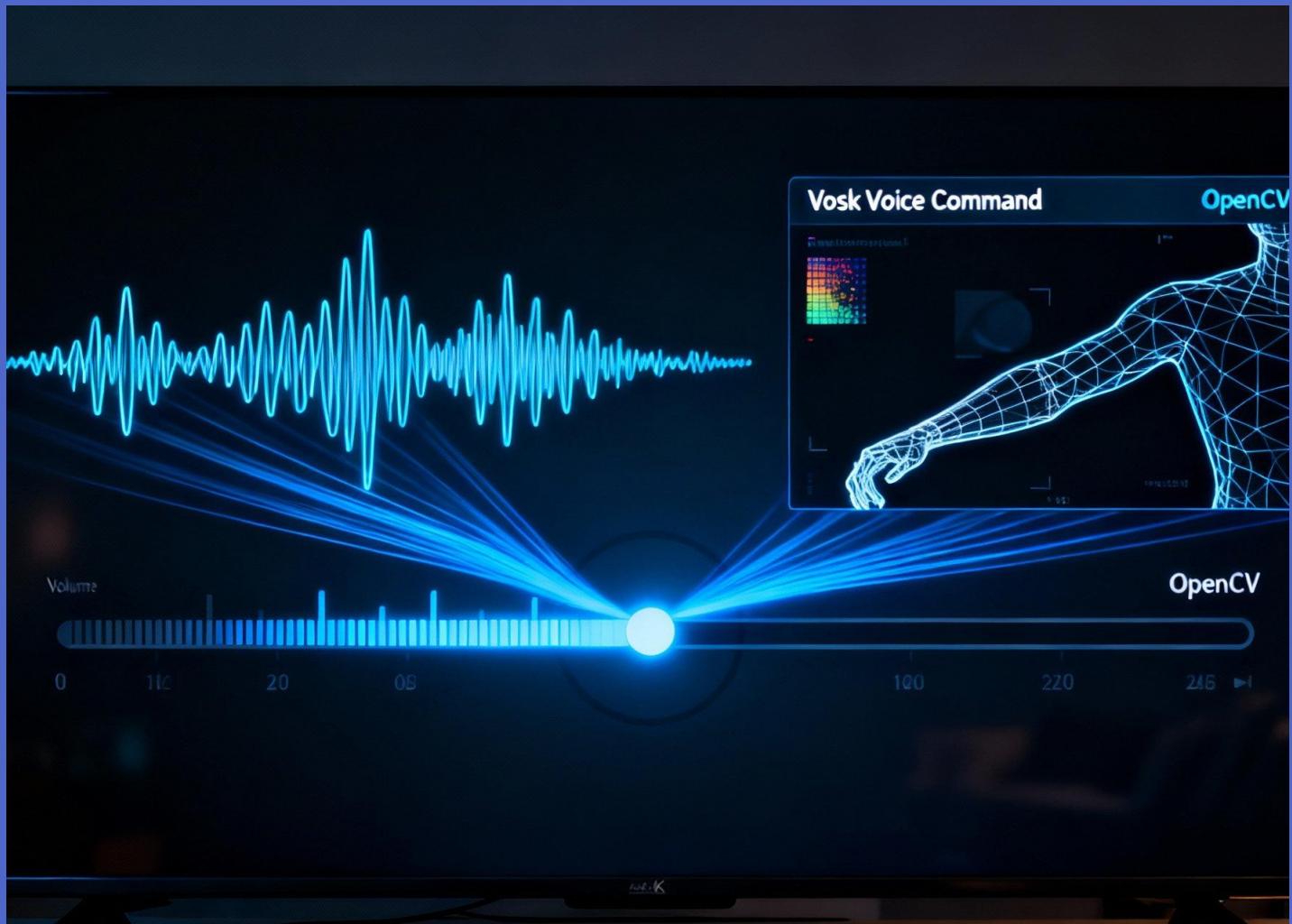


DESARROLLO DE INTERFACES

TAREA I UD2



ALUMNO CESUR

25/26

Alejandro Muñoz de la Sierra

PROFESOR

Manuel Gómez Lora

I N T R O D U C C I O N

Este caso práctico se centra en un tema que, aunque inicialmente pueda parecer cosa del futuro, está cada vez más presente en nuestra vida cotidiana: las interfaces naturales de usuario. Nos referimos a formas de interactuar con un sistema sin la necesidad del clásico teclado o ratón. La idea principal fue experimentar con métodos de interacción alternativos —como la voz o el movimiento corporal— aplicados a un escenario simple y muy reconocible: controlar el volumen de una televisión.

Para organizar el trabajo de forma clara, imaginamos el sistema como un pequeño “ecosistema” doméstico digital. En este ecosistema, un dispositivo, que sería la televisión, puede recibir órdenes a través de dos interfaces distintas: un control por voz y un control por gestos captados mediante una cámara. A partir de esta idea, decidimos dividir el proyecto en tres módulos principales:

MandoTelevision: Aquí reside la lógica del dispositivo.

MandoVoz: Se encarga del reconocimiento de los comandos hablados.

DeteccionMovimientoAR: Este gestiona los gestos mediante la imagen.

Este planteamiento modular nos ayudó a comprender cómo se comunican las distintas fuentes de entrada —audio y vídeo— con el procesado central y, finalmente, con las acciones que ejecuta el sistema. Personalmente, creo que este enfoque fue crucial para mantener la claridad y el orden en el proyecto.

0 1

O B J E T I V O S D E L C A S O P R Á C T I C O

Al iniciar el proyecto, nos planteamos una serie de objetivos que abarcaban tanto la parte conceptual como la técnica y la pedagógica.

Objetivos conceptuales:

- Comprender qué son las interfaces alternativas de entrada.
- Conocer las bases del reconocimiento de voz.
- Entender cómo funciona la detección de movimiento usando visión por computador.
- Iniciarnos en modelos ligeros de machine learning como **Vosk**.
- Aprender a trabajar con librerías multimedia complejas, como **OpenCV** y **JavaCV**.

Objetivos técnicos:

- Desarrollar un proyecto Java limpio y modular.
- Integrar dependencias externas a través de Maven.
- Capturar audio en tiempo real y procesarlo correctamente.
- Obtener vídeo desde la cámara del equipo y analizarlo.
- Implementar reglas de decisión basadas en gestos y comandos hablados.
- Gestionar adecuadamente los recursos (audio, vídeo y memoria) para evitar fallos.

