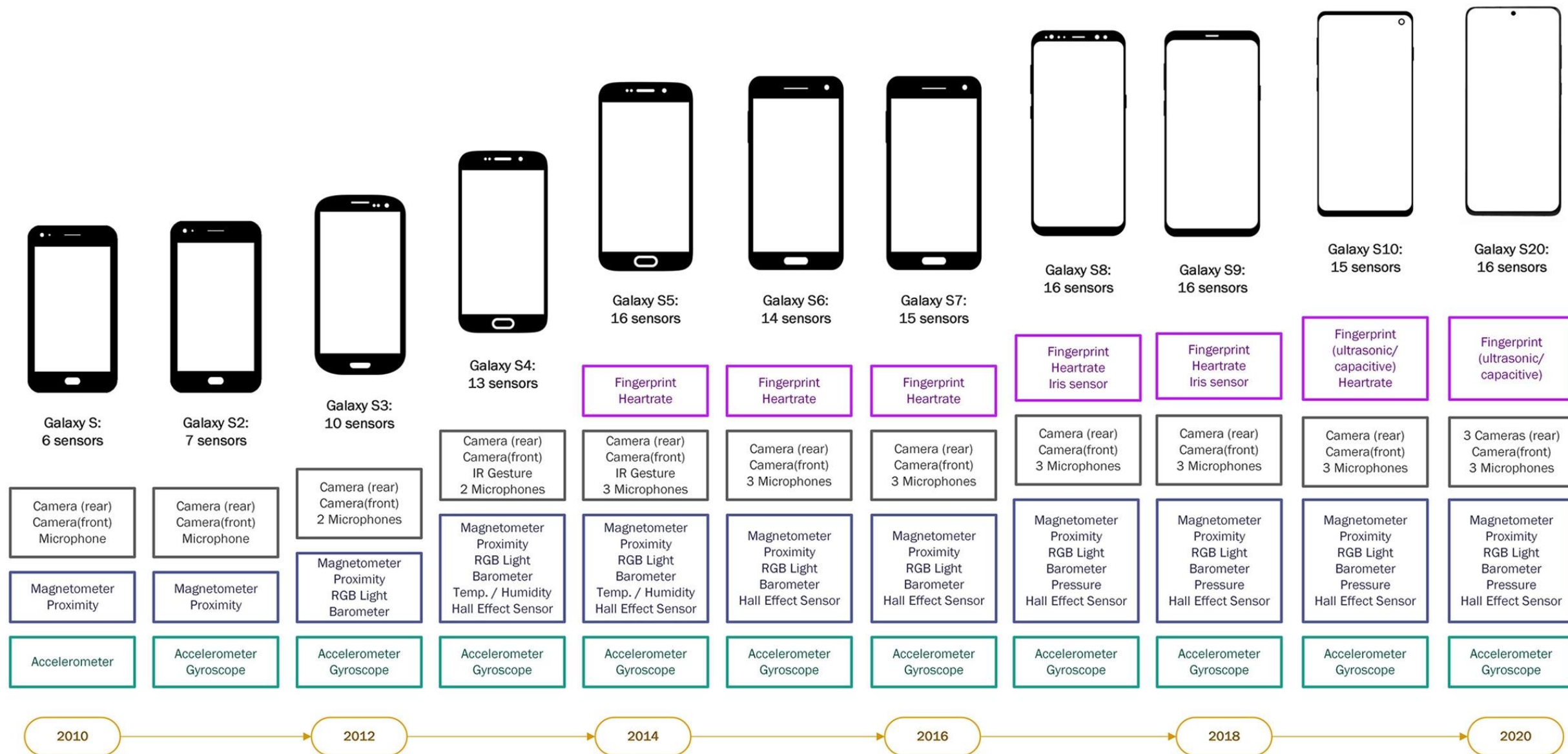


Les capteurs

- L'une des caractéristiques les plus intéressantes des smartphones réside dans leurs vastes possibilités d'exploration de l'environnement
- Au-delà des possibilités de communication, les capteurs disponibles et les *API* proposées permettent une interaction avec l'environnement
- Les capteurs peuvent être directement inclus dans le smartphone ou prendre la forme d'accessoires connectés, tels que des *wearables*
- Tous les terminaux ne possèdent pas forcément tous les capteurs pour lesquels une *API* existe

Toujours plus de capteurs sur les smartphones



Les capteurs de base

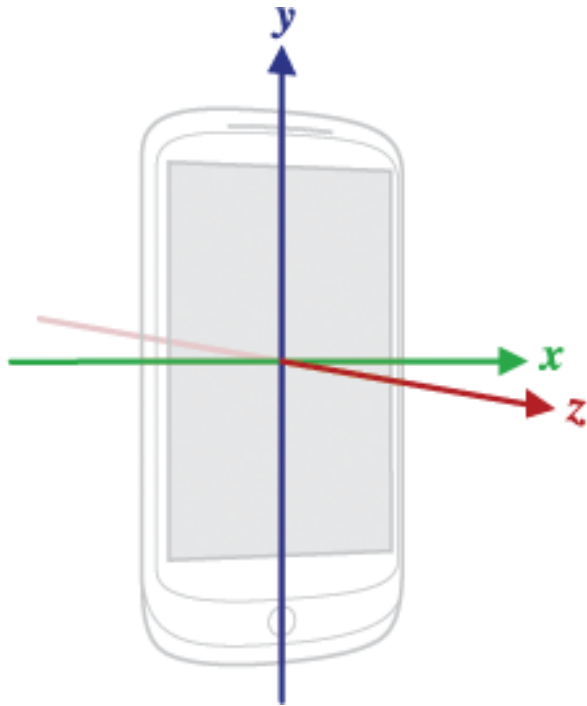
- Généralement basés sur un capteur physique, il y a tout de même une légère abstraction du système (correction, calibration, uniformisation)
- Tous les capteurs ne sont pas forcément disponibles sur tous les appareils
- Quelques exemples de capteurs :
 - Accéléromètre
 - Magnétomètre
 - Gyroscope
 - Luminosité
 - Proximité
 - Pression atmosphérique
 - Humidité
 - Température
 - Etc.

Les capteurs simples

- Luminosité [lx] Intensité lumineuse en face du smartphone
- Pression [hPa] Pression atmosphérique
- Proximité [cm] Distance avec un objet en face du smartphone
Généralement uniquement un résultat binaire (on/off),
utilisé pour désactiver l'écran tactile lors d'un appel
- Humidité [%] Humidité ambiante relative
- Température [°C] Température ambiante

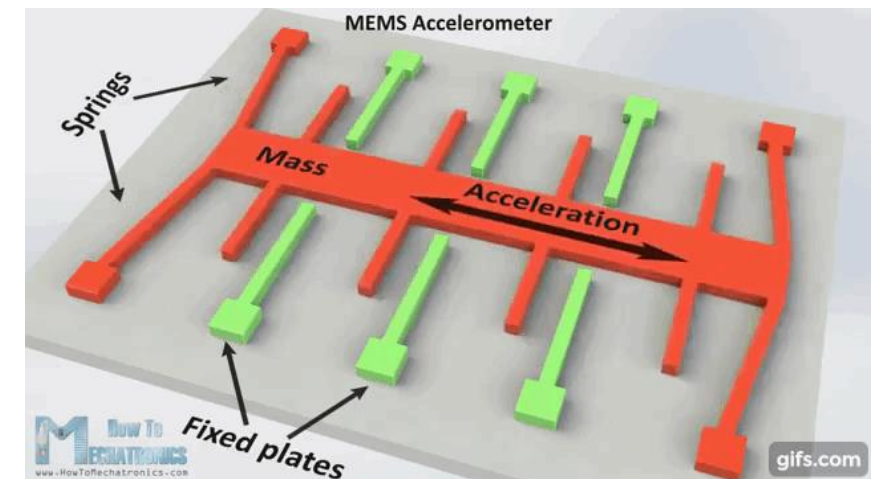
Les capteurs de mouvement

- Interprétation des données selon des axes :



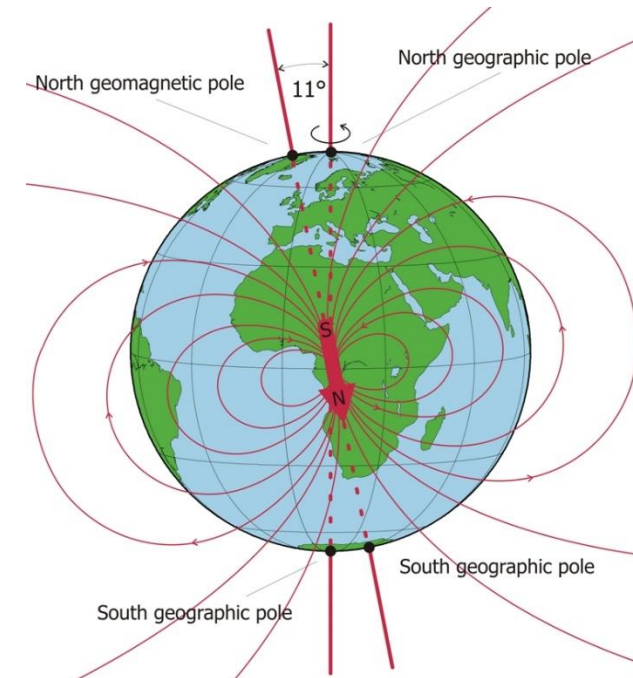
Les capteurs – Accéléromètre

- L'accéléromètre indique pour chaque axe l'accélération subie par le smartphone en $[m \cdot s^{-2}]$
- En situation «de repos», la quasi-totalité de l'accélération subie par le téléphone provient de la gravité terrestre. Il permet de détecter un basculement ou un pivotement
- Il permet aussi de détecter un mouvement, diverses applications utilisent ce dispositif pour déterminer :
 - La marche ou la course de l'utilisateur
 - Le fait de secouer son smartphone
 - L'immobilité
- La précision du dispositif est souvent insuffisante pour des applications de niveau professionnel



