



רובוט שמפענח את מה שהוא רואה
ושומע. משתמש ב Google cloud
API. התקשורת נעשית בעזרת שרת
בשפת JavaScript ועם אפליקציה
בשפת Java

Searcho

פרויקט 5 יחידות לימודיות – מדעי
המחשב

מנטור – יהודה אור
מנאול נמירובסקי

פרטי התלמיד:

שם התלמיד: מנואל נמירובסקי

תעודת זהות: 211338108

תאריך לידה: 8.4.2000

כתובת: החסידה 3ב', נתניה

טלפון התלמיד: 054-255-5687

פרטים כלליים:

שם בית הספר: הכפר הירוק ע"ש לוי אשכול

טלפון בית הספר: 03-6455666

מספר יחידות לימוד: 5 יחידות

מקצוע: מדעי המחשב

פרטי המנטור:

שם המנטור: יהודה אור

תעודת זהות: 023098007

תואר אקדמי: מהנדס מבנים. מוסמך הטכניון בהנדסה, הסבה אקדמאית למחשבים מטעם המדינה מאז שנת 2000. תעודה הסמכה מטעם Microsoft.

מקום העבודה: מכון וייצמן, מכללת ג'והן ברייס, הכפר הירוק, תיכון עירוני ד', ת"א.

טלפון: 050-734-4457

אי-מייל: yooda@gmail.com

תוכן עניינים

3.....	הקדמה	➤
4.....	חלוקת העבודה	➤
5-9.....	מבוא לשימוש ב-Raspberry Pi	➤
	○ הרצה וכתובת קוד	
	○ הסבר ושימוש בחיישנים	
	○ תוצר הרובוט הסופי	
10-12.....	מבוא לשימוש ב-API	➤
13-14.....	מסמך ייזום	➤
15-17.....	מסמך אפיון	➤
18-20.....	מסמך ארכיטקטורה	➤
21-27.....	איטרציות	➤
	○ הקמת שרת ותקשורת בסיסית	
	○ הכרה ולמידה של Google Cloud API's	
	○ קליטת קול וניתוח ההקלטה	
	○ פעולות הרובוט על פי הממצאים וניתוחם	
	○ פעולות האפליקציה על פי הממצאים ושיפורם	
28-29.....	מדריך למשתמש קצה	➤
30-32.....	מדריך לממשק ניהול	➤
33.....	רפלקציה	➤
34.....	ביבליוגרפיה	➤
35-51.....	קודים	➤

במהלך השנה הנוכחית, החלטתי להקדיש את זמני על מנת לבנות פרויקט שיתעסק במגוון טכנולוגיות שלא נגעתי בהם לפני.

לשם כך, החלטתי לשלב חומרה וקוד ביחד, ולגעת במגוון נושאים שלא נגעתי בהם לפני ואו שנגעתי בהם אך לא התעמקתי. תהליך ההכנה של הפרויקט היה מורכב על מנת שיהיה מסודר מאוד. החלטתי להתחיל לכתוב את הקוד בשלב מאוחר יחסית על מנת להכין חלוקת עבודה נכונה.

הפרויקט שלי הוא רובוט שמפענח את מה שהוא רואה ומה שהוא שומע. יחד עם הרובוט, ישנה אפליקציה שדרכה ניתן להעביר לרובוט הוראות ומשימות. התקשורת בין הרובוט לבין האפליקציה נעשית באמצעות שרת שיושב על מחשב באותה רשת של האפליקציה ושל השרת (LAN). המשתמש שולח הודעה באפליקציה ומעביר משימה לרובוט, לדוגמא: "find my keys". ההקלטה נשלחת לשרת והשרת מפענח אותה בעזרת Google Cloud Speech API. ה-API מבצע תהליך שנקרא Speech To Text והופך את ההקלטה למילים.

לאחר שפענחנו את ההקלטה, אנחנו ניקח את הטקסט שקיבלנו ונפרק גם אותו לפועל ולשם עצם (בעזרת Google Cloud Natural Translate API) כדי לדעת מה המשימה (הפועל) ואת מה יש למצוא (שם העצם).

אחרי שפירקנו את המשפט, הפועל נשלח מהשרת לרובוט והרובוט מבין שהוא צריך לבצע פעולה של find. הרובוט מצלם תמונות של הסביבה שלו ושולח אותן לשרת. השרת מבצע פיענוח של האובייקטים שיש בתמונה בעזרת Google Cloud Vision API. ואז מתבצעת השוואה בין האובייקטים שנמצאו לבין השם עצם מההקלטה. אם אין התאמה, השרת שולח תשובה לרובוט שצריך עוד תמונה לצילום (התמונה הבאה של הסריקה) ואם יש התאמה – השרת מחזיר לרובוט שיש התאמה ואין צורך לשלוח עוד תמונות, ונשלחת תשובה חזרה לאפליקציה של המשתמש.

****להוסיף משהו קשה שהיה לי ומה למדתי חדש****

חלוקת העבודה

כדי להימנע מבזבז זמן מיותר במהלך כתיבת קוד הפרויקט, החלטתי להתמקד בעיצוב הפרויקט ובארכיטקטורה שלו לפני שהקוד נכתב. לשם כך הכנתי מסמך ייזום שמטרתו סקירה כללית של הפרויקט, מסמך אפיון שמטרתו לפרט יותר לעומק על הפרויקט עד כדי הבאתו למפתח שיפתח במקום מי שכתב את המסמך ומסמך ארכיטקטורה שמטרתו לבנות את השלד העיקרי למימוש הפרויקט – האופן שבו הוא יבנה, הרכיבים השונים, חלוקת האחריות והקשרים ביניהם. המסמכים יעזרו לי לכתוב את הפרויקט בצורה נקייה ולסיים אותו במהירות.

בנוסף לכך, החלטתי לפרק את הפרויקט לאיטרציות:

איטרציה 1 - מתמקדת בהקמת שרת ותקשורת בסיסית בין כל חלקי הפרויקט.

איטרציה 2 – הכרה של כל API's של גוגל וניתוח קטעי קוד.

איטרציה 3 – קליטת קוד מהאפליקציה וניתוח ההקלטה.

איטרציה 4 – פעולת הרובוט על פי הממצאים ועבודה על צילום וניתוח תמונה.

איטרציה 5 – לבדוק את פעולת האפליקציה על פי הממצאים ושיפור זרימת התקשורת בין כל חלקי הפרויקט.

החלוקה הייתה כדי לעמוד בזמנים ולדעת מה אני עושה בכל שלב, מה שהקל עליי לעמוד בלוח הזמנים. חלוקת האיטרציות נתנה לי זמן גם ללמוד את החומר בצורה מעמיקה ורחבה וגם להספיק להגיש הכל בזמן.

מבוא לשימוש ב-Raspberry Pi

הראסברי פאי הוא מחשב זול המבוסס מערכת הפעלה לינוקס. הכח שלו מוגבל ואינו מתאים לשמש בתור מחשב אישי אך בתור מוצר הראסברי פאי מכשיר מצוין.

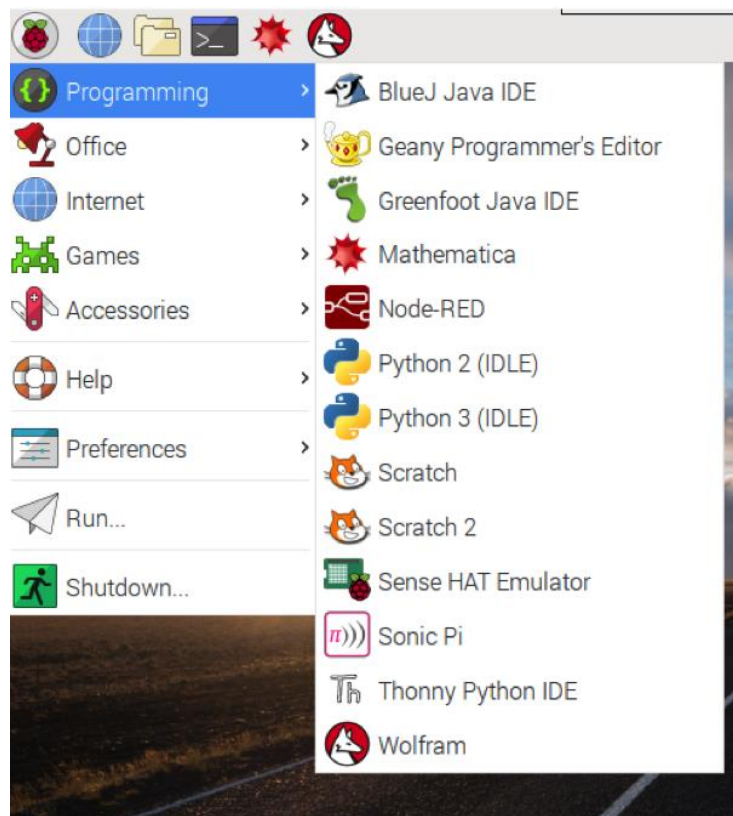
כל ראסברי פאי דורש ספק עם כניסת מיקרו USB שמספר 5 וולט ואמפר גבוה יחסית – 2 אמפר מספיקים בהחלט. לראסברי פאי יש כרטיס זיכרון שעליו נשמרת מערכת ההפעלה וכל המידע מהקבצים.

בשונה ממחשב רגיל יש לו יציאות וכניסות (GPIO) רגלי פלט/קלט כך שאפשר להתקין עליו חיישנים ולהפעיל איתו התקנים חיצוניים כמו מנועים ונורות.

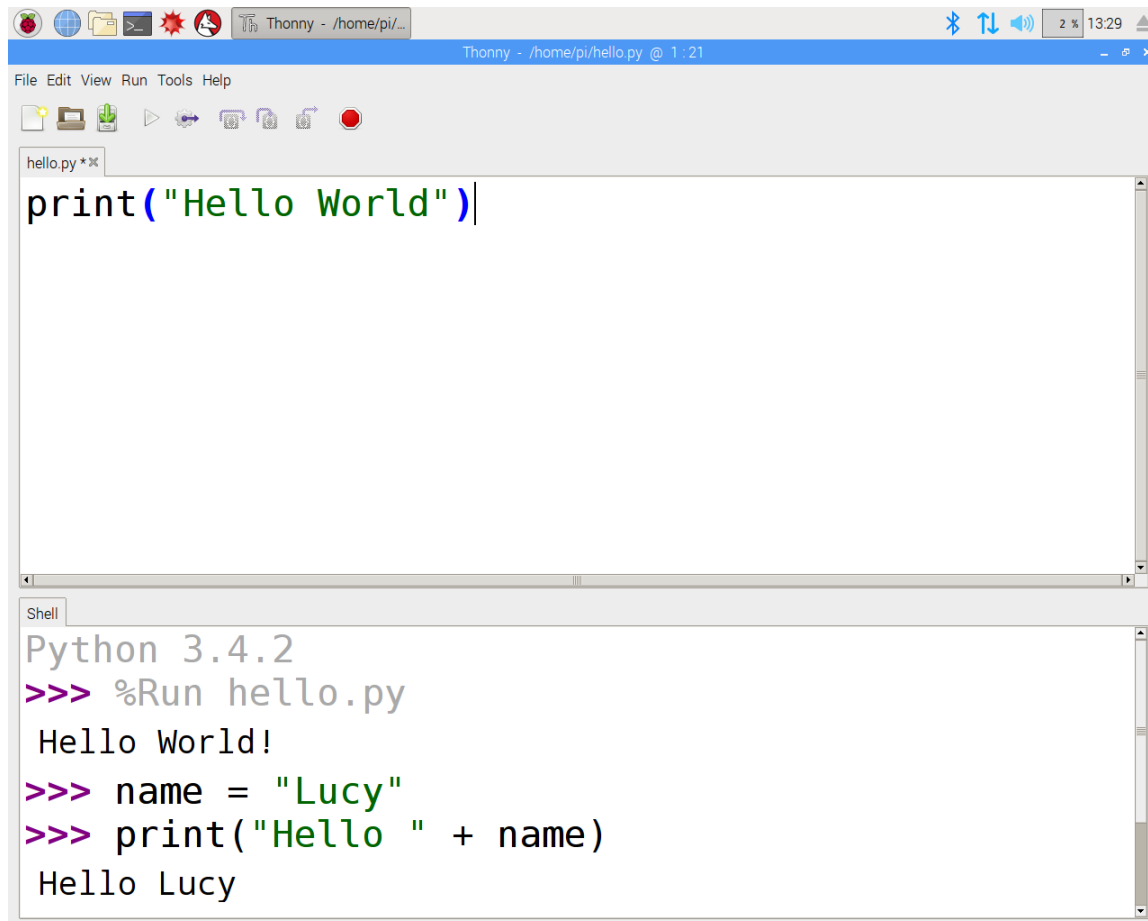
הרצה וכתיבת קוד

לאחר שמערכת ההפעלה של הראסברי פאי מותקנת על המחשב, ניתן להתחיל את כתיבת הקוד. הקוד נכתב בסביבת העבודה Thonny Python IDE, מערכת שאיתה ניתן לכתוב קוד ולהריץ אותו בשפת Python.

