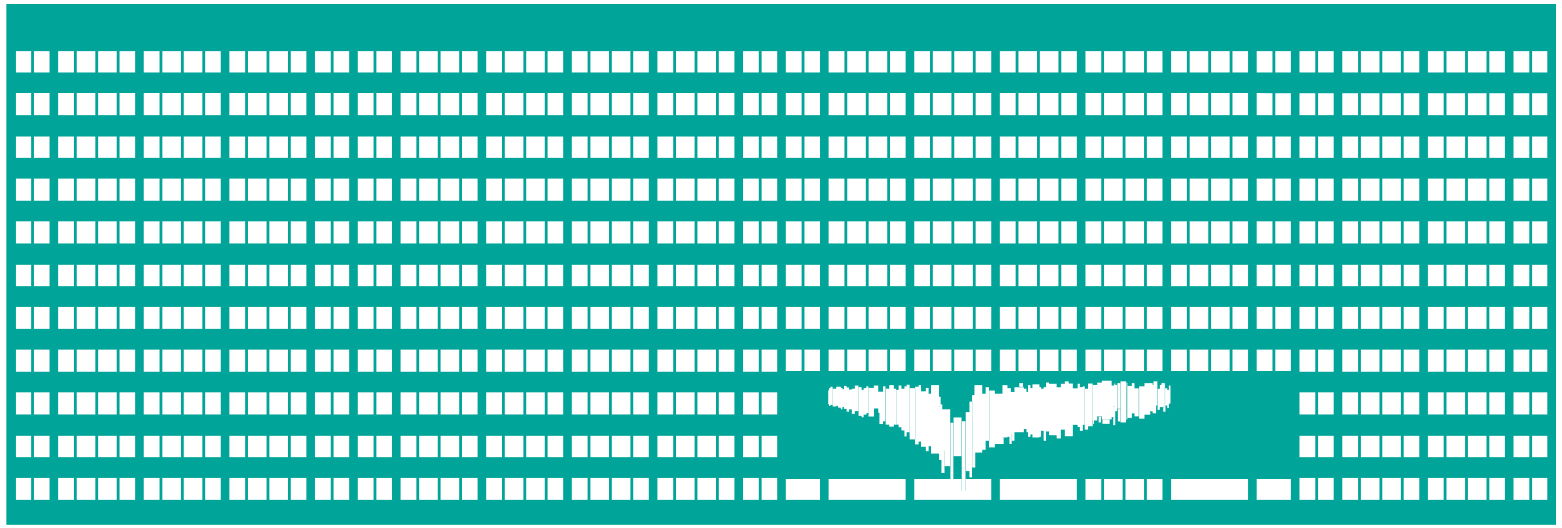
VŠB TECHNICKÁ UNIVERZITA OSTRAVA

VSB TECHNICAL UNIVERSITY OF OSTRAVA

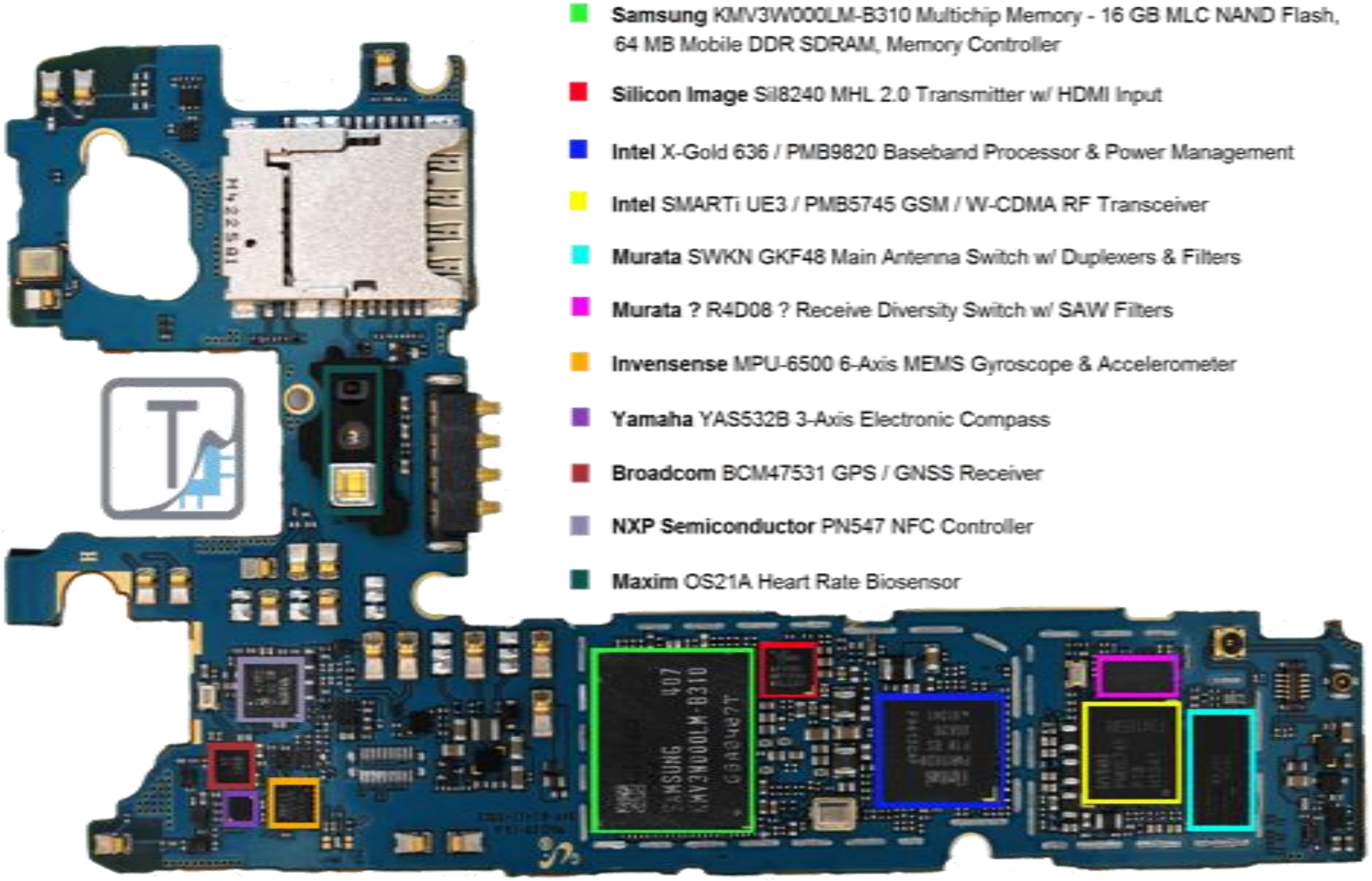


www.vsb.cz

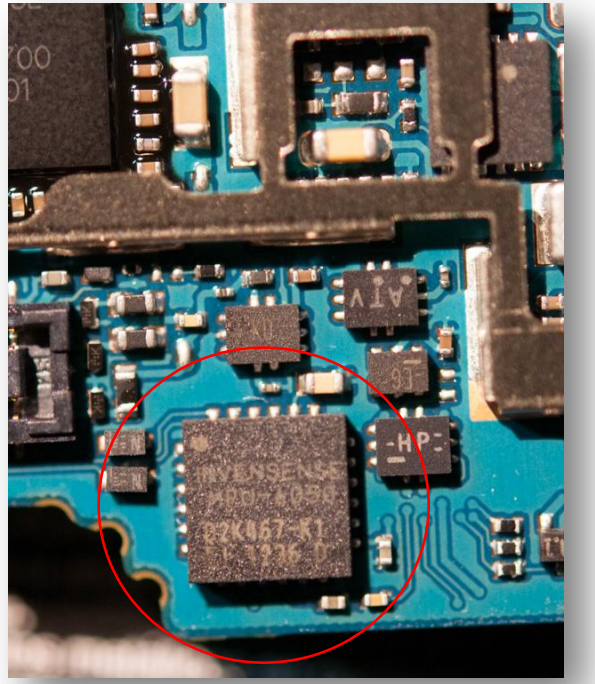
VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Location and Sensors

