文章编号:1671-4598(2016)06-0098-04 DOI:10.16526/j.cnki.11—4762/tp.2016.06.027 中图分类号:TP277 文献标识码:A

基于 Android 的综合航电数据监控平台的研制

王海斌, 钱 伟, 丁发军
(中国民航飞行学院 飞机修理厂, 四川 广汉 618307)

摘要: 分析了通航领域中综合航电系统数据传输的现状, 讨论了针对航电系统数据传输进行地面在线监控的必要性; 并根据航电系统的工作原理和数据监控的研发流程, 研制了一款基于嵌入式操作系统的数据监控平台; 监控平台硬件部分在 Arm 处理器基础上, 设计并实现了数据采集、处理和电源管理等一系列功能模块; 平台软件的开发基于嵌入式 Android 系统, 在该系统环境下设计了监控平台软件构架; 并开发了测试资源的底层驱动层、硬件抽象层 (HAL)、本地调用层 (JNI) 和应用程序; 通过多部被测航电系统进行验证, 该监控平台运行稳定, 系统误差可忽略, 满足了航电系统数据监控的实际需求; 从工程应用结果表明, 该综合航电系统数据监控平台已通过民航局地面测试项目认证, 并应用于实际生产中。

关键词: 航电系统; 数据监控; Android

Development of Comprehensive Avionics Data Monitoring Platform Based on Android

Wang Haibin, Qian Wei, Ding Fajun
(Aircraft Maintenance Factory of CAFUC, Guanghan 618307, China)

Abstract: The paper analyzes the current situation of data transmission of the integrated avionic system in the field of general aviation and discusses the necessity to conduct ground on—line monitoring of data transmission of the integrated avionic system. Besides, the paper develops a data monitoring platform based on the embedded operating system according to the working principle of the avionic system and the R&D procedures of data monitoring. The hardware part of the monitoring platform is mounted on the ARM microprocessor, which realizes a series of functional modules like data collection, data processing, power management etc. The platform software is developed based on the embedded Android system, under the environment of which the software architecture of this monitoring platform is designed. Moreover, the paper has also developed the bottom driver layer, hardware abstraction layer (HAL), java native interface (JNI) and application programs of the test resources. Verified by several tested avionic systems, this monitoring platform runs stably and the systemic errors can be ignored, which satisfies the practical needs of data monitoring of the avionic system. As indicated by the engineering application results, this data monitoring platform for integrated avionic system has passed the ground test project certification of the Civil Aviation Administration and has been applied in practical production.

Keywords: avionics system; data monitoring; Android

0 前言

随着通航产业的发展, 新置通用飞机已全部安装综合航电系统。综合航电系统功能先进、操作便捷, 但整个系统运行依赖于高速数据传输模式, 若数据传输通道出现异常, 将会严重影响飞机安全飞行[1-3]。为确保航电系统数据传输的安全性, 对传输通道进行定期检测显得十分重要[4]。如何在地面构建一个数据监控平台, 实现对航电系统数据传输高效地检测成为本文研究的重点。便捷高效地实现数据通道检测不但能够确保系统安全运行, 还能够提前发现系统潜在故障, 并且能够通过检测平台确认故障子部件, 为技术人员维修排故提供重要的依据[5]。

本文针对某型综合航电系统进行数据监控平台开发, 该型综合航电系统为目前通航应用最为广泛的型号, 其数据传输采用高速以太总线完成[6-7]。本文在搭建相关硬件平台的基础上, 应用了嵌入式开发技术, 并在 Android 系统下开发了相关驱动及应用程序。下面结合数据监控平台的研制过程, 详细阐述其系统组成构架、软硬件关键技术及其工程应用结果。

1 系统结构

综合航电系统主要由显示组件、核心组件、航姿组件、音频组件、大气参数组件、发动机参数组件和应答机组件等 7 个组件组成[8-9]。其中, 核心组件是交联综合航电系统的枢纽, 各组件的传输数据经过核心组件统一转换为高速以太数据格式, 传输给显示组件。系统控制指令也是通过该高速以太数据经核心组件, 传输给各功能组件[10]。因此, 实现对高速以太数据的监控和测试成为本监控平台研究的重点。数据监控平台作为一个节点嵌入到综合航电系统中, 处于实时接收和数据处理的位置, 如图 1 所示。

2 监控平台硬件组成

监控平台硬件结构主要由数据处理模块、数据采集模块、

收稿日期:2015-11-30; **修回日期**:2015-12-29。
基金项目:国家民航局科技项目(MHRDZ201003)。
作者简介:王海斌 (1984-),男,内蒙古赤峰市人,硕士研究生,工程师,主要从事机载航电设备的维修与飞机维修工程方面的工作。

