

CSDN新首页上线啦，邀请你来立即体验！[\(http://blog.csdn.net/\)](http://blog.csdn.net/)

立即体验

CSDN

博客 (<http://blog.csdn.net/?ref=toolbar>) 学院 (<http://edu.csdn.net/?ref=toolbar>)

下载 (<http://download.csdn.net/?ref=toolbar>) 更多 ▾

登录 (<https://passport.csdn.net/account/login?ref=toolbar>) 注册 (<http://passport.csdn.net/account/mobileregister?ref=toolbar&action=mobileRegister>)

1 /activity?utm\_source=csdnblog1)

android音频降噪webrtc

原创 2016年11月20日 13:13:58

评论

标签: [android](http://so.csdn.net/so/search/s.do?q=android&t=blog) / [webrtc](http://so.csdn.net/so/search/s.do?q=webrtc&t=blog) / [源码](http://so.csdn.net/so/search/s.do?q=源码&t=blog) / [降噪](http://so.csdn.net/so/search/s.do?q=降噪&t=blog)

2850

在音频处理的开源项目中，webrtc是一个很不错的例子。它包含降噪，去回声，增益，均衡等音频处理。这里我讲讲我所使用到的如何使用降噪方式。当然，具体它是如何降噪的，大家可以细看源码处理了。好了，线上源码。

以下是java 层MainActivity.java：

hesong (<http://blog.csdn.net/hesong1120>)

+ 关注

原创 14

粉丝 1

喜欢 0

码云未开通 (<https://gitee.com/hesong1120>)

他的最新文章  
更多文章 (<http://blog.csdn.net/hesong1120>)

Okio精简高效的IO库 (<http://blog.csdn.net/hesong1120/article/details/78652565>)

Okhttp解析（五）缓存的处理 (<http://blog.csdn.net/hesong1120/article/details/78584028>)

Okhttp解析（四）网络连接的建立 (<http://blog.csdn.net/hesong1120/article/details/78523308>)

u盘定制

在线课程

JDK9的新特性

[http://edu.csdn.net/course/detail/6134?utm\\_source=blog9](http://edu.csdn.net/course/detail/6134?utm_source=blog9)

SDCC 2017 之容器技术实践线上峰会

[http://edu.csdn.net/course/detail/73?utm\\_source=blog9](http://edu.csdn.net/course/detail/73?utm_source=blog9)

内容举报

返回顶部

他的热门文章

```
1 package com.test.jni;
2
3 import android.media.AudioFormat;
4 import android.media.AudioManager;
5 import android.media.AudioRecord;
6 import android.media.AudioTrack;
7 import android.media.MediaRecorder;
8 import android.os.Bundle;
9 import android.os.Environment;
10 import android.support.v7.app.AppCompatActivity;
11 import android.util.Log;
12 import android.view.View;
13 import android.widget.CheckBox;
14 import android.widget.CompoundButton;
15 import android.widget.SeekBar;
16
17 import java.io.FileNotFoundException;
18 import java.io.FileOutputStream;
19 import java.io.IOException;
20 import java.io.OutputStream;
21
22 public class MainActivity extends AppCompatActivity implements View.OnClickListener{
23
24     SeekBar skbVolume;//调节音量
25     boolean isProcessing = true;//是否录放的标记
26     boolean isRecording = false;//是否录放的标记
27
28     static final int frequency = 8000;
29     static final int channelConfiguration = AudioFormat.CHANNEL_CONFIGURATION_MONO;
30     static final int audioEncoding = AudioFormat.ENCODING_PCM_16BIT;
31     int recBufSize, playBufSize;
32     AudioRecord audioRecord;
33     AudioTrack audioTrack;
34
35     private String outFilePath;
36     private OutputStream mOutputStream;
37     private static final int FLAG_RECORD_START = 1;
38     private static final int FLAG_RECORDING = 2;
39     private static final int FLAG_RECORD_FINISH = 3;
40
41     private WebrtcProcessor mProcessor;
42
43     @Override
44     public void onCreate(Bundle savedInstanceState) {
45         super.onCreate(savedInstanceState);
46         setContentView(R.layout.activity_main);
47         //获得合适的录音缓存大小
48         recBufSize = AudioRecord.getMinBufferSize(frequency, channelConfiguration, audioEncoding);
49         Log.e("", "recBufSize:" + recBufSize);
50         //获得合适的播放缓存大小
51         playBufSize = AudioTrack.getMinBufferSize(frequency, channelConfiguration, audioEncoding);
52
53         //创建录音和播放实例
54         audioRecord = new AudioRecord(MediaRecorder.AudioSource.MIC, frequency, channelConfiguration, audioEncoding,
55         audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC, frequency, channelConfiguration, audioEncoding, play
```

android音频降噪webrtc (<http://blog.csdn.net/hesong1120/article/details/53240306>)

