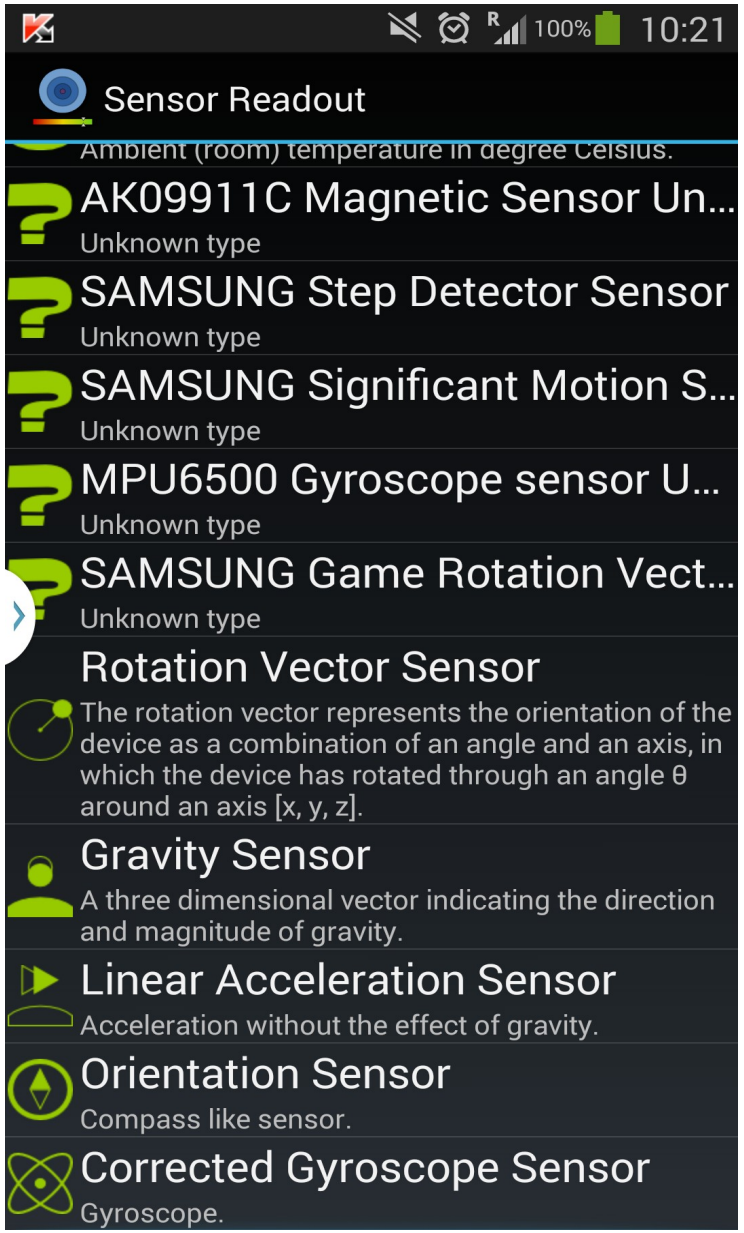
Android Sensors 101

Atilla Filiz **atilla@mind.be**

mind
embedded development
an ESSENSIUM division

Why

- A smartphone ...
  - is more than a phone
  - is more* than a PC
  - is aware of its surroundings
    - Acceleration
    - Rotation
    - Magnetism
    - Humidity
    - Air pressure
    - Temperature
    - Light
    - Proxymity



* Some restrictions apply

mind
embedded development
an ESSENSIUM division

Outline

- The Android software stack
  - Differences with “normal” Linux
- Android sensor architecture
  - Hardware components
  - Hardware abstraction layer
  - Sensor types
  - How to introduce new sensors
  - Power considerations
- Sensor fusion
  - What is it?
  - Implementation options

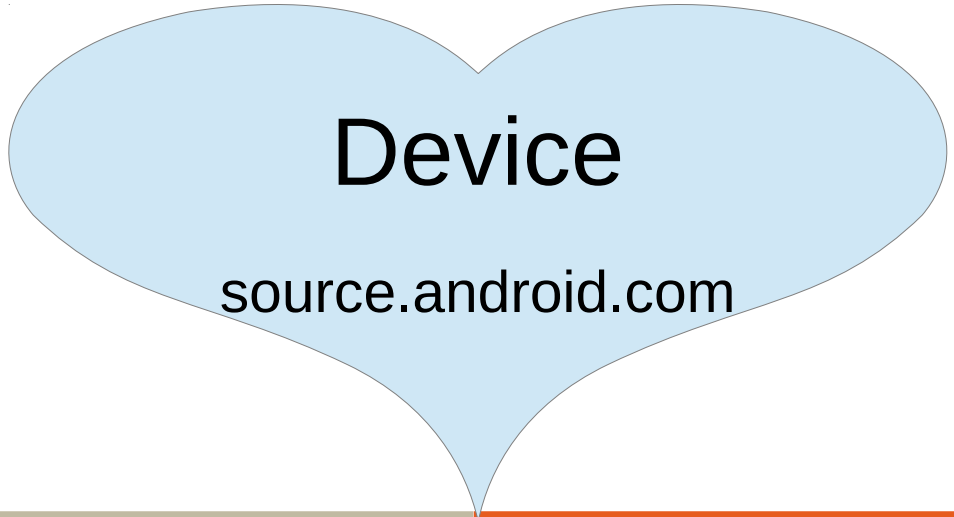
Android is an ...

