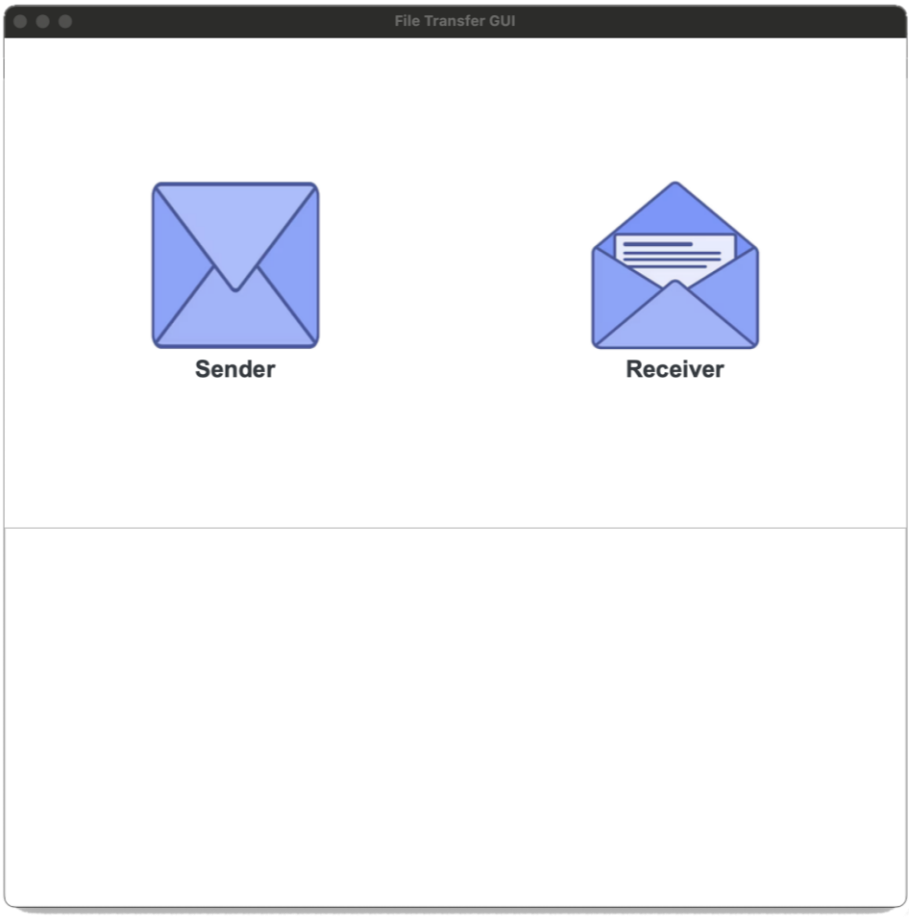
תיק פרויקט גמר

שידור מידע בין 2 מחשבים - רמקול ומיקרופון



זאק פורטנוי - 320497407

גיל עציוני - 318876737

תוכן עניינים

1. מבוא

תוכנת ה- FileOverSound הינה אפליקציה אשר מאפשרת לשדר מידע באמצעות מיקרופון ורמקול בין 2 מחשבים שונים.

לאחר הרצת התוכנית ואימות שאכן קיימים sender ו- receiver, ה- receiver יקבע היכן במחשב הוא רוצה שהמידע ישמר ולאחר מכן ה- sender יוכל להתחיל לשדר.

כאשר ה- sender יהיה מעוניין להפסיק את השידור הוא ילחץ על כפתור stop, וה- receiver יקבל הודעה על סיום שידור והמידע ישמר במקום המתאים.

בנוסף התוכנה תעביר ל- sender ול- receiver קובץ עם סטטיסטיקות על כמות הביטים והבתים ששדורו בהצלחה / נכשלו.

השינויים הנתמכים ע״י המערכת הם:

- יצירת קשר בין המחשב השולח למקבל.
- ה- sender יקבע מתי להתחיל ומתי לסיים את השידור.
- ה- receiver יקבע היכן הוא רוצה שהמידע ישמר במחשב שלו.

ממשק המשתמש הידידותי מאפשר שליחה וקבלה של מידע בין שני מחשבים שונים, כמו גם מידע סטטיסטי על איכות המידע שנשלח

2. דרישות קדם

המערכת מותאמת לעבודה בסביבת Windows, Linux ו- mac.

3. הוראות התקנה

בכדי להריץ את התוכנית נדרש לבצע את ההתקנות הבאות:

- נדרש לוודא שהמחשב מכיל גרסה עדכנית של שפת python - https://www.python.org/downloads/
לאחר ההתקנה מומלץ לוודא שהגרסה היא לפחות 3.13.0 (הסבר בהמשך הדף).

- הגרסה המקורית של פייתון מכילה את כל הסיפריות הסטנדרטיות הבאות:
  - threading
  - time
  - tkinter (מומלץ לוודא שאכן מותקן במחשב - הסבר בהמשך הדף).
  - שPIL
  - os
  - subprocess
  - logging
  - LogSetup
  - queue

- בנוסף, נדרש להתקין על המחשב את הסיפריות הבאות (הסבר בהמשך הדף):
  - Pillow
  - matplotlib
  - numpy
  - Sounddevice
  - ttkbootstrap
  - argparse
  - conf
  - sys

- כדי לבצע את כל ההתקנות, ולוודא שהמחשב מכיל את כל הסיפריות המעודכנות, נדרש לכתוב את הפקודות הבאות ל- terminal:

```
ensure you have python 3.13.0 installed  #  python --version
pip install Pillow matplotlib numpy sounddevice ttkbootstrap argparse conf sys # install the required libraries
ensure you have tkinter installed  # pip install tk
```

4. הוראות שימוש

4.1. הפעלת המערכת

- פתח את התוכנה בסביבת העבדודה (למשל .Visual Studio Code)
- הרץ את הקוד מה- terminal וחכה שמסך הבית של התוכנה יפתח (ראה 4.2).

