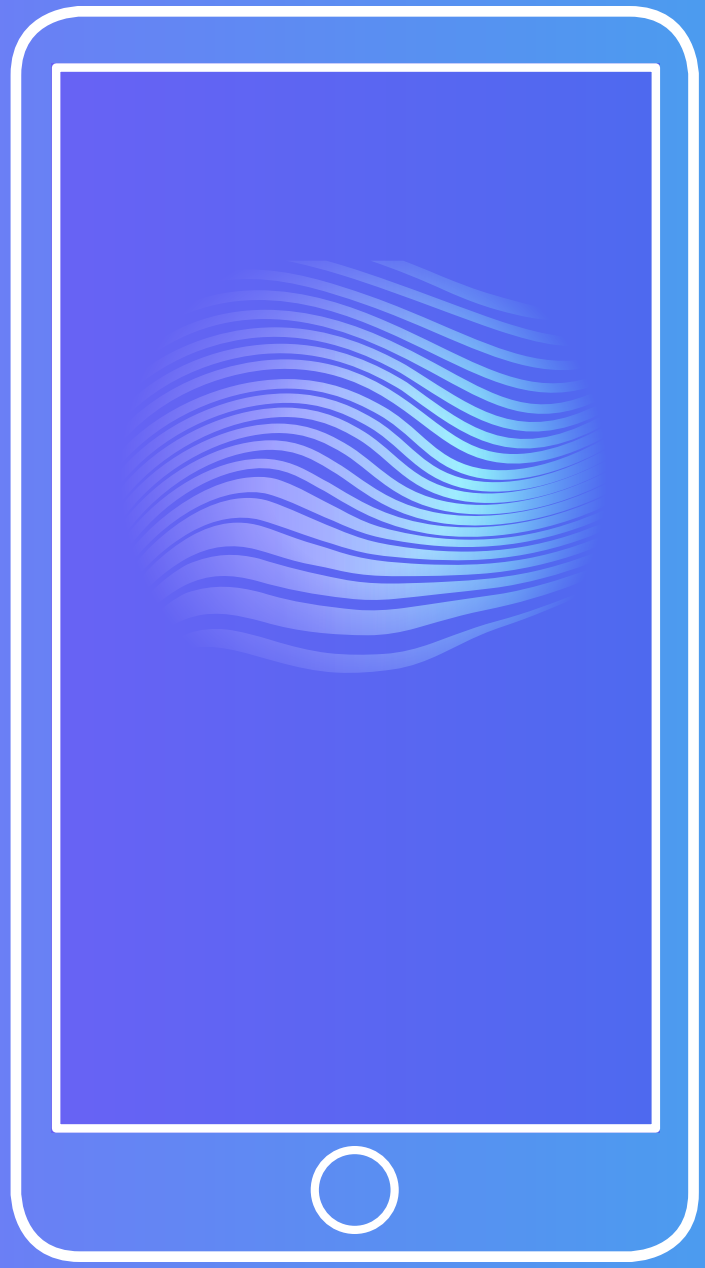
DON'T FORGET TO
BRING YOUR PHONE!