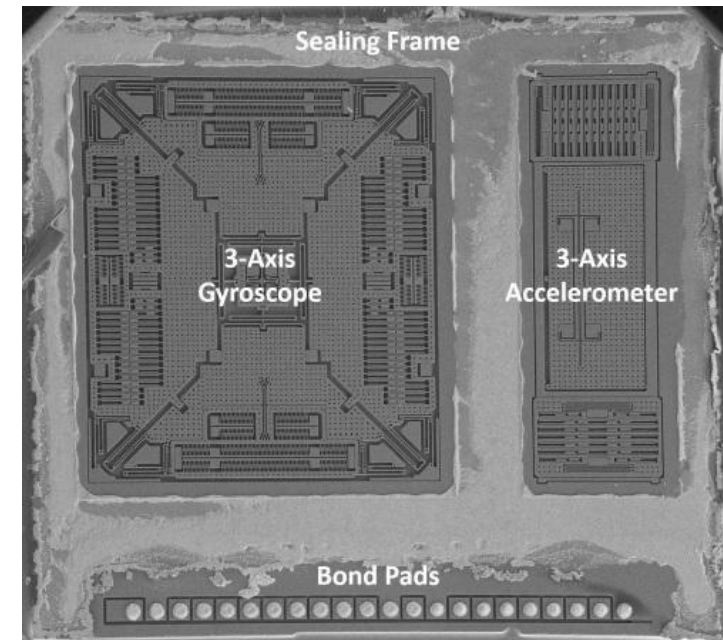
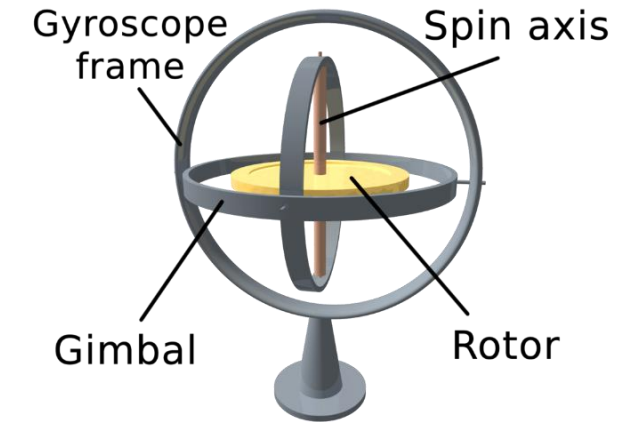
Les capteurs – Magnétomètre

- Le magnétomètre permet de connaître la position de l'appareil par rapport au champ magnétique
- Indique pour chaque axe l'intensité du champ magnétique en [μT]
- Principalement utilisé pour se positionner par rapport au pôle Nord magnétique
- Très facilement perturbé par des éléments externes: masse métallique ou aimant
- Sa précision n'est en général pas suffisante pour une utilisation professionnelle



Les capteurs – Gyroscope

- Disponibilité moindre que l'accéléromètre, en particulier sur les appareils d'entrée de gamme
- Consommation supérieure à l'accéléromètre
- Permet de détecter la rotation de l'appareil sur lui-même, vitesse de rotation sur les 3 axes en $[\text{rad} \cdot \text{s}^{-1}]$
- Plus rapide et précis que l'accéléromètre pour calculer une orientation
- Très utilisé pour les applications de réalité virtuelle



Les capteurs – Accès aux capteurs sur *Android*


- Tout comme le *LocationManager* unifie les accès aux différentes sources de géolocalisation, l'accès aux différents capteurs se fait par le *SensorManager*
- Le *SensorManager* propose une *API* unique permettant de lister les capteurs disponibles, de sélectionner un capteur particulier et de s'y inscrire pour obtenir des données, par exemple :

```
val sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
val accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)

// inscription au capteur
sensorManager.registerListener(listener, accelerometer, SensorManager.SENSOR_DELAY_NORMAL)

[...]

// désinscription
sensorManager.unregisterListener(listener)
```



SensorEventListener

Les capteurs – Accès aux capteurs sur *Android*

- Selon l'application que l'on souhaite réaliser, nous aurons besoin d'une précision temporelle plus ou moins importante, on doit spécifier le taux de rafraichissement désiré lors de l'inscription :
 - `SENSOR_DELAY_NORMAL`
 - `SENSOR_DELAY_UI`
 - `SENSOR_DELAY_GAME`
 - `SENSOR_DELAY_FASTEST`
- Depuis *Android 12*, l'accès aux capteurs avec une fréquence > 200 Hz nécessite des permissions particulières
- Il faut se désinscrire d'un capteur dès qu'il n'est plus nécessaire
- Les résultats étant reçus dans le *Thread-UI*, un taux trop important risque de ralentir l'*UI* inutilement

Les capteurs – Accès aux capteurs sur *Android*

- Le *SensorManager* va délivrer les mesures à une implémentation de *SensorEventListener* (par exemple l'*Activité*), qui doit mettre à disposition deux méthodes :

```
override fun onSensorChanged(event: SensorEvent) {}  
override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}
```

- La méthode *onSensorChanged* sera appelée pour chaque nouvelle valeur pour chacun des capteurs auxquels on est inscrit

```
override fun onSensorChanged(event: SensorEvent) {  
    when(event.sensor.type) {  
        Sensor.TYPE_ACCELEROMETER -> {  
            event.values // do something here with float[3]  
        }  
        Sensor.TYPE_LIGHT -> {  
            event.values // do something here with float[1]  
        }  
    }  
}
```

Les capteurs – Exploitation des données brutes

- event.**values** est un tableau de *float* qui contient les données brutes correspondant à une annonce d'un capteur
- La taille du tableau va dépendre du capteur, par exemple :
 - Les capteurs de pression, de proximité, d'humidité, etc. vont fournir une seule valeur : *float[1]*
 - L'accéléromètre, le magnétomètre et le gyroscope en fourniront une pour chaque axe : *float[3]*
 - Certains capteurs peuvent en fournir plus, par exemple 15 pour le *Sensor.TYPE_POSE_6DOF*
- Il est donc nécessaire de consulter la documentation :
<https://developer.android.com/reference/android/hardware/SensorEvent.html>

(Pool d'instances)

- La machine virtuelle *Java* (JVM) est basée sur un *garbage collector* pour la gestion de la mémoire
 - Les instances orphelines d'objets seront automatiquement détruites par celui-ci
- Dans un environnement où la mémoire est limitée et lorsque beaucoup d'instances sont créées, la *JVM* lancera très souvent le *garbage collector* pour récupérer de la mémoire. Dans les cas extrêmes, on peut se retrouver avec le *garbage collector* qui utilise plus de ressources que l'application elle-même...



- Une technique existe pour limiter ce problème : la réutilisation des instances
 - C'est ce qui est mis en place par le *SensorManager* qui peut potentiellement devoir annoncer plusieurs centaines de nouvelles valeurs chaque seconde

(Pool d'instances)

- Plutôt que d'instancier un nouveau *SensorEvent* à chaque fois que la méthode *onSensorChanged* doit être appelée. Le *SensorManager* va gérer un pool d'instances de *SensorEvent*, qu'il modifiera pour accueillir les nouvelles données d'un capteur
- Cela signifie qu'il ne faut pas garder de référence vers les *SensorEvent* reçus, car le contenu de ceux-ci pourra être modifié par la suite. Il faut soit traiter les événements directement dans la méthode ou alors, éventuellement, garder une copie des données reçues

onSensorChanged

Added in API level 3

```
public abstract void onSensorChanged (SensorEvent event)
```

NOTE: The application doesn't own the `event` object passed as a parameter and therefore cannot hold on to it. The object may be part of an internal pool and may be reused by the framework.

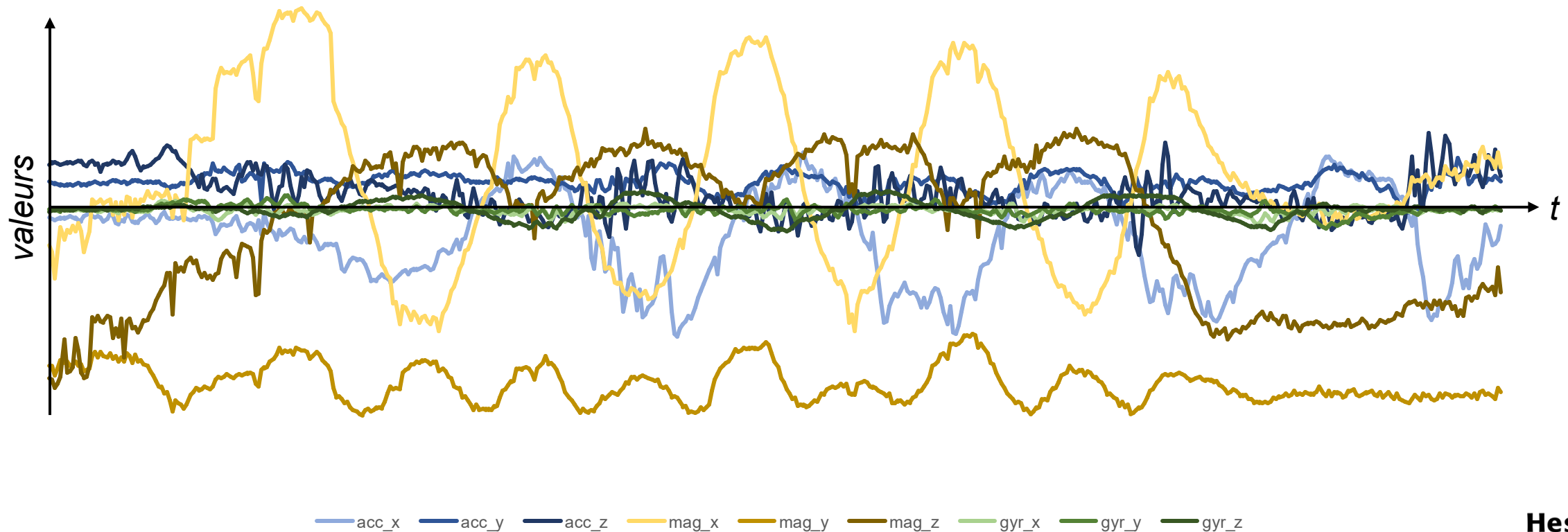
Les capteurs – Exploitation des données brutes

- Les capteurs de base fournissent des données brutes qui ne sont pas toujours directement exploitables
- Par exemple, la pression atmosphérique, seule, n'est pas très utile...
 - La variation de la pression atmosphérique dans le temps, à un point fixe, peut indiquer une tendance météo
 - Une variation rapide de la pression atmosphérique peut indiquer un changement d'altitude
 - Si on connaît la pression locale au niveau de la mer, on peut déduire notre altitude
- Sur *Android*, le *SensorManager* met à disposition des méthodes pour aider à l'interprétation des données, par exemple :
 - *SensorManager.getAltitude(float p0, float p)*
 $p0$: pression au niveau de la mer, p : pression mesurée, retourne l'altitude actuelle en [m]
Il est nécessaire de connaître la pression actuelle et locale au niveau de la mer



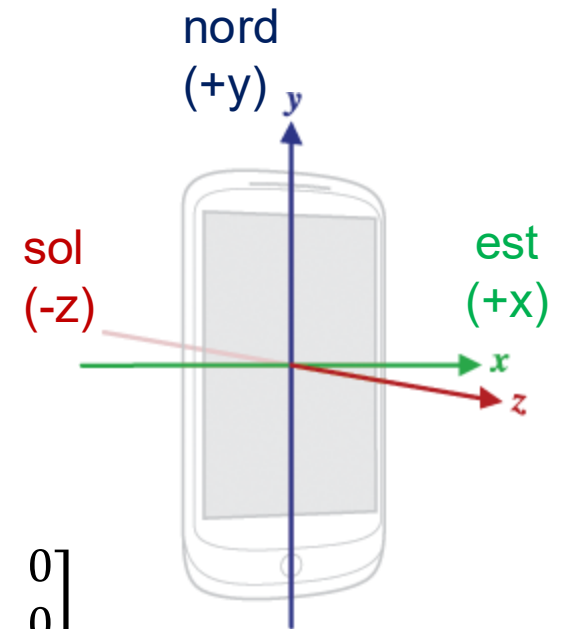
Les capteurs – Exploitation des données brutes

- De la même manière, les données brutes du gyroscope, de l'accéléromètre ou du magnétomètre sont difficilement utilisables telles quelles...