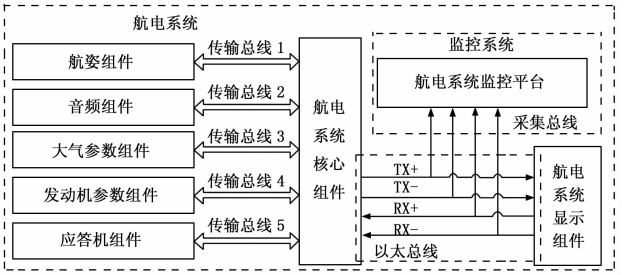

图 1　监控平台与航电系统连接图

电源模块、存储模块等组成。数据采集模块利用以太总线与航电系统实现连接，该模块实现了数据接收。平台通过数据处理模块进行算法处理，并通过用户界面实现系统控制，其系统组成如图 2 所示。

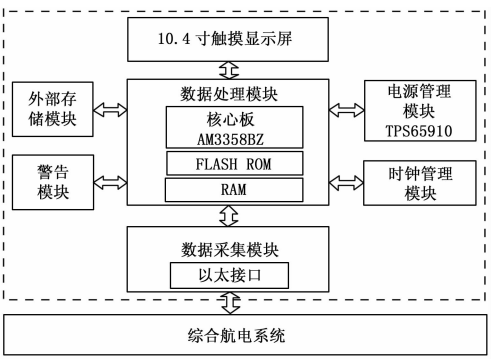

图 2　系统组成

数据处理模块中 CPU 处理器采用 TI AM3358BZ，其主频可支持 1 GHz，可轻松实现对以太总线数据进行一系列处理[11]。在总线传输数据协议方面采用 IEEE 802.3 标准网络协议，与航电系统以太协议保持一致。数据采集模块以太网收发器采用 RTL8211E 网络芯片，该芯片是瑞昱最新推出，并支持 1000Base－T。电源管理芯片采用 TPS65910 电源管理芯片，该芯片是 TI 推出的针对 CPU 供电的电源管理 IC，内部集成了 3 个降压，1 个升压以及 8 个差分电压，满足了平台所有特定的电源需求。显示模块采用三菱 GT1275－VB 10.4 寸触摸屏，功耗较低且操作便捷。平台的采集数据以及处理结果可自动至外部存储模块，该模块是由 SD 卡存储模式实现。

3　监控平台软件开发

3.1　监控平台 Android 系统架构

Android 是 Google 开发的基于 Linux 开源手机平台，它包括操作系统、应用程序和用户界面。虽然 Android 最初作为手机软件平台的操作系统，但其本质确是一款非常优秀的嵌入式开发平台[12]。考虑到 Android 比传统 Linux 系统更加强大，在设计开发航电系统监控平台时采用 Android4.0 作为操作系统。

但由于 Android 在系统构架很大程度上为实现手机功能进行设计，因此，在针对监控平台开发时，就必须对 Android 系统构架进行一系列修改。将通话、通讯录以及短信等模块删除，仅留下平台所需模块—Android 核心模块，主要包括 Package Manager 、System Service、Hardware Service 和 System Server 等。经过定制和修改过的 Android 系统架构如图 3 所示。

图 3　平台 Android 系统架构

监控平台软件系统共分 4 层：最底层为 Linux 核心及驱动层，该层封装了相关硬件设备，为上层提供了统一接口，可移植性极强；第 2 层为系统运行库层，提供系统函数库组件；第 3 层为应用程序框架层，包含所有开发所用各类库；顶层为应用程序层，包括系统应用程序模块的开发。

3.2　监控平台驱动开发

Android 系统驱动程序开发基于 Linux 内核基础上进行的，其应用程序使用 Java 开发，所以应用程序在调用设备驱动时并不能像直接调用，必须通过 Java 虚拟机的本地调用方法（JNI）实现。另一方面，为增强驱动程序可移植性，在 Android 架构添加一个硬件抽象层（HAL），从而为硬件设备的调用提供一个高级封装[13]。

监控平台功能运行最终通过应用程序来实现，在应用层中系统控制程序调用平台控制服务。平台控制服务通过 JNI 方法使得虚拟机加载本地库，然后向 HAL 层获取共享库文件（*. so），由共享库文件调用在 Linux 内核中的设备驱动。图 4 为监控平台 Android 驱动程序架构设计。从驱动程序架构来分，整个驱动程序设计可分为：底层驱动设计、硬件抽象层（HAL）设计、本地调用层（JNI）设计、平台控制服务层设计 4 个层次。