Objetivos pedagógicos:

- Mejorar la capacidad de documentar de forma clara y profesional.
- Afrontar problemas reales de configuración y aprender a resolverlos con criterio.

0 2

ENTORNO DE DESARROLLO

Para el desarrollo nos apoyamos en un entorno bastante completo:

JDK 25, instalado manualmente en C:\Java\jdk-25. Optamos por esta instalación local en lugar de usar JavaSE-25 para evitar problemas y mantener estabilidad.
Eclipse IDE, configurado para trabajar con Maven.

Dependencias externas, gestionadas íntegramente desde el pom.xml:

Vosk (para reconocimiento de voz).

JavaCV (captura de vídeo).

OpenCV (procesamiento de imagen).

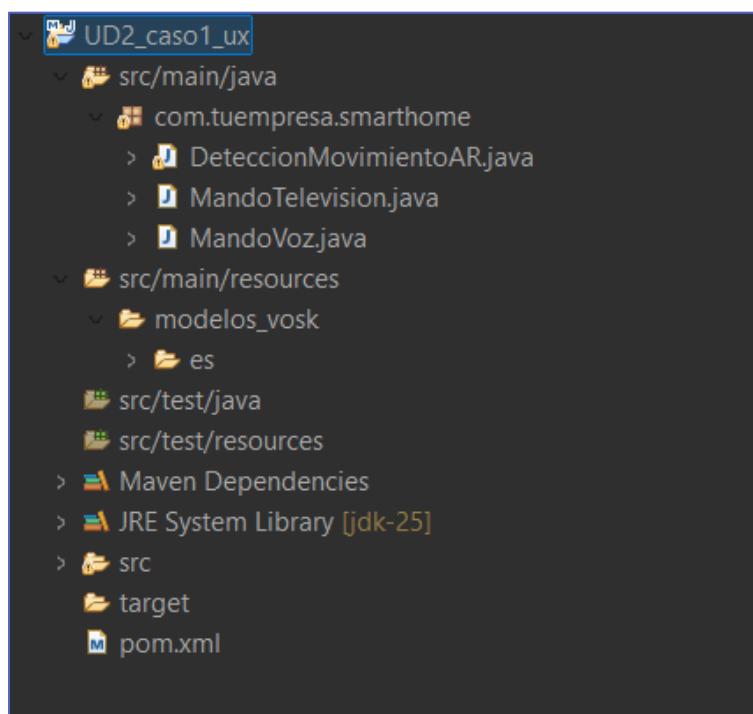
Modelos:

Modelo Vosk en español ubicado en: src/main/resources/modelos_vosk/es

Hardware:

Micrófono integrado.

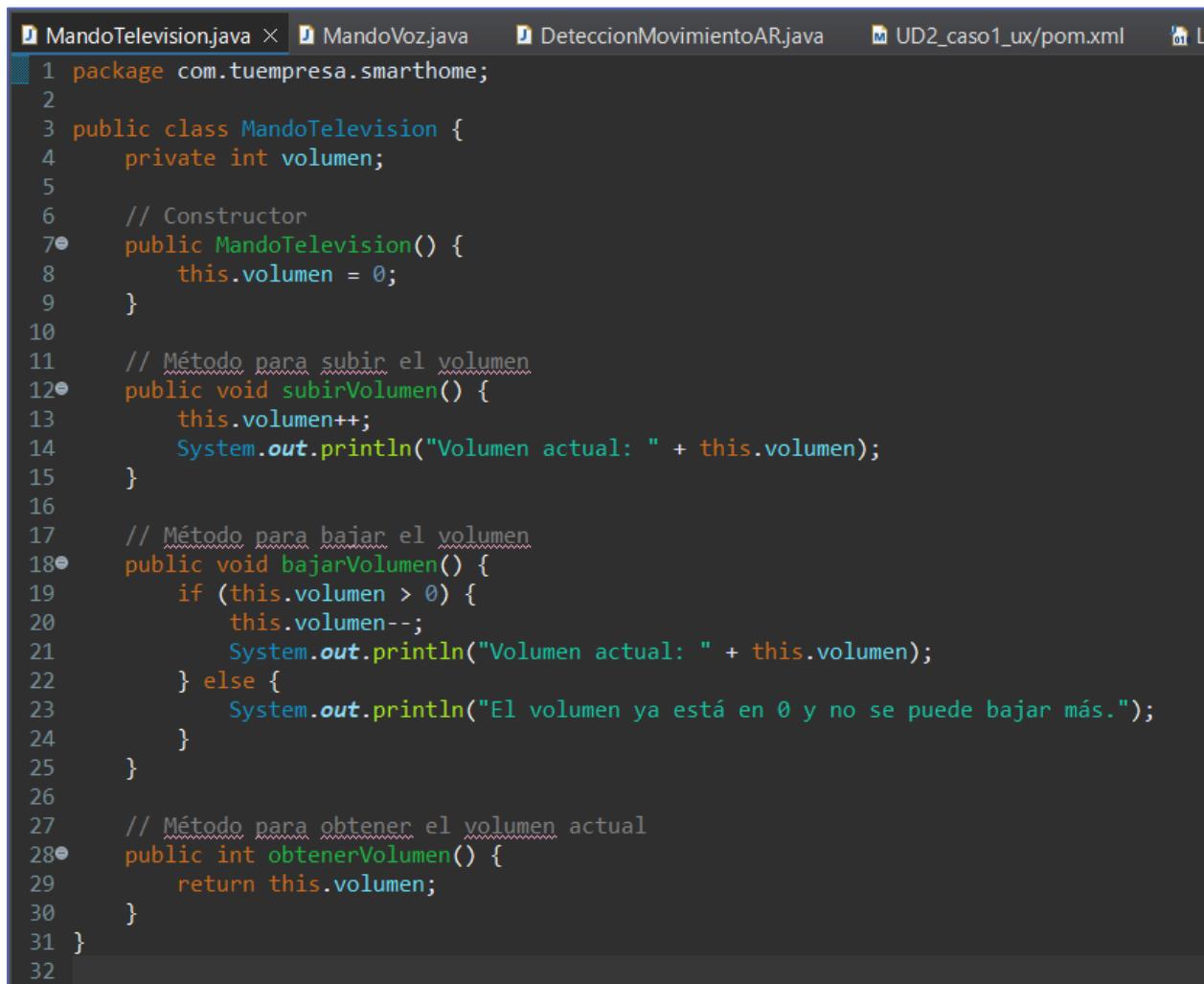
Webcam estándar (índice 0 en JavaCV).