Michal Krumnikl



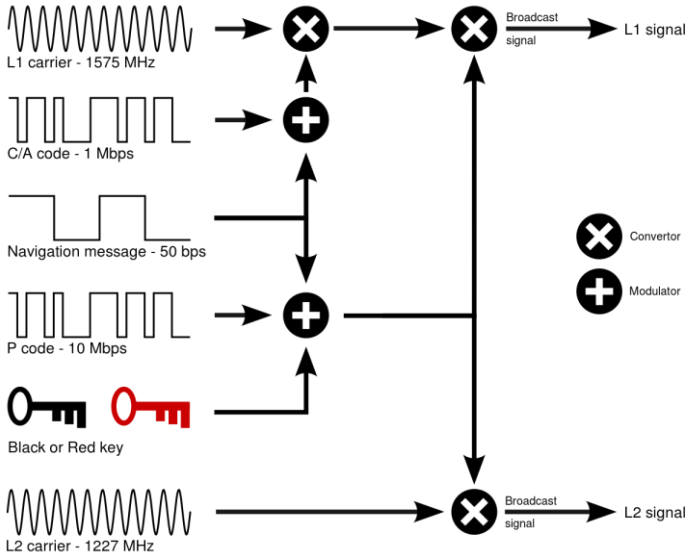
Source: EET Asia

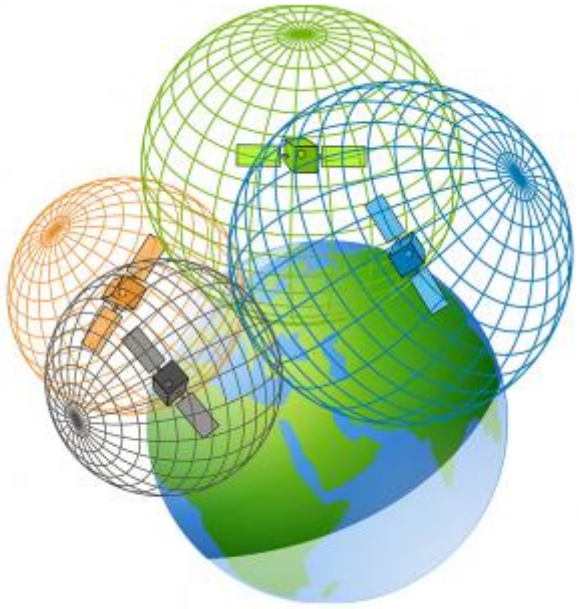


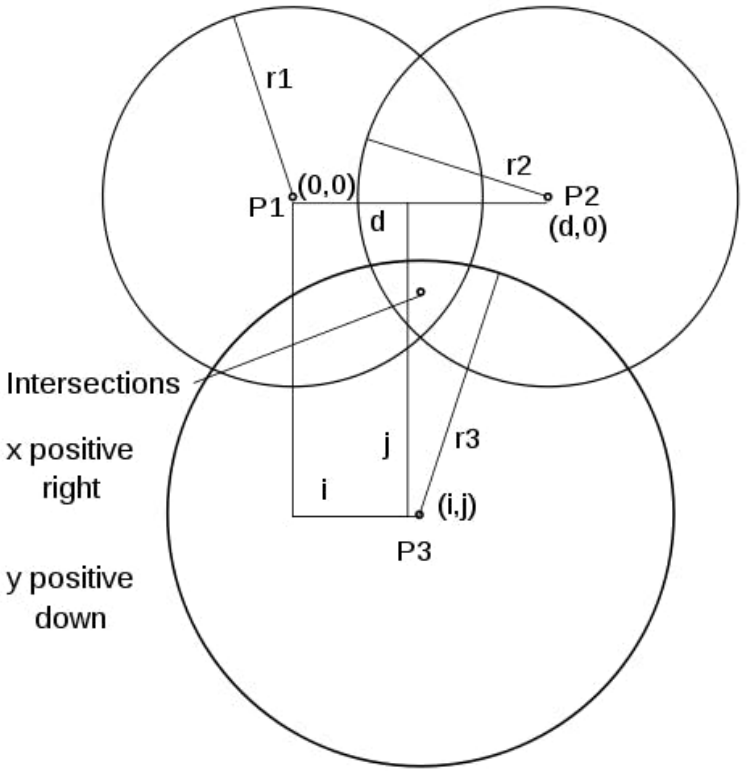
Source: www.anandtech.com

Location

- **GPS** is not only source of information.
- Based on known locations of **Wi-Fi** routers …
- Trilateration



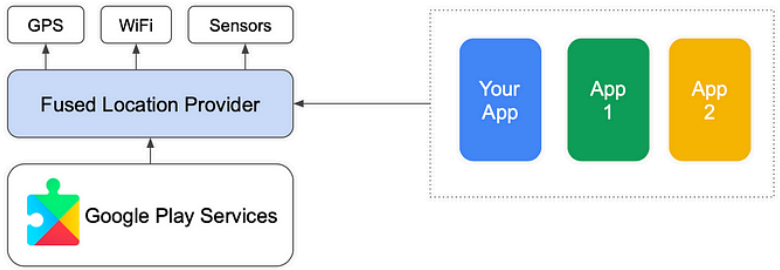




Source: Wikipedia, GPS: L1 C/A, L1 P(Y), L2 P(Y), SAASM; https://gisgeography.com/trilateration-triangulation-gps//

Location

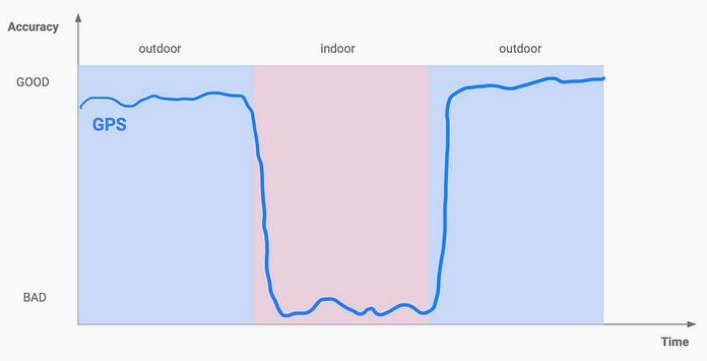
- **Fused Location Provider** - Google Location Services API is part of Google Play Services



- **GPS Module —** The dedicated module for location
- **WiFi Module** — WiFi-RTT (Round-trip-time) API, which is available in Android 9+
- **Sensors -** track the device movement

Location

- **Fused Location Provider** - Google Location Services API is part of Google Play Services

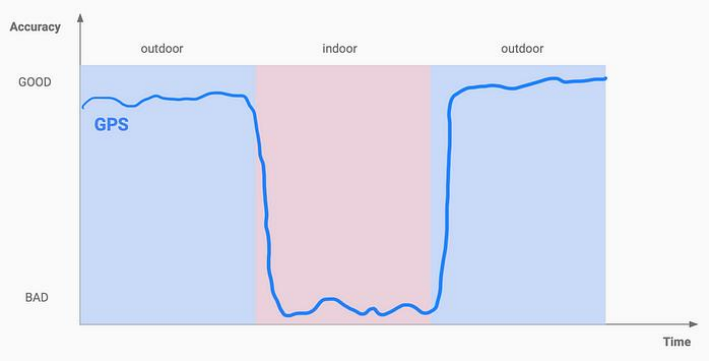


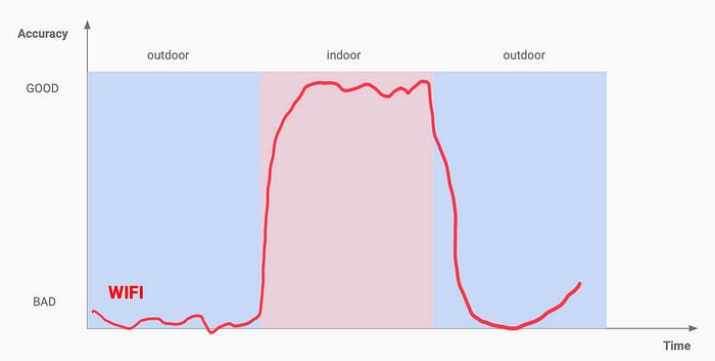
- **GPS Module —** The dedicated module for location
- **WiFi Module** — WiFi-RTT (Round-trip-time) API, which is available in Android 9+
- **Sensors -** track the device movement

Source: https://zhangqichuan.medium.com/understand-android-location-api-part-1-157ddb6df6a2

Location

- **Fused Location Provider** - Google Location Services API is part of Google Play Services