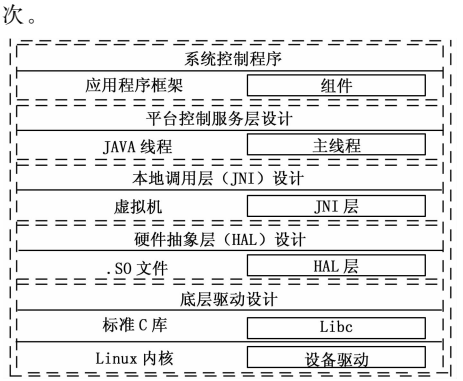

图 4　Android 驱动程序架构

3.2.1　底层驱动设计

驱动程序的最终目的，是为使上层应用程序能够使用这些硬件提供的服务来为用户提供软件功能。其处理过程是，驱动程序将硬件设备抽象成文件，应用程序将这些文件来进行处理[14]。

对于 Linux 标准设备驱动程序可直接使用，只需针对具体

设备参数修改相应内核驱动程序，然后再进行配置及编译，并增加应用层 JNI 接口。此类驱动包括以太总线驱动、显示驱动、音频驱动以及存储驱动。

本文以以太总线驱动为例进行说明，在 Linux 内核中已具备基本网卡驱动，但针对平台采用的 DM9000E 网卡芯片需要进行一定的修改。

步骤 1：需在头文件加入 # include＜linux/dm9000. h＞，再添加驱动描述如下：

```
Static struct platform_device TI3358_device_dm9k={
.name = "dm9000",
.id = 0,
.num_resources = ARRAY_SIZE(TI3358_dm9k_resource),
.resource = TI3358_dm9k_resource,
.dev = { .platform_data = &TI3358_dm9k_platdata,}
};
```

步骤 2：对相应的 MAC 地址进行修改：

```
#if defined(CONFIG_ARCH_ TI3358)
Printk("Now use the default MAC address:10:22:45:67:ad/n");
```

步骤 3：配置并编译内核；

利用配置菜单对 DM9000E 网卡进行配置，主要在 Networking support 和 Device Drivers 两项进行配置，具体实现如下：

```
[ * ]Networking support→
[ * ]TCP/IP networking
[ * ]Device Drivers→
[ * ]Ethernet(10 or 100Mbit)→
＜ * ＞ DM9000 support
```

菜单配置完毕，保存并编译出镜像，按此方式依次完成平台所需其余驱动模块的开发。将修改后内核驱动进行重新编译生成 zImage. bin 文件，移植到嵌入式系统并运行，以便软件系统上层进行调用开发。

3.2.2 硬件抽象层（HAL）设计

硬件抽象层是通过 HAL Stub 方法实现，HAL Stub 是一种代理方法，Stub 以共享库（ * . so）格式存在。HAL Stub 方法访问底层驱动一种间接调用方式，Stub 向硬件抽象层提供操作的回调函数。

JNI 层访问 HAL 层时，通过函数 hw _ get _ module（ ）获取设备模块 ID，并向 HAL 层申请设备 Stub，JNI 层获得 Stub 对象后，即可把 Stub 作为一个抽象硬件进行操作。图 5 为平台硬件抽象层（HAL）实现过程。

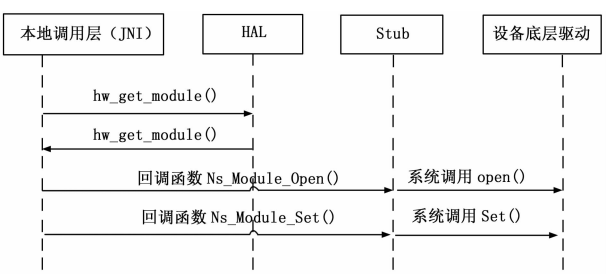

图 5 平台硬件抽象层（HAL）实现过程

下面程序为以太总线 Stub 的 HAL 结构体：

```
struct ethernet_module_t {struct hw_module_t common;}
struct ethernet_module_t {struct hw_module_t common;
int id;
int( * ns_set)(struct ethernet_control_device_t * dev,int32_t ethernet);}
```

将结构体 ethernet _ module _ t 初始化，该结构体包含了 Stub 的模块信息，主要包括：

Id：Stub 的模块 ID，通过 id 查找相关设备；

methods：定义回调函数 open（），负责申请结构体 ethernet _ control _ device _ t 的空间，注册回调函数接口，并打开相关设备驱动。

