X-Man答辩

部门：　BSP
　X-Man：　赵建强
Mentor：　丁宇辰
Date:　2019.5.7

一.自我介绍

学校：安徽工业大学　本科

时间：2015.09--2019.07

专业：电气工程及其自动化

爱好：乒乓球，写散文，读杂志，看电影

二.X-Man期间的学习和收获

<一>在Linux下工作的基础知识

1.办公环境的配置：

Chorme,Source Insight,Foxit PDF Editor，Wps,Mail

等常用办公环境配置成自己比较熟悉

2.linux的基本操作命令：ls、cd、cat、echo...

3.服务器上代码的下载和提交：

git下载代码到本地，Gerrit提交代码到服务器

4.服务器相关的操作：

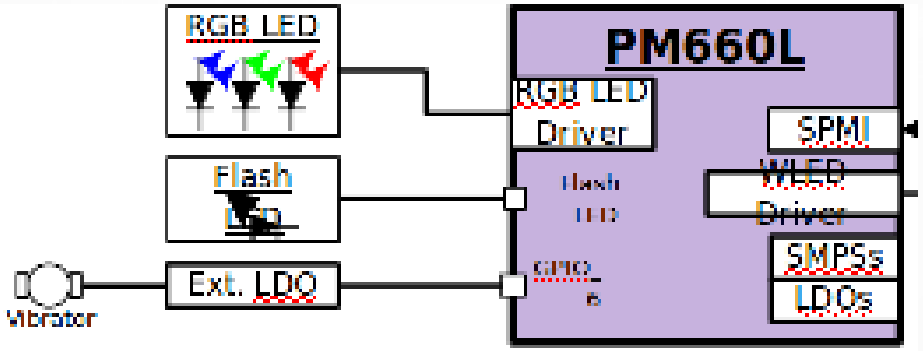
服务器的登录：ssh命令

//通过服务器修改、编译代码
服务器挂载到本地：sshfs命令
//可以在本地拷贝一些传在服务器上资料
5.大型项目代码阅读工具:
通过OpenGrok方便的阅读和快速搜索Android源码
6.VNC的配置使用：申请使用创达内网
7.使用测试工具：使用MD8475A工具（里面有windows系统），
测试GSM,WCDMA,LTE DDD
//测试2G,3G,4G网络
8.usb串口的配置

