在 **Raspberry Pi** 上使用 **AVS Device SDK** 制作 **Alexa** 智能音箱

Phodal Huang

September 8, 2017

目录

原文链接：https://www.wandianshenme.com/play/build-amazon-avs-device-sdk-on-raspberry-pi

在本玩法中，我们将逐步介绍如何设置 AVS Device SDK，并获取在 Raspberry Pi 3 上运行的 C++ 示例应用程序。在开始之前，我们建议您观看“入门视频”，以确定 AVS Device SDK 是如何工作的。

以下是您需要做的事情：

- 在 Amazon 上注册你的设备
- 在您的 Raspberry Pi 上安装，并配置 AVS Device SDK 的依赖关系。
- 构建 SDK，并在您的 Raspberry Pi 上运行示例应用程序。

步骤 1：在 Amazon 上注册你的设备

按照以下说明：Create Security Profile，注册你的产品，并创建安全配置文件。您将使用 client ID 和 client secret，来检索用于与 Alexa 进行通信的访问和刷新令牌。请注意，Web 设置下的允许 origin 和返回 URL 应分别为：http://localhost:3000 和 http://localhost:3000/authresponse。

注册设备后，转到安全配置（Security Profile）文件下的常规（General）选项卡，并记下 clientID，clientSecret 和 deviceTypeID。您将需要这些信息来配置 AuthServer。

步骤 2：在您的 Raspberry Pi 上安装和配置 AVS Device SDK 的依赖关系

这些说明适用于，在 Raspberry Pi 3 运行 Raspbian Jessie 设置开发环境。

这些说明将指导您，为每个依赖关系构建所需的最低版本。如果您计划使用较新版本，请验证每个命令，具体配置步骤。

提示：如果您不想安装 GUI 分发，所有的内容都在 Raspbian Jessie Lite 上测试、执行。

2.1 为依赖创建源文件夹

为了让您的 Raspberry Pi 能使用 AVS Device SDK，您需要从源代码构建一些依赖项。将源文件保存在预定义的文件夹中，是最佳做法。使用命令创建源文件夹。你可以随意修改 SOURCE_FOLDER值。但是请记住，本文档中将引用此变量：

```
1 echo "export SOURCE_FOLDER=$HOME/sources" >> $HOME/.bash_aliases
2 echo "export LOCAL_BUILD=$HOME/local-builds" >> $HOME/.bash_aliases
```

```
3 echo "export LD_LIBRARY_PATH=$HOME/local-builds/lib:$LD_LIBRARY_PATH" >>
    $HOME/.bash_aliases
4 echo "export PATH=$HOME/local-builds/bin:$PATH" >> $HOME/.bash_aliasesecho
    "export PKG_CONFIG_PATH=$HOME/local-builds/lib/pkgconfig:$PKG_CONFIG_PATH" >> $HOME/.bash
5 source $HOME/.bashrc
6 mkdir $SOURCE_FOLDER
```

2.2 下载构建工具

下一步是下载一些基本的构建工具：

```
1 sudo apt-get install git gcc cmake build-essential
```

2.3 配置 libcurl、nghttp2、openssl

这是搭建过程中最重要的部分，因为连接到 AVS 需要使用 HTTP2，SDK 使用 libcurl 建立该连接。

2.3.1 从源码中构建 nghttp2

重要信息：Jessie 中所提供的 nghttp2 开发包已过期，因此您需要从源代码构建正确的版本。

首先要获取源码：

```
1 cd $SOURCE_FOLDER
2 wget
    https://github.com/nghttp2/nghttp2/releases/download/v1.0.0/nghttp2-1.0.0.tar.gz
3 tar xzf nghttp2-1.0.0.tar.gz
```

然后便是通过下面的命令，配置、构建并安装：

```
1 cd $SOURCE_FOLDER/*nghttp2*/
2 ./configure --prefix=$LOCAL_BUILD --disable-app
3 make -j3
4 sudo make install
```

2.3.2 从源码中构建 openssl

重要信息：Jessie 中所提供的 openssl 开发包已过期，因此您需要从源代码构建正确的版本。

使用下面的命令来下载并安装 openssl：

```
cd $SOURCE_FOLDER
wget https://www.openssl.org/source/old/1.0.2/openssl-1.0.2a.tar.gz
tar xzf openssl-1.0.2a.tar.gz
cd *openssl*/
./config --prefix=$LOCAL_BUILD --openssldir=$LOCAL_BUILD shared
make -j3
sudo make install
```

2.3.2 从源码中构建 libcurl

现在 nghttp2 和 openssl 都已经从源码中构建完了，我们可以构建 cURL 了。

运行下面的命令：

```
cd $SOURCE_FOLDER
wget https://curl.haxx.se/download/curl-7.50.2.tar.gz
tar xzf curl-7.50.2.tar.gz
cd *curl*/
./configure --with-ssl=$LOCAL_BUILD --with-nghttp2=$LOCAL_BUILD
    --prefix=$LOCAL_BUILD
make -j3
sudo make install
```