2827

Java线程Thread.join方法解析 (<http://blog.csdn.net/hesong1120/article/details/64906716>)

404

android如何给整个视图view圆角显示 (<http://blog.csdn.net/hesong1120/article/details/52005895>)

306

侧滑菜单(抽屉效果)DrawerLayout实现原理 (<http://blog.csdn.net/hesong1120/article/details/78241690>)

147

SurfaceView原理简述 (<http://blog.csdn.net/hesong1120/article/details/78174195>)

105

## 相关推荐

基于Linux webRTC 语音对讲之一——获取代码及编译 (<http://blog.csdn.net/peixiuhui/article/details/46745247>)

madplay的使用方法 (<http://blog.csdn.net/u012951123/article/details/17067023>)

使用webRtc进行音频降噪（NS）和VAD检测 ([http://blog.csdn.net/qq\\_30948113/article/details/68928549](http://blog.csdn.net/qq_30948113/article/details/68928549))

webrtc--AudioProcessing-- 音频降噪的处理过程 (<http://blog.csdn.net/huozaisinian/article/details/53323045>)



内容举报



返回顶部

广告

```
57     findViewById(R.id.btnRecord).setOnClickListener(this);
58     findViewById(R.id.btnStop).setOnClickListener(this);
59
60     skbVolume = (SeekBar) this.findViewById(R.id(skbVolume);
61     skbVolume.setMax(100); //音量调节的极限
62     skbVolume.setProgress(50); //设置seekbar的位置值
63     audioTrack.setStereoVolume(0.7f, 0.7f); //设置当前音量大小
64     skbVolume.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
65         1
66         @Override
67         public void onStopTrackingTouch(SeekBar seekBar) {
68             float vol = (float) (seekBar.getProgress()) / (float) (seekBar.getMax());
69             audioTrack.setStereoVolume(vol, vol); //设置音量
70         }
71
72         @Override
73         public void onStartTrackingTouch(SeekBar seekBar) {
74         }
75
76         @Override
77         public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
78         }
79     });
80     ((CheckBox) findViewById(R.id.cb_ap)).setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
81
82         @Override
83         public void onCheckedChanged(CompoundButton view, boolean checked) {
84             isProcessing = checked;
85         }
86     });
87
88     initProccesor();
89 }
90
91 @Override
92 protected void onDestroy() {
93
94     releaseProcessor();
95
96     android.os.Process.killProcess(android.os.Process.myPid());
97     super.onDestroy();
98 }
99
100 @Override
101 public void onClick(View v) {
102     if (v.getId() == R.id.btnRecord) {
103         isRecording = true;
104
105         //启动线程,开始录音和一边播放
106
107         new RecordPlayThread().start();
108
109     } else if (v.getId() == R.id.btnStop) {
110         isRecording = false;
111     }
112 }
113
```



内容举报



返回顶部

广告

```
114 class RecordPlayThread extends Thread {
115     public void run() {
116         try {
117
118             short[] buffer = new short[recBufSize/2];
119             audioRecord.startRecording();//开始录制
120             audioTrack.play();//开始播放
121
122             1 saveToFile(FLAG_RECORD_START, null);
123
124             while (isRecording) {
125                 //setp 1 从MIC保存数据到缓冲区
126                 int bufferReadResult = audioRecord.read(buffer, 0, recBufSize/2);
127                 short[] tmpBuf_src = new short[bufferReadResult];
128                 System.arraycopy(buffer, 0, tmpBuf_src, 0, bufferReadResult);
129
130                 //setp 2 进行处理
131                 if (isProcessing) {
132
133                     processData(tmpBuf_src);
134
135                 } else {
136                 }
137                 //写入数据即播放
138                 audioTrack.write(tmpBuf_src, 0, tmpBuf_src.length);
139
140                 //saveToFile(FLAG_RECORDING, tmpBuf_src);
141
142             }
143
144             saveToFile(FLAG_RECORD_FINISH, null);
145
146             audioTrack.stop();
147             audioRecord.stop();
148         } catch (Exception t) {
149             t.printStackTrace();
150         }
151     }
152 };
153
154 class RecordPlayThread2 extends Thread {
155     public void run() {
156         try {
157
158             byte[] buffer = new byte[recBufSize];
159             audioRecord.startRecording();//开始录制
160             audioTrack.play();//开始播放
161
162             saveToFile(FLAG_RECORD_START, null);
163
164             while (isRecording) {
165                 //setp 1 从MIC保存数据到缓冲区
166                 int bufferReadResult = audioRecord.read(buffer, 0, recBufSize);
167                 byte[] tmpBuf_src = new byte[bufferReadResult];
168                 System.arraycopy(buffer, 0, tmpBuf_src, 0, bufferReadResult);
169
170                 //setp 2 进行处理
```