3.2.3 本地调用层（JNI）设计

硬件抽象层编译后生成“ * . so”文件，保存文件系统“/sysem/lib/”目录下。平台控制服务层运行后，由 JNI 虚拟机装载本地库函数，具体实现过程如图 6 所示。

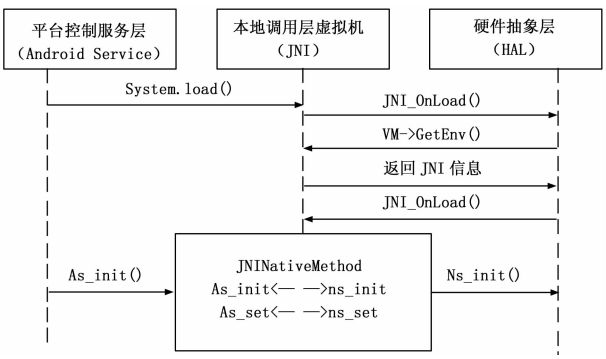

图 6 平台本地调用层（JNI）实现过程

平台控制服务层通过调用 System. load（）函数，使得 JNI 虚拟机加载本地库函数，这一过程是虚拟机通过调用 JNI _ OnLoad（）函数来实现：

1）把虚拟机信息保存到硬件抽象层结构体中；

2）建立平台控制服务层与硬件抽象层 JNI 函数表；

3）返回虚拟机、硬件抽象层使用的 JNI 信息。

加载完后，平台控制服务层就可以通过 JNI 函数表把 Java 函数转换为本地函数执行。

在本地调用层中定义 JNI 函数方式代码段如下：

```
public final class Service extends Service. Stub {
static {System. load("/system/lib/libled. so"); }
private static native boolean as_init();
private static native boolean as_set(int * * * );}
```

3.3 应用程序设计

监控平台应用程序设计方面考虑进行模块化设计，主要分为界面模块、测试模块、故障诊断模块和告警模块 4 个模块，如图 7 所示。

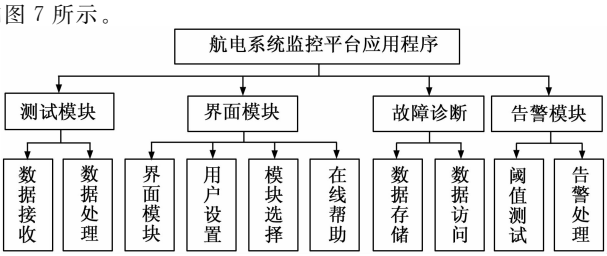

图 7 系统应用程序架构

表 1　主要参数实测值

测试参数	单位	理论值（允许误差）	被测系统 1		被测系统 2		被测系统 3		被测系统 4		最大误差
			通道 1	通道 2	通道 1	通道 2	通道 1	通道 2	通道 1	通道 2	
RPM	r/min	2500(±10)	2501	2500	2499	2498	2501	2502	2499	2500	+2
EGT	℉	1600(±20)	1602	1596	1605	1601	1597	1604	1605	1601	+5
OIL PRES	psi	115.0(±0.5)	115.1	115.0	115.0	115.0	115.1	114.8	115.0	115.0	−0.2
FFLOW GPH	gph	15(±2)	15	14	15	15	15	14	15	15	−1
ALT	Ft	1500(±20)	1502	1496	1503	1500	1500	1501	1502	1500	−4
HDG	°	45.0(±2.0)	45.0	45.1	45.0	45.0	45.2	49.7	45.0	45.1	−0.3

界面模块采用 Activity 方式实现，由主界面、模块选择、用户设置和帮助组成。模块选择用于选择测试部件、测试模式以及告警事件的显示和查询等。用户设置用于密码管理、语言切换、时区和时间的设置、网络地址的设置。帮助模块用于为使用者提供在线帮助。

测试模块采用 Service 方式实现，主要包括数据接收和处理模块，用于数据的采集和处理。数据接收实现监控平台与被测系统的数据通信，并将数据进行存储。数据处理模块针对采集的以太数据进行一系列算法处理，并与标准数据库数据进行参数比对，最终实现传输数据的监控。

故障诊断模块通过操作数据实现的，是在对测试数据分析、评估基础上实现的。针对数据异常情况进行处理，将故障进行定位，最终为用户提供排故指导策略。

告警模块利用 Broadcast Receiver 方式实现，负责数据通信异常的告警事件，同时调用告警显示界面显示对应的信息。

4　平台测试结果及误差分析

利用综合航电数据监控平台针对 4 部被测系统全部性能参数进行自动测试，提取测试结果，并进行分析和比对，最终验证监控平台系统误差。考虑到被测系统性能参数较多，本文仅列举了部分较重要参数，实测情况如表 1 所示。