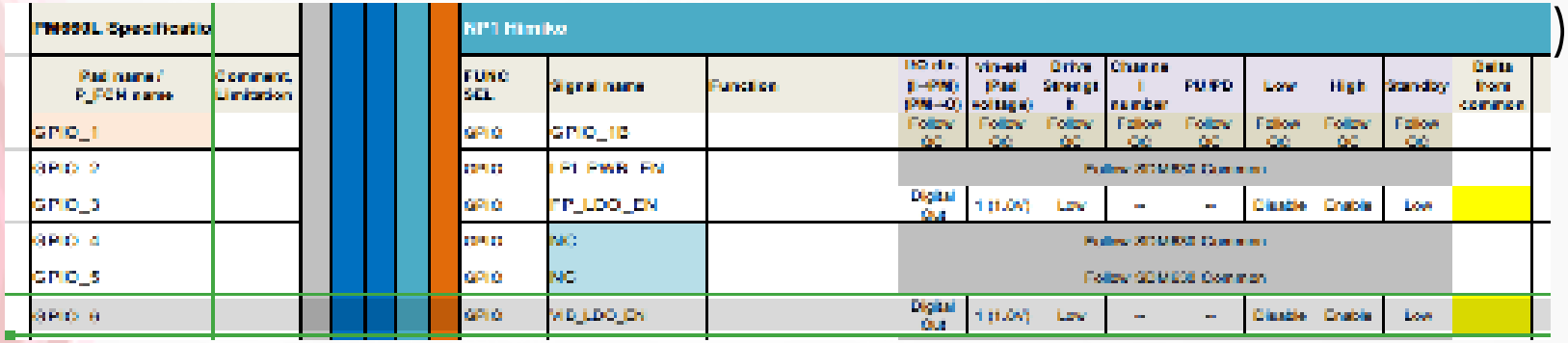
<二>基于索尼项目NP1（SDM630）的马达整体分析

1.马达的工作原理：是一个小电机，给个高电平就震动，给个低电平就关闭。

2.马达的硬件模块：



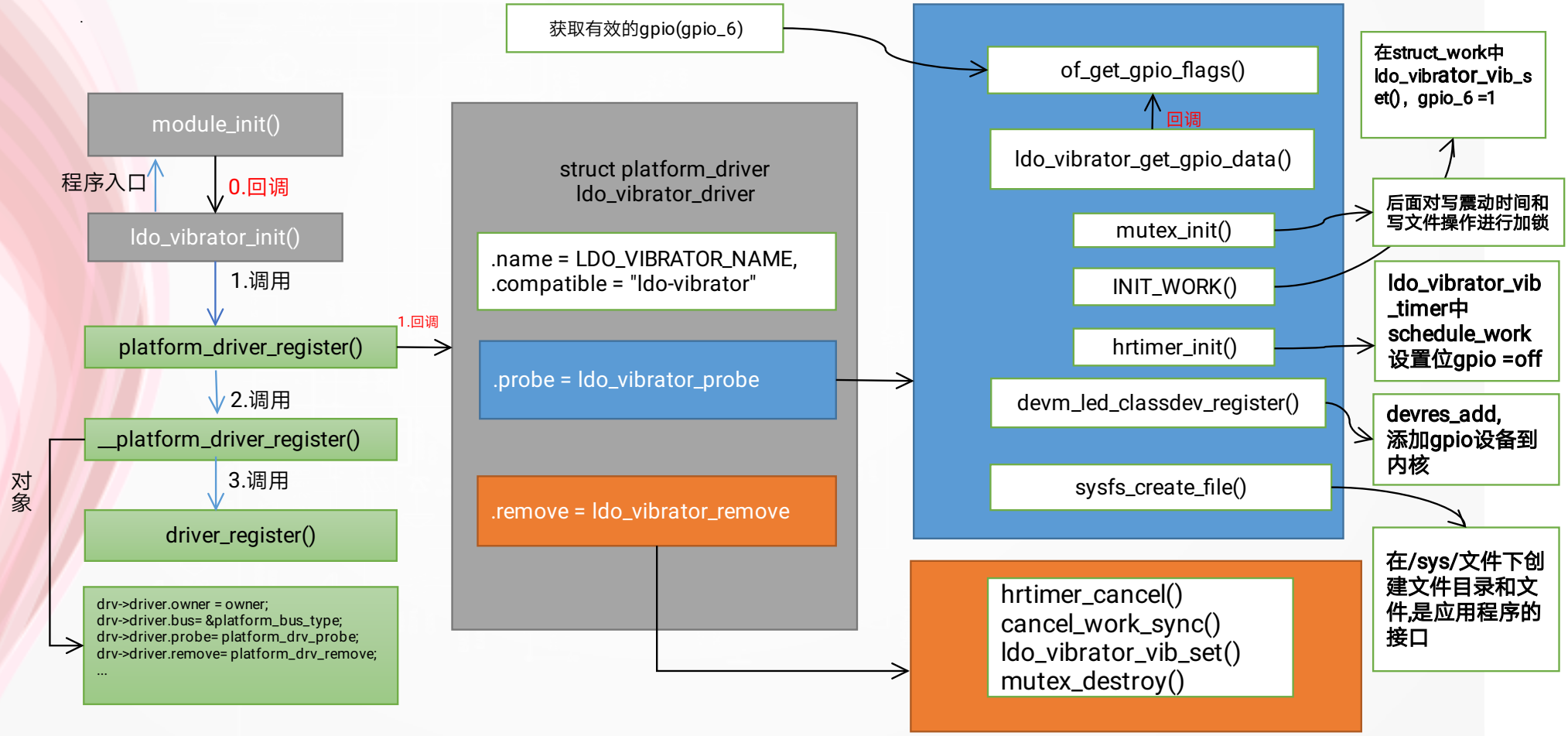
3.马达的GPIO



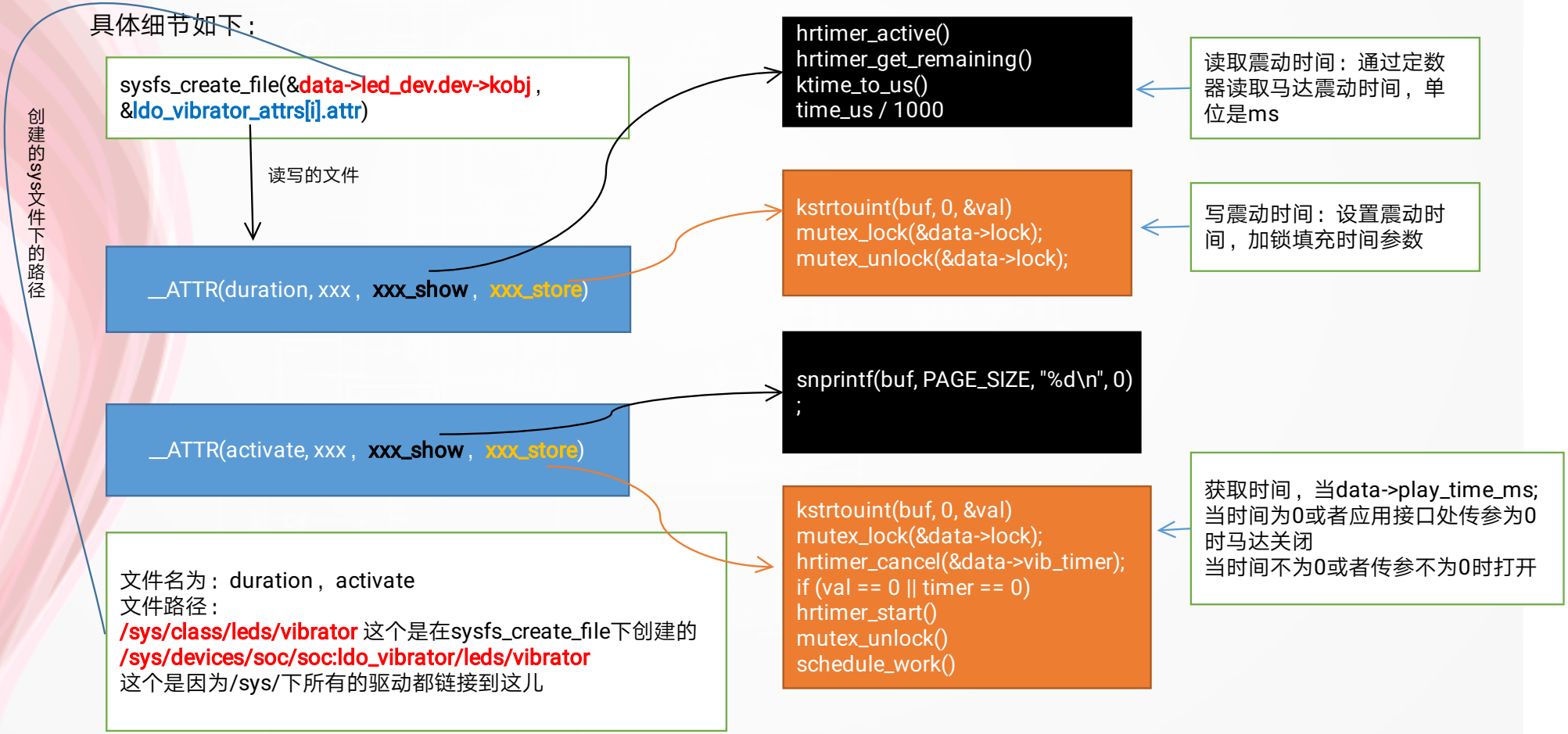
PM660L Specificatio			MP1 Hinako												
Pac name/ P_FCH name	Constraint Limitation		FUNC SEL	Signal name	Function	IO dir (I->PM) (PM->O)	vin-set (Pad voltage)	Drive Strength	Channel number	PU/PD	Low	High	Standby	Data from common
GPIO_1			GPIO	GPIO_1		Follow DC	Follow DC	Follow DC	Follow DC	Follow DC	Follow DC	Follow DC	Follow DC	
GPIO_2			FUNC	LPI_PWR_EN		Follow SDM630 Common								
GPIO_3			GPIO	TP_LDO_EN		Digital Out	1.8 (V)	Low	–	–	Disable	Disable	Low	
GPIO_4			FUNC	NC		Follow SDM630 Common								
GPIO_5			GPIO	NC		Follow SDM630 Common								
GPIO_6			GPIO	VIB_LDO_EN		Digital Out	1.8 (V)	Low	–	–	Disable	Disable	Low	

```
ldo_vibrator {
        compatible = "ldo-vibrator";
        gpios = <&pm660l_gpios 6 1>;
};
```

4.ldo_vibrator设备驱动的软件流程



5.ldo_vibrator的读写文件的处理



6.ldo_vibrator的HAL层

在kernel的驱动中，我们已经将马达的驱动注册到sys文件系统中/sys/class/leds/vibrator/。在vibrator.c中，通过读写“ctivate、duration、state”设备文件来实现对马达的操作，HAL层是对这些设备节点的封装，通过open、close、read、write来封装统一的对外接口：

HAL层的代码路径：ap/vendor/semc/hardware/vibrator/vibrator.c

注册到sys文件系统的路径：

```
static const char THE_DEVICE_ACTIVATE[]     = "/sys/class/leds/vibrator/activate";
static const char THE_DEVICE_DURATION[]     = "/sys/class/leds/vibrator/duration";
static const char THE_DEVICE_STATE[]        = "/sys/class/leds/vibrator/state";
```

对外的接口API：

```
static int vibra_on(vibrator_device_t* vibradev __unused, unsigned int timeout_ms);
static int vibra_off(vibrator_device_t* vibradev __unused);
static int vibra_supportsamplitudecontrol(vibrator_device_t* vibradev __unused);
static int vibra_setamplitude(vibrator_device_t* vibradev __unused, unsigned int amplitude);
```

