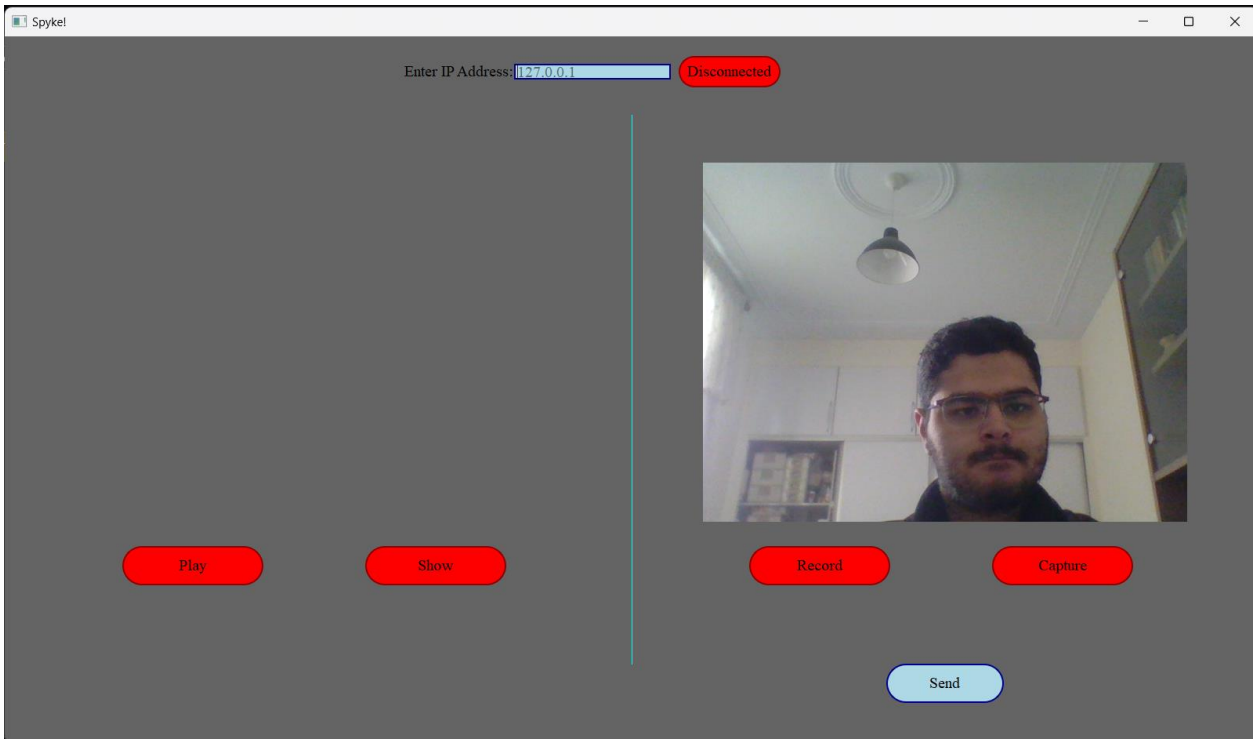
به نام خدا

پروژه اول درس مالتی مدیا

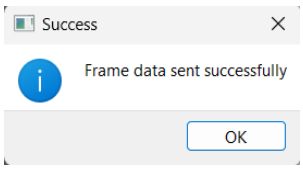
سید علیرضا تهامی، امیرمحمد رستم آبادی، مجتبی دهقانی فیروزآبادی

نحوه کار با برنامه

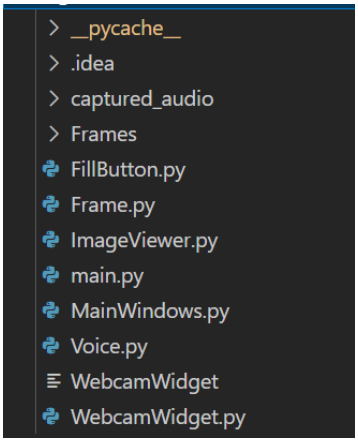
1. بعد از بالا آمدن برنامه ابتدا آی پی، به برنامه داده می شود تا در صورت نیاز به ارتباط برنامه به آی پی مد نظر متصل شود.
2. با استفاده از دکمه capture امکان گرفتن عکس فراهم می شود.
3. با استفاده از دکمه Record امکان ضبط صدا در مدت 30 ثانیه فراهم می شود. (توجه شود که صدای ضبط شده دارای یک کانال است)
4. با استفاده از دکمه send ویس و عکس ارسالی به سمت آی پی هدف ارسال می شود. در صورت عدم برقراری ارتباط خطای متناسب به کاربر داده می شود و در صورت ارسال نیز، پیام موفقیت آمیز بودن ارسال به کاربر داده می شود.



پیغام ارسال موفقیت آمیز عکس:



پس از ارسال داده ها امکان استفاده از دکمه های play,show فراهم می شود.



توضیح کد برنامه :

کد برنامه شامل سه بخش اصلیش:

- طراحی گرافیکی
- کد مربوط به مدیریت فریم در فایل frame
- کد مربوط به مدیریت ویس در فایل voice

پنجره گرافیکی در فایل main window ساخته شده است، که در آن دکمه ها و background صفحه تعریف شده است و همچنین هر دکمه به تابع خود connect شده است.