CLASE MANDO TELEVISION

La base de todo es el objeto, **MandoTelevision**, como el director de orquesta, recibiendo órdenes de dos fuentes distintas.

Es nuestra simulación de un mando real. Guarda celosamente el volumen actual y, se asegura de que nunca caiga por debajo de cero. Además, nos dice qué módulos externos están en acción, mostrando mensajes informativos.



The screenshot shows a Java code editor with the file `MandoTelevision.java` open. The code defines a class `MandoTelevision` with methods to control the volume. The code is as follows:

```
1 package com.tuempresa.smarthouse;
2
3 public class MandoTelevision {
4     private int volumen;
5
6     // Constructor
7     public MandoTelevision() {
8         this.volumen = 0;
9     }
10
11    // Método para subir el volumen
12    public void subirVolumen() {
13        this.volumen++;
14        System.out.println("Volumen actual: " + this.volumen);
15    }
16
17    // Método para bajar el volumen
18    public void bajarVolumen() {
19        if (this.volumen > 0) {
20            this.volumen--;
21            System.out.println("Volumen actual: " + this.volumen);
22        } else {
23            System.out.println("El volumen ya está en 0 y no se puede bajar más.");
24        }
25    }
26
27    // Método para obtener el volumen actual
28    public int obtenerVolumen() {
29        return this.volumen;
30    }
31}
32
```

```
1 package com.tuempresa.smarthouse;  
2
```

El **package**, como tal, agrupa las clases en lo que podríamos considerar una "carpeta virtual," denominada **com.tuempresa.smarthouse**. Esta organización, a mi parecer, es crucial para mantener el proyecto ordenado y evitar posibles choques de nombres con otras librerías.

```
3 public class MandoTelevision {
```

Ahora bien, declaramos la clase pública **MandoTelevision**. Que sea pública, creo yo, es esencial, ya que otras clases, como **MandoVoz** o **DeteccionMovimientoAR**, deben tener la posibilidad de usarla.

```
3 public class MandoTelevision {  
4     private int volumen;  
5 }
```

Por otro lado, definimos un atributo **volumen**, que es privado. En mi opinión, hacerlo privado es una buena práctica, ya que así nos aseguramos de que solo se pueda modificar a través de los métodos de la clase, garantizando así la consistencia.

```
6     // Constructor  
7     public MandoTelevision() {  
8         this.volumen = 0;  
9     }
```

El constructor, por su parte, inicializa el **volumen** a 0 al momento de crear un objeto **MandoTelevision**. Me parece una buena manera de empezar con un valor conocido.

```
11     // Método para subir el volumen
12     public void subirVolumen() {
13         this.volumen++;
14         System.out.println("Volumen actual: " + this.volumen);
15     }
```

Tenemos, además, un método para subir el volumen. Este método incrementa el volumen en 1 y, además, muestra el valor por consola.

También hay un método para bajar el volumen. Este método, a diferencia del anterior, comprueba que el volumen no sea menor que 0 antes de decrementar, y proporciona feedback si el volumen ya está en 0.

```
17     // Método para bajar el volumen
18     public void bajarVolumen() {
19         if (this.volumen > 0) {
20             this.volumen--;
21             System.out.println("Volumen actual: " + this.volumen);
22         } else {
23             System.out.println("El volumen ya está en 0 y no se puede bajar más.");
24         }
25     }
```

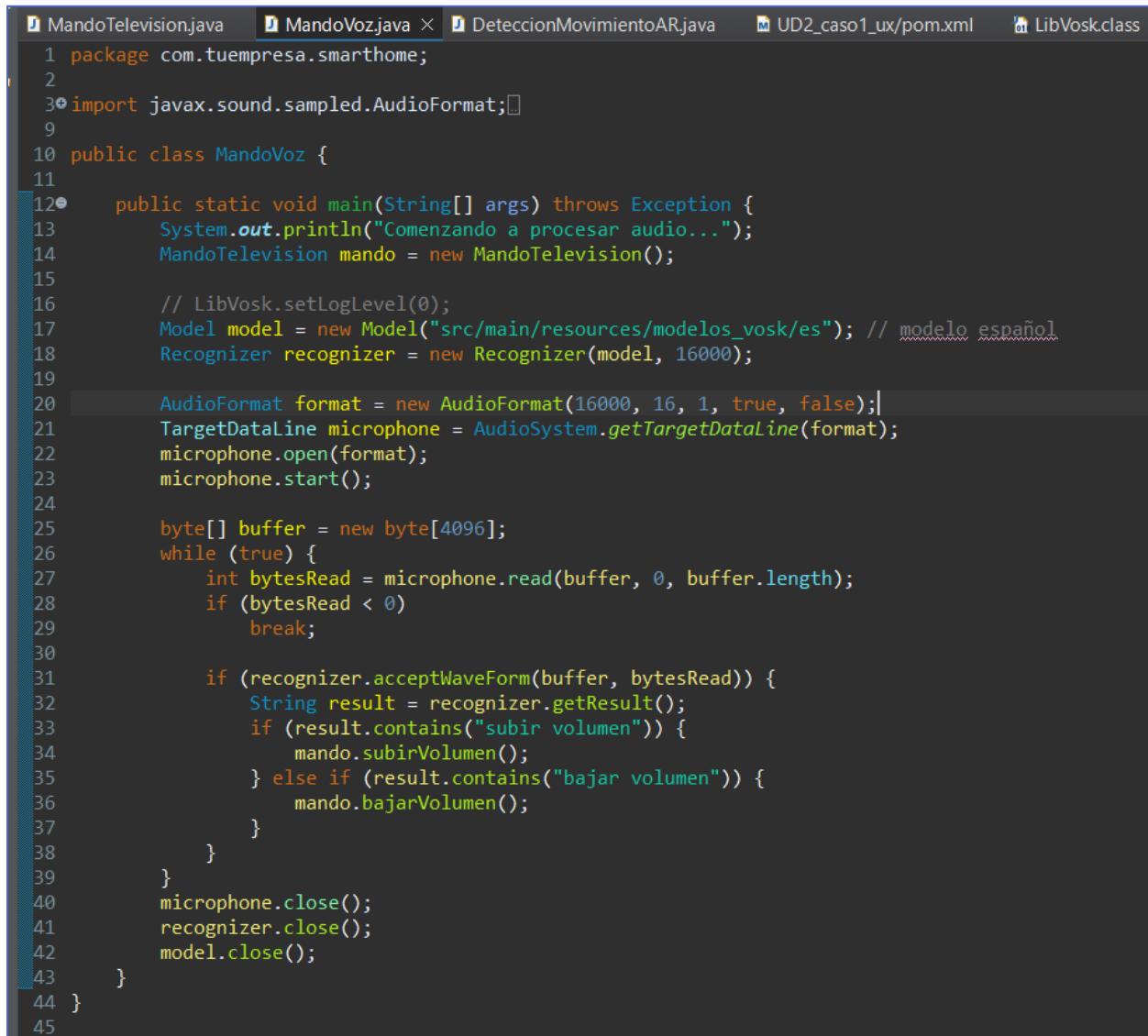
Finalmente, encontramos un método para consultar el volumen actual. En mi opinión, este método es bastante útil para otras clases que necesiten conocer el volumen sin modificarlo.

```
27     // Método para obtener el volumen actual
28     public int obtenerVolumen() {
29         return this.volumen;
30     }
31 }
```