Les capteurs – Exploitation des données brutes

- Il existe également des méthodes pour nous aider à exploiter ces données
- Le smartphone a une position de référence :
 - Le dos du smartphone (axe z négatif) aligné avec le centre de gravité de la Terre
 - Le haut du smartphone (axe y positif) aligné avec le nord magnétique
- L'utilisation du magnétomètre et de l'accéléromètre peut nous permettre de déterminer l'orientation réelle du smartphone par rapport à sa position de référence :



```
SensorManager.getRotationMatrix( float[] R,
                                null,
                                float[] gravity,
                                float[] geomagnetic)
```

$$R = \begin{bmatrix} M_0 & M_1 & M_2 & 0 \\ M_4 & M_5 & M_6 & 0 \\ M_8 & M_9 & M_{10} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(Matrices de transformation 3D)

- En visualisation 3D, les transformations géométriques sont représentées sous la forme de matrices, quelques exemples :
 - L'identité (aucune transformation) :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \cdot x + 0 \cdot y + 0 \cdot z + 0 \cdot 1 \\ 0 \cdot x + 1 \cdot y + 0 \cdot z + 0 \cdot 1 \\ 0 \cdot x + 0 \cdot y + 1 \cdot z + 0 \cdot 1 \\ 0 \cdot x + 0 \cdot y + 0 \cdot z + 1 \cdot 1 \end{pmatrix} = \begin{pmatrix} 1 \cdot x \\ 1 \cdot y \\ 1 \cdot z \\ 1 \cdot 1 \end{pmatrix}$$

Transformation
Matrice identité 4x4

On applique la
transformation

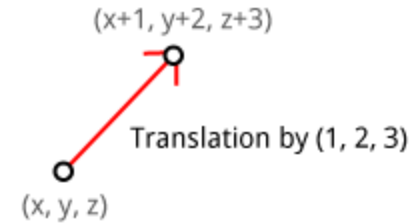
Coordonnées x, y, z
avant la transformation

Coordonnées après la
transformation

(Matrices de transformation 3D)

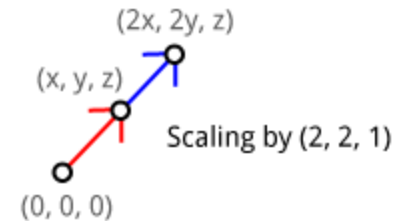
- Translation :

$$\begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + X \cdot 1 \\ y + Y \cdot 1 \\ z + Z \cdot 1 \\ 1 \end{pmatrix}$$



- Scaling :

$$\begin{bmatrix} SX & 0 & 0 & 0 \\ 0 & SY & 0 & 0 \\ 0 & 0 & SZ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} SX \cdot x \\ SY \cdot y \\ SZ \cdot z \\ 1 \end{pmatrix}$$



(Matrices de transformation 3D)

- Rotations autour de l'origine :

Rotation around X-axis:

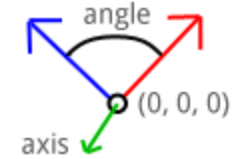
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ \cos \theta \cdot y - \sin \theta \cdot z \\ \sin \theta \cdot y + \cos \theta \cdot z \\ 1 \end{pmatrix}$$

Rotation around Y-axis:

$$\begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta \cdot x + \sin \theta \cdot z \\ y \\ -\sin \theta \cdot x + \cos \theta \cdot z \\ 1 \end{pmatrix}$$

Rotation around Z-axis:

$$\begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta \cdot x - \sin \theta \cdot y \\ \sin \theta \cdot x + \cos \theta \cdot y \\ z \\ 1 \end{pmatrix}$$



(Matrices de transformation 3D)

- Composition des transformations :
 - En multipliant deux matrices (4x4) de transformation on obtient une nouvelle matrice (4x4) représentant la composition des deux transformations

$$M_{\text{translate}} \cdot M_{\text{scale}} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 2 & 0 & 2 \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

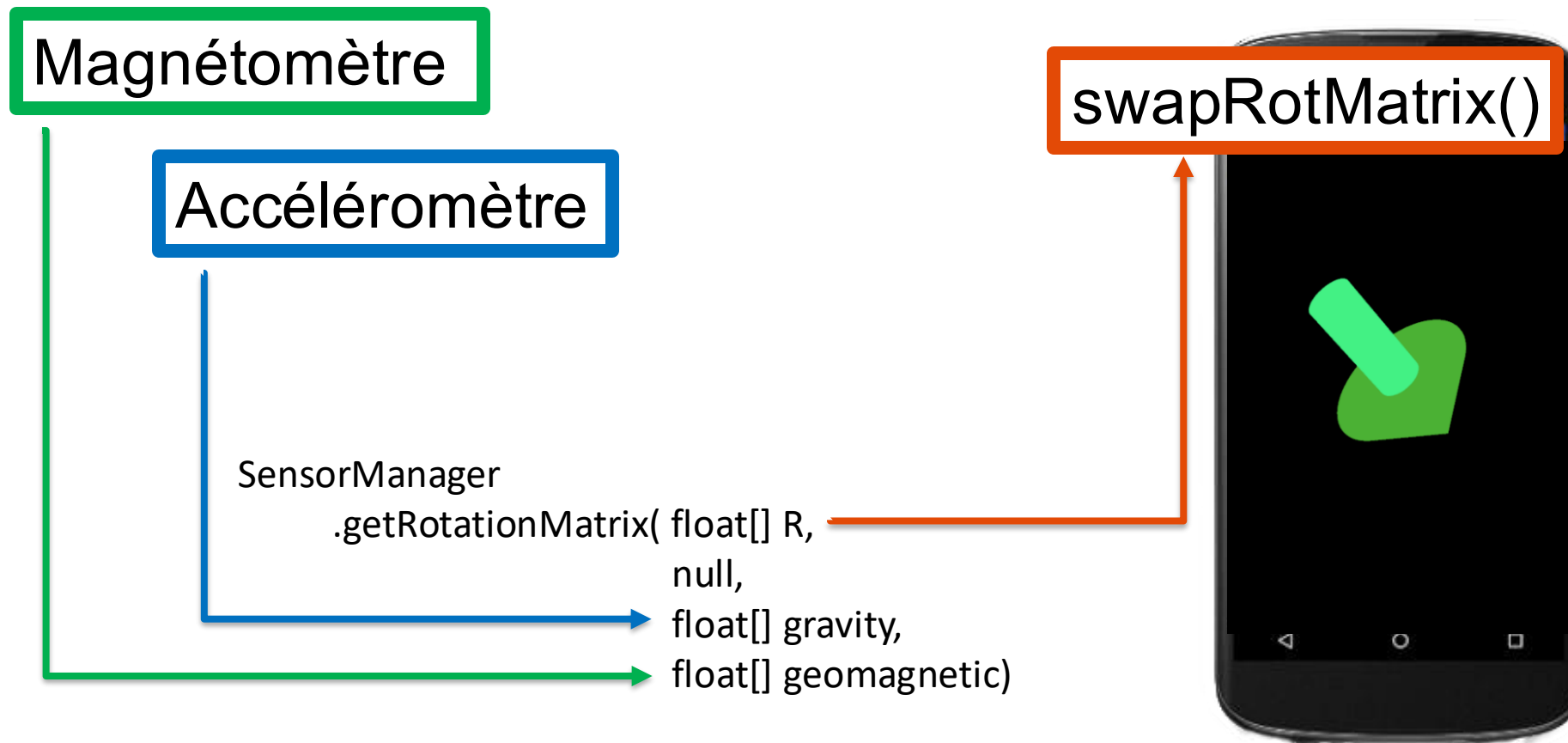
Transformation Translation par (1,2,3)
Transformation Scaling x2
La nouvelle transformation

$$\begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 2 & 0 & 2 \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} 2x + 1 \\ 2y + 2 \\ 2z + 3 \\ 1 \end{pmatrix}$$

On obtient bien des coordonnées «translatées» de (1,2,3) et «scalées» d'un facteur 2

Exercice 6 – Exploitation des données

- Réalisation d'une boussole 3D pointant vers le nord
- On utilise le magnétomètre pour obtenir la transformation permettant d'orienter le modèle 3D correctement (composition des rotations autour des 3 axes)



Les capteurs composites

- Pour faciliter l'exploitation des capteurs matériels, *Android* met à disposition des capteurs virtuels qui vont traiter et/ou fusionner les données provenant d'un ou plusieurs capteurs physiques
- Quelques exemples de capteurs composites, il en existe actuellement 15 :
 - *Step counter* et *step detector* (podomètre)
Habituellement basés sur l'accéléromètre, mais peuvent aussi utiliser d'autres capteurs si la précision et la consommation énergétique sont correctes
 - *Game rotation vector*
Accéléromètre, magnétomètre et (si présent) le gyroscope
 - *Linear Acceleration*
Accéléromètre et gyroscope (si présent) ou magnétomètre
 - Etc.