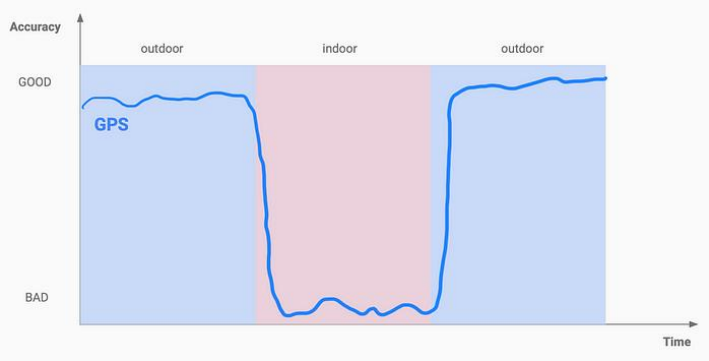


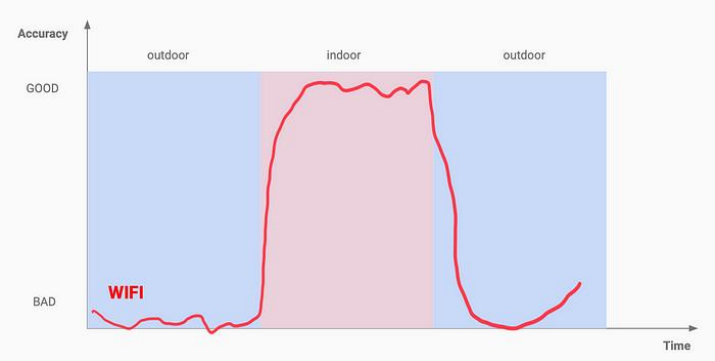
- **GPS Module —** The dedicated module for location
- **WiFi Module** — WiFi-RTT (Round-trip-time) API, which is available in Android 9+
- **Sensors -** track the device movement

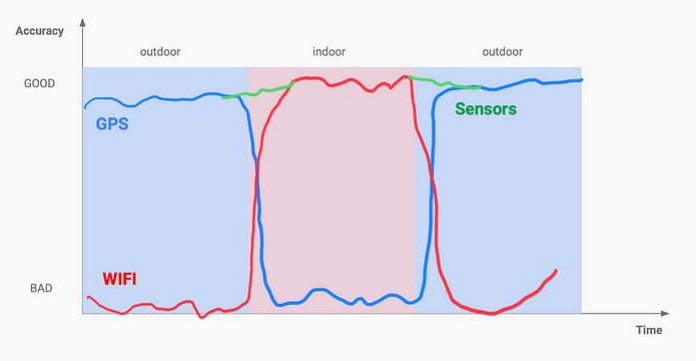
Source: https://zhangqichuan.medium.com/understand-android-location-api-part-1-157ddb6df6a2

Location

- **Fused Location Provider** - Google Location Services API is part of Google Play Services





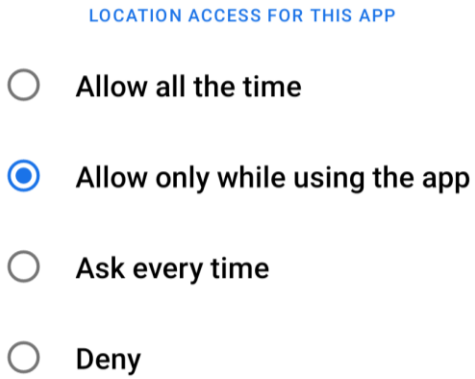


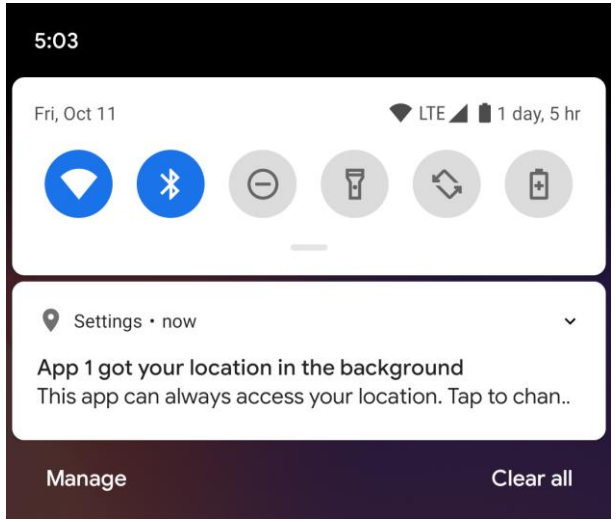
- **GPS Module —** The dedicated module for location
- **WiFi Module** — WiFi-RTT (Round-trip-time) API, which is available in Android 9+
- **Sensors -** track the device movement

Source: https://zhangqichuan.medium.com/understand-android-location-api-part-1-157ddb6df6a2

Location

- **Fused Location Provider** - Google Location Services API is part of Google Play Services –
  - **ACCESS_COARSE_LOCATION (Approximate location)**
  - **ACCESS_FINE_LOCATION (Precise location)**
- On Android 10 (API level 29) and higher, you must declare the **ACCESS_BACKGROUND_LOCATION** permission in your app's manifest in order to request background location access at runtime.





Location

- **Fused Location Provider** - Google Location Services API is part of Google Play Services -
  - **android.permission.ACCESS_COARSE_LOCATION** and **android.permission.ACCESS_FINE_LOCATION**

```
private FusedLocationProviderClient mFusedLocationClient;
protected void onCreate(Bundle savedInstanceState) {
    // ...
    mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

    mFusedLocationClient.getLastLocation()
        .addOnSuccessListener(this, new OnSuccessListener<Location>() {
            @Override
            public void onSuccess(Location location) {
                // Got last known location. In some rare situations this can be null.
                if (location != null) {
                    // Logic to handle location object
                }
            }
```

Location

- **Update interval**
  - setInterval() - sets the rate in milliseconds at which your app prefers to receive location updates.
  - setFastestInterval() - sets the fastest rate in milliseconds at which your app can handle location updates.
- **Priority**
  - PRIORITY_BALANCED_POWER_ACCURACY
  - PRIORITY_HIGH_ACCURACY
  - PRIORITY_LOW_POWER
  - PRIORITY_NO_POWER

```
protected void createLocationRequest() {
    LocationRequest mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(10000);
    mLocationRequest.setFastestInterval(5000);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY); }
```

Location

- **Update interval**
  - setInterval() - sets the rate in milliseconds at which your app prefers to receive location updates.
  - setFastestInterval() - sets the fastest rate in milliseconds at which your app can handle location updates.
- **Priority**
  - PRIORITY_BALANCED_POWER_ACCURACY
  - PRIORITY_HIGH_ACCURACY
  - PRIORITY_LOW_POWER
  - PRIORITY_NO_POWER

- In general, the higher the accuracy, the higher the battery drain.
- The more frequent location is computed, the more battery is used.
- Less latency usually requires more battery.

```
protected void createLocationRequest() {
    LocationRequest mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(10000);
    mLocationRequest.setFastestInterval(5000);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY); }
```

Location Performance

- **Background Location Limits introduced in Android 8.0 (API level 26)**
  - Background location gathering is throttled and location is computed, and delivered only a few times an hour.
  - Wi-Fi scans are more conservative, and location updates aren't computed when the device stays connected to the same static access point.
  - Geofencing responsiveness changes from tens of seconds to approximately two minutes.
- Recommendations
  - Starting updates based on the user's activity state
  - Consider if you really need to collect location in the background, since this can lead to undesirable battery drain. Also, consider geofencing as an option, since geofencing APIs are optimized for performance.

