智能电视远程传感器控制的实现方法①

谷诗萌, 胡启斌, 张 顺

(成都理工大学 信息科学与技术学院, 成都 610059)

摘 要: 主要介绍了一种在电视上加载虚拟传感器设备驱动的方法, 将具有传感器功能的手机、PAD 等智能终端作为电视传感器输入终端, 在智能电视上实现利用传感器控制应用和游戏的解决方案. 具体实现方法为在智能电视上加载各种传感器如重力加速度(G-sensors)、线性加速度、陀螺仪等的虚拟设备驱动, 并提供上层传感器数据注入方法和应用层传感器数据获取通用接口; 智能传感器输入终端通过 wifi 与智能电视建立网络连接, 获取智能终端的传感器实际数据, 并将数据发送到电视端; 电视端接收到数据并注入到指定传感器设备, 最终由系统上报提供到应用层, 使得各类普通的传感器类应用和游戏在电视上展现. 可解决现在智能电视上应用和游戏单一的局面, 实现应用和游戏控制与画面相分开, 以获得更好的视觉效果并很好的与他人分享.

关键词: 传感器; 智能电视; 驱动; 数据; 接口

Implementation of Smart TV Remote Sensor Control

GU Shi-Meng, HU Qi-Bin, ZHANG Shun

(Chengdu University of Technology, Chengdu 610059, China)

Abstract: This paper introduces a method of loading virtual sensor device drivers on a smart TV. With sensor-enabled devices as the input terminals of a smart TV, such as mobile phones/PADs or other smart terminals, a solution of sensor control for applications and games on a smart TV could be reached. The solution is to load various kinds of virtual sensor device drivers on the smart TV, including gravitational (G-sensors), linear acceleration, gyroscopes and other virtual device drivers, and to provide the data injection method of the upper layer sensors and the data of application layer sensors, so that the common interface was acquired. A network can be established via WIFI between the input terminal of the smart sensors and the smart television, then the data would be acquired and be transferred to the television. Once the data was got, it would be injected to certain devices, and to the application layer though operating system, making various sensor applications and games display on the TV. In this way, the diversity of applications and games on the smart television can be improved, and the frames from the control of games and applications can be separated. Thus a better visual experience and share would be achieved.

Key words: sensor; smart TV; driver; data; interface

① 收稿时间:2014-12-01;收到修改稿时间:2015-02-02

1 绪论

电视并不属于移动终端, 所以手机或 PAD 上的独有的传感器功能无法实现, 导致很多需要传感器来控制的 Android 游戏无法在电视得到很好的体验效果. 介于此功能的需求, 采用虚拟传感器设备可以很好的解决. 此方案主要思路是在 linux 内核层添加一个虚拟的 sensor 设备, 由上层的 Android 的 HAL 层进行识别. 在移动终端采集传感器数据通过 wifi 网络传送到电视机端, 通过 JNI 层将接收到的数据注入创建的虚拟传感器设备. 这便使得电视的 Android 系统具有传感器功能, 弥补不可移动的缺陷.

2 软件系统基本结构设计

智能电视端虚拟传感器系统包括虚拟传感器设备

驱动模块、传感器数据注入模块、传感器应用层数据提供模块和智能电视与传感器输入终端通信模块. 所述的手机PAD智能传感器输入终端装置将具有各类传感器设备的智能手机和PAD作为电视虚拟传感器设备的数据输入装置, 通过电视端的数据注入模块写入虚拟传感器设备, 并最终在电视应用层获取到传感器数据, 实现应用和游戏的控制. 虚拟传感器设备驱动模块包括线性加速度传感器设备驱动、G-sensor 传感器设备驱动、陀螺仪传感器设备驱动等, 虚拟底层驱动采用 Input 子系统处理输入事务, 通过 Input 输入子系统提供的接口注册到内核, 利用子系统提供的功能来与用户空间交互, 在这里主要是用此虚拟设备来接受手机或 PAD 端的传感器数据. 传感器应用层数据提供模块是指编写传感器系统的硬件抽象层(HAL)代码, 对传感器设备驱动进行封装, 向上层提供接口, 由智能系统 sensorservice 程序自动加载传感器并通过事件响应模型读取传感器数据, 作为应用层获取传感器数据的通用接口. 智能电视与传感器输入终端通信模块指接收来自智能输入终端的查找命令并响应, 接收来自智能输入终端的连接命令应答并建立连接, 接收来自智能输入终端的传感器数据注入相应的传感器设备.

为手机 PAD 智能传感器输入终端装置, 采用下述技术方案来实现: 将具有各类传感器设备的智能手机和 PAD 作为电视虚拟传感器设备的数据输入装置, 智能输入终端对同一网络的具有传感器设备的电视进行查找, 查找到设备后发起连接请求, 收到电视应答后建立连接; 用户可选择是否将某一类型的传感器数据注入到电视, 根据用户选择采集对应类型的传感器数据通过连接将传感器类型和传感器数据发送到已连接电视设备.

