Manual Tecnico

ArduinoC

```
public class ArduinoC {

}
```

Esta clase principal main del proyecto es el que conecta al puerto serial, lee los codigos de los botones enviados por el arduino, interpreta valor y ejecuta acciones en le GUI.
Función

- Abre el puerto usando la interfaz JSerialComm.
- Lee los codigos IR como 0x16, 0x17, etc.
- Llama al metodo Pantalla2 para mostar los valores en la pantalla.
- Inserta los datos en la base usando ArduinoDataDAO.
- Como acciona especial ejecuta botones con `.doClick()` en las interfaz (JButton1, pantalla.botonInicio, JBotton2).

Funcionamiento del switch

En el codigo es una cadena que representa el valor hexadeciaml de la señal del IR recibida por el Arduino y enviada a Java.
Con el `switch` se revisa que valor tiene `codigo` y se ejecuta la instruccion correspondiente.
Cada `case` representa un comando diferente del control remoto.

```
switch (linea) {
   case "0x16":
       System.out.println("Numero 0 visto en pantalla");
       pantalla.mostrarNumero("0");
       break;
   case "0xC":
       System.out.println("Numero 1 visto en pantalla");
       pantalla.mostrarNumero("1");
       break;
   case "0x18":
       System.out.println("Numero 2 visto en pantalla");
       pantalla.mostrarNumero("2");
       break;

   case "...":
```

Metodos

iniciar():

- Configura la conexion con Arduino usando la librería `PanamaHitek_Arduino`.
- Declara un `SerialPortEventListener` para reaccionar automáticamente cuando llega un dato desde Arduino.
- Cuando llega un dato, lo lee y lo interpreta como un comando IR.
- Llama a `procesarCodigoIR(String codigo)` para manejar la acción correspondiente.

procesarCodigoIR(String codigo):

- Analiza el código IR recibido y determina qué acción realizar.
- Actualiza la interfaz mediante actualizarPantalla(...).
- Guarda el código en la base de datos llamando a ArduinoDataDAO.insertarCodigo(codigo)

actualizarPantalla(String mensaje):

- Cambia dinámicamente el texto de una etiqueta (generalmente en la interfaz) para reflejar el último comando recibido.
- Lanza un hilo (SwingUtilities.invokeLater) para asegurar que el cambio gráfico se hace en el hilo de la interfaz.

Funcionamiento general

```
Arduino (envia IR) --> puerto serie --> PanamaHitek_Arduino -->
arduinoc.iniciar()
    ↓
procesarCodigoIR(codigo)
    ↓
actualizarPantalla("acción")
    ↓
insertar en base de datos
```

ArduinoDataDAO

Calse principal

```
public class ArduinoDataDAO {
}
```

Su principal funcion es el acceso y manipulación de la base de datos `datos_arduino.db`.

Metodos

- *insertCodigo(String codigo):* Inserta un nuevo código leido desde el Arduino.
- *connect():* Establece o retorna una conexión a una conexion abierta.

- *closeConnection():* Cierra la conexión con la base de datos.
  Fragamento de Código

```
String sql = "INSERT INTO registros_arduino(codigo_arduino) VALUES(?)";
```

Data Base Initializer

Clase Principal

```
public class DataBaseInitializer {
}
```

Esta clase tiene la responsabilidad de inicializar la base de datos *SQLite* usada por el sistema. Su proposito es asegurarse de que exista la base de datos y de que la tabla `registros_arduino` esté creada antes de que el sistema comience a guardar datos provenientes del arduino.
Funcion

- ***Conexion a la base de datos:*** Se conecta a una base de datos SQLite local llamada `datos_arduino.db` . Si el archivo no existe, lo crea automáticamente.
- ***Creación de la tabla:*** Crea la tabla `registros_arduino` si aun no existe, utilizando una sentencia SQL `CREATE TABLE IF NOT EXISTS`.

Metodos

- *initializeDatabase():* Punto de entrada estatico para crear o validar la estructura de la base de datos. Se conecta a la base de datos y si l conexion fue exitosa entonces llama a `createTables`.
- *createTables(Connection conn):* Ejecuta el SQL para crear la tabla `registros_arduino` si no existe.

Pantalla inicial 1

Clase Principal

```
public class pantallainicial1 extends JPanel {
}
```

pantallainicial1 es una clase que extiende `JPanel`, representando la pantalla principal del sistema.
Calses Utilizadas

- Pantalla2 que permite la navegacion a otra pantalla.
- MenuComando es una vetanata secundaria que permite la ejecucion de comandos.
- *Librerias:* Swing y AWT para la creacion de interfaces graficas(JPanel, JLabel, JButton, etc.).