7.ldo_vibrator的JNI层

JNI(Java Native Interface),中文是“Java本地接口”。是Java的一种技术，保证本地代码(C/C++代码)能在Java虚拟机下工作。具体的代码路径：

ap/frameworks/base/services/core/jni/com_android_server_VibratorService.cpp

方法列表：

```
static const JNINativeMethod method_table[] = {
     { "vibratorExists", "()Z", (void*)vibratorExists },
     { "vibratorInit", "()V", (void*)vibratorInit },
     { "vibratorOn", "(J)V", (void*)vibratorOn },
     { "vibratorOff", "()V", (void*)vibratorOff },
     { "vibratorSupportsAmplitudeControl", "()Z", (void*)vibratorSupportsAmplitudeControl},
     { "vibratorSetAmplitude", "(I)V", (void*)vibratorSetAmplitude},
     { "vibratorPerformEffect", "(JJ)J", (void*)vibratorPerformEffect}
};
```

注册及注册服务：

```
int register_android_server_VibratorService(JNIEnv *env)
{
    return jniRegisterNativeMethods(env, "com/android/server/VibratorService",
            method_table, NELEM(method_table));
}
```

马达的JNI与HAL层的关联方式：

com_android_server_VibratorService.cpp调用HAL层vibrator.c文件：把.c文件封装成.so库，然后在.cpp文件中调用.so库，就可以调用vibrator.c文件的接口了。

具体的如下：

在ap/vendor/semc/hardware/vibrator/Android.mk中把vibrator.c添加到

LOCAL_SRC_FILES变量中：

```
LOCAL_SRC_FILES := vibrator.c \
                   haptic_effect_table.c
```

在编译Android系统时会生成相应的.so库，vibrator.c在libradware_ legacy.so库中。这样就会在com_android_server_VibratorService.cpp对应的Android.mk中，导入

libradware_ legacy.so库。

8.ldo_vibrator的Framework层

应用层操作马达，是通过操作Framework层的接口进行操作的。跟马达主要相关的文件如下：

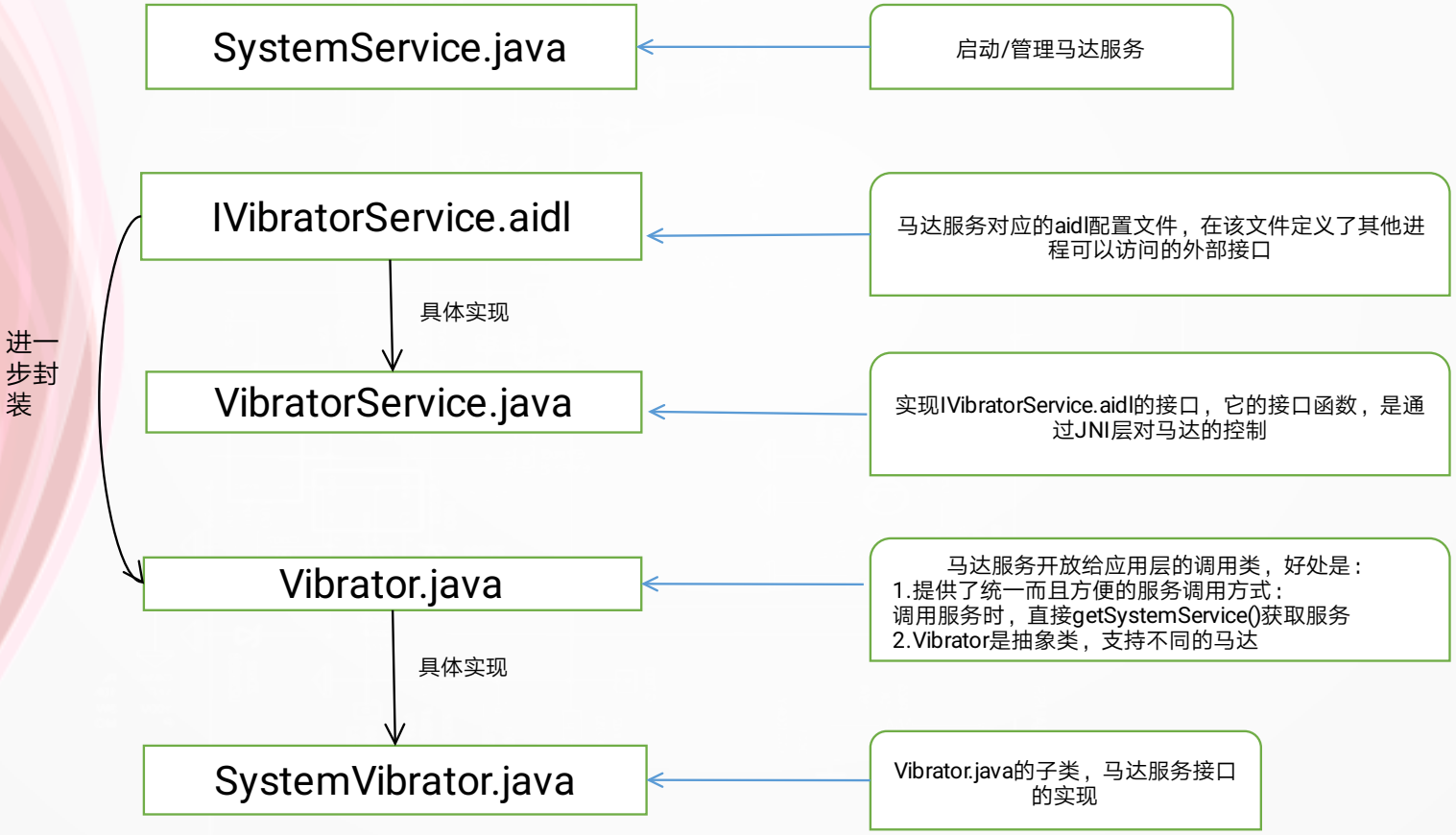
ap/frameworks/base/core/java/android/os/