3 linux底层驱动部分设计

linux 中设备驱动在 2.6 以后的版本中是可以作为 ko 文件类型动态加载到系统中的, 这样大大提高了开发进度, 不用因为每次修改驱动而去重新编译内核, 于是便可以编写一个虚拟的 sensor 设备驱动. 因为实际没有硬件的支持, 只需要让这个设备有读取数据的功能即可. 系统中 sensor 的数据是通过 linux input 子系统完成数据上报功能, 因此我们必须将设备注册为一个 input 的设备[1-4]. 下面将详细阐述 input 设备编写流程:

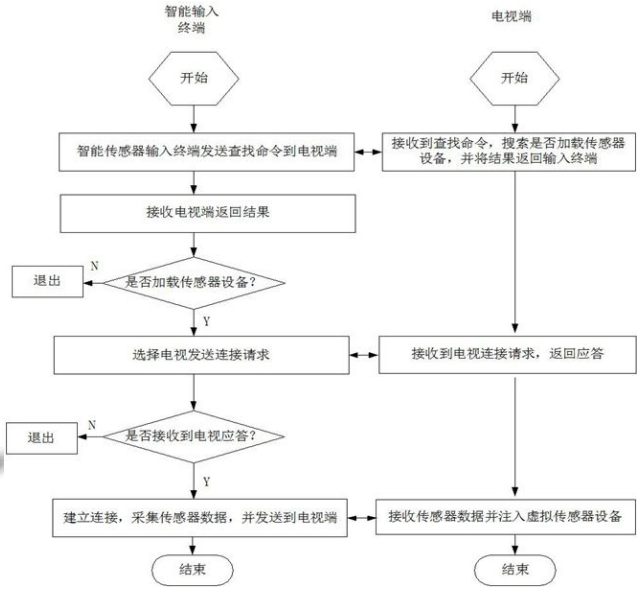

图 1 电视传感器系统的系统框架图

① 分配, 注册, 注销 input 设备:

```
struct input_dev *input_allocate_device(void)
int input_register_device(struct input_dev *dev)
void input_unregister_device(struct input_dev *dev)
```

② 设置 input 设备支持的事件类型、事件码、事件值的范围、input_id 等信息.

```
IAsensor_dev->evbit[0] =BIT_MASK(EV_ABS) ;
IAsensor_dev->absbit[0]=BIT_MASK(ABS_X)
|BIT_MASK(ABS_Y) | BIT_MASK(ABS_Z);
input_set_abs_params(IAsensor_dev, ABS_X, 0, 0xfff, 0, 0);
input_set_abs_params(IAsensor_dev, ABS_Y, 0, 0xfff, 0, 0);
input_set_abs_params(IAsensor_dev, ABS_Z, 0, 0xfff, 0, 0);
__set_bit(EV_ABS, IAsensor_dev->evbit);
__set_bit(ABS_X,IAsensor_dev->absbit);
__set_bit(ABS_Y,IAsensor_dev->absbit);
__set_bit(ABS_Z,IAsensor_dev->absbit);
```

include/linux/input.h 中定义了支持的类型:

```
#define EV_SYN 0x00
#define EV_KEY 0x01
#define EV_REL 0x02
#define EV_ABS 0x03
```

一个设备可以支持一个或多个事件类型. 每个事

件类型下面还需要设置具体的触发事件码. 比如: EV_KEY 事件, 需要定义其支持哪些按键事件码.

如果有需要, 设置 input 设备的打开、关闭、写入数据时的处理方法:

IAsensor_Open

IAsensor_Close

IAsensor_Init

该方案中是有 JNI 对设备写入数据, 所以并没有特殊的写入数据的具体方法. 在发生输入事件时, 向子系统报告事件, 对于不同的事件类型有着不同的事件函数:

void input_report_key(struct input_dev *dev, unsigned int code, int value)

void input_report_rel(struct input_dev *dev, unsigned int code, int value)

void input_report_abs(struct input_dev *dev, unsigned int code, int value)

void input_event(struct input_dev *dev, unsigned int type, unsigned int code, int value)

实现上报数据的函数为:

input_report_abs(IAsensor_dev,ABS_X,event.x);

input_report_abs(IAsensor_dev,ABS_Y,event.y);

input_report_abs(IAsensor_dev,ABS_Z,event.z);

input_sync(IAsensor_dev);

(为了同步 X, Y, Z, 3 个值, 视为一个时间点所采集到的传感器数值)

4 Android的HAL软件设计

Android 的 HAL 层就是硬件抽象层, 也就是对 Linux 内核驱动程序的封装并向上提供接口, 屏蔽底层的细节.

主要有 3 个结构体构成(存在与 Android 源代码 /hardware/libhardware/ include/hardware/hardware.h 中)

① hw_module_t 结构体用于定义硬件模块;

② hw_device_t 表示硬件设备, 其中存储硬件设备的公共属性和操作方;

③ hw_modules_methods_t 定义操作设备的方法[5].

传感器的 HAL 源代码中包括以下几个文件:

sensors.c: 主要实现 sensors_module_t 和 sensor_t 结构体, 定义了 sensors 模块的主要功能.

SensorBace.cpp: 为各 sensors 实现的基类, openDataFd 会返回 sensors EVENT 设备文件的句柄.