2.4 安装 sqlite

SQLite 是 Alerts 所需要的组件。为了安装它，只需要运行这个命令：

```
sudo apt-get install sqlite3 libsqlite3-dev
```

而为了使用自带的 Media Player 实现和构建示例应用程序，则需要以下的依赖。

2.5 构建 gstreamer 库

示例应用程序需要媒体播放器，来实现播放 MP3 文件; 我们的实现支持 GStreamer。而在构建 GStreamer 之前，需要安装一些实用程序和依赖项。

运行下面的命令来安装：

```
sudo apt-get install bison flex libglib2.0-dev libasound2-dev pulseaudio
    libpulse-dev
```

从 iHeartRadio 播放音频也需要以下内容：

```
sudo apt-get install libfaad-dev libsoup2.4-dev libgcrypt20-dev
```

2.5.1 构建 gstreamer-1.10.4

```
cd $SOURCE_FOLDER
wget https://gstreamer.freedesktop.org/src/gstreamer/gstreamer-1.10.4.tar.xz
tar xf gstreamer-1.10.4.tar.xz
cd *gstreamer*/
./configure --prefix=$LOCAL_BUILD
make -j3
sudo make install
```

2.5.2 构建 gst-plugins-base-1.10.4

```
cd $SOURCE_FOLDER
wget
    https://gstreamer.freedesktop.org/src/gst-plugins-base/gst-plugins-base-1.10.4.tar.xz
tar xf gst-plugins-base-1.10.4.tar.xz
cd *gst-plugins-base*/
./configure --prefix=$LOCAL_BUILD
make -j3
sudo make install
```

2.5.3 构建 gst-libav-1.10.4

```
cd $SOURCE_FOLDER
wget https://gstreamer.freedesktop.org/src/gst-libav/gst-libav-1.10.4.tar.xz
tar xf gst-libav-1.10.4.tar.xz
cd *gst-libav*/
./configure --prefix=$LOCAL_BUILD
make -j3
sudo make install
```

2.5.4 构建 gst-plugins-good-1.10.4

```
cd $SOURCE_FOLDER
wget
    https://gstreamer.freedesktop.org/src/gst-plugins-good/gst-plugins-good-1.10.4.tar.xz
tar xf gst-plugins-good-1.10.4.tar.xz
```

```
cd *gst-plugins-good*/
./configure --prefix=$LOCAL_BUILD
make -j3sudo make install
```

2.5.5 构建 gst-plugins-bad-1.10.4

```
cd $SOURCE_FOLDER
wget
    https://gstreamer.freedesktop.org/src/gst-plugins-bad/gst-plugins-bad-1.10.4.tar.xz
tar xf gst-plugins-bad-1.10.4.tar.xz
cd *gst-plugins-bad*/
./configure --prefix=$LOCAL_BUILD
make -j3
sudo make install
```

2.6 构建 portaudio

PortAudio 提供了一个简单的 API 来录制和播放声音。这也是使用示例应用程序所必需的。

运行这些命令，从源码下载和构建：

```
cd $SOURCE_FOLDER
wget http://www.portaudio.com/archives/pa_stable_v190600_20161030.tgz
tar xf pa_stable_v190600_20161030.tgz
cd *portaudio*/
./configure --prefix=$LOCAL_BUILD
make -j3
sudo make install
```

2.7 安装 Sensory

在本指南中，我们将使用 Sensory 作为唤醒词引擎来检测唤醒词 Alexa。按照以下说明，安装所需的软件包和依赖项。

2.7.1 安装 Sensory 依赖

```
sudo apt-get -y install libasound2-dev
sudo apt-get -y install libatlas-base-dev
sudo ldconfig
```

2.7.2 安装 Sensory

```
cd $SOURCE_FOLDER
git clone git://github.com/Sensory/alexa-rpi.git

bash alexa-rpi/bin/license.sh

cp alexa-rpi/lib/libsnsr.a $LOCAL_BUILD/lib
cp alexa-rpi/include/snsr.h $LOCAL_BUILD/include
mkdir $LOCAL_BUILD/models
cp alexa-rpi/models/spot-alexa-rpi-31000.snsr $LOCAL_BUILD/models
```

2.8 确保一切都是最新的

确保您具有每个依赖关系的最新版本。

```
sudo apt-get update
```

步骤 3：在你的 Raspberry Pi 上构建 SDK，并运行示例

3.1 下载或克隆 AVS Device SDK

地址是:https://github.com/alexa/avs-device-sdk，命令如下:

```
cd $HOME
mkdir AVS_SDK
cd AVS_SDK
git clone git://github.com/alexa/avs-device-sdk.git
echo "export SDK_SRC=$HOME/AVS_SDK/avs-device-sdk" >> $HOME/.bash_aliases
source $HOME/.bashrc
```

