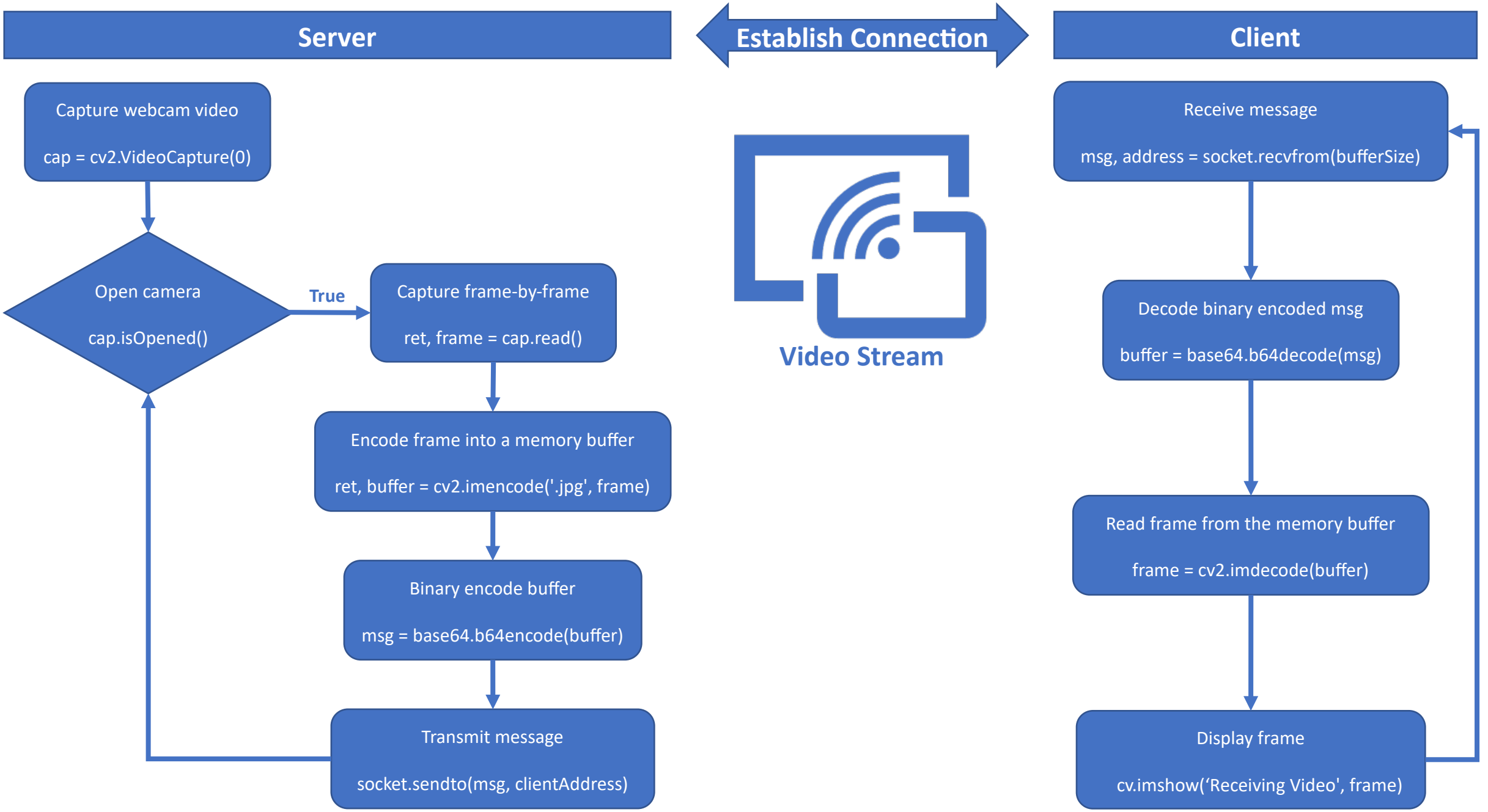
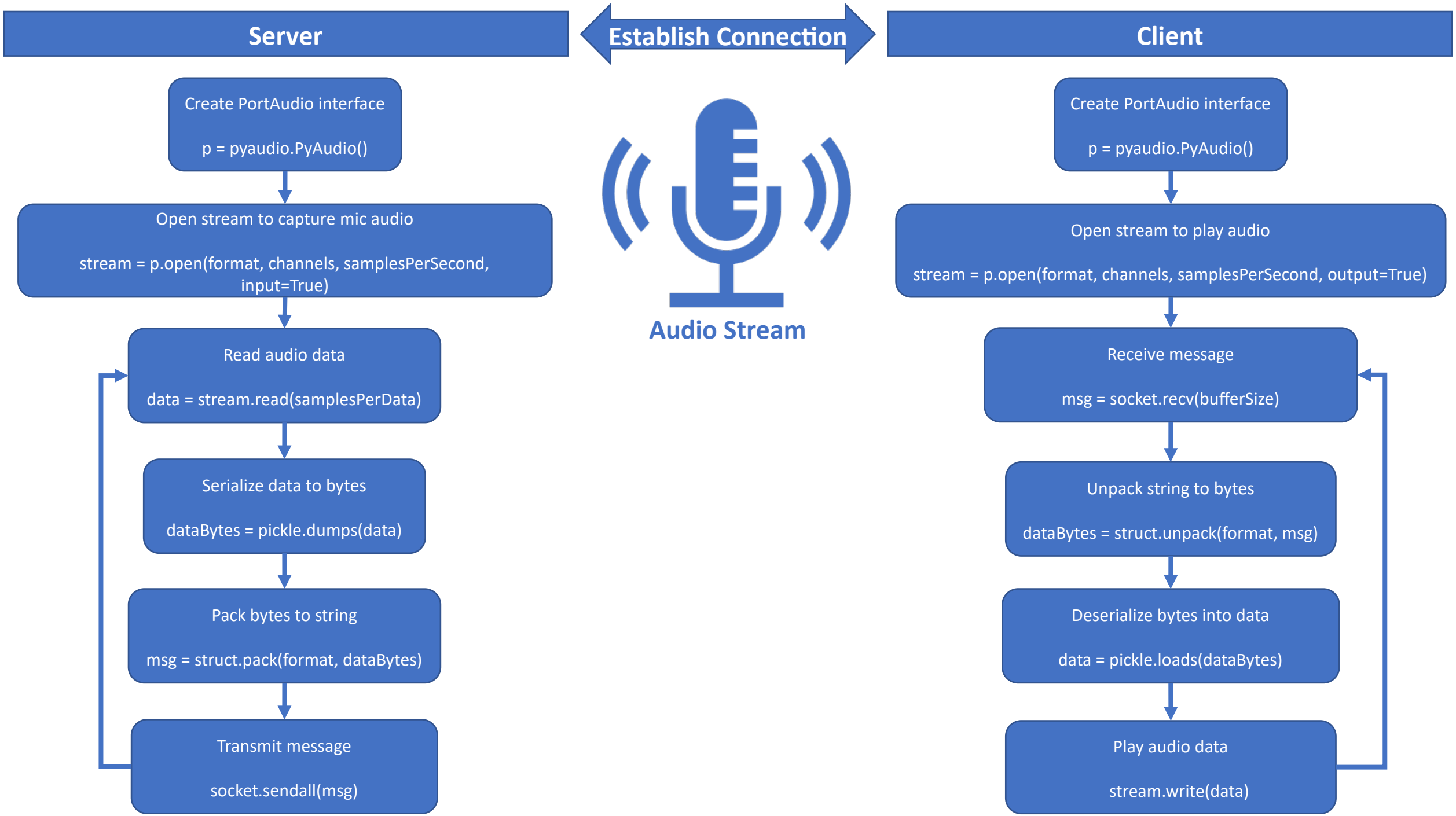
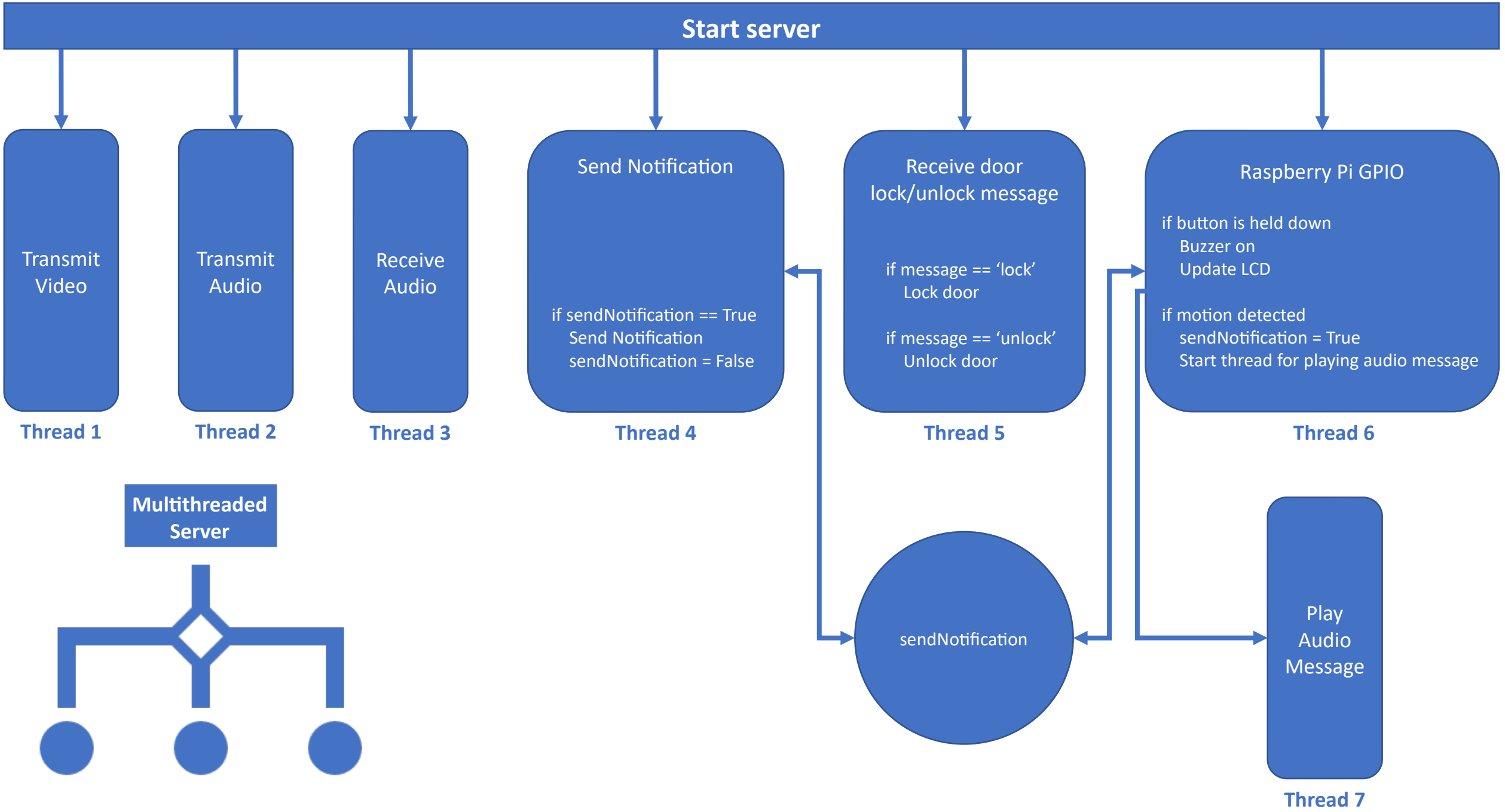

Server
Establish Connection
Client
Capture webcam video
cap = cv2.VideoCapture(0)
Open camera
cap.isOpened()
True
Capture frame-by-frame
ret, frame = cap.read()
Encode frame into a memory buffer
ret, buffer = cv2.imencode('.jpg', frame)
Binary encode buffer
msg = base64.b64encode(buffer)
Transmit message
socket.sendto(msg, clientAddress)
Video Stream
Receive message
msg, address = socket.recvfrom(bufferSize)
Decode binary encoded msg
buffer = base64.b64decode(msg)
Read frame from the memory buffer
frame = cv2.imdecode(buffer)