- … API for developing Apps
  - Java based
  - But also compiled source code is possible
- … SDK for developing Apps
- … open source implementation of that API: AOSP
  - Based on a modified Linux kernel
  - Using a few other open source tools
  - With many packages © Google

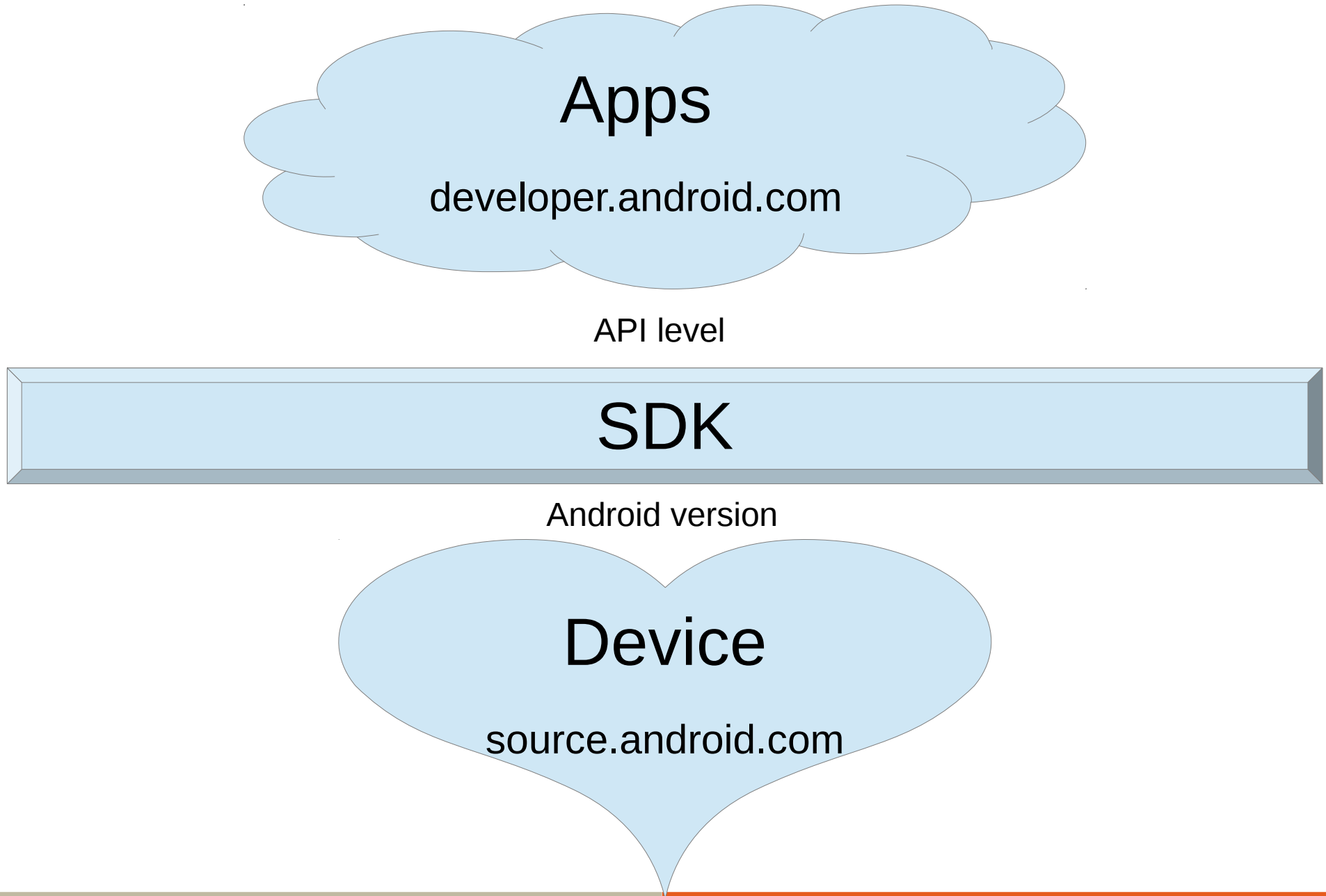
Androidisms and the Linux kernel

- Androidisms
  - Wakelocks (power management)
  - lowmem handler
  - Binder (IPC mechanism)
  - ashmem – Anonymous Shared Memory
  - RAM console
  - Logger
- Some are already in mainline(3.8+)
- Device makers lag behind the mainline ☹
- No change in device drivers or board files 😀
- SoC vendors do the kernel porting
- https://android.googlesource.com/kernel/common/

