Overview

Using a standard Yocto Poky environment, you are required to create a bootable disk image (for QEMU) that includes a set of microservices responsible for processing video data. You should be able to demonstrate what you have implemented, and run through your code and design choices when we meet next in person or virtually.

Requirements

1. Disk Image with Microservices:
   - Build a disk image with Yocto that can run in QEMU.
   - The image should contain several microservices that communicate via a queue system.

2. Microservices Functionality:
   - Service A (Frame Publisher):
     - Reads an MP4 video file.
     - Extracts individual frames from the video.
     - Publishes each frame to a messaging queue (you can use MQTT, Linux message queues, sockets, etc.).

   - Service B (Frame Resizer):
     - Subscribes to the queue and receives frames published by Service
     - Resizes each frame.
     - Publishes the resized frame back into the queue.

   - Service C (Frame Saver):
     - Subscribes to the queue to receive resized frames.
     - Writes the received frames to disk.

   - Service D (Optional - Queue Monitor):
     - Optionally, implement a service that monitors and displays statistics (e.g., message counts, latencies) from the queuing mechanism.

3. Implementation Languages:
   - At least one of the microservices must be implemented in C++.
   - The remaining services can be developed in Rust, C#, Python, or any other language of your choice.

4. Final Deliverable:

- A Yocto-generated disk image that, when launched in QEMU, runs all the above microservices seamlessly.