4.2. מסך הבית

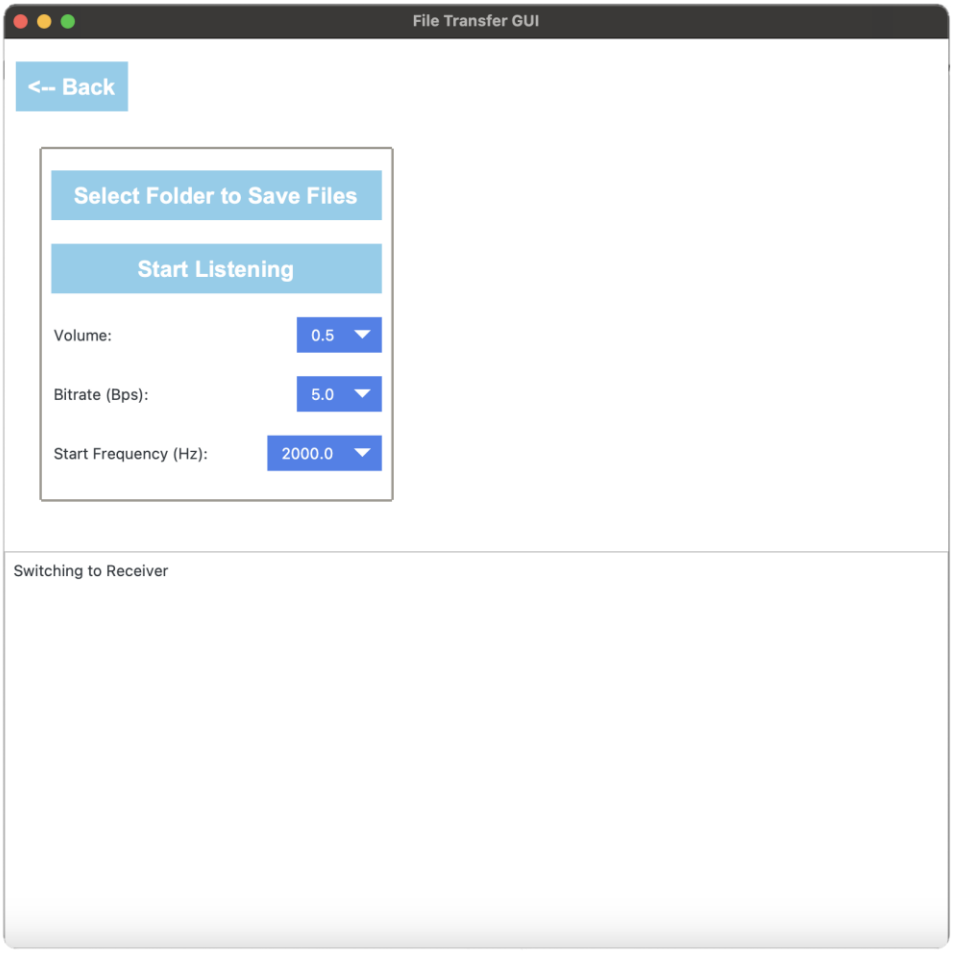
אחרי שמסך הבית ייפתח במחשב של המשתמש, עליו לבחור האם הינו השולח (sender) או המקבל (receiver), וללחוץ על התמונה המתאימה בהתאם.

- במידה ומהמשתמש ילחץ על התמונה של ה- receiver, הוא יועבר למסך של המקבל(ראה 4.3).

- במידה ומהמשתמש ילחץ על התמונה של ה- sender, הוא יועבר למסך של השולח (ראה 4.4).

4.3. receiver

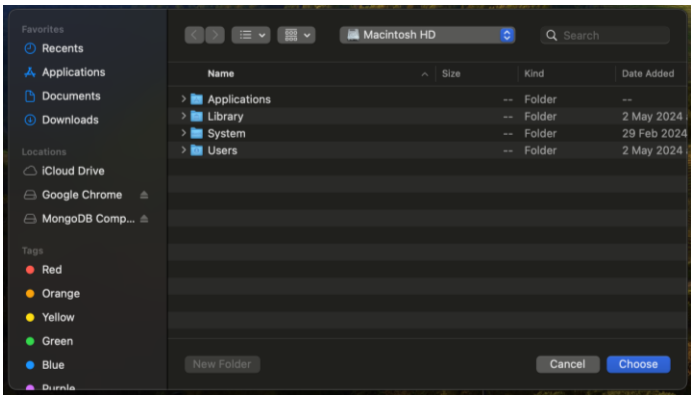
במידה והמשתמש לחץ על הציור של ה- receiver במסך הבית, נעבור לדף זה.



במסך זה ישנם 3 כפתורים, slider שמשנה את ההגדרות (configurations) ומסך אשר מציג את מצב ה- receiver אחרי כל פעולה שהמשתמש מבצע.

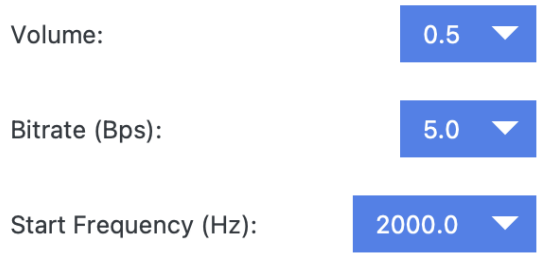
- הכפתור <-- Back מאפשר לחזור למסך הבית בכל רגע נתון.

- הכפתור Select Folder to Save Files מאפשר למשתמש לבחור את התיקייה הפיזית על המחשב המקומי בו ישמרו הקבצים שה- sender יעביר לו.
  כאשר המשתמש לוחץ על כפתור זה, נפתחת מערכת הקבצים של המחשב.
  על המשתמש לבחור את התיקייה המתאימה בה הוא מעוניין שהקבצים ישמרו.



- ישלח לו מידע sender, כלומר מחכה שה- senderהכפתור Start Listening מאזין ל- כדי שיוכל לקבל אותו.
  לפני שהתוכנה תתחיל להקשיב, על המשתמש לבחור את התיקייה בה הוא רוצה שהקבצים שימרו (בעזרת הכפתור " Start Listening.")

- המשתמש יכול לשנות את ההגדרות (configurations) בעזרת ה- sliders:
  - Volume - שולטת בעוצמת הרעש.
  - Bitrate (Bps) - כמות המידע שעוברת בשנייה.
  - Start Frequency (Hz) - שולט בתדר הנמוך ביותר.