0 4

CLASE MANDO VOZ

Este objeto escucha a 16 kHz, digiere bloques de 4096 bytes y usa el modelo de **Vosk** para descifrar lo que le decimos. Luego, traduce esos comandos en acciones concretas sobre el televisor. Un intérprete de primera.



```
1 package com.tuempresa.smarthome;
2
3 import javax.sound.sampled.AudioFormat;
4
5
6 public class MandoVoz {
7
8     public static void main(String[] args) throws Exception {
9         System.out.println("Comenzando a procesar audio...");
10        MandoTelevision mando = new MandoTelevision();
11
12        // LibVosk.setLevel(0);
13        Model model = new Model("src/main/resources/modelos_vosk/es"); // modelo español
14        Recognizer recognizer = new Recognizer(model, 16000);
15
16        AudioFormat format = new AudioFormat(16000, 16, 1, true, false);
17        TargetDataLine microphone = AudioSystem.getTargetDataLine(format);
18        microphone.open(format);
19        microphone.start();
20
21        byte[] buffer = new byte[4096];
22        while (true) {
23            int bytesRead = microphone.read(buffer, 0, buffer.length);
24            if (bytesRead < 0)
25                break;
26
27            if (recognizer.acceptWaveForm(buffer, bytesRead)) {
28                String result = recognizer.getResult();
29                if (result.contains("subir volumen")) {
30                    mando.subirVolumen();
31                } else if (result.contains("bajar volumen")) {
32                    mando.bajarVolumen();
33                }
34            }
35            microphone.close();
36            recognizer.close();
37            model.close();
38        }
39    }
40}
```

Al pertenecer al mismo **paquete**, podemos acceder fácilmente a la clase **MandoTelevision** sin tener que importarla.

```
1 package com.tuempresa.smarthome;
2
3• import javax.sound.sampled.AudioFormat;
4 import javax.sound.sampled.AudioSystem;
5 import javax.sound.sampled.TargetDataLine;
6
7 import org.vosk.Model;
8 import org.vosk.Recognizer;
```

Para empezar, necesitamos importar ciertas clases. Estas clases nos permiten tanto capturar audio —con **AudioSystem** y **TargetDataLine**—, como entender lo que decimos, o sea, reconocer comandos de voz usando **Model** y **Recognizer**.

```
10 public class MandoVoz {
11
12•     public static void main(String[] args) throws Exception {
13         System.out.println("Comenzando a procesar audio...");
14         MandoTelevision mando = new MandoTelevision();
```

Luego, creamos una instancia de **MandoTelevision** y, para que el usuario lo sepa, mostramos un mensajito de inicio.

```
16     // LibVosk.setLevel(0);
17     Model model = new Model("src/main/resources/modelos_vosk/es"); // modelo español
18     Recognizer recognizer = new Recognizer(model, 16000);
```

Ahora, algo crucial: cargamos el modelo **Vosk** en **español**. También, ajustamos el reconocedor a una frecuencia de muestreo de 16kHz, algo importante para la claridad.

```
20     AudioFormat format = new AudioFormat(16000, 16, 1, true, false);
21     TargetDataLine microphone = AudioSystem.getTargetDataLine(format);
22     microphone.open(format);
23     microphone.start();
```

Por supuesto, debemos configurar el micrófono para escuchar el audio en vivo, al momento.

```
25     byte[] buffer = new byte[4096];
26     while (true) {
27         int bytesRead = microphone.read(buffer, 0, buffer.length);
28         if (bytesRead < 0)
29             break;
30
31         if (recognizer.acceptWaveForm(buffer, bytesRead)) {
32             String result = recognizer.getResult();
33             if (result.contains("subir volumen")) {
34                 mando.subirVolumen();
35             } else if (result.contains("bajar volumen")) {
36                 mando.bajarVolumen();
37             }
38         }
}
```

En un ciclo continuo, como una escucha constante, leemos el audio y lo pasamos al reconocedor. Si Vosk "entiende" un comando, pues ejecutamos la acción que corresponda.

```
40     microphone.close();
41     recognizer.close();
42     model.close();
43 }
```

Finalmente, para evitar problemas y liberar memoria, cerramos con cuidado todos los recursos usados.

