אוניברסיטת חיפה-החוג למדעי המחשב



פרוייקט שנתי-Adaptive Noise Cancellation

מגישים:

גדעון אפל – 034733949

גלעד דרמון– 046248357

**תוכן עניינים:**

מטרת הפרוייקט 3

דרישות הפרוייקט 3

מחקר ראשוני 3

AudioRecord 4

AudioTrack 4

MediaRecorder 5

מסקנות מחקר ראשוני 8

המשך מחקר 8

SIP 8

RTP 10

Multithreading 11

תהליך פיתוח האלגוריתם 11

מושגי יסוד בעיבוד אותות 12

התמרת פורייה בדידה 13

דציבל 15

האלגוריתם 16

מעבר ל real-time 18

שיפורים עתידיים 19

מדריך למתכנת 20

בדיקות 30

מדריך למשתמש 31

תודות 36

**מטרת הפרוייקט:**

מטרת הפרוייקט, כפי שהוגדרה ע"י מנחה הפרוייקט הינה בניית תוכנה שתהיה open source למכשירים מבוססי אנדרואיד ,שתרוץ בזמן אמת(real-time) על שיחת טלפון ותסנן את רעשי הרקע הקיימים כך שהצד האחר בשיחה יישמע את הקול בצורה צלולה ובהירה יותר ללא רעשי רקע חזקים כגון:מוזיקהרועשת,רכבת שעוברת ,ורעשי רקע אחרים.

**דרישות הפרוייקט:**

1. תוכנה שתהיה open source.
2. סינון רעשי רקע בשיחה.
3. אפשרות הקלטת שיחה ללא רעשי הרקע.
4. תמיכה במכשירים מבוססי אנדרואיד.
5. מדריך למתכנת.
6. מדריך למשתמש.

**מחקר ראשוני:**

בתחילת הפרוייקט נתבקשנו לבדוק האם ניתן להקליט שיחה וזאת מכיוון שבמידה וניתן להקליט שיחה אנו יכולים לגשת לבאפר של השיחה וכך ניתן לעשות על באפר השיחה מניפולציות שונות כולל סינון רעשים.

כאשר בדקנו את האופצייה להקלטת שיחה במכשירי אנדרויד ,גילינו שיש 3 מחלקות התומכות בהקלטה ובהשמעת ההקלטה.

המחלקות הללו הן:

1. AudioRecord
2. AudioTrack
3. MediaRecorder

**AudioRecord**[[1]](#footnote-1)

להלן תיאור המחלקה:

The AudioRecord class manages the audio resources for Java applications to record audio from the audio input hardware of the platform. This is achieved by "pulling" (reading) the data from the AudioRecord object. The application is responsible for polling the AudioRecord object in time using one of the following three methods: [**read(byte[], int, int)**](http://developer.android.com/reference/android/media/AudioRecord.html#read(byte[], int, int)), [**read(short[], int, int)**](http://developer.android.com/reference/android/media/AudioRecord.html#read(short[], int, int)) or [**read(ByteBuffer, int)**](http://developer.android.com/reference/android/media/AudioRecord.html#read(java.nio.ByteBuffer, int)). The choice of which method to use will be based on the audio data storage format that is the most convenient for the user of AudioRecord.

Upon creation, an AudioRecord object initializes its associated audio buffer that it will fill with the new audio data. The size of this buffer, specified during the construction, determines how long an AudioRecord can record before "over-running" data that has not been read yet. Data should be read from the audio hardware in chunks of sizes inferior to the total recording buffer size.

**AudioTrack**[[2]](#footnote-2)

להלן תיאור המחלקה:

The AudioTrack class manages and plays a single audio resource for Java applications. It allows streaming PCM audio buffers to the audio hardware for playback. This is achieved by "pushing" the data to the AudioTrack object using one of the [**write(byte[], int, int)**](http://developer.android.com/reference/android/media/AudioTrack.html#write(byte[], int, int)) and [**write(short[], int, int)**](http://developer.android.com/reference/android/media/AudioTrack.html#write(short[], int, int)) methods.

An AudioTrack instance can operate under two modes: static or streaming.  
In Streaming mode, the application writes a continuous stream of data to the AudioTrack, using one of the **write()** methods. These are blocking and return when the data has been transferred from the Java layer to the native layer and queued for playback. The streaming mode is most useful when playing blocks of audio data that for instance are:

* too big to fit in memory because of the duration of the sound to play,
* too big to fit in memory because of the characteristics of the audio data (high sampling rate, bits per sample ...)
* received or generated while previously queued audio is playing.

The static mode should be chosen when dealing with short sounds that fit in memory and that need to be played with the smallest latency possible. The static mode will therefore be preferred for UI and game sounds that are played often, and with the smallest overhead possible.

Upon creation, an AudioTrack object initializes its associated audio buffer. The size of this buffer, specified during the construction, determines how long an AudioTrack can play before running out of data.  
For an AudioTrack using the static mode, this size is the maximum size of the sound that can be played from it.  
For the streaming mode, data will be written to the hardware in chunks of sizes less than or equal to the total buffer size.

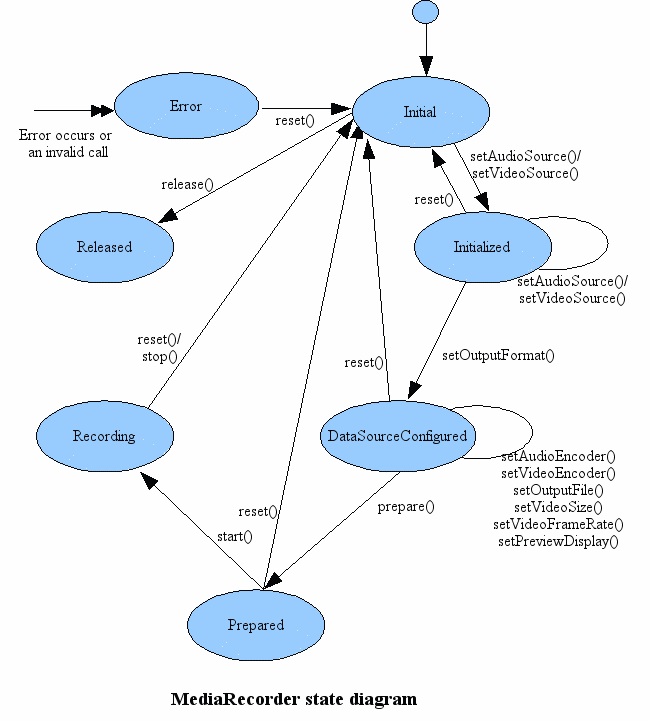
במחלקות אלו השתמשנו והצלחנו לבצע הקלטה של שיחה רגילה מהמיקרופון ללא בעייה וללא התערבות כלשהי מצידנו בשיחה(כלומר ללא הפעלת אלגוריתם כלשהו).

לאחר שגילינו שניתן להקליט שיחה החל שלב המחקר השני שלנו והוא כיצד אנו מתערבים בשיחה ומשנים את הפלט,כלומר מסננים את רעשי הרקע הקיימים.

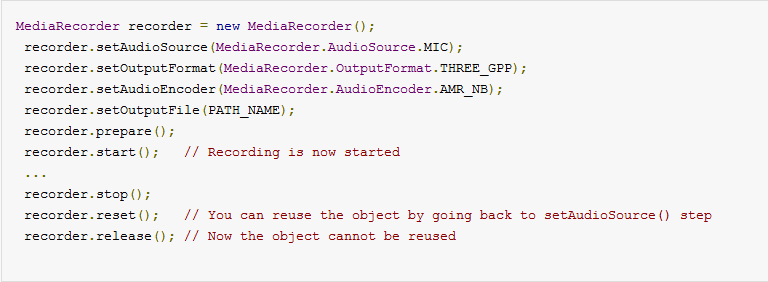
חיפשנו מחלקה שתתמוך בשתי האפשרויות שקיימות עבור שיחה שהן:הוצאת שיחה וקבלת שיחה ומצאנו מחלקה בשם MediaRecorder שהיא מגדירה את כל הפרמטרים של ההקלטה.

**MediaRecorder**[[3]](#footnote-3)

Used to record audio and video. The recording control is based on a simple state machine (see below).

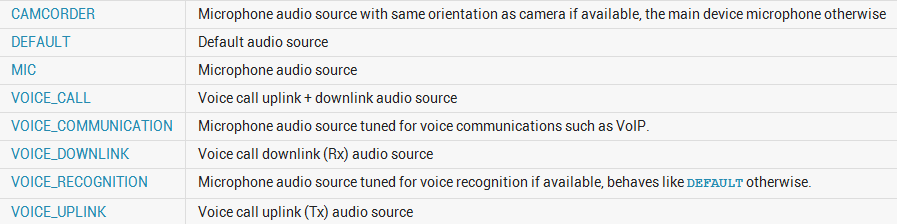


A common case of using MediaRecorder to record audio works as follows:



Applications may want to register for informational and error events in order to be informed of some internal update and possible runtime errors during recording. Registration for such events is done by setting the appropriate listeners (via calls (to [**setOnInfoListener(OnInfoListener)**](http://developer.android.com/reference/android/media/MediaRecorder.html#setOnInfoListener(android.media.MediaRecorder.OnInfoListener))setOnInfoListener and/or[**setOnErrorListener(OnErrorListener)**](http://developer.android.com/reference/android/media/MediaRecorder.html#setOnErrorListener(android.media.MediaRecorder.OnErrorListener))setOnErrorListener). In order to receive the respective callback associated with these listeners, applications are required to create MediaRecorder objects on threads with a Looper running (the main UI thread by default already has a Looper running).