- המסך התחתון מציג את מצב ה- receiver אחרי כל פעולה של המשתמש
  1. כאשר נעבור ל- receiver מהמסך הראשי נקבל את ההודעה - Switching to Receiver

  2. כאשר המשתמש יבחר את התיקייה בה הוא ירצה שהקובץ ישמר הוא יקבל את ההודעה - Folder selected: /Users/giletzioni/Desktop/FileOverSound-gil/src/SendAndSaveFiles/GetFiles
     במידה והמשתמש לא בחר תיקייה הוא יקבל את ההודעה - No folder selected.

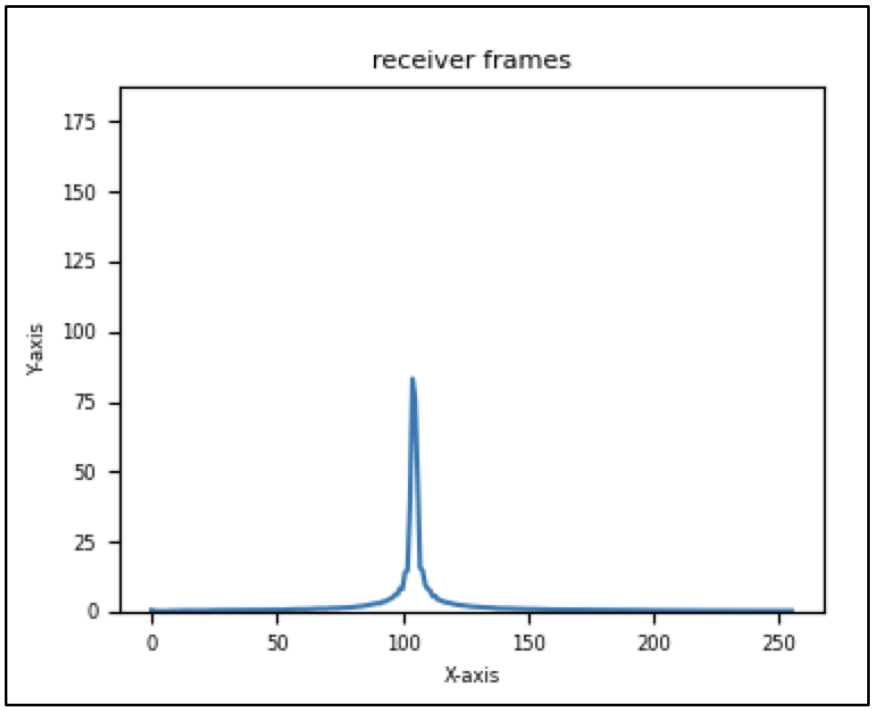
3. במידה והמתמש התחיל להאזין לפני שבחר קובץ, הוא יקבל הודעה על כך:
Please select a folder before receiving data.

4. אחרי כל פקאטה שהתקבלה בהצלחה,נקבל את ההודעה הבאה עם המספר המתאים - Receiving raw frame number 1

5. אחרי שכל ההודעה התקבלו בהצלחה, נקבל את ההודעה על כך שהקובץ הגיע ליעד, ואת ה- path שבו הרובץ נשמר.

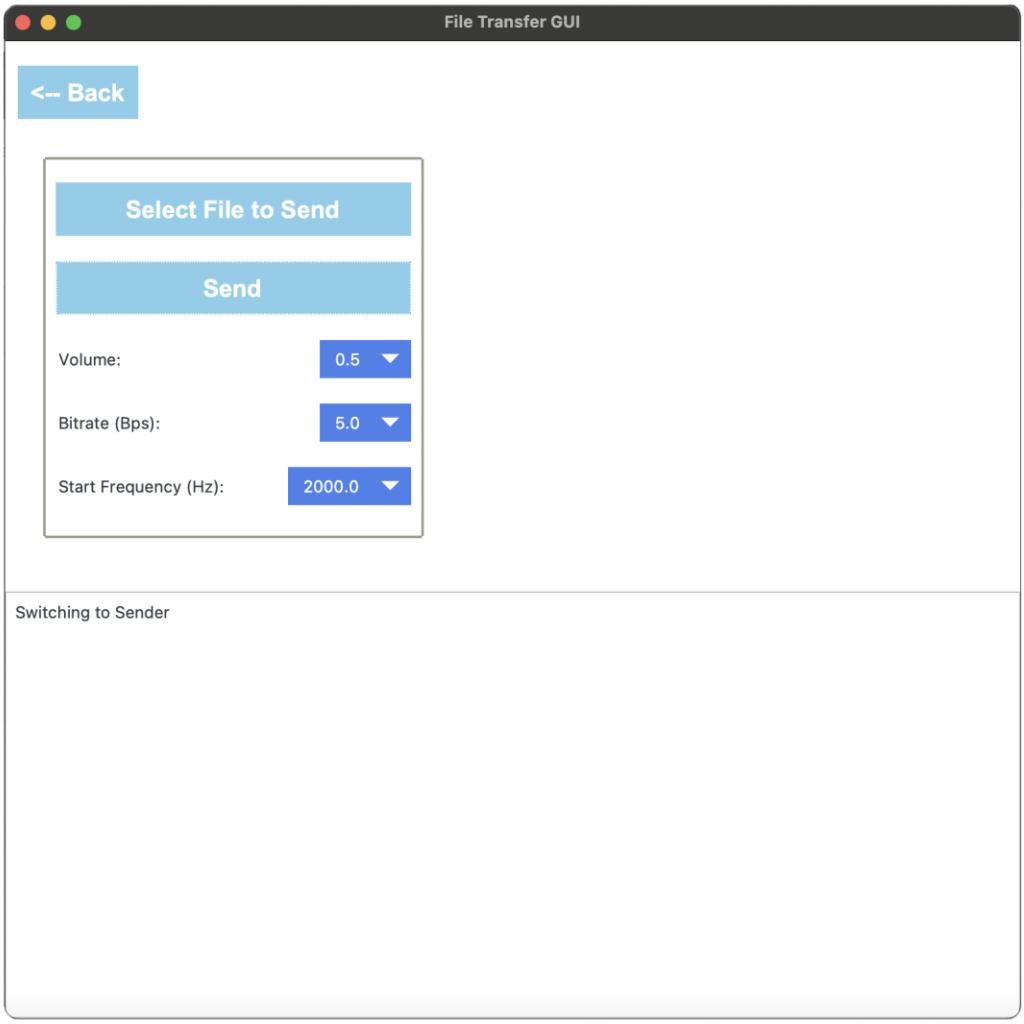
File received and saved as '/Users/giletzioni/Desktop/FileOverSound-gil/src/SendAndSaveFiles/getFiles/example.txt'.