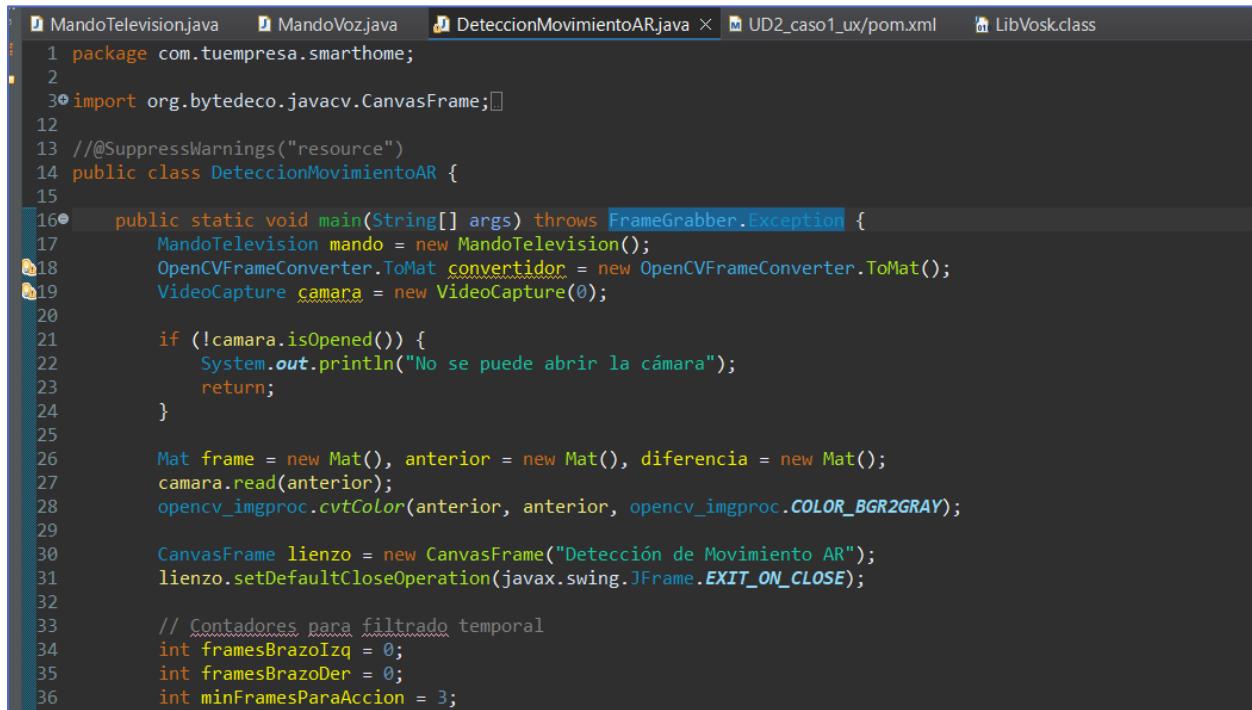
0 5

C L A S E

D E T E C C I O N M O V I M I E N T O A R

Aquí es donde la cámara entra en juego, conectándose con un simple **VideoCapture(0)**. Transformamos las imágenes a escala de grises y buscamos movimiento comparando fotogramas consecutivos. Analizamos los contornos más llamativos para reconocer gestos y aplicamos un filtro de repetición para evitar esos falsos positivos que tanto molestan.

La regla a implementar: mover la mano a la izquierda sube el volumen; a la derecha, lo baja.



The screenshot shows a Java code editor with several tabs at the top: "MandoTelevision.java", "MandoVoz.java", "DeteccionMovimientoAR.java" (which is the active tab), "UD2_caso1_ux/pom.xml", and "LibVosk.class". The code in the editor is as follows:

```
1 package com.tuempresa.smarthome;
2
3 import org.bytedeco.javacv.CanvasFrame;
4
5 // @SuppressWarnings("resource")
6 public class DeteccionMovimientoAR {
7
8     public static void main(String[] args) throws FrameGrabber.Exception {
9         MandoTelevision mando = new MandoTelevision();
10        OpenCVFrameConverter.ToMat convertidor = new OpenCVFrameConverter.ToMat();
11        VideoCapture camara = new VideoCapture(0);
12
13        if (!camara.isOpened()) {
14            System.out.println("No se puede abrir la cámara");
15            return;
16        }
17
18        Mat frame = new Mat(), anterior = new Mat(), diferencia = new Mat();
19        camara.read(anterior);
20        opencv_imgproc.cvtColor(anterior, anterior, opencv_imgproc.COLOR_BGR2GRAY);
21
22        CanvasFrame lienzo = new CanvasFrame("Detección de Movimiento AR");
23        lienzo.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24
25        // Contadores para filtrado temporal
26        int framesBrazoIzq = 0;
27        int framesBrazoDer = 0;
28        int minFramesParaAcción = 3;
```

```

37     try {
38         while (lienzo.isVisible() && camara.read(frame)) {
39             opencv_imgproc.cvtColor(frame, frame, opencv_imgproc.COLOR_BGR2GRAY);
40             opencv_core.absdiff(anterior, frame, diferencia);
41             opencv_imgproc.threshold(diferencia, diferencia, 25, 255, opencv_imgproc.THRESH_BINARY);
42
43             MatVector contornos = new MatVector();
44             opencv_imgproc.findContours(diferencia, contornos, new Mat(), opencv_imgproc.RETR_EXTERNAL,
45                                         opencv_imgproc.CHAIN_APPROX_SIMPLE);
46
47             boolean brazoIzq = false, brazoDer = false;
48
49             int ancho = frame.cols();
50             int alto = frame.rows();
51             int limiteY = alto / 2; // Solo mitad superior (brazo levantado)
52
53             for (int i = 0; i < contornos.size(); i++) {
54                 Rect rect = opencv_imgproc.boundingRect(contornos.get(i));
55                 int centroX = rect.x() + rect.width() / 2;
56                 int centroY = rect.y() + rect.height() / 2;
57
58                 // Solo contornos grandes en la mitad superior
59                 if (centroY < limiteY && rect.height() > 50 && rect.width() > 20) {
60                     if (centroX < ancho / 2)
61                         brazoIzq = true; // brazo izquierdo levantado
62                     else
63                         brazoDer = true; // brazo derecho levantado
64                 }
65             }
66
67             // Filtrado temporal
68             if (brazoIzq)
69                 framesBrazoIzq++;
70             else
71                 framesBrazoIzq = 0;
72             if (brazoDer)
73                 framesBrazoDer++;
74             else
75                 framesBrazoDer = 0;
76
77             if (framesBrazoIzq >= minFramesParaAccion) {
78                 mando.subirVolumen();
79                 framesBrazoIzq = 0;
80             }
81
82             if (framesBrazoDer >= minFramesParaAccion) {
83                 mando.bajarVolumen();
84                 framesBrazoDer = 0;
85             }
86
87             lienzo.showImage(convertidor.convert(frame));
88             anterior = frame.clone();
89         }
90     } finally {
91         camara.release();
92         lienzo.dispose();
93     }
94 }
95 }
96 }
```

```
MandoTelevision.java    MandoVoz.java    DeteccionMovimientoAR.java
1 package com.tuempresa.smarthome;
2
3 import org.bytedeco.javacv.CanvasFrame;
4 import org.bytedeco.javacv.FrameGrabber;
5 import org.bytedeco.javacv.OpenCVFrameConverter;
6 import org.bytedecoopencv.global.opencv_core;
7 import org.bytedecoopencv.global.opencv_imgproc;
8 import org.bytedecoopencv.core.Mat;
9 import org.bytedecoopencv.core.MatVector;
10 import org.bytedecoopencv.core.Rect;
11 import org.bytedecoopencv.videoio.VideoCapture;
12
```

