*2012 4th International Conference on Signal Processing Systems (ICSPS 2012)*
***IPCSIT vol. 58 (2012) © (2012) IACSIT Press, Singapore***
*DOI: 10.7763/IPCSIT.2012.V58.10*

# The Design of Bottom Layer Sensor Interfaces Based on Android OS

Jie Song [+]

Key Laboratory of Intelligent Computering and signal Processing, Anhui University, Ministry of Education
230039, China

**Abstract**. This paper focuses on the research about sensor module driver at bottom layer of popular Android OS. The main structure of Android is decomposed and illuminated, and the sensor system structure is decomposed in detail. The design of driver and Hardware Abstract Layer (HAL) is presented, and the steps, procedure, data structure of development about Sensor interfaces are also presented.

**Keywords**: android, driver, sensor, migration.

## 1. Introduction

As a new operation system (OS) developed by Google, Android has a rapid amazing developing speed [1]. This year, in American, the amount of smart phone with Android OS is in the first place. From smart phone to MID or other embedded application, more and more people believe it has good future and Android has been migrated to more and more different hardware platform. The number of programmer working for Android software increases quickly. However, most of programmer is working for applications. Because developing driver must have enough hardware knowledge, only a few people are working for hardware. Bottom layer is the foundation of OS, all software based on OS need its support for controlling hardware system[2][3].

## 2. Android software system

Android is a software collection. It includes OS, middle ware and key applications. Now the version of Android has been updated to version 3.2[4].

The kernel code of Android OS is based on Linux kernel. Its kernel is Linux 2.6 at present. In user space, its code is native code (c and c++) or Java code.

Since Java is independent of platform, a program can be easily migrated from a hardware platform to a new one.

From bottom to top, software system can be divided into four layers [5]:

First layer: Linux OS and drivers

Second layer: Native code framework and Java VSM

Third layer: Java framework

Fourth layer: Java application program

The structure is shown in Figure 1.

With this structure, driver is in the bottom layer of the software system. So its development method has obvious difference with application development.

[+] Corresponding author.
*E-mail address*: jsong@ahu.edu.cn.

Porting an OS to a new hardware platform, the key management of OS, such as memory management, process management, and network protocols has common features, usually have no large modification, and the main work is cropping system by system requirement. So the major migration work is developing drivers of various interfaces to different hardware platform [6].

The blocks in shadow of Figure 1 are the parts we should develop while porting a new hardware interface.

The major work of migration includes two parts [7]:

- Linux driver
- Hardware abstract layer of Android system

Drivers work in kernel space, Android hardware HAL works in user space. These two part's cooperation makes large Android system work on special hardware platforms.

Besides basic Linux OS, the main content is drivers of various devices. In Android system, some devices use standard driver code, such as framebuffer driver, Event input driver, Flash MTD driver, Wi-Fi driver, Bluetooth driver, serial port driver etc.

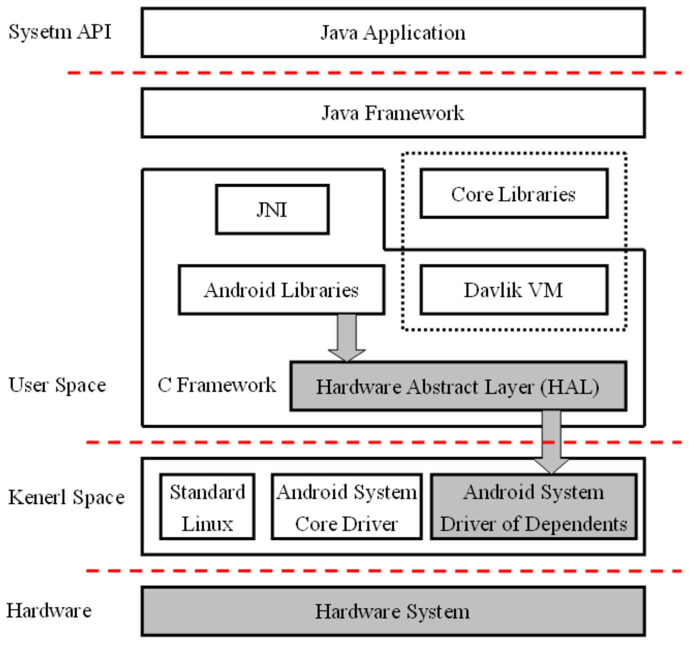

Fig.1. Major work of migration.