根据表 1 列举的测试参数实测情况可知，测试参数误差均在理论允许误差范围内。针对实测参数误差进行分析可知，其误差来源为监控平台和被测系统两部分，而对于监控平台而言，其误差来源于平台软件和硬件资源。

1）误差源 1：平台软件方面。

软件方面误差源主要指传输数据处理的实现，即监控平台针对被测系统的传输数据进行一系列处理过程是否产生误差。

被测系统传输数据总线为 Ethernet 协议，该总线协议为国际通用标准格式，监控平台的数据处理基于标准协议基础上，实现 Ethernet 数据的编解码算法。因此，数据处理分析不产生误差。

2）误差源 2：平台硬件方面。

硬件方面误差源主要指数据传输线路。数据转换电路在转换瞬间会产生瞬时脉冲信号，对系统 Ethernet 信号会造成瞬时干扰，存在产生误码的可能性。根据被测系统 Ethernet 总线通信协议规定，传输数据具有自校、奇偶检验功能。经验证，瞬时干扰误码不会对传输信号产生附加误差。

综上分析，监控平台对测试结果产生的附加误差可忽略。并通过表 1 中 4 部被测系统实测数据验证，测试误差均在性能指标要求范围内，且误差来源为被测部件本身。

5　结束语

基于 Android 系统的航电数据监控平台作为综合航电系统上的一个关键节点，通过以太总线与系统进行通信，对航电系统的航姿、大气数据、发动机参数等组件进行地面在线监测。最终实现了对综合航电系统传输数据测试，以及各飞行参数组件的功能测试。经实测，监控平台以 10 Mbps 的通信速率运行，且保持测试状态稳定。目前，该项目已通过民航局地面测试项目认证，已应用于综合航电系统定检达 20 余套，取得了良好的应用效果。

参考文献：

[1] 赵　明. 通用飞机综合航电技术发展综述 [J]. 电讯技术，2014，54 (3)：374－378.

[2] 郑　澜，王运盛. 适用于民机 IMA 的通用机载软件开发平台 [J]. 电讯技术，2012，52 (6)：1027－1029.

[3] 周　庆，刘　斌，余正伟，等. 综合模块化航电软件仿真测试环境研究 [J]. 航空学报，2012，33 (4)：722－733.

[4] 孙　兵，何　瑾，陈广厦. 基于 DSP 的 CAN 总线与以太网互联系统研制 [J]. 仪器仪表学报，2008，29 (2)：377－380.

[5] 滕秋琴. ARM 嵌入式系统网络接口设计 [J]. 电讯技术，2008，48 (10)：84－86.

[6] Littlefield－Lawwill J，Viswanathan R. Advancing open standards in integrated Modular avionics：An industry analysis [A]. AIAA/IEEE Digital Avionics Systems Conference－Proceedings. Piscataway [C]. NJ：IEEE，2007：2B11－2B114.

[7] 梁永生，张基宏，张乃通. IEEE 标准容限内以太网转发时延的测试与分析 [J]. 电子学报，2008，36 (1)：46－50.

[8] Steve Gorshe，Jeff Mandin. Introduction to IEEE 802. 3 av 10Gbit/s Ethernet Passive Optical Networks (10G EPON) [J]. China Communications，2009，6 (4)：136－147.

[9] 熊华钢，周贵荣，李　峭. 机载总线网络及其发展 [J]. 航空学报，2006，27 (6)：1135－1144.

[10] 汪健甄，许宗泽. 航空电子高速数据总线性能分析及其实时性仿真 [J]. 南京航空航天大学学报，2008，40 (3)：345－347.

[11] 曾祥文，宋树祥，宾相邦. 嵌入式 Linux 下波特率自适应的 CAN 总线驱动的实现 [J]. 测控技术，2015，34 (8)：104－107.

[12] 詹成国，朱　伟，徐　敏. 基于 Android 的测控装置人机界面的设计与开发 [J]. 电力自动化设备，2012，32 (1)：119－122.

[13] 农丽萍，王力虎，黄一平. Android 在嵌入式车载导航系统的应用研究 [J]. 计算机工程与设计，2010，31 (11)：2473－2475.

[14] 韩　迪，潘志宏. 基于 Android 移动设备传感器的体感应用 [J]. 华南理工大学学报 (自然科学版)，2012，40 (9)：75－77.