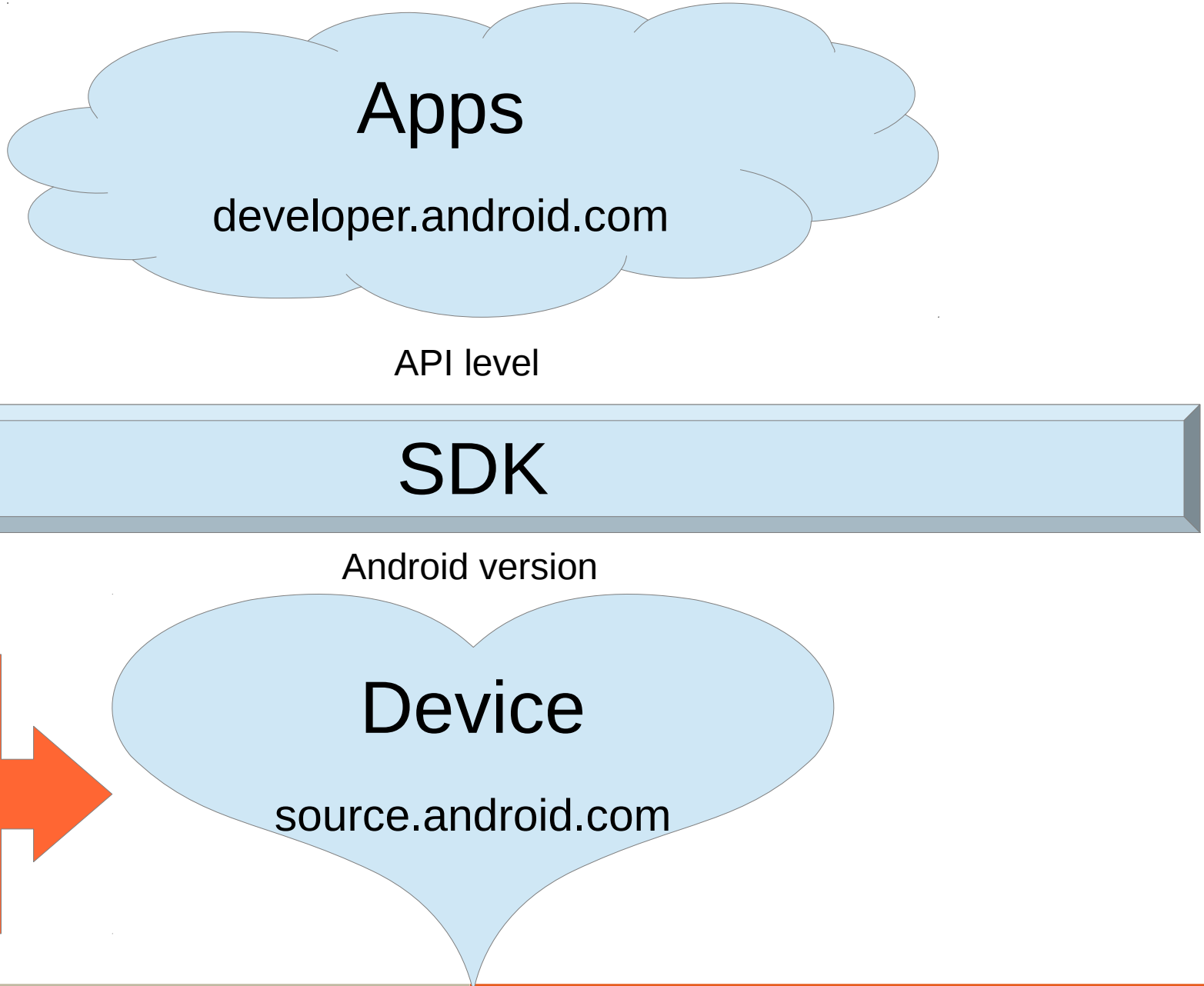
The Android wall



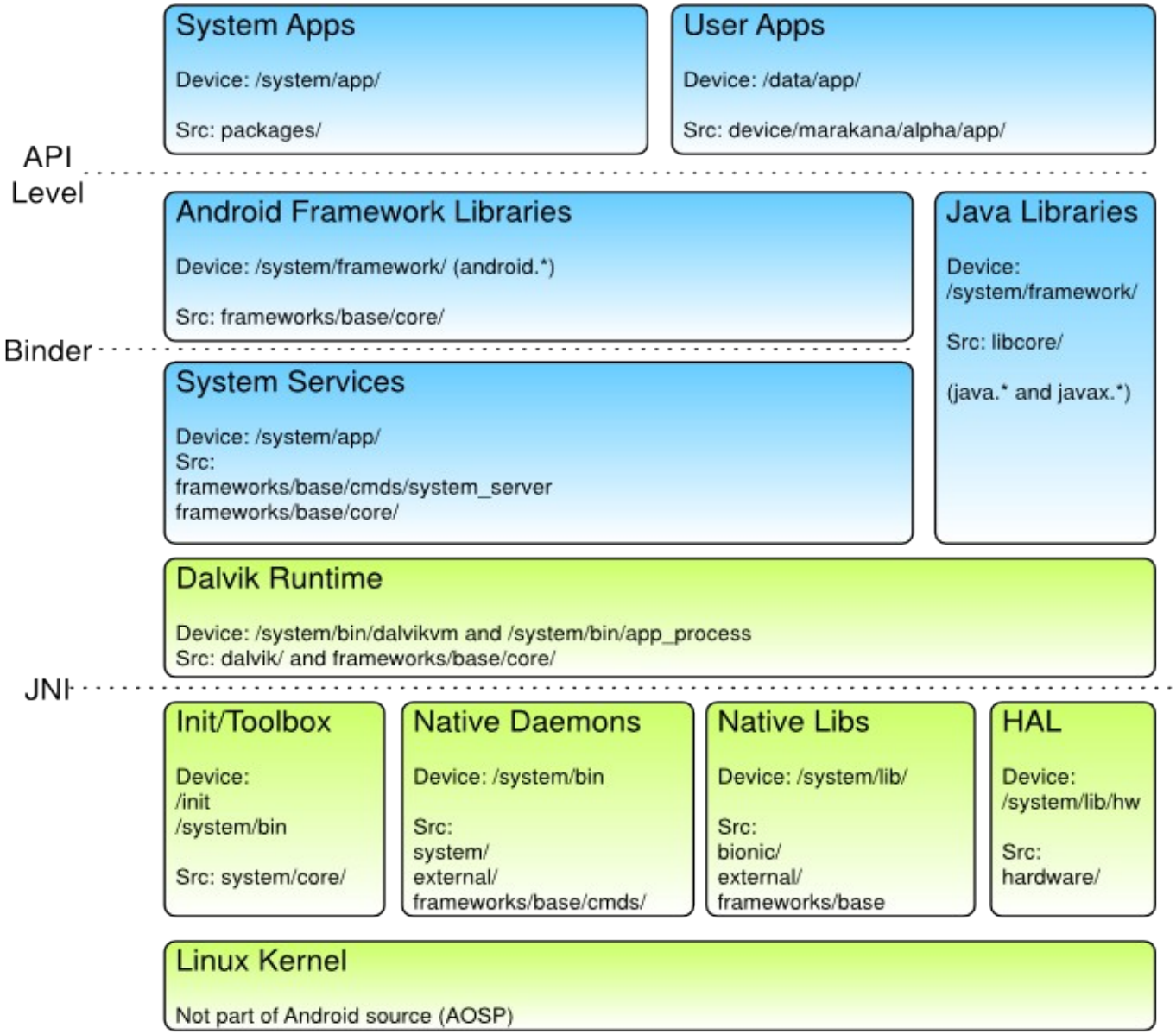
The Android wall



The Android wall



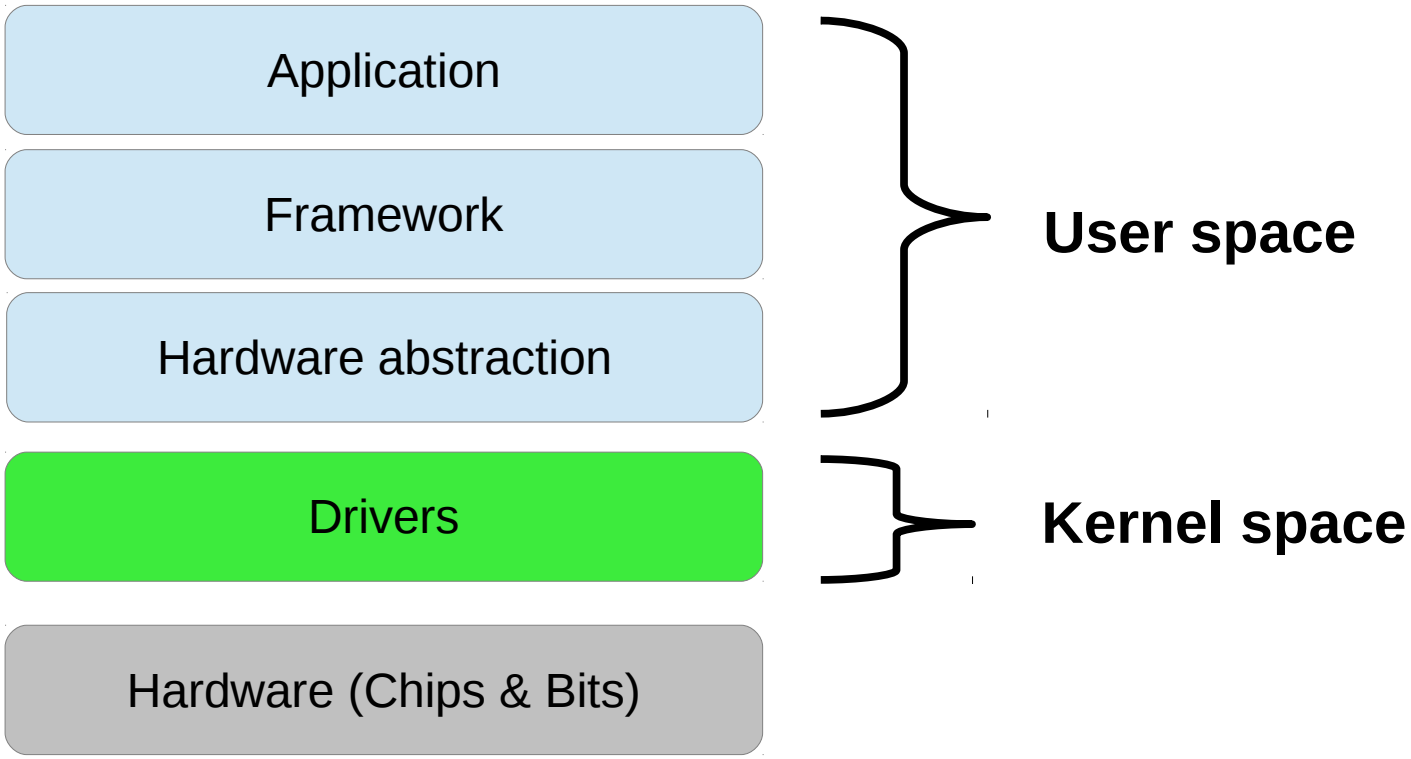
Overview of the Android stack



@MarkoGargenta

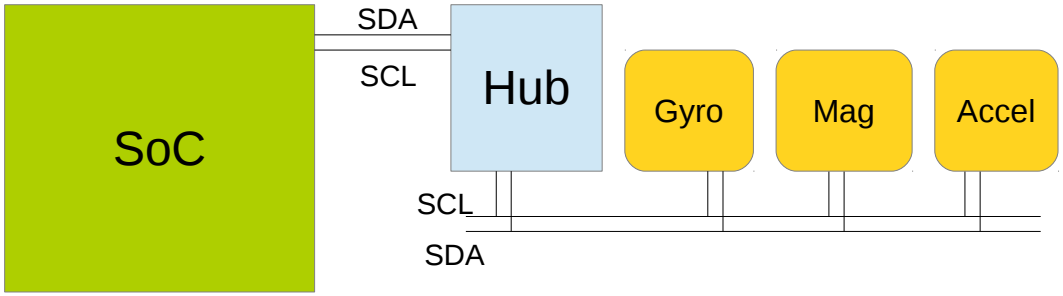
https://thenewcircle.com/s/post/1044/remixing_android

Android Sensor Architecture



Hardware

- Sensor chips
  - Sensor interface (i2c, spi, usb etc.)
- Sensor “hub”
  - Aggregate data
  - Control signals
    - Easy to manage
    - Power saving by turning off main CPU
  - Interrupts
  - Offload processing
    - Filtering
    - Fusion



Hardware Abstraction Layer

- Written in C++
- Main tasks are:
  - Communication via sysfs nodes
  - Callbacks for framework requests
- Sensors can be virtualized at this level

