**דוח תיעוד פרוייקט – נדב ברזני ודר טופולנסקי**

תחילה חשבנו על בחירת הנושא: רצינו לבחור משהוא שיתמקד בטכנולוגיות מתקדמות ורלוונטיות כיום. בחרנו לעשות פרויקט שישתמש בזיהוי פנים על מנת לזהה אנשים מתוך רשימה ויכניס את האנשים שברשימה, בהתאם נוציא הודעה קולית ברמקול.

שלבי ותהליך העבודה:

יצרנו שרטוט של הרכיבים של הפרויקט ודוח הגשת הצעת פרויקט.

חיברנו ראשית לכרטיס את הCOLOR LCD שהוא הלוח מגע למשתמש ונוכל להשתמש בו ולבדוק עוד רכיבים בעזרתו. לאחור מכן חיברנו את הRTC (real time clock) על מנת שנוכל להראות למשתמשים ולעבוד עם זמנים עדכניים.

בדקנו את שדתי הרכיבים הללו בעזרת קוד שכתבנו בארדואינו:







לאחר שבדקנו בעזרת הקטע קוד וראינו שאכן הרכיבים עובדים המשכנו להרכיב את הרכיב הבא ISD40004 שהוא מכיל את הרכיבים מיקרופון ורמקול (צריך לחבר לתוך חיבור הרמקול). חיברנו את הרכיב כמה פעמים בגלל שטעינו בחיבור ההדקים, לאחר מכן אחרי שתיקנו את החיבורים בדקנו את הרכיב, הרמקול והמיקרופון שהם אכן עובדים בעזרת קטע קוד שכאשר לוחצים על המסך מגע הוא מקליט בעזרת המיקרופון וכאשר עוזבים את המגע הוא מוציא את מה שהוקלט ברמקול.



כעת התחלנו לעבוד על הקוד שיזהה את הפנים למדנו על ספריות שיעזרו לנו. חיפשנו איך ניקרא מהמצלמת ווידאו ועשינו זאת בעזרת ספריה cv2 לאחר מכן העלינו תמונות שלנו והשתמשנו בספריה face\_recognition שתעזור לנו לזהות פנים. אנחנו נקרא כל frame שני ונקטין אותו ברבע כדי שתהליך הזיהוי יהיה יעיל יותר, ולאחר מכן ניתן את הפריים לספרייה שתבדוק את המיקום שלו והכל מול התמונה שנמצאת ב database.

import face\_recognition

import cv2

import numpy as np

# Get a reference to webcam #0 (the default one)

video\_capture = cv2.VideoCapture(0)

# Load a sample picture and learn how to recognize it.

nadav\_image = face\_recognition.load\_image\_file("dataset/Nadav.jpg")

nadav\_face\_encoding = face\_recognition.face\_encodings(nadav\_image)[0]

# Load a second sample picture and learn how to recognize it.

dur\_image = face\_recognition.load\_image\_file("dataset/Dur.jpg")

dur\_face\_encoding = face\_recognition.face\_encodings(dur\_image)[0]

# Create arrays of known face encodings and their names

known\_face\_encodings = [

    nadav\_face\_encoding,

    dur\_face\_encoding

]

known\_face\_names = [

    "Nadav",

    "Dur"

]

# Initialize some variables

face\_locations = []

face\_encodings = []

face\_names = []

process\_this\_frame = True

while True:

    # Grab a single frame of video

    ret, frame = video\_capture.read()

    # Resize frame of video to 1/4 size for faster face recognition processing

    small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face\_recognition uses)

    rgb\_small\_frame = small\_frame[:, :, ::-1]

    # Only process every other frame of video to save time

    if process\_this\_frame:

        # Find all the faces and face encodings in the current frame of video

        face\_locations = face\_recognition.face\_locations(rgb\_small\_frame)

        face\_encodings = face\_recognition.face\_encodings(rgb\_small\_frame, face\_locations)

        face\_names = []

        for face\_encoding in face\_encodings:

            # See if the face is a match for the known face(s)

            matches = face\_recognition.compare\_faces(known\_face\_encodings, face\_encoding)

            name = "Unknown"

            # Or instead, use the known face with the smallest distance to the new face

            face\_distances = face\_recognition.face\_distance(known\_face\_encodings, face\_encoding)

            best\_match\_index = np.argmin(face\_distances)

            if matches[best\_match\_index]:

                name = known\_face\_names[best\_match\_index]

            face\_names.append(name)

    process\_this\_frame = not process\_this\_frame

    # Display the results

    for (top, right, bottom, left), name in zip(face\_locations, face\_names):

        # Scale back up face locations since the frame we detected in was scaled to 1/4 size

        top \*= 4

        right \*= 4

        bottom \*= 4

        left \*= 4

        # Draw a box around the face

        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

        # Draw a label with a name below the face

        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)

        font = cv2.FONT\_HERSHEY\_DUPLEX

        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

    # Display the resulting image

    cv2.imshow('Video', frame)

    # Hit 'q' on the keyboard to quit!

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# Release handle to the webcam

video\_capture.release()

cv2.destroyAllWindows()