목적

Icom IC-706 모델을 기반으로 원격 무선국을 운영하기 위한 장치를 구축.

ESP32 AudioKit 상용 개발 키트를 활용한 저렴한 원격 시스템 구축

구성

IC-706 무전기 1대 / 리모트 케이블(OPC-581) / ESP32 AudioKit 2EA / Raspberry pi 3B or 4

작동 원리
*참고사이트 :

1. SM7LCB, Radio ICOM IC-706 (lcbsweden.com)
2. IC-706 Archives - OZ9AEC Website

라즈베리파이를 이용한 리모트 프로그램을 A1S를 사용하여 바꾸어 제작.

아두이노 IDE를 이용하여 누구나 사용하기 쉽게 프로그램 제작.

A1S는 서로 UDP 프로토콜을 이용하여 Server, Client를 구성하였으며 Audio는 AudioKit 라이브러리를 사용, 데이터량 및 컨트롤 데이터 부하를 줄이기 위해 오디오 압축 코덱(OPUS)을 사용

*사용 라이브러리

1. https://github.com/pschatzmann/arduino-audiokit
2. https://github.com/sh123/esp32_opus_arduino

나머지는 아두이노 기본 라이브러리를 사용함.

IC-706 판넬과 RIG 본체는 시리얼 데이터(TTL)로 연결되 작동하고 있으며

참고 사이트에서 참고할만한 Keepalive(연결유지 및 데이터 분리하는 요령) 정도 참고하였습니다. 그외 나머지 부분은 시리얼을 직접 읽어 가며 해석해서 구현하였습니다.

판넬에서는 조작에 관련된 데이터를 본체로 보내주고 본체는 받은 데이터를 판넬에 보이도록 다시 종합적으로 한번에 데이터를 보내줍니다. (판넬에서는 주파수만 변경하는 데이터를 보내는데 본체는 주파수를 포함한 LCD 전체 화면 데이터를 한번에 판넬이 보내 주는 구조입니다.)

아두이노는 3-4가지 프로세스가 동시에 구동되고 있습니다.

1. 시리얼 브릿지 프로세스(공통)

2. 오디오 송수신 프로세스(공통)
3. 버튼 송수신 프로세스 (공통)
4. 연결유지 프로세스 (서버만 해당)
5. Wifi 매니지먼트 프로세스 (클라이언트만 해당)

4가지가 아두이노에서 구동되고 있습니다. 아두이노는 기본적으로 멀티테스킹이 지원 되지 않으므로 ESP32의 Vtask 구현으로 멀티테스킹으로 구현되어 4가지가 동시에 구동 되고 있습니다.

서버와 클라이언트는 UDP 프로토콜로 오디오 데이터 및 시리얼 데이터를 주고 받고 있습니다.

한번에 과도한 데이터를 보내거나 받으면 프로세스가 처리를 못하여 다운되는 증상이 있어 컨트롤을 좀더 더디게 만들었습니다. 오디오 품질을 확보하려니 방법이 없었습니다. (개선사항)

전원 온/오프 프로세스는 눈속임용으로 구현했습니다.

클라이언트에서 판넬 power버튼을 누르면 서버쪽으로 데이터가 전달되어 서버에서는 리그 본체의 PWK와 GND를 릴레이로 붙여 주어 전원이 켜집니다. 전원 꺼짐은 판넬에서 0.5초이상 누르면 꺼지게 되는데 판넬에 공급되는 외부 8V 전원이 1초 정도 꺼졌다가 켜짐니다. 이때 판넬 전원은 off 되는 것 처럼 보이나 실제로는 8V 전원이 계속 판넬에 공급이 되고 있는 중입니다. (이 부분도 개선 사항) 버튼 데이터를 계속 감지하고 있으면 프로세스 부하율이 올라가 A1S가 오작동 합니다.

오디오 송수신 파트는 OPUS 코덱을 사용 압축해서 데이터를 전송 합니다. 품질을 높이기 위해 CD급 음질로 전송하는데 데이터 부하가 상당합니다. 낮추면 음질은 손해보나 컨트롤을 좀더 민첩하게 만들 수 있습니다.

PTT 구현 파트는 실제로도 판넬과 본체는 PTT를 GND 접점 신호가 아닌 시리얼 데이터로 PTT가 작동하고 있습니다. 마이크 커넥터에는 PTT 포트가 있으나 내부적으로는 모두 시리얼 통신으로 작동 되고 있습니다.