Hardware Abstraction Layer

hardware/invensense/libsensors_iio/CompassSensor.IIO.9150.h

```
class CompassSensor : public SensorBase {

public:
    CompassSensor();
    virtual ~CompassSensor();

    virtual int getFd() const;
    virtual int enable(int32_t handle, int enabled);
    virtual int setDelay(int32_t handle, int64_t ns);
    virtual int getEnable(int32_t handle);
    virtual int64_t getDelay(int32_t handle);

    // unnecessary for MPL
    virtual int readEvents(sensors_event_t *data, int count) { return 0; }

    int readSample(long *data, int64_t *timestamp);
    int providesCalibration() { return 0; }
    void getOrientationMatrix(signed char *orient);
    long getSensitivity();
    int getAccuracy() { return 0; }
    void fillList(struct sensor_t *list);
    int isIntegrated() { return (mI2CBus == COMPASS_BUS_SECONDARY); }
private:
    ….
}
```

hardware/invensense/libsensors_iio/Android.mk:

```
LOCAL_SRC_FILES += CompassSensor.IIO.9150.cpp
```

Hardware Abstraction Layer

device/samsung/tuna/libsensors/sensors.cpp

```
static struct sensor_t sSensorList[LOCAL_SENSORS + MPLSensor::numSensors] = {
      { "GP2A Light sensor",
          "Sharp",
          1, SENSORS_LIGHT_HANDLE,
          SENSOR_TYPE_LIGHT, powf(10, 125.0f/ 24.0f) * 4, 1.0f, 0.75f, 0, { } },
      { "GP2A Proximity sensor",
          "Sharp",
          1, SENSORS_PROXIMITY_HANDLE,
          SENSOR_TYPE_PROXIMITY, 5.0f, 5.0f, 0.75f, 0, { } },
      { "BMP180 Pressure sensor",
          "Bosch",
          1, SENSORS_PRESSURE_HANDLE,
          SENSOR_TYPE_PRESSURE, 1100.0f, 0.01f, 0.67f, 20000, { } },
};
```

Framework

- Implemented in Java
- Passes subscriptions to HAL
- Passes sensor events to apps
- No hardware specific code here
- Generic filtering/virtualization/fusion

Adding New Sensors

- Sensor hub firmware (if any)
- Kernel drivers
  - The usual Linux way
  - Interrupts, work queues etc.
- Hardware abstraction layer
  - Edit existing libsensors
    - Just add own sensor to sensors.cpp if you have a regular sensor
  - Add new library
    - For a new type of sensor
    - See hardware/libhardware/include/hardware/sensors.h

Sensor types(1/2)

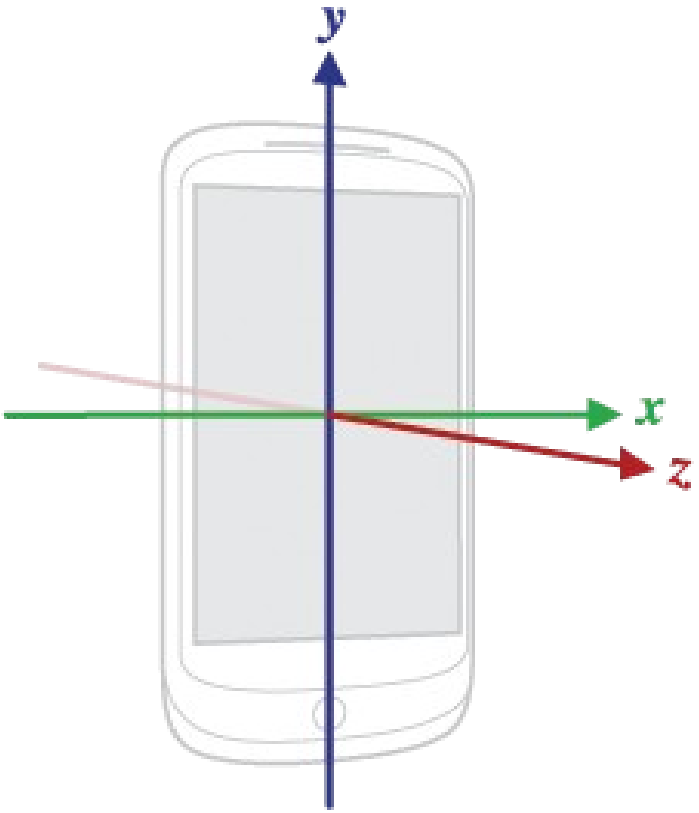
```
#define SENSOR_TYPE_ACCELEROMETER                   (1)
#define SENSOR_TYPE_GEOMAGNETIC_FIELD               (2)
#define SENSOR_TYPE_MAGNETIC_FIELD  SENSOR_TYPE_GEOMAGNETIC_FIELD
#define SENSOR_TYPE_ORIENTATION                     (3)
#define SENSOR_TYPE_GYROSCOPE                       (4)
#define SENSOR_TYPE_LIGHT                           (5)
#define SENSOR_TYPE_PRESSURE                        (6)
#define SENSOR_TYPE_TEMPERATURE                     (7)
#define SENSOR_TYPE_PROXIMITY                       (8)
#define SENSOR_TYPE_GRAVITY                         (9)
#define SENSOR_TYPE_LINEAR_ACCELERATION            (10)
#define SENSOR_TYPE_ROTATION_VECTOR                (11)
#define SENSOR_TYPE_RELATIVE_HUMIDITY              (12)
#define SENSOR_TYPE_AMBIENT_TEMPERATURE            (13)
#define SENSOR_TYPE_MAGNETIC_FIELD_UNCALIBRATED    (14)
#define SENSOR_TYPE_GAME_ROTATION_VECTOR           (15)
#define SENSOR_TYPE_GYROSCOPE_UNCALIBRATED         (16)
#define SENSOR_TYPE_SIGNIFICANT_MOTION             (17)
#define SENSOR_TYPE_STEP_DETECTOR                  (18)
#define SENSOR_TYPE_STEP_COUNTER                   (19)
#define SENSOR_TYPE_GEOMAGNETIC_ROTATION_VECTOR          (20)
```