Other devices, such as vibrator, background light, power system, use sysfs as interface between kernel space and user space.

Sensors and GPS devices have not been specified with fixed type of driver. They can select different interface (kernel space to user space) to realize their function according to hardware environment.vbzx

The development of Android system drivers has characters of normal Linux environment, and it also has its own specific characters.

It means that in the bottom layer it use the develop way of Linux drivers, but in upper layer, to meet with the requirements of user space, it must have Hardware Abstract Layer (HAL).

HAL is a layer between the Android system controlled by user and Linux drivers in kernel space.

HAL separates direct relations between Android and hardware. It setup a relatively universal, standard interface, makes it easily be used by upper layer, and the programmer of applications can easily write code without knowing the details of hardware.

## 3. Android Sensor system

The sensor of Android system is used to acquire the external information, and the hardware is various sensors. The layer structure is in Fig. 2.

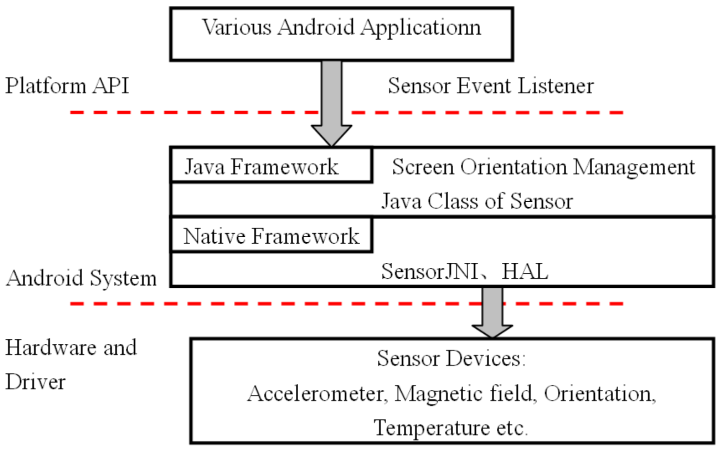

Fig.2. Basic layer structure of Android sensor system.

There are eight standard types of sensors, including accelerometer, magnetic field, orientation, gyroscope, light, pressure, temperature, proximity sensors. Each sensor uses different hardware to realize their function.

The interfaces of sensor system to upper layer are used to send the data and accuracy of sensor and it also provides the interface of accuracy. These interfaces are used in Java framework and Java application.

From bottom to top, there are drivers, HAL, sensor Java framework, the use of sensor in Java framework, Java application. It is shown in Fig. 3.

The framework code has following parts:

- The driver of sensor for different specified platform
- The hardware layer of sensor
- JNI (Java Native Interface) part of sensor system
- Java part of sensor system
- The calling of Java API of sensor in Java layer
- In Java layer, sensor system provides standard API of sensor platform.

In essence, the use of sensors in Android system is a procedure of upper layer program callbacked based on bottom layer drivers.

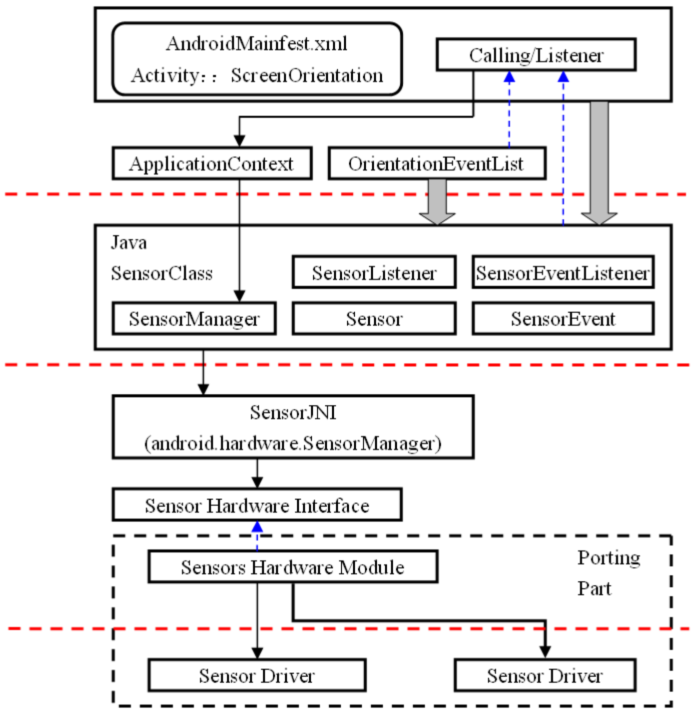

Fig.3. The structure of Android sensor system.