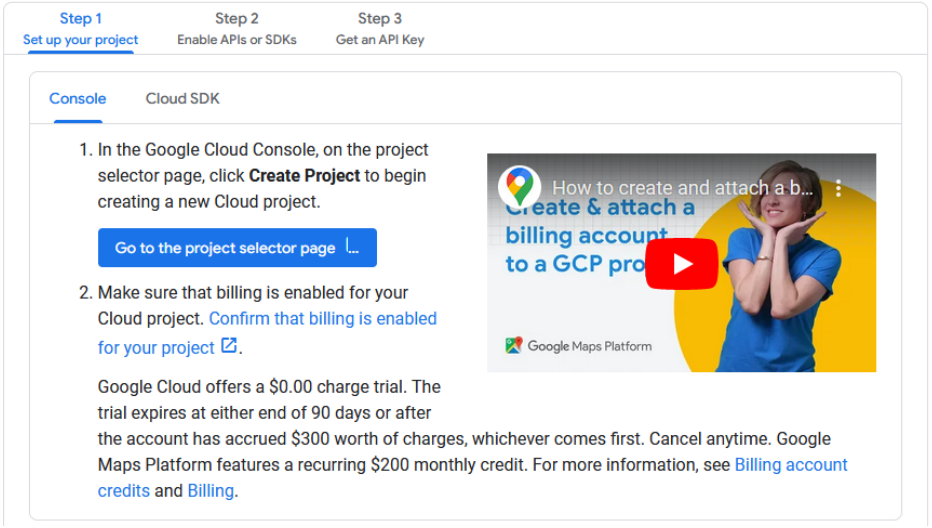
Geolocation API

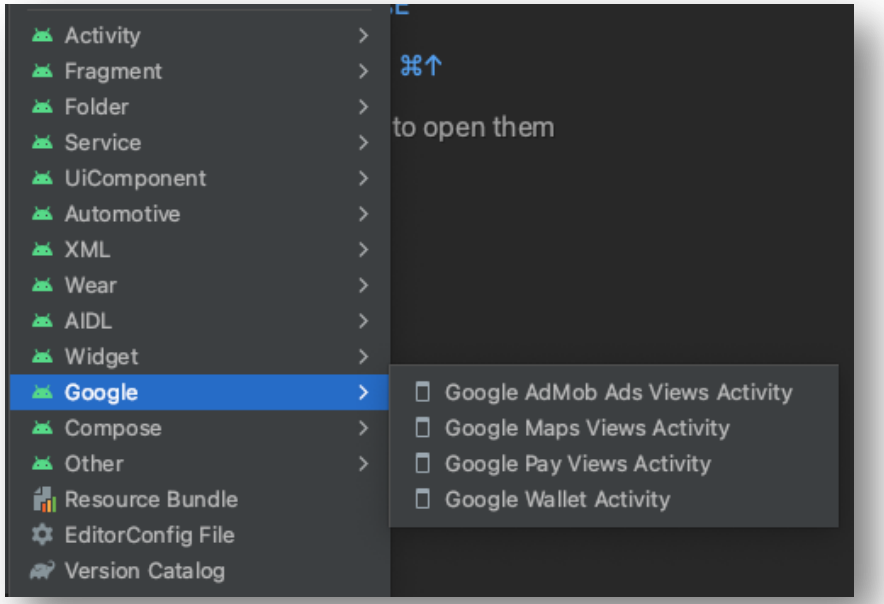
- Geolocation API returns a location and accuracy radius based on information about cell towers and WiFi nodes that the mobile client can detect.
- https://developers.google.com/maps/documentation/geolocation/

```
{
  "homeMobileCountryCode": 310,
  "homeMobileNetworkCode": 410,
  "radioType": "gsm",
  "carrier": "Vodafone",
  "considerIp": "true",
  "cellTowers": [
{
      "cellId": 21532831,
      "locationAreaCode": 2862,
      "mobileCountryCode": 214,
      "mobileNetworkCode": 7
    }
  ],
  "wifiAccessPoints": [
 {
      "macAddress": "00:25:9c:cf:1c:ac",
      "signalStrength": -43,
      "age": 0,
      "channel": 11,
      "signalToNoiseRatio": 0
}
```

Maps SDK

- Requires Android 4.0 or higher and Google APIs
- You need to set up your Google Cloud project
  - Set up project
  - Enable APIs or SDKs
  - **Get an API Key**





AndroidManifest.xml

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="${MAPS_API_KEY}" />
```

Maps SDK

Activity

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        SupportMapFragment mapFragment =
           (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney and move the camera
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new MarkerOptions()
                .position(sydney)
                .title("Marker in Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
```

build.gradle

```
dependencies {
    // Maps SDK for Android
    implementation 'com.google.android.gms:play-services-
maps:18.2.0'
}
```

XML

```
<fragment
xmlns:android="http://schemas.android.com/apk/res/androi
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/map"
    tools:context=".MapsActivity"
    android:name="com.google.android.gms.maps.SupportMapFragment
" />
```

Google Maps SDK Alternatives

- **OpenStreetMap based APIs -** https://wiki.openstreetmap.org/wiki/Android

Library	Minimum API level	License	Notes
Carto Mobile SDK	11	BSD	Supports 3D city overlays, editable vector overlays and is highly customizable.
CartoType	14	Proprietary	Offline rendering and routing library using OpenGL ES; customisable style sheets; 2.5D perspective view with 3D buildings
GLMap	14	Proprietary	Offline or online vector map rendered on device using OpenGL ES
LocationMapViewer	10	GPLv3+	Android-Intent; "geo:"-Uri; gpx and kml file/url
Mapbox Android SDK	15	BSD	Customizable, interactive vector maps styled in Mapbox Studio and rendered using OpenGL ES. Hybrid components available for Cordova, NativeScript, React Native, and Xamarin.
Mapbox Android Services	15	MIT	Connects to Mapbox's Static API
mapsforge	9	LGPLv3	Map rendering, map overlays, and more
Navmii Mobile SDK	?	Proprietary	
OSMBonusPack	10	LGPL with exceptions	Addon for osmdroid: Markers, Bubbles, Routes, Directions, KML and more...
osmdroid	7	Apache 2	OSM based replacement for Android's MapView (v1 API) class. Supports online and offline tile sources and overlays for plotting icons, tracking location, drawing shapes.
OsmSharp	?	GPLv2 or commercial	Offline vector map rendering in C# using Xamarin
Skobbler Android SDK	14	Proprietary	
Tangram ES	15	MIT	2D and 3D map renderer using OpenGL ES, used to have its own vector tile service but as Mapzen went bankrupt it no longer exists.
V™	10	GPLv3+	2D and 3D map renderer with own vector tile service; compatible with Mapsforge.
WhirlyGlobe-Maply	?	Apache 2	Geospatial display kit for iOS and Android based on OpenGL ES. Implements a 2D map and a 3D globe and can handle image base maps and tiled vector maps based on OSM data.

- **OpenLayers -** https://openlayers.org/
- **WMS libraries**
  - Maply - https://mousebird-consulting-inc.github.io/WhirlyGlobe/

WMS

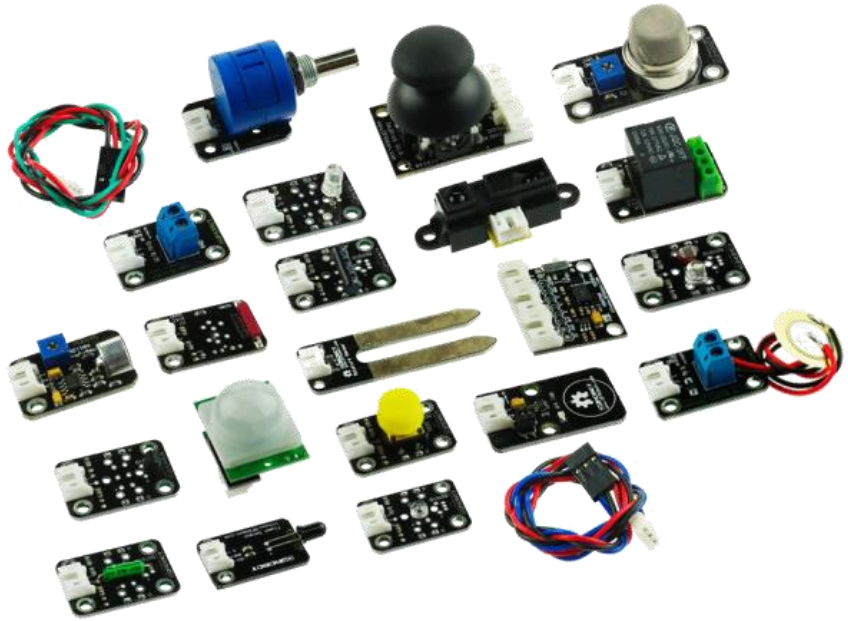
- **Web Map Service** (**WMS**)
  - WMS server usually serves the map in a bitmap format, e.g. PNG, GIF, JPEG, etc. In addition, vector graphics can be included, such as points, lines, curves and text, expressed in SVG or WebCGM format.
- Basic Methods
  - ***GetCapabilities*** – returns parameters about the WMS (such as map image format and WMS version compatibility) and the available layers (map bounding box, coordinate reference systems, URI of the data and whether the layer is mostly opaque or not)
  - ***GetMap*** – returns a map image. Parameters include: width and height of the map, coordinate reference system, rendering style, image format

https://ahocevar.com/geoserver/wms?REQUEST=GetMap&SERVICE=WMS&VERSION=1.3.0&FORMAT=image/png&STYLES=&TRANSPARENT=true&LAYERS=topp:states&WIDTH=821&HEIGHT=367&CRS=EPSG:3857&BBOX=-13887371.228454567,2866760.3006637404,-7452685.7715454325,5743166.588953207