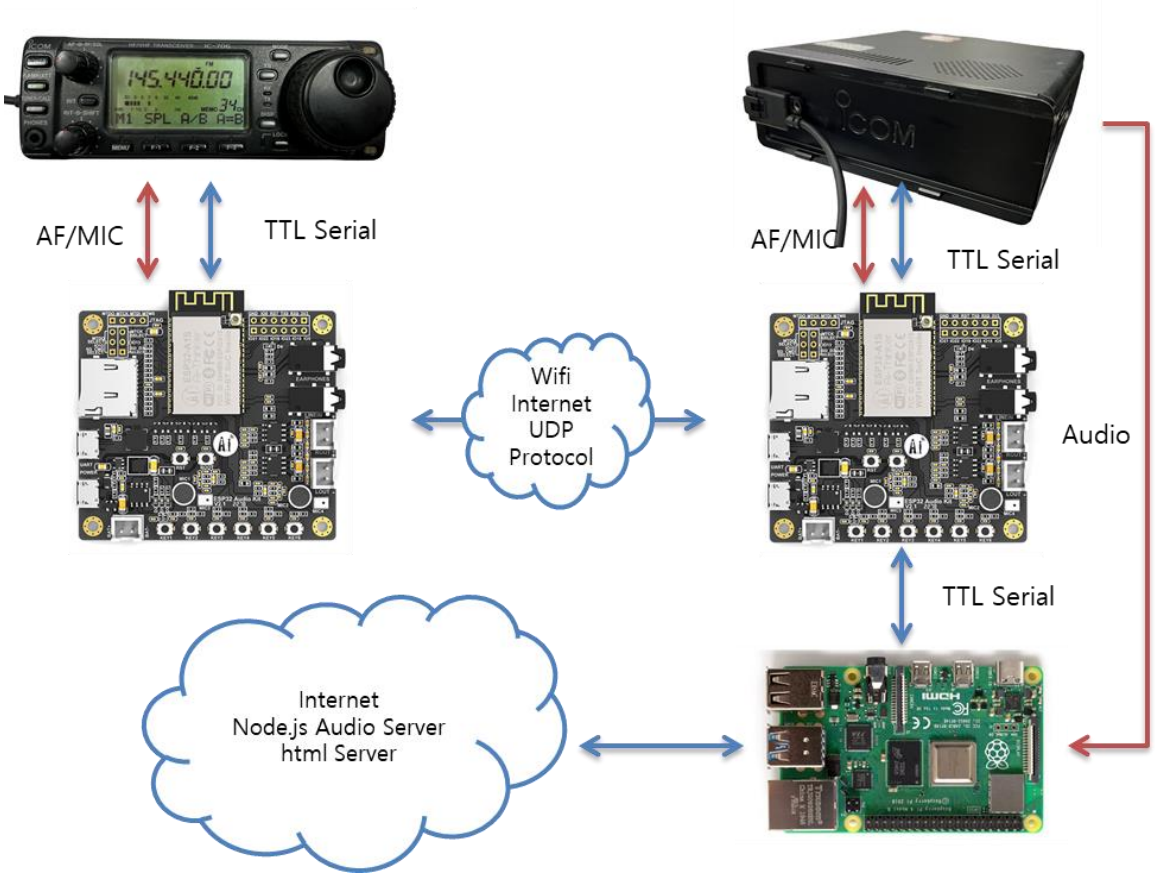
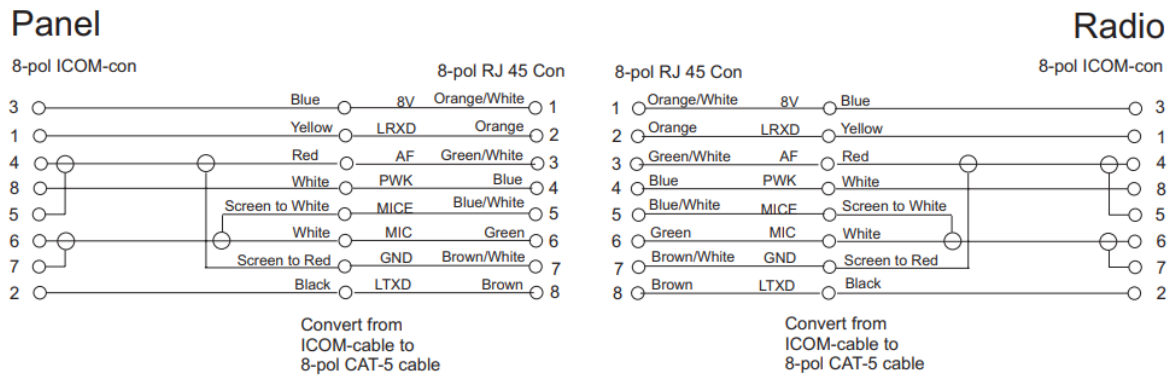