Les capteurs – Analyse avancée

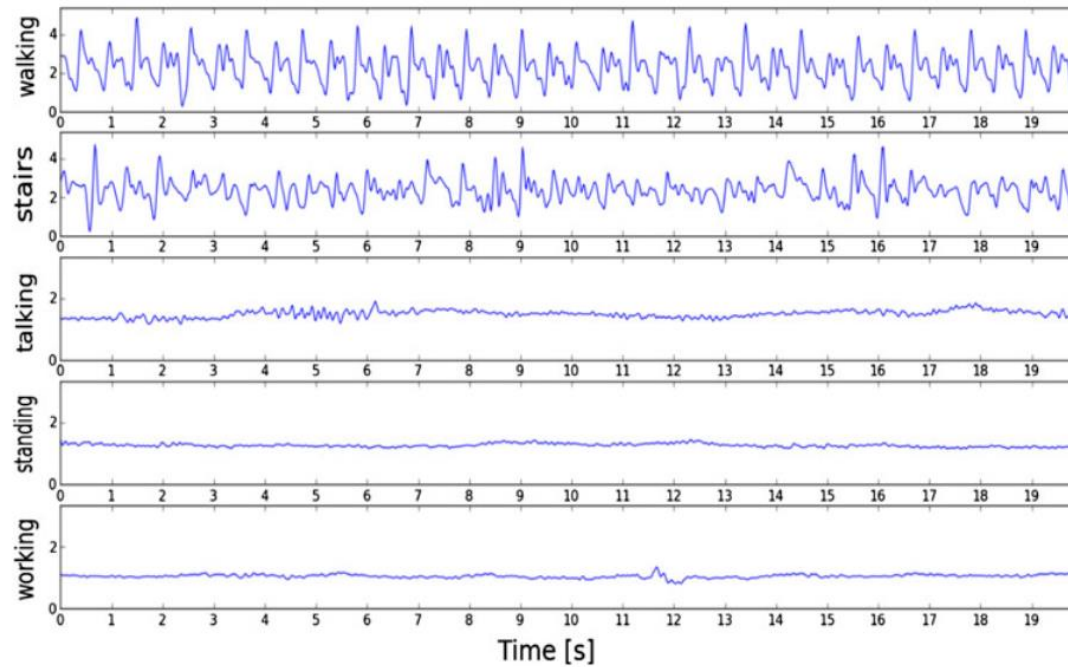
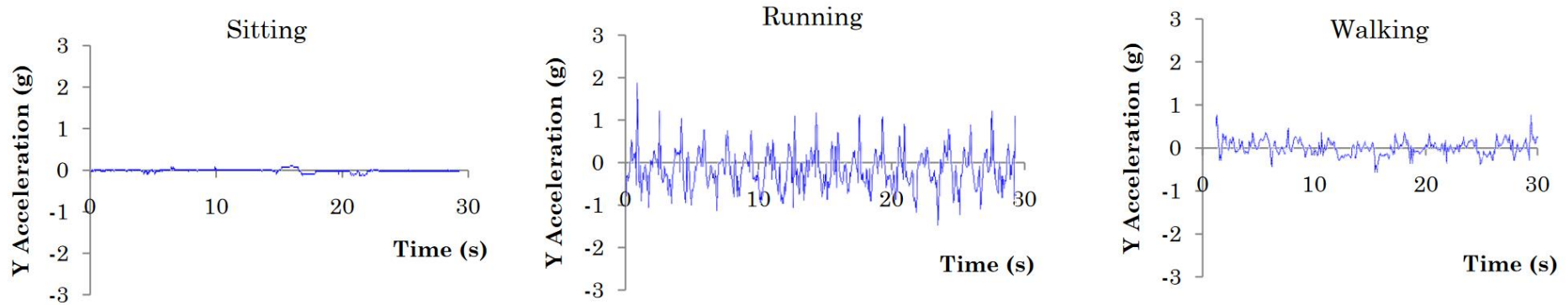
- Les capteurs peuvent être utilisés pour catégoriser les conditions d'utilisation
 - Par exemple, reconnaissance de l'activité humaine

Group	Activities
Ambulation	Walking, running, sitting, standing still, lying, climbing stairs, descending stairs, riding escalator, and riding elevator.
Transportation	Riding a bus, cycling, and driving.
Phone usage	Text messaging, making a call.
Daily activities	Eating, drinking, working at the PC, watching TV, reading, brushing teeth, stretching, scrubbing, and vacuuming
Exercise/fitness	Rowing, lifting weights, spinning, Nordic walking, and doing push ups.
Military	Crawling, kneeling, situation assessment, and opening a door.
Upper body	Chewing, speaking, swallowing, sighing, and moving the head.

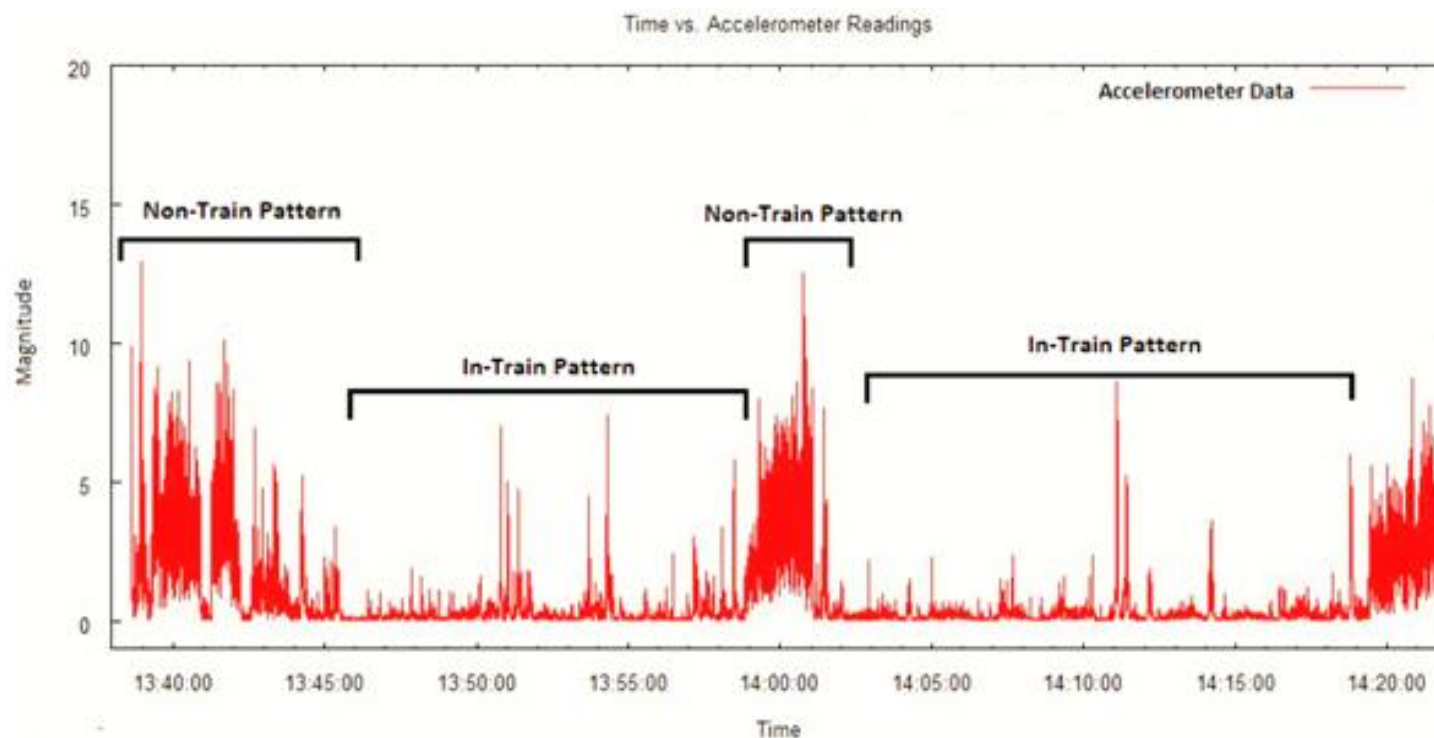
Les capteurs – Analyse avancée

- Quatre groupes d'attributs peuvent être utilisés pour reconnaître une activité humaine :
 - *Environnemental*
température, bruit, etc.
 - *Mouvement*
accéléromètre, gyroscope, magnétomètre
 - *Localisation*
GPS, etc.
 - *Physiologique*
température corporelle, rythme cardiaque, etc.

Les capteurs – Analyse avancée

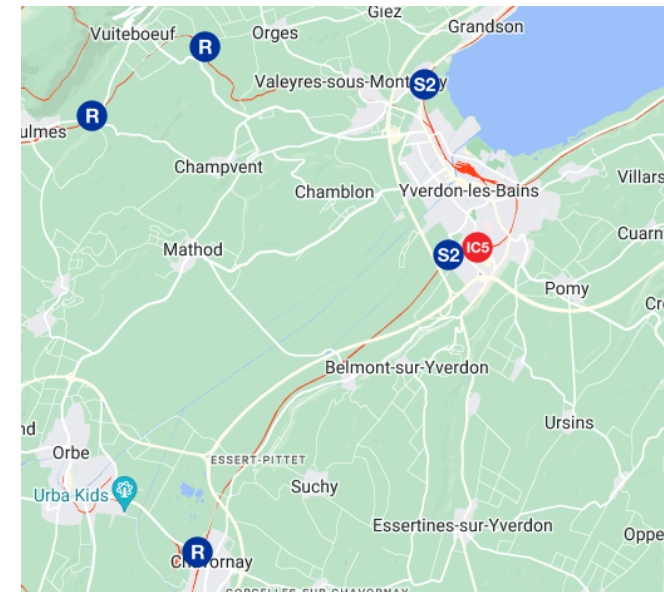
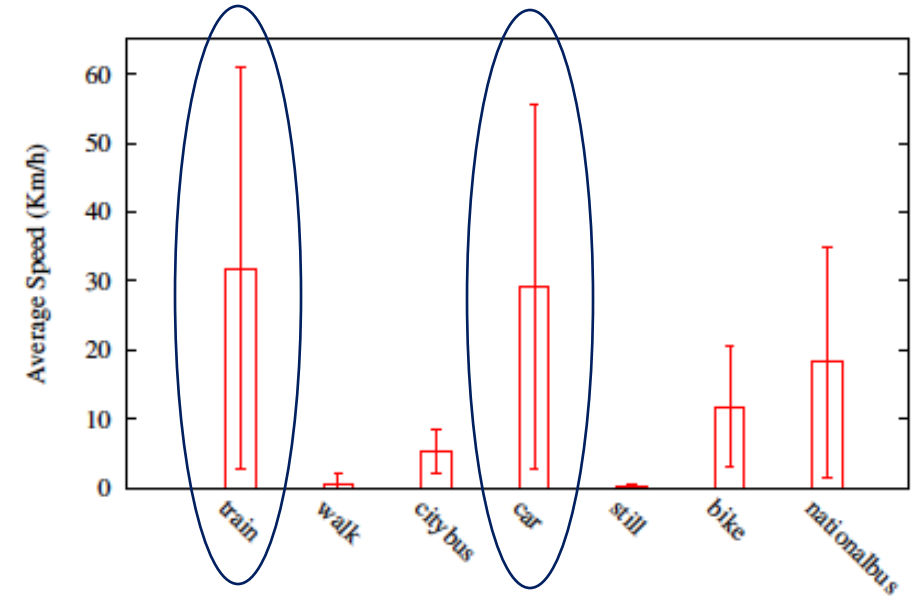