ל AudioSource יש מספר אפשרויות לבחור את הקלט ממנו הוא מקליט והן:



לאחר מכן חיפשנו כיצד אנו משנים את הבאפר עצמו של השיחה,כלומר מבצעים איזושהי השהייה קטנה מאוד כך שאנו נעביר בשיחה את הפלט ללא רעשי הרקע שהיו בקלט.

לאחר חיפוש רחב באינטרנט וללא תשובה פנינו למקומות אחרים שהם:

1. google hangout- קבוצה שקיימת ב google+ ומטרתה לעזור למפתחים לפתור בעיות העולות להם בזמן הפיתוח.
2. 2 קבוצות בפייסבוק:
   * Android Developers
   * Android Developers Israel

לאחר שיחה ב google hangout קיבלנו תשובה מהמפתחים של google שהם לא מכירים אפשרות של התערבות ממשית בשיחה קיימת וזאת משתי סיבות:

1. חוקית – הקלטה של שיחה ללא ידיעת שני הצדדים היא אסורה בהרבה מדינות בעולם. בישראל למשל החוק מאשר להקליט שיחה כל עוד הצד המקליט הוא אחד מהצדדים המשתתפים בשיחה. ישנו פתרון לכך שהרבה תוכנות מאפשרות צליל צפצוף כל כמה שניות על מנת לזהות שהשיחה אכן מוקלטת.

לא הסתפקנו בתשובה זו והמשכנו לחקור ולאחר מספר שאלות בקבוצה בפייסבוק הגענו ל:

pallabsarkar - Lead Engineer at [Samsung India Software Operations Pvt Ltd](https://www.facebook.com/Samsung.SISO)

בשיחה איתו הבנו שאין אפשרות לבצע את הפרוייקט במתכונת הנוכחית מכיוון שהחומרה אינה מאפשרת זאת,וזאת מכיוון שיש בעיות משפטיות שיכולות להתעורר במצבים כאלו,לדוג' שיחה שאני מבצע עם איזשהו גורם וכאשר ישנה התערבות בשיחה אני יכול לשנות ממש את כל התוכן ולכן מנעו זאת חומרתית.

**מסקנות מהמחקר הראשוני:**

לאחר שהגענו למסקנה הזו נקבעה פגישה עם מנחה הפרוייקט מר חננאל חזן והצגנו את הממצאים שיש בידינו ובנוסף את מה שעשינו עד לאותו הרגע מבחינת האפליקציה והעלנו את האפשרות לבצע את הפרוייקט על בסיס שיחות אינטרנטיות כמו תוכנת SKYPE ושם לבצע את האלגוריתם שלנו.

**המשך מחקר:**

מצאנו שקיים פרוטוקול בשם Session Initiation Protocol – SIP המאפשר לנו לבצע את הפרוייקט במתכונת החדשה.

התחלנו בחיפוש אחר תוכנות open source המאפשרות לבצע שיחות ע"י SIP ומצאנו מספר תוכנות:

1. Csipsimple
2. Sipme
3. Sipdroid

כאשר התחלנו לקרוא ולחקור את שתי התוכנות הראשונות ורצינו להתיקן אותם גילינו שהם דורשות קומפילציה ב Linux ולכן המשכנו בחיפוש אחר תוכנות נוספות עד שמצאנו את התוכנה sipdroid.

**:Sipdroid** יישום עבור שיחות over IP עבור מערכת ההפעלה אנדרואיד שמשתמשת בSIP.

**מהו SIP(**[**Session Initiation Protocol**](http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CG0QFjAA&url=http%3A%2F%2Fen.wikipedia.org%2Fwiki%2FSession_Initiation_Protocol&ei=kLUPUMeIPIbUsgaO7IDgAQ&usg=AFQjCNHEPYhtrKJx8SSnBBAHlS7Enu-nSA&sig2=cf1v_gGXy6jcM2_r7-p59w)**):**[[4]](#footnote-4)

זהו פרוטוקול איתות(signaling) בשימוש נרחב לשליטה עלSESSION של תקשורת כגון שיחות קול או וידאו על גבי IP,הפרוטוקול יכול לשמש ליצירה,שינוי וסיום של שיחות unicast או multicast.כל session יכול להיות מורכב מכמה זרמים של מדיה(video conference- מורכב מזרם עבור הקול וזרם עבור המדיה אך קשור לאותו ה session).  
פרוטוקול זה ,הוא פרוטוקול בשכבת היישום(application) שנועד להיות עצמאי בשכבת התעבורה הבסיסית. פרוטוקול זה יכול לעבוד מעל:

1. TCP-Transmission Control Protocol
2. UDP- User Datatgram Protocol
3. SCTP- Stream Control Transmission Protocol

זהו פרוטוקול המבוסס על טקסט המשלב אלמנטים רבים של HTTP ו- SMTP.

SIP מזכיר אלמנטים דומים למודל של HTTP request/response,כל טרנזקציה מורכבת מבקשת לקוח שגורמת להפעלה של פעולה מסויימת על השרת ולפחות תגובה אחת,SIP עושה שימוש ברוב   
שדות הכותרת (header) ,מקודדת חוקים וכללים של בקשות HTTP,ומעבירה בפורמט קריא מבוסס טקסט.

כל משאב ברשת SIP מזוהה ע"י מזהה משאבים ייחודי(URI-uniform resource identifier) המבוסס על תחביר סטנדרטי כללי הנמצא בשימוש גם בשרותי האינטרנט והדואר האלקטרוני. ה URI המשמש את SIP הוא sip:.

SIP עובד בתיאום עם מספר פרוטוקולים אחרים לקוחות SIP בד"כ משתמשים ב TCP או UDP בport 5060 או 5061 כדי להתחבר לשרת הSIP וליחידות קצה שונות של SIP.port 5060 הוא הנפוץ יותר לשימוש לא מוצפן בSIP בעוד 5061 הוא הנפוץ לשימוש מוצפן,ומשתמשת ב TLS-Transport Layer Security.

SIP משמש בעיקר בהקמת וסגירת קשרים של שיחות קוליות ווידאו.בנוסףSIP מאפשר גם התערבות בשיחות קיימות.ההתערבות יכולה לכלול שינוי של כתובות או יציאות(ports),הזמנת שותפים נוספים,והוספה או מחיקה של זרמי מדיה רלוונטיים/לא.

SIPקשורה ל IETF- Internet Engineering Task Force ומגדירה התנהגות עבור יישומי הודעות מיידיות.  
תקשורת של קול ווידאו ביישומים מבוססי SIP נישאות על ידי פרוטוקול אחר פרוטוקול RTP- Real-time Transport Protocol. הפרמטרים עבור זרמי מדיה אלו מוגדרים ע"י SDP- Session Description Protocol אשר מועבר בגוף חבילת SIP.

המניע העיקרי עבור SIP הייתה לספק פרוטוקול המאפשר לבצע תקשורת מבוססת IP שיכול לתמוך בפונקציות של עיבוד שיחות ובתכונות נוספות.SIP בפני עצמו אינו מגדיר את התכונות הללו,אלא תפקידו הוא לפתוח חיבור וכל שאר הדברים(חיוג מספר,צלצולהטלפון,שמיעת צליל חיוג/תפוס) מבוצעות על ידי שרתי proxy ו user agents. הטרמינולוגיה והמימוש בעולם הSIP הם שונים אך למשתמש הקצה הם אותו הדבר.

SIP הוא פרוטוקול peer-to-peer ובכך הוא דורש רשת פשוטה עם מודיעין המופץ ברשת,מוטמע בנקודות הקצה.תכונות הSIP ממומשות ביחידות הקצה .

**מהו RTP(**[**Real-Time Transfer Protocol**](http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CG0QFjAA&url=http%3A%2F%2Fen.wikipedia.org%2Fwiki%2FSession_Initiation_Protocol&ei=kLUPUMeIPIbUsgaO7IDgAQ&usg=AFQjCNHEPYhtrKJx8SSnBBAHlS7Enu-nSA&sig2=cf1v_gGXy6jcM2_r7-p59w)**):**[[5]](#footnote-5)

פרוטוקול זה מגדיר פורמט סטנדרטי עבור חבילת מידע של וידאו או קול על גבי רשתות IP. RTP עושה שימוש נרחב במערכות תקשורת ובידור הכוללים הזרמת מדיה,כגוןטלפוניה,יישומיוידאו,יישומיועידה,שירותי טלוויזיה ושיחות ווקי טוקי(push-to-talk).