Forschungsseminar

android + sensoren

07

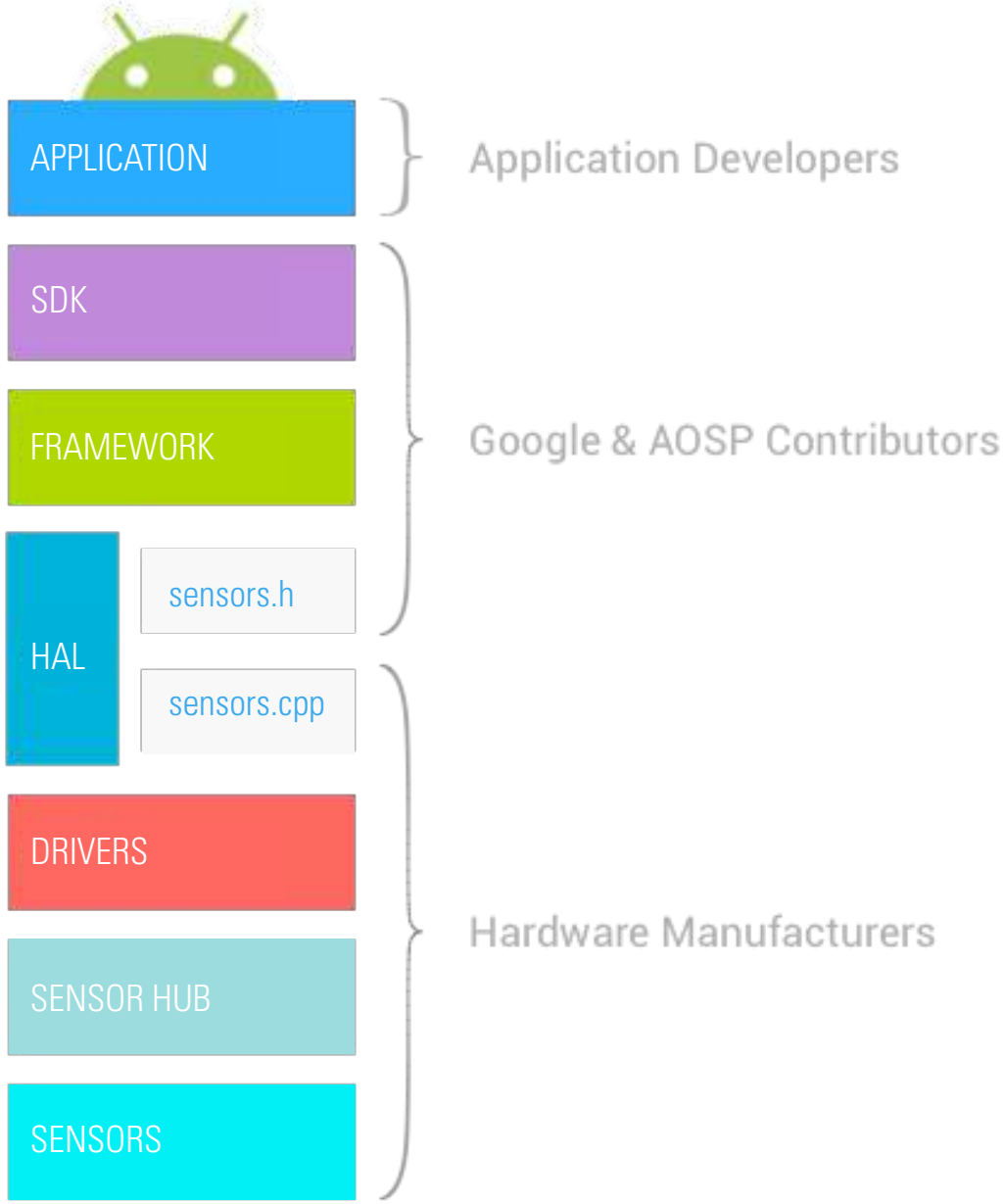
Carolin Scholl & Anne Münzner



INHALT

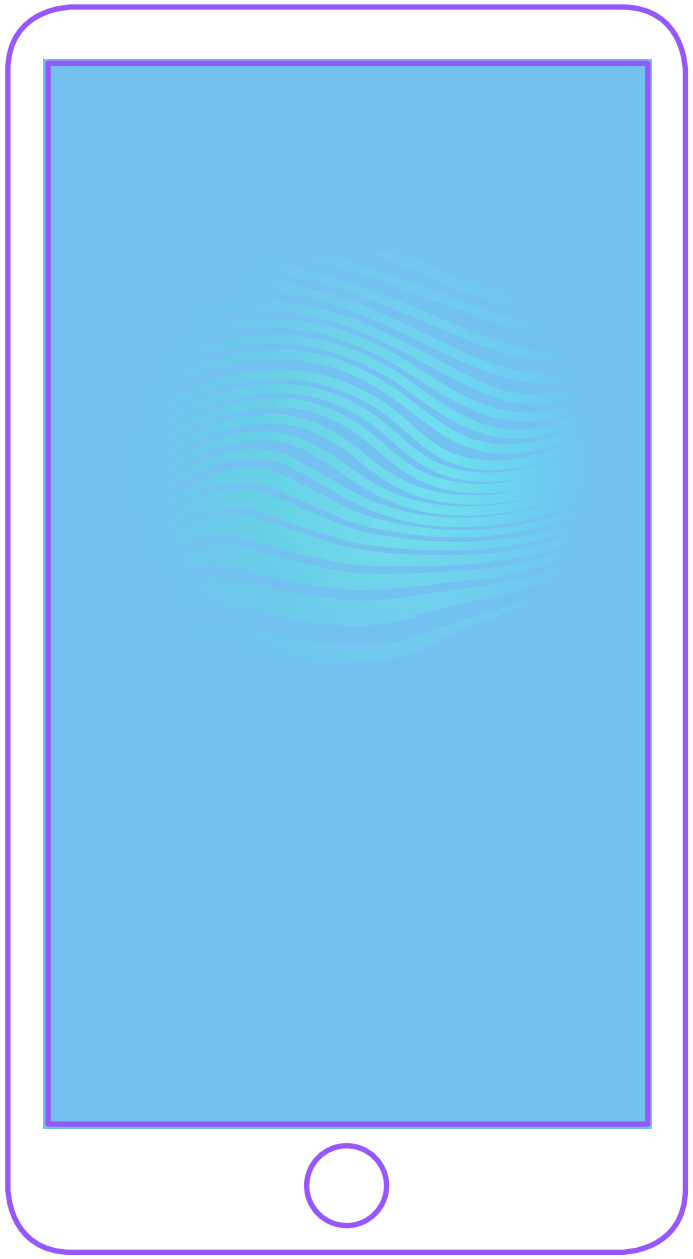
01 android sensors

- geben Anwendung Zugang zu physikalischen Sensoren
- unterschiedliche Sensortypen, die bestimmen wie sich ein Sensor verhält und welche Daten er zurückgibt
- stellen Daten in einer Reihe von Sensor Events bereit
- Android sensor stack beschreibt Schichtmodell für Sensorennutzung