Les capteurs – Analyse avancée



Les capteurs – Analyse avancée

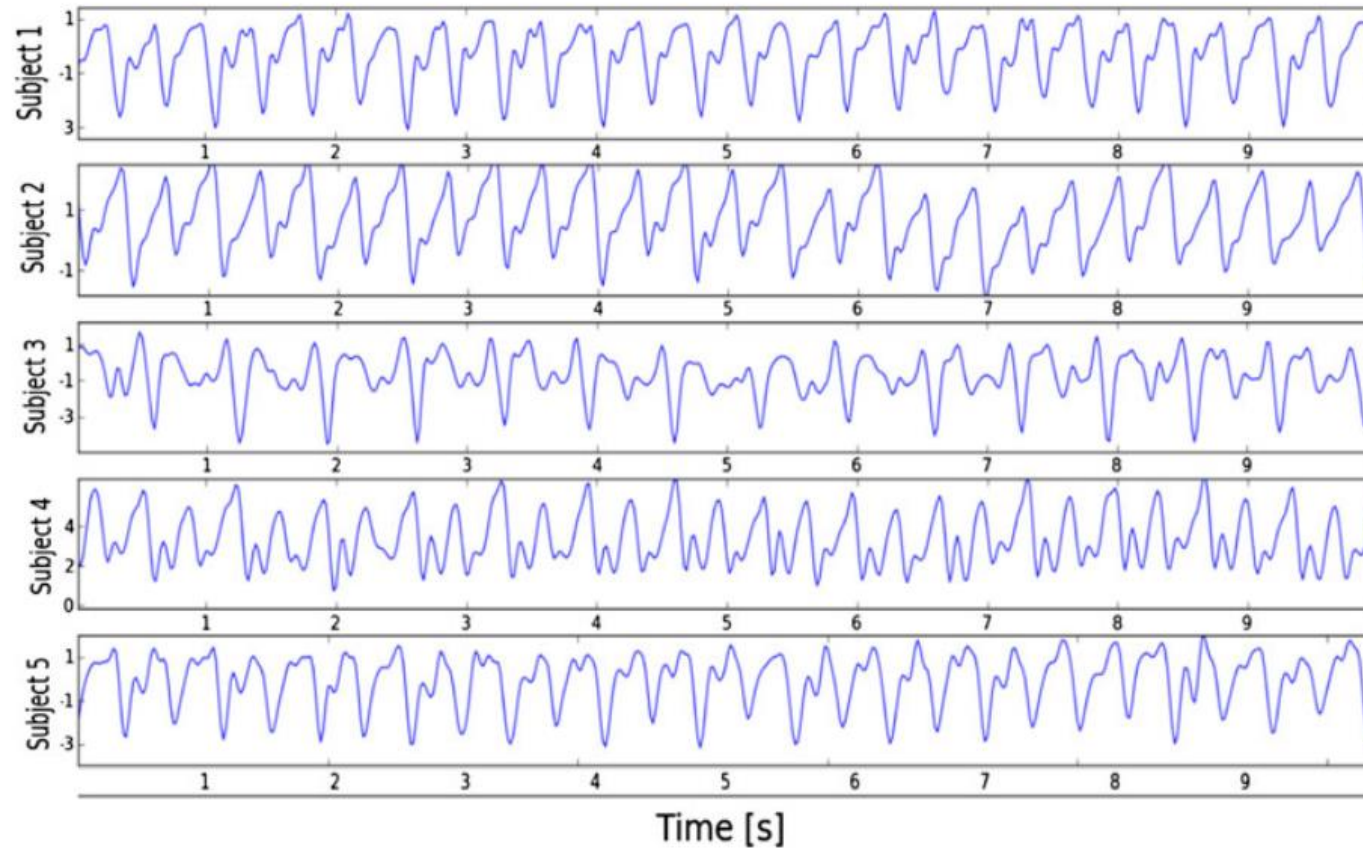
- Un seul capteur ne permettra généralement pas de déterminer avec précision l'activité courante
 - Par exemple, avec uniquement l'information de vitesse moyenne, il n'est pas possible de différencier certains moyens de transport
- Avec plusieurs capteurs et plusieurs sources d'information il devient possible d'obtenir une bonne précision :
 - *ActivityRecognition* dans les *Play Services*



<http://maps.vasile.ch/transit-sbb/>

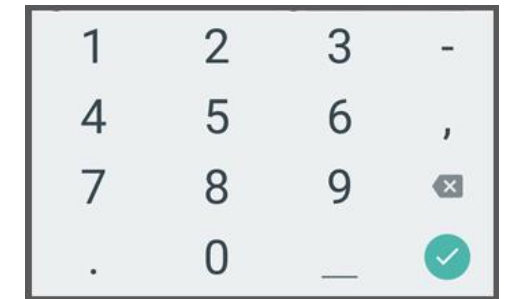
Les capteurs – Analyse avancée

- Certaines études ont montré qu'il était possible également d'identifier la personne :



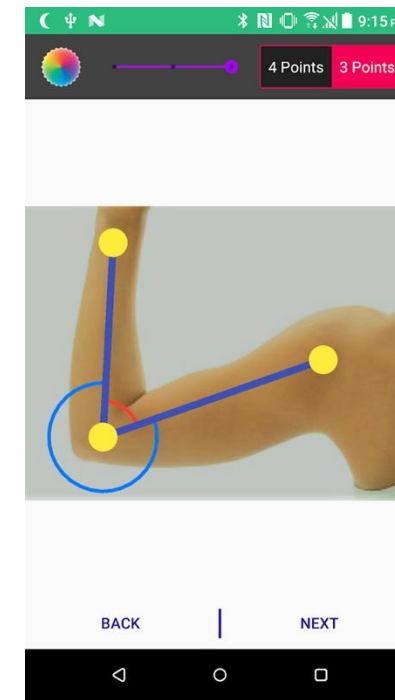
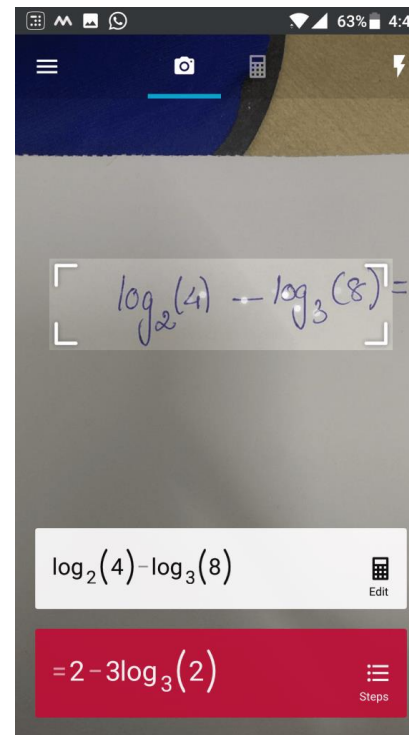
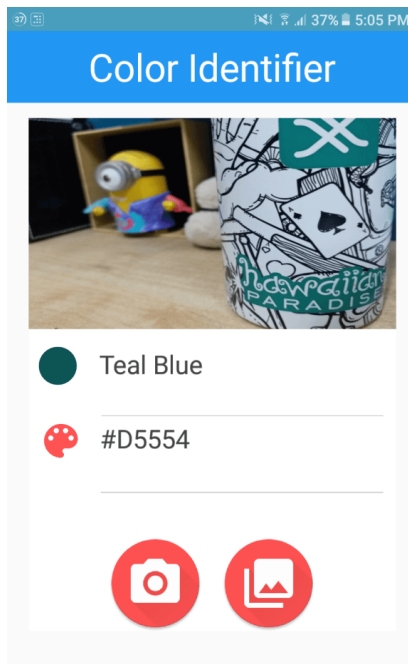
Les capteurs – Analyse avancée

- Traditionnellement l'accès aux capteurs du smartphone ne demandait pas de permission particulière :
 - *Android 10* (2019), ajoute la permission *ACTIVITY_RECOGNITION* pour pouvoir accéder à certains capteurs tels que le podomètre et certaines librairies de reconnaissance de l'activité
- L'accès aux données brutes est toujours libre (< 200 Hz) :
 - Etude *TouchLogger* qui a réalisé un démonstrateur de keylogger basé sur les petites rotations du smartphone lors de l'appui sur les touches du clavier virtuel (> 70% de classification correcte des chiffres)
 - Une autre étude a montré la faisabilité d'utiliser l'accéléromètre pour mesure et interpréter les vibrations du smartphone provoquées par son microphone, permettant ainsi « d'écouter » une conversation téléphonique...
- TouchLogger: Inferring Keystrokes On Touch Screen From Smartphone Motion
Liang Cai, Hao Chen, Univ. of California
- Towards Device Independent Eavesdropping on Telephone Conversations with Built-in Accelerometer
Weigao Si, Daibo Liu, Taiyuan Zhang, Hongbo Jiang, Hunan University,



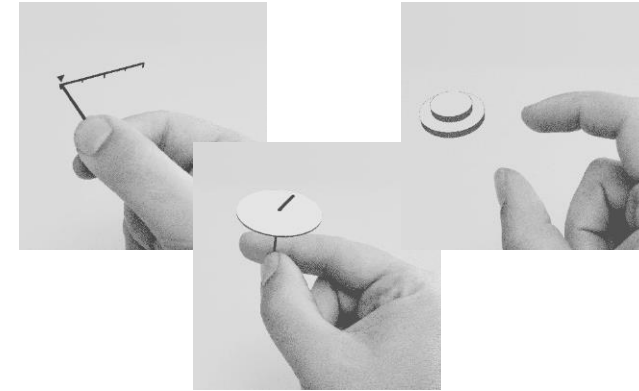
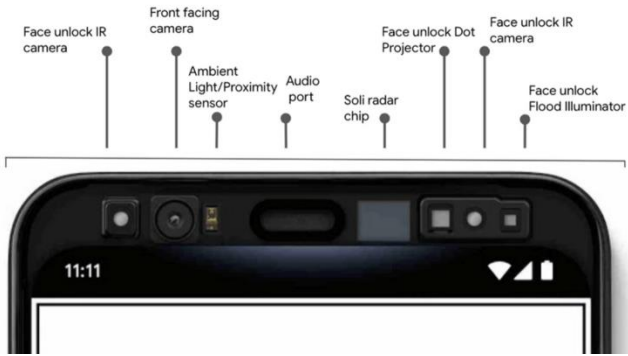
Les capteurs – Usage détournés

- Les smartphones possèdent aujourd'hui au minimum 1 capteur optique, leur but principal est de prendre des photos, mais pas uniquement :

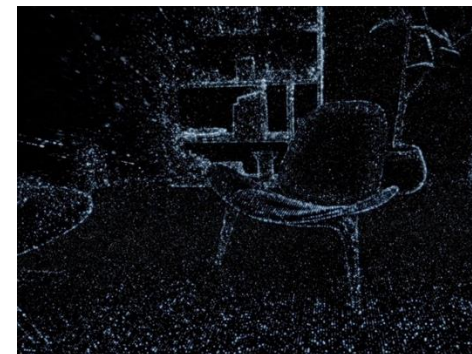


Les capteurs – Le futur ?