Vale, pues, arrancamos importando las librerías que nos permiten pillar el video (con **VideoCapture**), manipular cada fotograma (**Mat**, **Rect**) y pintar los resultados en un **CanvasFrame**.

```
13 //@SuppressWarnings("resource")
14 public class DeteccionMovimientoAR {
15
16     public static void main(String[] args) throws FrameGrabber.Exception {
17         MandoTelevision mando = new MandoTelevision();
18         OpenCVFrameConverter.ToMat convertidor = new OpenCVFrameConverter.ToMat();
19         VideoCapture camara = new VideoCapture(0);
```

Luego, preparamos el terreno: creamos el "mando", el conversor de **frames** y le decimos que use la cámara que viene por defecto (la 0).

```
21         if (!camara.isOpened()) {
22             System.out.println("No se puede abrir la cámara");
23             return;
24         }
```

Aquí: comprobamos que la cámara responda bien; si no, no podemos avanzar. Una vez seguros, leemos el primer **frame** y lo pasamos a **grises**. Para que el proceso de detectar movimiento sea más fácil, evitamos problemas con colores.

```
26     Mat frame = new Mat(), anterior = new Mat(), diferencia = new Mat();
27     camara.read(anterior);
28     opencv_imgproc.cvtColor(anterior, anterior, opencv_imgproc.COLOR_BGR2GRAY);
29
```

Abrimos una ventana para ir viendo cómo quedan los frames según los vamos trabajando.

```
30     CanvasFrame lienzo = new CanvasFrame("Detección de Movimiento AR");
31     lienzo.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

También, definimos unas variables para un "filtro temporal". Esto es importante, porque no queremos que un simple movimiento rápido active la acción, sino que el brazo tenga que estar arriba un rato.

```
33     // Contadores para filtrado temporal
34     int framesBrazoIzq = 0;
35     int framesBrazoDer = 0;
36     int minFramesParaAcción = 3;
```

Ahora sí, procesamos cada frame: lo convertimos a gris, calculamos la diferencia con el frame anterior y aplicamos un umbral para que las zonas donde hay movimiento se vean más claras.

```
38     try {
39         while (lienzo.isVisible() && camara.read(frame)) {
40             opencv_imgproc.cvtColor(frame, frame, opencv_imgproc.COLOR_BGR2GRAY);
41             opencv_core.absdiff(anterior, frame, diferencia);
42             opencv_imgproc.threshold(diferencia, diferencia, 25, 255, opencv_imgproc.THRESH_BINARY);
43     }
```

Con esto, detectamos los bordes de los objetos que se mueven; estos contornos nos van a chivar si alguien ha levantado un brazo.

```
44     MatVector contornos = new MatVector();
45     opencv_imgproc.findContours(diferencia, contornos, new Mat(), opencv_imgproc.RETR_EXTERNAL,
46                                 opencv_imgproc.CHAIN_APPROX_SIMPLE);
```

Necesitamos unas variables que nos digan si hemos visto el brazo izquierdo o el derecho.

```
48     boolean brazoIzq = false, brazoDer = false;
```

Analizamos cada contorno con lupa: si está en la parte de arriba de la imagen y es grande, lo consideramos un brazo levantado.

```

54         for (int i = 0; i < contornos.size(); i++) {
55             Rect rect = opencv_imgproc.boundingRect(contornos.get(i));
56             int centroX = rect.x() + rect.width() / 2;
57             int centroY = rect.y() + rect.height() / 2;
58
59             // Solo contornos grandes en la mitad superior
60             if (centroY < limiteY && rect.height() > 50 && rect.width() > 20) {
61                 if (centroX < ancho / 2)
62                     brazoIzq = true; // brazo izquierdo levantado
63                 else
64                     brazoDer = true; // brazo derecho levantado
65     }

```

Y, por último, usamos contadores para los frames consecutivos. Solo si el brazo está arriba durante suficientes frames, ejecutamos la acción, evitando que cualquier espasmo active algo por error.

```

68         // Filtrado temporal
69         if (brazoIzq)
70             framesBrazoIzq++;
71         else
72             framesBrazoIzq = 0;
73         if (brazoDer)
74             framesBrazoDer++;
75         else
76             framesBrazoDer = 0;
77
78         if (framesBrazoIzq >= minFramesParaAccion) {
79             mando.subirVolumen();
80             framesBrazoIzq = 0;
81         }
82
83         if (framesBrazoDer >= minFramesParaAccion) {
84             mando.bajarVolumen();
85             framesBrazoDer = 0;
86     }

```

Vamos mostrando el frame procesado y, para no perder información, lo clonamos para usarlo en la siguiente vuelta.

```

88         lienzo.showImage(convertidor.convert(frame));
89         anterior = frame.clone();

```

Para terminar, toca ser ordenados: liberamos todos los recursos, cerramos la cámara y la ventana. Y listo, personalmente, me parece una buena manera de detectar gestos simples.

