PROSES AKUISISI DATA MULTIMEDIA BERUPA GAMBAR DAN VIDEO MENGGUNAKAN COAP

Kelompok: Tangan Bicara
Anggota
1. 18220089 David Nathanio Gabriel Siahaan
2. 18220100 Hughie Raymonelika Manggala

Proses akuisisi data multimedia berupa gambar dan video menggunakan protokol CoAP (Constrained Application Protocol) yaitu lingkungan pengembangan diinstal dengan menginstal library AioCoAP pada perangkat laptop/komputer yang akan bertindak sebagai client dan server

1. Untuk pengiriman gambar:
    a. client menggunakan kode program https://github.com/dsp-mc-itb/multimedia-http-mqtt-coap/blob/main/4-COAP/4.1-send-image-client.py untuk mengirimkan data gambar ke server menggunakan protokol CoAP.

```
import aiocoap.optiontypes

import cv2

import aiocoap

import asyncio

import base64



async def send_image():

    # Capture image from webcam

    cap = cv2.VideoCapture(0)

    ret, frame = cap.read()

    cap.release()
```

```
    if not ret:
        print("Failed to capture image")
        return

    # Encode the image to JPEG
    _, buffer = cv2.imencode('.jpg', frame)
    image_data = base64.b64encode(buffer.tobytes())

    # Create CoAP client context
    context = await aiocoap.Context.create_client_context()

    # Create CoAP POST request
    request = aiocoap.Message(code=aiocoap.POST,
uri='coap://127.0.0.1/image', payload=image_data)

    # Send the request and await response
    response = await context.request(request).response

    print('Result: %s\n%r'%(response.code, response.payload))

if __name__ == "__main__":
    asyncio.run(send_image())
```

b. server menggunakan kode program https://github.com/dsp-mc-itb/multimedia-http-mqtt-coap/blob/main/4-COAP/4.1-receive-image-server.py untuk menerima dan memproses data gambar yang dikirimkan oleh client.

```
import aiocoap.resource as resource

import aiocoap

import asyncio

import base64



class ImageResource(resource.Resource):

    def __init__(self):

        super().__init__()



    async def render_post(self, request):

        # Decode the image from the request payload

        image_data = base64.b64decode(request.payload)

        with open('received_image.jpg', 'wb') as f:

            f.write(image_data)

        print("Image received and saved.")

        return aiocoap.Message(code=aiocoap.CHANGED,
payload=b'Image received')



async def main():

    # Resource tree creation
```

```
    root = resource.Site()

    root.add_resource(['image'], ImageResource())



    # Create and start the server

    await aiocoap.Context.create_server_context(root,
bind=('127.0.0.1',5683))

    print("CoAP Server running...")

    await asyncio.get_running_loop().create_future()   # Run
forever



if __name__ == "__main__":

    asyncio.run(main())
```

2. Untuk pengiriman video:
    a. client menggunakan kode program https://github.com/dsp-mc-itb/multimedia-http-mqtt-coap/blob/main/4-COAP/4.2-send-video-client.py untuk mengirimkan data video ke server menggunakan protokol CoAP.

```
import cv2

import aiocoap

import asyncio

import base64



async def send_frame(context, frame):

    # Encode the frame to JPEG

    _, buffer = cv2.imencode('.jpg', frame)
```

```
    frame_data = base64.b64encode(buffer.tobytes())

    # Create CoAP POST request
    request = aiocoap.Message(code=aiocoap.POST,
uri='coap://127.0.0.1/video', payload=frame_data)

    # Send the request and await response
    response = await context.request(request).response
    print('Result: %s\n%r' % (response.code,
response.payload))

async def send_video():
    # Capture video from webcam
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
    cap.set(cv2.CAP_PROP_FPS, 20)

    if not cap.isOpened():
        print("Failed to open webcam")
        return

    # Create CoAP client context
    context = await aiocoap.Context.create_client_context()
```

```
    try:
        while True:
            ret, frame = cap.read()
            if not ret:
                break
            await send_frame(context, frame)
            await asyncio.sleep(0.05)  # Send frames at
approximately 20 FPS
    finally:
        cap.release()


if __name__ == "__main__":
    asyncio.run(send_video())
```