- *Google* a tenté d'ajouter un capteur radar sur les *Pixel 4* (2019)



- Depuis 2020, *Apple* a ajouté un capteur *LiDAR* aux iPhones



Les *Wearables*



Les *Wearables* – Définitions

- En français la distinction n'est pas évidente...
 - *Wearable* se traduit comme *portable*
 - *Handheld* se traduit également comme *portable*
- Il ne faut pas confondre ces deux termes :



Wearables → Portable ← Handheld



Les *Wearables* – Définitions

- Qu'est-ce que la technologie *wearable* ?
 - Une technologie portable, ou technologie «mettable», est un vêtement ou un accessoire comportant des éléments informatiques et électroniques avancés (≈ informatique vestimentaire)
 - Sous-catégorie de l'*Internet des Objets*
 - Conçu pour permettre à la personne le portant d'interagir sans devoir appuyer sur des boutons ou effectuer des manipulations trop complexes
 - Apporte sa fonctionnalité en tout temps, même quand l'utilisateur est occupé par de multiples autres activités
 - Fonctionne en permanence, toujours disponible

Les *Wearables* – Définitions

- Les *wearables* peuvent être classés en 3 catégories :
 - *Tracking, Measuring, Sensing*
Capteurs corporels, très souvent orientés fitness. Interface utilisateur limitée, faible consommation, les données collectées sont transférées pour être analysées et restituées
 - *Réalité améliorée*
Aide à la vie courante, l'information dont vous pourriez avoir besoin est mise automatiquement en avant (navigation, rappels, recommandations)
 - *Second écran (Scaled down electronic gadgets)*
Extension du smartphone permettant de réaliser des tâches simples sans devoir sortir le téléphone de sa poche

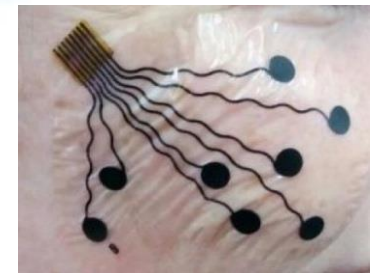
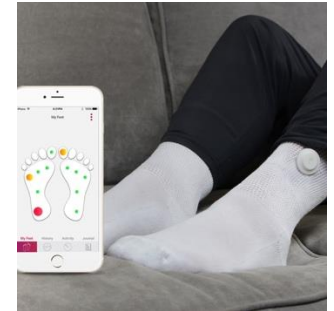
Les *Wearables* – Quelques exemples

- Capteurs fitness et bien être :
 - *Oura Ring* Destinée à surveiller la qualité du sommeil à l'aide d'un cardio optique, d'un accéléromètre, d'un gyroscope et une mesure de la température corporelle
 - *Garmin HRM-Pro* Ceinture de poitrine intégrant un capteur cardiaque
 - *FINIS Smart Goggle* lunettes de natation connectées intégrant un écran permettant le suivi des entraînements, et la mesure de certains paramètres
 - *WHOOP* Bracelet connecté intégrant plusieurs capteurs tels que les battements cardiaques, température corporelle, gps, etc.

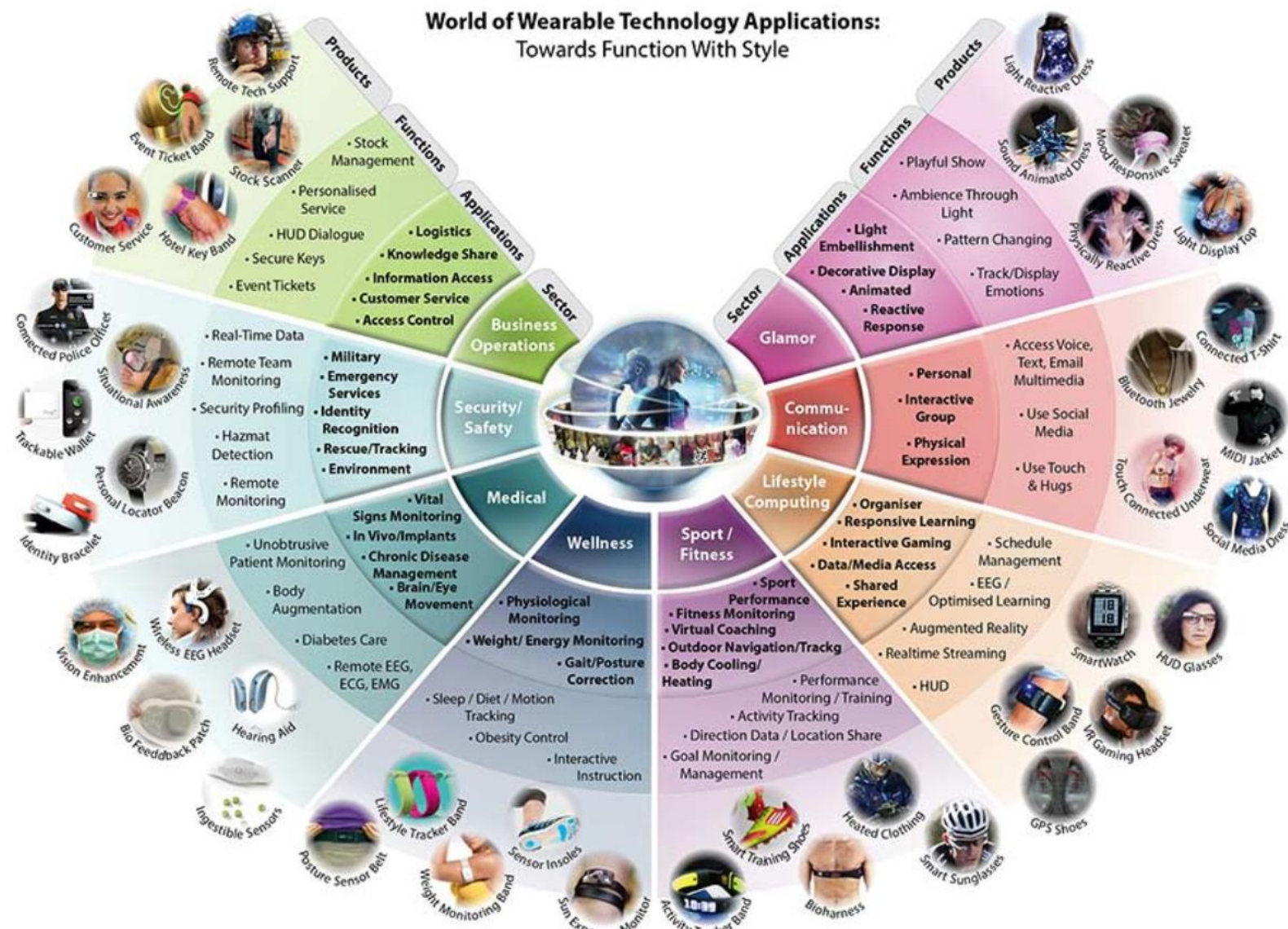


Les *Wearables* – Quelques exemples

- Capteurs médicaux :
 - Monitoring du taux de glucose en continu
 - Détection des ulcères du pied
Permet de prévenir et d'éviter des infections chez les personnes diabétiques
 - Mesure des quantités d'eau ingérées, reconnaissance du mouvement: eau bue, eau renversée ou jetée
Détection des cycles de lavage
 - Temp tech tattoos
Par exemple, mesure de l'activité musculaire de patients atteints de maladies neurodégénératives