- https://geoportal.cuzk.cz
- https://geoportal.gov.cz

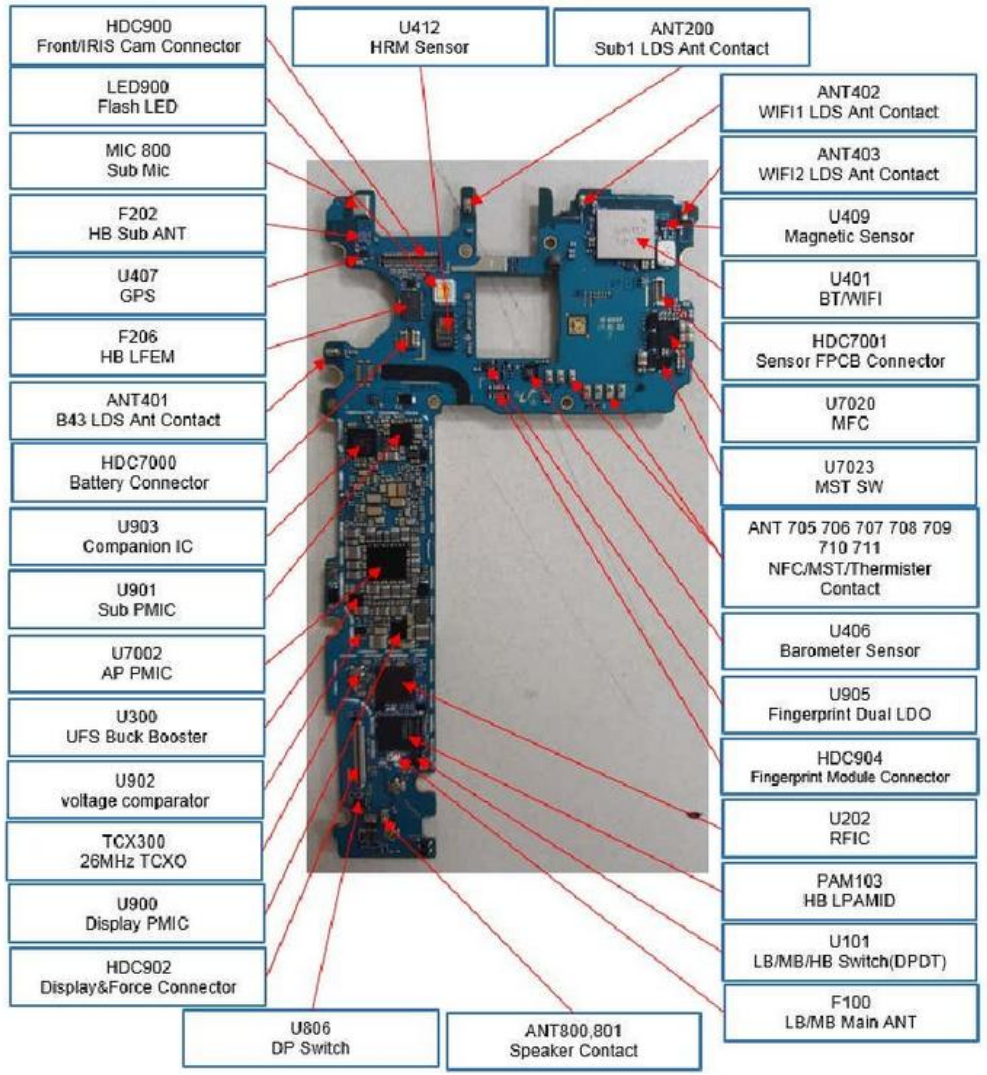
Sensors

- *„A sensor is a device that converts real world data (Analog) into data that a computer can understand using ADC (Analog to Digital converter)“ - Wikipedia*
- Sensors have been used in the mobile phone since the beginning.
- Android phones have around 10 sensors inside. Built-in sensors measure motion, orientation, and various environmental conditions.



Types of Sensors

- **Motion sensors**
  - These sensors measure acceleration forces and rotational forces along three axes, e.g. accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.
- **Environmental sensors**
  - These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, e.g. barometers, photometers, and thermometers.
- **Position sensors**
  - These sensors measure the physical position of a device.



Source: SM-G950F Tshoo 7 PDF

Types of Sensors

- **Motion sensors**
  - These sensors measure acceleration forces and rotational forces along three axes, e.g. accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.
- **Environmental sensors**
  - These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, e.g. barometers, photometers, and thermometers.
- **Position sensors**
  - These sensors measure the physical position of a device.

- Microphone
- Camera
- Temperature
- Location (GPS or Network)
- Orientation
- Accelerometer
- Proximity
- Pressure
- Light
- Fingerprint sensor

Interesting Applications

Resizing screen / tilt

Environment adjustment of apps, user comfort

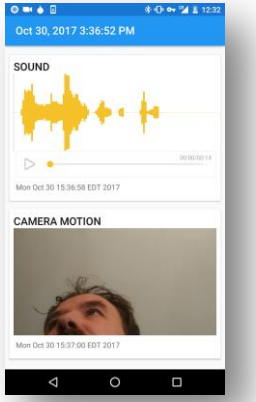
Gaming

AR Gaming / Navigation

Geo – tagging, grafitti, recomendations..

Network of objects, locations and people, 3D social

Distributed sensor system



Android in Near Space

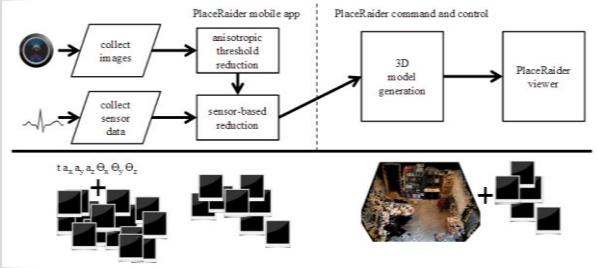
- http://android.hibal.org/

PlaceRaider

- https://arxiv.org/pdf/1209.5982.pdf

Haven

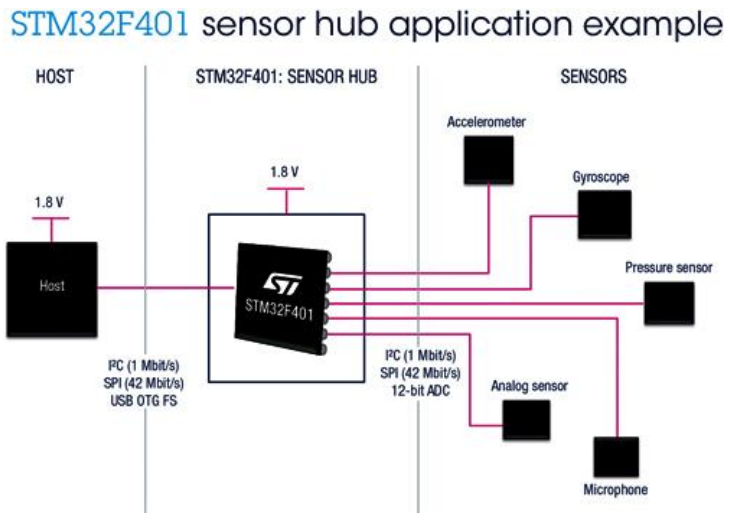
- https://guardianproject.github.io/haven/