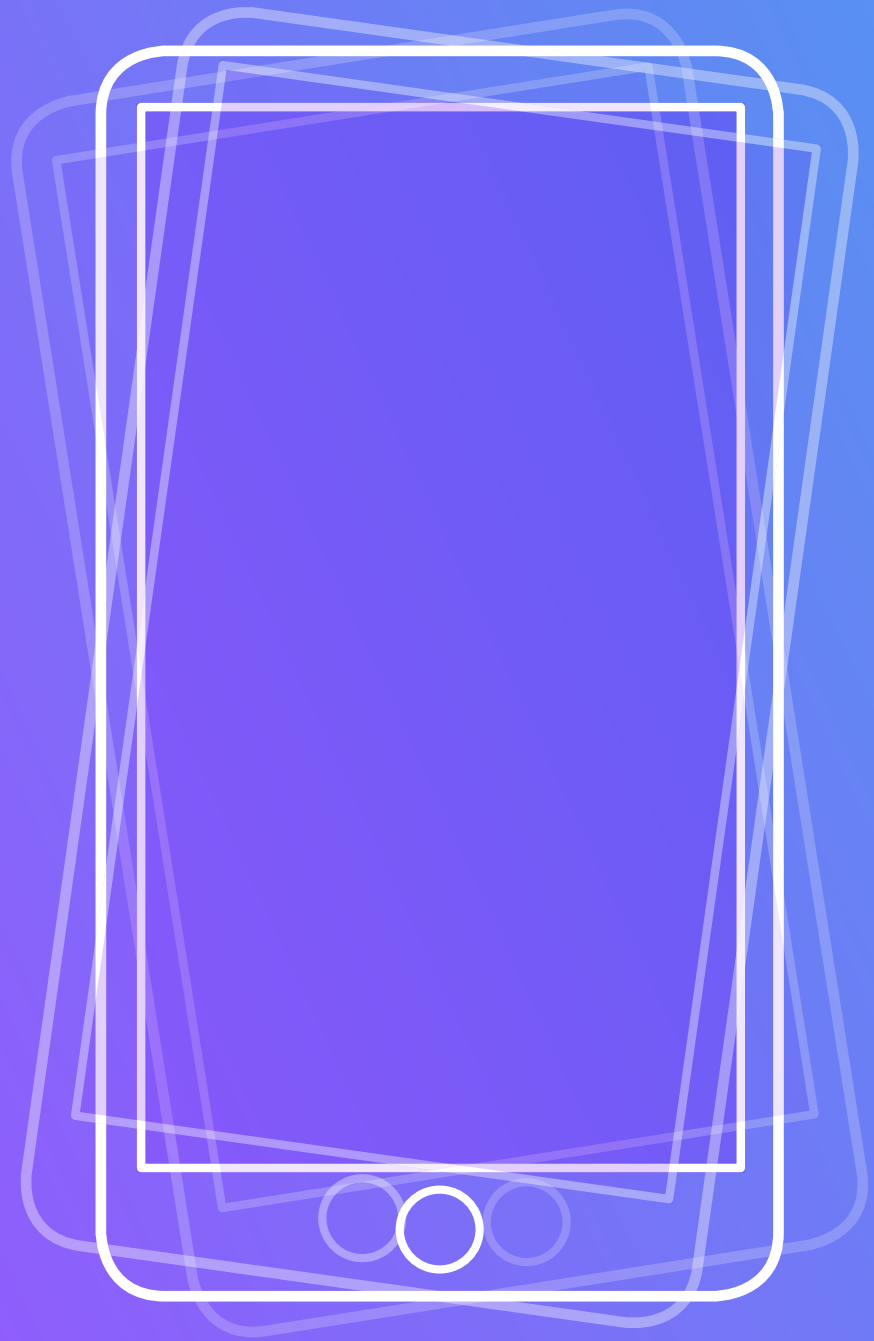
01 Sensorentypen

- motion
- environment
- position



01 Motion Sensors

- Erfassen von Bewegungen wie Schütteln, Drehen, Kippen
- multidimensionale Arrays als Rückgabe, z.B. Daten für drei verschiedene Koordinatenachsen



01 Environment Sensors

- Reaktion auf Veränderungen im Umfeld des Smartphones: Licht, Luftfeuchtigkeit, Temperatur, Luftdruck
- hardwarebasiert
- optional
- einzelner Wert als Rückgabe, z.B Temperatur in C°

01 Position Sensors

- zur Messung der physikalischen Position eines Geräts
  - Magnetfeld der Erde
  - Orientierung
  - Nähe zu einem Objekt (cm)
- Orientierung: Kombination von Magnetometer und Beschleunigungssensor
- Nähe- und geomagnetischer Feldsensor sind hardware-basiert

01 Location Strategies

- Lokalisierungsmethoden gehören weder zu Position Sensors, noch zu Sensor Framework
- GPS fällt unter Location Strategies API android:location
- Lokalisierungsdaten fehleranfällig

01 Location Manager

```
LocationManager lm = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
LocationListener locationListener = new MyLocationListener();
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000, 10, locationListener);
```

- Android empfiehlt Nutzung von Google Location Services API
- LocationManager statt SensorManager
- Wichtig: Vor Lokalisierung Nutzererlaubnis einholen!

```
<manifest ... >
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-feature android:name="android.hardware.location.gps" />
...
</manifest>
```

01 Exkurs: permissions

- Alle permissions müssen im Manifest der App gelistet werden

normal permissions

Riskieren nicht direkt die Privatsphäre des Nutzers

z.B. Internet, Vibrationsfunktion

Wenn die App eine normale Permission im Manifest listet, gibt das System sie automatisch frei

dangerous permissions

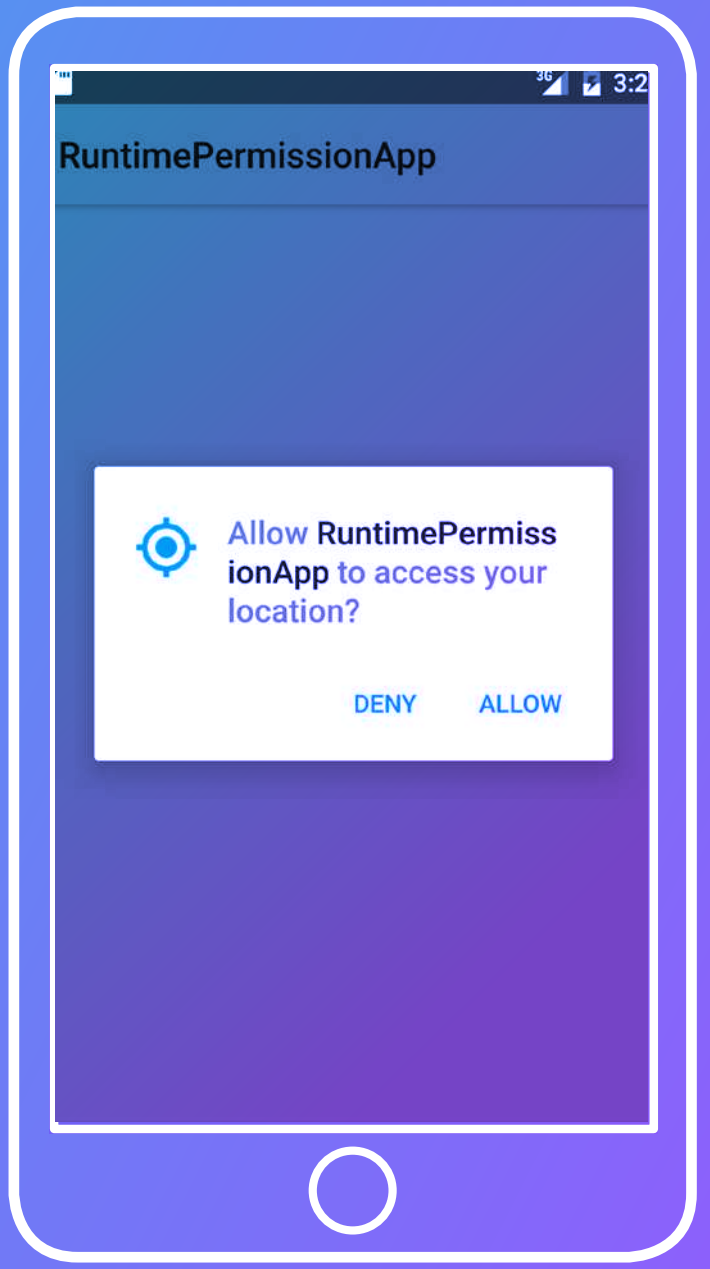
Können der App Zugriff auf sensible Daten des Nutzers ermöglichen

z.B. Kamera, GPS

Wenn die App eine dangerous Permission im Manifest listet, muss Nutzer explizit Erlaubnis erteilen
- vor Android 6.0: bei Installation
- seit Android 6.0: Dialog während Nutzung