- אחרי שה- receiver יתחיל להאזין, הוא יראה גרף שמציג את ה- frames שהוא מקבל:



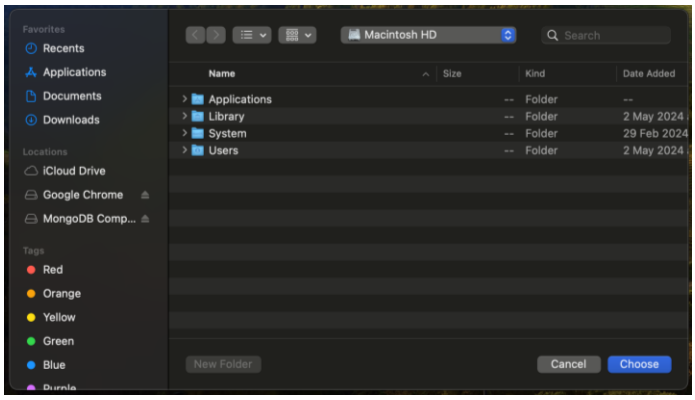
4.4. sender

במידה והמשתמש לחץ על הציור של ה- receiver במסך הבית, נעבור לדף זה.

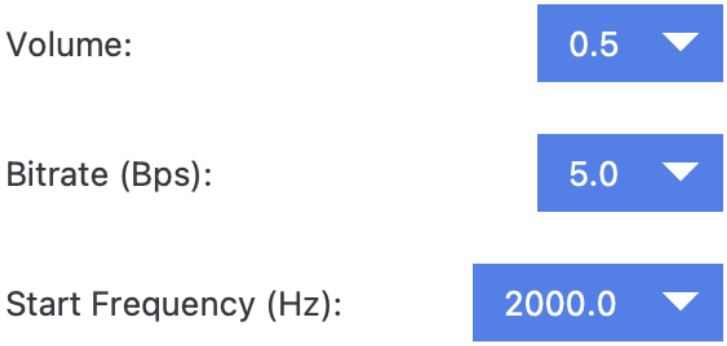


במסך זה ישנם 3 כפתורים, slider שמשנה את ההגדרות (configurations) ומסך אשר מציג את מצב ה- sender אחרי כל פעולה שהמשתמש מבצע.

- הכפתור <-- Back מאפשר לחזור למסך הבית בכל רגע נתון.
- הכפתור Select File to Send מאפשר למשתמש לבחור מתוך התיקייה הפיזית על המחשב המקומי את הקובץ אותו ה- sender ישלח.
  כאשר המשתמש לוחץ על הכפתור, נפתחת מערכת הקבצים של המחשב.



- הכפתור Send שולח את הקובץ שהמשתמש בחר אל ה- receiver.
- המשתמש יכול לשנות את ההגדרות (configurations) בעזרת ה- sliders:



- המסך התחתון מציג את מצב ה- sender אחרי כל פעולה של המשתמש.

1. כאשר נעבור ל- sender מהמסך הראשי נקבל את ההודעה -
   Switching to Sender

2. כאשר המשתמש יבחר קובץ, נראה את ה- path שלו על המסך -
   File selected: /Users/giletzioni/Desktop/FileOverSound-gil/src/SendAndSaveFiles/SendFiles/example.txt
   כאשר המשתמש יבחר קובץ שאינו קובץ txt או pdf, המשתמש יקבל הודעת שגיאה:
   Invalid File - Please select .txt or .pdf file.
3. כאשר המשתמש לוחץ על הכפתור של send, הוא יקבל הודעה שהקובץ נשלח -
   Starting SendFile from /Users/giletzioni/Desktop/FileOverSound-gil/src/SendAndSaveFiles/SendFiles/example.txt

4. אחרי כל פקאטה שנשלחת בהצלחה, המשתמש מקבל הודעה על כך שההודעה נשלחה בהצלחה, ובנוסף הוא יכול לראות את הפקאטה. לדוגמא -

```
Frame sent successfully b'hi!\ncasfd\n\nfdsfds\nfdsfdssdfgse\ngse\ngsegw'
```

אחרי כל פקאטה שלא מגיעה בהצלחה, המשתמש מקבל את ההודעה הבאה -

```
Timeout receiving ACK from receiver
```

5. אחרי שכל ההודעה הגיעה ליעדה בהצלחה, ה- sender יקבל הודעת הצלחה, וגם את הקבוץ וה- path. לדוגמא -

```
File '/Users/giletzioni/Desktop/FileOverSound-gil/src/SendAndSaveFiles/SendFiles/example.txt' sent successfully.
```

במידה ושלוש פקאטות לא הגיעו ברצף, השליחה תיפסק, וה- sender יקבל את ההודעה הבאה -

```
Error while sending file parts: Frame sending failed after 3 retries
Sending stopped because Frame sending failed after 3 retries
```

4.5. הפעלה מהירה דרך ה- terminal

אפשר גם להתחיל את התוכנה אם אפשרויות נתונות בשביל לחסוך לחיצות על כפתורים help לדוגמא:

src\main.py --sender --path pytest.ini"\. " python.exe

אפשר לראות את האפשרויות בעזרת --help

```
PS C:\Users\ZakPortnoy\Documents\GitHub\FileOverSound> python.exe .\src\main.py --help

usage: main.py [-h] [--sender] [--receiver] [--path PATH]

File Transfer GUI

options:

  -h, --help   show this help message and exit

  --sender     Run as sender

  --receiver   Run as receiver

               --path PATH  Path to the file to send or folder to save received files
```

5. סיפריות חיצוניות

5.1. הסיפרייה tkinter

הסיפרייה מאפשרת יצירת GUI - graphical user interface.
בחרנו להשתמש בסיפרייה זו כי היא פשוטה לשימוש ומכילה מספר גדול של פונקציות..

5.2. הסיפרייה ttkbootstrap

הרחבה לסיפרייה tkinter, אשר מאפשרת שימוש בסיפרייה Bootstrap על גבי tkinter. סיפרייה זו הקלה עלינו את יצירת ה- GUI, ועזרה לנו לעצב אותו בצורה נוחה יותר לשימוש.