```

91     } finally {
92         camara.release();
93         lienzo.dispose();
94     }

```

0 6

PROBLEMAS ENCONTRADOS

Nos topamos con unos cuantos problemas reales:

Problema 1: Librerías

Intentamos meter las librerías a mano y... Errores por todos lados, culpa de los archivos nativos.

Solución: Maven.

Problema 2: Vosk

El micrófono parecía funcionar, pero solo nos devolvía silencio.

Solución: Ajustamos el formato del TargetDataLine y aumentamos el tamaño de los buffers. Funcionó.

Problema 3: OpenCV Muy Sensible

La cámara veía cualquier mínimo movimiento.

Solución: Escala de grises + absdiff + umbral de 25 + filtro temporal de 3 fotogramas consecutivos.

Problema 4: Cámaras HD Lentas

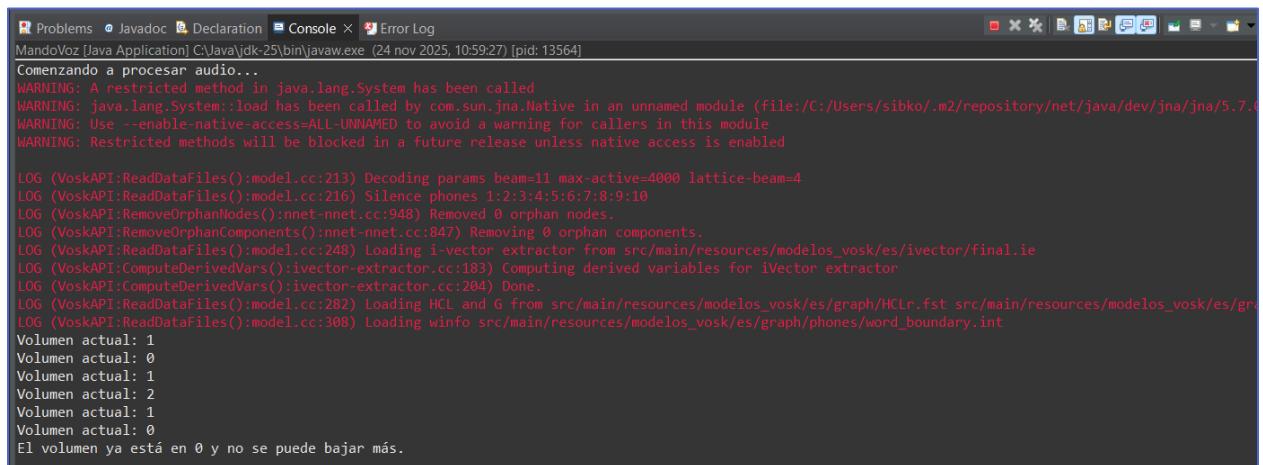
A resoluciones altas, el programa se arrastraba.

Solución: Gris y regiones más pequeñas.

07

PRUEBAS Y EJECUCION

Prueba de Reconocimiento de Voz. Ejecución de la clase MandoVoz.java



```
Problems Javadoc Declaration Console Error Log
MandoVoz [Java Application] C:\Java\jdk-25\bin\javaw.exe (24 nov 2025, 10:59:27) [pid: 13564]
Comenzando a procesar audio...
WARNING: A restricted method in java.lang.System has been called
WARNING: java.lang.System::load has been called by com.sun.jna.Native in an unnamed module (file:/C:/Users/sibko/.m2/repository/net/java/dev/jna/jna/5.7.0)
WARNING: Use --enable-native-access=ALL-UNNAMED to avoid a warning for callers in this module
WARNING: Restricted methods will be blocked in a future release unless native access is enabled

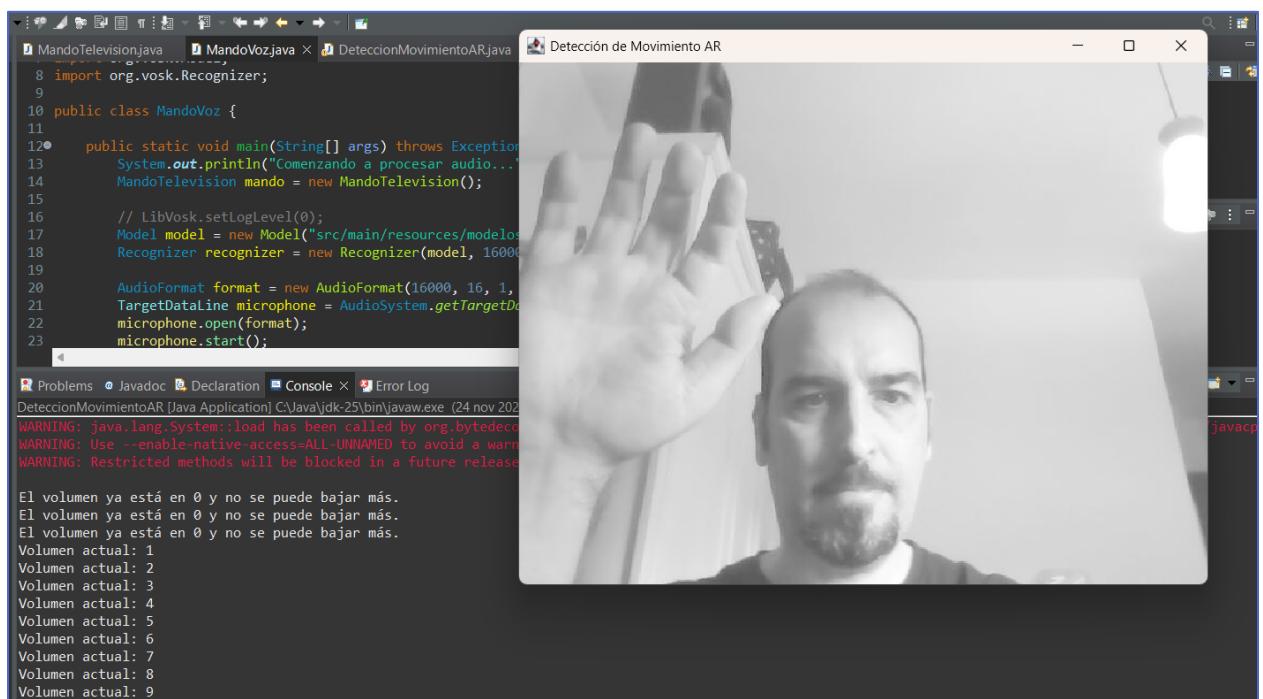
LOG (VoskAPI:ReadDataFiles():model.cc:213) Decoding params beam=11 max-active=4000 lattice-beam=4
LOG (VoskAPI:ReadDataFiles():model.cc:216) Silence phones 1:2:3:4:5:6:7:8:9:10
LOG (VoskAPI:RemoveOrphanNodes():nnet-nnet.cc:948) Removed 0 orphan nodes.
LOG (VoskAPI:RemoveOrphanComponents():nnet-nnet.cc:847) Removing 0 orphan components.
LOG (VoskAPI:ReadDataFiles():model.cc:248) Loading i-vector extractor from src/main/resources/modelos_vosk/es/ivector/final.ie
LOG (VoskAPI:ComputeDerivedVars():ivector-extractor.cc:183) Computing derived variables for iVector extractor
LOG (VoskAPI:ComputeDerivedVars():ivector-extractor.cc:204) Done.
LOG (VoskAPI:ReadDataFiles():model.cc:282) Loading HCL and G from src/main/resources/modelos_vosk/es/graph/HCLr.fst src/main/resources/modelos_vosk/es/graph/G.fst
LOG (VoskAPI:ReadDataFiles():model.cc:308) Loading winfo src/main/resources/modelos_vosk/es/graph/phones/word_boundary.int