Figure 1 장치 구성도

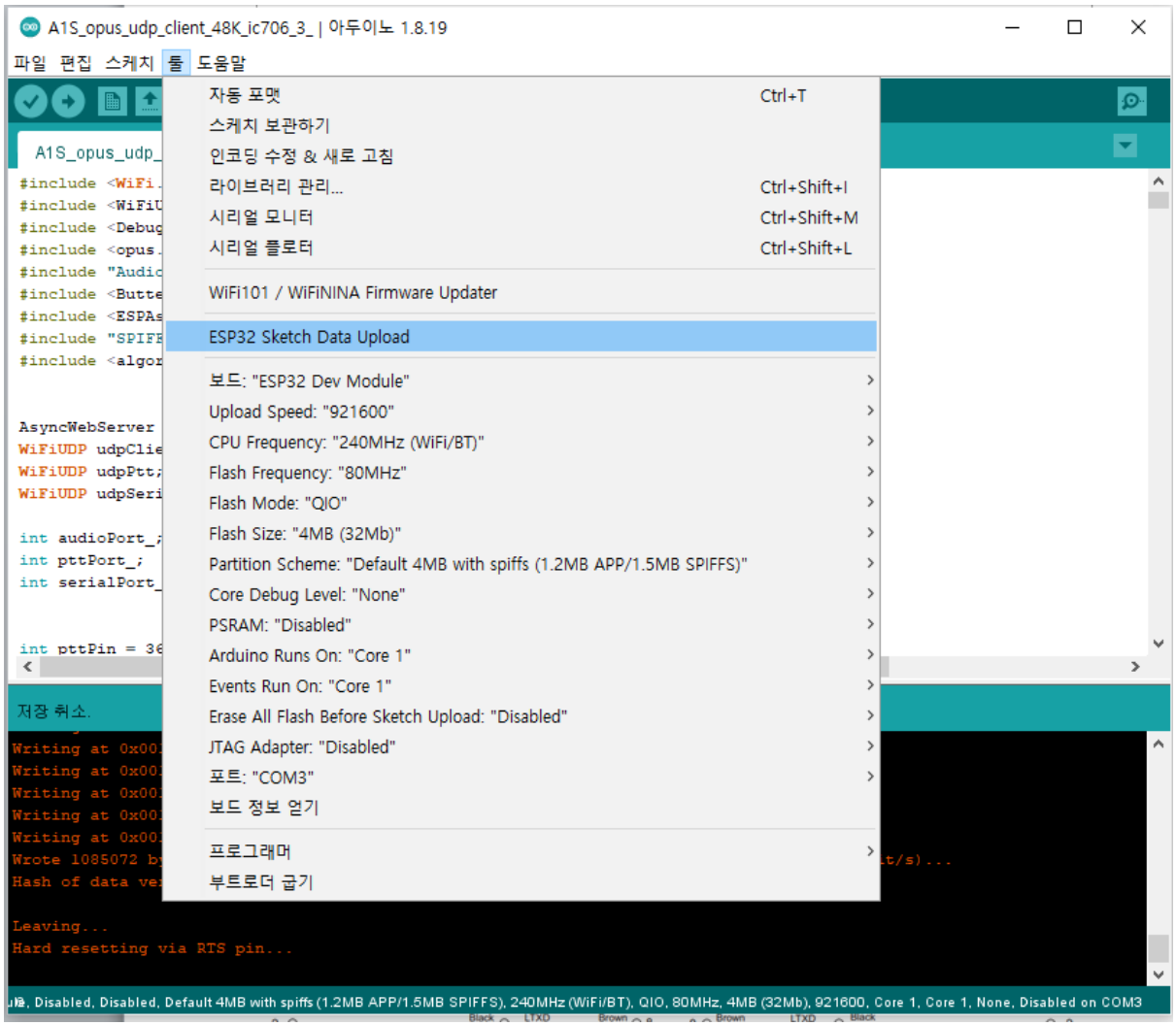
ESP32 AudioKit A1S(이하 A1S)와 연결할 OPC-581 케이블에서 LRXD,LTXD,AF,MIC,PWK,8V,GND 를 사용합니다.