5.3. הסיפרייה PIL

הסיפרייה עוזרת לעבד ולהציג תמונות ב- tkinter. tkinter מסוגלת לעבד קבצי gif ו- pgm בלבד.
השתמשנו בספרייה כדי להציג את התמונות של ה- receiver וה- sender.

5.4. הסיפרייה argparse

הסיפרייה מאפשרת ליצור תוכנות עם ממשק cli בצורה פשוטה ובטוחה יותר.
בתוכנית שלנו, הסיפרייה עזרה לנו לוודא שאין משתמשים סותרים על אותו GUI.
בנוסף, הסיפרייה איפשרה לנו לשלוט באיזו תצוגה תתחיל ב- GUI (ממסך בית / מסך שולח / מסך מקבל)..

5.4. הסיפרייה os

הסיפרייה מאפשרת לתקשר עם מערכת ההפעלה.

השתמשנו בספרייה זו כי היא מאפשרות לבדוק אם קובץ קיים (os.path.exists), לקבל את שם הקובץ בלבד מתוך נתיב (os.path.basename), וליצור נתיבים (paths) בין מחשבים שונים (os.path.join). כך ניתן להבטיח שהקוד יעבוד בצורה תקינה על מערכות הפעלה שונות, ואל מחשבים שונים.

5.4. הסיפרייה subprocess

הסיפרייה מאפשרת לנו להריץ שני קבצים במקביל.

5.7. הסיפריות logging ו- LogSetup

הסיפריות עוזרות ליצור ולייצר הודעות. השתמשנו בסיפרייה תוך כדי כתיבת הפרויקט כדי לבצע debug בקלות. לאחר סיום הפרויקט, הסיפריות סייעו לנו להבדיל בין סוגי ההדעות, בעזרתה קבענו מה יוצג על המסך

5.8. הסיפרייה matplotlib

הסיפרייה עוזרת להציג נתונים בצורה ויזואלית, ובפרט גרפים.
השתמשנו בסיפרייה כדי להציג סטטיסטיקות ונתונים אודות אחוזי ההצלחה של הפקטות.

5.9. הסיפרייה sounddevice

הסיפרייה עוזרת לשלוח מידע שמע (sd.play) ולקבל אותו (sd.InputStream) בקלות.

5.9. הסיפרייה numpy

הספרייה תומכת בעבודה עם מערכים ומטריצות גדולות ורב-ממדיות,וגם מספקת פונקציות מתמטיות.
פונקציות אלה מאפשרות לטפל ביעילות בנתוני שמע

<u>סיכום הפונקציות של הסיפריה numpy בהן השתמשנו:</u>

שם הפונקציה + מה היא מקבלת	מה הפונקציה עושה	מה הפונקציה מחזירה
(num_of_items)np.zeros	יוצרת מערך בגדול של num_of_items שמלא ב-אפסים	מערך מלא באפסים
(values)np.abs	מחשבת את הערכים המוחלטים של כל האיברים במערך	מערך עם עכים מולחטים
(signal)np.fft	מבצעת המרה של מערך signal לתדרים ((FFT	מערך
(original_array)np.copy	יוצרת עותק של המערך original_array	עותק של המערך
shift_steps)np.roll ,(array	מזיז את האיברים במערך ב-shift_steps צעדים	מערך חדש שבו האיברים הוזזו
value_to_find)np.searchsorted ,(sorted_array	מחפש איפה אפשר להכניס את הערך value_to_find במערך מבלי לפגוע בסדר	אינדקס שבו ניתן להכניס את value_to_find
(array_to_check)np.max	מחפש את הערך המקסימלי במערך	הערך המקסימלי במערך
shift_steps)np.roll ,(array	מזיז את האיברים במערך ב-shift_steps צעדים	מערך חדש שבו האיברים הוזזו בסיבוב
np.argmax(frequency_amplitudes)	מחפשת את האינדקס של הערך המקסימלי במערך	האינדקס של הערך המקסימלי במערך
np.sum(frequency_amplitudes)	מחשבת את הסכום הכולל של כל הערכים במערך	סכום הערכים במערך

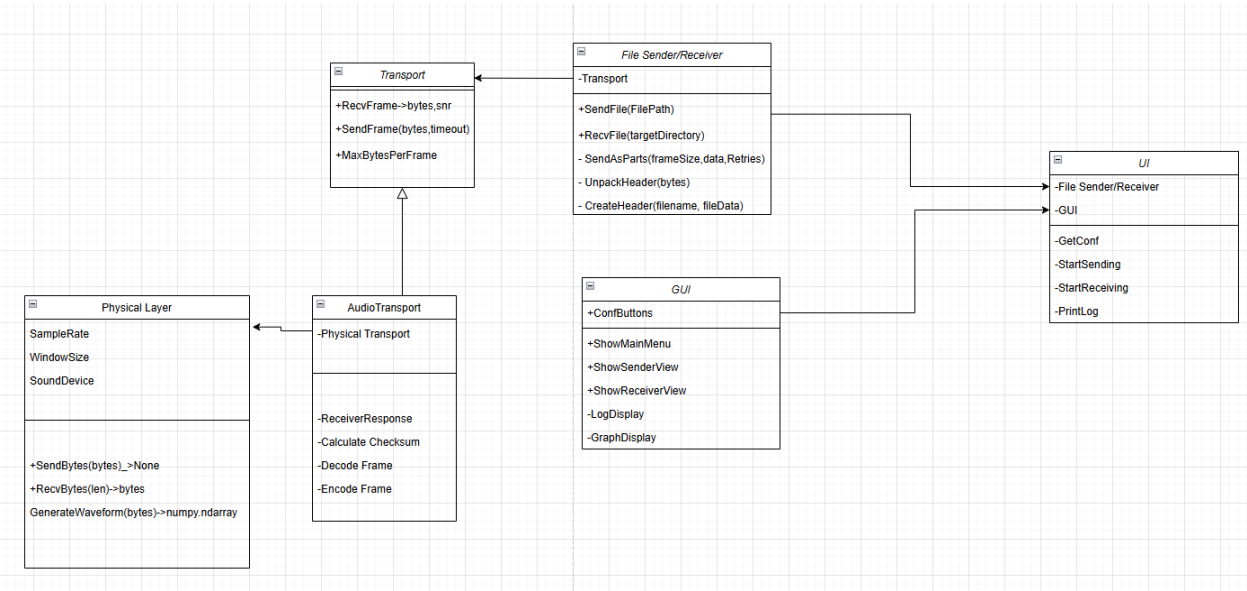
6. ארכיטקטורת תוכנה

6.1. כללי

המערכת מבוססת על קובץ הרצה בודד.
קובץ זה מכיל הן את הלוגיקה העסקית של המערכת (back-end) והן את שכבת התצוגה שלה (GUI).