RTP משתמשת יחד עם פרוטוקול נוסף RTCP- RTP Control על מנת לבצע בקרה,בעודRTP נושאת את המידע RTCP משמש כדי לפקח על איכות השידור והשירות(QoS) וסנכרון של מספר זרמים.  
RTP מיוצר ומתקבל ע"י כל מספר יציאה(port) וקשור ל RTCP שמתקבל ע"י היציאה(port) הבאה אחרי היציאה(port) הגבוהה ביותר.  
RTP הוא אחד מהיסודות של VOIP- Voic Over IP,ובהקשר זה משמש לעיתים קרובות עם פרוטוקול איתות המסייע בהקמת הקשרים ברשת.

RTP הינו פרוטוקול המאפשר להעביר מידע ומידע בקרה,מקצה לקצה בהקשר של שידורים הדורשים Qos,לכן זהו פרוטוקול הנמצא ברמת ה Transport. RTP מספק פיצוי עבור מצבים בהם הרשת "עצבנית" או מלאה דבר שנפוץ בהעברת מידע על גבי רשת IP.RTP תומך בהעברת מידע עבור מספר יעדים ע"י IP multicast.RTP נחשב לסטנדרט העיקרי להעברת מידע של קול/וידאו על גבי רשתות IP.

יישומים הדורשים זרימת מולטימדיה בזמן אמת דורשים שתהיה אספקה של חבילות כל הזמן אך הם יכולים לספוג אובדן של חבילות ע"מ להשיג מטרה זו,TCP שגם מותאם לשימוש עםRTP אינו בשימוש בד"כ מכיוון שTCP מעדיף אמינות על פני זמן/עיתוי,לכן רוב מימושי RTP בנויים על UDP.פרוטוקולים נוספים אשר נבנו במיוחד עבוד העברת מולטימדיה הם SCTP ו DCCP אף על פי שהם לא נמצאים כיום בשימוש נרחב.

**מרכיבי הפרוטוקול:**

RTP מתאר שני תתי פרוטוקולים:

1. פרוטוקול העברת הנתונים,RTP,שעוסק בהעברת הנתונים בזמן אמת,נתונים המועברים ע"י פרוטוקול זה כוללים חותמון זמן לסנכרון החבילות,מספרי רצף(על מנת לזהות אובדן חבילות וזיהוי ואז ד=סידור מחדש),ואת סוג קידוד הנתונים.
2. פרוטוקול הבקרה,RTCP,משמש כדי לציין את איכות המשוב(Qos),שירות סנכרון בין זרמי המידע.  
   אופציה נוספת הוא שיתוף עם פרוטוקול איתות (signaling) כלשהו כדוגמת SIP.

RTP Session מוקם עבור כל זרם מולטימדיה,session כולל כתובת IP עם זוג יציאות(ports) עבור RTP ו RTCP.לדוגמא:זרם של קול וזרם של וידאו יקבלו כ"א מהם sessionשונה,מה שמאפשר למקבל להחליט אם הוא מוותר על זרם מסויים.היציאות שיש לsession הם מגיעים ע"י פרוטוקולים אחרים כדוגמת SIP.  
לפי תיאור מרכיבי הפרוטוקול היציאות של RTP ו RTCP צריכות להיות עוקבות כך ששל ה RTCP היא הגבוהה יותר,בנוסף פרוטוקולים אלו בד"כ משתמשים ביציאות מסוג UDP אבל יכולים להשתמש גם ביציאות מסוג שונה.  
עיצוב הפרוטוקול מאפשר לו להיות עצמאי בנושא העברת המידע.

**Multithreading:**

Sip משתמש ב threads שונים ע"מ לספק לנו את חווית השיחה הטובה ביותר.  
ה threads הקיימים במערכת הם:

1. Thread עבור החבילות שאנו שולחים אל הצד השני(אמצעי קלט) –class RtpStreamSender
2. Thread עבור החבילות שאנו מקבלים מהצד השני(האינטרנט) –class RtpStreamReciever

בנוסף קיים גם ה thread הראשי של התוכנית שמריץ לנו את ה GUI של האנדרויד.  
השימוש ב multithreading חשוב באפליקציה שלנו מכיוון שישנה אפשרות שיהיו חישובים ארוכים מאוד לסינון רעשי הרקע ובמידה ולא היינו משתמשים בשיטה זו האפליקציה שלנו הייתה נתקעת.

**תהליך פיתוח האלגוריתם:**

1. קראנו את המאמר המצורף[[6]](#footnote-6) מן המנחה על Spectral Subtraction של boll מ 1979.–לצרף מאמר.
2. לאחר שהפנמנו את המושגים הבסיסים של עיבוד אותות פנינו אל תוכנת Matlab, אשר עבורה מצאנו מימוש ראשוני של האלגוריתם. יש לציין שזוהי הדרך המומלצת לעבודה עבור מתכנתים חדשים ב DSP . מכיוון שMatlab מממשת הרבה פונקציות כמו hamming() ועוד שהן ייעודיות לעיבוד אותות ומבצעת חישובים מורכבים מאוד במהירות מפני שהפונקציות ממומשות כבר.
3. <http://dea.brunel.ac.uk/cmsp/Home_esfandiar/Sample%20Wave%20Files.htm>אתר זה מכיל את המימוש של האלגוריתם של Boll עבור Matlab כולל דגימת קול והשוואה בין לפני ואחרי הSpectral Subtraction
4. האלגוריתם שלנו יהווה בסיס עיבוד real-time עבור שאר האלגוריתמים אם מתכנת ירצה יוכל לשפר את האלגוריתם שלנו ע"י שינויים מינוריים בקוד ולתכנת את שאר האלגוריתמים של Spectral Subtraction.
5. לאחר שהבנו היטב את המימוש של האלגוריתם בתוכנת Matlab , בנינו אלגוריתם זהה לזה של Matlab עבור Java. ראה קובץ מצורף[[7]](#footnote-7). ואכן הצלחנו לקבל את אותן התוצאות.
6. Real-time- מכיוון שהאלגוריתם עובד על עיבוד בזמן אמת, אין אנו יכולים לפעול באותה הדרך בה פועל האלגוריתם של Matlab כלומר לעבור על הקובץ כולו. פרוטוקול RTP בונה חבילות RTP packets בגודל 512 דגימות ושולח אותן חבילה אחת בכל פעם. כלומר הבנו שאנו צריכים להתאים את מבנה האלגוריתם שלנו לדרך עבודה זאת.
7. לטובת העיבוד המהיר יותר עברנו להשתמש בסיפרייתJtransforms - <https://sites.google.com/site/piotrwendykier/software/jtransforms>

אשר מציגה benchmark טובים מאוד עבור עיבוד FFT ו IFFT.כלומר מיועדת לעיבוד Real-Time.

**מושגי יסוד בעיבוד אותות (DSP – Digital Signal Processing):**

:segmentation תהליך של חלוקת קטע הקול לחלונות עם overlapping.

Overlapping: חפיפה בגודל מוגדר בין חלונות של קטע קול מסויים.

Phase:משמש לזיהוי הצלילים השונים ואת הכיוונים מהם הגיעו.עוזר לנו לדעת מה הכיוון   
 שממנו נכנס גל הקול אל תוך אמצעי הקלט שלנו.

Magnitude: גודל כל אובייקט, בדגימות שלנו כל אובייקט הוא מס' מרוכב ולכן הגודל הוא:

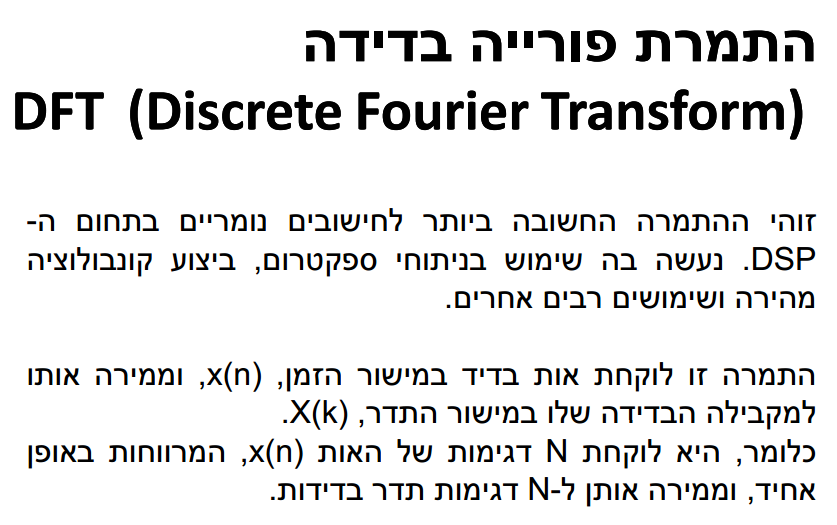
:Amplitude ההפרש בין הגבול העליון והתחתון של ה magnitude בקו מאונך ביניהם.