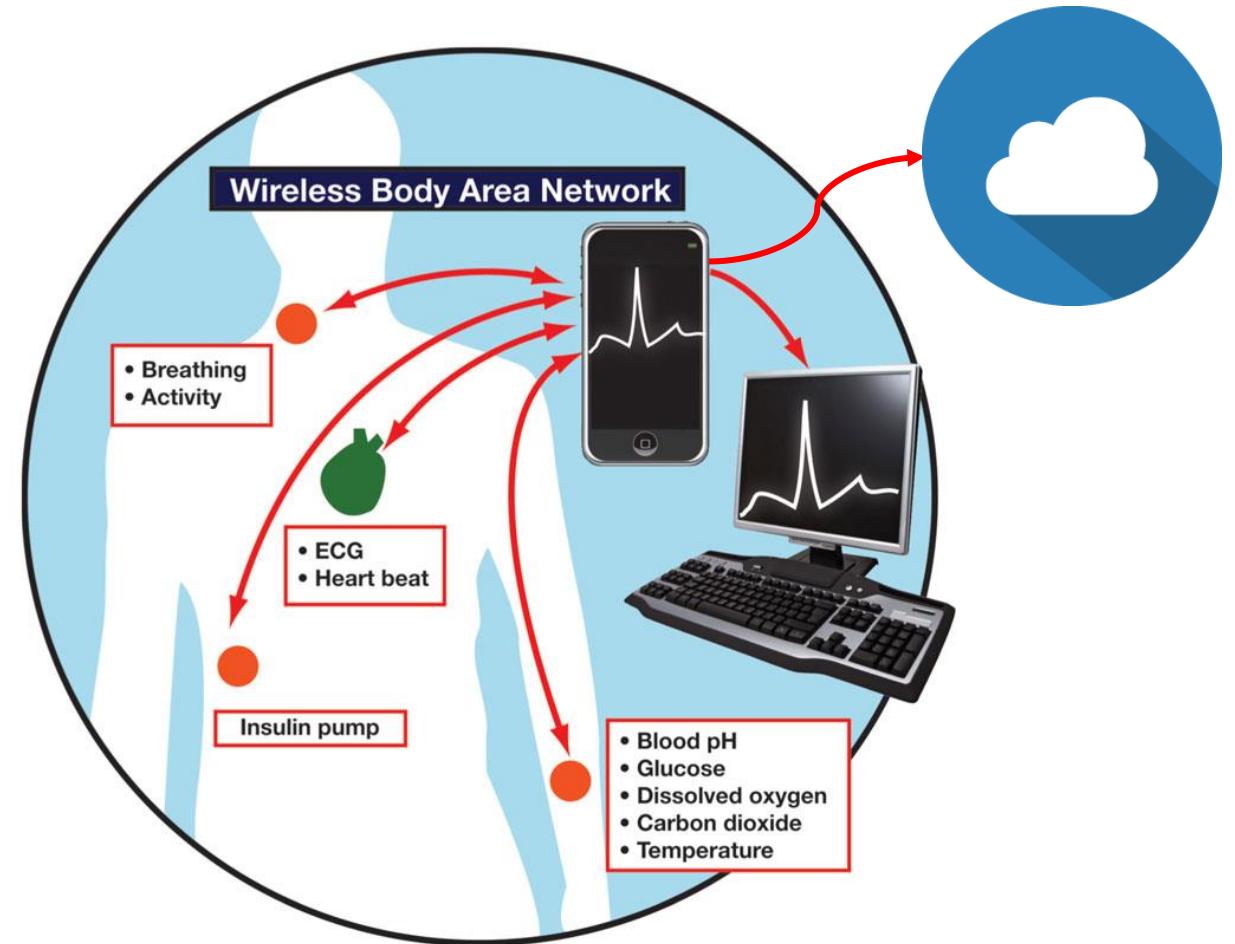


De nombreuses applications pour les *Wearables*



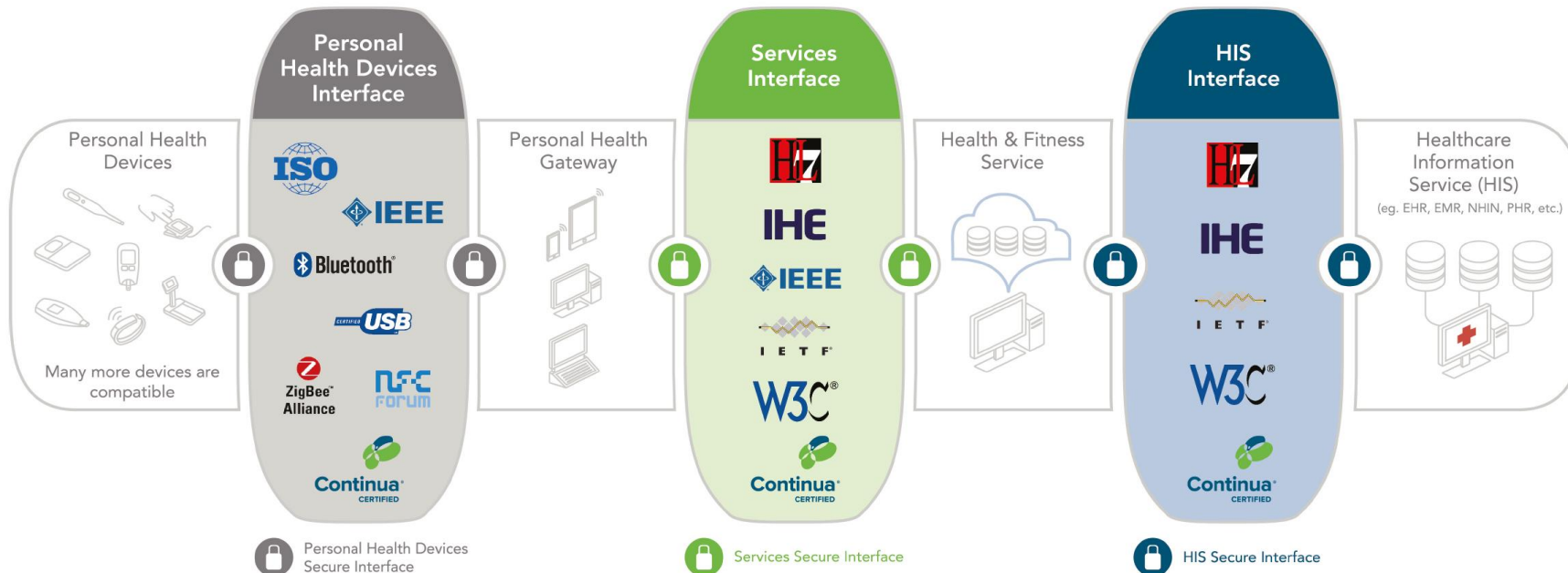
Les *Wearables* – Body Area Network

- Ces systèmes font partie intégrante de l'environnement de l'utilisateur
- Ils sont souvent reliés au travers d'un réseau propre, le *BAN* (*Body Area Network*) avec au centre un smartphone
- La communication se fait le plus souvent avec du *Bluetooth Low Energy*



Les *Wearables* – Interopérabilité

- Le point d'attention avec les *wearables* est leur manque d'interopérabilité
 - Généralement chaque *wearable* est accompagné de sa propre application mobile et de son propre service cloud avec des protocoles propriétaires
- Dans le domaine de la santé connectée, il existe toutefois la norme *Continua* censée promouvoir l'interopérabilité (peu d'appareils compatibles à ce jour)



Dispositifs médicaux – Réglementation

- La mise sur le marché de *wearables* et/ou d'applications mobiles destinés à la santé humaine nécessite de passer de lourdes procédures de contrôle de conformité
- Depuis 2021, la nouvelle législation suisse *ODim* précise et renforce les procédures nécessaires à la mise sur le marché d'une application médicale
 - Est-ce que mon application est un dispositif médical ?
Ils donnent l'exemple d'une application mobile accompagnant une aide auditive et permettant de régler son volume sonore, qui est à considérer comme une app médicale
 - A présent toutes les app médicales sont au minimum de la classe *IIa* qui implique de devoir confier l'évaluation de la conformité de celles-ci, lors de leur lancement et à chaque mise à jour, à une organisation tierce...
- Cela explique pourquoi la plupart des technologies mises sur le marché le sont dans les catégories fitness et bien-être

Google Wear OS

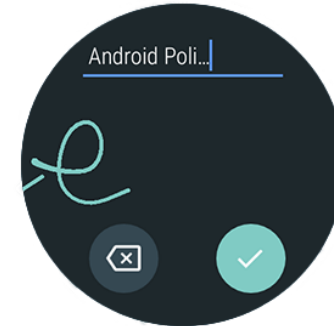


Wearables – Les montres connectées

- Les montres connectées sont actuellement la catégorie la plus développée des *wearables*, il en existe plusieurs familles, les principales étant :
 - *Apple Watch*
 - *Android Wear*
- Certains constructeurs de montres ont développé leur propre écosystème basé sur un OS propre, par exemple :
 - *Samsung Tizen* (passage sur *Wear OS* depuis fin 2021)
 - *Garmin*
 - *Fitbit*
- Pour pouvoir bénéficier du plein potentiel d'une montre connectée, celle-ci devra être appairée avec un smartphone. A l'exception des *Apple Watches*, toutes peuvent être utilisées avec un smartphone *Android* ou *iOS*

Wearables – Les montres connectées

- Initialement vues comme des smartphones miniatures portés au poignet, il y a eu beaucoup d'expérimentations durant leurs premières années d'existence pour développer des applications complexes et complètes, ainsi que des jeux :
 - Difficulté à saisir des données...



- Durée de vie de la batterie, interface utilisateur très réduite, inconfort d'utilisation ...
- Aujourd'hui, le marché semble avoir convergé sur des applications mettant en avant uniquement les informations importantes, ainsi que les fonctionnalités phares (1 ou 2) réalisables en « 2-3 taps », avec des interfaces épurées

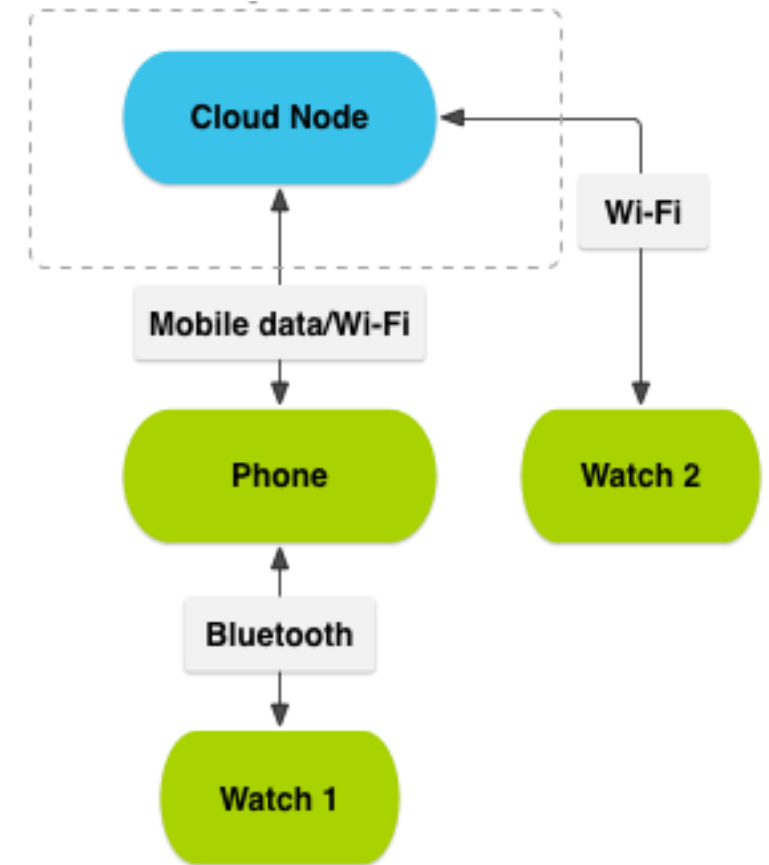
Wear OS

- Lancé en mars 2014 sous le nom d'*Android Wear*
- Version d'*Android* dédiée aux objets *wearables*, en particulier aux montres connectées
- Plusieurs évolutions :
 - *Android Wear 1.0* (2014)
 - Présence obligatoire du smartphone (pas de *Wi-Fi* ou de *LTE*)
 - Pas d'applications autonomes
 - *Android Wear 2.0* (2017)
 - Renommé ultérieurement en *Wear OS*
 - Arrivée des connexions *Wi-Fi* et *LTE*
 - Les applications autonomes sont encouragées
 - *Wear OS 3.0* (2022 – «fusion» avec *Tizen*)
 - Arrivée des tuiles applicatives tierces
 - Possibilités offertes aux constructeurs de personnaliser l'OS et l'application compagnon
 - *Wear OS 4.0* (2023), *Wear OS 5.0* (2024), *Wear OS 5.1* (2025) - optimisations



Wear OS – Communication

- Une montre *Wear OS* est appairée par *BLE* avec un smartphone
- *Android Wear 1.0* impliquait que la montre doive systématiquement passer par la connexion du smartphone pour accéder à *Internet* (proxy via *BLE*)
- Depuis *Android Wear 2.0* les montres peuvent directement accéder à *Internet* via *Wi-Fi* ou *LTE*
 - Toutefois la montre privilégiera toujours de passer par le smartphone s'il est à proximité (économie d'énergie)
 - La connexion *Wi-Fi* / *LTE* sera utilisée : en l'absence du smartphone, lorsque la montre est en charge, ou si une application demande explicitement un accès *Internet* à haut débit (par exemple, pour le téléchargement d'une ressource volumineuse). Il est conseillé de décaler et de planifier de tels téléchargements lorsque la montre sera en charge