6.2. מבנה המחלקות

המערכת מורכבת ממספר מחלקות. להלן תרשים המציג את מחלקות המערכת השונות והקשר ביניהן:



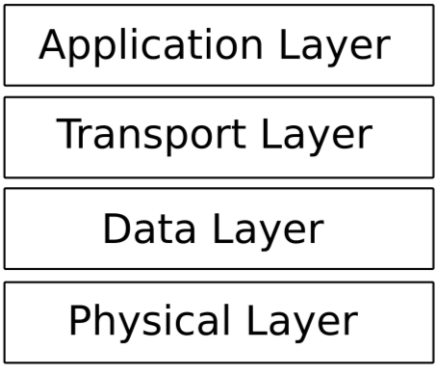
6.3. תיאור המחלקות

שם המחלקה	תפקיד
UI	מטפל בשינויים שמוצגים על ה- GUI, והעברת השינויים שהמשתמש עושה ב- front-end אל ה- back-end.
Tkinter	מכיל את ה- GUI ואת כל ה- components שלו.
File Sender/Receiver	מקבל מה- sender את הקובץ שהוא רוצה לשלוח, ומקבל מה- receiver את התיקייה שהוא רוצה לשמור. מעביר ל- Transpot את הקובץ והתיקייה, ומטפל בהודעות של תחילת וסוף שידור.
Transport	מנהלת העברת קבצים אמינה על ידי חלוקת הקובץ לפאקטות של 40 בתים, הוספת כותרות, והבטחת מסירה רצופה באמצעות מספרי מסגרות.
Audio Transport	מנהלת את מספרי הפריימים את את ה stop and wait ואת מנגנון שליחה מחדש של פקטות שלא התקבל עליהם תשובה
Physical layer	מנהלת את הצורה של הפקטה כולל אורך ובדיקת צקסם מקודד\מפענח מאות של קול לבתים מקנפג ודואג לעבודה עם החומרה

7 Protocols

7.1. Introduction:

Our project is structured into four distinct layers. Each layer is responsible for its specific functionality, and interacts with the next layer.



7.2. Application layer:

The application layer handles file transmission and reception:

- It receives the file path where the user wants to save the file from the receiver.
- It receives the file path and file name from the sender.

Additionally, the application layer print to the GUI:

- To the sender and the receiver when the file transfer begins.
- Confirming once all packets have been successfully received or sent.

7.3. Transport layer:

7.3.1. Overview:

The transport layer manages reliable file transfer by segmenting files into chunks of 40 bytes, adding headers for metadata, and ensuring sequential and complete delivery using frame numbers.

The header contain:

- **File Name length (int) -** represents the length of the file's name, encoded as a 4-byte integer in big-endian format. For example, example.txt has 11 characters (0x0b in hexadecimal), so it will be represented as \x00\x00\x00\x0b.
- **File Name (string) -** The file name (e.g., example.txt), padded to 32 bytes. If the name is shorter than 32 bytes, it is padded with zeros (\x00).
- **File Length (int) -** Indicates the file size in bytes using 4 bytes.

Example for

FileNamelen 4 Bytes \x00\x00\x00\x0b	FileName 32 Bytes example.txt\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00	FileLen 4 Bytes \x00\x00\x00x

7.3.2. sender:

The sender reads the file, splits it into chunks of 40 bytes, and sends them sequentially. It first creates and sends a header containing the file's metadata (name, name length, and size). After a short delay (that ensure reliability), it reads the file in fixed-size chunks (40 bytes) and sends each chunk with an increasing frame number.

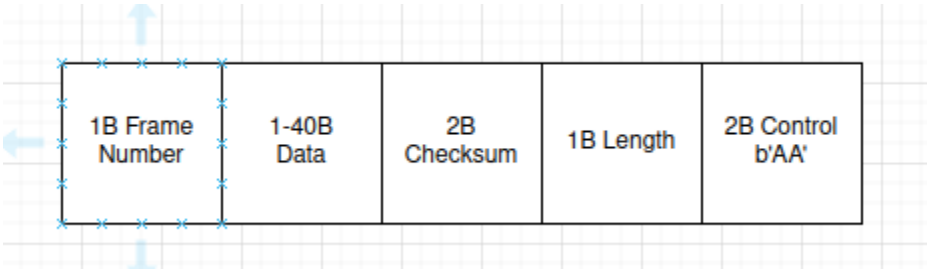
7.3.3. receiver:

The receiver retrieves the header to extract metadata (file name and size) and uses this information to determine how many chunks to expect. It then reads each incoming chunk in sequence, writes it to a file, and tracks progress to ensure all data is received. If the file size doesn't match the received data, it raises exceptions.

7.4. Data link layer:

Data is transmitted in frames of up to 40 bytes.
After the transmission of each frame the sender will wait for a response frame from the receiver, if it does not receive this within a certain time frame it will retransmit the frame up to a total of 3 times.

Each frame consists of



A frame number which is used on the receiver side to identify if a packet was retransmitted in the case of a lost response packet this is increased each packet and returns to 0 after 256 packets are sent.

The checksum length and control blocks are used to enable the receiver to identify valid packets, details provided in the receiver section

7.5. Physical layer:

7.5.1 Overview:

The physical layer has configuration options that allow tuning different frequencies, transmission speeds and volumes.

These need to be set the same way for the transmitter and receiver for it to work correctly.

256 frequencies are chosen, each representing a different byte.

The low frequency can be configured directly, the space between each frequency is defined by the number of samples in each fft window at the receivers side. This allows to match the frequency domain sending and receiving resolution.

Some considerations that currently limit the transmission speed.