IVibratorService.aidl	H A D	25-Jan-2019	925
Vibrator.java	H A D	25-Jan-2019	9.9 KiB
SystemVibrator.java	H A D	25-Jan-2019	2.0 KiB

ap/frameworks/base/services/core/java/com/android/server/

SystemService.java	H A D	25-Jan-2019	9.6 KiB
VibratorService.java	H A D	25-Jan-2019	52.2 KiB

具体的文件分析


SystemService.java
启动/管理马达服务
IVibratorService.aidl
马达服务对应的aidl配置文件，在该文件定义了其他进程可以访问的外部接口
具体实现
进一步封装
VibratorService.java
实现IVibratorService.aidl的接口，它的接口函数，是通过JNI层对马达的控制
Vibrator.java
马达服务开放给应用层的调用类，好处是：
1.提供了统一而且方便的服务调用方式：
调用服务时，直接getSystemService()获取服务
2.Vibrator是抽象类，支持不同的马达
具体实现
SystemVibrator.java
Vibrator.java的子类，马达服务接口的实现

对public boolean hasVibrator()的追踪

IVibratorService.java 对外api接口：

```
interface IVibratorService
{
    boolean hasVibrator();
    boolean hasAmplitudeControl();
    void vibrate(int uid, String opPkg, in VibrationEffect effect, int usageHint, IBinder token);
    void cancelVibrate(IBinder token);
}
```

在IVibratorService.java中的调用：

boolean hasVibrator()--->boolean doVibratorExists()---->boolean vibratorExists()

--->static jboolean vibratorExists(JNIEnv* /* env */, jobject /* clazz */)

(在jni/com_android_server_VibratorService.cpp)

在SystemVibrator.java中对 boolean hasVibrator() 作了进一步封装public boolean hasVibrator()

9.在应用中的使用

代码路径：ap/vendor/semc/packages/apps/service-menu/src/com/sonyericsson/android/servicemenu/servicetests/VibratorTest.java

startButton.setOnClickListener()　开始按钮后

startVibrator(); //开始马达

主要获取 mVibrator.vibrate(vibratorTime);填充马达的时间

private Vibrator mVibrator = null;

