Código del Mando de Televisión:

MandoTelevision.java

```
public class MandoTelevision {
    private int volumen;
    // Constructor
    public MandoTelevision() {
        this.volumen = 0;
    }
    // Método para subir el volumen
    public void subirVolumen() {
        this.volumen++;
    }
    // Método para bajar el volumen
    public void bajarVolumen() {
        if (this.volumen > 0) {
            this.volumen--;
        } else {
            System.out.println("El volumen ya está en 0 y no se
puede bajar más.");
        }
    }
    // Método para obtener el volumen actual
    public int obtenerVolumen() {
        return this.volumen;
    }
}
```

Dependencias de Maven necesarias para este ejercicio:

dependencias

```
<dependencies>
            <dependency>
                <groupId>com.alphacephei</groupId>
                <artifactId>vosk</artifactId>
                <version>0.3.45</version>
            </dependency>

      <dependency>
        <groupId>org.bytedeco</groupId>
        <artifactId>javacv-platform</artifactId>
        <version>1.5.4</version>
    </dependency>
      <dependency>
         <groupId>org.bytedeco</groupId>
          <artifactId>javacv-platform</artifactId>
          <version>1.5.10</version>
      </dependency>
<dependency>
    <groupId>org.bytedeco</groupId>
    <artifactId>opencv-platform</artifactId>
    <version>4.9.0-1.5.10</version>
</dependency>


      </dependencies>
```

Código para detectar brazos:

MandoTelevision.java

```
import org.bytedeco.javacv.CanvasFrame;
import org.bytedeco.javacv.Frame;
import org.bytedeco.javacv.FrameGrabber;
import org.bytedeco.javacv.OpenCVFrameConverter;
import org.bytedeco.opencv.global.opencv_core;
import org.bytedeco.opencv.global.opencv_imgproc;
import org.bytedeco.opencv.opencv_core.*;
import org.bytedeco.opencv.opencv_videoio.VideoCapture;

import javax.swing.*;

public class DeteccionMovimiento {
    public static void main(String[] args) throws
FrameGrabber.Exception {
        OpenCVFrameConverter.ToMat convertidor = new
OpenCVFrameConverter.ToMat();

        // Abrir la cámara
        VideoCapture captura = new VideoCapture(0);
        if (!captura.isOpened()) {
            System.out.println("No se puede abrir la cámara");
            return;
        }

        Mat fotograma = new Mat();
        Mat fotogramaPrevio = new Mat();
        Mat fotogramaDiferencia = new Mat();
        Mat fotogramaSuavizado = new Mat();

        CanvasFrame lienzo = new CanvasFrame("Detección de
Movimiento", CanvasFrame.getDefaultGamma() / 2.2);
        lienzo.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Leer el primer fotograma
        captura.read(fotogramaPrevio);
        opencv_imgproc.cvtColor(fotogramaPrevio, fotogramaPrevio,
opencv_imgproc.COLOR_BGR2GRAY);

        while (lienzo.isVisible() && captura.read(fotograma)) {
            opencv_imgproc.cvtColor(fotograma, fotograma,
opencv_imgproc.COLOR_BGR2GRAY);
            opencv_core.absdiff(fotogramaPrevio, fotograma,
fotogramaDiferencia);
            opencv_imgproc.GaussianBlur(fotogramaDiferencia,
fotogramaSuavizado, new Size(5, 5), 0);
            opencv_imgproc.threshold(fotogramaSuavizado,
fotogramaSuavizado, 25, 255, opencv_imgproc.THRESH_BINARY);

            // Encontrar contornos
            MatVector contornos = new MatVector();
            Mat jerarquia = new Mat();
            opencv_imgproc.findContours(fotogramaSuavizado,
contornos, jerarquia, opencv_imgproc.RETR_EXTERNAL,
opencv_imgproc.CHAIN_APPROX_SIMPLE);

            // Inicializar variables de detección
            boolean brazoIzquierdoLevantado = false;
```

```
            boolean brazoDerechoLevantado = false;

            // Crear una copia del fotograma para dibujar
rectángulos
            Mat fotogramaConRectangulos = fotograma.clone();

            // Dibujar rectángulos alrededor de las áreas de
movimiento detectadas y verificar posición
            for (int i = 0; i < contornos.size(); i++) {
                Rect rectanguloDelimitador =
opencv_imgproc.boundingRect(contornos.get(i));

                // Filtrar contornos pequeños que podrían ser ruido
                if (rectanguloDelimitador.width() < 30 ||
rectanguloDelimitador.height() < 30) {
                    continue;
                }

                // Verificar si el contorno está en la parte
superior del fotograma y tiene un tamaño significativo
                if (rectanguloDelimitador.y() < fotograma.rows() / 2
&& rectanguloDelimitador.height() > 100) {
                    if (rectanguloDelimitador.x() < fotograma.cols()
/ 2) {
                        brazoIzquierdoLevantado = true;

opencv_imgproc.rectangle(fotogramaConRectangulos,
rectanguloDelimitador, new Scalar(0, 255, 0, 0));
                    } else {
                        brazoDerechoLevantado = true;

opencv_imgproc.rectangle(fotogramaConRectangulos,
rectanguloDelimitador, new Scalar(0, 0, 255, 0));
                    }
                }
            }

            // Indicar si se detecta el brazo levantado y cuál
            if (brazoIzquierdoLevantado) {
                opencv_imgproc.putText(fotogramaConRectangulos,
"Brazo Izquierdo Levantado", new Point(10, 30),
opencv_imgproc.FONT_HERSHEY_SIMPLEX, 1.0, new Scalar(0, 255, 0, 0),
2, opencv_imgproc.LINE_AA, false);
            }
            if (brazoDerechoLevantado) {
                opencv_imgproc.putText(fotogramaConRectangulos,
"Brazo Derecho Levantado", new Point(10, 60),
opencv_imgproc.FONT_HERSHEY_SIMPLEX, 1.0, new Scalar(0, 0, 255, 0),
2, opencv_imgproc.LINE_AA, false);
            }

            // Mostrar el fotograma con los rectángulos y el texto
            Frame fotogramaMostrar =
convertidor.convert(fotogramaConRectangulos);
            lienzo.showImage(fotogramaMostrar);

            // Actualizar fotograma previo solo si hay movimiento
significativo
            if (contornos.size() > 0) {
                fotogramaPrevio = fotograma.clone();
            }
```

```
            }

            captura.release();
            lienzo.dispose();
        }
}
```