- The microphones tested do not pick up a clean signal some frequencies appear to be received for longer than they are sent for and overlap other frequencies.
- It takes some time between when the signal is first identified and when it stabilizes. This seems to be affected both by the frequency received and the preceding signal.
- Not all signals are received at the same strength while higher frequency signals (above 1800hz) can be picked up they will be much lower magnitude than lower frequency ones even when played at the same power. This can be seen in the response curve of many microphones.

This limits the speed at which signals can be sent reliably. And we found that 5 –10 bps is what seems to work.

Some options that might be worth exploring if we would like to improve the transfer speed.

- We can use more than one frequency at a time to enable us to represent more than one byte per encoded signal. This will reduce the maximum amplitude per frequency but considering that stabilization time is our problem this should still give us a good signal. Using the same 256 steps and encoding 2 frequencies at once should double the transmission speed. 4 at once should quadruple it and so on.
- When processing the audio, we can look at not only the loudest frequency and try to identify the top few
- We can look into different hardware or if it is possible to disable the processing that is causing the issues.
- We can increase the bandwidth used or the sampling speed at the receivers end to allow for each signal to represent more than 8 bits. For example, using 1024 different frequencies can allow us to send 12 bits per signal increasing our speed by 50%.
- We can encode the data to allow for error recovery at the receiver

7.5.2 Encoding and Transmision

The frame is then encoded as a unit with each frequency being played for a duration based on the configured bytes per second option.

This is done according to a chosen sample rate using the numpy library.
Care is taken to keep track of the phase between different bytes to as not to have a jump when transitioning which can add noise to the signal.

The encoded frame is then played using the soundDevice library.

For example frame number 1 which contains 1 data byte “1” (or int 49) would be encoded as:

Frame Number= 1

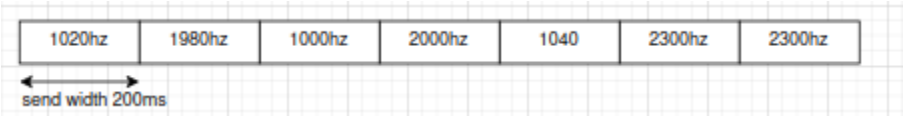
Data = 49

Checksum = 0,50

Len = 2

Control = 65,65

And with a configuration of 5bps and an fft window of 50ms it would be translated to frequencies like so:



7.5.3 Receiving and Decoding

The receiver exposes a blocking function call that will return when a frame is received or after a timeout has passed.

The receiver has a configurable sample rate which it reads the audio signal from the microphone and a configurable decoding rate and FFT window size, the FFTdata array stores audio samples in a buffer of the size of the FFT window and each time new audio samples are received the FFTdata array is updated by removing the old samples and adding in the new ones at the end.

Each sample is then decoded and stored as the maximum frequency for that window the magnitude of that frequency and the total energy in that signal (for evaluating signal to noise)

The sample array stores all samples that can still be relevant to a frame based on transmission speed and maximum bytes per frame. After the maximum frame time has passed for a given sample, it will be deleted to free up memory.

Each time a new sample is added the receiver will use the samples stored to try to identify if a frame is present. It chooses samples spaced according to the sending rate and check the following

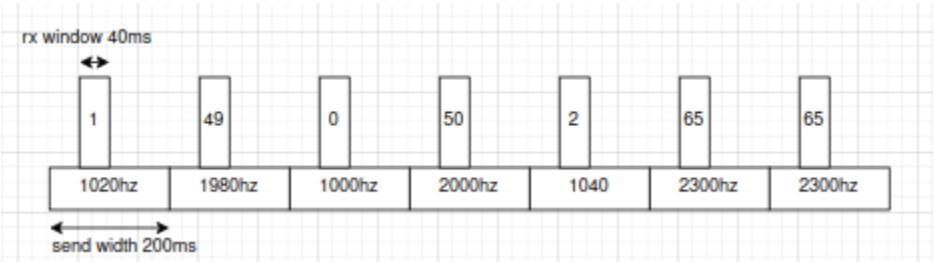
- The last two bytes are the control signal
- The length byte is within the required size
- The checksum is equal to the sum of the length number of bytes

If all these are true a frame will be marked as found.

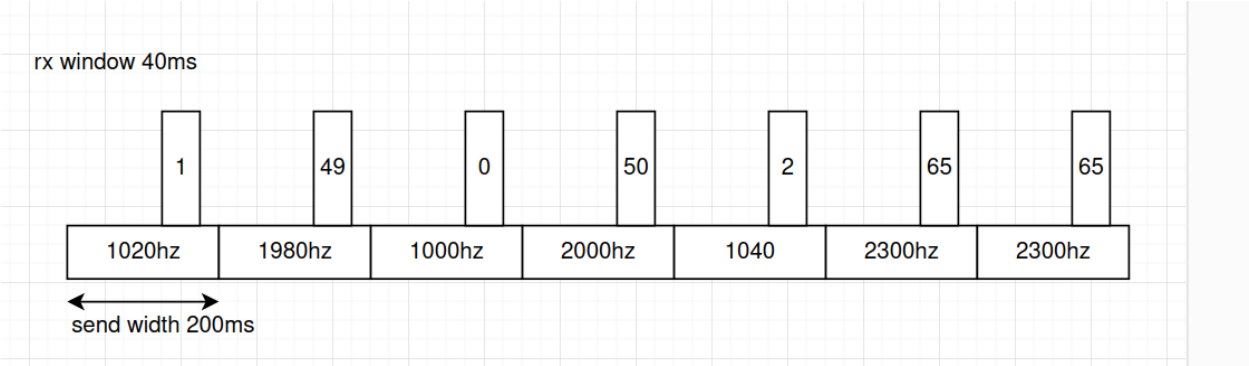
In order to read the best signal to noise ratio the receiver will continue listening for the duration of one byte and will record the maximum seen.

The function will then return the frame received.