内容举报



返回顶部

广告

```
171         if (isProcessing) {
172
173             processData(tmpBuf_src);
174
175         } else {
176         }
177         //写入数据即播放
178         audioTrack.write(tmpBuf_src, 0, tmpBuf_src.length);
179     }
180     saveToFile(FLAG_RECORDING, tmpBuf_src);
181 }
182 }
183
184 saveToFile(FLAG_RECORD_FINISH, null);
185
186 audioTrack.stop();
187 audioRecord.stop();
188 } catch (Exception t) {
189     t.printStackTrace();
190 }
191 }
192 };
193
194 /**
195  * 保存录音数据到本地wav文件
196  * @param flag
197  * @param data
198  */
199 private void saveToFile(int flag, byte[] data){
200
201     switch (flag){
202         case FLAG_RECORD_START:
203
204             String pcmPath = Environment.getExternalStorageDirectory().getAbsolutePath() + "/record/record.pcm";
205             try {
206                 mOutputStream = new FileOutputStream(pcmPath);
207             } catch (FileNotFoundException e) {
208                 e.printStackTrace();
209             }
210
211             break;
212         case FLAG_RECORDING:
213
214             if(mOutputStream != null){
215                 try {
216                     mOutputStream.write(data);
217                 } catch (IOException e) {
218                     e.printStackTrace();
219                 }
220             }
221
222             break;
223         case FLAG_RECORD_FINISH:
224
225             try {
226                 if(mOutputStream != null){
227                     mOutputStream.close();
```



内容举报



返回顶部

广告

```
228         }
229     } catch (IOException e) {
230         e.printStackTrace();
231     }
232
233     pcmPath = Environment.getExternalStorageDirectory().getAbsolutePath() + "/record/record.pcm";
234     String wavePath = Environment.getExternalStorageDirectory().getAbsolutePath() + "/record/record.wav";
235
236     1 AudioEncodeUtil.convertPcm2Wav(pcmPath, wavePath);
237
238     break;
239
240
241
242
243     /**
244     * 初始化降噪
245     */
246     private void initProcessor(){
247         mProcessor = new WebrtcProcessor();
248         mProcessor.init(frequency);
249     }
250
251     /**
252     * 释放降噪资源
253     */
254     private void releaseProcessor(){
255         if(mProcessor != null){
256             mProcessor.release();
257         }
258     }
259
260     /**
261     * 处理需要降噪的音频数据
262     * @param data
263     */
264     private void processData(byte[] data){
265         if(mProcessor != null){
266             mProcessor.processNoise(data);
267         }
268     }
269
270     /**
271     * 处理需要降噪的音频数据
272     * @param data
273     */
274     private void processData(short[] data){
275         if(mProcessor != null){
276             mProcessor.processNoise(data);
277         }
278     }
279
280 }
```



内容举报



返回顶部

以上代码主要是实现一边录音，一边播放录音的声音，类似ktv，在其中对每次获取的录音数据tmpBuf\_src交给WebrtcProcessor处理，这里可以读取为byte[]或者short[] 数据，但是交给底层webrtc处理时，都是需