InputEventReader.cpp: 文件的主要功能会读取 sensors 的输入事件.

nusensors.cpp: 为 sensors 的控制文件, 实现 sensors_poll_device_t 结构体, 在 init_sensors_control 函数中写义了 sensors_poll_context_t 类的对像, 在 sensors_poll_context_t 类的构造函数中, 定义了具体 sensors 类的对像, init_sensors_control 函数是在 sensors 模块被打开时调用的.

AccelerometerSensor.cpp: 为具体重力加速度传感器的实现.

以上是定制的 sensor 需要修改的文件. 在 nusensors.cpp 与 sensors.c 中加入新添加的传感器名称以及一些初始化参数. 主要是在 AccelerometerSensor.cpp 编写需要上传的数据方式即可. 之后编写 Android.mk, 使得将其编译为动态链接库的形式, 由系统自动去加载.

```
void AccelSensor::processEvent(int code, int value)
{
    switch (code) {
    case EVENT_TYPE_ACCEL_X:
    mPendingEvent.acceleration.x = value*0.00001f ;
    break;
    ...
  }
}
```

可以根据不同的需求更改参数, 使得将由 linux 驱动上报的数据处理的符合使用要求[6-8].

特别注意的是, HAL 层需要在开机时就去遍历 /dev 下的设备, 所以开机必须自动加载 sensor 驱动, 不然系统是无法检测到 sensor 设备的, 替换完 HAL 的 so 库之后需要重启系统.

5 JNI注入数据部分

虚拟传感器无法从硬件采集数据, 只是在 JAVA 层通过 wifi 网络来获得数据. 为了让系统将这些数据识别为传感器数据就必须将这些接收到的数据注入虚拟传感器设备中. JNI 分为 2 个部分.

① 根据不同数据类型打开不同的设备:

```
virtualACCSensorDev = open("/dev/Asensor",O_RDWR);
```

通过 wirte 函数直接将接受到了的数据写入刚才打开的设备.

```
nwrite = write(virtualACCSensorDev, buf, sizeof(buf));
```

② 对应 JAVA 层的 JNI 方法则是:

```
JNICALL Java_com_skyworth_device_SRTSensor_init()
JNIEXPORT jboolean JNICALL Java_com_skyworth_device_SRTSensor_write
(JNIEnv *env, jobject obj, jint sen, jint acc, jfloatArray dataArray)
JNIEXPORT void JNICALL Java_com_skyworth_device_SRTSensor_cleanup
(JNIEnv *, jobject)
```

当数据注入到我们之前注册好的虚拟设备中之后就会通过 input 子系统上报到 HAL 层接受. 这样系统就会认为接收到实际的传感器数据了. 系统的结构图如图 2 所示.

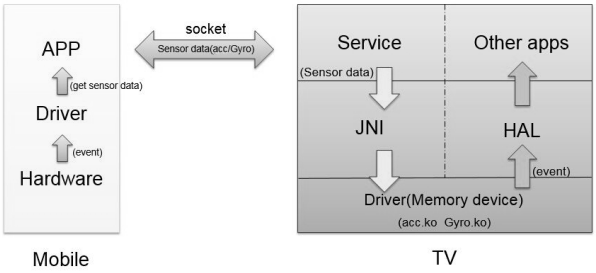

图 2 电视传感器系统的系统结构图

6 结论

电视上加载虚拟传感器设备驱动, 将具有传感器设备的手机或PAD终端的传感器数据通过网络连接发送到电视并注入其虚拟传感器设备, 实现智能电视上传感器类应用和游戏的使用. 本文设计的底层传感器设备采用内核标准驱动, 不是专用设备, 任何传感器类游戏都能应用在电视上, 任何智能设备都可以作为传感器数据输入设备. 在此技术上, 普通的传感器类应用和游戏无须额外的开发和控制器, 即可完成在电视上的简单操作和控制, 而电视的视觉效果也远远加优越于其他游戏设备. 在体验的同时还可以与亲友一起分享, 提高电视的娱乐性, 加速电视成为家庭娱乐中心.

参考文献

1 尹慧,李允俊.嵌入式 Linux 下字符型设备驱动程序的开发.延边教育学院学报,2008,22(1):26–31.
2 钱晨,徐荣华,王钦若.基于 Linux 操作系统的设备驱动程序开发.微计算机信息,2004,20(9):131–133.
3 王策,张连芳,董淼,赵宇,郑武.基于 Linux 的嵌入式系统开发.计算机应用,2002,22(7):54–56.
4 董志国,李式巨.嵌入式 Linux 设备驱动程序开发.计算机工程与设计,2006,27(20):37–40.
5 李永吉.基于 Android 系统平台实现智能电视模块化设计与研究.济南:山东大学,2013.
6 兰晓红.嵌入式 Linux 中断设备驱动程序设计.计算机应用研究,2003,(5):96–98.
7 路广,张伯明,孙宏斌.嵌入式实时 Linux 及其在电网自动化系统中的应用.电力系统自动化,2002,(7):62–65.
8 王粉花.基于 Linux 字符设备驱动程序的设计与实现.计算机工程,2006,32(23):278–280.