class Vibrator 里面的方法是Framework封装的函数：

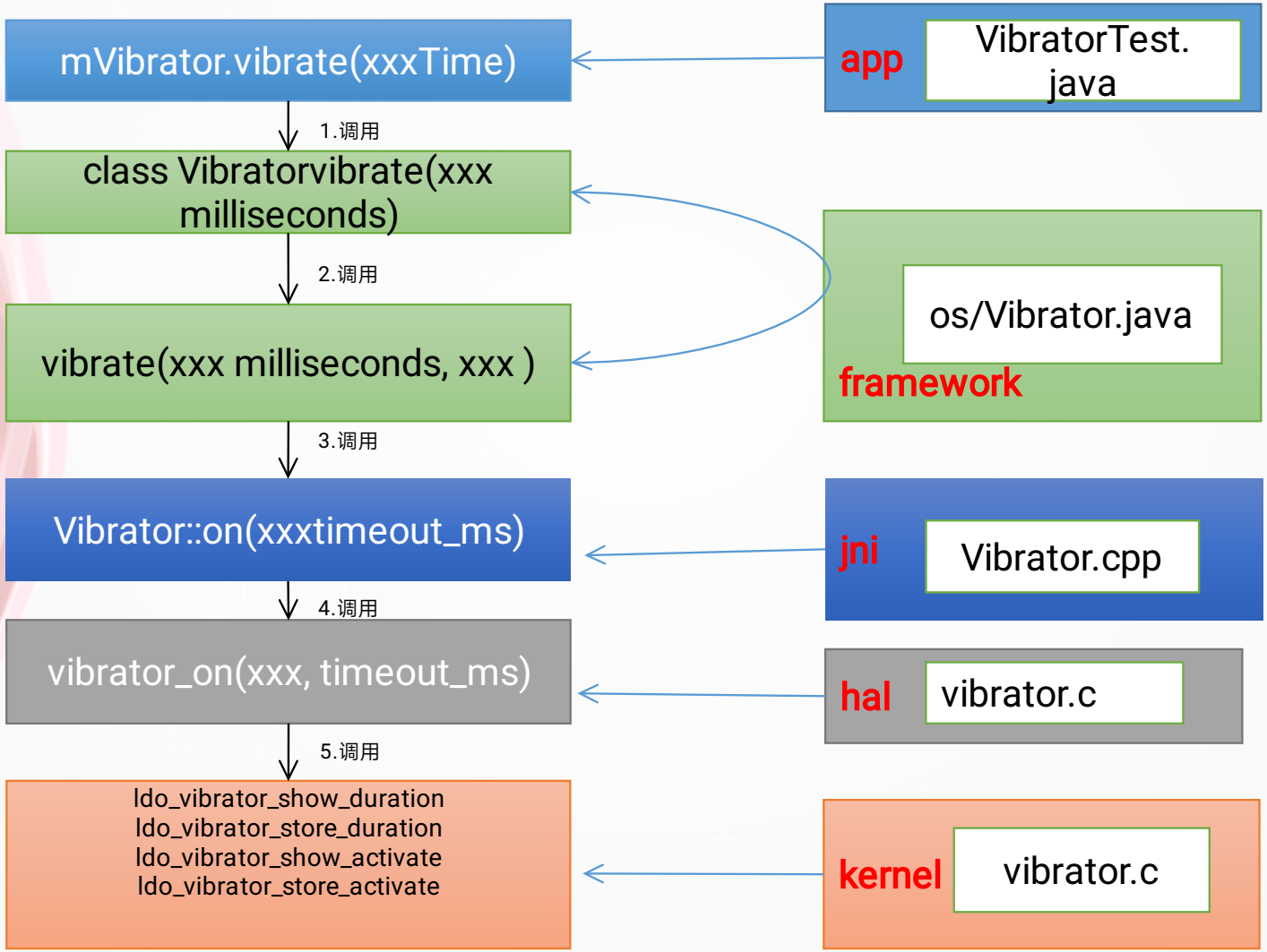
```
public abstract boolean hasVibrator();

/**
 * Check whether the vibrator has amplitude control.
 *
 * @return True if the hardware can control the amplitude of the
 */
public abstract boolean hasAmplitudeControl();
```

```
@RequiresPermission(android.Manifest.permission.VIBRATE)
public abstract void vibrate(int uid, String opPkg,
        VibrationEffect vibe, AudioAttributes attributes);

/**
 * Turn the vibrator off.
 */
@RequiresPermission(android.Manifest.permission.VIBRATE)
public abstract void cancel();
```

各层代码的调用过程


mVibrator.vibrate(xxxTime)
1.调用
class Vibratorvibrate(xxx milliseconds)
2.调用
vibrate(xxx milliseconds, xxx)
3.调用
Vibrator::on(xxxtimeout_ms)
4.调用
vibrator_on(xxx, timeout_ms)
5.调用
ldo_vibrator_show_duration
ldo_vibrator_store_duration
ldo_vibrator_show_activate
ldo_vibrator_store_activate
app
VibratorTest.java
os/Vibrator.java
framework
jni
Vibrator.cpp
hal
vibrator.c
kernel
vibrator.c