## 4. The program design of sensor interface

There is a great variety of different hardware in different system and the drivers are also designed with different methods.

As usual, the general method is:

First, be familiar with the interface of HAL.

Second, integrate and multiplex drivers the system already used.

The main work is in the realization of HAL.

To understand and debug sensor system, understanding and familiar with upper layer calling of HAL is necessary.

Classical way is to realize HAL and drivers, HAL calling drivers. In this method, Android system only concern HAL, and they do not concern drivers. The advantages of this method are that separating some function of Android from Linux drivers. Android dose not directly depend on Linux drivers.

The HAL of sensor is called by Sensor JNI, and sensor JNI is called by Java program. So the key work of sensor system's realization is HAL, Sensor's HAL must meet with the interfaces of HAL.

In some conditions, HAL is standard, so it only needs to write driver code. This kind of driver usually is standard driver of Linux.

However, sensor's driver is not standard.

### 4.1. Rrealization of driver

The sensor system under HAL is not standard. So sensor system must developing sensor driver and HAL.

Sensor's HAL use the interface of standard hardware module of Android. It is written by C language, realized basically by filling the function pointer.

Sensor's driver can be implemented with following interfaces:

- Event device
- Misc character device
- Direct use a master device of character device
- Using sys file system

Sensor must provide the mechanism relation with hardware: reading information, blocking, control. These three function's classical interfaces are read, poll, and ioctl. In fact, only reading is necessary.

Since sensor is a tool of collecting information, it is a very natural way to implement it as an input Event device. In our design, Event device method is used.

### 4.2. Realization of HAL

a). The interface of Sensor's HAL

Sensor.h in directory of Hardware/libhardware/include/ hardware/ is the interface of hardware layer of Android sensor system. This is a standard hardware module of Android. In these, constant SENSOR_TYPE_* stand for the type of sensor.

b). Realization of sensor's HAL

The realization of sensor's HAL is to use following structure according the detail specifications of sensor step by step[8].

In sensor module, sensors_module_t is defined as follows:

```
struct sensors_module t {
    struct hw_moduie_t common;
    int (*get_sensors_list) (struct sensors_module_t *
    module,struct sensor_t const ** list);
};
```

Sensor_t is used to define a sensor, as follows:

```
struct sensor_t {
  const char*     name;     /*name of sensor*/
```

```
        const char*     vendor;    /* vendor of sensor*/
        int             version;                 /*version of sensor*/
        int             handle;                  /*handle of sensor*/
        int             type;              /*type of sensor*/
        float           maxRange;    /*max range of sensor*/
        float           resoluation;   /*resolution of sensor*/
        float           power;               /*power of sensor(unit:mA)*/
        void*           reserved [9] ;
      }
```

sensor_vec_t is the structure to present sensor's amount, as follows :

```
    typedef struct {
       union {
            float v[3];            /*3 floats*/
            struct {               /*polar coordinate*/
                 float x;
                 float y;
                 float z ;
                        }
       struct {                    /*polar coordinate*/
                 float azimuth;
                 float pitch;
                 float roll ;
      };
      int8_t status;        /*status information*/
      unit8_t reserved[3];
    }  sensors_vect_t;
```

The realization of Sensor HAL is not very difficult, its main task is using poll call function acquire the sensor_data_t data from bottom layer

sensor_data_t structure is to present the data of sensor. Its structure is :

```
    typedef struct{
      int sensor;                              /*sensor ID*/
      union {
             sensors_vect_t vector;    /*x,y,z  vector*/
             sensors_vect_t orientation;
                                  /*orientation(unit:degree)*/
             sensors_vect_t acceleration;
                                    /*acceleration(unit:m/s2)*/
             sensors_vect_t magnetic;
                                    /*magnetic(unit:uT)*/
             float temperature;          /*temperature(unit:°C)*/
      };
      int64_t  time                 /*time(unit:ns)*/
      uint32_t            resered
    } sensors_data_t;
```

Among them, a union is used to show different type sensor data, and sensor is the ID of sensor.