כדי להגיע לThonny יש ללחוץ על Thonny Python IDE -> Programming -> Menu



תמונה של המערכת:



```
Thonny - /home/pi/...
Thonny - /home/pi/hello.py @ 1:21
File Edit View Run Tools Help
hello.py *
print("Hello World")

Shell
Python 3.4.2
>>> %Run hello.py
Hello World!
>>> name = "Lucy"
>>> print("Hello " + name)
Hello Lucy
```

בחלק העליון של המסך יש את ה menu bar שאיתו אנחנו יכולים להריץ תכנית, להפסיק אותה באמצע, לשמור אותה ועוד.

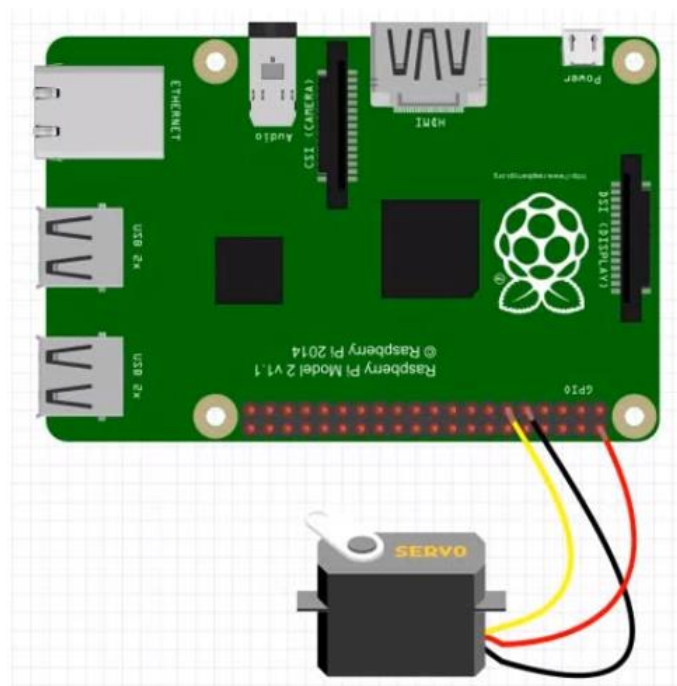
באמצע המסך יש את האיזור שבו אנו רושמים את קוד המשחק ובתחתית המסך יש את התוצאה אותה הרצנו.

הסבר ושימוש בחיישנים

החיישנים שאיתם התעסקתי בפרויקט הם מנוע HS-625MG Servo ומצלמה מסוג Camera Module V2.



למנוע יש 3 פינים המקשרים בינו לבין הראסברי פאי. פין אחד הוא ההארקה לאדמה, ועוד שני פינים בשביל חשמל ובשביל תקשורת עם הראסברי.



שימוש בקוד

כדי להשתמש במנוע יש לייבא את הספרייה RPi.GPIO לפרויקט. לאחר שהספרייה מיובאת, יש "להפעיל" קשר בין הפינים של המנוע לראסברי בעזרת הפקודה

```
GPIO.setup(pinNum,GPIO.OUT)
```

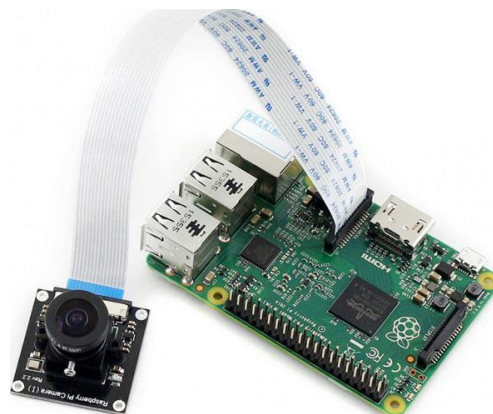
כאשר pinNum מייצג את מספר הפין שאנחנו רוצים לתקשר איתו. פקודה זו מאפשרת תקשורת של הפין לחיישנים במערכת.

לאחר שקישרנו בין הפינים, כדי להזיז את המנוע יש להשתמש בפקודה

```
ChangeDutyCycle(angleNum)
```

כאשר angleNum מייצג את מספר הזווית שאותה אנחנו רוצים לייצג. המנוע יכול להסתובב 180 מעלות. כך שאנחנו מסובבים אותו ל3 זוויות שונות כדי לצלם תצלום מלא של החדר.

כדי לעבוד עם המצלמה של הראסברי פאי, יש לחבר את המצלמה בחיבור המתאים לראסברי (למצלמה אין פינים ויש לה "דף" מסוים שאותו מחברים לראסברי). צריך להפעיל את שימוש המצלמה בהגדרות של הראסברי ולאחר מכן ניתן להשתמש במצלמה.



כדי לעבוד עם המצלמה בשפת פייתון נצטרך להוריד תיקייה בשם picamera ולעבוד יחד איתה. נרצה ליצור אובייקט מסוג Picamera כדי להשתמש במצלמה. נעשה זאת בעזרת הפקודה

```
camera = picamera.PiCamera()
```

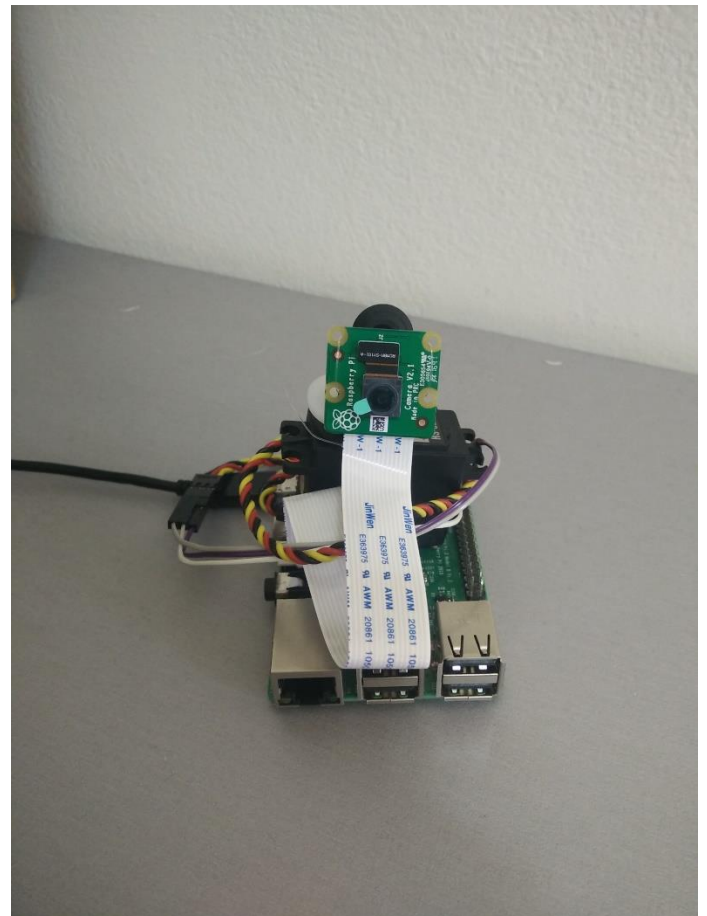
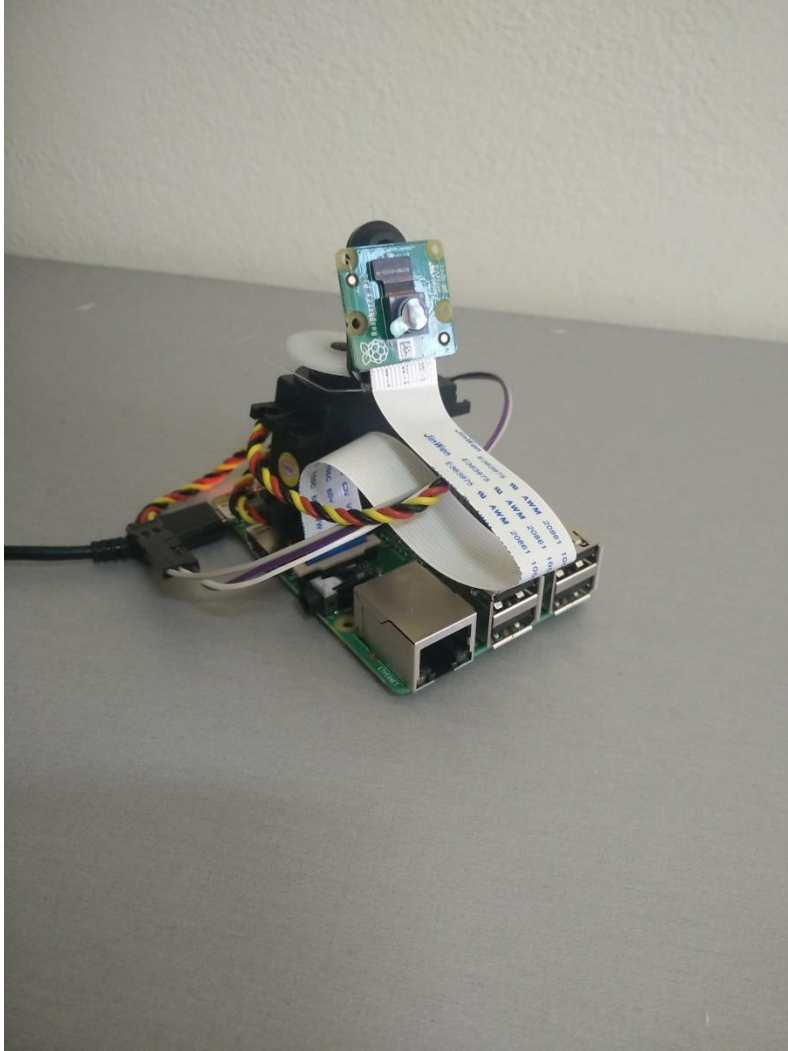
כעת יש לנו אובייקט camera מסוג picamera. כדי לצלם תמונה נשתמש בפקודה

```
camera.capture('image.jpg')
```

הפקודה מצלמת תמונה ושומרת אותה באותה תיקייה של הפרויקט תחת השם שהכנסנו לפונקציה (בדוגמא זו התמונה תישמר בתור image.jpg).

