Institut für Informatik
Lehrstuhl Human Centered Multimedia

Prof. Dr. Elisabeth André
Simon Flutura

Universität Augsburg University

SSI on Android, SS 2015

1. Introduction

SSI - Social Signal Iterpretation Framework has been portet to Linux with an Android-port in the works. Devices like the Raspberry PI, as well as several mobile devices can now be used. This guide shows you how to set up SSI on Android

Task 1: Prepare the setup (**note:** crosscompile from Linux)

Some requirements have to be met to be able to use SSI on Linux. On any reasonable Distribution you should simply be able to install the dependencies via package manager.

1. GCC and G++ 4.8 or later have to be installed, because SSI relies on C++11 standard.
2. CMake is needed as build-system.
3. Android NDK is needed and has to be setup on the your system

Task 2: Get SSI

SSI can be found here:

1. from git `https://hcm-lab.de/git/groups/ssi`
2. from subersion `https://hcm-lab.de/svn/Johannes/openssi/trunk/`
3. from `cleanport.tar.gz` archive

Task 3: Prepare Buildsystem

1. To build SSI for Android a crosscompilation toolchain needs to be setup
   cd /to/where/android-ndk/build/tools/
   `./make-standalone-toolchain.sh \`
   `--toolchain=arm-linux-androideabi-4.9 --ndk-dir=/opt/android-ndk/`
   this line will generate an archive containing your stand alone toolchain
   `/android-cmake-7e4cd91276af/toolchain/android.toolchain.cmake`
   file is needed in trunk/../android-cross/
   Or simply copy from `doc/ssi-port-cmake/intro-android/android.toolchain.cmake`

Task 4: Prepare Buildsystem for APK

for apk creation some additional steps are neccessary.

1. copy * from `trunk/docs/apk` to trunk.
2. additionally copy `trunk/docs/apk/android/AndroidManifest.xml` to trunk.
3. edit android.apk.cmake to match your device regarding `ANDROID_APK_API_LEVEL`

your apk will be build in `trunk/bin_cmake/Apk/bin/`

Task 5: Now you are ready to build ssi

1. create a build directory next to the ssi root directory:
   ```
   mkdir ./trunk/../build
   ```
2. run cmake from the build directory to create the build environment:
   ```
   export ANDROID_ABI=armeabi-v7a
   export ANDROID_NATIVE_API_LEVEL=android-21
   cmake ../trunk/ -DCMAKE_TOOLCHAIN_FILE=\
   ../android-cross/android.toolchain.cmake \
    -DANDROID_ABI=armeabi-v7a
   ```
3. run make to build and install ssi(with 6 threads):
   ```
   make -j6 install
   ```
   note to rebuild the APK you have to:
   ```
   make clean -f plugins/androidSensors/tools/Makefile
   make -j6 install
   ```
   Adapt `ANDROID_ABI` and `ANDROID_NATIVE_API_LEVEL` according to your device or emulator!

Task 6: Run test

The above step has created an APK in ssi roots `bin_cmake/Apk/bin` directory.

Install with `adb install android_xmlpipe-debug.apk`.

`adb logcat` lets you see ssi output for debuging.

Task 7: Run tests on rooted device

Your freshly build binarys are located in ssi roots `bin_cmake` directory:

`trunk/cmake_bin/Android`

1. create and start your emulator:
   ```
   android create avd -t 2 -n name
   ```
   where -t names the correct target from android list targets

2. copy files to emulator:
   ```
   adb -e push trunk/cmake_bin/Android /data/test
   adb -e shell chmod 777 /data/test/*
   adb -e shell
   ```
3. To test a simple pipeline: then run `./ssiandroid_test` from `data/test/`. The correct working directory is needed, so the shared plugins can be found.
4. Apart from c++ code one can as well run xml pipelines via:
   `./xmlpipe test.pipeline`
5. The same steps are valid for rooted Android devices. Use `adb -d` to connect to your device. Do not forget to set the
   `export LD_LIBRARY_PATH=".:/system/lib/"` or fixed to the SSI root path, so shared libraries can be found.

Task 8: Choose IDE

To further develop SSI on linux QtCreator is recommended. Do not forget to install the GNU debugger GDB.

Setting up remode GDB follows

Have fun.