# דו"ח עבודה

מגישים: אריה גרוס

אבי ווסרברגר

## תיאור המערכת

המערכת כוללת שתי פונקציות פומביות:

**הפונקציה toCsv** שמקבלת נתיב של תיקיה, לוקחת את כלל קבצי הcsv מהאפליקציה wiggle wifi **התקינים** שנמצאים בתיקייה ומכניסים אותם לקובץ csv בפורמט שנדרשנו אליו בשם wifi.csv ושמה אותו באותה התיקייה, הפונקציה משתמשת בפונקציות עזר שיוצרות לה מערך של אובייקטים של wifi מתוך קבצי האפליקציה, ויוצרות לה מערך חד ממדי שמייצג איזה שורות במערך לרשום ואיפה, ובפונקציה שרושמת בפועל.

הפונקציות משתמשות בפונקציה שמצאנו ברשת שסופרת מספר שורות בקובץ (המקור רשום שם).

במקרה שמתקבלים קבצי אפליקציה לא תקינים (למשל בלי שניות), כלל הקובץ עם השגיאה לא ייכתב (מתוך נקודת הנחה שמישהו התעסק בקובץ חוץ מהאפליקציה).

**הפונקציה writeKml** מקבלת נתיב של קובץ csv **שהאפליקציה הקודמת יצרה** ונתיב של תיקיה לשים בתוכו קובץ kml בשם wifi.kml בפורמט שנדרשנו, הפונקציה שואלת את המשתמש האם הוא רוצה נתונים רק מ ID מסויים, רק מזמן מסויים והלאה ורק מעוצמת אות מסויימת ומעלה ומסננת לפי הדרישות שלו, הפונקציה משתמשת בפונקציית עזר שעוברת שורה שורה על קובץ הcsv ובעזרת ממשק מסננת אותו לפי דרישות המשתמש ומכניסה לקובץ kml החדש, שזה מבוצע על ידי java api for kml, במקרה שקובץ הcsv לא קיים בנתיב \ פגום הפונקציה תפסיק את מהלכה, תדפיס שהיתה שגיאה ותחזיר 0 (בשונה מהקודמת שאם קובץ פגום בה היא ממשיכה לבא בתור, זאת מקבלת רק קובץ אחד ולכן אין טעם שהיא תמשיך), אחרת היא מחזירה 1.

הקובץ מכיל את הזמנים כך שהם מוצגים בtimeline

## כלי תוכנה

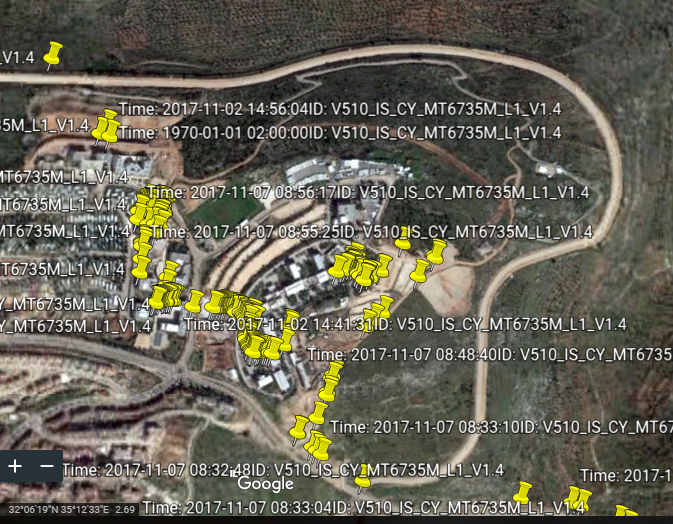
השתמשנו בספריות הרגילות בג'אווה, בממשק, בלכידת שגיאות [try-catch](על מנת לסנן קלט שגוי) וב java api for kml (jak)