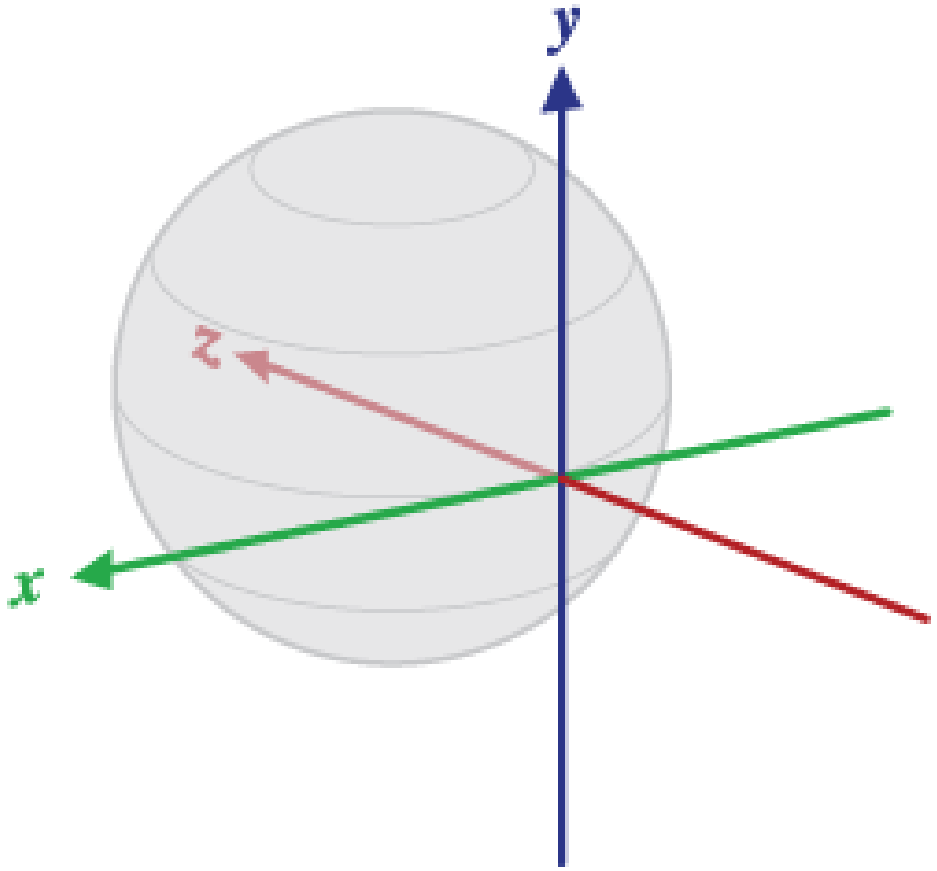
Sensor types(2/2)

- Real sensors:
  SENSOR_TYPE_GEOMAGNETIC_FIELD:
  Magnetic field strength in (x,y,z) relative to device, in uT, including all compensations
- Virtual sensors:

  SENSOR_TYPE_GEOMAGNETIC_ROTATION_VECTOR:
  Virtual sensor that gives rotation vector, using magnetometer.