فریم ارسالی و دریافتی در پوشه frame و صوت ارسالی و دریافتی در پوشه captured_audio ضبط و ذخیره می شود.

همچنین حداکثر نگه داشتن چند فریم یا صوت ضبطی به صورت همزمان وجود ندارد و با ضبط یک فایل، فایل ضبطی قبلی از بین می رود و همچنین با دریافت یک فایل، فایل دریافتی قبلی از بین می رود.

برای اینکه سرور در لحظه به صورت بر خط به سوکت گوش بدهد و همزمان برنامه هم در حال اجرا باشد، نیاز به برنامه نویسی چند هسته ای وجود دارد که این امکان از طریق کتابخانه سیستم پایتون انجام شده هر چند از طریق کتابخانه pyqt نیز امکان پذیر بود. استفاده از چندین هسته همچنین باعث کم شدن ریسک بسته شدن برنامه می شود.

ابتدا یک تابع به صورت واسط بالای فایل های main, main window تعریف شده، سپس در زمان تعریف پراسس فراخوانی شده.

```python
# Server function that simulates a server running
def voice_server_function():
    voice.voice_server()
def frame_server_function():
    frame.server()
```

```python
# Create a separate process for the server function
voice_server_process = Process(target=voice_server_function)
voice_server_process.start()

frame_server_process = Process(target=frame_server_function)
frame_server_process.start()
```

با توجه به جدا بودن سرور های فریم و ویس، هر کدام به صورت یک پراسس جدا تعریف می شوند.

در زمان ارسال اطلاعات، از تکنیک های مدیریت خطا برای رفع برخی از fetal error ها استفاده شده. در صورت عدم استفاده از این بخش، با بسته شدن پروسه کلاینت، برنامه به پایان می رساند.

```python
def send_button_clicked(self):
    self.send_button.setText("Sent")
    # Create a separate process for the server function
    try:
        server_process = Process(target=frame_client_function, args=(self.ip_value,))
        server_process.start()
        server_process.join()  # Wait for the process to finish
        QMessageBox.information(None, "Success", "Frame data sent successfully")
    except ConnectionRefusedError as e:
        QMessageBox.critical(self, "Error", "Can not find the server!! Check your IP again")
    except Exception as e:
        QMessageBox.critical(self, "Error", f"An error occurred: {str(e)}")
    try:
        server_process = Process(target=voice_client_function, args=(self.ip_value,))
        server_process.start()
        server_process.join()  # Wait for the process to finish
        QMessageBox.information(None, "Success", "Voice data sent successfully")
    except ConnectionRefusedError as e:
        QMessageBox.critical(self, "Error", "Can not find the server!! Check your IP again")
    except Exception as e:
        QMessageBox.critical(self, "Error", f"An error occurred: {str(e)}")
```

توضیح سرور و کلاینت، فریم و ویس به صورت کامنت گذاری در هر بخش آورده شده. فریم با استفاده از کتابخانه opencv ذخیره شده و به صورت یک فایل باینری در کلاینت فراخوانی و در سرور هم به صورت فایل ذخیره می شود.

کتابخانه های sounddevice, scipy برای ضبط و پخش و فراخوانی صدا در برنامه استفاده می شود. در زمان ارسال صدا، برای بازسازی صدا در سرور، قبل از ارسال اطلاعات صدا، حجم و فرکانس نمونه برداری صدا فرستاده می شود.

```python
while data_receive:

# sound input
    sample_rate, audio_data = wavfile.read('captured_audio/audio.wav')

# sending request to the server
# print(f"the original sample rate is {bin(sample_rate)}")
    client_object.send(bin(sample_rate).encode())
    client_object.send(bin(audio_data.size).encode())
    # print(f"the original sample rate is {audio_data.size}")
    print("Sending!!")
    client_object.send(audio_data.tobytes())
    print("Before closing")
    client_object.close()
    # client_object.send((b"stop"))
    # receiving response from the server
    # data_receive = client_object.recv(1024)
    # if data_receive:
    #     print("{}: {}".format("SERVER",data_receive.decode('utf-8')))
```

شرط پایان دریافت داده در سرور صدا، رسیدن اطلاعات به حجم دریافت فایل از سوی کلاینت است.

```
while True :

    #  server_input = random.choice(string.ascii_letters)
    #  connection_object.send(server_input.encode('utf-8'))
    print("Reciving voice!!")
    audio_data = np.frombuffer(audio_data,dtype=np.float32)
    input_data = np.append(input_data, audio_data)
    # print(f"The input size of data is {input_data.size}")

    audio_data = connection_object.recv(1024)


    if input_data.size >= audio_size:
        write('captured_audio/Recieved_audio.wav', sample_rate, input_data)  # Save as WAV file
        print("Voice recived")
        # connection_object.close()
        break
```

سرور ها پس از بالا آمدن آی پی خود را از سیستم دریافت می کنند. این کار با تابع get host name انجام می شود.