## בדיקות

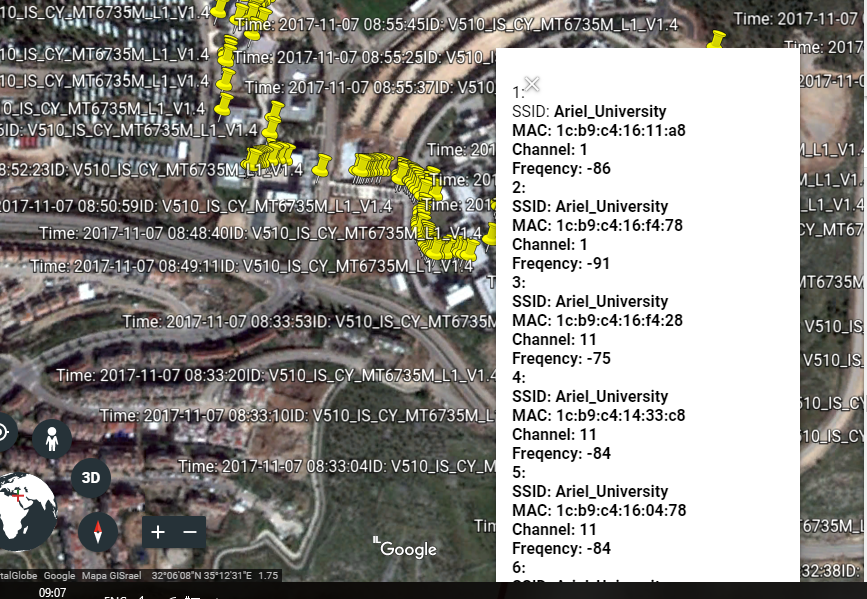
מצורפת תיקיה בשם test עם קבצי אפליקציה תקינים ולא תקינים וכן קבצי csv כנ"ל ומחלקת junit בשם test שבודקת את האפליקציה בעזרתם

# מטלה 0

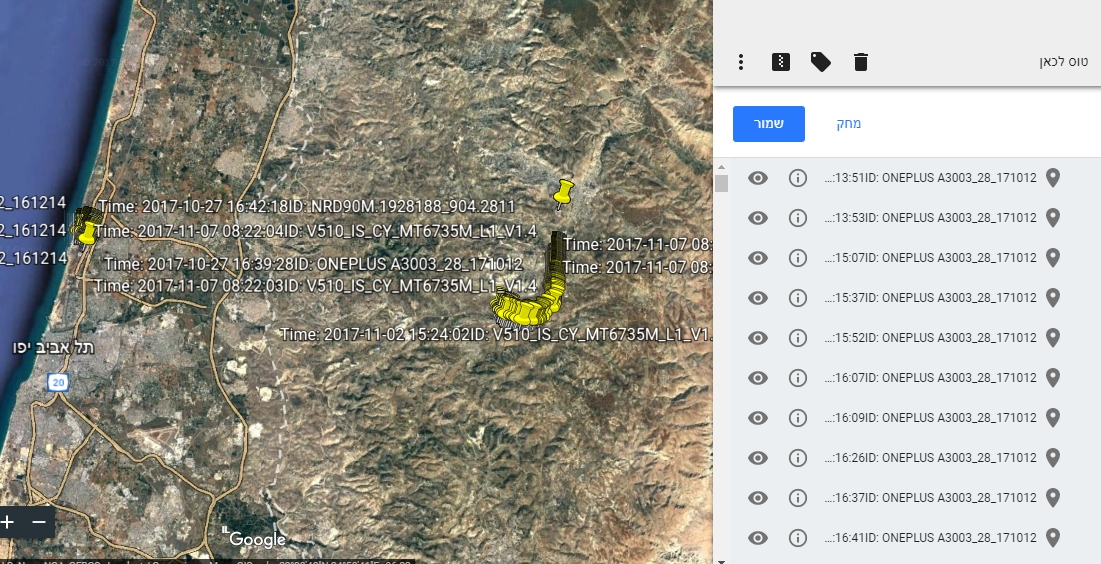
## ניסויים

ביצענו ניסוי באוניברסיטה

למעלה ניתן לראות את נקודות הדגימה עם זמן והID של הבודק

למטה לחצנו על אחת נקודות הדגימה כדי שתציג את התוכן

תמונה של כלל הדגימות כולל הדגימות שהועלו למודל ונסיעה מהר ברכה לאוניברסיטה (אפשר לראות שהgps של הפלאפון לא מדוייק בדגימה הבודדת שנמצאת בשכם בערך, עושה הרבה בעיות עם waze):



# מטלה 1

1. בוצע
2. <https://github.com/amgross/munche_prog>
3. ביצענו חלוקה למחלקות והוספנו עצמים
4. קיים, בתיקייה test
5. השתמשנו בjak, היא עושה רושם התוכנה הכי ידועה ורשמית שעושה את זה ולכן יש לה הרבה תוספות ופיצ'רים וכן הכי הרבה שאלות והסברים ברשת עליה
6. הוספנו
7. הוספנו