Display frame
cv.imshow('Receiving Video', frame)


Server
Establish Connection
Client
Create PortAudio interface
p = pyaudio.PyAudio()
Open stream to capture mic audio
stream = p.open(format, channels, samplesPerSecond, input=True)
Read audio data
data = stream.read(samplesPerData)
Serialize data to bytes
dataBytes = pickle.dumps(data)
Pack bytes to string
msg = struct.pack(format, dataBytes)
Transmit message
socket.sendall(msg)
Audio Stream
Create PortAudio interface
p = pyaudio.PyAudio()
Open stream to play audio
stream = p.open(format, channels, samplesPerSecond, output=True)
Receive message
msg = socket.recv(bufferSize)
Unpack string to bytes
dataBytes = struct.unpack(format, msg)
Deserialize bytes into data
data = pickle.loads(dataBytes)
Play audio data
stream.write(data)


Start server
Transmit Video
Transmit Audio
Receive Audio
Send Notification
if sendNotification == True
Send Notification
sendNotification = False
Receive door lock/unlock message
if message == 'lock'
Lock door
if message == 'unlock'
Unlock door
Raspberry Pi GPIO
if button is held down
Buzzer on
Update LCD
if motion detected
sendNotification = True
Start thread for playing audio message
Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Thread 6
Multithreaded Server
sendNotification
Play Audio Message
Thread 7