01 Exkurs: dangerous permissions

- Bei App mit Dangerous permissions: ab Android 6.0 muss jede einzelne Permission bei Nutzung der App abgefragt und erteilt werden
- Nutzer hat das Recht alle oder einzelne abzulehnen: App kann mit reduzierter Funktionalität weiterlaufen
- Zum Umgang mit Permissions und zur Gestaltung solcher Dialoge empfiehlt Android die Support Library



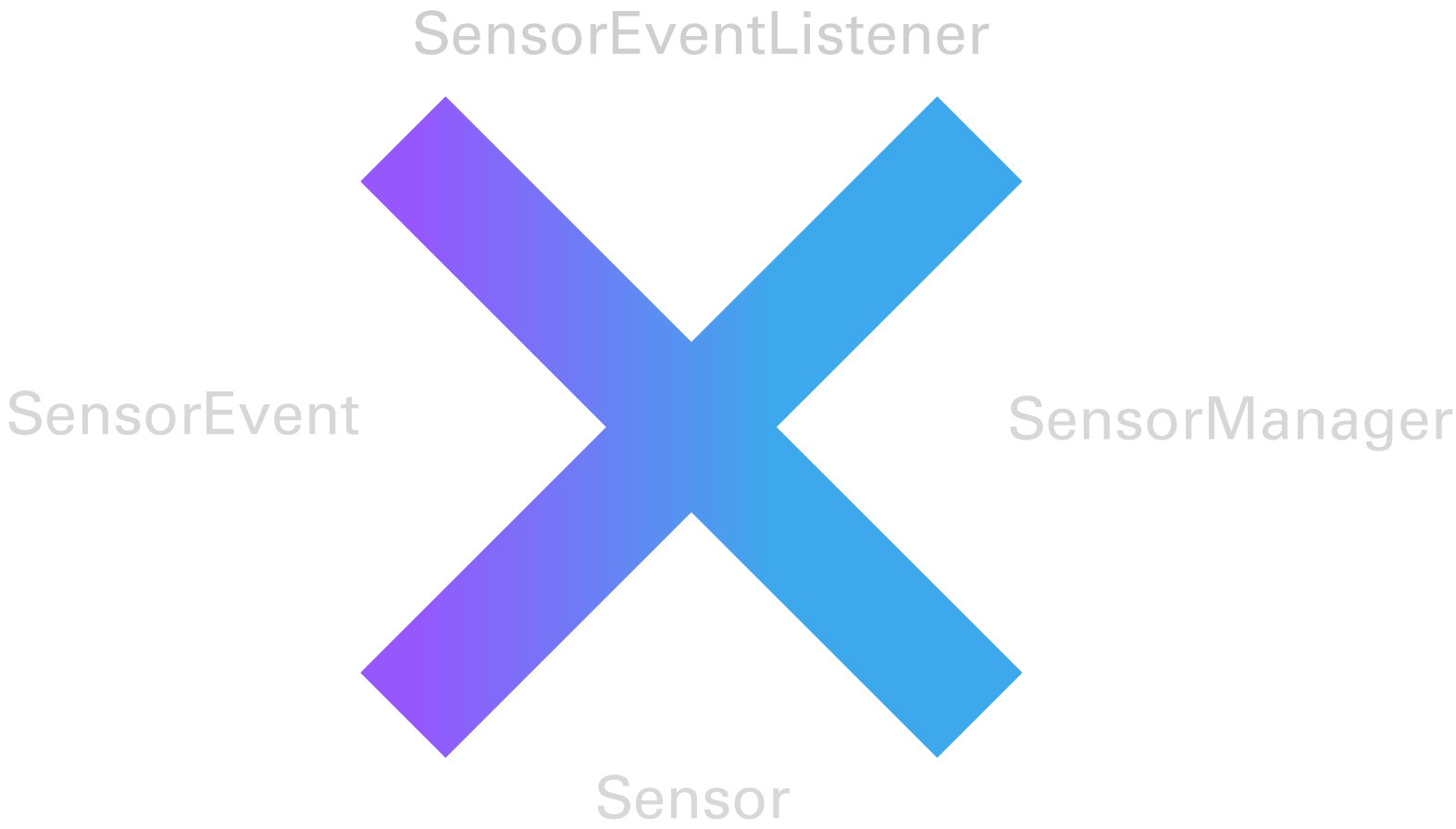
PRAXIS Sensorenliste ausgeben

Main Activity:

```
public boolean createSensorList(){ ... }
```

02 Sensor Framework

- Teil des android.hardware package
- Erlaubt Zugriff auf Sensoren eines Geräts
- Stellt Interfaces und Java-Classes zur Verfügung, um Sensordaten verarbeiten zu können



02 Sensor Manager

```
SensorManager sm = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
```

- Klasse android.hardware.SensorManager
- Zur Kommunikation mit den Sensoren
- Enthält Methoden, um auf Sensoren zuzugreifen, sie an- und abzumelden
- Enthält Konstanten zum mittleren Messfehler und der gewünschten Abtastrate

02 Sensor

```
Sensor acc = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

- Klasse android.hardware.Sensor
- Enthält Methoden, um Informationen über diesen Sensor zu erhalten, z.B. Messbereich, Auflösung, Stromverbrauch usw.
- Zum Instanziieren eines spezifischen Sensors
- Außerdem sind in dieser Klasse die verschiedenen Arten von Sensoren definiert, z.B. TYPE ACCELEROMETER

02 Sensor Event Listener

Registrierungsmethoden für den SensorEventListener
in der Klasse Sensor Manager

```
public boolean registerListener(SensorEventListener listener,
Sensor sensor, int rate){
// listener bekommt von sensor Daten mit Änderungsrate rate
// true, wenn Sensor verfügbar ist
}
public void unregisterListener(SensorEventListener listener){
// listener von Sensoren abmelden
}
```

02 Activity Lifecycle

- Sensoren können viel Akku beanspruchen
- Deshalb ist es sinnvoll, den SensorEventListener während Pausierung einer Activity abzumelden und bei Rückkehr wieder anzumelden