Volumen actual: 1
Volumen actual: 0
Volumen actual: 1
Volumen actual: 2
Volumen actual: 1
Volumen actual: 0
El volumen ya está en 0 y no se puede bajar más.
```

Responde lento a las ordenes de voz, pero funciona.

Prueba de Detección de Gestos. Ejecución de la clase DeteccionMovimientoAR

Brazo derecho (elemento a la izquierda de la pantalla): subir volumen



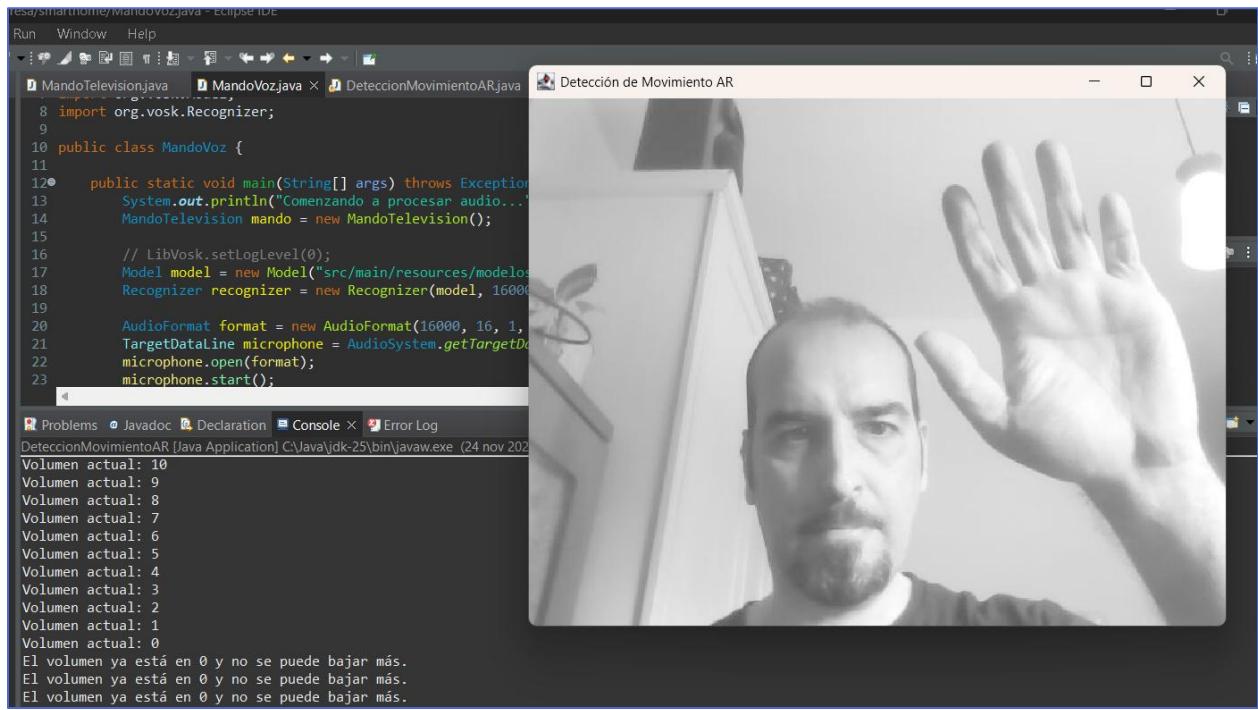
```
MandoTelevision.java MandoVoz.java DeteccionMovimientoAR.java
8 import org.vosk.Recognizer;
9
10 public class MandoVoz {
11
12     public static void main(String[] args) throws Exception {
13         System.out.println("Comenzando a procesar audio...");
14         MandoTelevision mando = new MandoTelevision();
15
16         // LibVosk.setLevel(0);
17         Model model = new Model("src/main/resources/modelos_vosk");
18         Recognizer recognizer = new Recognizer(model, 16000);
19
20         AudioFormat format = new AudioFormat(16000, 16, 1,
21         TargetDataLine microphone = AudioSystem.getTargetDataLine(format);
22         microphone.open(format);
23         microphone.start();
```

```
Problems Javadoc Declaration Console Error Log
DeteccionMovimientoAR [Java Application] C:\Java\jdk-25\bin\javaw.exe (24 nov 2025, 11:00:27) [pid: 13604]
WARNING: java.lang.System::load has been called by org.bytedeco.javacpp.presets.Vosk$Native
WARNING: Use --enable-native-access=ALL-UNNAMED to avoid a warning for callers in this module
WARNING: Restricted methods will be blocked in a future release unless native access is enabled

El volumen ya está en 0 y no se puede bajar más.
El volumen ya está en 0 y no se puede bajar más.
El volumen ya está en 0 y no se puede bajar más.

Volumen actual: 1
Volumen actual: 2
Volumen actual: 3
Volumen actual: 4
Volumen actual: 5
Volumen actual: 6
Volumen actual: 7
Volumen actual: 8
Volumen actual: 9
```

Brazo izquierdo (elemento a la derecha de la pantalla): bajar volumen



Ajustamos los tamaños de contorno mínimos para evitar que confundiera manos pequeñas, cabezas o sombras.

C O N C L U S I O N E S

Este proyecto nos ha permitido tocar una tecnología que solemos ver en dispositivos modernos, pero rara vez en proyectos educativos. Aprendimos a capturar audio y vídeo en tiempo real, a usar modelos de reconocimiento de voz locales, a detectar movimiento mediante diferencias de imagen y a estructurar un proyecto por módulos.

Además, Maven fue un salvavidas. Pero lo mejor de todo es la satisfacción de haber unido multimedia y machine learning en un proyecto Java funcional.

El sistema es ampliable: gestos nuevos, más comandos de voz, control de otros dispositivos... Pero lo importante es que ahora entendemos cada pieza y cómo encajan en este rompecabezas. Personalmente, creo que la curva de aprendizaje fue pronunciada, pero el resultado final bien valió el esfuerzo. La satisfacción de ver algo así funcionando es impagable.

0 9

REF E R E N C I A S

<https://maven.apache.org/guides/getting-started/>

<https://maven.apache.org/pom.html>

<https://docs.oracle.com/en/java/javase/25/>

<https://help.eclipse.org/latest/index.jsp>

<https://alphacepheli.com/vosk/install>

<https://alphacepheli.com/vosk/>

<https://github.com/bytedeco/javacv/wiki>

<https://bytedeco.org/>