Laboratorio 5

Universidad Nacional de Costa Rica

Sede Regional Brunca

Campus Coto

Autores: **Jorge Rojas, Ismael Marchena**

Curso: **Diseño de Plataformas Móviles**

Profesor: **Pablo Noguera**

Año: **2025**

Tema: Gyroscope

Objetivo: Experimentar el uso del sensor de giroscopio en dispositivos móviles.

Justificación de dependencias: Para este laboratorio se hizo el uso de la dependicia llamada `Sensors_plus` con la versión 6.1.1, la cual fue instalada desde la con el comando `flutter pub add sensors_plus`.

Problemas encontrados:

1. Centrar el Widget
    - **Soluciones:**
        - Utilizamos la clase `MediaQuery` para obtener las dimensiones de la pantalla.
        - Obtener las dimenciones de la pantalla para dividirla entre 2, pero al hacer esto no se centraba ya que del elemnto se obtine sus coordenadas en la esquina superior izquierda, asi que para esto tambien le restamos el ancho y alto del elemnto.
2. Trancisiones Bruscas
    - **Soluciones:**
        - Buscamos una formula para suavizar las trancisiones, y encontramos una formula llamada "Lerping" que se utiliza para el suavizado exponencial. La fórmula de interpolación lineal (lerp) es: **resultado = inicio * (1 - porcentaje) + fin * porcentaje**. Esta fórmula calcula un valor entre dos valores, inicio y fin, basado en un porcentaje (que debe estar entre 0 y 1) que indica qué tan cerca del fin se encuentra el resultado.
3. Limites con los bordes
    - **Soluciones:**
        - Definimos una distancia máxima que el elemento interior no podía superar, y calculamos su distancia al centro utilizando la fórmula de distancia desde el origen hasta el punto (x, y): $\sqrt{x^2 + y^2}$ con el fin de verificar si se encontraba dentro del área permitida.

Código fuente:

```dart
import 'dart:async';
import 'dart:math';
import 'package:flutter/material.dart';
import 'package:sensors_plus/sensors_plus.dart';

class GyroscopeScreen extends StatefulWidget {
  const GyroscopeScreen({super.key});

  @override
  State<GyroscopeScreen> createState() => _GyroscopeScreenState();
}

class _GyroscopeScreenState extends State<GyroscopeScreen> {
  // Position X and Y.
  double _posX = 0.0;
  double _posY = 0.0;

  // Circles constants sizes
  final double _blueCircleSize = 250;
  final double _whiteCircleSize = 60;

  // Smoothing factor to make movement less jumpy
  final double _smoothingFactor = 0.2;

  // Sensitivity multiplier for movement scale
  final double _sensitivity = 80.0;


  // Gyroscope event subscription
  StreamSubscription<AccelerometerEvent>?
      _accelerometterSubcription;

  @override
  void initState() {
    super.initState();
    _initGyroscope(); // Start listening to gyroscope data
  }

  @override
  void dispose() {
    _accelerometterSubcription?.cancel(); // Stop listening when
        disposed
    super.dispose();
  }

  void _initGyroscope() {
    _accelerometterSubcription = _accelerometterEvents.listen((
```

```
        AccelerometterEvent event) {
      setState(() {
        // Apply smoothing to the input values
        _posX = _posX * (1 - _smoothingFactor) + event.y *
            _smoothingFactor;
        _posY = _posY * (1 - _smoothingFactor) + event.x *
            _smoothingFactor;

        final double maxDistance = 30;

        // Calculate distance from center using Pythagoras
        final double distance = sqrt(pow(_posX, 2) + pow(_posY,
            2));

        // Limit movement to a maximum distance using
            trigonometry
        if (distance > maxDistance) {
          final double angle = atan2(_posY, _posX);
          _posX = maxDistance * cos(angle);
          _posY = maxDistance * sin(angle);
        }
      });
    }, onError: (e) {
      print("Gyroscope error: $e");
    });
  }

  @override
  Widget build(BuildContext context) {
    final size = MediaQuery.of(context).size;

    return Scaffold(
      body: Center(
        child: Stack(
          children: [
            // Red circular area (boundary)
            Positioned(
              top: (size.height - _blueCircleSize) / 2,
              left: (size.width - _blueCircleSize) / 2,
              child: Container(
                height: _blueCircleSize,
                width: _blueCircleSize,
                decoration: const BoxDecoration(
                  color: Colors.lightBlue,
                  shape: BoxShape.circle,
                ),
              ),
            ),
            // Blue moving widget
```

```
90              Positioned(
91                top: (size.height - _whiteCircleSize) / 2 + _posY *
                      _sensitivity,
92                left: (size.width - _whiteCircleSize) / 2 + _posX *
                      _sensitivity,
93                child: Container(
94                  height: _whiteCircleSize,
95                  width: _whiteCircleSize,
96                  decoration: const BoxDecoration(
97                    color: Colors.white,
98                    shape: BoxShape.circle,
99                  ),
100               ),
101             ),
102           ],
103         ),
104       ),
105     );
106   }
107 }
```