要转换为short[] 数据的。然后这边采样率我采用8000，采样编码位16，单声道。



接下来看看WebrtcProcessor.java的处理：



1



内容举报



返回顶部

广告

```
1 package com.test.jni;
2
3 import android.util.Log;
4
5 /**
6  * 音频降噪处理
7  */
8 public class WebrtcProcessor {
9     1
10     static {
11         try {
12             //加载降噪库
13             System.loadLibrary("webrtc");
14         } catch (UnsatisfiedLinkError e) {
15             Log.e("TAG", "Couldn't load lib: - " + e.getMessage());
16         }
17     }
18 }
19
20 /**
21  * 处理降噪
22  * @param data
23  */
24 public void processNoise(byte[] data){
25
26     if(data == null) return;
27
28     int newDataLength = data.length/2;
29     if(data.length % 2 == 1){
30         newDataLength += 1;
31     }
32
33     //此处是将字节数据转换为short数据
34     short[] newData = new short[newDataLength];
35
36     for(int i=0; i<newDataLength; i++){
37         byte low = 0;
38         byte high = 0;
39
40         if(2*i < data.length){
41             low = data[2*i];
42         }
43         if((2*i+1) < data.length){
44             high = data[2*i+1];
45         }
46
47         newData[i] = (short) (((high << 8) & 0xff00) | (low & 0x00ff));
48     }
49
50     // 交给底层处理
51     processNoise(newData);
52
53     //处理完之后, 又将short数据转换为字节数据
54     for(int i=0; i<newDataLength; i++){
55         if(2*i < data.length){
56             data[2*i] = (byte) (newData[i] & 0xff);
```



内容举报



返回顶部



广告

```
57     }
58     if((2*i+1) < data.length){
59         data[2*i+1] = (byte) ((newData[i] >> 8) & 0xff);
60     }
61 }
62
63 }
64
65 /**
66  * 初始化降噪设置
67  * @param sampleRate 采样率
68  * @return 是否初始化成功
69  */
70 public native boolean init(int sampleRate);
71
72 /**
73  * 处理降噪
74  * @param data
75  * @return
76  */
77 public native boolean processNoise(short[] data);
78
79 /**
80  * 释放降噪资源
81  */
82 public native void release();
83
84 }
```

此处你可能需要将字节数据转换为short数据，要特别小心，如果不小心转错了，你的音频数据就乱码了，来的后果是，听到的声音基本都是沙沙声，我之前就是在这里踩了坑，底层调试了很久也没解决，后面才意识到可能上层出错了，调试之后发现是这里。正常呢，调试时看short数据时，如果它们的数值不是很大，那应该是没问题的，如果大部分都是4000以上，或者 - 4000以下的，那很可能是转换的时候出问题了，一般来说数值都是几十，几百的样子。

好了，现在看看底层大概是如何实现的：



内容举报



返回顶部

广告

```
1  #include <jni.h>
2  #include "audio_ns.h"
3  #include "noise_suppression.h"
4
5  //此处是为了里面的底层方法能被java层识别
6  extern "C" {
7      /L
8      //降噪的实例,句柄
9      NsHandle* handle = NULL;
10
11      //降噪处理
12      void innerProcess(short in_sample[], short out_sample[], int length){
13
14          inCurPosition = 0;
15
16          //此处以160为单位, 依次调用audio_ns_process处理数据,因为这个方法一次只能处理160个short音频数据
17          while(curPosition < length){
18
19              audio_ns_process((int) handle, in_sample + curPosition, out_sample + curPosition);
20
21              curPosition += 160;
22
23          }
24
25      }
26
27      JNIEXPORT jboolean JNICALL
28      Java_com_test_jni_WebRTCProcessor_init(JNIEnv *env, jobject instance, jint sample_rate) {
29
30          //初始化降噪实例
31          handle = (NsHandle *) audio_ns_init(sample_rate);
32
33          return false;
34      }
35
36      JNIEXPORT jboolean JNICALL
37      Java_com_test_jni_WebRTCProcessor_processNoise(JNIEnv *env, jobject instance, jshortArray sample) {
38
39          if(!handle)
40              return false;
41
42          //获取数据长度
43          jsize length = env->GetArrayLength(sample);
44
45          //转换为jshort数组
46          jshort *sam = env->GetShortArrayElements(sample, 0);
47
48          //将sam的数据全部复制给新的in_sample
49          short in_sample[length];
50          for(int i=0; i<length; i++){
51              in_sample[i] = sam[i];
52          }
53
54          //传入in_sample作为需要处理音频数据, 处理之后的数据返回到sam中
55          innerProcess(in_sample, sam, length);
56      }
```



