HAL

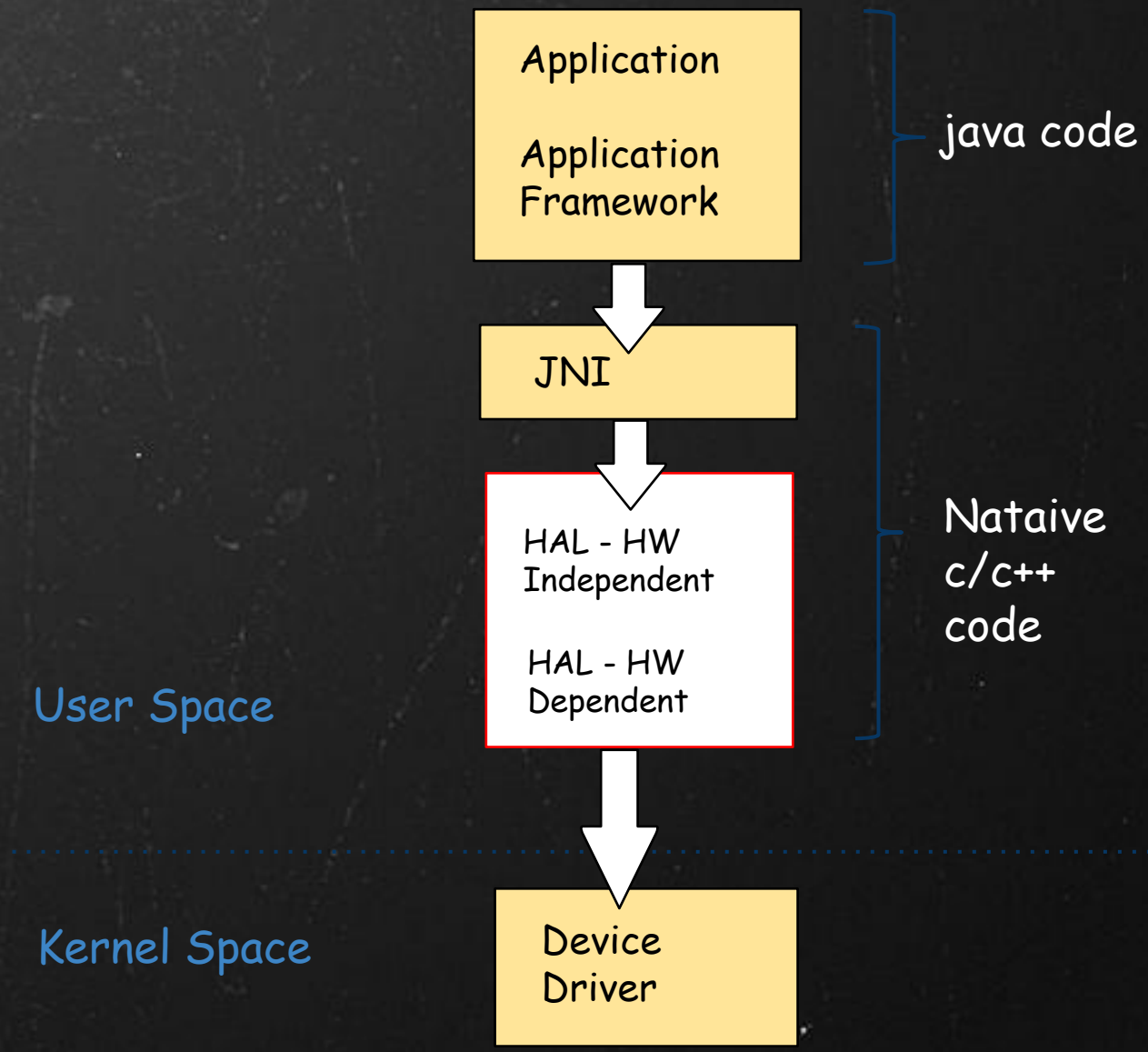
Hardware Abstraction Layer

Harry Chiu-hao Chen

Understanding of HAL

- HAL is everywhere, Qualcomm's HAL, Vendor's HAL, Linux's HAL...
- A kind of idea to cook some interfaces to abstract hardware platform
- Focus efforts onto HAL layer when migrating hardware
- Provide a set of universal APIs (for each drivers), such as HAL_SENSOR_LIGHT_XXX
  HAL_GPS_XXX
  HAL_DISPLAY_XXX
  ...

Software Layer



Code Study

Based on Qualcomm 7227 Android (R342)

- Light/ Proximity Sensor
- GPS

Light & Proximity Sensor - App

Two entries to trace light & proximity sensor

1) Sensor Manager - data
2) Sensor Service - control

CODE::

```
    static native int
sensors_data_init();
```

```
  private static native boolean
_sensors_control_activate(int sensor,
boolean activate);
```

FILE::

/android/framework/base/service/java/acom/android/server/SensorManager.java

/android/framework/base/service/java/acom/android/server/SensorService.java

Light & Proximity Sensor - JNI(1)

CODE::

FILE:

:/android/frameworks/base/core/jni/android_hardware_SensorManager.cpp

```
/* JNI interface, JAVA <-------> Native C/C++*/
static JNINativeMethod gMethods[] =
{
    {"nativeClassInit", "()V", (void*)nativeClassInit },
    {"sensors_module_init","()I", (void*)sensors_module_init },
    {"sensors_module_get_next_sensor","(Landroid/hardware/Sensor;I)I",
                                        (void*)
sensors_module_get_next_sensor },
    {"sensors_data_init", "()I",            (void*)sensors_data_init },
    {"sensors_data_uninit", "()I",          (void*)sensors_data_uninit },
    {"sensors_data_open",  "(Ljava/io/FileDescriptor;)I",  (void*)
sensors_data_open },
    {"sensors_data_close", "()I",           (void*)sensors_data_close },
    {"sensors_data_poll",  "([F[I[J)I",     (void*)sensors_data_poll },
};


/*Associate to HAL, using the same sensor_data_device_t struc ptr  to map
function*/
sensor_data_open(){
    ....
   return (sensors_data_device_t*) sSensorDevice->data_open(sSensorDevice,
fd);
}
```

Light & Proximity Sensor - JNI(2)

CODE::

```
static JNINativeMethod gMethods[] =
{
    {"_sensors_control_init",     "()I",    (void*)
android_init },
    {"_sensors_control_open",     "()
Landroid/os/ParcelFileDescriptor;",  (void*)
android_open },
    {"_sensors_control_activate", "(IZ)Z", (void*)
android_activate },
    {"_sensors_control_wake",     "()I", (void*)
android_data_wake },
    {"_sensors_control_set_delay","(I)I", (void*)
android_set_delay },
};
```

```
android_activate(JNIEnv *env, jclass clazz, jint
sensor, jboolean activate){
int active = sSensorDevice->activate(sSensorDevice,
sensor, activate);
}
```

FILE:

:/android/frameworks/base/service/jni/com_android_server_SensorService.cpp

HAL - Independent

CODE::

```
/*
HAL_Device_open()
Giving dev struct ptr to caller
And then giving underlying data_open pointing
to the HAL_Data_data_open function ptr,
Within HAL_XXX, function call will point to
vendor's sensor class (hw dependent)
*/

((sensors_data_device_t*) dev = &dev->common;
dev->date_open = HAL_Data_data_open;
dev->data_close = HAL_Data_data_close;
...
)
```

FILE::

android/hardware/msm7k/libecompass/yamaha/ms3cdriver/apache2.0/src/sensormodule_yamaha/hal_yamaha.cpp

HAL - HW Depedent

CODE::

```
        read (PSALS_fd, rawData, sizeof
(rawData));
            YAMAHA_LOGD("Ms3CSensor::
mActiveSensors=%x\n",mActiveSensors);
            YAMAHA_LOGD("Ms3CSensor::PSALS_fd
read raw[0]=%d\n",rawData[0]);
            YAMAHA_LOGD("Ms3CSensor::PSALS_fd
read raw[1]=%d\n",rawData[1]);
            if
(mActiveSensors&YAMAHA_FLAG_LIGHT)
            {
                if (mALSData.time != (int64_t)
rawData[0])
                {
                    //mALSData.vector.v[0] =
(float)(1.0*rawData[0]);
                    //mALSData.
Lightsensor_data = rawData[0];
                    //mALSData.time =
(int64_t)rawData[0];//time_start;
                    mALSData.lightSensorStatus
= (int64_t)rawData[0];
                    sensors_updated |=
YAMAHA_FLAG_LIGHT;
                }
            }
```

FILE::

android/hardware/msm7k/libecompass/yamaha/ms3cdriver/apache2.0/src/sensormodule_yamaha/SensorSystem.cpp

android/hardware/msm7k/libecompass/yamaha/ms3cdriver/apache2.0/src/sensor_yamaha/MS3C**Sensor.cpp**

After Study...

We want to know what's the good way to communicate through bottom (HAL) to the top (app/app framework)

Got some methods,
1) Just fill the right data structure from HAL (or driver) so that App can get it correctly via Android's official API
2) Add new data structures into standard SDK and cook a customized SDK to make new data/prototype possible
3) Just use signal/event system.

Case Study

- Story

We want to let FQC app get the correct light & proximity sensor status via Android's Sensor APIs but there's no these kinds of definition on Android's Sensor class.

so, just try to add it ...

Case Study - howto(1)

From bottom to the top, we need to modify the following data structures and files

1) [HAL-dependent]

```
read (PSALS_fd, rawData, sizeof(rawData));
mALSData.lightSensorStatus = (int64_t)rawData[0];
```

2) [HAL-independent]

```
//sensor.h
 sensors_data_t;
```

3) [JNI]

```
sensors_data_poll(JNIEnv *env, jclass clazz,
        jfloatArray values, jintArray status, jlongArray timestamp, jlongArray
proximityStatus, jlongArray lightStatus)
```

4) [App/SensorManager]

```
       final int sensor = sensors_data_poll(values, status, timestamp, proximityStatus,
lightStatus);
        listener.onSensorChangedLocked(sensorObject,
                                       values, timestamp, accuracy, proximityStatus,
lightStatus);
```

5) [App/FQC]

```
ShowALS.java/ShowProximity.java
```

Case Study - howto(2)

**

Add two fields into /frameworks/base/api/current.xml

```
<field name="lightStatus"
 type="long"
 transient="false"
 volatile="false"
 static="false"
 final="false"
 deprecated="not deprecated"
 visibility="public"
>
</field>
<field name="proximityStatus"
 type="long"
 transient="false"
 volatile="false"
 static="false"
 final="false"
 deprecated="not deprecated"
 visibility="public"
>
</field>
```

Built it !!

Ongoing Job

1. Find out why the values are not correct after sensor_data_poll??
2. Find out the way to use event