NAO Configration IOS + Website

Lakshya Saini

```
from flask import Flask, Response
import cv2
import numpy as np
from naoqi import ALProxy
import vision_definitions

app = Flask(__name__)

# NAO robot IP and Port
NAO_IP = "192.168.86.95"  # Replace 'your_nao_ip' with your NAO
NAO_PORT = 9559  # Default port

# Create a proxy to ALVideoDevice on the robot
try:
    video_service = ALProxy("ALVideoDevice", NAO_IP, NAO_PORT)
except Exception as e:
    print("Could not create proxy to ALVideoDevice: {}".format(
    exit(1)

# Subscribe to the camera feed
resolution = vision_definitions.kVGA  # 640x480
colorSpace = vision_definitions.kRGBColorSpace
fps = 20

camera_index = 0  # 0 for the top camera, 1 for the bottom came

try:
    video_client = video_service.subscribeCamera(
        "python_client", camera_index, resolution, colorSpace, 
except Exception as e:
    print("Could not subscribe to camera: {}".format(e))
    exit(1)

def gen_frames():
    while True:
        # Obtain an image from the robot's camera
        nao_image = video_service.getImageRemote(video_client)

        if nao_image is None:
            continue

        # Get the image size and pixel array.
        image_width = nao_image[0]
        image_height = nao_image[1]
        array = nao_image[6]
        image_string = bytes(bytearray(array))

        # Convert the string to an image
        img = np.frombuffer(image_string, dtype=np.uint8)
        img = img.reshape((image_height, image_width, 3))

        # Encode the frame in JPEG format
        (flag, encodedImage) = cv2.imencode(".jpg", img)
        if not flag:
            continue

        # Yield the encoded image in byte format
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' +
               encodedImage.tobytes() + b'\r\n')



@app.route('/video_feed')
def video_feed():
    # Return the response generated along with the specific medi
    # type (mime type).
    return Response(gen_frames(), mimetype='multipart/x-mixed-re

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

```
NAO Config
from naoqi import ALProxy
import vision_definitions

IP = "192.168.86.95"  # Replace with your NAO robot's IP address
PORT = 9559  # Default port for NAOqi

# Create a proxy to ALVideoDevice
video_service = ALProxy("ALVideoDevice", IP, PORT)

# Subscribe to the camera feed
resolution = vision_definitions.kVGA  # 640x480
colorSpace = vision_definitions.kRGBColorSpace
fps = 20

camera_index = 0  # 0 for the top camera, 1 for the bottom camer

video_client = video_service.subscribeCamera(
    "python_client", camera_index, resolution, colorSpace, fps)

# Now video_service.getImageRemote(video_client) can be used to
```

```
Live Stream on Website
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>NAO Robot Live Feed</title>
</head>
```

```
<body>
    <h1>NAO Robot Live Stream</h1>
    <img src="http://10.211.55.7:5000/video_feed" alt="Live Stre
</body>
</html>
```

```
Livestream Application

import SwiftUI
import AVKit

// Define a UIViewRepresentable struct to wrap the LivestreamVie
struct LivestreamViewControllerPreview: UIViewRepresentable {

    func makeUIView(context: Context) -> UIView {
        // Create an instance of your LivestreamViewController
        let viewController = LivestreamViewController()

        // Add the view controller's view to a UIView
        let view = UIView()
        view.addSubview(viewController.view)

        // Set autoresizing mask to make the view resize with it
        viewController.view.translatesAutoresizingMaskIntoConstr

        // Add constraints with margins for iPhone
        NSLayoutConstraint.activate([
            viewController.view.leadingAnchor.constraint(equalTo
            viewController.view.trailingAnchor.constraint(equalT
            viewController.view.topAnchor.constraint(equalTo: vi
            viewController.view.bottomAnchor.constraint(equalTo
        ])

        return view
    }
    func updateUIView(_ uiView: UIView, context: Context) {
        // Update the view if needed
    }
}

// Preview Provider
struct LivestreamViewControllerPreview_Previews: PreviewProvide
    static var previews: some View {
        LivestreamViewControllerPreview()
    }
}
```

```
LiveStream Application

// LivestreamViewController.swift

import UIKit
import AVFoundation

class LivestreamViewController: UIViewController {

    var player: AVPlayer!

    override func viewDidLoad() {
        super.viewDidLoad()

        // URL of the livestream
        guard let url = URL(string: "http://10.211.55.7:5000/vi
            fatalError("Invalid URL")
        }

        // Create AVPlayerItem
        let playerItem = AVPlayerItem(url: url)
        // Initialize AVPlayer with the player item
        player = AVPlayer(playerItem: playerItem)

        // Create AVPlayerLayer to display video
        let playerLayer = AVPlayerLayer(player: player)
        playerLayer.frame = view.bounds

        // Add AVPlayerLayer to the view's layer
        view.layer.addSublayer(playerLayer)

        // Start video playback
        player.play()
    }
}
```