内容举报







返回顶部

```
57 //将sam中的数据,再转换回sample中
58 env->ReleaseShortArrayElements(sample, sam, 0);
59
60 return true;
61 }
62
63 JNIEXPORT void JNICALL
64 Java_com_test_jni_WebRTCProcessor_release(JNIEnv *env, jobject instance) {
65     1
66     // 释放降噪资源
67     if(handle){
68         audio_ns_destroy((int) handle);
69     }
70
71
72 }
73
74 }
```

上面代码描述的比较清晰了，就是实际上webrtc降噪一次性只处理了80个short数据,在8000采样率中是这样的，意思就是说webrtc每次只能处理10毫秒，0.01秒的数据。那么依次类推，针对44100采样率的数据处理的话，每次能处理的数据长度就应该是441个short数据了，有不同采样率需求的朋友，可以自行修改测试。接下来看看webrtc的降噪是如何初始化和处理的：

[内容举报](#)[返回顶部](#)

广告

```
1  #include "audio_ns.h"
2  #include "noise_suppression.h"
3
4  #include <stdio.h>
5
6  int audio_ns_init(int sample_rate){
7      
8      NsHandle* NS_instance;
9      int ret;
10      //创建WebRtcNs实例
11     if ((ret = WebRtcNs_Create(&NS_instance)) ) {
12          printf("WebRtcNs_Create failed with error code = %d", ret);
13     }
14     
15
16     //初始化WebRtcNs实例,此处需要指定采样,告诉它一次可以处理多少个short音频数据,
17     //如果是8000, 则一次可以处理80,如果是44100, 则一次可以处理441个
18     //也就是说,一次性可以处理10ms时间的数据
19     if ((ret = WebRtcNs_Init(NS_instance, sample_rate)) ) {
20         printf("WebRtcNs_Init failed with error code = %d", ret);
21         return ret;
22     }
23
24     //设置降噪的力度,0,1,2, 0最弱,2最强
25     if ( ( ret = WebRtcNs_set_policy(NS_instance, 2) ) ){
26         printf("WebRtcNs_set_policy failed with error code = %d", ret);
27         return ret;
28     }
29
30     return (int)NS_instance;
31 }
32
33
34 int audio_ns_process(int ns_handle , short *src_audio_data ,short *dest_audio_data){
35     //get handle
36     NsHandle* NS_instance = (NsHandle* )ns_handle;
37
38     //noise suppression
39     if(
40         //此处这么做,是因为,真正的WebRtcNs_Process,一次只能处理80个shorts音频数据
41         WebRtcNs_Process(NS_instance ,src_audio_data ,NULL ,dest_audio_data , NULL) ||
42         WebRtcNs_Process(NS_instance ,&src_audio_data[80] ,NULL ,&dest_audio_data[80] , NULL) ){
43         printf("WebRtcNs_Process failed with error code = " );
44         return -1;
45     }
46
47     return 0;
48 }
49
50
51 void audio_ns_destroy(int ns_handle){
52     //释放WebRtcNs资源
53     WebRtcNs_Free((NsHandle *) ns_handle);
54 }
```

以上是调用真正的webrtc代码处理降噪了，注释也比较详细，大家自己看。



内容举报



返回顶部

那么真正底层的webrtc处理降噪是怎么样的呢？这个，呵呵，我觉得吧，浅尝则止，这个不是一般能看懂的，我是看不懂，核心大部分是算法，如果不熟悉降噪的算法和各个数据的意义的话，那看着简直是看天书啊。当然啦，大家想看的或者想要源码进行测试的话，我后面会提供项目源码下载的。

广告

你以为这样就完了吗？那好像还不够丰富啊，因此还有一点我想分享给大家的，就是音频处理中，还有最开始我所说过的各种音频处理，绝不仅仅只有降噪，在当前网上开放的android音频处理项目源码如此稀缺的环境中（我想说，真是百度了好久的android音频处理，却找不出几个可以运行测试的android项目源码，实在香菇），该如何进行其它的音频处理呢。幸运的是，webrtc这个项目里，提供了很多音频处理的模块，大家可以去网上把它下载下来，找到对应的模块，比如增益，在webrtc/modules/audio\_processing/agc目录下，把里面的文件拷到自己项目中编译，当然可能还会设计到其它目录的文件，找到拷过来，后面应该就可以编译了。至于怎么编译，找到我项目中的CMakeList.txt文件，依葫芦画瓢，替换修改就是了。