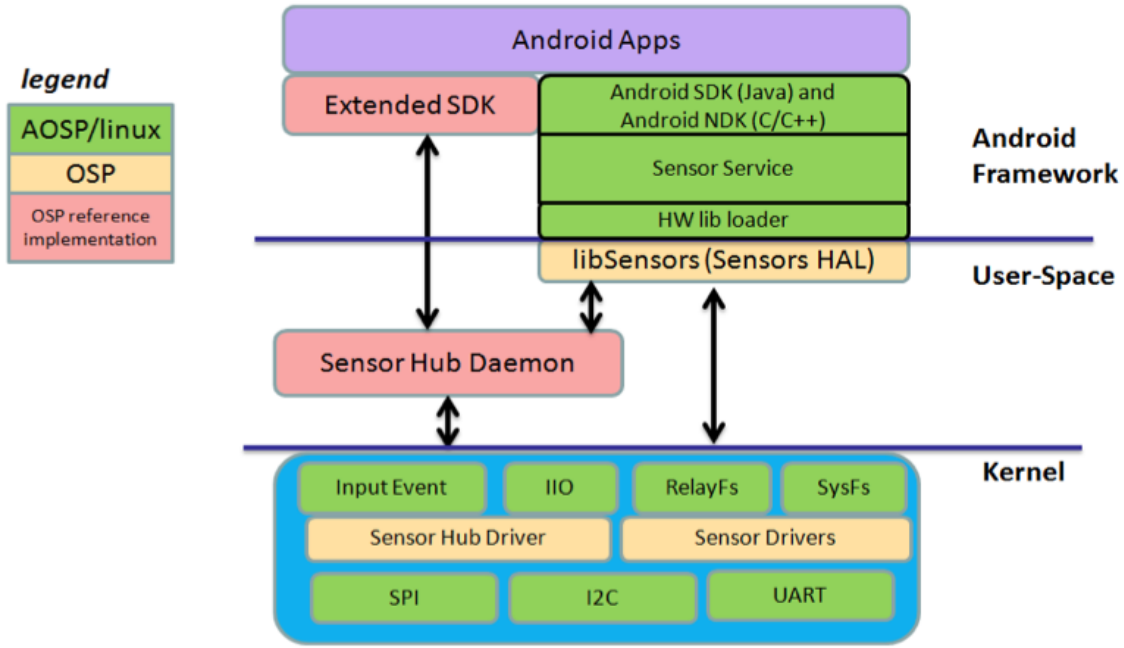
Sensor Hub

- Sensor chips are connected to the SoC through a **sensor hub**, allowing some low-power monitoring and processing of the data.
- To reduce power consumption, some architectures are hierarchical, with **processing done in the ASIC**.

Hardware



Software stack



Source: http://www.techdesignforums.com/blog/2014/06/25/sensor-hub-infrastructure-moves-open-source/

Sensors in Android

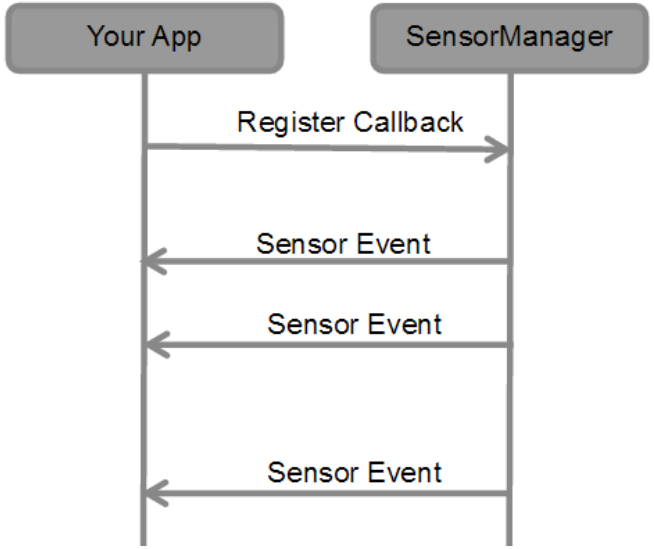
- **Continuous**
  - Events are generated at a constant rate.

    e.g. accelerometer, gyroscope
- **On-change**
  - Events are generated only if the measured values changed.

    e.g. step counter, proximity, heart rate sensor
- **One-shot**
  - Upon detection of an event, the sensor deactivates itself and then sends a single event through the HAL.

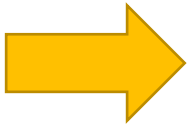
    e.g. significant motion

- Android's sensors are controlled by **external services** and **only send events** when they choose to.
- Each sensor has a related Listener interface that your callback must implement.



Android.hardware

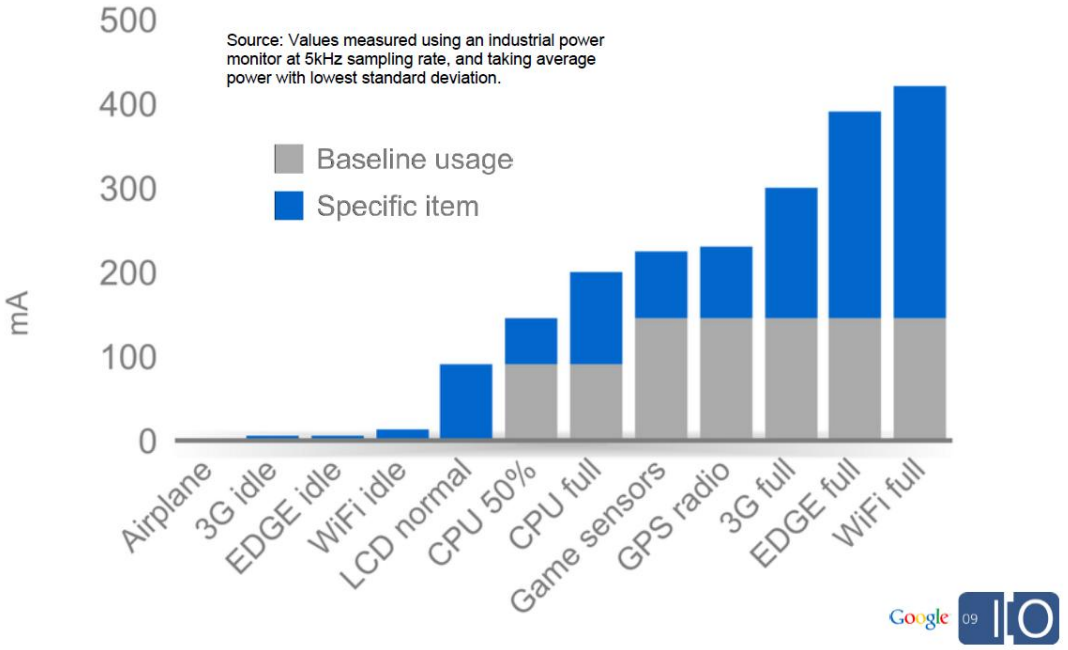
- **Sensor**
  - Class representing a sensor.
- **SensorAdditionalInfo**
  - This class represents a Sensor additional information frame, which is reported through listener callback onSensorAdditionalInfo.
- **SensorEvent**
  - This class represents a Sensor event and holds information such as the sensor's type, the time-stamp, accuracy and of course.
- **SensorEventCallback**
  - Used for receiving sensor additional information frames.
- **SensorManager**
  - SensorManager lets you access the device's sensors.
- **TriggerEvent**
  - This class represents a Trigger Event - the event associated with a Trigger Sensor.



Hardware manufacturer agnostic

Sensor Methods

- **float getMaximumRange()**
  - maximum range of the sensor in the sensor's unit.
- **int getMinDelay()**
  - the minimum delay allowed between two events in microsecond or zero.
- **String getName()**
  - name string of the sensor.
- **float getPower()**
  - the power in mA used by this sensor while in use.
- **public float getResolution()**
  - resolution of the sensor in the sensor's unit.



