

# Acelerômetros

**N**este módulo, vamos introduzir os fundamentos dos acelerômetros. Discutiremos o que são acelerômetros, como funcionam e suas principais aplicações. Acelerômetros são sensores que medem a aceleração linear em uma ou mais direções. Eles são amplamente utilizados em dispositivos móveis para detectar movimento, inclinação e orientação. Esta aula fornecerá uma base sólida sobre os conceitos básicos dos acelerômetros e suas funções essenciais.

## Fundamentos de Acelerômetros

### O que é um Acelerômetro?

Um acelerômetro é um sensor que mede a aceleração linear em uma ou mais direções. A aceleração é a taxa de mudança da velocidade de um objeto. Acelerômetros podem medir a força da gravidade e movimentos dinâmicos, como vibrações ou movimentos rápidos.

### Como Funcionam os Acelerômetros?

Os acelerômetros contêm pequenos componentes que se movem em resposta à aceleração. Quando o dispositivo se move, esses componentes se deslocam, e o sensor converte esse movimento em sinais elétricos que podem ser medidos.

### Aplicações Comuns dos Acelerômetros

- **Deteccção de Movimento:** Utilizado em dispositivos móveis para detectar movimentos e gestos.

- **Inclinação e Orientação:** Determina a inclinação e orientação do dispositivo.
- **Atividades Físicas:** Medição de passos e atividades físicas em smartwatches e dispositivos de fitness.
- **Automotivo:** Monitoramento de segurança em airbags e sistemas de estabilidade.

### Benefícios dos Acelerômetros

- **Precisão:** Fornece medições precisas de movimento e orientação.
- **Versatilidade:** Pode ser usado em várias aplicações, desde dispositivos móveis até equipamentos industriais.
- **Baixo Consumo de Energia:** Ideal para dispositivos móveis devido ao seu baixo consumo de energia.

Nesta aula, aprenderemos como acessar dados de acelerômetros em aplicativos Flutter. Utilizaremos o pacote `sensor_plus` para obter leituras de acelerômetros e integrar esses dados em nosso aplicativo. Abordaremos a instalação do pacote, a configuração necessária e como exibir dados do acelerômetro em tempo real na interface do usuário.

## **Acessando Acelerômetros com Flutter**

### Passos para Acessar Acelerômetros

#### **1. Adicionar Dependências ao `pubspec.yaml`**

dependencies:

flutter:

  sdk: flutter

sensors\_plus: ^2.0.0

## 2. Importar o Pacote sensors\_plus

```
import 'package:flutter/material.dart';
```

```
import 'package:sensors_plus/sensors_plus.dart';
```

## 3. Código para Acessar Dados do Acelerômetro

```
import 'package:flutter/material.dart';
```

```
import 'package:sensors_plus/sensors_plus.dart';
```

```
void main() {
```

```
  runApp(MyApp());
```

```
}
```

```
class MyApp extends StatelessWidget {
```

```
  @override
```

```
  Widget build(BuildContext context) {
```

```
    return MaterialApp(
```

```
      home: SensorScreen(),
```

```
    );
```

```
  }
```

```
}
```

```
class SensorScreen extends StatefulWidget {
```

@override

\_SensorScreenState createState() => \_SensorScreenState();

}

class \_SensorScreenState extends State<SensorScreen> {

double x = 0.0, y = 0.0, z = 0.0;

@override

void initState() {

super.initState();

accelerometerEvents.listen((AccelerometerEvent event) {

setState(() {

x = event.x;

y = event.y;

z = event.z;

});

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

```
appBar: AppBar(  
  
  title: Text('Dados do Acelerômetro'),  
  
),  
  
body: Center(  
  
  child: Column(  
  
    mainAxisAlignment: MainAxisAlignment.center,  
  
    children: [  
  
      Text('Eixo X: $x'),  
  
      Text('Eixo Y: $y'),  
  
      Text('Eixo Z: $z'),  
  
    ],  
  
  ),  
  
),  
  
);  
  
}
```

Explicação: Este código configura um listener para eventos do acelerômetro e atualiza a interface do usuário com os valores do eixo X, Y e Z em tempo real.

Nesta aula, veremos algumas aplicações práticas de acelerômetros em aplicativos Flutter. Aprenderemos como usar dados de acelerômetros para criar funcionalidades interativas, como detecção de movimento e orientação do dispositivo. Discutiremos também alguns exemplos práticos de uso de acelerômetros em jogos e aplicativos de fitness.