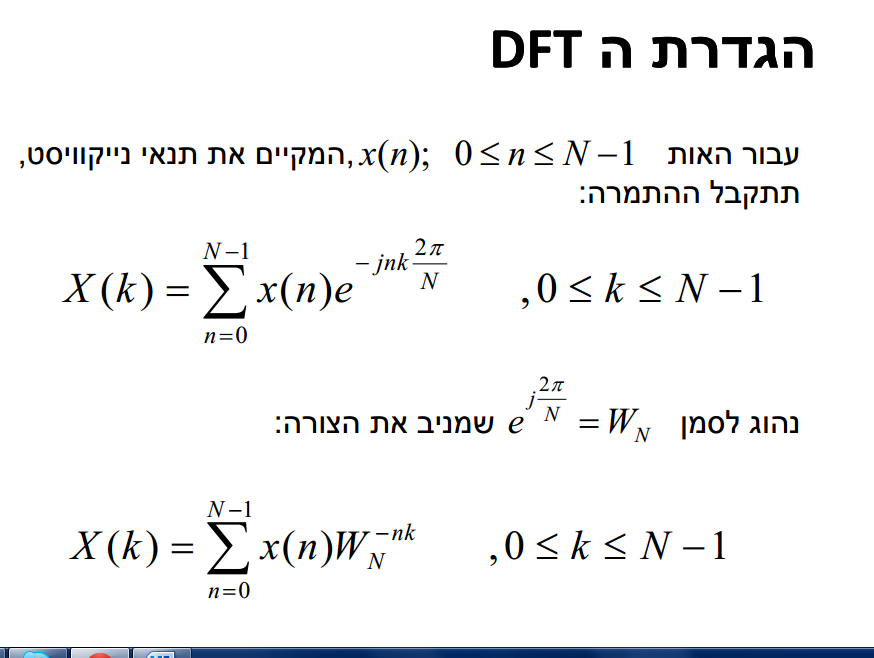
:Spectogram תצוגה של התדר על ציר הזמן,מיוצג ע"י העוצמה של התדר בנק' זמן.

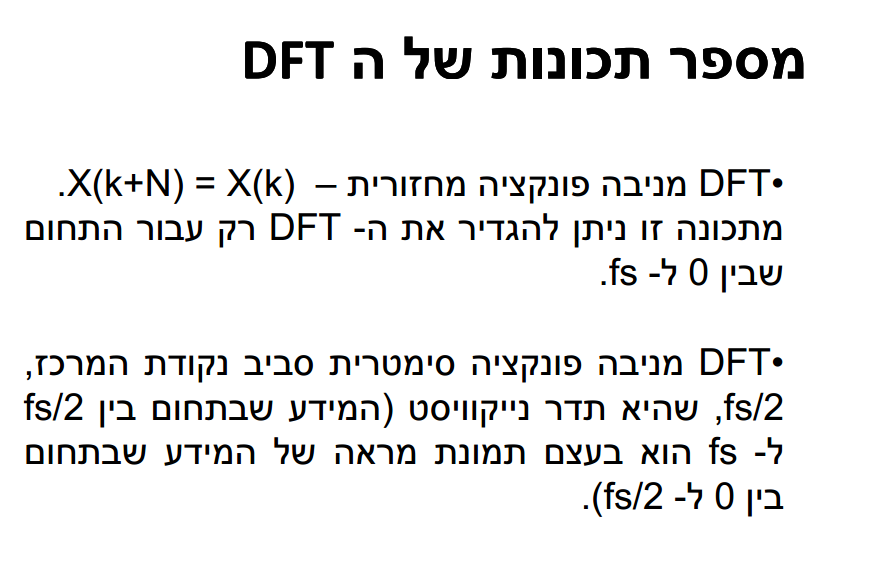
Window Function:בעיבוד אותות Window Function היא פונקציה מתמטית שמוגדרת כ-'0'   
 מחוץ לאינטרוול מסויים שנבחר.לדוגמא,הפונקציה הקובעת כי הוא קבוע  
 בתוך האינטרוול ו-0 מחוצה לו נקראת- 'rectangular window', המתארת  
 את הייצוג הגרפי שלה. כאשר מכפילים פונקציה או אות נוסף בWindow   
 Function גם כאן כל מה שמחוץ לאינטרוול המוגדר הוא יהיה '0' ולכן כל  
 מה שיישאר הוא החלקים החופפים. יישומים של Window Function  
 כוללים ניתוח ספקטרלי וסינון.

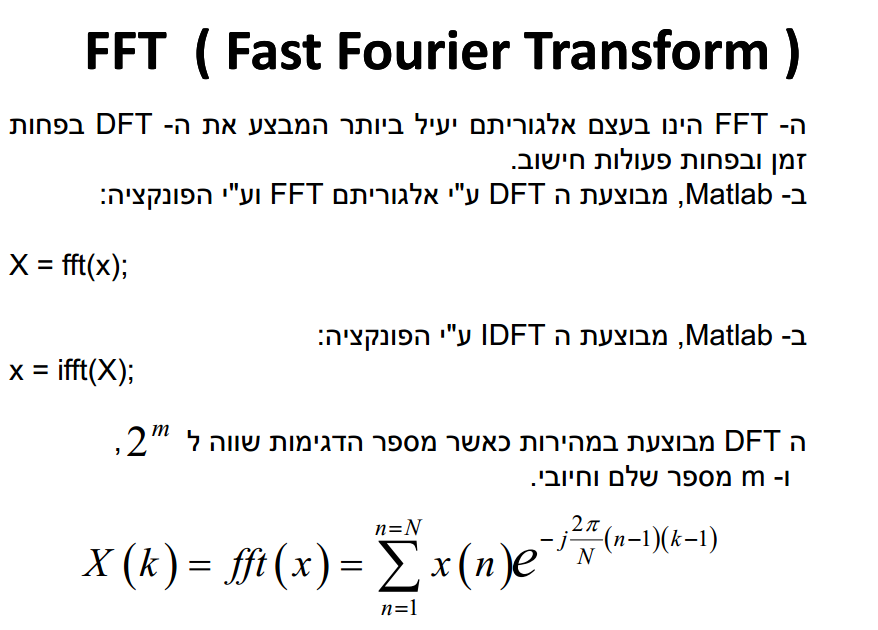
ברוב היישומים ה- Window Function שיותר נעשה בהם שימוש הן  
 הפונקציות המתייחסות יותר לצורות גרפיות של "פעמון".

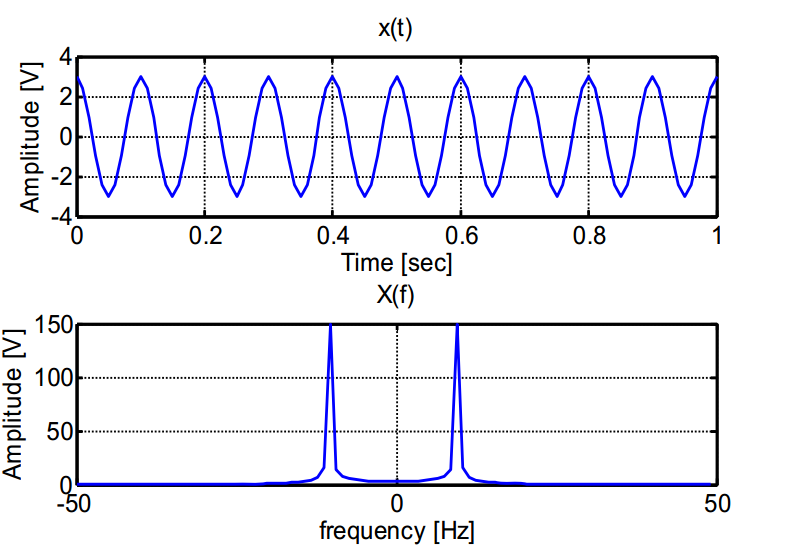
באלגוריתם שלנו אנו משתמשים ב-Hamming Window.











**דציבל:** [[8]](#footnote-8)

דציבל (אנגלית: Decibel) היא יחדית מידה חסרת מימדים המשמשת להשוואה בין שתי רמות הספק,שם היחידה נגזר מהתחילית "דצי" (מ-decimus,בלטינית:עשירית) ו"בל".את הדציבל מסמנים באותיות dB.

בל היא יחידה שהוגדרה במעבדות בל בשנת 1923 או 1924,ושמה נקבע לכבוד מייסד המעבדות וחלוף התקשורת,אלכסנדר גרהם בל. יחידה זו היא יחסית,והיא משווה שתי רמות הספק בסקאלה לוגריתמית לפי בסיס 10,כלומר היא שווה ללוגריתם לפי בסיס 10 של היחס בין שני הספקים. גידול של בל אחד פירושו לפיכל גידול של פי 10 בהספק.בנוסחא מבטאים זאת כאן ו הן רמות הספק.בל היא יחידת מידה גדולה למדי לרוב השימושים,ולכן מקובל יותר השימוש בדציבל,השווה לעשירית הבל: .

**האלגוריתם:**

1. **תרשים זרימה:**

2. **תיאור החלקים:**

2.1: Segmentation: מחלק את ה-signal לאוסף של קטעים המכסה את כל קטע הקול עם  
 חפיפה בין הקטעים.