```
protected void onPause() {
super.onPause();
sm.unregisterListener(this);
}

protected void onResume() {
super.onResume();
sm.registerListener(this, acc,
SensorManager.SENSOR_DELAY_NORMAL);
}
```

02 Sensor Event Listener Interface

```
public void onSensorChanged(SensorEvent sensorEvent){
// wird gerufen, wenn ein neuer Sensorwert ankommt
}

public void onAccuracyChanged(Sensor sensor, int accuracy) {
// wird gerufen, wenn sich die Genauigkeit des Sensors ändert
}
```

- onSensorChanged()-Methode wird extrem häufig aufgerufen.
- Für viele Anwendungszwecke ist eine solch hohe Rate nicht nötig
- Besser nur eine Stichprobe der SensorEvents verarbeiten (z.B. nach konstantem Zeitintervall)
- Methode nicht „überfordern“. Berechnungen mit Sensor-daten lieber in gesonderter Methode durchführen

02 Sensor Event

- Klasse android.hardware.SensorEvent
- Stellt die Sensordaten zur Verfügung

```
public Sensor sensor
public long timestamp
public final float[] values
public int accuracy
```

PRAXIS Sensor initialisieren

Manifest:

```
<uses-feature
android:name="android.hardware.sensor.accelerometer"
android:required="true" />
```

Main Activity:

```
public boolean initiateSensor(){ ... }
```

03 Bewegungssen-soren im Fokus

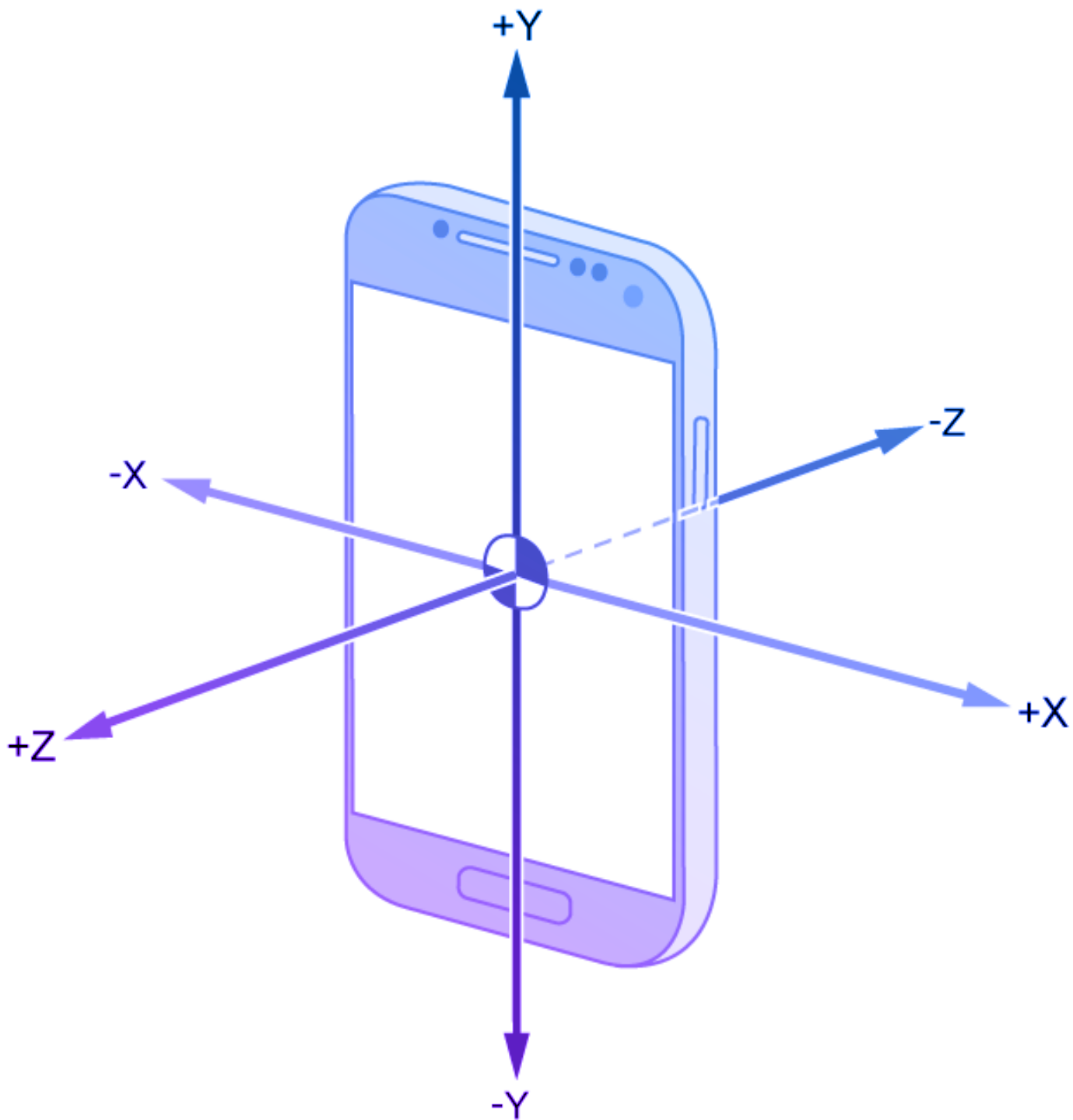
- Für unterschiedliche Bewegungen eignen sich unterschiedliche Sensoren oder deren Kombination

03 Beschleunigung

- Beschleunigung = Änderung der Geschwindigkeit pro Zeiteinheit
- Erste Ableitung der Funktion v(t)
- Einheit: m/s2
- Beschleunigungssensor arbeitet mit Kondensatoren
- Kapazitätsänderung dieser wird gemessen und in Beschleunigung umgerechnet

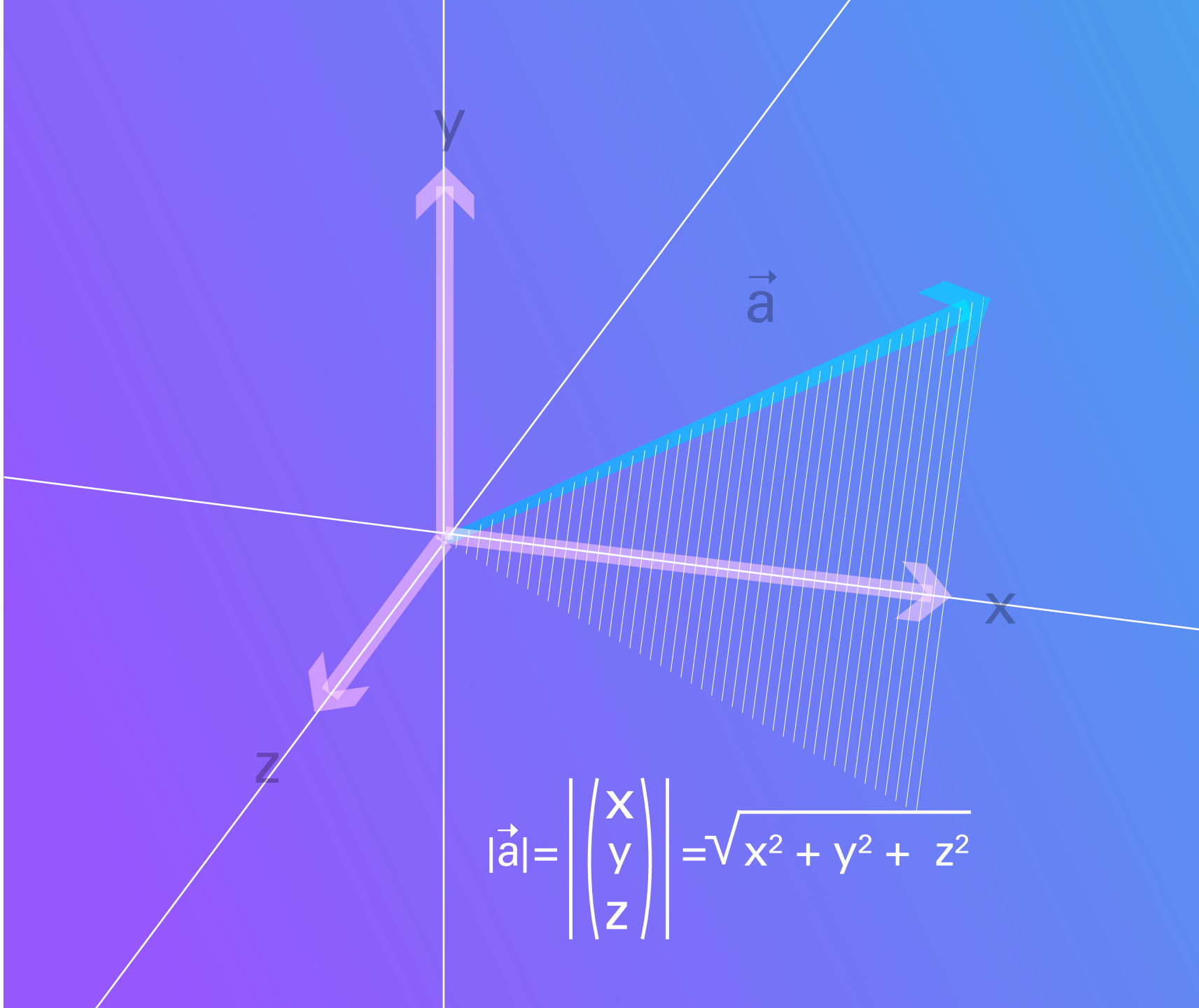
03 Beschleunigung

- Einzelner Beschleunigungssensor kann Beschleunigung nur in einer Richtung erfassen
- Deshalb besitzen Smartphones drei solcher Sensoren (x, y, z-Achse: Array values[ ] eines SensorEvents)
- Problem: Beschleunigung durch Gravitation nicht von Beschleunigung durch Bewegung unterscheidbar
- Bereinigen der Daten nötig



03 Länge eines Vektors

- Die Länge eines Vektors ist die Wurzel aus der Summe der Komponentenquadrate:



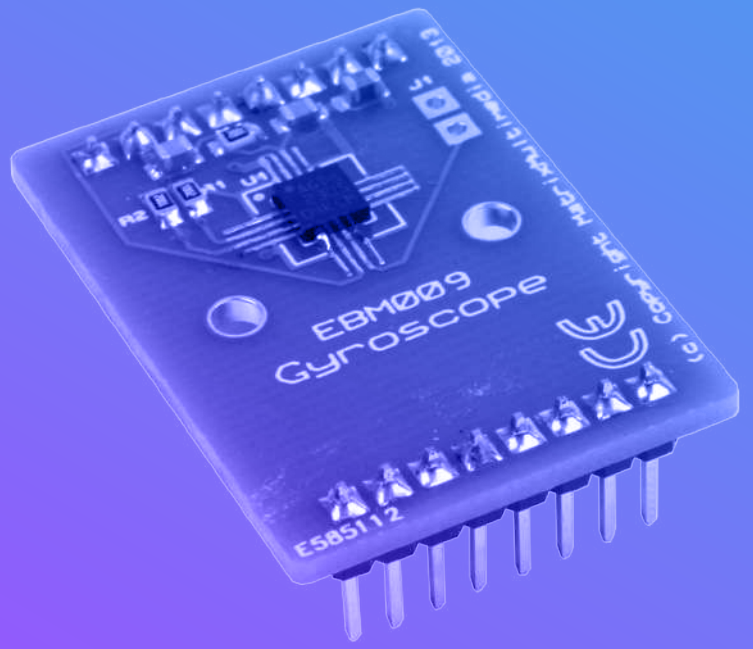
PRAXIS Beschleunigung berechnen

Main Activity:

```
public void CalculateAcc(SensorEvent sensorEvent) { ... }
```

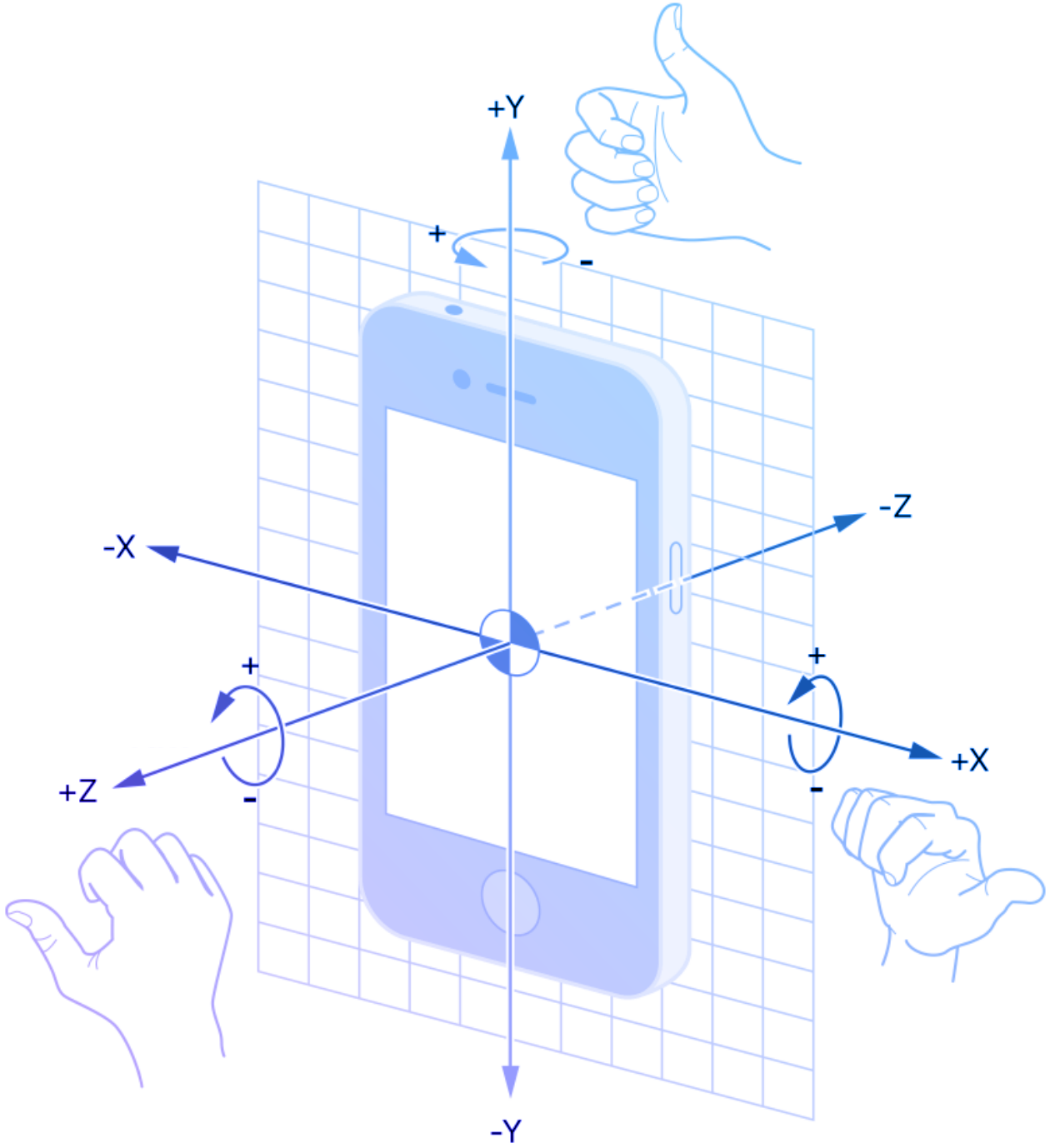
03 Rotation

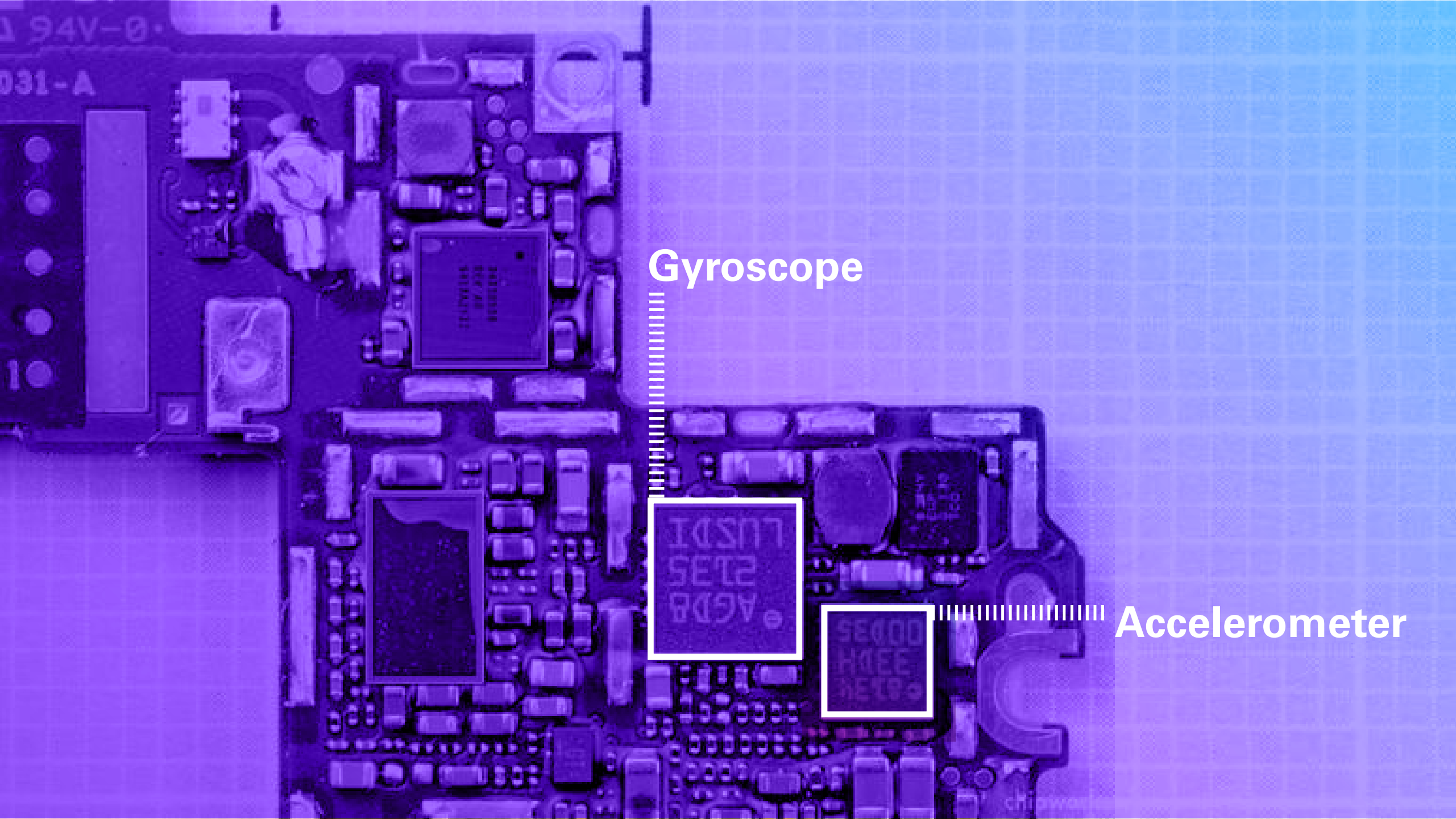
- Wir haben den Beschleunigungssensor zum Erfassen des Schütteln des Geräts verwendet
- Fast alle Smartphones haben einen solchen Beschleunigungssensor
- Nun wollen wir auch ein Drehen des Gerätes messen
- Dafür bietet sich ein Gyroskop an



03 Rotation

- Ein Gyroskop liefert im values[ ]-Array die Drehgeschwindigkeit entlang der drei Achsen
- Einheit °/s
- Problem: Nicht alle Smartphones haben ein Gyroskop
- Drehung ist jedoch auch mithilfe der Daten eines Beschleunigungssensors berechenbar




Gyroscope
Accelerometer

PRAXIS Rotation berechnen

Activity Cheers:

```
public void CalculateRot(SensorEvent sensorEvent) { ... }
```

CHEERS!

AND DON'T DRINK AND CODE...

04 Quellen

Verknüpfung Hardware/ Software
https://source.android.com/devices/sensors/ [05.06.2017]

Accelerometer und Gyroskop:
http://www.instructables.com/id/Accelerometer-Gyro-Tutorial/ [25.05.2017]

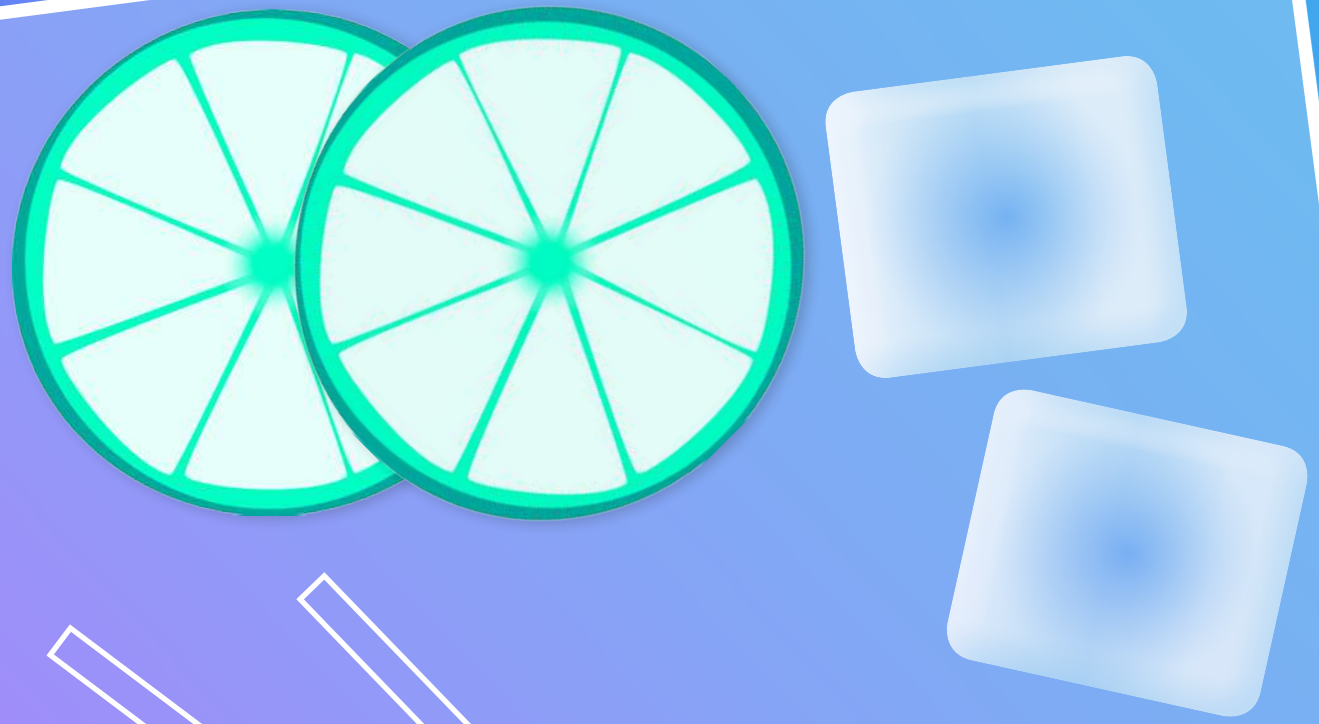
Überblick über Sensoren
https://developer.android.com/guide/topics/sensors/sensors_overview.html [30.05.2017]

Bewegungssensoren:
https://developer.android.com/guide/topics/sensors/sensors_motion.html [30.05.2017]

WaveView:
https://github.com/john990/WaveView [25.05.2017]

Material Design Guide for Android:
https://developer.android.com/guide/topics/sensors/sensors_motion.html [30.05.2017]

04 Tutorials



Liste der Sensoren:

http://www.android-examples.com/
get-all-list-of-available-sensors-inandroid-
device-programmatically/ [30.05.2017]

Accelerometer und Gyroskop:

https://code.tutsplus.com/tutorials/android-sen-
sors-in-depth-proximity-and-gyroscope--cms-28084
[05.06.2017]

Accelerometer und Orientierung:

http://www.vogella.com/tutorials/AndroidSensor/article.html
[05.06.2017]

GPS:

http://www.androidhive.info/2012/07/
android-gps-location-manager-tutorial/
[05.06.2017]