תוצר הרובוט הסופי

תוצר הרובוט הסופי מייצג את החיבור בין המנוע למצלמה לראסברי. שלושתם ביחד עוזרים לנו להשלים משימות שנשלחות לרובוט מהשרת



מבוא לשימוש ב-API

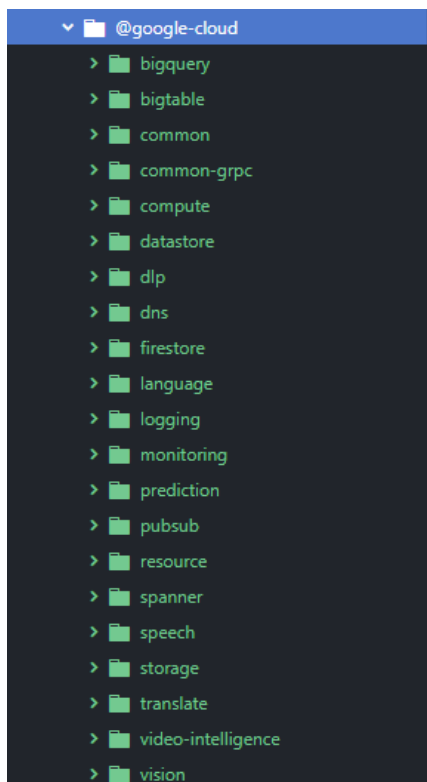
API – Application Programming Interface זה בעצם רשימה של פונקציות שניתנו לכל מיני פרויקטים כדי לחסוך זמן עבודה למתכנתים. כדי להתממשק ל-API של גוגל מסוים יש צורך בהפעלת מפתח להתחברות ל-API.

לאחר שהתחברנו ל-API עם המפתח, יש להעלות את המפתח שבא בקובץ json. לקוד הפרויקט

```
export default {
  "type": "service_account",
  "project_id": "mindful-primer-189620",
  "private_key_id": "2086e1256aeec7a8d6b0a86986539b2aafc8394",
  "private_key": "-----BEGIN PRIVATE
KEY-----\nMIIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYYggSiAgEAAoIBAQDZLpIjPx9EjiS3\n0L/4cLCC/RGVctNgIDT1TtGrIQJ3MDYPDFc8c145ismda0QgOKPL
lToaUe2cwy1urzSdq+QONAKXHAMe6KfYzCMk8\nCv8ydsrKSx0d0sUGZB30qr19Wp8PFoBfoiP284VFkDGWZ/9WgTfk3MbJPC+4qJ\nnciGBHz08ZsDCxkKzhJnzk5
AB\n4k1Uzm5JTyZ4NTyRn1YDd5RDbvXl1WCwOemTCxsJl16SdFvFwotDpFJGlu6lv40\nXZBWJKsvM9ChdaJehpu06A9AS3L5IRo9a0r+Lwi1RC66eyRuVbIZDHqeVe
07s2HDkCgYBb4lfSgbTc62jGQ6t1Gy0m\n6833bSmpZnYXWmzZEE80qh4z4Tlnsi6MZ2IiokwXeCAHIMw5qtVZ0U0ldZudgTD\nnJ76E5MGjgyKrr5Mc7dte9uuhrho
"client_email": "starting-account-gwxeuwmvmvj@mindful-primer-189620.iam.gserviceaccount.com",
"client_id": "108233958679680209887",
"auth_uri": "https://accounts.google.com/o/oauth2/auth",
"token_uri": "https://accounts.google.com/o/oauth2/token",
"auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
"client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/starting-account-gwxeuwmvmvj%40mindful-primer-189620"
}
```

כדי להשתמש ב-API של גוגל בפרויקט בשפת JS יש להוריד את כל הקבצי קוד של גוגל בעזרת

(Node Package Manager) NPM



```
npm install --save @google-cloud/storage
```

לאחר הרצת השורה בתוך CMD עם ה-PATH של הפרויקט, כל קבצי הקוד של google cloud api יעלו לפרויקט ונוכל להשתמש בהם.

לאחר שהתחברנו ל-API's הנחוצים, ניתן להשתמש בכל מיני תוכנות ופונקציות שנחוצות לפרויקט שלנו וגוגל מימשו בעזרת מכונות לומדות שהן פיתחו לדוגמא, בפרויקט זה השתמשתי בפיתוח של גוגל לפירוק אובייקטים מתמונה (google cloud vision api).

```
const Promise = require('promise');

// Imports the Google Cloud client library
const vision = require('@google-cloud/vision');

exports.detect = function(img_path){
  // Creates a client
  const client = new vision.ImageAnnotatorClient();

  return new Promise(function(resolve, reject){
    // Performs label detection on the image file
    client
      .labelDetection(img_path)
      .then(results => {
        const labels = results[0].labelAnnotations;
        let sendLabels = [];
        labels.forEach(label => sendLabels.push(label.description));
        resolve(sendLabels);
      })
      .catch(err => {
        console.error('ERROR:', err);
        reject(err);
      });
  });
};
```

בקוד זה יש יצירת אובייקט מסוג client שיפנה לשרת של גוגל ויתקשר איתו. הclient מקבל path (מיקום לאיפה היא שמורה בתכנית) לתמונה מסוימת והוא שולח אותה בתוך אובייקט promise (שיוצר תגובה סינכרונית לקוד ולא קופץ לקוד אחר) ומחזיר תשובה מהclient לגבי התמונה. התשובה חוזרת בתור אובייקט של labels שהופיעו בתוך התמונה.



```
"responses": [  
  {  
    "labelAnnotations": [  
      {  
        "mid": "/m/0bt9lr",  
        "description": "dog",  
        "score": 0.97346616  
      },  
      {  
        "mid": "/m/09686",  
        "description": "vertebrate",  
        "score": 0.85700572  
      },  
      {  
        "mid": "/m/01pm38",  
        "description": "clumber spaniel",  
        "score": 0.84881884  
      },  
      {  
        "mid": "/m/04rky",  
        "description": "mammal",  
        "score": 0.847575  
      },  
      {  
        "mid": "/m/02wbgd",  
        "description": "english cocker spaniel",  
        "score": 0.75829375  
      }  
    ]  
  }  
]
```

האובייקטים שיצאו מהתמונה הזו יהיו:

מכאן ניתן לראות שהAPI מחזיר גם את הדבר המרכזי בתמונה (שזה הכלב) ואפילו את הסוג שלו (קוקר ספנייל אנגלי) וגם את הסוג של החיה (יונק) וכו'.

מסמך ייזום

מטרת הפרויקט

נועד על מנת להקל על משימות יומיומיות של אנשים כגון חיפוש ברחבי החדר ולאפשר להם להשתמש ברובוט בשביל משימות שהם לא יכולים לעשות באותו רגע. האדם מדבר אל הרובוט ומבקש ממנו למצוא בשבילו חפץ אשר אמור להימצא ברחבי החדר.

סוגי משתמשים / קהל היעד של היישום

אנשים בעלי צרכים ומוגבלויות יוכלו להשתמש ברובוט על מנת להקל על אורח חייהם. וגם כלל האוכלוסייה תוכל להשתמש ברובוט על מנת להקל במציאת חפצים במרחב.

פיצ'רים ותהליכים עיקריים

- הסתכלות לכיוון של אובייקטים
- שמיעה דרך המיקרופון בפלאפון
- ראייה של האובייקטים
- תקשורת עם השרת
- קבלת הוראות מהשרת
- קבלת הוראות מהלקוח
- הסבר של הרובוט בקול איכן נמצא הפריט המבוקש

טכנולוגיות עיקריות ושיקולים עיקריים

תכנות בnode.js (בשביל השרת)
תכנות בPython (בשביל הראסברי)
שימוש בטכנולוגיית שרת-לקוח
http server & requests
שימוש בראסברי פאי
שימוש בAPI
המערכת תפנה לשרת ב requests http, ודרך השרת לAPI

ידע, סיכונים ושאלות

פלטפורמות שאני לא מנוסים בהן

שימוש בראסברי פאי

תכנות בnode.js

פלטפורמות שאנחנו מנוסים בהם

תכנות בPython

הקמת שרת

שימוש בAPI

android studio

בניית אפליקציות

לדעתי החלק שייקח הכי הרבה זמן הוא להבין כיצד יש לבנות ולהרכיב את הרובוט ואיך ליצור תקשורת בין מנוע הרובוט לבין הרaspberry.

דרישות חומרה

rasberry pi 3

חיישן מצלמה

חיישן מיקרופון

מנוע להזזת ראש המצלמה

כעת אין לי ניסיון בהתעסקות עם raspberry pi 3, עם החיישנים ועם המנוע והתקשורת בין שלושת הדברים.

פתרונות קיימים

קיימים עוד רובוטים שמנסים לעזור לאוכלוסייה המבוגרת יותר כמו:

<https://news.walla.co.il/item/530043>

<https://www.geektime.co.il/pr2-personal-robot/>

מסמך אפיון

תהליך ראשי:

הלקוח מקליט את הפקודה באפליקציה אשר בטלפון ושולח את ההקלטה אל השרת, השרת מעבד את המידע, מפרק למילים ושולח ל-API של google פיד שצולם במצלמה של הרובוט ונשלח לשרת ואת שם האובייקט המבוקש. ה-API מחזיר תשובה האם האובייקט נמצא בפיד המבוקש. סריקת הסביבה של הרובוט היא בעזרת סיבוב של 90 מעלות בכיוון השעון של המצלמה, ושליחת פיד של המצלמה לשרת יחד עם השם של האובייקט שאותו אנו מחפשים. ברגע שאובייקט נמצא באחד הפידים ששלחנו, אנו יודעים שהרובוט מסתכל לכיוון הנכון, כעת נשאר רק לכוון את ראש הרובוט לכיוון האובייקט, ואת זה אנו עושים גם כן בעזרת ה-API.

חלקי הרובוט:

- חיישן מיקרופון
- מצלמה
- רסברי פאי
- גלגלים - שני גלגלים קדמיים, אחד אחורי. משתמשים בגלגלים על מנת שבעתיד נוכל להמשיך לפתח את הרובוט ולהזיזו אל עבר החפץ המבוקש.
- המצלמה תהיה בראש הרובוט, כדי שהרובוט יוכל לצלם את חלל החדר מגובה מסוים ודרך כך לקלוט את החפצים שבסביבתו.
- המצלמה תסתובב על ציר ותסרוק את החדר בעזרת 4 תמונות
-

חיבור הלוח לרשת ולשרת

חיבור לוח הראסברי לשרת יהיה דרך שליחת http post request ל-http server עם ה-ip של הראסברי. לאחר שהשרת ישמור את ה-ip של הראסברי. הראסברי שולח את הפיד של התמונה ואת הפקודה (חיפוש אובייקט מסוים) לשרת והשרת יוצר בקשה שנשלחת ל-API של גוגל, שאיתו אנחנו משתמשים כדי לזהות את האובייקט המדובר. לאחר שזיהינו האם האובייקט נמצא בתמונה הנוכחית, השרת שולח פקודה לראסברי והראסברי מבצע את הפקודה שהשרת שלח.

הקמת שרת

השרת יחכה שה raspberry ישלח לו את ה ip שלו. לאחר מכן יחכה השרת שהאפליקציה תשלח לו את קובץ האודיו שמכיל את הפקודה. השרת יפענח את ההוראה שנשלחה (בעזרת אלגוריתם שנקרא speech to text שממיר קובץ אודיו למילים).. לאחר מכן, השרת ישלח הודעה ל raspberry ויבקש פיד (כלומר תמונה מהזווית הראשונה בה נמצאת המצלמה). לאחר קבלת הפיד מחיישן המצלמה, השרת יבדוק האם ישנה התאמה בין החפץ המבוקש לבין החפצים שבתמונה. אם אכן החפץ נמצא בפיד השרת יחשב כמה מעלות צריך להזיז את ראש הרובוט. לאחר מכן, השרת בונה הוראה לראסברי ומחזיר את ההוראה כדי שהרובוט יוכל להסתכל ישירות על האובייקט. במידה ולא, יבקש השרת מהראסברי לסובב את ראש המצלמה בכתשעים מעלות לכיוון ימין במידה ולא צולמו כבר ארבע תמונות כלומר לא צולם כל המרחב. אם החפץ לא ימצא באף אחת מתמונות המרחב ישלח השרת הודעה לרובוט פקודה להשמיע בעזרת חיישן המיקרופון הודעה המבשרת כי החפץ לא נמצא במרחב הקרוב.