b. Server menggunakan kode program https://github.com/dsp-mc-itb/multimedia-http-mqtt-coap/blob/main/4-COAP/4.2-receive-video-server.py untuk menerima dan memproses data video yang dikirimkan oleh client.

```
import aiocoap.resource as resource
import aiocoap
import asyncio
import base64
import cv2
```

```
import numpy as np



class VideoResource(resource.Resource):

    def __init__(self):

        super().__init__()

        self.frame_count = 0

        self.video_writer = None

        self.frame_width = 640

        self.frame_height = 480

        self.fps = 20.0



    async def render_post(self, request):

        # Decode the frame from the request payload

        frame_data = base64.b64decode(request.payload)

        frame = np.frombuffer(frame_data, dtype=np.uint8)

        frame = cv2.imdecode(frame, cv2.IMREAD_COLOR)



        # Initialize the video writer if not already done

        if self.video_writer is None:

            self.video_writer =
cv2.VideoWriter('received_video.avi',


cv2.VideoWriter_fourcc(*'XVID'),

                                                self.fps,
```

```
(self.frame_width, self.frame_height))


        # Write the frame to the video file
        self.video_writer.write(frame)
        self.frame_count += 1
        print(f"Frame {self.frame_count} received and saved.")


        return aiocoap.Message(code=aiocoap.CHANGED,
payload=b'Frame received')


    async def shutdown(self):
        if self.video_writer:
            self.video_writer.release()


async def main():
    # Resource tree creation
    root = resource.Site()
    video_resource = VideoResource()
    root.add_resource(['video'], video_resource)


    # Create and start the server
    context = await
aiocoap.Context.create_server_context(root,
bind=('127.0.0.1',5683))
```

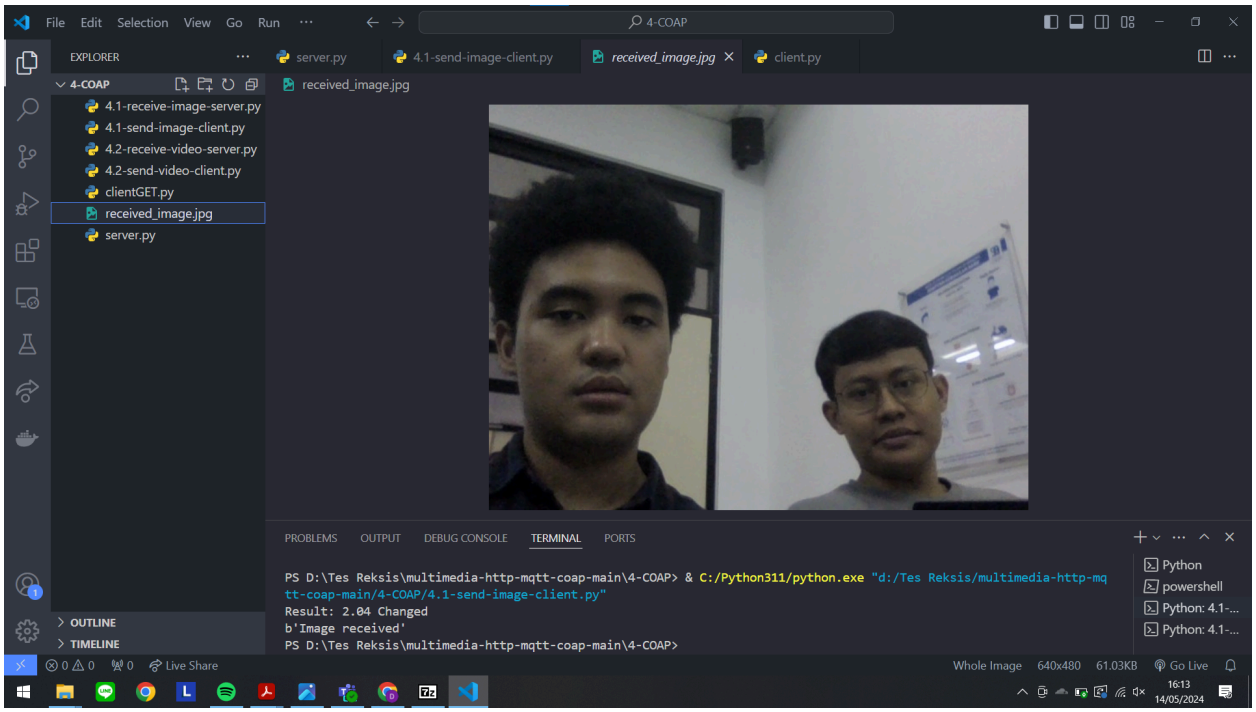
```
    print("CoAP Server running...")


    try:
        await asyncio.get_running_loop().create_future()  # Run forever
    finally:
        await video_resource.shutdown()


if __name__ == "__main__":
    asyncio.run(main())
```