## **Aplicações Práticas de Acelerômetros**

### Exemplo 1: Detecção de Movimento

#### **1. Código para Detecção de Movimento**

```
import 'package:flutter/material.dart';

import 'package:sensors_plus/sensors_plus.dart';

void main() {

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: MotionDetectionScreen(),

    );

  }

}
```

```
}
```

```
class MotionDetectionScreen extends StatefulWidget {
```

```
  @override
```

```
    _MotionDetectionScreenState createState() =>  
    _MotionDetectionScreenState();
```

```
}
```

```
class _MotionDetectionScreenState extends  
State<MotionDetectionScreen> {
```

```
  String motionStatus = "Nenhum movimento detectado";
```

```
  @override
```

```
  void initState() {
```

```
    super.initState();
```

```
    accelerometerEvents.listen((AccelerometerEvent event) {
```

```
      if (event.x > 1 || event.y > 1 || event.z > 1) {
```

```
        setState(() {
```

```
          motionStatus = "Movimento detectado!";
```

```
        });
```

```
      } else {
```

```
        setState(() {
```

```
          motionStatus = "Nenhum movimento detectado";
```

```

    });

    }

});

}

@override

Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

            title: Text('Detecção de Movimento'),

        ),

        body: Center(

            child: Text(motionStatus),

        ),

    );

}

}

```

Explicação: Este código detecta movimento baseado nos valores do acelerômetro e atualiza a interface do usuário para exibir “Movimento detectado!” quando um movimento significativo é detectado.

**Exemplo 2:** Controle de Jogo por Inclinação



## 1. Código para Controle de Jogo

```
import 'package:flutter/material.dart';

import 'package:sensors_plus/sensors_plus.dart';

void main() {

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: GameControlScreen(),

    );

  }

}

class GameControlScreen extends StatefulWidget {

  @override

  _GameControlScreenState createState() => _GameControlScreenState();

}

class _GameControlScreenState extends State<GameControlScreen> {
```

```
double x = 0.0, y = 0.0;
```

```
@override
```

```
void initState() {
```

```
    super.initState();
```

```
    accelerometerEvents.listen((AccelerometerEvent event) {
```

```
        setState(() {
```

```
            x = event.x;
```

```
            y = event.y;
```

```
        });
```

```
    });
```

```
}
```

```
@override
```

```
Widget build(BuildContext context) {
```

```
    return Scaffold(
```

```
        appBar: AppBar(
```

```
            title: Text('Controle de Jogo por Inclinação'),
```

```
        ),
```

```
        body: Center(
```

```
            child: Transform.translate(
```

```

offset: Offset(x * 10, y * 10),

child: Container(

  width: 50,

  height: 50,

  color: Colors.blue,

),

),

),

);

}

}

```

Explicação: Este código usa os dados do acelerômetro para controlar a posição de um quadrado azul na tela, movendo-o conforme a inclinação do dispositivo.

Nesta aula, discutiremos o tratamento e a filtragem de dados de sensores em Flutter. Aprenderemos como aplicar filtros para suavizar os dados do acelerômetro, removendo ruídos e obtendo leituras mais estáveis. Veremos também técnicas para tratar dados de sensores para garantir precisão e consistência.

## **Tratamento e Filtragem de Dados de Sensores**

### Aplicando Filtros nos Dados do Acelerômetro

## 1. Filtro de Média Móvel

```
import 'package:flutter/material.dart';

import 'package:sensors_plus/sensors_plus.dart';

void main() {

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: FilteredSensorScreen(),

    );

  }

}

class FilteredSensorScreen extends StatefulWidget {

  @override

  _FilteredSensorScreenState createState() =>

  _FilteredSensorScreenState();

}
```

```
class _FilteredSensorScreenState extends State<FilteredSensorScreen> {
```

```
  double x = 0.0, y = 0.0, z = 0.0;
```

```
  List<double> xValues = [], yValues = [], zValues = [];
```

```
  int windowSize = 10;
```

```
  @override
```

```
  void initState() {
```

```
    super.initState();
```

```
    accelerometerEvents.listen((AccelerometerEvent event) {
```

```
      xValues.add(event.x);
```

```
      yValues.add(event.y);
```

```
      zValues.add(event.z);
```

```
      if (xValues.length > windowSize) {
```

```
        xValues.removeAt(0);
```

```
        yValues.removeAt(0);
```

```
        zValues.removeAt(0);
```

```
      }
```

```
    setState(() {
```

```
      x = xValues.reduce((a, b) => a + b) / xValues.length;
```

```
      y = yValues.reduce((a, b) => a + b) / yValues.length;
```

```
z = zValues.reduce((a, b) => a + b) / zValues.length;
```

```
});
```

```
});
```

```
}
```

```
@override
```

```
Widget build(BuildContext context) {
```

```
  return Scaffold(
```

```
    appBar: AppBar(
```

```
      title: Text('Dados Filtrados do Acelerômetro'),
```

```
    ),
```

```
    body: Center(
```

```
      child: Column(
```

```
        mainAxisAlignment: MainAxisAlignment.center,
```

```
        children: [
```

```
          Text('Eixo X: $x'),
```

```
          Text('Eixo Y: $y'),
```

```
          Text('Eixo Z: $z'),
```

```
        ],
```

```
      ),
```

```
),  
  
);  
  
}  
  
}
```

Explicação: Este código aplica um filtro de média móvel aos dados do acelerômetro para suavizar as leituras, reduzindo o ruído e proporcionando valores mais estáveis.