Structure sensors_control_device_t and sensors_data device_t presents user control device and data device of sensor system, they extends the class of hw_device_t.

The definition of sensors_control_ device_t as follows:

```
    struct sensors_control_device_t {
        struct hw_device t common;
          native_handle_t* (*open_data_source) (struct sensors_control_device_t *dev) ;
      int (*activate) (struct sensors_control_device_t *dev, int handle, intenabled);
```

```
        int (*set_delay) (struct sensors_control_device_t *dev, int32_t ms) ;
     int (*wake) (struct  sensors_control_device_t *dev);
       };
```

Open_data_source function pointer is used to get the handle of sensor device, and then setup data device with it as context, the three pointers of activate, set_delay and wake function are used to implement assistant function.

The definition of sensors data_device_t as follows:

```
        struct sensors_data_device_t {
          struct hw_device_t common;
        int (*data_open) (struct sensors_data_device_t *dev, native_handle_t *nh);
      int (*data_close) (struct  sensors_data_device_t *dev);
    int (*poll) (struct sensors_data_device_t *dev, sensors_data_t *data);
      }
```

The pointers of data_open and data_close function are used to open and close sensor data device, and the process start from native_ handle_t.

A few special points must be concerned in the process of realization of Sensor HAL.

- Maintain of context
- Realization of  data device poll function
- Realization of activate, set_delay and wake of control device
- Support of multiple sensors
- Conclusion

## 5. Application of sensor system

From Figure 2, it is shown that the upper layer contains:

- JNI part of sensors and the Java framework of sensor
- Calling of sensor part in Java framework
- Calling of sensor part in applications

When the previous work is completed, the development of applications using these sensor modules is same with ordinary Java application program. These sensor devices can be used with API of upper layer.

## 6. Conclusion

The microprocessor and peripherals of different platform are different, but the most software of Android system is common (including local framework, virtual machine, Java framework and Java applications).

Java realized the aim that different platform or OS using the same langue. But we must know that all these must support by bottom layer drivers and HAL.The work should be done by some people.

It is shown that in a real system, sensor's implement is not very difficult. Since it is not standard, therefore, it is flexible to implement. It can be implemented with a convinent way.

In a computer system, especially not a mobile phone, not all type of standard type sensor is used. We can replace sensors which are not used in system with other type of sensors or other data collection devices. Using the HAL and the libraries system providing can make it easier to develop and it also can use many API function of upper layer.

## 7. Acknowledgment

Thanks of all people who give me good ideas and suggestions. This paper is supported by a grant of National Science foundation of China (60773114), a grant of teaching and education project of education department of Anhui Province, China (2008jyxm277) , financed by the 211 Project  of Anhui University(The theory and applications of security algorithm).

## 8. References

[1] Z. L. She, Y. X. Chen, and M. J. Zheng, "Develop examples on Google Android SDK," *Post and telecom Press*. 2010.

[2] X. H. Wang, G. Y. Zhang, and J. Shen, "Application program development on Android," *Tsinghua university press*, 2010.

[3] R. Meier, "Android 2 Professional Android application development (Ver.2)," *Tsinghua university press*, 2010.

[4] Electronic Publication: http://code.google.com/android/what-is-android.html

[5] F. S. Yang, "The secrets of Android application development," *China Machine Press*. 2010.

[6] W. F. Ableson, C. Colins, and R. Sen, "Unlocking Android a developer's guideer," *Post and Telecom Press*. 2010.

[7] C. Han, Q. Liang, "Deep level development of Android system -migration and debug," *Publish house of electronics industry*.2011.

[8] Electronic Publication: http://code.google.com/android/.

[9] S. Hashimi and S. Komatineni, "Pro Android 2 [M],"*Post and telecom Press*. 2010.

[10] R. Rogers, B. Meike, and Z. Mednieks, "Android application development," *Post and telecom Press*. 2010.

[11] Y. F. Wu and Y. N. Suo, "Kernel technology and samples description of Android," *Publish house of electronics industry*. 2010.