Coordinate system





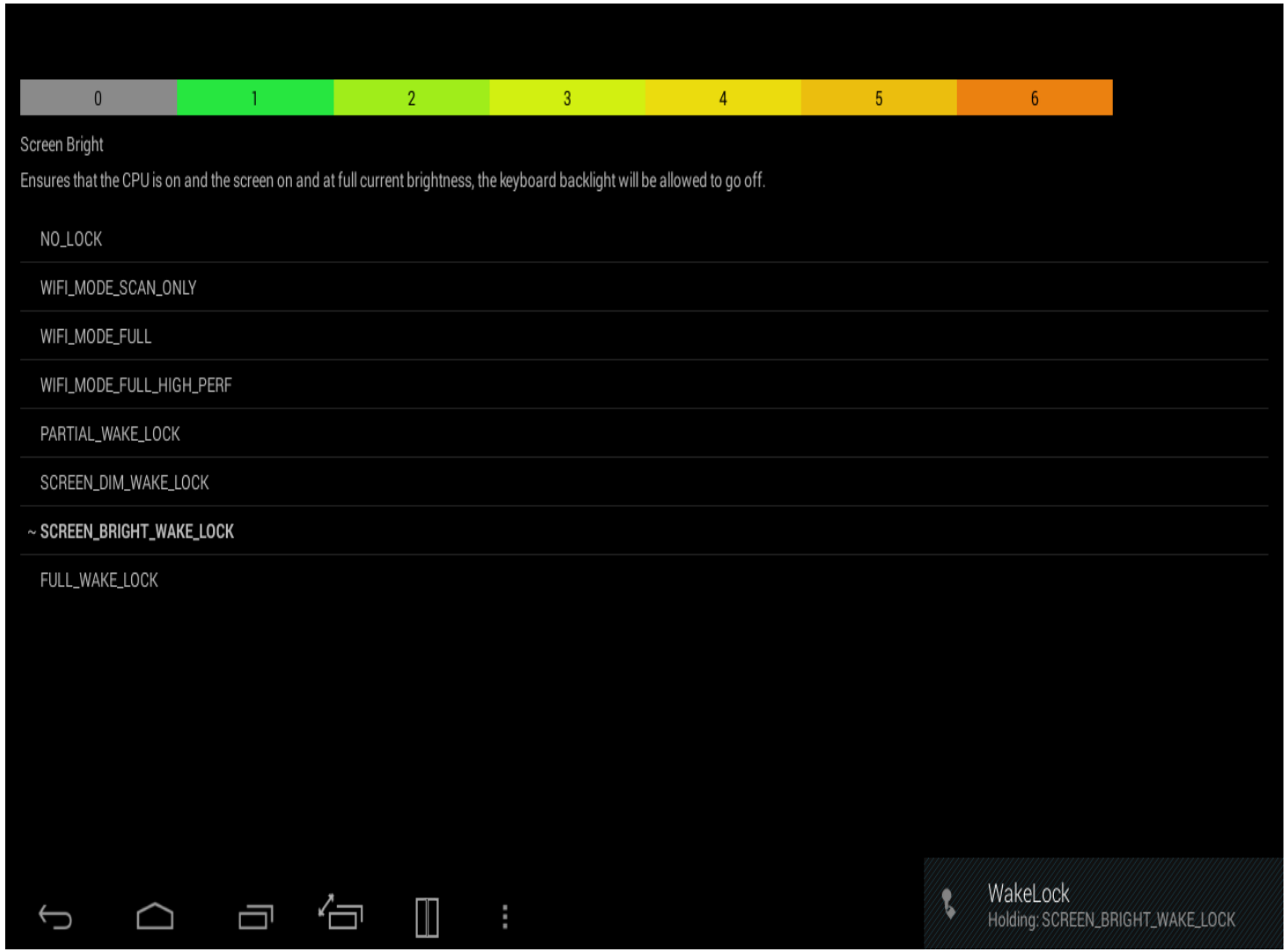
Sensor working modes

- Continuous
  - Fixed interval, set by setDelay()
- On change
  - When the value changes
  - With a minimum time between events
- One-shot
  - SW activates sensor
  - Sensor detects event/takes sample
  - Sensor deactivates self
  - SW event is created
- Special
  - Sensor type dependant

Power considerations

- Wake up main CPU?
  - The default answer is NO. Any generated events are dropped.
  - An app can hold a wakelock if always-on is desired.
  - Some sensors (eg. Proximity) always wake up the CPU.
- Respond to subscribe/unsubscribe.
  - Apps unsubscribe when they are not listening.
  - Turn off sensors when no subscription
  - Might need to keep on for e.g. continuous calibration
    - Add a timeout

Wakelocks

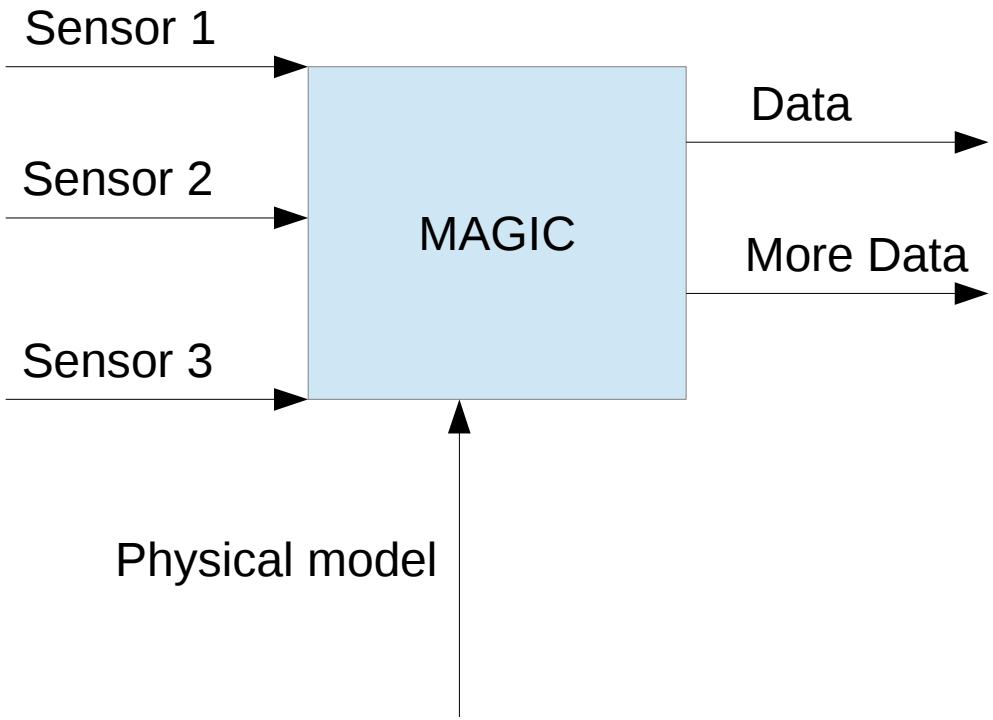


- Predefined in the API, no custom wakelocks
- No sensor specific locks, but that is not necessary

Sensor Fusion

- Higher accuracy
- Possibly faster convergence
- Higher power cost
- Can be done in different ways
  - In the hardware
  - In the framework
  - In the application

Sensor Fusion



Magic == Kalman Filter?