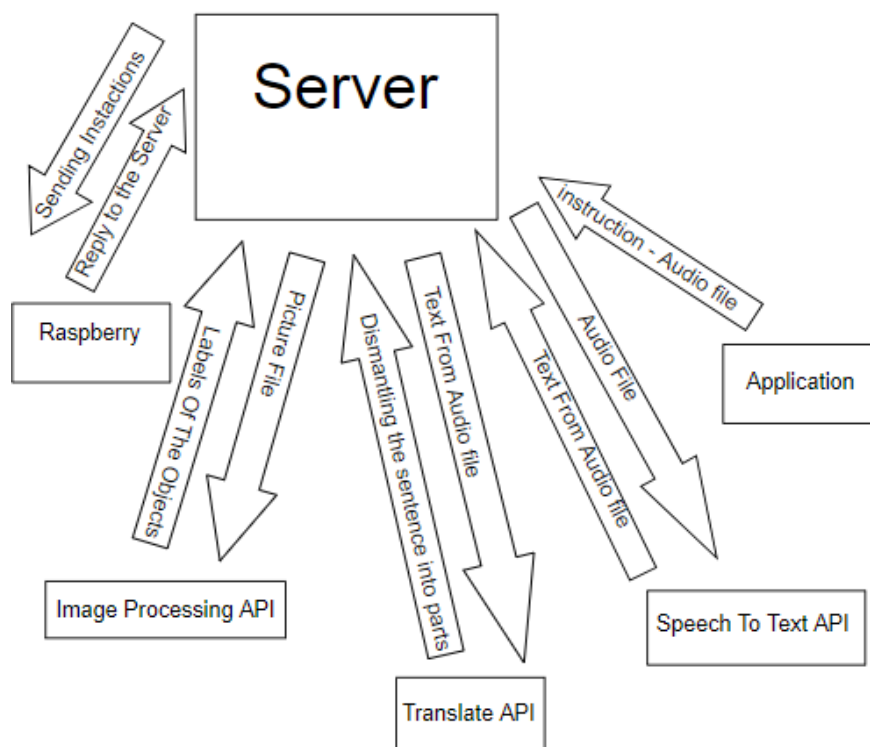
תהליך פירוק הפקודה מהמיקרופון

לאחר שהאפליקציה תקלוט את הפקודה מהמיקרופון, המיקרופון ישלח את ההקלטה לשרת בתור קובץ אודיו, והשרת יפענח את הקובץ בעזרת Google Cloud speech API שיהפוך את קובץ האודיו למילים ולאחר מכן יחפש את שם העצם שבמשפט אשר הוא החפץ המבוקש.

תהליך קליטת האובייקטים מהמצלמה

כל פיד שהמצלמה תצלם ישלח לשרת והפיענוח של התמונה יבוצע בעזרת Google Cloud Vision API, שמוציא מידע מהתמונה ושולח בחזרה לשרת. המצלמה תסרוק את חלל החדר בתמונות, עד שהאובייקט יימצא. אם המצלמה סרקה את כל החדר והאובייקט לא נמצא, הרובוט יחזיר תשובה ללקוח

מסכים והמחשה ויזואלית



דרישות שאינן פונקציונליות

התקשורת בין הרובוט לשרת ובין השרת תתבצע דרך `http server` הרובוט ישלח את הפיד דרך המצלמה לשרת ואת האודיו שהמיקרופון הקליט, לאחר תהליך עיבוד הנתונים, השרת ישלח לרובוט פקודות לאיזה כיוון להזיז את הראש. יש צורך ליצור תקשורת מאובטחת כדי למנוע פריצות והתקפות שיוכלו לשנות את מטרת הרובוט ולשמור כי התמונות הנשלחות לא יועברו ויפצו למקום זדוני אחר שיוכל לנצלם לרעה.

מסמך ארכיטקטורה

מבט על

- Networking רכיב תקשורת - אחראי על ביצוע התקשורת מול הרשת (שימוש ב-google api), המידע המתקבל נשלח לרכיב הלוגיקה. כולל טיפול בתקלות תקשורת
- Secutiry רכיב אבטחה - אחראי לכך שהתקשורת ברשת והעברת המידע תהיה מאובטחת. אחראי לדאוג שהתקשורת מול כל לקוח תהיה מאובטחת ברמה גבוהה, שתמנע התחזות, זיוף או חזרה של פקטות מידע על ידי צד שלישי עוין. על מנת לדאוג שהמתרחש בבית הפרטי ישאר חסוי ומוגן.
- Camera Wrapper רכיב תצלום - אחראי על תקשורת נכונה מול המצלמה שברובוט
- microphone Wrapper רכיב שמע - אחראי על תקשורת נכונה מול המיקרופון שברובוט
- Logic רכיב לוגיקה - אחראי על עיבוד הקלט מרכיב השמיעה והתצלום. אחראי גם על עיבוד התשובות המתקבלות מהשרת JS (ברכיב התקשורת) וכתיבת פקודה מתאימה להצגת הממצאים ללקוח. יפותח בשפת python.
- Hardware Wraper רכיב חומרה - תפקידו לעטוף באופן נח את השימוש ברכיבי החומרה המחוברים לרובוט (raspberry pi3).

עיצוב הנתונים וישויות מידע

- קובץ אודיו - מכיל את ההוראות שהלקוח נתן דרך האפליקציה, לאחר שהמשתמש לחץ על כפתור ההקלטה שבאפליקציה ושלח אותה אל השרת.
 - פיד - מכיל קובץ של תמונה שצולם באמצעות חיישן המצלמה, ונשלח לשרת ובעזרת הקובץ הזה השרת מפענח אם האובייקט נמצא בתמונה או לא
- אלו הן יישויות אשר גם מועברות בין רכיבים שונים במערכת, גם בין הלקוח/הרובוט/האפליקציה לשרת (על גבי הרשת).

טכנולוגיות עיקריות

שפות התכנות בהן אשתמש בפרויקט הן: python ו- java script
אשתמש בpython משום שזוהי הדרך הקלה והטובה ביותר לתקשר עם raspberry pi
שהוא רכיב החומרה.

אני משתמש בשירותים החיצוניים - בAPI של google :

Google Cloud speech

ו- Google Cloud vision

איתן נוח וקל לתקשר באמצעות java script ובאמצעות http request/server .
בחרתי להשתמש בhttp server ולעשות זאת דרך js לתקשורת בין הרכיבים השונים משום
שניסיתי תחילה לבנות את התקשורת באמצעות socket אך נתקלתי בכמה בעיות והבנתי כי
הדרך הקלה ביותר לתקשר היא באמצעות http שהיא בעצם הדרך בה עובד ופועל האינטרנט
ובה נשלחות בקשות ותשובות ברשת.

אני מרגיש מנוסה בתכנות בשפת python ובשימוש בAPI חיצוניים.
אינני מכיר עדין את השימוש והתפעול של raspberry pi אבל התנסיתי בעבר בעבודה עם
androino.

חשבתי להשתמש בשפת c אבל לא מצאנו מאגר גדול של ספריות שניתן להשתמש בהם
למטרתינו.

לאחר מכן התלבטנו בין ++c לpython אבל הבנו שב-python יש יותר ספריות פתוחות וזוהי
שפה שאנו מנוסים בה.

התאמה לאפיון

<u>פצ'ר</u>	<u>רכיבים רלוונטים</u>
שמיעה דרך המיקרופון	שימוש במיקרופון-<לוגיקה-<תקשורת+אבטחה
ראייה של האובייקטים	שימוש במצלמה-<לוגיקה-<תקשורת+אבטחה
הסבר של הרובוט בקול איכן נמצא הפריט המבוקש	לוגיקה-<שימוש במיקרופון-<תקשורת+אבטחה
הסתכלות לכיוון של האובייקטים	לוגיקה-<תקשורת עם רכיבי החומרה-<תקשורת+אבטחה
קבלת הוראות מהלקוח	שימוש במיקרופון-< לוגיקה-<תקשורת+ אבטחה
קבלת הוראות מהשרת	תקשורת+אבטחה-<לוגיקה

איטרציות

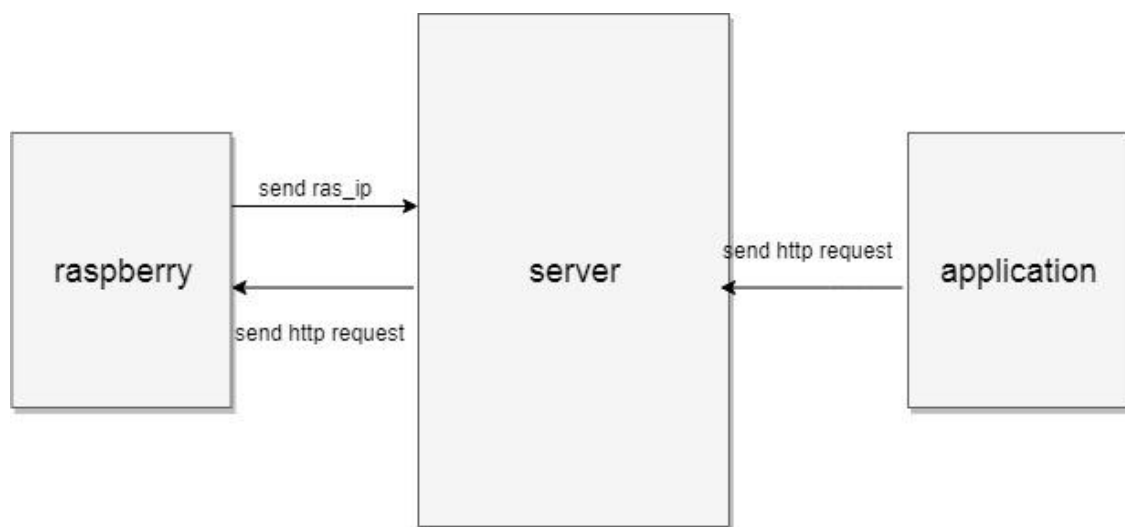
איטרציה 1 – הקמת שרת ותקשורת בסיסית

תקציר, הסבר ותוצר:

- מכיוון שזו איטרציה ראשונה, והיא מתמקדת בעיקר בפיתוח לפורמה שאין לנו נסיון בה, היא מוקדשת כולה ללמידה מעמיקה של הנושא
- התוצר של איטרציה זו יהיה שרת בסיסי שיודע לקבל בקשות http מהאפליקציה, יודע את ה ip של ה raspberry ויודע לשלוח בקשות http לרובוט.

תכולות טכנולוגיות:

- תכנון השרת הבסיסי שיודע לתקשר עם האפליקציה ועם ה raspberry:
 - השרת יחכה לפניה מהאפליקציה ולשליחת קובץ האודיו
 - בו זמנית יחכה גם השרת לקבלת ה ip של ה raspberry
 - המחשה:



- אם ישאר זמן:
 - פיתוח האפליקציה והוספת אופציה להקלטה
 - פיתוח האפליקציה והוספת אופציה לשליחת ההקלטה

קשיים ומקרי קצה שעשויים להתקל:

- ניסיון פניה מהאפליקציה ומהרובוט בו זמנית
- שליחת פקודה מהאפליקציה אל השרת לפני שהשרת קיבל את ה ip של הרובוט
- איבוד חלק מן ההקלטה בעת השליחה מהאפליקציה

איטרציה 2 – הכרה ולמידה של Google Cloud API's

תקציר, הסבר ותוצר:

- מכיוון שאני מכיר ומנוסה בשימוש ב-google cloud api's , איטרציה זו מוקדשת להכרתן וללימודן בצורה מעמיקה ויסודית.
- המטרה היא כי לאחר איטרציה זו נלמד איך לקרוא ל-api's איך לקבל מהם מידע, איך להשתמש בצורה חכמה ובטוחה במידע שקיבלנו ואיך להתמודד עם שגיאות , בעיות ותקלות הקשורות לתקשורת עם ה-api's.
- נשחק במעט עם ה-api's כדי לראות כיצד משתנות התשובות בעת שינים קטנים וגדולים כאחד ועל מנת להבינם עד היסוד והשורש.

