Android Frameworks and HAL Implementation

Objectives

- Explore the Android source code architecture
  - The Android init process
  - System services
  - The Android Binder
  - The Android Application Framework
  - The Android Hardware Abstraction Layer
  - The Android Multimedia Framework and OpenMAX
- Understand the static and dynamic framework structure
  - Class structure
  - Split between Java and C++ code
  - Data flow through the frameworks
  - Control structure of the frameworks

Labs are conducted on the Android emulator
We use the last released AOSP (Android Open Source Platform) version.
For on-site trainings, if suitable Linux workstations are not available, we provide virtual machine images for VirtualBox; in all cases the requisite is a recent 64bit PC (at least 4 cores) with 64Gb of RAM (32Gb may work but is not recommended) and 600Gb of free disk space.

Who should attend this course?

- Engineers that must work on the Android port on a new platform
  - Writing the HAL for a new board
  - Debugging the HAL and Android frameworks

Prerequisite

- Good Linux kernel and driver programming experience (see our D3 - Linux Driverscourse)
- Android installation knowledge (see our G1 - Android Installationcourse)
- Basic knowledge of the structure of an Android application(see our G2 - Android Programmingcourse)
- Good C++ and Java programming skills (see our L3 - Embedded C++course and L4G - Java for Androidcourse)

Course Environment

- Theoretical course
  - PDF course material (in English) supplemented by a printed version.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- Practical activities
  - Practical activities represent from 40% to 50% of course duration.
  - Code examples, exercises and solutions
  - One PC (Linux ou Windows) for the practical activities with, if appropriate, a target board.
    - One PC for two trainees when there are more than 6 trainees.
  - For onsite trainings:
    - An installation and test manual is provided to allow preinstallation of the needed software.
    - The trainer come with target boards if needed during the practical activities (and bring them back at the end of the course).

- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Course modularity

- This course can be dispensed from 3 to 5 days:
  - The first three days are mandatory to understand the architecture of the Android Frameworks
  - The fourth day covers in more depth the multimedia and video handling
  - The fifth day covers the audio framework in detail (this requires part of the knowledge dispensed in the fourth day, so is only available in a full 5 days course)

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.
- Trainee progress is assessed in two different ways, depending on the course:
  - For courses lending themselves to practical exercises, the results of the exercises are checked by the trainer while, if necessary, helping trainees to carry them out by providing additional details.
  - Quizzes are offered at the end of sections that do not include practical exercises to verifythat the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

First Day

Android Architecture Overview

- Linux and Android
- Android Licensing

The Android Linux kernel enhancements

- The Android-specific kernel drivers
  - Ashmem
  - Logger
  - Low_memory_killer
  - Timed_output
  - Timed_gpio
  - Buttons and Keypad management
- Android Power Management
  - The Linux Power Management architecture
  - Android Wake Locks
  - Android Power Management in Linux drivers
- The Android Kernel debugger

Exercise: *Configuration and build of the Android kernel for the target board*
Exercise: *Checking the first phases of kernel boot*

The Android Build system

- The Android code base

- Building Android
  - The Android build environment
  - The Android build system
  - The Android.mk files
- Adding new components to the build system
  - Java components
  - Native components
  - Applications

Exercise: *Compiling the Android platform*

Android System Initialization

- Android properties
  - Automatic properties
  - Default properties
  - Persistent properties
- The Android initialization
  - Structure of the init process
  - The Android initialization language
- The Dalvik Java virtual machine
  - The Dalvik machine structure
  - The Dalvik bytecodes
  - The Dalvik zygote process

Second Day

Android Native Interface

- The bionic C library
  - Why a new C library
  - The bionic Android-specific features
  - What is missing in bionic
- Adding native components
  - Adding native executables
  - Defining Java methods in C or C++
  - JNI for Android
- Platform interface for native components
  - Accessing system properties
  - Accessing the Android log system
  - Interacting with daemon services

Exercise: *Creating a new native component*

Android as a Massively Distributed System

- The Android Binder architecture
  - Why a new IPC mechanism
  - The Binder in action
  - The Binder kernel driver
- Binder implementation
  - The AIDL language
  - The AIDL tool
  - Binder Java classes
  - C++ binder implementation classes
- Writing Services
  - Standard Java services
  - Services and C++
  - Using the binder from C++
- System services
  - What is a system service

  - Static and context-dependent services
  - Structure of a system service
  - Adding a new system service
  - The system ServiceManager process

Exercise: *Coding a system service*

The Android Power Manager

- The Driver API
- The user-mode API
- The Java API

Third Day

The Hardware Abstraction Layer

- Why a HAL?
- HAL component structure
  - Defining HAL components
  - Loading and using HAL component
- The standard HAL components
  - Graphics
  - Audio
  - Camera
  - Bluetooth
  - GPS
  - Sensors
  - WiFi
  - The Radio Interface Layer (RIL)

Exercise: *Create a simple HAL component*

The Android Sensors Framework

- Sensors in Android
  - The sensor types
  - The Sensor Manager
  - Accessing Sensors
- Framework Architecture
  - Sensor discovery
  - Sensor Calibration
- The HAL components

The Android Multimedia Framework

- Multimedia in an Android device
  - Data formats and File formats
  - Codec and Demux
- Multimedia for Applications
  - Audio and video playback (MediaPlayer class)
  - Audio and video capture (MediaRecorder class)

Exercise: *Implementation of an mp3 playback service*

- Framework Architecture
  - General framework architecture
  - General data and control flows
  - The MediaPlayer service layer
  - Stagefright and OpenMAX

Fourth Day

OpenMAX for Android Multimedia

- OpenMAX Overview
  - The Khronos Group
  - OpenMAX/DL: the Development Layer
  - OpenMAX/IL: the Integration Layer
  - OpenMAX/AL: the Application Layer
  - OpenMAX and OpenSL/ES
- OpenMAX in the Android Media Framework
  - Interface between Android and OpenMAX
  - The OpenMAX/IL Architecture
- Stagefright the Android playback engine
  - The Stagefright class structure
- The OpenMAX/IL Bellagio implementation
  - OpenMAX/IL LGPL implementation of the core
  - Sample implementation of components
- Anatomy of a component
  - Configuration interface
  - Data interface
  - Buffer allocation
  - Bellagio specific setup
- Integration of OpenMAX/IL in Stagefright
  - Component registration
  - Component configuration
  - Component Quirks

Android Graphics low-level components

- The Surface Flinger
  - The Binder interface
  - OpenGL/ES interface
  - Using hardware accelerators
  - Double buffering using framebuffer page-flip
- The HAL graphics components
  - Framebuffer control
  - Graphic memory allocator
  - Bit blitting
  - The Hardware composer

Fifth Day

The Android Audio Manager

- Audio routing
- General architecture of Audio manager
  - Audio system
  - Audio policy manager
  - Audio policy service
  - Sound effects
- Control flow
  - Playback and recording control
  - Time source generation

The Audio Flinger

- Output/input audio flow
  - Buffer management
- Audio track
  - Overview
  - Track live cycle
  - Data flow
- Audio mixer:
  - Overview
  - Mixer life cycle
  - Fast mixer
  - Resampling
  - Volume
- Audio recorder:
  - Overview
  - Data flow

The HAL Audio components

- Audio policies
  - Audio Policies and policy devices
  - HAL Audio policy provided services
  - Use from the audio policy manager
  - Use of audio services by the HAL audio policy
  - Output duplication
  - Suspend/resume
- Audio devices
  - Audio device classes
  - Data flow
  - Interaction with audio track and record
- Audio effects

Renseignements pratiques

Inquiry : 5 days