Source: Coding for Life—Battery Life, That Is .Jeff Sharkey. May 27, 2009

Sensor Workflow

- **Checking for sensor existence**

```
Sensor defaultGyroscope = sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
//(Returns null if none)

// Get a list of all sensors of a type:
    List<Sensor> pressureSensors = sensorManager.getSensorList(Sensor.TYPE_PRESSURE);

//Get a list of all sensors of a type:
    List<Sensor> allSensors  = sensorManager.getSensorList(Sensor.TYPE_ALL);
```

Sensor Workflow

- **Register SensorEvents**

```
public class MainActivity extends Activity implements SensorEventListener
{
     private SensorManager sm = null;

     public void onCreate(Bundle savedInstanceState) {
           sm = (SensorManager) getSystemService(SENSOR_SERVICE);
     }
     protected void onResume() {

     List<Sensor> typedSensors =  sm.getSensorList(Sensor.TYPE_LIGHT);
         // also: TYPE_ALL
     if (typedSensors == null || typedSensors.size() <= 0) ... error...
     sm.registerListener(this, typedSensors.get(0), SensorManager.SENSOR_DELAY_GAME);
         // Rates: SENSOR_DELAY_FASTEST, SENSOR_DELAY_GAME
     }
}
```

Sensor Workflow

- **Process Events**
- Properties used to describe a Sensor event:
  - **Sensor**
    - The sensor that triggered the event.
  - **Accuracy:**
    - The accuracy when the event occurred.
  - **Values:**
    - A float array that contains the new value(s).
  - **Timestamp:**
    - The time in nanosecond at which the event **occurred.**

```
public class MainActivity extends Activity implements
SensorEventListener
 {

     private float currentValue;
     private long lastUpdate;

     public void onSensorChanged(SensorEvent event) {
          currentValue = event.values[0];
          lastUpdate = event.timestamp;
     }

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

Base and Composite Sensors

- **Base sensor** types are named after the physical sensors they represent. These sensors relay data from a single physical sensor
  - `SENSOR_TYPE_ACCELEROMETER`
  - `SENSOR_TYPE_GYROSCOPE`
  - `SENSOR_TYPE_MAGNETOMETER`
- A **composite sensor** generates data by processing and/or fusing data from one or several physical sensors.
  - Step detector / significant motion
  - Game rotation vector
  - Uncalibrated gyroscope

Low power sensor

Sensor type	Type	Underlying physical sensors	Reporting mode
Game rotation vector	Attitude	Accelerometer, Gyroscope MUST NOT USE Magnetometer	Continuous
Geomagnetic rotation vector	Attitude	Accelerometer, Magnetometer, MUST NOT USE Gyroscope	Continuous
Glance gesture	Interaction	Undefined	One-shot
Gravity	Attitude	Accelerometer, Gyroscope	Continuous
Gyroscope uncalibrated	Uncalibrated	Gyroscope	Continuous
Linear acceleration	Activity	Accelerometer, Gyroscope (if present) or Magnetometer (if gyro not present)	Continuous
Magnetic field uncalibrated	Uncalibrated	Magnetometer	Continuous
Orientation (deprecated)	Attitude	Accelerometer, Magnetometer PREFERRED Gyroscope	Continuous
Pick up gesture	Interaction	Undefined	One-shot
Rotation vector	Attitude	Accelerometer, Magnetometer, AND (when present) *Gyroscope*	Continuous
Significant motion	Activity	Accelerometer (or another as long as very low power)	One-shot
Step counter	Activity	Accelerometer	On-change
Step detector	Activity	Accelerometer	Special
Tilt detector	Activity	Accelerometer	Special
Wake up gesture	Interaction	Undefined	One-shot

Light and Proximity Sensor

- **TYPE_LIGHT (Reporting-mode: On-change)**
- Return ambient light level in SI lux units
- SensorManager's constants
  - LIGHT_CLOUDY: 100
  - LIGHT_FULLMOON: 0.25
  - LIGHT_NO_MOON: 0.001
  - LIGHT_OVERCAST: 10000.0 (cloudy)
  - LIGHT_SHADE: 20000.0
  - LIGHT_SUNLIGHT: 110000.0
  - LIGHT_SUNLIGHT_MAX: 120000.0

Source: http://siansmobiles.co.uk

- **TYPE_PROXIMITY (Reporting-mode: On-change)**
  - A proximity sensor reports the distance from the sensor to the closest visible surface.
  - Up to Android KitKat, the proximity sensors were always wake-up sensors, waking up the SoC when detecting a change in proximity. After Android KitKat, we advise to implement the wake-up version of this sensor first, as it is the one that is used to turn the screen on and off while making phone calls.

Temperature, Pressure and Humidity Sensor

- **TYPE_PRESSURE (Reporting-mode: Continuous)**
  - Reports the atmospheric pressure in hectopascal (hPa).
    - The readings are calibrated using temperature compensation, factory bias calibration, factory scale calibration
  - The barometer is often used to estimate elevation changes. To estimate absolute elevation, the sea-level pressure (changing depending on the weather) must be used as a reference.

- **TYPE_RELATIVE_HUMIDITY (Reporting-mode: On-change)**
  - A relative humidity sensor measures relative ambient air humidity and returns a value in percent.
- **AMBIENT_TEMPERATURE (Reporting-mode: On-change)**
  - Provides the ambient (room) temperature in degrees Celsius.

Heart Rate Sensor

- **TYPE_HEART_RATE (Reporting-mode: On-change)**
  - A heart rate sensor reports the current heart rate of the person touching the device.
  - The current heart rate in beats per minute (BPM) is reported and the status of the sensor is reported. In particular, upon the first activation, unless the device is known to not be on the body, the status field of the first event must be set to SENSOR_STATUS_UNRELIABLE.

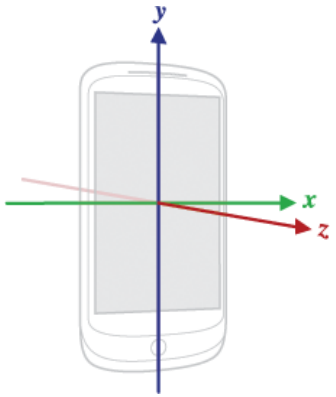
http://www.phonearena.com/

Magnetic field sensor

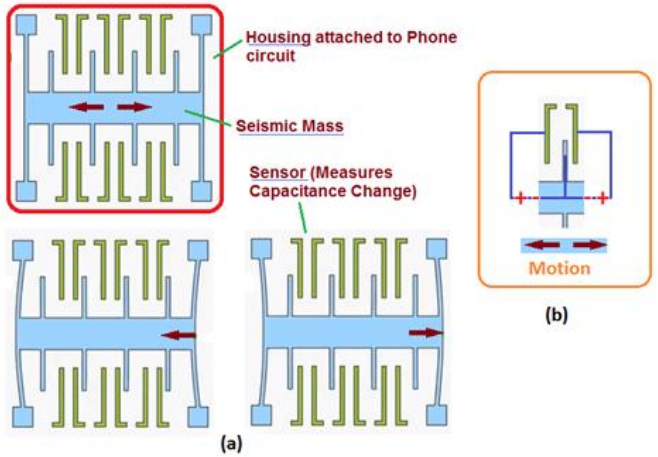
- **TYPE_MAGNETIC_FIELD (Reporting-mode: Continuous)**
  - A magnetic field sensor (also known as magnetometer) reports the ambient magnetic field, as measured along the 3 sensor axes.
  - The measurement is reported in the x, y and z fields of sensors_event_t.magnetic and all values are in micro-Tesla (uT).

Accelerometer

- Acceleration is defined as **the rate of change of velocity**. Measures how quickly the speed of the device is changing in a given direction.
- Detect movement and corresponding speed's rate of change. Accelerometers do not measure velocity.



- **Internal structure**
  - The "proof mass" shown above is allowed to move in a plane.
  - The attached fingers form a capacitor with the two plates around it.
  - The rate of change of the capacitance is measured and translated into an acceleration



Accelerometer

- **TYPE_ACCELEROMETER (Reporting-mode: Continuous)**
  - Reports the acceleration of the device along the 3 sensor axes. The measured acceleration includes both the physical acceleration (change of velocity) and the gravity.
  - The readings are calibrated using: temperature compensation, online bias calibration, online scale calibration

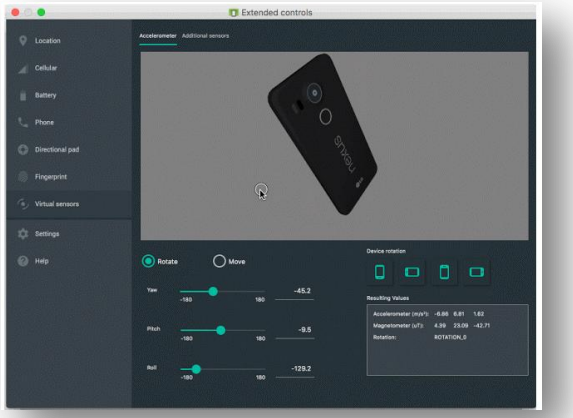
https://developer.android.com/reference/android/hardware/SensorManager.html

float	GRAVITY_DEATH_STAR_I Gravity (estimate) on the first Death Star in Empire units (m/s^2)
float	GRAVITY_EARTH Earth's gravity in SI units (m/s^2)
float	GRAVITY_JUPITER Jupiter's gravity in SI units (m/s^2)
float	GRAVITY_MARS Mars' gravity in SI units (m/s^2)
float	GRAVITY_MERCURY Mercury's gravity in SI units (m/s^2)

Accelerometer

- **TYPE_ACCELEROMETER (Reporting-mode: Continuous)**
  - Phone laying on the table rarely gives [0, 0, -1]
  - Calibration + filtration needed

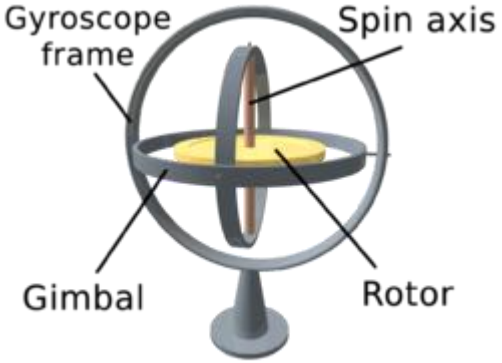
Emulator supports simulated environment

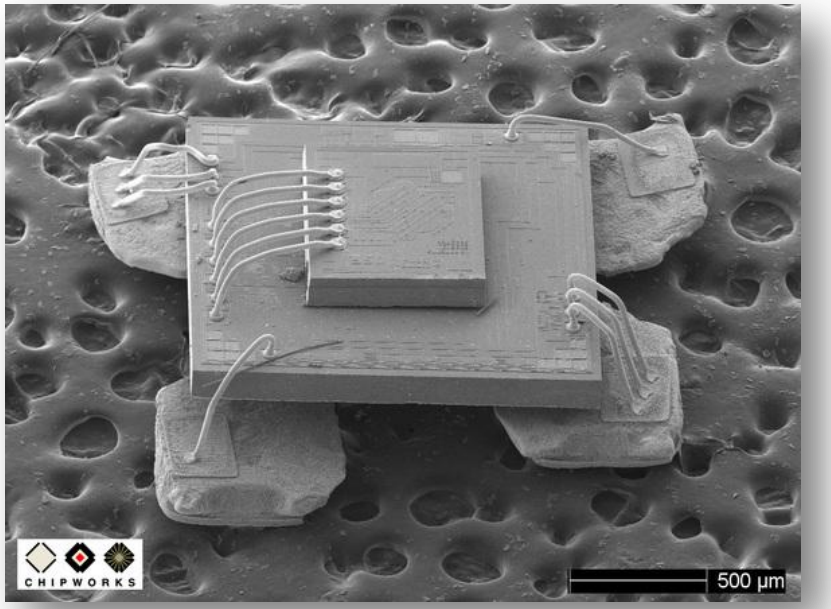