Protokol CoAP dirancang khusus untuk perangkat IoT (Internet of Things) yang memiliki sumber daya terbatas, seperti sensor, aktuator, atau perangkat embedded lainnya. CoAP menggunakan model client-server seperti HTTP, tetapi dengan kepala pesan yang lebih sederhana dan ringan, sehingga cocok untuk perangkat dengan daya komputasi, memori, dan bandwidth yang terbatas. Dalam konteks akuisisi data multimedia, CoAP memungkinkan perangkat client (seperti kamera atau perangkat multimedia lainnya) untuk mengirimkan data gambar atau video ke server secara efisien dan hemat sumber daya.

Instalasi dan Dokumentasi:

Kode yang disimpan dalam repositori GitHub telah diuji pada laptop David yang menggunakan sistem operasi Windows. Dalam proses pengujian, file client dan file server dijalankan secara bersamaan pada laptop yang sama. Tujuan dari pengujian ini adalah untuk memastikan bahwa tidak ada error yang terjadi pada kode dan untuk memeriksa apakah gambar dapat diterima dengan baik oleh server. Dalam kasus ini, client dan server dijalankan pada satu komputer yang sama, yaitu laptop David. Hal ini dilakukan untuk menyederhanakan proses pengujian dan memudahkan identifikasi masalah yang mungkin timbul. Dengan menjalankan client dan server pada satu komputer, komunikasi antara keduanya terjadi dalam lingkungan lokal, sehingga faktor-faktor eksternal seperti koneksi jaringan tidak mempengaruhi hasil pengujian. Setelah file client dan server dijalankan, gambar berhasil diterima di sisi server. Ini menunjukkan bahwa kode yang ditulis berfungsi dengan baik dalam mentransfer data gambar dari client ke server melalui komunikasi lokal. Proses pengiriman dan penerimaan data gambar berlangsung tanpa gangguan atau error yang berarti.

Untuk memvisualisasikan hasil yang diperoleh, gambar yang diterima di sisi server ditampilkan pada antarmuka atau server. Ini memungkinkan pemrogram untuk memverifikasi bahwa gambar yang dikirim oleh client telah diterima dengan benar oleh server dan dapat ditampilkan dengan baik.



Hasil Video

Ini adalah hasil disaat menjalankan file *client* dan *server* untuk proses pengiriman video secara bersamaan. Video berhasil dibentuk dengan format file .avi dan digunakan media player bawaan dari Windows untuk melakukan *play* video tersebut. Gambar ini adalah *screenshot* ketika video berhasil di *play* kemudian dilakukan *pause* di tengah-tengah video.

Properties video