3.2 使用 Sensory，GStreamer 和 PortAudio 创建 SDK 构建:

假设 $SDK_SRC 是 SDK 源位置。

创建一个构建目录:

```
cd $HOME
mkdir BUILD
```

```
cd BUILD
```

进入构建目录，并通过在终端中复制下面的 cmake 命令，来创建一个源代码外的 SDK 构建。

```
cmake $SDK_SRC -DSENSORY_KEY_WORD_DETECTOR=ON
    -DSENSORY_KEY_WORD_DETECTOR_LIB_PATH=$LOCAL_BUILD/lib/libsnsr.a
    -DSENSORY_KEY_WORD_DETECTOR_INCLUDE_DIR=$LOCAL_BUILD/include
    -DGSTREAMER_MEDIA_PLAYER=ON -DPORTAUDIO=ON
    -DPORTAUDIO_LIB_PATH=$LOCAL_BUILD/lib/libportaudio.a
    -DPORTAUDIO_INCLUDE_DIR=$LOCAL_BUILD/include
    -DCMAKE_PREFIX_PATH=$LOCAL_BUILD -DCMAKE_INSTALL_PREFIX=$LOCAL_BUILD
```

3.3 更新 AlexaClientSDKConfig.json

创建源代码外的构建后，转到构建目录并使用自己喜欢的文本编辑器，打开 Integration 文件夹中的 AlexaClientSDKConfig.json 文件。填写你在设备注册期间，记录下的信息。

```
{
   "authDelegate":{
       "clientSecret":"${SDK_CONFIG_CLIENT_SECRET}",
       "deviceSerialNumber":"${SDK_CONFIG_DEVICE_SERIAL_NUMBER}",
       "refreshToken":"${SDK_CONFIG_REFRESH_TOKEN}",
       "clientId":"${SDK_CONFIG_CLIENT_ID}",
       "deviceTypeId":"${SDK_CONFIG_DEVICE_TYPE_ID}"
    },

  "alertsCapabilityAgent":{
    "databaseFilePath":"${SDK_SQLITE_DATABASE_FILE_PATH}",
    "alarmSoundFilePath":"${SDK_ALARM_DEFAULT_SOUND_FILE_PATH}",
    "alarmShortSoundFilePath":"${SDK_ALARM_SHORT_SOUND_FILE_PATH}",
    "timerSoundFilePath":"${SDK_TIMER_DEFAULT_SOUND_FILE_PATH}",
    "timerShortSoundFilePath":"${SDK_TIMER_SHORT_SOUND_FILE_PATH}"
  }
}
```

确保路径中没有任何额外的字符（或空格）。该配置还包含声音文件的路径，该文件

将用于播放警报（Alarms）和计时器（Timer）的声音。您可以从以下链接，获取“定时器和警报”获取所需的声音文件：Alexa Voice Service UX Design Guidelines

这时，需要数据库来存储预定的警报。在您的配置文件中，将文件路径位置包含到要用于存储和读取警报的数据库中。如果数据库不存在，将在该位置创建一个数据库文件。例如 `/home/anExistingFolder/anotherExistingFolder/alerts.db`。

3.4 安装

填写完 JSON 文件，并重新检查之后，进入终端的构建目录并运行“make”。在我们使用的 Raspberry Pi 上，这部分大概在 30~45 分钟的时间。

```
1 cd $HOME/BUILD
2 make -j3
3 make install
```

3.5 运行 AuthServer

构建完成后，您需要为设备获取刷新令牌。只需运行 AuthServer，一个将处理令牌交换的本地服务器将实现它：

```
1 python AuthServer/AuthServer.py
```

注意：如果您在这里遇到错误，将在浏览器中弹出错误，请单击以获取更多详细信息，以确保您能够返回并更正它们。

3.6 更新刷新令牌

从您的 Pi，打开浏览器并导航到 http//localhost3000。登录您的 Amazon 开发人员凭据，并在指示时关闭窗口。

注意：如果有任何错误，请仔细检查，因为它们会清楚出现什么问题。JSON 文件中的设备信息不正确，是最常见的问题原因。如果需要，请打开 JSON 文件，并确认您的刷新令牌有一行新行。

3.7 运行示例应用程序

您现在可以运行示例应用程序了。从构建目录导航到 SampleApp/src 文件夹并运行以下命令：

```
1 TZ=UTC ./SampleApp <REQUIRED-path-to-config-json> $LOCAL_BUILD/models
```

然后将弹出命令行界面。因为你设置了一个唤醒字引擎，因此你需要做的只是说“Alexa”。

在使用 SDK 构建您的第一个原型，并设置开发环境后，请访问我们的 Github 页面以获取更多资源和指南。

原文地址：https://github.com/alexa/avs-device-sdk/wiki/Raspberry-Pi-Quick-Start-Guide

原文链接：https://www.wandianshenme.com/play/build-amazon-avs-device-sdk-on-raspberry-pi