```
const float kFilteringFactor = 0.1f;

float accel[3]; // previous iteration

accel[0] = acceleration.x * kFilteringFactor +
                    accel[0] * (1.0f - kFilteringFactor);
accel[1] = acceleration.y * kFilteringFactor +
                    accel[1] * (1.0f - kFilteringFactor);
accel[2] = acceleration.z * kFilteringFactor +
                    accel[2] * (1.0f - kFilteringFactor);

result.x = acceleration.x - accel[0];
result.y = acceleration.y - accel[1];
result.z = acceleration.z - accel[2];
```

Gyroscope

- **TYPE_GYROSCOPE (Reporting-mode: Continuous)**
  - Reports the rate of rotation of the device around the 3 sensor axes. Rotation is positive in the counterclockwise direction (right-hand rule).
  - The measurement is reported in the x, y and z field and all values are in radians per second (rad/s).





Source: iPhone 4 Gyroscope Teardown - iFixit

Gyroscope vs. Accelerometer

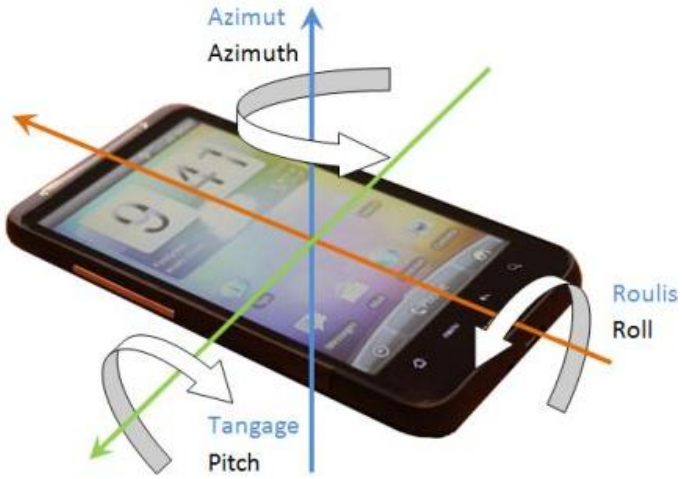
- **Accelerometer**
  - Senses linear movement, but worse rotations, good for tilt detection,
  - Does not know difference between gravity and linear movement, shaking, jitter can be filtered out, but the delay is added
- **Gyroscope**
  - measure all types of rotation
  - not movement
  - does not amplify hand jitter
- **Accelerometer + Gyroscope = both rotation and movement tracking possible**

```
SensorManager.getRotationMatrix
 (matrixR,
  matrixI,
  matrixAccelerometer,
  matrixMagnetic);
```

- ***matrixR*** – rotation matrix R
  - device coordinates -> world's coordinates
- ***matrixI*** - inclination matrix I
  - rotation around the X axis

Orientation Sensor

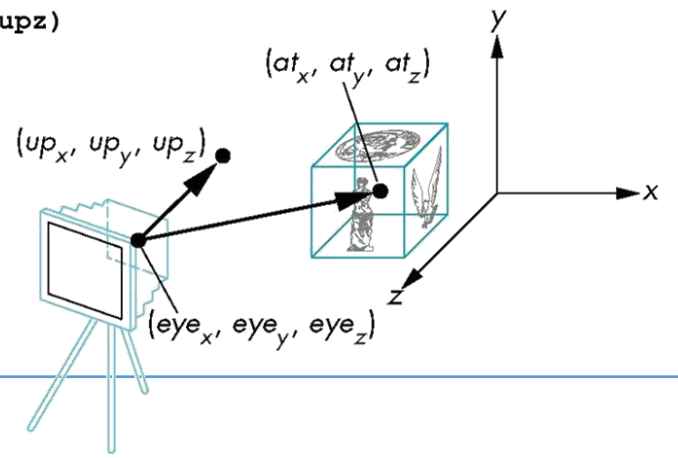
- Orientation Sensor is a combination of the **magnetic field Sensors**, which function as an electronic compass, and **accelerometers**, which determine the pitch and roll.
- Two alternatives for determining the device orientation.
  - Query the orientation Sensor directly
  - Derive the orientation using the accelerometers and magnetic field Sensors.
- x-axis (azimuth)
  - 0/360 degrees is north,
  - 90 east, 180 south, and 270 west
- y-axis (pitch)
  - 0 flat on its back, -90 standing upright.
- z-axis (roll)
  - 0 flat on its back, -90 is the screen facing left



Orientation with AR and OpenGL

```
float[] matrixR = new float[9];
float[] matrixI = new float[9];

SensorManager.getRotationMatrix(matrixR, matrixI, matrixAccelerometer, matrixMagnetic);

float[] lookingDir = Math3D.matrixMultiply(matrixR, new float[] {0.0f, 0.0f, -1.0f}, 3);
float[] topDir = Math3D.matrixMultiply(matrixR, new float[] {1.0f, 0.0f, 0.0f}, 3);

GLU.gluLookAt(gl,0.4f * lookingDir[0], 0.4f * lookingDir[1], 0.4f * lookingDir[2],
  lookingDir[0], lookingDir[1], lookingDir[2],
  topDir[0], topDir[1], topDir[2]);
```

```
glLookAt(eyex, eyey, eyez, atx, aty, atz, upx,
 upy, upz)
```



References

- Applications Links
  - https://techviral.net/android-apps-work-android-sensors/

Thank you for your attention

Mgr. Ing. Michal Krumnikl, Ph.D.

+420 597 325 867

michal.krumnikl@vsb.cz

www.vsb.cz