תכולות טכנולוגיות:

- מאחר ואיטרציה זו מוקדשת ללמידה מקדימה לעשייה באיטרציה זו כמעט ולא ייווצרו תכולות טכנולוגיות ממשיות איך היווצר הבסיס להן.
- לאחר שאלמד להשתמש ב google cloud speech to text api אצור אלגוריתם בסיסי שישלח ויקבל את המידע.

איטרציה 3 – קליטת קול וניתוח תמונה

תקציר, הסבר ותוצר:

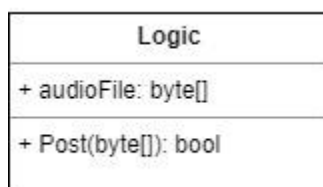
- מטרת האיטרציה היא לספק צינור בין המשתמש לשרת, כלומר אפליקציה בסיסית שיודעת להקליט ולשלוח את ההקלטה לשרת
- אני מקווה שאספיק גם את תהליך המרת ההקלטה למילים ופירוקם לחלקי המשפט

תכולות טכנולוגיות:

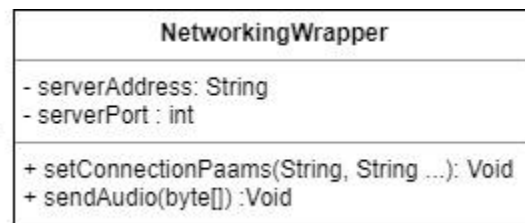
- אפליקציה המכילה מסך עם אפשרות להקלטה
 - משמעות הדבר היא כמובן גם מחלקת Activity חדשה, אשר תפעיל את הפונקציות המתאימות במחלקת הלוגיקה (כפי שיפורט בסעיף הבא של התכולות) ותציג חיווי ויזואלי למשתמש בהתאם לכך
 - למשתמש יוצג באפליקציה כפתור המאפשר הקלטה בעת הלחיצה עליו, מיד לאחר עזיבת האצבע מן הפתור תישלח ההקלטה אל השרת.
 - המחשה ויזואלית בסיסית:



- מחלקות חדשות בצד הלקוח - כולל ממשק, פונקציונליות, תקשורת ומידע
 - מחלקה אשר אחראית לניהול התקשורת מול השרת
 - מחלקה אשר אחראית ללוגיקה שמאחורי ממשק המשתמש, ובונה/מעבדת את המידע שנשלח/מתקבל מהשרת
 - המטרה בכך היא להפריד בין שכבות שונות, באופן אשר מאפשר להן להשתמש אחת בשניה אך מבלי להיות תלויות אחת בשניה בכל שינוי קטן
 - המחשת UML של הבסיס למחלקות החדשות:



Use



- פונקציונליות ולוגיקה חדשה בצד השרת שמטרתה לפענח את ההקלטה למילים
 - מחלקה אשר אחראית לתקשר עם google cloud speech to text api.
 - מחלקה אשר אחראית לפרק את הטקסט שהתקבל למילים יחידות ולזהות את החלק המשפטי של כל מילה: שם עצם, פועל וכו'...
 - מחלקה אשר אחראית לזהות את שם העצם המבוקש אותו יש לחפש ברחבי החדר.
 - המחשת UML של הבסיס למחלקות החדשות:

speech to text
+ audio_file : byte[] + text : string
+ convert(byte[]) : string + analyze(string) : dictionary + find_required_noun(dictionary) : string

איטרציה 4 – פעולות הרובוט על פי הממצאים וניתוחם

תקציר, הסבר ותוצר:

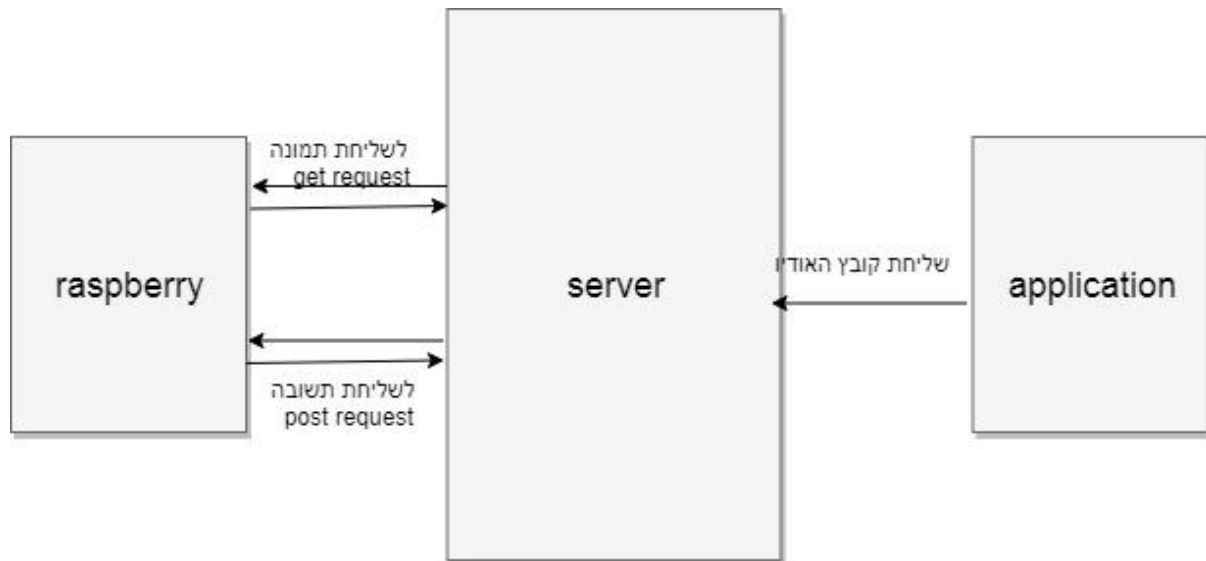
- מטרת האיטרציה היא לייצר שרת שיודע לנתח את ההקלטה המגיעה אליו ופועל בהתאם עליה, רובוט שיודע לפעול בהתאם לפקודת המגיעות אליו מן השרת ושרת שיודע לנתח את התמונה שנשלחה אליו מהרובוט.
- השרת אמור לאחר איטרציה זו להיות מסוגל לנתח את קובץ האודיו ולפנות אל הרובוט על פי ממצאיו. לאחר התשובה מהרובוט
- והרובוט מצידו יהיה מסוגל לקחת תמונות, לסובב את ולסרוק את כל המרחב.
- לאחר איטרציה זו כל חלקי הפרויקט יזרמו באופן שוטף וכל שישאר הוא להעניק ללקוח באמצעות האפליקציה את התשובה הסופית.

תכולות טכנולוגיות:

- בקשה מהרובוט לתמונה- לאחר שהשרת קיבל מהאפליקציה את ההקלטה, המיר אותה לקובץ טקסט, ניתח אותו ומצא את שם העצם אותו יש לחפש ברחבי החדר, שולח השרת פקודה לרובוט ומבקש פיד אחד מהמצלמה.
- תקשורת בטוחה ומאובטחת בין השרת לרובוט- הרובוט משמש בעצם כ http server , אליו שולח השרת בקשות get ו post .

- get request - על מנת לבקש תמונה.
- post request - על מנת לשלוח לרובוט הודעה האם יש לשלוח פיד נוסף או לחילופין האם האובייקט המבוקש אכן נמצא.

• נמחיש זאת באופן גרפי בעזרת דיאגרמה:



- ניתוח התמונה- לאחר שהשרת ביקש וקיבל את תמונה מזווית אחת של החדר , השרת מנתח אותה. בעזרת פנייה ל google cloud vision api מוצאים את האובייקטים שנמצאים בתמונה , כלומר במקרה שלנו החפצים שנמצאים במרחב.
- הצלבה בין הממצאים- לאחר שגילנו מהו הפריט אותו מבקש הלקוח בעזרת ההקלטה ואת הפריטים הנמצאים בזווית מסוימת של החדר בעזרת התמונה , השרת מבצע השוואה ובודק האם הפריט המבוקש נמצא בתמונה. במידה וכן מחזיר הוא תשובה ללקוח ומודיע כי הוא הצליח במשימתו ומצא את החפץ אותו ביקש הלקוח. במידה ואין התאמה בין שני הדברים, בודק הרובוט האם צילם כבר את כל החדר או שרק את חלקו. במידה וצילם את כל החדר מחזיר הוא תשובה ללקוח כי החפץ אינו נמצא בסביבתו וכמו כן מחזיר גם הוא תשובה ל raspberry. במידה והוא עוד לא צילם וסרק את כל החדר , מבקש השרת מהרובוט שיסובב את המצלמה ב 90 מעלות ויקח תמונה נוספת, וכך חוזר חזרה לניתוח התמונה ולהצלבה.

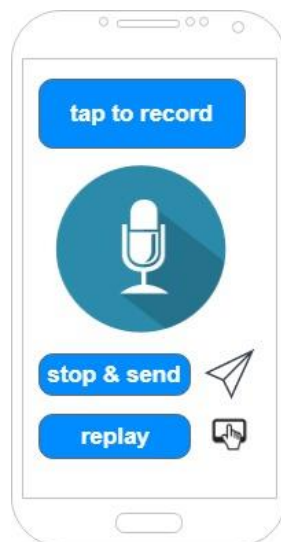
איטרציה 5 – פעולות האפליקציה על פי הממצאים ושיפורים

תקציר, הסבר ותוצר:

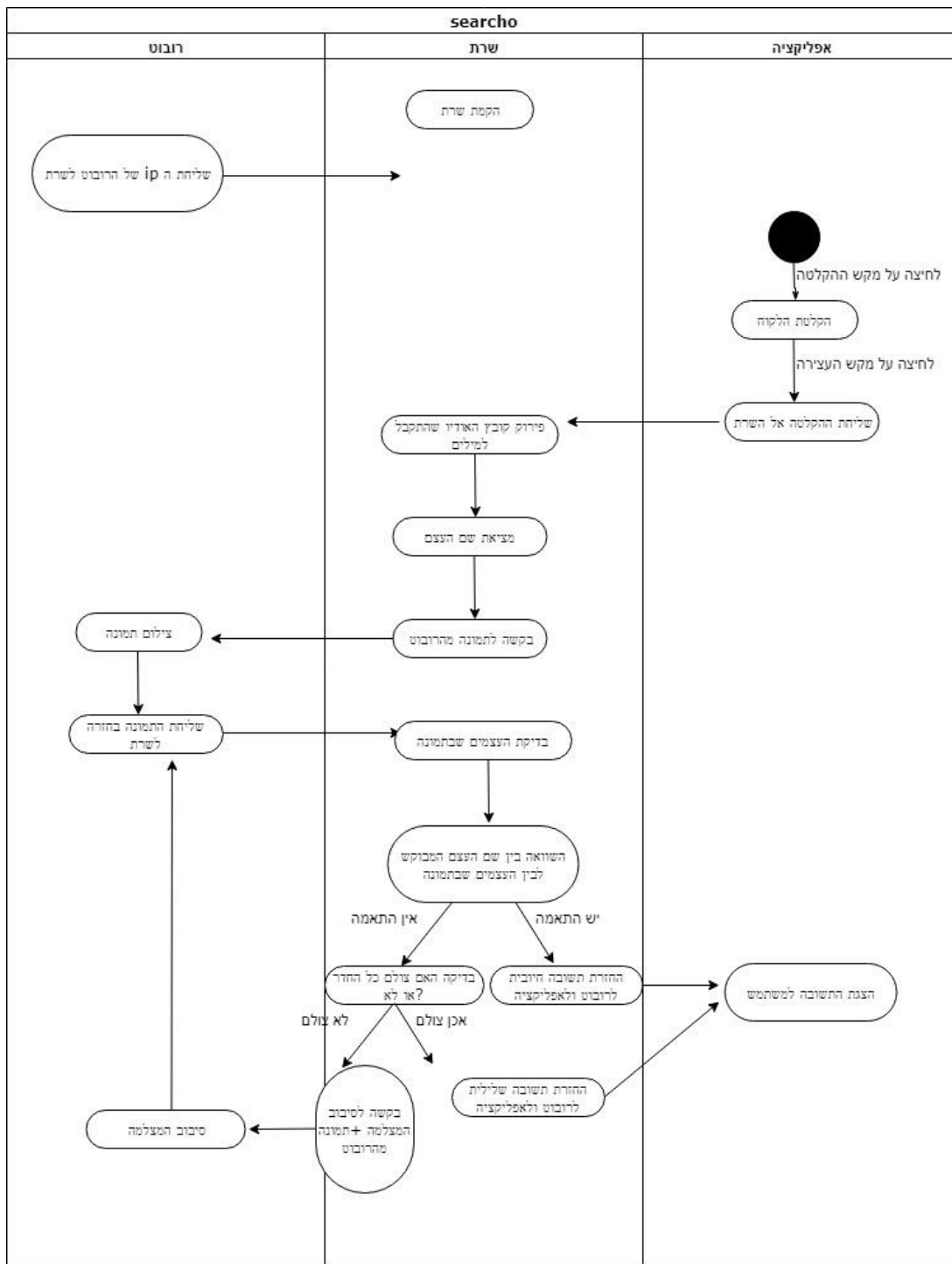
- מטרת האיטרציה היא לחבר ולשלב את כל חלקי הפרויקט לפרויקט מוגמר.
- השרת אמור לאחר איטרציה זו להיות מסוגל לתקשר תקשורת מלאה ורציפה עם הלקוח דרך האפליקציה ועם הרובוט.
- לאחר איטרציה זו כל חלקי הפרויקט יזרמו באופן שוטף, והלקוח יוכל לבקש את בקשותיו מהרובוט ולקבל את תשובותיו באופן מלא, מהיר וטוב.
- לאחר איטרציה זו יהיה למשתמש צינור נוח וקל לתפעול - האפליקציה בה יוכל להקליט את דבריו, לשלוחם לשרת ולהשמיע אותם שמיעה חוזרת לעצמו.