10.编译驱动

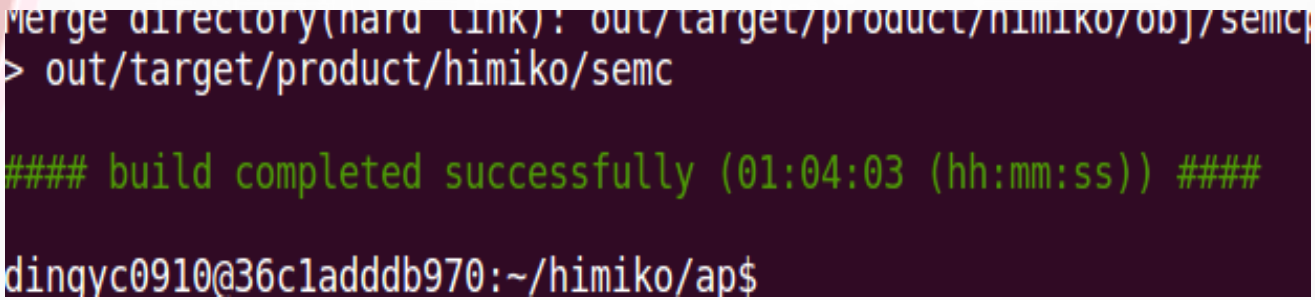
ldo_vibrator的驱动放在ap侧,选择整编ap侧代码：

source build/envsetup.sh

lunch himiko-userdebug

make fullbuild -j12

在终端的结果如下：编译成功时



11.刷机：

在ap/out/target/product/himiko/semc 目录下,生成了两个红色的.zip文件 ， 是我们编译的结构，需要把这两个文件烧到手机里面

```
dingyc0910@36c1adddb970:~/himiko/ap/out/target/product/himiko/semc$ ls
APP-SW_HIMIKO_GLOBAL-A1_USERDEBUG_PRIVATE_1319-3767_X-FLASH-ALL-9B8D.zip  oem_other
FS_HIMIKO_GENERIC_USERDEBUG_PRIVATE_X-FLASH-ALL-9B8D.zip                 ota
adsp_proc                                                                recoveryroot
bluetooth                                                                root
data                                                                     system
modem                                                                    system_other
oem                                                                      vendor
```

12.调试抓alog

1.dmesg：（没有bootload阶段的信息）静态的打印内核信息

代码出：（使用两种查看）

```
 dev_err(data->dev,
                  "%s: 0000000000000000 before activate\n", __func__);
printk(KERN_INFO"00000000000000 printk befoe activate\n");
    dev_err(data->dev,
                    "%s: 0000000000000000 after activate\n", __func__);

    printk(KERN_INFO"00000000 printk  befoe activate\n");
```

写文件前：dmesg 打印出的信息：

```
[  265.313107] ldo_vibrator soc:ldo_vibrator: ldo_vibrator_store_activate: 0000000000000000 before activate
[  265.313133] 00000000000000 printk befoe activate
[  265.313177] ldo_vibrator soc:ldo_vibrator: ldo_vibrator_store_activate: 0000000000000000 after activate
[  265.313185] 00000000 printk  befoe activate
```

写文件后：

```
385.805295] ldo_vibrator soc:ldo_vibrator: ldo_vibrator_store_activate: 0000000000000000 before activate
385.805324] 00000000000000 printk befoe activate
385.805396] ldo_vibrator soc:ldo_vibrator: ldo_vibrator_store_activate: 0000000000000000 after activate
```

写文件操作

```
Himiko:/sys/class/leds/vibrator # ls
activate    device    max_brightness state      trigger
brightness duration power            subsystem uevent
Himiko:/sys/class/leds/vibrator # echo 4000 > duration
Himiko:/sys/class/leds/vibrator # echo 1 > activate
Himiko:/sys/class/leds/vibrator # _
```

2.动态打印内核信息： cat /proc/kmsg

```
Himiko:/ # cat  /proc/kmsg
<3>[ 4298.536142] ldo_vibrator soc:ldo_vibrator: ldo_vibrator_store_activate: 0000000000000000 before activate
<6>[ 4298.536169] 00000000000000 printk befoe activate
<3>[ 4298.536218] ldo_vibrator soc:ldo_vibrator: ldo_vibrator_store_activate: 0000000000000000 after activate
<6>[ 4298.536226] 00000000 printk  befoe activate
<3>[ 4298.544243] ldo_vibrator soc:ldo_vibrator: ldo_vibrator_store_activate: 0000000000000000 before activate
<6>[ 4298.544277] 00000000000000 printk befoe activate
<3>[ 4298.544348] ldo_vibrator soc:ldo_vibrator: ldo_vibrator_store_activate: 0000000000000000 after activate
<6>[ 4298.544362] 00000000 printk  befoe activate
<3>[ 4298.601011] afe_get_cal_hw_delay: Unable to find delay for sample rate 8000
Himiko:/ # _
```

<三>sensor部分的学习

1.阅读高通的sensor部分文档：

80-nm328-44_e_presentation__adsp.bf.2.6_sensors_overview

//传感器在整个Android中的硬件架构和软件架构

80-nh058-1_j_qualcomm_snapdragon_sensors_core_(ssc)_features_for_linux_android

//关于ssc的一些特征的列表说明

80-nb925-1_r_snapdragon_sensors_core_compatible_sensor_drivers_list

// 关于传感器一些兼容性的表格

80-n7635- snapdragon_sensors_core_(ssc)_new_sensor_driver_integration_guide_for_linux_android //移植新的传感器驱动程序的指南和一般性说明

