LP @1kHz, gan=2, R1=10kΩ por defecto
        this(TipoFiltroActivo.L_P, 1000.0, 2.0, 10000.0, null);
    }

    public FiltroActivo1erOrd(TipoFiltroActivo tipo, double frecCort, double gan, Double R1, Double C) {
        if (frecCort <= 0) throw new IllegalArgumentException("frecCort > 0");
        if (gan < 1.0)     throw new IllegalArgumentException("gan ≥ 1");
        if (R1 == null && C == null)
            throw new IllegalArgumentException("Especifique R1 o C");
        this.tipo     = tipo;
        this.frecCort = frecCort;
        this.gan      = gan;
        this.R1       = R1;
        this.C        = C;
    }

    @Override
    public void calcular() throws ExcepcionCalculo {
        // frecCort = 1/(2π·R1·C)   y   gan = 1 + R2/R1 ⇒ R2 = (gan−1)·R1
        if (R1 != null) {
            C = 1.0 / (2 * Math.PI * R1 * frecCort);
        } else {
            R1 = 1.0 / (2 * Math.PI * C * frecCort);
        }
        R2 = (gan - 1) * R1;
        if (R1 <= 0 || C <= 0 || R2 <= 0)
            throw new ExcepcionCalculo("Parámetros inválidos");
    }

    public TipoFiltroActivo getTipo()    { return tipo; }
    public double           getFrecCort(){ return frecCort; }
    public double           getGan()     { return gan; }
    public double           getR1()      { return R1; }
    public double           getR2()      { return R2; }
    public double           getC()       { return C; }
}
```

```java
package electool;

public class FiltroPasivo implements Calculador {
    private final TipoFiltroPasivo tipo;
    private final double frecCort;
    private Double R;
    private Double C;

    public FiltroPasivo() {
        // LP @1kHz, R=10kΩ por defecto
        this(TipoFiltroPasivo.L_P, 1000.0, 10000.0, null);
    }


    public FiltroPasivo(TipoFiltroPasivo tipo, double frecCort, Double R, Double C) {
        if (frecCort <= 0) throw new IllegalArgumentException("frecCort > 0");
        if (R == null && C == null)
            throw new IllegalArgumentException("Especifique R o C");
        this.tipo     = tipo;
        this.frecCort = frecCort;
        this.R        = R;
        this.C        = C;
    }

    @Override
    public void calcular() throws ExcepcionCalculo {
        // frecCort = 1/(2π·R·C)
        if (R != null) {
            C = 1.0 / (2 * Math.PI * R * frecCort);
        } else {
            R = 1.0 / (2 * Math.PI * C * frecCort);
        }
        if (R <= 0 || C <= 0)
            throw new ExcepcionCalculo("Resultado inválido");
    }

    public TipoFiltroPasivo getTipo()    { return tipo; }
    public double           getFrecCort(){ return frecCort; }
    public double           getR()       { return R; }
    public double           getC()       { return C; }
}
```

Aprendizajes

Aplicamos varias de las competencias vistas a lo largo del curso para la creación de este proyecto, especialmente se utilizaron las técnicas de manejo de clases como herencia, polimorfismo, enumeraciones, además de que se implementaron métodos especiales para el manejo de la interfaz de usuario de nuestra aplicación, como por ejemplo, Jpanel, ActionListener, Graphics, las cuales se investigaron para su implementación.

Se buscó también mantener las buenas prácticas de programación de Java vistas durante el curso como una correcta indentación de los códigos, formato adecuado de variables y métodos, así como la separación de códigos en diferentes paquetes según su funcionalidad.

Por último, también se trabajó la parte de compatibilidad y portabilidad del proyecto ya que este se exportó como un archivo JAR ejecutable por la JVM, por lo que se tuvieron que tener algunas consideraciones para el manejo de imágenes y librerías, así como una versión adecuada de JRE para el proyecto.

Referencias

Trail: Creating a GUI with Swing (The JavaTM Tutorials). (n.d.). https://docs.oracle.com/javase/tutorial/uiswing/

Trail: 2D Graphics (The JavaTM Tutorials). (n.d.). https://docs.oracle.com/javase/tutorial/2d/

Lesson: Writing event listeners (The JavaTM Tutorials > Creating a GUI with Swing). (n.d.). https://docs.oracle.com/javase/tutorial/uiswing/events/

Lesson: Working with Images (The JavaTM Tutorials > 2D Graphics). (n.d.). https://docs.oracle.com/javase/tutorial/2d/images/

Reading/Loading an Image (The JavaTM Tutorials > 2D Graphics > Working with Images). (n.d.). https://docs.oracle.com/javase/tutorial/2d/images/loadimage.html

how to package images into a Runnable JAR. (n.d.). Stack Overflow. https://stackoverflow.com/questions/8532114/how-to-package-images-into-a-runnable-jar

Wichit Sombat. (2012, 28 julio). How add images to runnable jar using Eclipse 4.2 [Vídeo]. YouTube. https://www.youtube.com/watch?v=rCoed3MKpEA