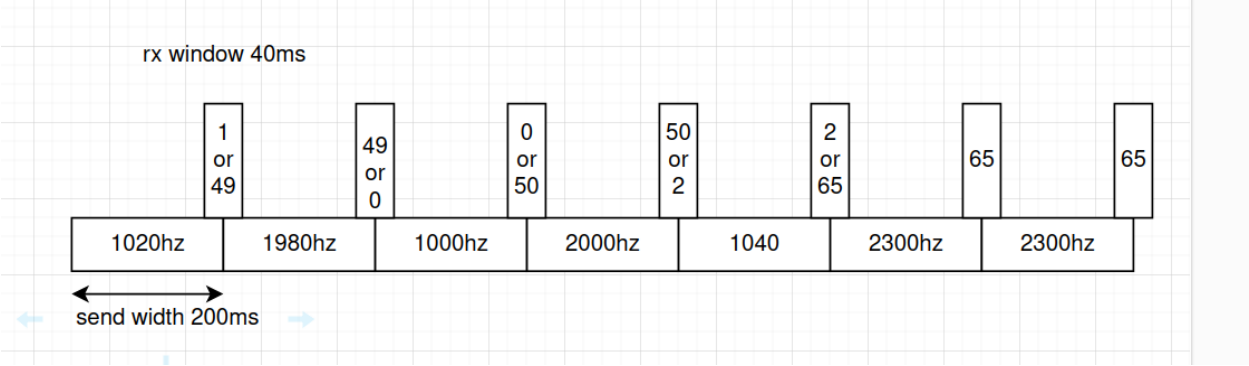
This is an example of the receiver examining the frame. This will repeat every calculation interval.



And can be read at multiple points in the sending blocks



Samples where the fft window is partially in two sent bytes will probably get bad data and not be identified as a frame, the fft window needs to be smaller than the sending size.



8. מקרי בדיקה

המערכת עברה סידרה של בדיקות לפני הגשתה.
הבדיקות בוצעו בעזרת הסיפרייה unittest שמסייעת לבצע בדיקות ב- python.

מכיוון שרוב הפונקציות בפרויקט קוראות לפונקציות אחרות, השתמשנו ב- patch
וב- MagicMock אשר סייעו לנו "לחקות" (mock) פונקציות ואובייקטים.
שיטה זו אפשרה לנו לבדוק כל פונקצייה בנפרד, ללא תלות בהצלחתה של הפונקציות האחרות.
נדגיש כי לא ביצענו בדיקה ייחודית עבור הקבצים שמטפלים ב- GUI. את הקבצים הללו כתבנו בנפרד מהפרויקט, ולאחר סיום הפרויקט עצמו חיברנו בין שני החלקים.
לא ביצענו בדיקה עבור הקבצים הללו מפני שה- GUI רק מפעיל את הפונקציות הקיימות ומציג אותן על המסך.

לצורך בדיקת טיב התקשורת השתמשנו בפקטות רנדומליות ומדד של signal to noise שאפשר לנו לראות איך שינויים בקונפיגורציות השפיע.

בסוף בשביל לבדוק את התוכנה כולה הרצנו בדיקות ידניות בשביל להעביר קבצים שונים בין שני מחשבים בתדרים ובמרחקים שונים. ראינו שהפרוטוקול מצליח לעבוד גם אם יש רעש בחדר ואפילו להצליח אם פקטות שנפלו. לקח זמן אבל הצלחנו להעביר קובץ של אלפי בתים.

10. דוח בדיקות

Testing setup

Delay between frames is the time it takes to send 5 bytes to recreate the timing of waiting for ack.

SNR = the minumum of all the bytes received in the array. Calculated by taking the magnitude of the relevant frequency and dividing it by the sum of the others.

Test report:

Each test sends a randomly generated frame 20 times and calculates results

We will use the following default settings and vary them on at a time and see how they affect transmission.

Default settings

- Audio sampling rate 41200
- Sending speed 10 BPS
- Receiver fft window 25 ms
- Frame size random bytes
- Volume 0.5
- Frequency range 5000-15200
- Distance between transmitter and receiver 1 meter

Test:

Modifying frequency range

Receiver fft window	Average snr	Percent of frames received	Frequency range Hz
25	0.11	35	8000-18200
50	0.24	95	8000-13100

25	0.35	95	5000-15200
25	0.25	95	2000-12200
25	0.29	95	1000-11200
25	0.27	85	400-10600
75	0.13	75	413-3813

Result:

Based on these results we will continue with the settings of 5000-15200 Hz as they have the best reception rate and snr.

Test:

Modify the transmission speed.

notes	Average snr	Percent of frames received	Bps
	0.2	65%	15
	0.2	50	12
Close to previous result	0.35	90	10
	0.43	95	8
	0.32	100	5
	0.6	100	2

Result:

We can see that at 10 bps where the settings are the same as before the results came out close to what we had before.
We see that the slower the transmission speed the better snr and reception rate is.

Test:

Modify volumes

notes	Average snr	Percent of frames received	volume
	0.50	95	1.0
	0.42	90	0.80
	0.35	90	0.6
	0.35	90	0.4
	0.23	95	0.2
	0.14	95%	0.1
	0.05	100	0.05
Could barely hear this	0.05	95	0.03
	0.03	75	0.02
	0.02	45	0.01

	N/A	0	0.005

Result:

The volume didn’t make much of a difference in frame detection as most of the lost frames are a result of the transmission interfering with itself

Test:

Add ambient noise and see how it affects the results
We will have a song playing in the background and loud volume during transmission:
https://www.youtube.com/watch?v=tbNlMtqrYS0

Average snr	Percent of frames received	volume
0.08	100	0.8
0.08	95	0.6
0.02	100	0.4
0.03	45	0.2
0.07	35	0.1

Result:

Here we do see higher volumes do better than lower volumes

Test:

Varying frame lengths and speeds

Effective bps is calculated as adding 15 bytes overhead for stop and wait and headers and multiplying by percent of frames received

Notes	Effective bps	Average snr	Percent of frames received	Frame length-bps
	4.36	0.18	60	40-10
	3.63	0.33	100	40-5
	4.6	0.19	70	30-10
	5.4	0.26	95	20-10
Default transmissoin	3.8	0.32	95	10-10
	3.56	0.31	95	5-15
	3.75	0.23	75	5-20

Result:

We can see that under ideal conditions a frame length of 20 with a bps of 10 does best with and effective bps of 5.4.

Test:

Distance transmission.

We recorded the transmission and played it from a cell phone at varying distances

Average snr	Percent of frames received	Distance (meters)
1.5	100	0.5
0.35	65	1
0.27	80	2
0.24	75	3
0.2	60	4
0.12	70	3 and in a drawer

Result:

Similar to varying volume and we can see that it can transmit across a room