可是好像还有一个问题，那就是编译之后我该怎么用啊？？这个，才是重点啊！是啊，我当时也是一头雾水。好吧，本着助人为乐的精神（嘿嘿），我就传授一个从不外传的绝技吧（好像好高级，好期待啊），那就是搜索github（程序员都该知道的超牛逼网站），比如我想找webrtc降噪，那么我就搜WebRtcNs\_Process，找到一些合适的项目，看人家是如何调用实现的，这样就可以实现啦。

好了，就说到这里，等着吃饭了。下面是项目下载地址。

http://download.csdn.net/detail/hesong1120/9687830 (http://download.csdn.net/detail/hesong1120/9687830)

版权声明：本文为博主原创文章，未经博主允许不得转载。



hesong1120 (/hesong1120) 2017-10-19 22:06

5楼

我记得好像webrtc只支持采样率8000，16000，32000的降噪，不支持44100，所以你要么设置这3种采样率，要么转换之后降噪。

回复

wanhun7547 (/wanhun7547) 2017-10-18 11:20

4楼

请问怎么降噪，需要改C文件重新打包吗

回复

walid1992 (/walid1992) 2017-09-18 17:56

3楼

没声音啊 尴尬

回复

内容举报


返回顶部

查看 5 条热评

相关文章推荐

## 基于Linux webRTC 音语对讲之一——获取代码及编译 (<http://blog.csdn.net/peixiuhui/>...


由于一个音语对讲项目需要使用linux webRTC 在ARM上实现实时对讲，所以简单的记录下开发过程及碰到的问题。1. 获取webRTC代码，开始参考了很多网页，但是都不行，也许曾经可以...

 peixiuhui (<http://blog.csdn.net/peixiuhui/>) 2015年07月03日 20:00 1395



## madplay的使用方法 (<http://blog.csdn.net/u012951123/article/details/17067023>)


管理madplay的主程序，包括播放，暂停播放，恢复播放，停止播放 system("madplay north.mp3 &");//利用system函数调用madplay播放器播放\*.mp3音乐 s...

 u012951123 (<http://blog.csdn.net/u012951123>) 2013年12月02日 09:12 1775




## 使用webRtc进行音频降噪（NS）和VAD检测 ([http://blog.csdn.net/qg\\_30948113/article/...](http://blog.csdn.net/qg_30948113/article/...)

webRtc整个项目在windows下编译还是很难搞定的一件事，本人是下载别人已经编译好的工程进行开发的，整个工程有200多个项目，音频降噪和VAD检测只是其中的2个项目。一、音频降噪 ...

 qg\_30948113 ([http://blog.csdn.net/qg\\_30948113](http://blog.csdn.net/qg_30948113)) 2017年03月31日 19:22 922

## webrtc--AudioProcessing-- 音频降噪的处理过程 (<http://blog.csdn.net/huozaisinian/artic...>

1.AudioProcessing的实例化和配置：AudioProcessing\* apm = AudioProcessing::Create(0); apm->level\_estima...

 huozaisinian (<http://blog.csdn.net/huozaisinian>) 2016年11月24日 17:49 486

## 单独编译和使用webrtc音频降噪模块（NS） (<http://blog.csdn.net/godloveyuxu/article/d...>

单独编译和使用webrtc音频增益模块(附完整源码+测试音频文件) 单独编译和使用webrtc音频回声消除模块(附完整源码+测试音频文件) webrtc的音频处理模块分为降噪ns，回音消除aec，回声...

 godloveyuxu (<http://blog.csdn.net/godloveyuxu>) 2017年06月29日 15:51 675



### android 利用 speex 音频降噪，回声消除demo (<http://download.csdn.n...>

[http://download.csdn.net/detail/jinzeyu\\_36524243/9...](http://download.csdn.net/detail/jinzeyu_36524243/9...) 2017年11月29日 17:54 1.03MB 下载()



### webrtc降噪和增益的部分代码 ([http://download.csdn.net/detail/jinzeyu\\_...](http://download.csdn.net/detail/jinzeyu_...)

[http://download.csdn.net/detail/jinzeyu\\_36524243/9...](http://download.csdn.net/detail/jinzeyu_36524243/9...) 2017年07月10日 23:08 92KB 下载()