2.2: Hamming Window: חידוד דגימת הקול ע"י נתינת ערכים גבוהים במרכז החלון וערכים  
 נמוכים יותר כאשר מתקרבים לקצוות ובכך בעצם אנו גורמים לכך שהטווח הרצוי שלנו יהיה  
 חד יותר לעומת השאר בדגימה.

2.3: Initial Noise Power Spectrum Mean: אנו מגדירים את הזמן שבו אנו מחליטים שזהו   
 הרעש הקיים(אצלנו 0.25 שניות) ומבצעים ממוצע כך שאנו מקבלים לבסוף מערך בגודל של   
 החלון שלנו שמגדיר את הרעש.

2.4: FFT: מעבר מתחום הזמן לתחום התדר-התוצאות המוחזרות מכאן הינם מספרים מרוכבים.

2.5: Compute Magnitude Average: חישוב ממוצע העוצמה של התדרים בנק' זמן מסויימת.  
 נעשה על ה spectrogram של דגימת הקול שקיבלנו.

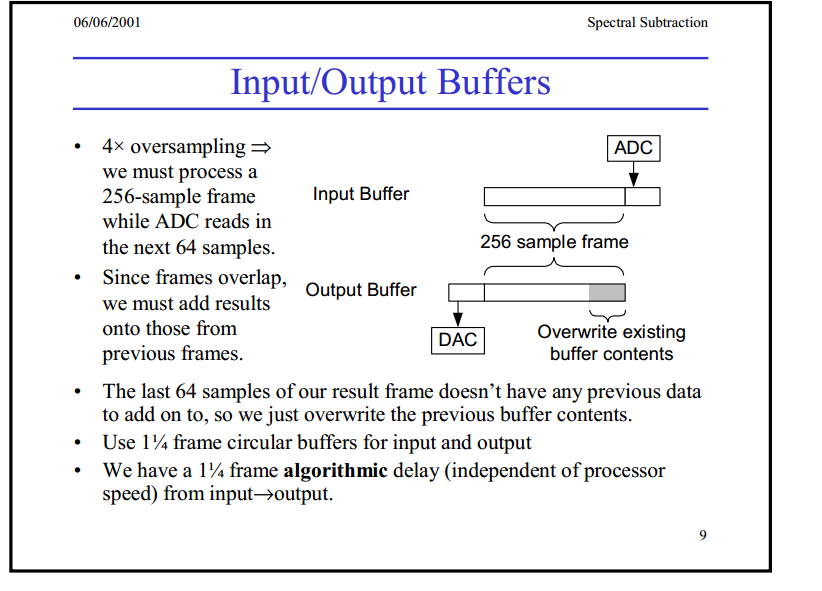
2.6: VAD: בדיקה האם ההפרש בין ה signal לבין הרעש שחישבנו הוא גדול מהסף   
 המוגדר,במידה והוא גדול אנו מגדירים שקיים דיבור בדגימה זו ובמידה ולא ומצאנו   
 מס' חלונות רצוף כזה אנו מחליטים שאין דיבור.

2.6.1: speechFlag = 1 (החלטה שקיים דיבור) : אנו בודקים בכל נקודה בחלון מה היחס   
 שלה אל מול מקסימום הרעש שאנו יודעים שיש עד לנקודה זו,אם אנו קטנים יותר   
 אז אנו לוקחים את המינימום בין נק' הזמן שלנו ,נק' הזמן לפני,נק' הזמן אחרי.   
 במילים פשוטות- מגבירים את עוצמת הדיבור ומנמיכים את עוצמת הרעש.

2.6.2: speechFlag = 0 (החלטה שקיים דיבור) : אנו מגדירים משתנה שעוזר לנו להחליק   
 את הרעש(להנמיך אותו),מעדכנים את מקסימום עוצמת הרעש עד עכשיו   
 ומקטינים את עוצמת חלון זה. במילים פשוטות- מקטינים את עוצמת הרעש לרמה   
 נמוכה.

2.7:OverlapAdd2: החלק הזה משתמש ב IFFT אשר מעביר אותנו ממספרים מרוכבים חזרה   
 למספרים ממשיים כלומר מעביר אותנו חזרה מתחום התדר לתחום הזמן. הפונקציה מייצרת   
 לנו את ה signal-out ע"י הכנסה של הערכים המתקבלים מה IFFT למקום מוגדר בכל   
 איטרציה כך שלבסוף אנו מקבלים signal שלם ללא החפיפות שייצרנו בהתחלה בחלק של ה   
 segmentation.

**המעבר לreal-time**



האלגוריתם הבסיסי עובד על קובץwav שלם, כלומר מעבד את הדגימות בזו אחר זו וממצע את הרעש בהתאם לזיהוי האם יש דיבור או אין. מכיוון שבדיבור בזמן אמת אין לנו את הדגימה שהבא לאחר מכן לא ניתן למצע את הרעש אלא על שתי דגימות בזו אחר זו. כלומר ניצור השהייה של דגימה אחת 512 דגימות קול ורק לאחר שקראנו את הדגימה הבאה נבצע ממוצע לשתי הדגימות יחדיו. על ידי כך נוכל לפתור את בעיית הממוצע של הרעש. הדבר דרש מאיתנו לבנות את האלגוריתם מחדש לגמרי. כמוכן שימוש בjava ב real-time הוא אינו דבר של מה בכך. לשם כך נעזרנו במומחה real-time, אור גושן שעזר לנו באופטימיזציות עבור עיבוד בזמן אמת. כמה טיפים למתכנתים: המילה new היא יקרה ביותר בזמן עיבוד real-time יש לשים טוב טוב לב לשימוש בה. כמובן שימו לב שיש הבדל לשימוש בין linkedLlist והשימוש ב listArray . השימוש ברשימה מקושרת מתבקש פה מכיוון שאיננו צריכים לשמור את המידע הקודם רק את המידע הנוכחי. ולכן לאחר העיבוד נפטרנו מהדגימה הקודמת.

שימו לב כמוכן לשימוש ב pool of Segments. על מנת שלא נבצע הקצאת זכרון מחדש בכל פעם למעשה הקצאנו פעם אחת את כל הדגימות מראש ולאחר מכן רק החזרנו אותם למאגר לאחר שביצענו איפוס, והוצאנו אותם בזמן הצורך. עקרון זה הוא בסיסי בעיבוד בזמן אמת ויש להשתמש בו כמעט בכל תוכנת real-time.

**שיפורים עתידיים:**

1. אנו משתמשים באלגוריתם פשוט יחסית של spectral subtraction,לכן ניתן לשפר את אלגוריתם הסינון לאלגוריתם מתקדם יותר של הפחתה ספקטרלית, כמוכן ניתן להשתמש ב Low-Pass Filter.
2. ניתן לבצע סינון עבור וידאו פרוטוקול RTP מאפשר גם העברת וידיאו וגם שם יש מקום לשיפור רעשי רקע בתמונה.
3. פרוטוקול SIP תומך גם בשיחה לטלפונים קווים או אלחוטיים רגילים יש לקנות דקות שיחה ולבדוק האם האלגוריתם אכן משפר את איכות השיחה.
4. יש לבצע התאמה למכשירי אנדרויד חלשים יותר ( המכשיר נוסה על SGS3, SGS2, galaxy

nexus).

1. ניתן לממש את שאר האלגוריתמים הקיימים באתר הבא: <http://dea.brunel.ac.uk/cmsp/Home_esfandiar/Sample%20Wave%20Files.htm>

אלא הם אלגוריתמים של הפחתה ספקטרלית שכולם מבוססי האלגוריתם של Boll ולכן התוכנה שלנו מהווה 80% מן העבודה.

1. אפשרות נוספת היא לבנות תוכנה אשר נותנת את האפשרות למשתמש לבחור בין האלגוריתמים השונים, בזמן שיחת VOIP. ובכך לקבל את התוצאה הטובה ביותר עבור כל שיחה.

**מדריך למתכנת:**

במדריך זה אנו נעבור על רשימת הפונקציות כלומר נפילת התוכנית בכללי עד לרגע שבו אנו מקצים את האלגוריתם שלנו ובו יהיה הסבר מפורט על של הפונקציות.

הקוד שלנו מבוצע במחלקה:

* RtpStreamSender– מחלקה הפועלת בזמן שיחה ומטפלת בצד שלנו בשיחה,בין אם אנו יוזמים את השיחה ובין אם אנחנו מקבלים שיחה.