תכולות טכנולוגיות:

- בקשה מהרובוט לתמונה- לאחר שהשרת קיבל מהאפליקציה את ההקלטה, המיר אותה לקובץ טקסט, ניתח אותו ומצא את שם העצם אותו יש לחפש ברחבי החדר, שולח השרת פקודה לרובוט ומבקש פיד אחד מהמצלמה.
- אפליקציה משופרת המכילה מסך עם אפשרות להקלטה, שליחה והשמעה חוזרת
 - משמעות הדבר היא כמובן גם מחלקת Activity חדשה, אשר תפעיל את הפונקציות המתאימות במחלקת הלוגיקה (כפי שיפורט בסעיף הבא של התכולות) ותציג חיווי ויזואלי למשתמש בהתאם לכך
 - למשתמש יוצג באפליקציה כפתור המאפשר הקלטה בעת הלחיצה עליו
 - ההקלטה תשלח אל השרת לאחר לחיצה על כפתור ה- send & stop.
 - המחשה ויזואלית בסיסית:



- הצגת התשובה למשתמש- לאחר שהשרת ניתח את התמונה והשווה בינה לבין דברי הלקוח, שולח השרת תשובה חיובית או שלילית על הימצאות החפץ המבוקש אל האפליקציה. האפליקציה תציג את התשובה אל המשתמש ותבשר לו את בשורותיו.
- באופן סופי אמור הflow להיראות באופן הבא:



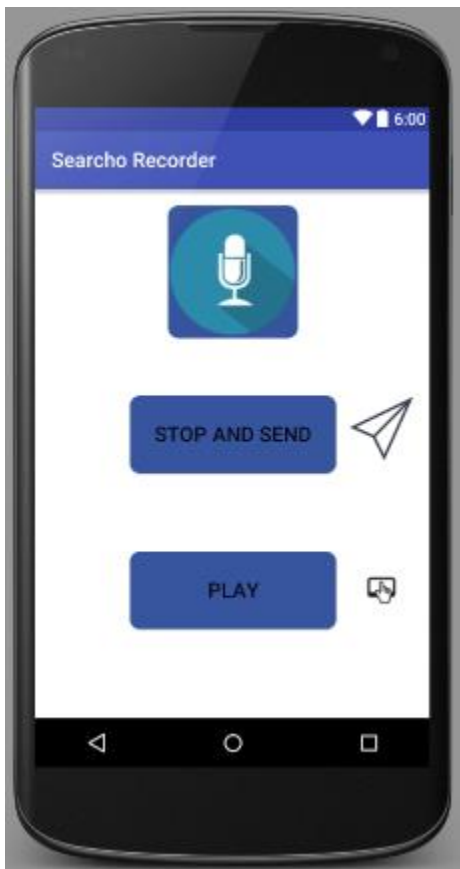
מדריך לממשק קצה

הסבר בסיסי על התוכנית

תוכנית זו מסוגלת לחפש פריט במרחב במקום האדם. משום כך, היא מהווה פתרון מצוין למקרים בהם אדם אינו מסוגל או שאינו מצליח למצוא את החפץ לו הוא חפץ. התוכנית מופעלת באמצעות אפליקציה, שרת חיצוני ורובוט השוכן בחדר המשתמש. התוכנית מסוגלת לזהות שפות רבות ומגוונות כגון: אנגלית, ספרדית, ערבית ועוד.

הוראות הפעלה

לפני הפעלת האפליקציה יש להתקינה על הטלפון. לאחר שהאפליקציה הותקנה על המכשיר הסולרי יש ללחוץ על לוגו האפליקציה ולהפעילה. לאחר שהאפליקציה עלתה והתוכנית רצה, מופיעה המסך הראשי.



במסך הראשי מופעים שלושה כפתורים:

1. **RECORD** – בעת לחיצה על כפתור זה, תתחיל האפליקציה להקליט את המשתמש.
2. **Stop And Send** - בעת לחיצה על כפתור זה, תפסיק האפליקציה את הקלטת המשתמש ותשלח את ההקלטה אל השרת לעיבודה וניתוחה.
3. **Play** - לחיצה על כפתור זה, ניתן לשמוע את ההקלטה האחרונה אותה ביצע המשתמש.

על מנת לבקש מהרובוט לחפש את החפץ אותו אתה רוצה למצוא ברחבי החדר, יש ללחוץ על כפתור ה-RECORD. מיד לאחר זאת יש להגיד באופן ברור ועם דיקציה כשפיכם קרוב למיקרופון שבטלפון והמיקרופון איננו חסום את בקשתכם. לדוגמה: "Searcho, find my keys". לאחר שסיימת להגיד את בקשתך יש ללחוץ על כפתור ה-STOP. ברגע זה תישלח ההקלטה מן האפליקציה אל השרת לניתוחה ועיבודה. אם ברצונך לשמוע שוב ולוודא כי דיברת בצורה ברורה ואמרת את הדברים אותם רצית לאמור ולהם התכוונת, יש ללחוץ על כפתור ה-PLAY ולהקשיב.

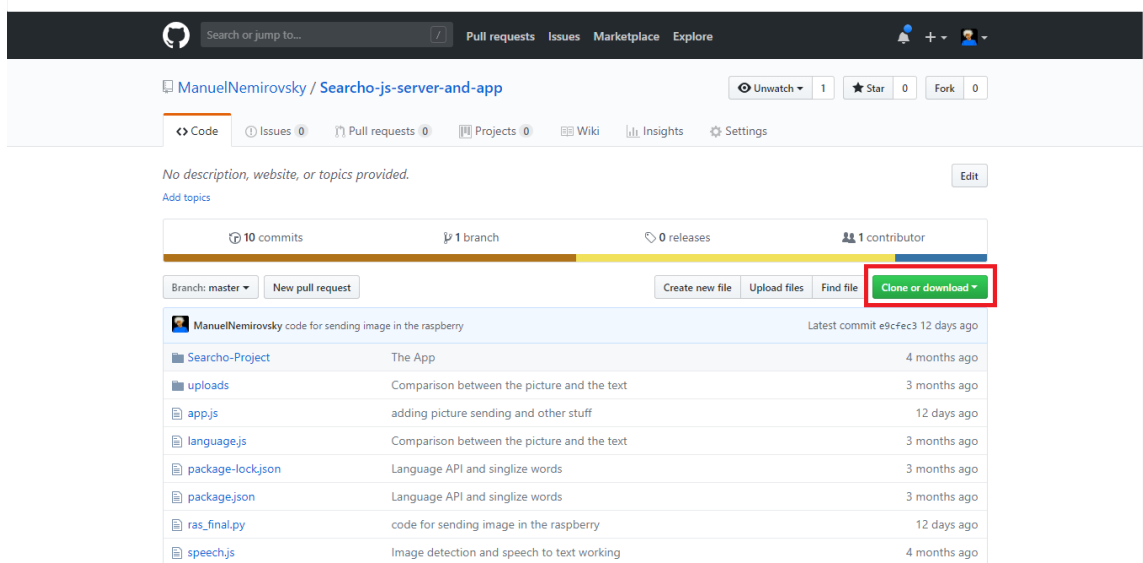
לאחר שהשרת יעבד וינתח את המידע הוא יבקש מהרובוט לצלם למענו את החדר על מנת שיוכלו לעבדו ולסרוקו גם כן. תוכל לראות ולשמוע את הרובוט לוקח תמונה ואולי אף מסובב את ראשו(מצלמתו). לבסוף יחזיר לך הרובוט תשובה האם החפץ אותו ביקשת אכן שוכן בחדרך, מקום הימצאך.

בעתיד מקווים אנו כי נשפר ביצענו והרובוט יספר לך גם במילים בעזרת המיקרופון שיוסף לו בעתיד את ממצאיו ותוצאותיו, על מנת להקל עליך ולעזור לך כמה שיותר.

מדריך לממשק ניהול

היכנס לכתובת : <https://github.com/ManuelNemirovsky/Searcho-js-server-and-app>

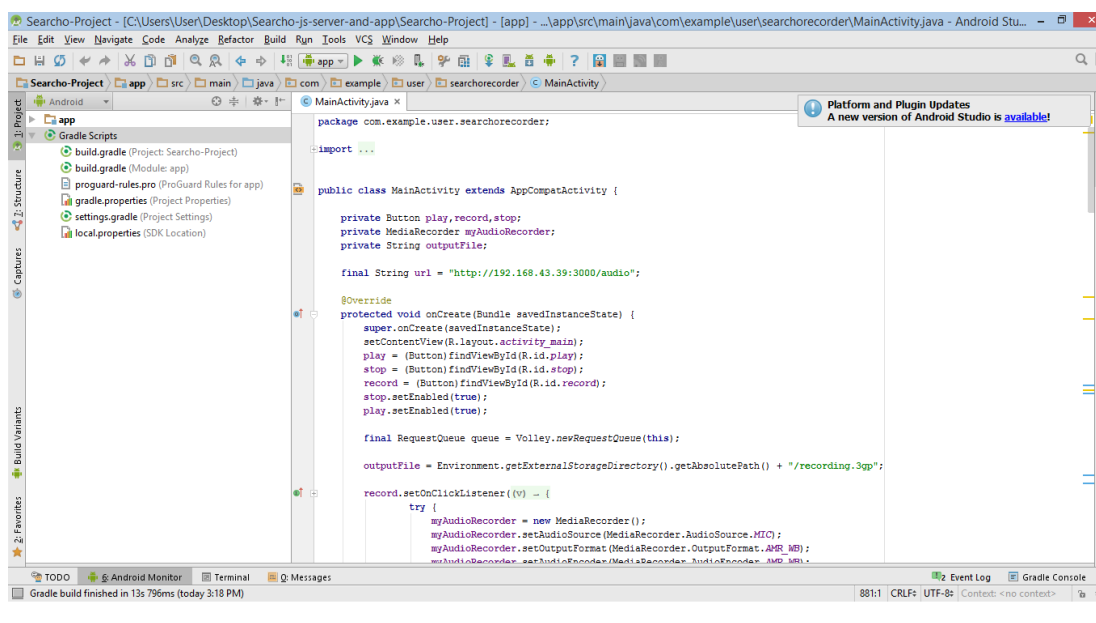
שם לחץ על : clone or download



התקנת האפליקציה:

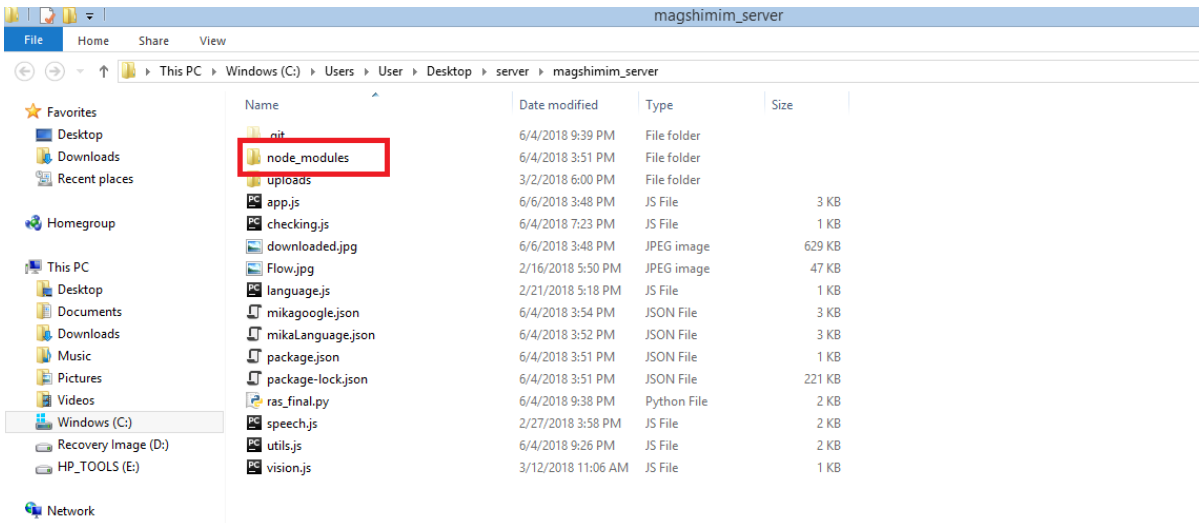
יש להיכנס לתיקייה searcho-project ולהריץ את התיקיה תחת android studio על פלאפון

אנדרואיד מסוים.

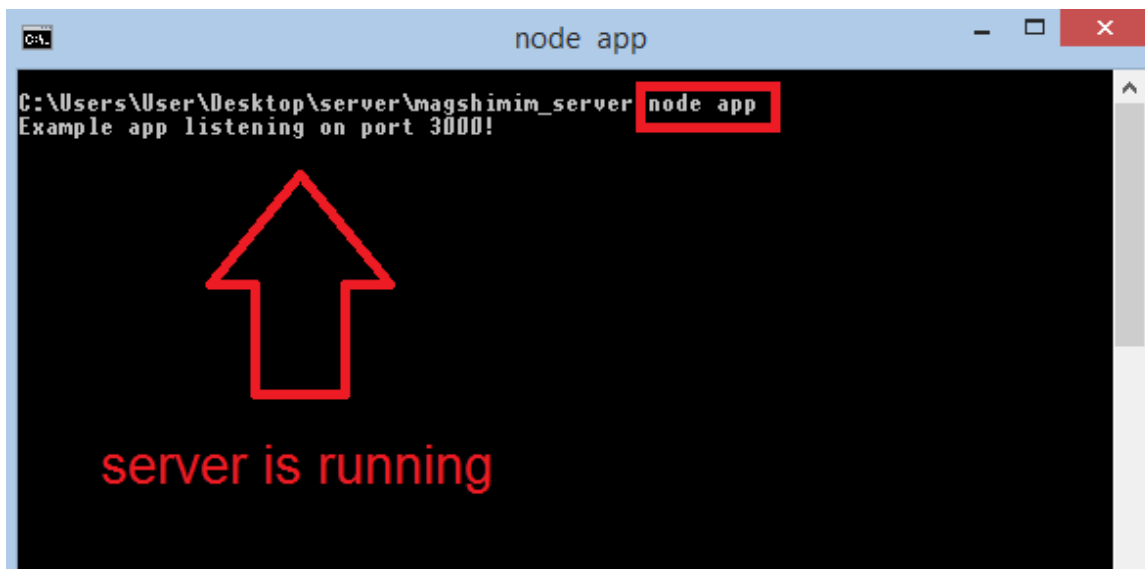


התקנת השרת:

התקנת השרת תבוצע דרך `node` יש להכניס את כל הקודים בJS לתיקייה אחת ובה להתקין את הnode modules שמפורטים בדף ההסבר בgithub ולהכניס `ccredentials` מתאימים של `google vision, speech and natural language API's`



לאחר שכל הדברים הותקנו, יש לכתוב `node app` על התיקייה של הפרויקט דרך `cmd`.



התקנת והפעלת הקוד של rasperry (הרובוט):

דרך הראסברי יש להריץ את הפרויקט ras_pinal.py שנמצא בgithub דרך terminal של הראסברי (הרצה זו תתבצע לאחר הרצת השרת).

```
python ras_final.py_
```

לאחר מכן קוד הראסברי ירוץ ואני נראה שהתקבלה כתובת IP במסך הראסברי וכתובת במסך המחשב (השרת).

ברגע שהכתובות התקבלו, אנחנו יודעים שהראסברי והשרת מחוברים אחד לשני ויכולים לשלוח הודעות ומידע אחד אל השני.

רפלקציה

בתחילת פרויקט זה, החלטתי שאני מעוניין לגעת בנושאים ובטכנולוגיות שלא נגעת בהם לפני, וכך בעזרת המדריך והמנטור בכיתה ללמוד אותם ולחזק אותם. אני מאמין שהצלחתי להתעסק בדברים שלא התעסקתי בהם לפני ברמה גבוה.

נורא נהינתי במהלך העבודה והרגשתי שאני לומד ממנה המון. הבנתי כיצד עובדים עם node ואיך מממשים שרת בשפת JS, גיליתי איך מתעסקים עם Raspberry Pi והבנתי כיצד ניתן לחבר למכשיר חיישנים ולתקשר איתם.

במהלך העבודה נתקלתי במספר קשיים שבעזרת שותפתי לפרויקט התמודדתי איתם בצורה טובה מאוד ומצאתי פתרונות יצירתיים וחכמים לתקלות רבות.

בעזרת הפרויקט, למדתי איך לעבוד בטכנולוגיית git בצורה רחבה והצלחנו לשמור על שקיפות מלאה בפרויקט ביני לבין שותפתי לפרויקט.

איתי, המנטור שלנו, נתן לנו ציוד לעבוד איתו ועזר לנו לכתוב את הקוד, מה שהקל עלינו בלמידה ובעבודה על הפרויקט ועזר לנו להתקדם וללמוד דברים חדשים. בנוסף לכך, היה נורא נחמד להכיר מישהו שעובד בתעשייה ומתעסק בטכנולוגיה שאיתה התעסקנו בפרויקט על בסיס יומיומי.

עם זאת, המדריך בכיתה העביר לנו במהלך השיעורים מצגות שעזרו לנו לנהל את הפרויקט ולסדר אותו בצורה טובה. למדנו את המשמעות החשוב שיש לUI ולUX וכיצד לממש אותן בפרויקט שלנו, ולמדנו כיצד לנהל ו"למכור" לאנשים מוצר.

לסיכום, נורא נהינתי מכל תקופתי בתכנית מגשימים, אני מרגיש שהגעתי למצב שבו אני יכול ללמוד כל דבר בצורה אוטוידקטית והגעתי לשלב שבו יש לי מספיק ידע לעבוד בחברות הייטק ולהשתלב בתעשייה הטכנולוגית. אני מרגיש שהתכנית העניקה לי ידע שלא הייתי מסוגל להשיג לבד בצורה מסודרת כמו שהשגתי אותם בתכנית.

ביבליוגרפיה

Google Cloud API'S

Google Cloud vision API

[/https://cloud.google.com/vision](https://cloud.google.com/vision)

Google Cloud Speech API

[/https://cloud.google.com/speech-to-text/docs](https://cloud.google.com/speech-to-text/docs)

Google Cloud Vision API

[/https://cloud.google.com/vision](https://cloud.google.com/vision)

Application

קבלת קולות אודיו מהשרת -

<https://stackoverflow.com/questions/1936828/how-get-sound-input-from-microphone-in-python-and-process-it-on-the-fly>

Raspberry Pi

Raspberry Pi Camera Robot

<https://www.youtube.com/watch?v=5eSSExIWkgY>

<https://www.youtube.com/watch?v=3BJFxn0AI>

setting wifi up via the command line

<https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md>

how to connect raspberry pi to wifi

<https://www.youtube.com/watch?v=lfHRLLRbErw>

<https://hackstore.co.il/books/raspberry-pi-for-beginners/#1> - ראסברי פאי

קישור לחיישנים:

חיישן מצלמה:

<https://www.or-e.co.il/raspberry-pi-camera-module-pi-noir.html>

- Raspberry Pi LESSON 28. Controlling a Servo with Raspberry Pi and Python: מנוע:

<https://www.youtube.com/watch?v=SGwhx1MYXUs&t=882s>

App.js

```
const express = require('express')
()const app = express
var bodyParser = require('body-parser')
;var fs = require('fs')
;var speech = require('./speech')
;var vision = require('./vision')
;var language = require('./language')
;var utils = require('./utils')
;const request = require('request')
"" = var partsOfText
;const fileUpload = require('express-fileupload')
"" = var ras_ip

;app.use(bodyParser.json({limit: '50mb'}))
;app.use(bodyParser.urlencoded({limit: '50mb', extended: true}))
;app.use(fileUpload())

app.get('/', (req, res) => res.send('Hello World!'))

search for ip address of the raspberry//
} (app.post('/ip', function (req, res
console.log(req.query.ip)
;(200)res.sendStatus
ras_ip = req.query.ip
({
```

```

gets the audio file from mobile //
}<= (app.post('/audio', (req, res
speech.recognize(Buffer.from(req.body.data, 'base64'))
){then(function(text.
;console.log(text)
;audioText = text
){language.analyze(text).then(function(parts
;res.send(parts) //
;let singlized = utils.singlize(parts.map(part=>part.content)) //
"" = partsOfText
partsOfText = parts
;console.log("singlized: " + singlized) //
;(200)res.sendStatus
({
((then(utils.detect(ras_ip , partsOfText.({
({

```

```

gets the labels from picture //
}<= (app.post('/detect', (req, res
if (!req.files)
;return res.status(400).send('No files were uploaded.')
The name of the input field (i.e. "sampleFile") is used to retrieve the uploaded file //
;let sampleFile = req.files.sampleFile
Use the mv() method to place the file somewhere on your server //
} (sampleFile.mv('./downloaded.jpg', function(err
if (err)

```

```

;return res.status(500).send(err)

call detection //

vision.detect('./downloaded.jpg')
}{then(function(labels.
;console.log("The lables from the picture:\n")
;console.log(labels)

)res.send(utils.compare(labels , utils.singlize
)partsOfText.reduce
}function(filtered_array,x)
}{if(x.tag=='NOUN')
filtered_array.push(x.content)
{
return filtered_array
;([[0]]([[] ],{
({
;({
({

app.listen(3000, () => console.log('Example app listening on port 3000!'))

