תיק פרויקט

סנכרון מוזיקה

# תוכן עניינים

# מבוא

Syncalong הינה תוכנה המאפשרת לנגן מוזיקה בצורה מסונכרנת בין מספר מכשירים. התוכנה מאפשרת לשתף קבצי מוזיקה בין מספר תחנות, ולהפעיל קבצי מוזיקה בו-זמנית בכל התחנות. באופן זה, מאפשרת syncalong להפוך מספר מכשירים המחוברים רשתית זה לזה למערכת סאונד שלמה.

המערכת מורכבת משרת יחיד, אליו ניתן לחבר מספר תחנות לקוח. ממשק המערכת בשרת מאפשר לנהל את החיבורים לשרת ואת רשימת השירים להשמעה, וכן לשלוט בנגינת המוזיקה בתחנות המרוחקות. ממשק המערכת בצד הלקוח מאפשר להתחבר לשירות או להתנתק ממנו, ומראה את המוזיקה המנוגנת כעת.

הפעולות הנתמכות על ידי המערכת הינן:

* שליחת קבצי מוזיקה לתחנות במחוברות לשרת, במידה והם אינם קיימים בתחנת הקצה
* ניגון, השהייה ועצירה של קובץ מוזיקה
* בניית רשימת השמעה והפעלתה באופן רציף בכל התחנות

# דרישות קדם

המערכת מותאמת לעבודה על כל מערכות ההפעלה, ונבדקה על מחשבי Windows 10 ו-Ubuntu 12.04. יש לוודא כי על תחנות הקצה מותקן כרטיס קול וישנם רמקולים מחוברים.

על מנת להריץ את התוכנה יש להתקין python בגרסה 3.7.4 ומעלה.

# הוראות התקנה

על מנת להתקין את המערכת, יש להיכנס לתיקייה הראשית (syncalong), ומשם להריץ את הפקודה: python setup.py install. הפקודה תתקין על המחשב הנוכחי הן את תוכנת השרת והן את תוכנת הלקוח, יחד עם כל התלויות הנלוות למערכת.

הגדרות המערכת ניתנות לשינוי ישירות בממשק התוכנה.

# הוראות שימוש

בלה בלה בלה

# ארכיטקטורת תוכנה

## סקירה

התוכנה בנויה מצד שרת, וצד לקוח. באחריות השרת לקבל חיבורים חדשים, לנהל את רשימת השירים להשמעה ולסנכרן את הפעולות השונות מול הלקוחות. בצד הלקוח ניתן לקבל קבצים מהשרת, וכן להשמיע מוזיקה על פי הוראות השרת. לצורך התקשורת בין השרת ללקוח, המערכת עושה שימוש במספר טכנולוגיות:

* התקשורת בין השרת ללקוחות עבור שליחת פקודות וסנכרון קבצים נעשית מעל פרוטוקול TCP. לצורך העברת המסרים יצרנו פרוטוקול משלנו, עליו יפורט בהמשך.
* לבניית ופרסור פקטות השתמשנו בספרייה scapy, המאפשרת להגדיר פרוטוקול ייחודי להגדרת המתכנת.
* הסנכרון בין השרת ללקוחות, שבאמצעותו כלל התחנות מנגנות את המוזיקה באותו רגע בדיוק, נעשה באמצעות פרוטוקול NTP. השרת משמש גם כשרת NTP, כך שכלל התחנות מסונכרנות לפיו. בהמשך הפרק נציג הסבר נוסף על הפרוטוקול ואופן השימוש בו בתוכנה.
* קבלת חיבורים חדשים בצד השרת נעשית באופן א-סינכרוני, כך שניתן לבצע פעולות למול לקוחות קיימים גם בזמן טיפול בלקוחות חדשים. דבר זה נעשה באמצעות שימוש ב-thread ייעודי לקבלת לקוחות בצד השרת.

השימוש ב-TCP מאפשר למערכת לשמור על אמינות גבוהה הנדרשת בתהליך סנכרון הקבצים והעברת האותות. יחד עם זאת, משמעות הבחירה בפרוטוקול זה היא שהתקשורת בין השרת ללקוחות אורכת זמן רב, יותר מאשר לו היינו בוחרים להשתמש ב-UDP. בחרנו להעדיף אמינות על פני מהירות מכמה סיבות:

* על העברת הקבצים מהשרת ללקוחות להיות אמינה. התוכנה תומכת בנגינת כל הפורמטים המוכרים לקבצי מוזיקה, והרבה מפורמטים אלו מסתמכים על שלמות המידע שבקובץ. אילו חלק מהקובץ יועבר באופן פגום או לא שלם, לא יהיה ניתן לנגן את הקובץ בצורה תקינה אצל הלקוח.
* קובץ שהועבר באופן חלקי עלול ליצור פערים בסנכרון שבין הלקוחות. אילו הקובץ נקטע לפני הסוף, למשל, הלקוח יעבור לנגן את השיר הבא לפני הלקוחות האחרים, וייצא מסנכרון.
* עלינו לוודא כי כל הלקוחות מקבלים את הפקודות מהשרת. מכיוון שהלקוחות מסתנכרנים מול שרת ה-NTP על מנת לבצע את הפקודה ברגע הנכון, עלינו לוודא רק שכלל הלקוחות אכן קיבלו את הפקודה. זמן ההגעה של האות מהשרת נלקח בחשבון בצד הלקוח בעת חישוב הזמן המתאים לביצוע הפקודה (ראו פירוט על אופן הסנכרון בהמשך).

## מבנה מחלקות

להלן תרשים של מבנה המחלקות בתכנית היחסים ביניהן:

<להשלים>

## תיאור המחלקות

### החבילה Common

#### LengthSocket(socket.socket)

המחלקה LengthSocket הינה האובייקט הבסיסי העוטף socket במערכת, לכתיבה ולקריאה הן בצד השרת והן בצד הלקוח. המחלקה יורשת מהאובייקט socket הפייתוני, ומשמשת לשליחה נוחה של הפרוטוקול שבשימוש המערכת. כל הודעה שנשלחת באמצעות LengthSocket נעטפת ב-header המתאים (ראו GeneralPacket), ולפניו נשלח גם גודל

### החבילה Server