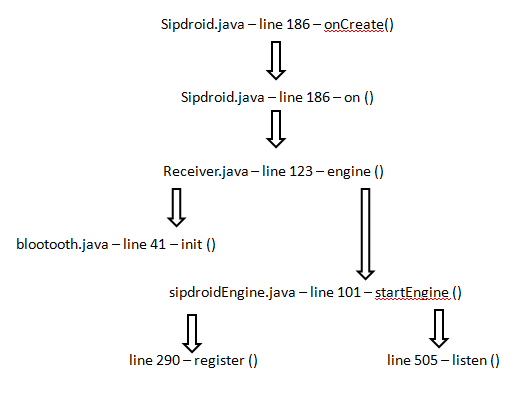
**PRD-(Procedure Relationship Diagram):**

**עליית האפליקציה בפעם הראשונה:**

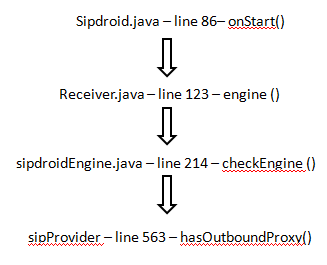
בעליית האפליקציה פועלים מספר תהליכים:

**Sipdroid main activity:**

ב flow זה אנו מפעילים את האפליקציה ומגדירים את המחלקות הרלוונטיות להפעלת האפליקציה,התחברותלשרת,האזנה למצב הטלפון ואת אפשרות ה blootooth.

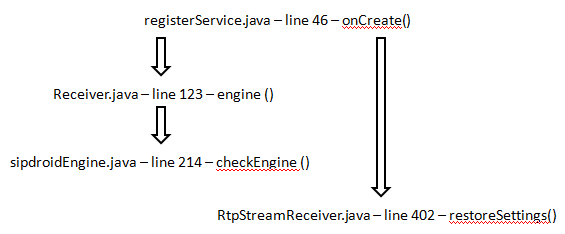


Flow זה מגדיר את שכבת התעבורה.



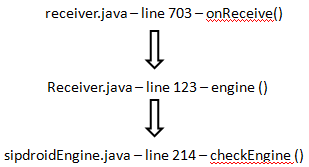
**Service:**

Flow זה מתאר שירות עבור האפליקציה שבו הואמאתחל את ההגדרות של RtpStreamReciever.

****

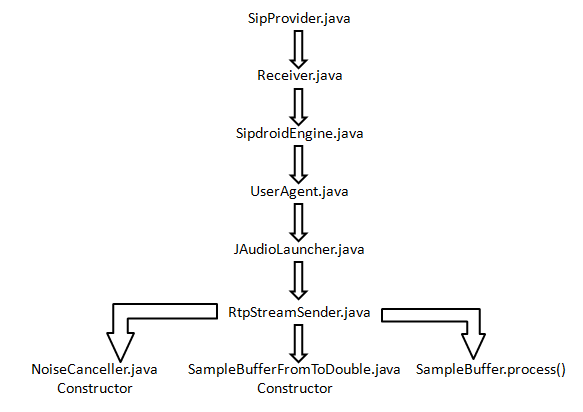
**Idle state:**

flow זה מתאר את מצב ההמתנה לביצוע intent כלשהוא ע"י המשתמש והפעלתו במידה והתבצע.



לאחר flow אלה אנו מגיעים למצב המתנה של האפליקציה בעמוד הראשי שלה.

כאשר מוציאים שיחה מתבצע ה flow עד להקצאת האלגוריתם שלנו שה flow שלו מופיע בהמשך:

****

**תיאור מחלקות האלגוריתם:**

**Class SampleBuffer:**

מחלקה המחזיקה את הדגימות שנקראו מהמיקרופון (גודל 512 דגימות) הסינון נעשהלאחר השהייה של דגימה אחת 64 אלפיות שנייה.

בנוסף אחראית על חלוקת הדגימה שלנו לחלונות חופפים המכסים את כל הדגימה הנוכחית והחזרה של הדגימה מסוננת לתוך ה buffer שאנו שולחים החוצה ללא החפיפה בין החלונות.

|  |
| --- |
| SampleBuffer |
| +buffers: LinkedList<double[]>  -nc:NoiseCanceller  +is\_first\_sample:boolean  +incomplete:List<Segment> |
| #getCurrentInputSampleBuffer():double[]  #process():double[]  -segmentize():void  -desegmentize():void  -combineOperation(buffer : double[],pos :int,value :double):void |

**getCurrentInputSampleBuffer()**– מחזיר את הדגימה עליה אנו רוצים להפעיל את האלגוריתם.

**process()**– מבצע הרצה של האלגוריתם על דגימה מסויימתתו"כ כך שהוא משלים את מה שחסר   
 לו, מהדגימה הבאה בתור.

**segmentize()**- מבצע את החפיפה בין הדגימות.

**desegmentize()**- מפריד את החפיפה בין הדגימות ומחליף את הדגימה הקיימת עם הרעש לדגימה   
 שעברה סינון

**combineOperation(double[] buffer,intpos,double value)**- מציב את הערך המסונן בבאפר   
אותו אנו מוציאים החוצה.

|  |
| --- |
| SampleBufferFromToShort |
|  |
| + process(sample:short[],pos:int):void |

|  |
| --- |
| SampleBufferFromToDouble |
|  |
| + process(sample:double[],pos:int):void |

**Process()**–הפעלת האלגוריתם עצמו.

**Class Segment:**

מחלקה האחראית על החזקת מידע עבור סגמנט בודד (לאחר חלוקה של הדגימה לסגמנטים חופפים בגודל 256 דגימות בכל אחד מהם. עם חפיפה של 50%, ניתן לסמן בכל סגמנט האם הוא מלא עד הסוף או לא, עבור המעבר בין הדגימות השונות ועיבוד הקול החופף).

**copy(Segment seg)**- העתקה של הbuffer המקורי ל buffer זמני.

**copyPos(Segment seg)**– העתקת המיקום בו אנו נמצאים כרגע ואליו אנו מעתיקים את הדגימה   
 לאחר הסינון

**clear()**- איפוס המיקום אליו אנו רוצים להעתיק את הדגימה המסוננת.

**markInput()**-סימון האם הסגמנט מלא עד הסוף.

**markOutput()**-סימון האם הסגמנט מלא עד הסוף.

**input(double value)**- מכניסה את הערך לפני השינוי למקום שלו ב buffer.

**output()**- מחזירה את הערך שלנו לאחר השינוי שלו.

**isSharingSampleBuffers()**- פונקציה הבודקת אם הדגימה מכילה גם חלק מדגימה נוספת.

**isHalfFull()**- פונקציה הבודקת האם הדגימה לא מלאה לחלוטין,כלומר צריכה גם חלק מהדגימה   
 הבאה בשביל להמשיך לבצע את הסינון

**getBuffer()**- מחזירה את הדגימה עליה אנו עובדים.

**getAngleBuffer()**- מחזיר את המערך של הזוית ממנה מגיע הקול בחלון הנוכחי.

**mean(Segment prev, Segment current, Segment next)**-ממוצע האמפליטודה עבור 3 סגמנטים צמודים.

|  |
| --- |
| Segment |
| -buffer:double[]  -angle:double[]  -from\_pos:int  -to\_pos:int |
| +copy(seg :Segment):void  +copyPos(seg :Segment):void  +clear():void  +markInput():void  +markOutput():void  +input(value :double):boolean  +output():double  +isSharingSampleBuffers():boolean  +isHalfFull():boolean  +getBuffer():double[]  +getAngleBuffer():double[]  +mean(prev:Segment,current:Segment,next:Segment):void |

**Class NoiseCanceller:**

מחלקה זו מבצעת על הדגימות את האלגוריתם Spectral Subtraction

**getSampleSize()**–מחזיר את הגודל של הדגימה.

**getShiftSize()**–מחזיר את הגודל של החפיפה בין החלונות.

**getInputSegment()**– מחזיר את הדגימה עליה אנו עומדים לבצע את הסינון של הרעש

**getOutputSegment()**–מחזיר את הדגימה לאחר הסינון של הרעש.

**process()**–פונקציה שמבצעת את הסינון(פונקציית אב).

**createHammingWindow(buffer:double[])**– פונקציה שיוצרת את חלון hamming.

**applyHammingWindow(buffer:double[],hamming:double[])** -

**performFFT(buffer:double)**–פונקציה שמפעילה FFT על הדגימה.(מעבירה ל complex)

**performInverseFFT(buffer:double)** - פונקציה שמחזירה את הדגימה למצב מקורי לאחר סינון.(מחזירה ל real)

**calcAngle(buffer:double[],angle:double[])**– פונקציה שמחשבת את הזוית ממנו הגיע הקול.

- **calcABS(buffer:double[])**מחזיר את האמפליטודה.

- **calcMeanNoise(buffer:double[])** מחשבת את ממוצע הרעש הקיים על סמך הזמן שהגדרנו.

- **hasSpeech(buffer:double[],noise\_count:int)** בודקת האם קיים דיבור בחלוון הנוכחי של הדגימה.

**minMaxZero(buffer:double[],prev:double[],current:double[],next:double[],noise:double[],NRM, :double[])**– מוצא את המינימום שארית רעש ומפחית אותה מהדיבור הקיים על 3 סגמנטים חופפים רצופים.

- **minMaxZeroSingle(buffer:double[],current:double[],noise:double[])**מוצא את המינימום שארית רעש ומפחית אותה מהדיבור הקיים על סגמנט יחיד.