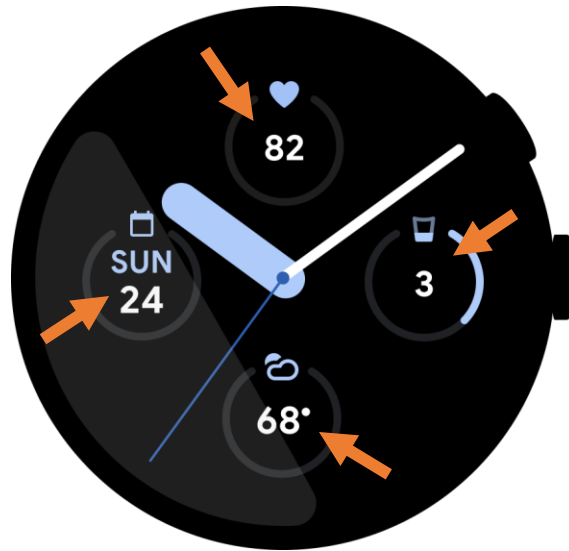


Wear OS – Développement

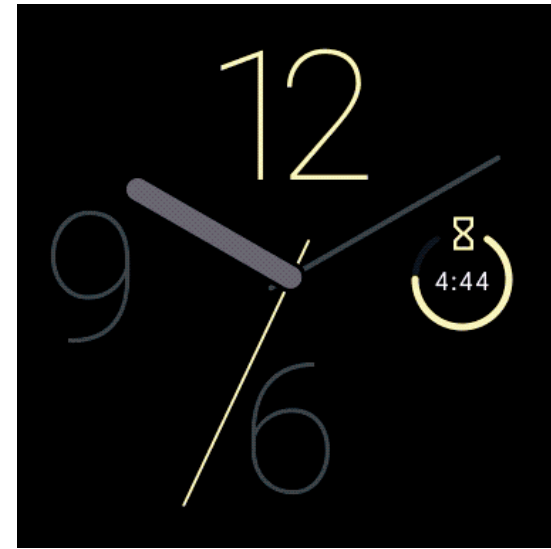
- Il existe plusieurs approches pour développer sur une montre *Android* :



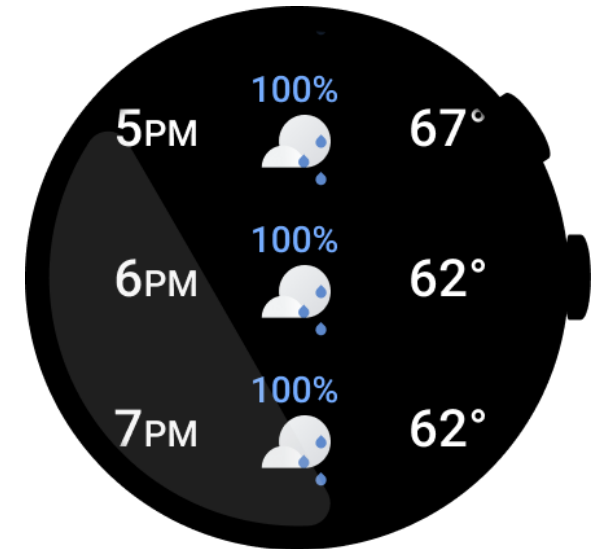
Notifications



Complications



Tuiles



Application

Wear OS – Développement

- Sur une montre on va se concentrer sur les fonctionnalités principales, on ne va pas chercher à réimplémenter toutes les fonctions de l'application smartphone
- Les fonctionnalités seront à prioriser en fonction de leur importance et de la fréquence à laquelle les utilisateurs en auront besoin. Cela permettra de décider sur quelle(s) surface(s) les mettre en œuvre, par exemple pour une application météo :



Complication

- Météo actuelle



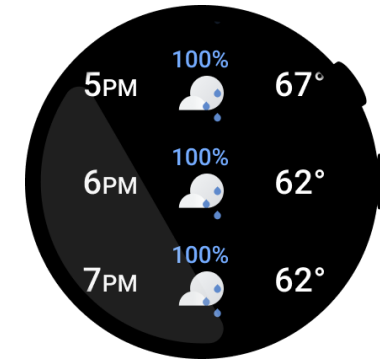
Notification

- Alertes météo



Tuile

- Météo actuelle
- Prévisions du jour

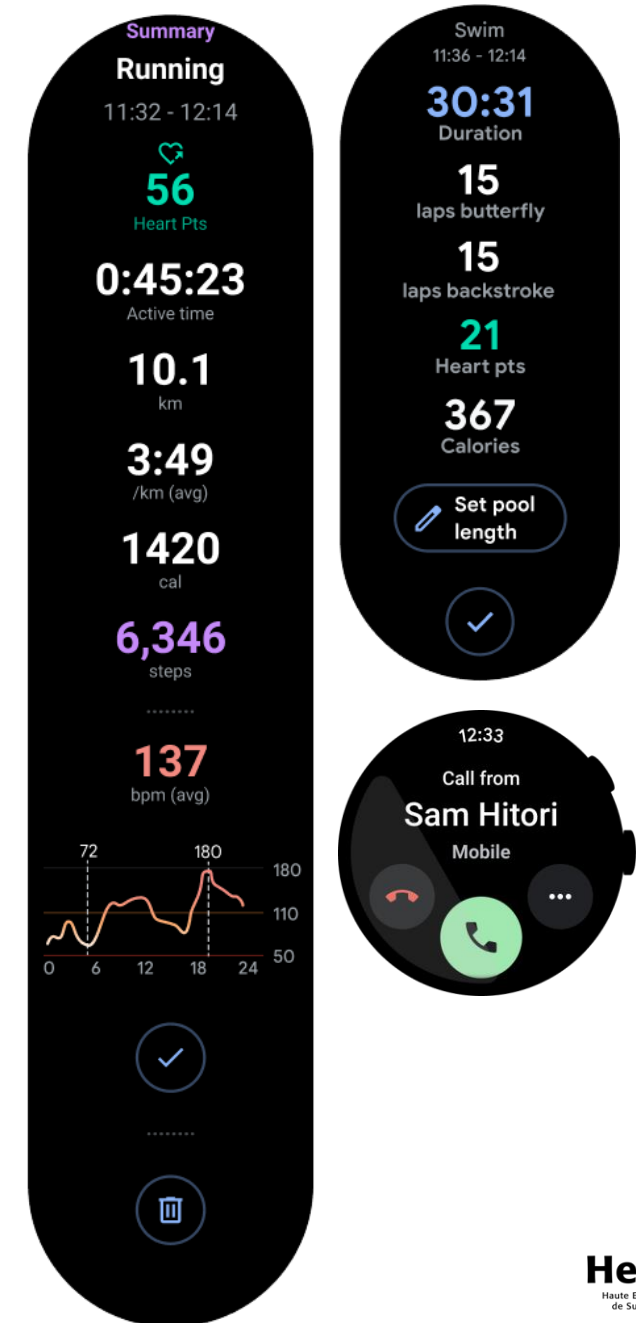


Application

- Météo actuelle
- Prévisions et évolution du jour
- Prévisions de la semaine
- Paramètres

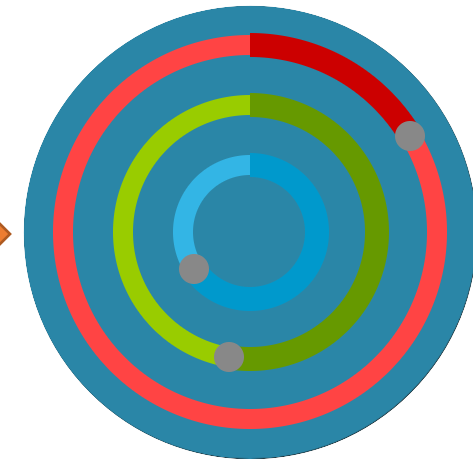
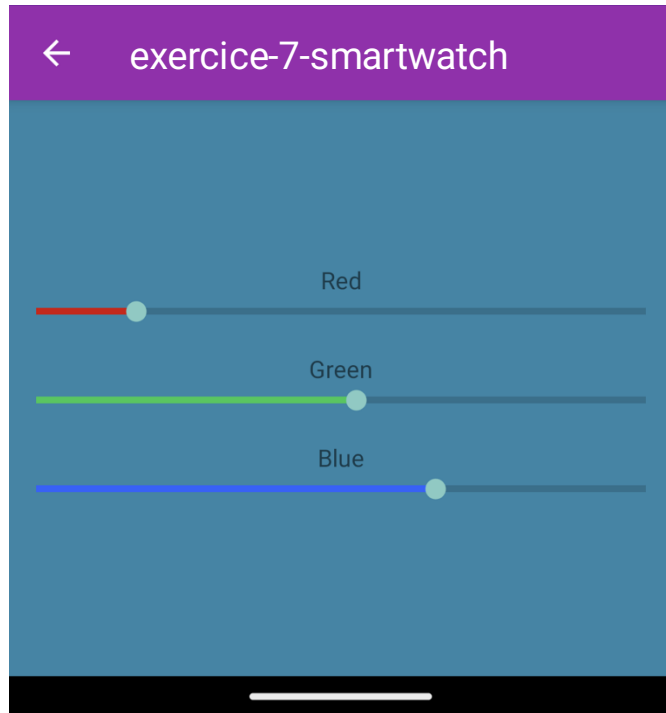
Wear OS – Développement

- Le développement d'une application pour *Wear OS* est très similaire au développement d'une application smartphone *Android*
 - *Activités* (les *Fragments* ne sont pas recommandés)
 - *ViewModels*
 - Bibliothèques *Jetpack* (dont *Compose* qui est recommandé)
 - Etc.
- Un effort tout particulier devra être consacré à la réalisation de l'interface graphique pour assurer une bonne expérience utilisateur
 - Le défilement vertical est à privilégier



Wear OS – Application compagnon

- Une application montre vient compléter, la plupart du temps, sa version smartphone, celles-ci devront alors partager et synchroniser des données
 - Mise à disposition de la *Data Layer API* :



Wear OS – Wearable Data Layer API

- Fait partie des *Play Services*
 - *Annonce des fonctionnalités*
 - Les applications installées sur la montre ou le smartphone peuvent indiquer qu'elles prennent en charge certaines fonctionnalités
 - *Envoi de messages* (Message Client)
 - Pour des appels RPC avec un payload limité
 - *Transférer des données* (Channel Client)
 - Par exemple pour du streaming audio ou vidéo
 - *Synchroniser les données* (Data Client)
 - Espace de stockage privé d'une application
 - Chaque nœud (smartphone, montre ou cloud) peut lire ou écrire des données
 - Les autres nœuds sont notifiés des changements
 - Synchronisation automatique entre les différents nœuds, évtl. après une reconnexion

Wear OS – Wearable Data Layer API

	Data Client	Message Client	Channel Client
Data size greater than 100 kb	Yes	No	Yes
Requires a network connection, such as saving data to the cloud	Yes Evtl. via Bluetooth	Uses Bluetooth	Uses Bluetooth
Can send messages to nodes that aren't currently connected	Yes	No	No
Can send from one device to another device, such as from wear to mobile	No, from the cloud to all the nodes	Yes	Yes, for both one-way requests and bidirectional requests
Supports fanning out data to all the devices	Yes	No	No
Reliability	Yes	Yes, when Bluetooth connection is established	Yes