## **Tratamento de Dados para Precisão**

### **1. Tratamento de Dados para Precisão**

```
import 'package:flutter/material.dart';  
  
import 'package:sensors_plus/sensors_plus.dart';  
  
void main() {  
  
  runApp(MyApp());  
  
}  
  
class MyApp extends StatelessWidget {  
  
  @override  
  
  Widget build(BuildContext context) {  
  
    return MaterialApp(  
  
      home: PreciseSensorScreen(),  
  
    );  
  
}
```

```
}
```

```
}
```

```
class PreciseSensorScreen extends StatefulWidget {
```

```
  @override
```

```
    _PreciseSensorScreenState createState() =>
```

```
    _PreciseSensorScreenState();
```

```
}
```

```
class _PreciseSensorScreenState extends State<PreciseSensorScreen> {
```

```
  double x = 0.0, y = 0.0, z = 0.0;
```

```
  @override
```

```
  void initState() {
```

```
    super.initState();
```

```
    accelerometerEvents.listen((AccelerometerEvent event) {
```

```
      setState(() {
```

```
        x = _treatData(event.x);
```

```
        y = _treatData(event.y);
```

```
        z = _treatData(event.z);
```

```
      });
```

```
    });
```



```
}
```

```
double _treatData(double value) {
```

```
    // Aplicar tratamento de dados, por exemplo, arredondamento
```

```
    return double.parse(value.toStringAsFixed(2));
```

```
}
```

```
@override
```

```
Widget build(BuildContext context) {
```

```
    return Scaffold(
```

```
        appBar: AppBar(
```

```
            title: Text('Tratamento de Dados do Acelerômetro'),
```

```
        ),
```

```
        body: Center(
```

```
            child: Column(
```

```
                mainAxisAlignment: MainAxisAlignment.center,
```

```
                children: [
```

```
                    Text('Eixo X: $x'),
```

```
                    Text('Eixo Y: $y'),
```

```
                    Text('Eixo Z: $z'),
```

```
                ],
```

```
    ),  
  
    ),  
  
);  
  
}  
  
}
```

Explicação: Este código aplica um tratamento simples de arredondamento aos dados do acelerômetro para melhorar a precisão das leituras exibidas.

Esses exemplos e explicações fornecem uma base sólida para iniciantes em Flutter aprenderem a integrar e usar acelerômetros em seus aplicativos. Para mais detalhes, consulte a documentação oficial do Flutter: [Flutter Documentation](#).

## **Materiais Extras**

Você pode realizar o download do arquivo contendo os materiais extras utilizados ao longo das aulas por meio do seguinte link: <https://drive.google.com/file/d/1mg7lqMI8Pt2zl0rHlsFS0Qew00YN-sEX/view?usp=sharing>.

## **Conteúdo Bônus**

Para aprofundar seus conhecimentos em Desenvolvimento Mobile com foco na utilização de acelerômetros, recomendo o seguinte recurso gratuito:

Curso “Desenvolvimento de Aplicativos Móveis com PHONEGAP e PHP”: Este curso aborda a criação de aplicativos móveis utilizando o PhoneGap,

permitindo o acesso a funcionalidades nativas dos dispositivos, como o acelerômetro, por meio de linguagens como HTML, CSS e JavaScript.

## **Referências Bibliográficas**

BOYLESTAD, R. L.; NASHELSKY, L. Dispositivos Eletrônicos e Teoria de Circuitos. 11. ed. Pearson, 2013.

DEITEL, P. J.; DEITEL, H. M. Ajax, Rich Internet Applications e Desenvolvimento Web para Programadores. Pearson, 2008.

DUARTE, W. Delphi para Android e iOS: Desenvolvendo Aplicativos Móveis. Brasport, 2015.

FELIX, R.; SILVA, E. L. da. Arquitetura para Computação Móvel. 2. ed. Pearson, 2019.

LEE, V.; SCHNEIDER, H.; SCHELL, R. Aplicações Móveis: Arquitetura, Projeto e Desenvolvimento. Pearson, 2005.

MARINHO, A. L.; CRUZ, J. L. da. Desenvolvimento de Aplicações para Internet. 2. ed. Pearson, 2019.

MOLETTA, A. Você na Tela: Criação Audiovisual para a Internet. Summus, 2019.

SILVA, D. (Org.) Desenvolvimento para dispositivos móveis. Pearson, 2017.

**Ir para exercício**