- **ratioAverage(noise:double[],original:double[],total:int,part:int)**מבצע החלקה של הרעש,ע"י מציאת ממוצע הרעש אותו אנו מפחיתים בהמשך.

- **updateMaxNRM(buffer:double[],average\_noise:double[],noise:double[])**פונקציה שמעדכנת את מקסימום הרעש הידוע לנו עד עכשיו.

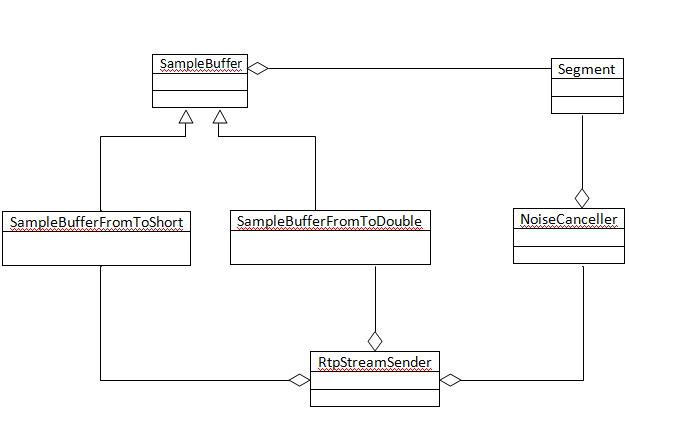
- **addMultipliedSegment(buffer:double[],original:double[],N:double)**מבצע הכפלה של הסגמנט בקבועN.

- **performExp(buffer:double[],angle:double[])**הכפלה הסגמנט באקספוננט של הפאזה.

- **reverseComplexArray(buffer:double[])**פונקציה המבצעת היפוך של המערך שלנו(מקביל ל-flipud של matlab)

|  |
| --- |
| NoiseCanceller |
| -$ HANGOVER\_CONST:int=8  -$ NOISE\_MARGIN:int=3  -$ NOISE\_LENGTH:int=9  -$ WINDOW\_SIZE\_CONST:double = 0.032  -$ SHIFT\_PERCENTAGE:double = 0.5  -$ NOISE\_PERIOD:double = 0.25  -$ NOISE\_AVERAGE\_RATIO:int = 9  -$ BETA:double = 0.03  - sample\_size:int  - window\_size:int  + recycled:LinkedList<Segment>  + segments:LinkedList<Segment>  + preprocessed:LinkedList<Segment>  + noise:LinkedList<Segment>  + processed:LinkedList<Segment>  - average:ArrayList<Segment>  - shift\_size:int  + fft:DoubleFFT\_1D  -hammingWindow:Segment  - average\_noise:Segment  -NRM:Segment  -noise\_segments:int  -spectralDist:double[] |
| + getSampleSize():int  + getShiftSize():int  + getInputSegment():Segment  + getOutputSegment():Segment  + process():void  - createHammingWindow(buffer:double[]):void  - applyHammingWindow(buffer:double[],hamming:double[]):void  - performFFT(buffer:double):void  - performInverseFFT(buffer:double):void  - calcAngle(buffer:double[],angle:double[]):void  - calcABS(buffer:double[]):void  - calcMeanNoise(buffer:double[]):void  - hasSpeech(buffer:double[],noise\_count:int):Boolean  - minMaxZero(buffer:double[],prev:double[],current:double[],  next:double[],noise:double[],NRM, :double[]):void  - minMaxZeroSingle(buffer:double[],current:double[],noise:double[]):void  - ratioAverage(noise:double[],original:double[],total:int,part:int):void  - updateMaxNRM(buffer:double[],average\_noise:double[],noise:double[]):void  - addMultipliedSegment(buffer:double[],original:double[],N:double):void  - performExp(buffer:double[],angle:double[]):void  - reverseComplexArray(buffer:double[]):void |

**UML**



לאחר שאנו יוזמים שיחה אל מקור אחר או שאנו עונים לשיחה ממקור אחר האלגוריתם שלנו מופעל ב classRtpStreamSender.  
קודם כל אנו מבצעים אתחול של כל הנתונים הרלוונטיים לנו לסינון הרעש,וזה מתבצע ב Constructor של המחלקה NoiseCanceller.

הנתונים אותם אנו מאתחלים:

1. נתונים עבור החלטה האם בדגימה הנוכחית קיים דיבור או לא:
   1. HANGOVER\_CONST- מגדיר את מס' החלונות שיש לזהות כדי להגדיר שלא קיים דיבור.
   2. NOISE\_MARGIN– קבוע בעזרתו מחליטים האם יש רק רעש או גם דיבור.
   3. NOISE\_LENGTH -
2. WINDOW\_SIZE\_CONST– בעזרתו מוגדר גודל החלון שלנו.
3. SHIFT\_PERCENTAGE– גודל החפיפה בין החלונות שלנו.
4. NOISE\_PERIOD– הזמן המוגדר בתחילת השיחה שהוא "רעש".
5. hammingWindow– החלון בעזרתו אנו מחדדים את הדיבור בדגימה שלנו.
6. BETA– קבוע שבעזרתו אנו מנמיכים את הרעש כאשר מזהים שאין דיבור.
7. window\_size– גודל כל חלון כאשר אנו מבצעים overlapping לדגימה.
8. shift\_size– גודל החפיפה ממש(מספר שלם) שיש בין החלונות.

**PRD-(Procedure Relationship Diagram):**

SampleBuffer.process()

Segment.segmentize()

Segment.input()

NoiseCanceller.getInputSegment()

Segment.clear()

Segment.markInput()

Segment.isHalfFull()

Segment.isSharingSampleBuffer()

NoiseCanceller.getShiftSize()

Segment.markOutput()

Segment.desegmentize()

SampleBuffer.combineOperation()

NoiseCanceller.getOutputSegment()

Segment.isSharingSampleBuffer()

NoiseCanceller.process()

NoiseCanceller.performInverseFFT()

NoiseCanceller.addMultipliedSegment()

NoiseCanceller.ratioAverage()

NoiseCanceller.updateMaxNRM()

NoiseCanceller.hasSpeech()

NoiseCanceller.calcMeanNoise()

NoiseCanceller.applyHammingWindow()

NoiseCanceller.performFFT()

NoiseCanceller.reverseComplexArray()

NoiseCanceller.performEXP()

Segment.copy()

NoiseCanceller.calcAngle()

Segment.copyPos()

NoiseCanceller.calcABS()

SampleBuffer.process()

**בדיקות:**

1. מציאת אלגוריתם הממומש בשפת matlab ובדיקה האם הוא באמת מסנן רעשי רקע,מצ"ב תיקייה[[9]](#footnote-9) בה קיים האלגוריתם והדגימה לפני ואחרי הסינון, האלגוריתם נבדק על כמה דגימות קול שונות בכמה תדירויות דגימה שונות, SipDroid עובד בתדירות8khz כלומר 8000 דגימות לשנייה.
2. המרת האלגוריתם ל java עד להגעה לתוצאה הרצוייה כמו באלגוריתם ה matlab. מצ"ב תיקייה[[10]](#footnote-10) עם האלגוריתם והדגימה לפני ואחרי הסינון, גם הוא נבדק על דגימות שונות.
3. חיבור האלגוריתם לאפליקציית sipdroid והבנה שהאלגוריתם במימושו הנוכחי אינו מתאים למימוש Real-time, השתמשנו בprofiler של netbeans על מנת להבין שלוקח שנייה וחצי לעבד שניה אחת של דגימה.
4. שכתוב הקוד והתאמתו ל Real-time, עברנו למימוש Sepctral Subtraction for RTP.

על מנת לבצע בדיקות לקוד שהותאם ל Real-time בנינו מחלקה בשם NoiseCancellerTester שתפקידה לקרוא קובץ wav ולדמות את פעולת הRTP ב- SIP כלומר בכל איטרציה אנו קוראים דגימה בגודל 512 ומבצעים עליה הרצה של הסינון ולבסוף כותבים את התוצאה לתוך קובץ wav חדש ובודקים את השמע.

לאחר מחקר מצאנו מחלקה המאפשרת לנו לבצע קריאה וכתיבה לקבציwav בצורה נוחה מאוד,למחלקה זו קוראים WavFile.

תיעוד והורדה של המחלקה ניתן למצוא בכתובת הבאה: <http://www.labbookpages.co.uk/audio/javaWavFiles.html>

דוגמאות ניתן למצוא האתר הקורס.