Atributos

```
Pantalla2 frame2 = new Pantalla2();
public JButton botonInicio, jButton1, jButton2;
public JPanel jPanel1;
public JLabel titleLabel, etiquetaNumero, logoLabel;
public JTextField numeroTextField;
```

Estos atributos definen los componentes de la interfaz gráfica, como botones, paneles y etiquetas.

Metodos

Constructor
Inicializa los componentes llamando al método `initComponents()`.

initComponents()

- Define el `Layout` general.
- Agrega tres secciones principales: superior, central e inferior.

createComponents()

- Muestra el logo y el título de la aplicación.
- Lista de los nombres de manera centrada.

createCenterPanel()

- Muestra el contenido principal en una tarjeta.
- Contiene la etiqueta de número y de los botones.

createBottomPanel()
Crea tres botones:

- Iniciar Sistema (verde)
- Menu de comandos (azul)
- Salir (rojo)

createStyledButton(String texto, Color baseColor)

- Método reutilizable para crear botones estilizados.
- Agrega comportamiento hover.

botonInicioActionPerformed()

- Abre la interfaz de la `pantalla2`.
- Cierra la ventana actual.

setLogo(String rutalogo)

- Se encarga de actualizar y cargar dinamicamente el logo desde un archivo externo.

Dependencia

clase/libreria	descripción
javax.swing.*	Componentes de interfaz gráfica
java.awt.*	Componentes de diseño y color
java.io.File	Carga del logo desde archivos
pantalla2	Otra pantalla del sistema
MenuComando	Panel para ejecutar comandos

Menu comandos

Clase Principal

```
public class MenuComando extends JDialog {
}
```

Esta clse tiene la funcion para abrir una pantalla emergente (JDialog) con botones con funciones para el historial y cerrar.

Metodo

- *initUI() :* Tiene como funcion presentar los botones y el diseño de la ventana emergente y de brindar mediantes colores una mayor estetica.

Pantalla 2

Clase Principal

```
public class Pantalla2 extends JPanel {
}
```

Esta clase muestra el estado del sistema de recepcion de señales IR desde el Arduino y permite visualizar el numero detectado mediante una imagen correspondiente.
Atribitos Principales

Atributos	Tipo	Descripción
titleLabel	JLabel	Muestra el titulo principal
instructionLabel	Jlabel	Muestra instrucciones al usuario
statusLabel	Jlabel	Muestra el estado del sistema
displaylabel	JLabel	Muestra el texto cuando no hay señal recibida

Atributos	Tipo	Descripción
backButton	JButton	Botón para regresar a la pantalla inicial
keyDisplayPanel	JPanel	Panel para mostrar el número detectado

Metodos

- *Pantalla2():* Inicializa los componentes y costruye la interfaz grafica.
- *initComponents():* Configura el `layout` y agrega a los paneles superior, centarl e inferior al `JPanel` principal.
- *createTopPanel():* Construye el panel superior con el titulo.
- *createCenterPanel():*

1. Muestra el mensaje de espera de señal IR.
2. Muestra el panel con imagen o texto.
3. Muestra el estado del sistema.

- ***mostrarNumero(String numero):***
  Este metodo es clave ya que se llama desde la comunicacion Arduino-Java para actualizar la interfaz cunado se recibe la señal IR.

1. Si el numero recibido es valido se carga la imagen correspondiente.
2. Si no existe imagen se muestra un mensaje de error.
3. en cualquier caso se actualiza el color del panel para dar un feedback visual.

History Dialog

Clase Principal

```
public class HistoryDialog extends JDialog {
}
```

Esta calse nos permite visualizar un resumen de los codigos IR registrados junto a la calidad de veces que fueron recibidos permitiendo asi una tranzabilidad de los comandos utilizados en el sistema.

Metodos

- *HistoryDialog(JFrame parentFrame):* Es el constructor que inicializa el cuadro de dialogo.
- *initUI():* Metodo privado encargado de de configurar y organizzar toda la interfaz grafica.

CONEXION CON LA BASE DE DTOS

```
Map<String, Integer> codigoCounts = ArduinoDataDAO.getCodigoCounts();
```

- Consulta y recupera un `Map` con:

1. **Clave:** Codigo IR(String)
2. **Valor:** NUmero de veces recibido(Integer).

Integracion del Sistema

- Este cuadro de diálogo es llamado desde MenuComandos.java al presionar el botón "Historial de comandos".
- Se comunica indirectamente con la clase ArduinoDataDAO, que contiene la lógica de acceso a la base de datos.

```
+-----------------------------------------------+
|         Resumen de Comandos Recibidos         |
+-----------------------------------------------+
| Código: 0x1C - Veces recibido: 4              |
| Código: 0x15 - Veces recibido: 3              |
| Código: 0x0C - Veces recibido: 1              |
| ...                                           |
+-----------------------------------------------+
|                  [Cerrar]                     |
+-----------------------------------------------+
```