```

speech.js

```

// Imports the Google Cloud client library
const speech = require('@google-cloud/speech');
const fs = require('fs');
const Promise = require('promise');

// Your Google Cloud Platform project ID
const projectId = 'mindful-primer-189620';

// Creates a client
const client = new speech.SpeechClient({

```

```

    projectId: projectId,
  });

let recognize = function(audioBytes){
  // The audio file's encoding, sample rate in hertz, and BCP-47 language code
  const audio = {
    content: audioBytes,
  };
  const config = {
    encoding: 'AMR_WB',
    sampleRateHertz: 16000,
    languageCode: 'en-US',
  };
  const request = {
    audio: audio,
    config: config,
  };

  return new Promise(function(resolve, reject){
    // Detects speech in the audio file
    client
      .recognize(request)
      .then(data => {
        const response = data[0];
        const transcription = response.results
          .map(result => result.alternatives[0].transcript)
          .join('\n')
        //console.log(`Transcription: ${transcription}`)
        resolve(transcription)
      })
      .catch(err => {

```

```

        if(err) reject(err);
        console.error('ERROR:', err);
    })
}
);
}

exports.recognize = recognize

                                vision.js

const Promise = require('promise');

// Imports the Google Cloud client library
const vision = require('@google-cloud/vision');

exports.detect = function(img_path){
    // Creates a client
    const client = new vision.ImageAnnotatorClient();

    return new Promise(function(resolve, reject){
        // Performs label detection on the image file
        client
            .labelDetection(img_path)
            .then(results => {
                const labels = results[0].labelAnnotations;
                let sendLabels = [];
                labels.forEach(label => sendLabels.push(label.description));
                resolve(sendLabels);
            })
            .catch(err => {

```



```

        console.error('ERROR:', err);
        reject(err);
    });
}
);
}

```

Language.js

```

const Promise = require('promise');

// Imports the Google Cloud client library
const language = require('@google-cloud/language');

// Instantiates a client
const client = new language.LanguageServiceClient();

//function exported that receives the text to be analyzed
//returns a promise with object of {}
let analyze = function(text){
    return new Promise(function(resolve, reject){
        const document = {
            content: text,
            type: 'PLAIN_TEXT',
        };

        // Detects the sentiment of the text
        client
            .analyzeSyntax({document: document})
            .then(results => {
                const syntax = results[0];
            });
    });
}

```

```

let analyzed = [];
syntax.tokens.forEach(part => {
  analyzed.push({
    tag: part.partOfSpeech.tag,
    content: part.text.content
  });
});

resolve(analyzed);
console.log(analyzed);
})
.catch(err => {
  console.error('ERROR:', err);
  reject(err);
});
})
}
exports.analyze = analyze;

```

utils.js

```

var pluralize = require('pluralize')
const request = require('request');
var vision = require('./vision');
var fs = require('fs');

function singlize(words){
  // let singlizedWords = words.map(word => pluralize.singular(word))
  return words.map(word => pluralize.singular(word));
}

function compare(lables, noun)

```

```
{
    console.log(noun);
    console.log(lables);
    return lables.indexOf(noun.toLowerCase()) >= 0;
}
```

//code added from notepad

```
function ask_for_photo(ras_ip){

    return new Promise((resolve, reject ) => {
        url='http://'+ras_ip+':4040/image'
        request.get(url, {encoding: 'binary'}, function(error, response, body) {
            if (error) {
                reject(error);
            } else {
                resolve("SUCCESS");
            }
        });
    })
}
```

```
exports.detect = function(ras_ip , partsOfText){
    if (ras_ip.length != 0)
    {
        ask_for_photo(ras_ip)
    }
}
```

Ras_final.py

```
from BaseHTTPServer import BaseHTTPRequestHandler,HTTPServer
import os
import requests
```

```

import socket
import netifaces as ni
import urlparse

PORT_NUMBER = 4040

data = ni.ifaddresses('wlan0')[ni.AF_INET][0]['addr']
def send_my_ip():
    PARAMS = {'ip':data}
    print data
    requests.request(method='post', url='http://192.168.43.39:3000/ip', data="find
my that", params=PARAMS)

class webServerHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        try:
            if self.path.endswith("/image"): # a sign to resolve the camera # #resolve
the camera
                print "entered to the rout"
                f = open('./example.jpg')
                size = os.stat('./example.jpg')
                self.send_response(200)
                self.send_header('Content-type' , 'image/jpg')
                self.send_header('Content-length' , size.st_size)
                self.end_headers()
                with open('./example.jpg', 'rb') as file:
                    self.wfile.write(file.read())
                print "sended"
                return
        except:
            self.send_error(404, "File was not found %s" %self.path)

```

```

def do_POST(self):
    print "entered to the post"
    try:
        if self.path.endswith("/answer"): # a sign to resolve the camera # #resolve
the camera
            ans = self.rfile.read(int(self.headers['Content-Leangth']))
            print ans
            self.send_response(200)
            print "sended posttttt"
            return
    except:
        self.send_error(404, "File was not found %s" %self.path)

if __name__ == '__main__':
    server = HTTPServer((str(data), PORT_NUMBER) , webServerHandler)
    print "web server running on port %s" % PORT_NUMBER
    send_my_ip()
    print "wow"
    try:
        server.serve_forever()
    except KeyboardInterrupt:
        print "^C entered, stopping web server..."
        server.socket.close()

```

engine.py

```

import os
import requests
import socket
import netifaces as ni
import urlparse

```

```
PORT_NUMBER = 4040
```

```
data = ni.ifaddresses('wlan0')[ni.AF_INET][0]['addr']  
def send_my_ip():  
    PARAMS = {'ip':data}  
    print data  
    requests.request(method='post', url='http://192.168.43.39:3000/ip', data="find  
my that", params=PARAMS)
```

```
class webServerHandler(BaseHTTPRequestHandler):  
    def do_GET(self):  
        try:  
            if self.path.endswith("/image"): # a sign to resolve the camera # #resolve  
the camera  
                print "entered to the rout"  
                f = open('./example.jpg')  
                size = os.stat('./example.jpg')  
                self.send_response(200)  
                self.send_header('Content-type' , 'image/jpg')  
                self.send_header('Content-length' , size.st_size)  
                self.end_headers()  
                with open('./example.jpg', 'rb') as file:  
                    self.wfile.write(file.read())  
                print "sended"  
                return  
        except:  
            self.send_error(404, "File was not found %s" %self.path)  
  
    def do_POST(self):  
        print "entered to the post"
```

```

        try:
            if self.path.endswith("/answer"): # a sign to resolve the camera # #resolve
the camera
                ans = self.rfile.read(int(self.headers['Content-Leangth']))
                print ans
                self.send_response(200)
                print "sended posttttt"
                return
        except:
            self.send_error(404, "File was not found %s" %self.path)

if __name__ == '__main__':
    server = HTTPServer((str(data), PORT_NUMBER) , webServerHandler)
    print "web server running on port %s" % PORT_NUMBER
    send_my_ip()
    print "wow"
    try:
        server.serve_forever()
    except KeyboardInterrupt:
        print "^C entered, stopping web server..."
        server.socket.close()

```

mainActivity.java

```

package com.example.user.searchorecorder;

import android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.os.AsyncTask;
import android.os.Environment;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Base64;

```

```

import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.HashMap;

public class MainActivity extends AppCompatActivity {

    private Button play, record, stop;
    private MediaRecorder myAudioRecorder;
    private String outputFile;

    final String url = "http://192.168.43.39:3000/audio";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        play = (Button) findViewById(R.id.play);
        stop = (Button) findViewById(R.id.stop);
        record = (Button) findViewById(R.id.record);
        stop.setEnabled(true);
        play.setEnabled(true);

        final RequestQueue queue = Volley.newRequestQueue(this);

        outputFile =
Environment.getExternalStorageDirectory().getAbsolutePath() + "/recording.3gp";

        record.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    myAudioRecorder = new MediaRecorder();

myAudioRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);

myAudioRecorder.setOutputFormat(MediaRecorder.OutputFormat.AMR_WB);

myAudioRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_WB);
                    myAudioRecorder.setOutputFile(outputFile);

                    myAudioRecorder.prepare();
                    myAudioRecorder.start();
                } catch (IOException ioe) {

```



```

        //make something
    }

    record.setEnabled(false);
    stop.setEnabled(true);

    Toast.makeText(getApplicationContext(), "Recording started",
    Toast.LENGTH_LONG).show();
    }
    });

    stop.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            myAudioRecorder.stop();
            myAudioRecorder.release();
            myAudioRecorder = null;
            record.setEnabled(true);
            stop.setEnabled(false);
            play.setEnabled(true);
            Toast.makeText(getApplicationContext(), "Audio Recorder
    Successfully", Toast.LENGTH_LONG).show();

            new Thread(new Runnable() {
                public void run() {
                    File file = new File(outputFile);
                    int size = (int) file.length();
                    byte[] bytes = new byte[size];
                    try {
                        queue.start();
                        //read the bytes from the file to the bytes object
                        BufferedInputStream buf = new
    BufferedInputStream(new FileInputStream(file));
                        buf.read(bytes, 0, bytes.length);
                        buf.close();

                        //decode the bytes
                        String base64 = Base64.encodeToString(bytes,
    Base64.DEFAULT);

                        System.out.println("Finish writing the file");

                        HashMap<String,String> params = new
    HashMap<String,String>();
                        params.put("data", base64);

                        System.out.println(base64);

                        System.out.println("Entered the file to params");

                        JSONObjectRequest jsObjRequest = new
                            JSONObjectRequest(Request.Method.POST,
                                url,
                                new JSONObject(params),
                                new Response.Listener<JSONObject>() {
                                    @Override
                                    public void onResponse(JSONObject
    response) {
                                        //System.out.println("worked
    yayy");
                                    }
                                }, new Response.ErrorListener() {
                                    @Override

```

```

        public void onErrorResponse(VolleyError error)
    {
        System.out.println("Error Getting 500");
    }
    });
    queue.add(jsObjRequest);

    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    }
    }).start();

    play.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v){
            MediaPlayer mediaPlayer = new MediaPlayer();

            try{
                mediaPlayer.setDataSource(outputFile);
                mediaPlayer.prepare();
                mediaPlayer.start();

                Toast.makeText(getApplicationContext(), "Playing Audio",
                Toast.LENGTH_LONG).show();
            } catch (Exception e){
                e.printStackTrace();
            }
        }
    });
}
}

```

activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.user.searchorecorder.MainActivity"
    android:background="@color/blue">

    <Button
        android:layout_width="150dp"
        android:layout_height="75dp"
        android:text="Record"
        android:textSize="30sp"
        android:id="@+id/record"
        android:background="@drawable/round_button"
        android:layout_marginTop="57dp"

```

```

        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

<Button
    android:layout_width="150dp"
    android:layout_height="75dp"
    android:text="Stop"
    android:textSize="30sp"
    android:id="@+id/stop"
    android:background="@drawable/round_button"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true" />

<Button
    android:layout_width="150dp"
    android:layout_height="75dp"
    android:text="Play"
    android:textSize="30sp"
    android:id="@+id/play"
    android:layout_alignParentBottom="true"
    android:background="@drawable/round_button"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="78dp" />

</RelativeLayout>

```

buttonShape.xml

```

<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >
    <corners
        android:radius="14dp" />
    <gradient
        android:angle="45"
        android:centerX="35%"
        android:centerColor="#47A891"
        android:startColor="#E8E8E8"
        android:endColor="#000000"
        android:type="linear"
    />
    <padding
        android:left="0dp"
        android:top="0dp"
        android:right="0dp"
        android:bottom="0dp"
    />
    <size
        android:width="270dp"
        android:height="60dp"
    />
    <stroke
        android:width="3dp"
        android:color="#878787"
    />
</shape>

```

Nice_Shape.xml

```

<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <corners android:radius="10dp"/> <!-- increasing the value, increases the

```

rounding. And as TTransmit said, to make it like a capsule make the radius half of your button height -->

```
<solid android:color="#37549e"/> <!-- the button color -->
</shape>
```

Android Manifests.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.user.searchorecorder">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```