Simpler algorithms also possible

Sensor Fusion

Complementary filter(by Shane Colton, MIT)

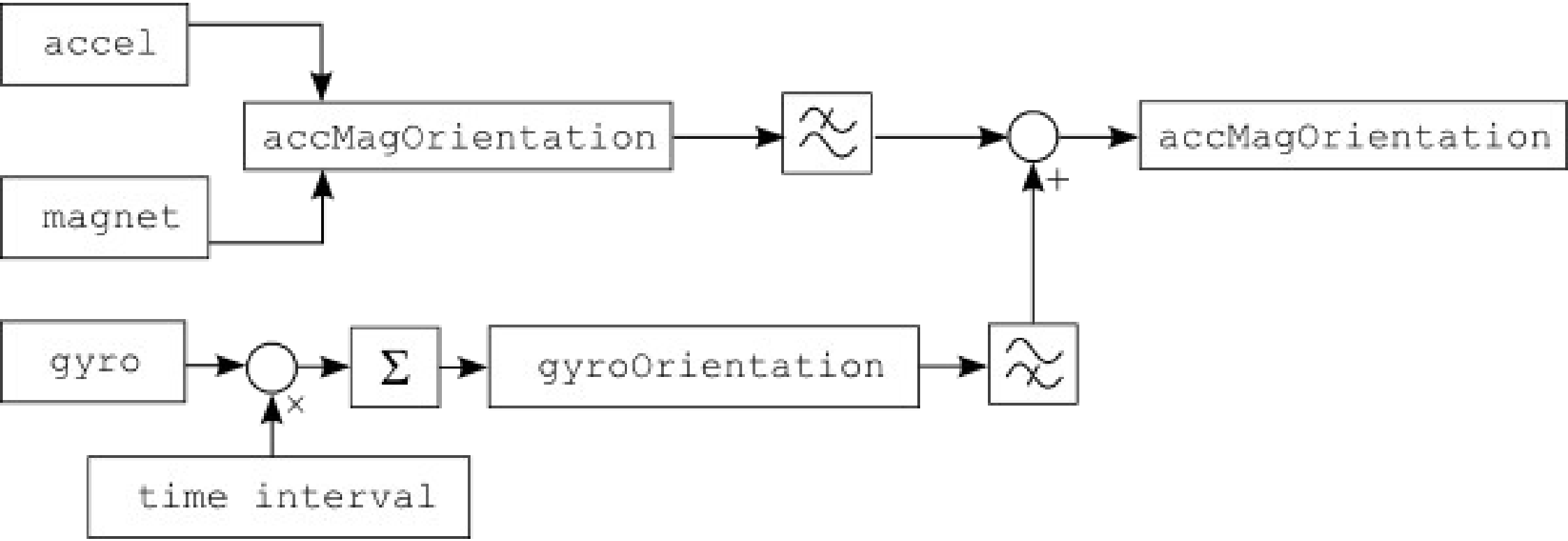


Image by Paul Lawitzky

Sensor Fusion may happen ...

- In hardware (sensor hub firmware)
  - If you are a device or sensor maker
  - Run the fusion algorithm on the hub.
  - Add a virtual sensor driver to the kernel.
- In the framework
  - If you are a device maker, AOSP contributor
  - Hardware agnostic
  - Exists in Android 4.x
- In app
  - If you are an app developer
  - Subscribe to multiple sensors
  - Process the events in your app logic
  - Use NDK if necessary

mind
embedded development
an ESSENSIUM division

That's all folks!

Questions?

mind
embedded development
an ESSENSIUM division

www.mind.be

www.essensium.com

Essensium NV
Mind - Embedded Software Division
Gaston Geenslaan 9, B-3001 Leuven
Tel : +32 16-28 65 00
Fax : +32 16-28 65 01
email : info@essensium.com

References

- http://developer.android.com
- Presentation on sensors: http://www.slideshare.net/cvs26/sensors-on-android-10220894
- Example fusion application: http://www.thousand-thoughts.com/2012/03/android-sensor-fusion-tutorial/
- SensorManager API for Apps: http://developer.android.com/reference/android/hardware/SensorManager.html
- Course page on sensor fusion: http://www.control.isy.liu.se/student/tsrt14/

Android userspace

- Custom, limited libc: bionic
- Custom, limited shell and tools
- Different directory structure
  - /system/ → System files, apps etc
  - /data/ → user data, storage for apps
  - /sdcard/ → mount point for default sdcard
- “Normal” Linux tree can be installed side by side if needed

Android codenames

Code name	Version	API level
	...	
Eclair	2.0	5
Eclair	2.0.1	6
Eclair	2.1	7, NDK 3
Froyo	2.2.x	8, NDK 4
Gingerbread	2.3 – 2.3.2	9, NDK 5
Gingerbread	2.3.3 – 2.3.7	10
	...	
Ice Cream Sandwich (ICS)	4.0.1 – 4.0.2	14, NDK 7
Ice Cream Sandwich (ICS)	4.0.3 – 4.0.4	15, NDK 8
Jelly Bean	4.1.x	16
Jelly Bean	4.2.x	17
Jelly Bean	4.3.x	18
KitKat	4.4	19

http://source.android.com/source/build-numbers.html

Sensor Fusion in Hardware

- Use an external uC as a sensor hub.
- Run the fusion algorithm on the hub.
- Add a virtual sensor driver to the kernel.
- Real-time accuracy
- Doesn't bother main CPU
- Might save power (main CPU off)
- Cost of additional hardware
- Hub firmware is device-dependent
  - But several sensors + hub can be bundled

Sensor Fusion in Framework

- Somewhat present in Android 4.
- Run the fusion in the framework code.
  - Virtual sensors are supported in the in the framework.
  - Generate events with fused data.
- Does not require additional hardware.
- Sensor-agnostic.
- Takes up main CPU time/energy.
- No real-time guarantee.

Sensor Fusion in App

- Platform-agnostic
- Probe raw-data and fuse to make sure.
  - Or probe for fused sensors and fall back on own code.
- Example: Orientation
  - Listen for acc, mag and gyro events.
  - Acc or mag event
    - getRotationMatrix(R, null, acc, mag); getOrientation(R, O_accmag);
  - Gyro event:
    - dt=event.time – prevtime; rot_gyro=gyro*dt;
    - O_final = k_gyr*rot_gyro + k_accmag*O_accmag
  - Or Kalman Filter
- Takes up main CPU time/energy.
- No real-time guarantee.

mind
embedded development
an ESSENSIUM division

So how do I do fusion?

- Nothing (Android 4.x has in framework)
- Hardware bundle
- Framework patch
- App