## EDIUS是怎么给音频降噪的 (<http://blog.csdn.net/shipinbianji/article/details/52846914>)

如果我们的录音环境比较嘈杂，就会导致我们录的音频中有杂音。如果听的时候杂音非常小可以忽略掉那就最好不过了，但是如果杂音的声音很大有的甚至高出了我们的主题声音，那就是问题了。在时间和工作量允许的情况下，...

 shipinbianji (<http://blog.csdn.net/shipinbianji>) 2016年10月18日 10:53 281



内容举报



返回顶部



### 视频降噪处理,浙大数字视音频技术 (<http://download.csdn.net/detail/u0...>

<http://download.csdn.net/detail/u012951123/17067023> 2013年06月09日 16:27 744KB 下载()



### Matlab实现音频降噪 ([http://download.csdn.net/detail/baidu\\_36524243/9...](http://download.csdn.net/detail/baidu_36524243/9...)



2016年10月26日 21:26    70KB    [下载\(\)](#)

[/http://download.csdn.net/detail/yazhouren/5111111/](#)

### 【单独编译使用WebRTC的音频处理模块 - android】(http://blog.csdn.net/yazhouren/article/details/5111111)


http://billhoo.blog.51cto.com/2337751/1213801 更新    【2014年5月14日】    昨天有幸在 Google 论坛里询问到 AECM 模块的延迟问题...

 yazhouren (http://blog.csdn.net/yazhouren)    2014年05月27日 17:20    1099



2013年05月18日 10:33    667KB    [下载\(\)](#)

[/http://download.csdn.net/detail/qianwenerrui/5111111/](#)



2017年11月14日 17:09    83KB    [下载\(\)](#)

[/http://download.csdn.net/detail/u012555555/5111111/](#)


### 单独编译使用WebRTC的音频处理模块 - android (http://blog.csdn.net/chinabinlang/article/details/5111111)

单独编译使用WebRTC的音频处理模块 - android 原创作品，允许转载，转载时请务必以超链接形式标明文章 原始出处、作者信息和本声明。否则将追究法律责任。http://billhoo....

 chinabinlang (http://blog.csdn.net/chinabinlang)    2014年12月11日 15:20    12213

### 单独编译使用WebRTC的音频处理模块 - android (http://blog.csdn.net/wd\_cloud/article/details/5111111)

前言    WebRTC是时下比较热门的新技术，由于bill接触时间尚短，对该项目的理解和认知定存在不足甚或偏差，文中有描述不当之处还望各位悉心指出，感激不尽。前言 ...

 wd\_cloud (http://blog.csdn.net/wd\_cloud)    2014年11月27日 19:40    854

### 单独编译使用WebRTC的音频处理模块 - android (http://blog.csdn.net/wd\_cloud/article/details/5111111)

更新【2015年2月15日】    Bill 这段时间没有再关注 WebRTC 以及音频处理的相关信息，且我个人早已不再推荐单独编译 WebRTC 中的各个模块出来使用。实际上本文的参考价...

 wd\_cloud (http://blog.csdn.net/wd\_cloud)    2016年02月03日 17:05    815

### Android 系统 '七夕'巨献 VIVO Xplay 基于ViVo官方稳定内核,完美root,适度美化,降噪点,...

ROM版本 VIVO-Xplay-PD2.13.2 ROM作者大盛 http://weibo.com/DaShengdd Android版本...

 u011145350 (http://blog.csdn.net/u011145350)    2013年08月27日 14:40    1685



2017年11月29日 17:36    3.52MB    [下载\(\)](#)


[/http://download.csdn.net/detail/dxpxxb/5111111/](#)




2017年05月27日 17:41    516KB    [下载\(\)](#)

[/http://download.csdn.net/detail/ab7936573/9111111/](#)


### WebRTC源码分析：音频模块结构分析 (http://blog.csdn.net/dxpxxb/article/details/5111111)

 内容举报

 返回顶部

一、概要介绍WebRTC的音频处理流程，见下图： webRTC将音频会话抽象为一个通道Channel，譬如A与B进行音频通话，则A需要建立一个Channel与B进行音频数据传输。上图中有三个C...

广告

 dxpqxb (<http://blog.csdn.net/dxpqxb>) 2016年06月22日 16:53 1254