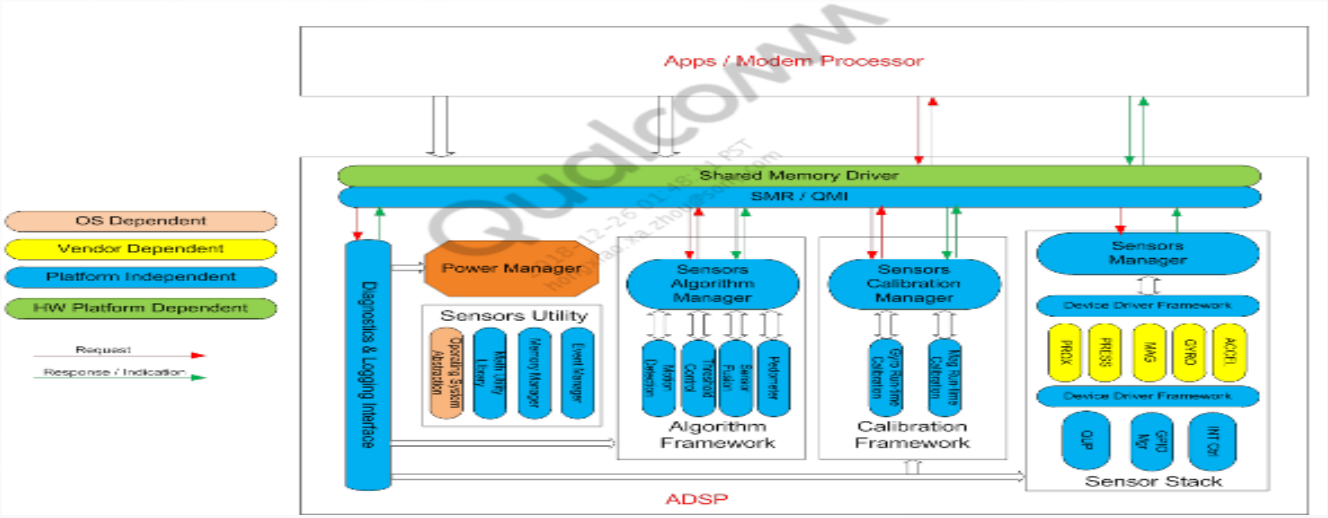
80-na811- 2_b_presentation__adding_a_custom_sensors_algorithm_on_adsp_using_sensors_algorithm_manager_(sam)_2.0 //有关传感器的数据.命令的改变在.idl文件中

2.文档学习的收获：

1>了解了BSP开发者主要做的是：在Android中添加一个新的device后，分析各种bug

2>.sensor的实现放在adsp下，如下图所示的框架：底层的sensor集成ADSP中，sensor manager 将数据进行打包，然后共享内存中，由上层共享内存解析数据包，最后拿到数据给上层作各种操作。

/vendor/firmware_mnt/image :



3.bp侧的单独编译

编译：

source build/envsetup.sh

lunch ganges-userdebug

make adsp -j8

通过adb工具烧写到手机：

adb shell + 回车键

mount -o rw,remount /vendor/firmware_mnt   //以读写的方式挂载文件

rm /vendor/firmware_mnt/image/adsp.*   //删除上次的所有adsp文件

push . /vendor/firmware_mnt/image/   //push 当前服务器上的adsp*.到手机目录下

rm /persist/sensors/sns.reg   //删掉上次的配置信息

adb reboot   //重启手机

<三>技术之外的收获

1.一般一个项目是5.6个人来一起完成某一方面的功能，项目中如果有自己不能十分确定的，及时跟大家确定，不太确定的结果可能导致整个项目的due date推迟；

2.跟别人讨论问题的时候，直接在电脑上清清楚楚的显示，不然浪费别人的时间；

3.有些时候一些功能测试搞不定，还需要主动学一点自动化测试方面的；

4.思路不太清楚的时候，及时问别人，不要自己瞎琢磨。

5.及时处理邮箱，养成利用邮箱办公的习惯

6.给客户的邮箱内容精简，能一句话说明白的，不要写太多；同时不能有歧义，让客户去判断，在哪种版本，有哪种现象，要很具体的写。

三.计划和期望

1.下阶段开始sensor部分的学习，希望能解决一些简单的bug
2.继续kernel部分基础知识的学习，能快速理清代码框架，找清楚关键点
3.形成一个推敲bug比较高效的方法，积累项目经验

四.需要与帮助

1.推荐一些基础书籍

2.推荐一些看内核的方法或者资料

ThunderSoft

Thanks