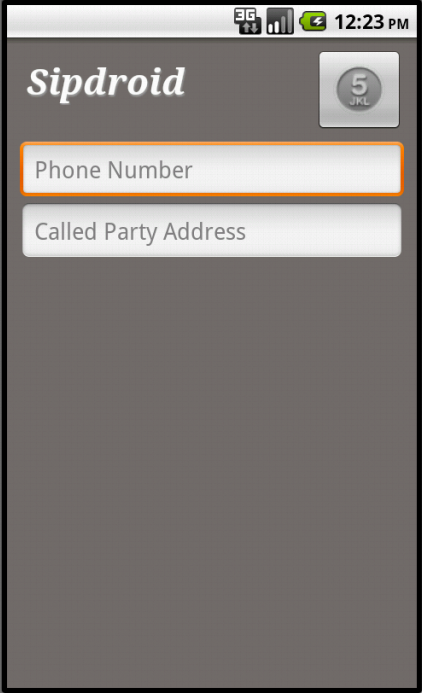
**מדריך למשתמש:**

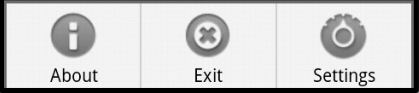
1. התקן את גירסתsipdroid המצורפת למסמך זה במכשירך.
2. פתח חשבון sip.ניתן לפתוח בכתובת הבאה: <https://mdns.sipthor.net/register_sip_account.phtml>

לאחר הפתיחה ייתקבל אצלכם מייל אם שם המשתמש והסיסמא שעליכם להזין הבמשך,לכן חשוב מאוד לשמור את המייל הזה ואת הפרטים שלו.

1. לאחר שפתחנו חשבון sip אנו יכולים להכנס לתוכנה ע"י לחיצה על הכפתור הבא הנמצא במסף היישומים שלנו:

לאחר מכן ייפתח המסך הראשי של האפליקציה:



1. כעת עלינו להגדיר את פרטי החשבון שיש לנו באפליקציה,לחץ על כפתור ה Menu:

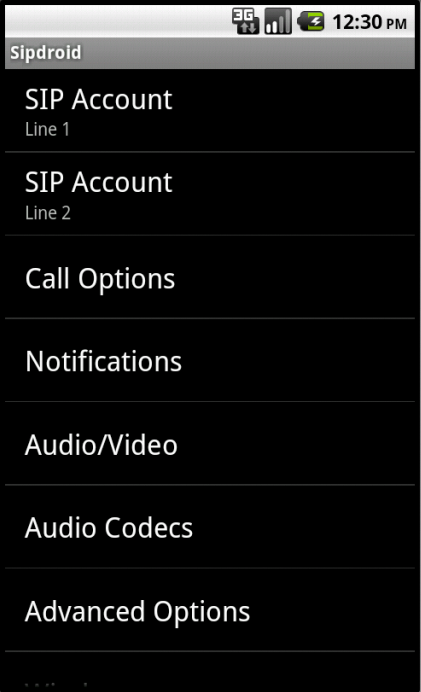
בתחתית המסך ייפתחו הכפתורים הבאים:

הגדרות האפליקציה

יציאה מהאפליקציה

על   
האפליקציה

1. לחץ על כפתור Settings והמסך הבא ייפתח:



הגדרות עוצמה ואיכות של מיקרופון ווידאו

הגדרה של תזכורות(שיחה שלא נענתה,הודעה וכו')

הגדרת סוג קידוד השיחה

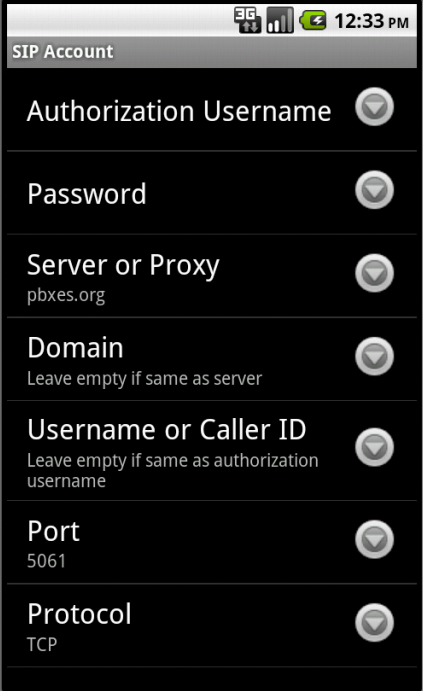
הגדרות נוספות ומתקדמות

הגדרות שיחה (הקלטה למשל)

הגדרת חשבון נוסף במערכת

הגדרת חשבון במערכת

1. לחץ על SIP Accountוההמסך הבא ייפתח:



הגדרת שם המתקשר לפי ההגדרה בשרת

על איזה פרוטוקול לבצע את השיחה

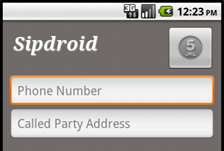
הגדרת הכניסה

הגדרת שרת sip

הזנת סיסמת משתמש

הגדרת שרת proxy

הגדרת שם משתמש

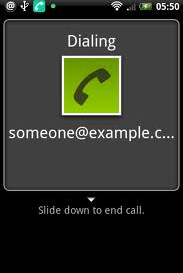
1. כעת יש למלא את הפרטים הבאים:
   1. Authorization Username- שם המשתמש שהזנת בפתיחת חשבון ה SIP שלך.
   2. Password- הסיסמא שהזנת או שקבלת כאשר פתחת את חשבון ה SIP שלך.
   3. Server or Proxy- לדוגמא עבור חשבון ב sip2sip :proxy.sipthor.net.
   4. Domain- לדוגמא עבור חשבון ב sip2sip : sip2sip.info
   5. Port- 5060, הport 5061 משמש יותר בשביל שיחות הדורשות הצפנה ואז צריך גם שרת שמאפשר הצפנה.
   6. Protocol- UDP.
   7. לסמן ב 'V' את התיבה Use WLAN.
2. כעת חיזרו אל המסך הראשי ע"י לחיצה על חזור.
3. על מנת להתקשר לחשבון אחר יש ללחוץ על אחד מהשדות הבאים ולאחר מכן תפתח מקלדת.

במידה ויש לנו חשבון שאנו משלמים בו ואנו יכולים להתקשר למספרי טלפון אנו יכולים להשתמש בשדה Phone Number.

במידה ואנו מתקשרים לחשבון sip אחר או לחשבון skype לדוגמא, אנו נשתמש בשדה "Called Party Address".

נכניס את הטלפון או הכתובת:לדוג' עבור skype: username@skype ונלחץ על "התקשר"/ send.

1. לאחר שאנו לוחצים על להתקשר המסך ייראה כך:



1. כאשר אנו נמצאים בשיחה המסך ייראה כך:



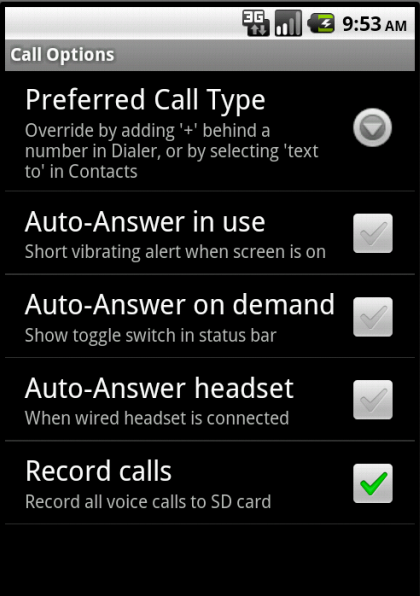
1. במידה ואנו רוצים להעביר לרמקול יש ללחוץ על menu והמסך ייראה כך ואז ללחוץ על speaker :



1. על מנת לנתק יש להחליק את הריבוע כלפי מטה,במידה ואנו נמצאים במצב שלאחר לחיצה על menu אז יש לבצע לחיצה על הכפתור .
2. כאשר מתקבלת שיחה המסך ייראה כך:



1. על מנת לנתק יש להחליק את הריבוע כלפי מטה, במידה ואנו נמצאים במצב שלאחר לחיצה על menu אז יש לבצע לחיצה על הכפתור .
2. הקלטה:
   1. לחץ על Menu.
   2. לחץ על Settings.
   3. לחץ על Call Options.
   4. סמן V על Record calls.כך שהמסך ייראה כך:



**תודות:**

ברצוננו להודות לאנשים הבאים:

1. מר חננאל חזן-מנחה הפרוייקט.
2. אור גושן - מומחה .Real-time
3. גלעד כהן – על עזרה בהבנת הרקע התיאורטי הנרחב.
4. אלי-זיו כהן – על הדרכת .matlab

1. <http://developer.android.com/reference/android/media/AudioRecord.html> [↑](#footnote-ref-1)
2. <http://developer.android.com/reference/android/media/AudioTrack.html> [↑](#footnote-ref-2)
3. <http://developer.android.com/reference/android/media/MediaRecorder.html> [↑](#footnote-ref-3)
4. <http://en.wikipedia.org/wiki/Session_Initiation_Protocol> [↑](#footnote-ref-4)
5. http://en.wikipedia.org/wiki/Real-time\_Transport\_Protocol [↑](#footnote-ref-5)
6. Suppression of Acoustic Noise in Speech Using Spectral Subtraction - 1979 [↑](#footnote-ref-6)
7. Testing.rar [↑](#footnote-ref-7)
8. <http://he.wikipedia.org/wiki/%D7%93%D7%A6%D7%99%D7%91%D7%9C> [↑](#footnote-ref-8)
9. matlabTest [↑](#footnote-ref-9)
10. testing [↑](#footnote-ref-10)