Link : https://www.remoterig.com/schema/ICOM-706%20cable-cat5-1a.pdf

아두이노 설명

클라이언트



클라이언트는 WIFI 매니지먼트 웹 서버가 같이 운영중에 있습니다. ESP32 Sketch Data Upload를 통해 DATA 폴더에 있는 파일이 ESP32 내부 저장소에 업로드 되어야 합니다.

즉 업로딩을 sketch 따로 dada 따로 두번 해야합니다.

업로딩 후 A1S는 와이파이 설정을 해줘야합니다. 처음 구동시에는 192.168.4.1로 AP 모드로 작동되어 사용하고자 하는 공유기의 SSID PASSWORD를 넣습니다. 그럼 공유기에서 아이피를 받아와서 클라이언트 아이피로 접속하면 IC-706 Remote Control 페이지가 보이며 여기에서 서버쪽 아이피 및 각 사용할 포트 입력 해주면 됩니다. 오디오:4000 / PTT : 40010 / Serial : 40020 그러면 클라이언트 셋팅은 끝났습니다.

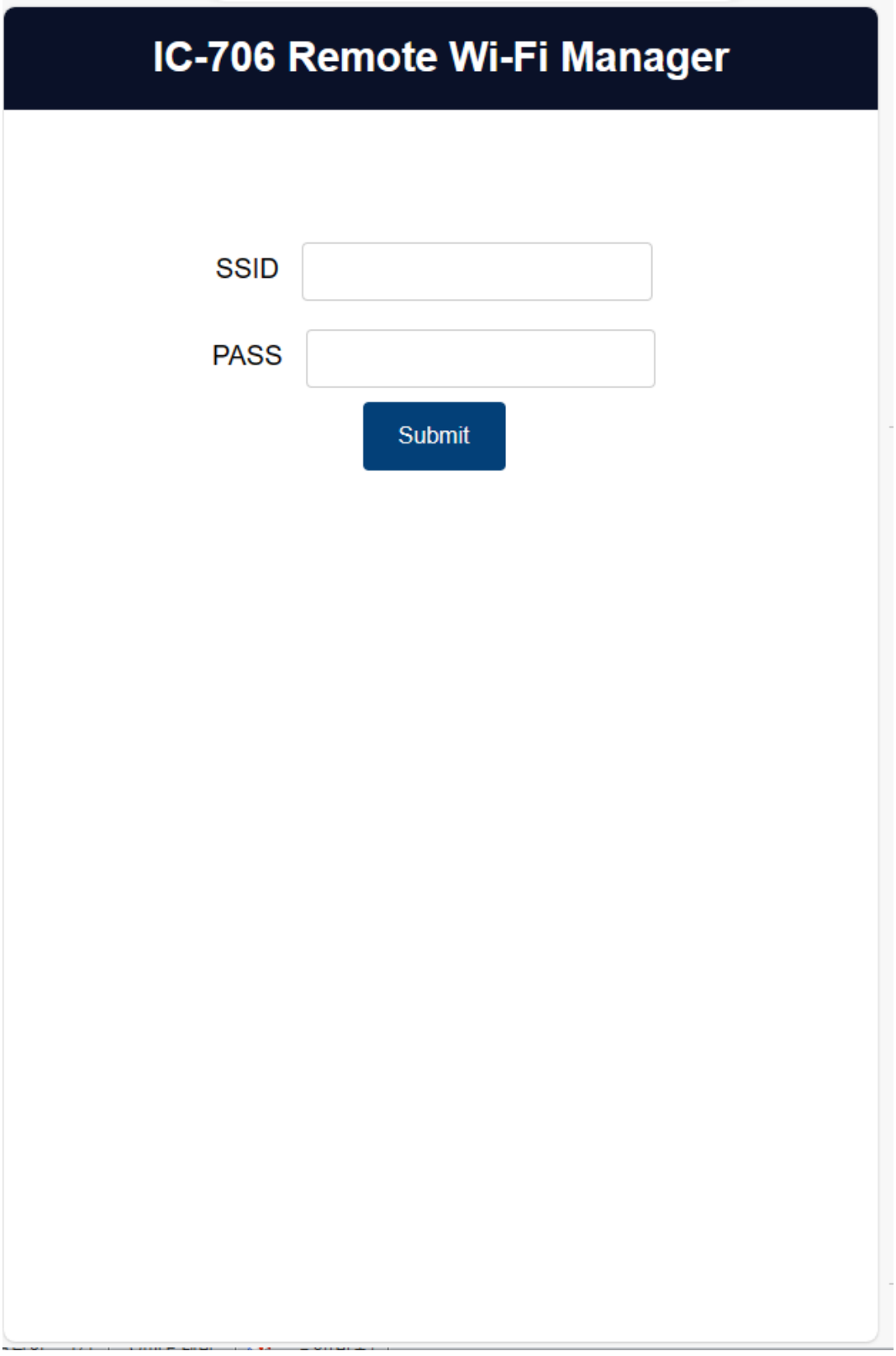


Figure 2 Wifi 매니지먼트 페이지

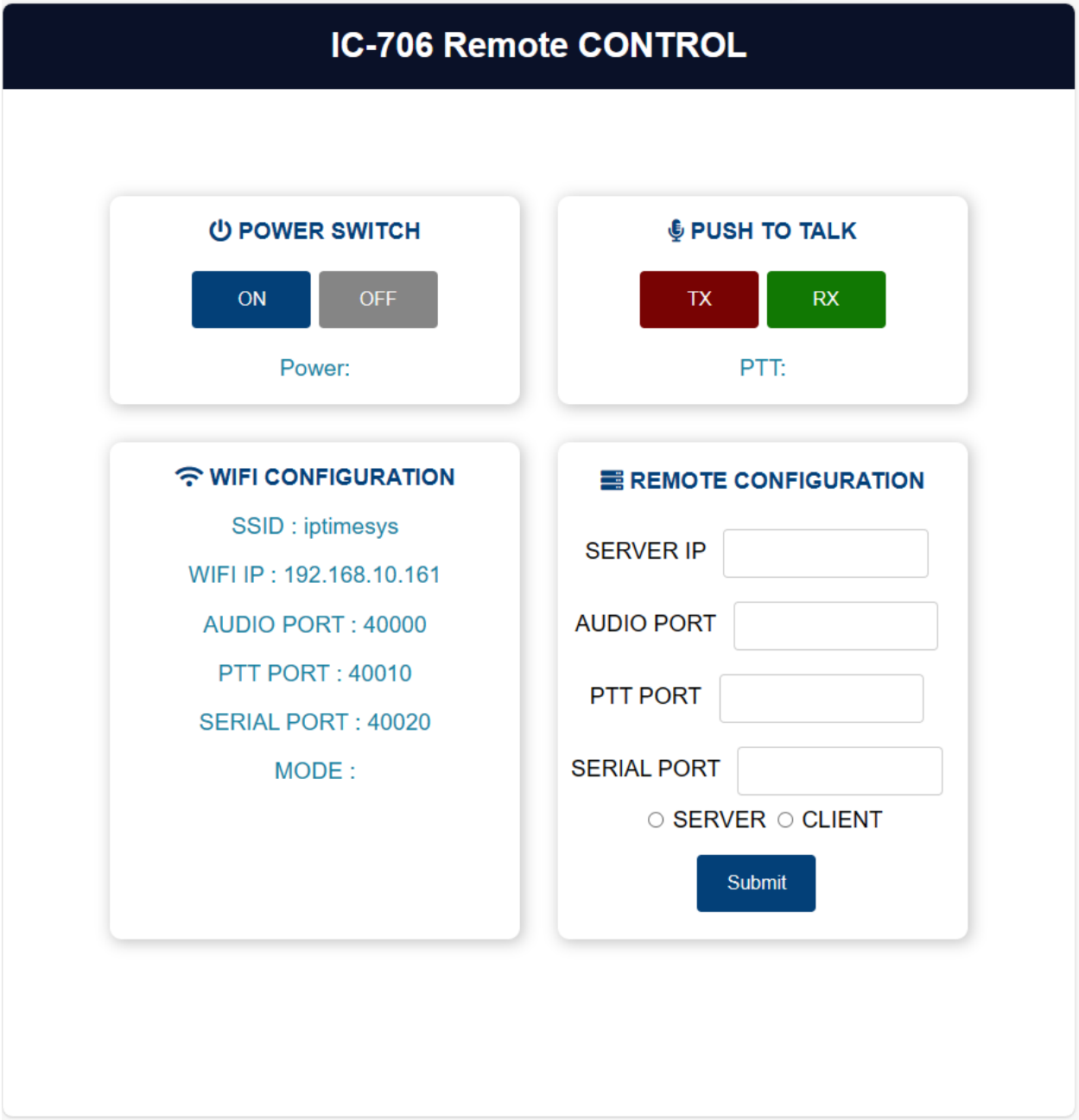
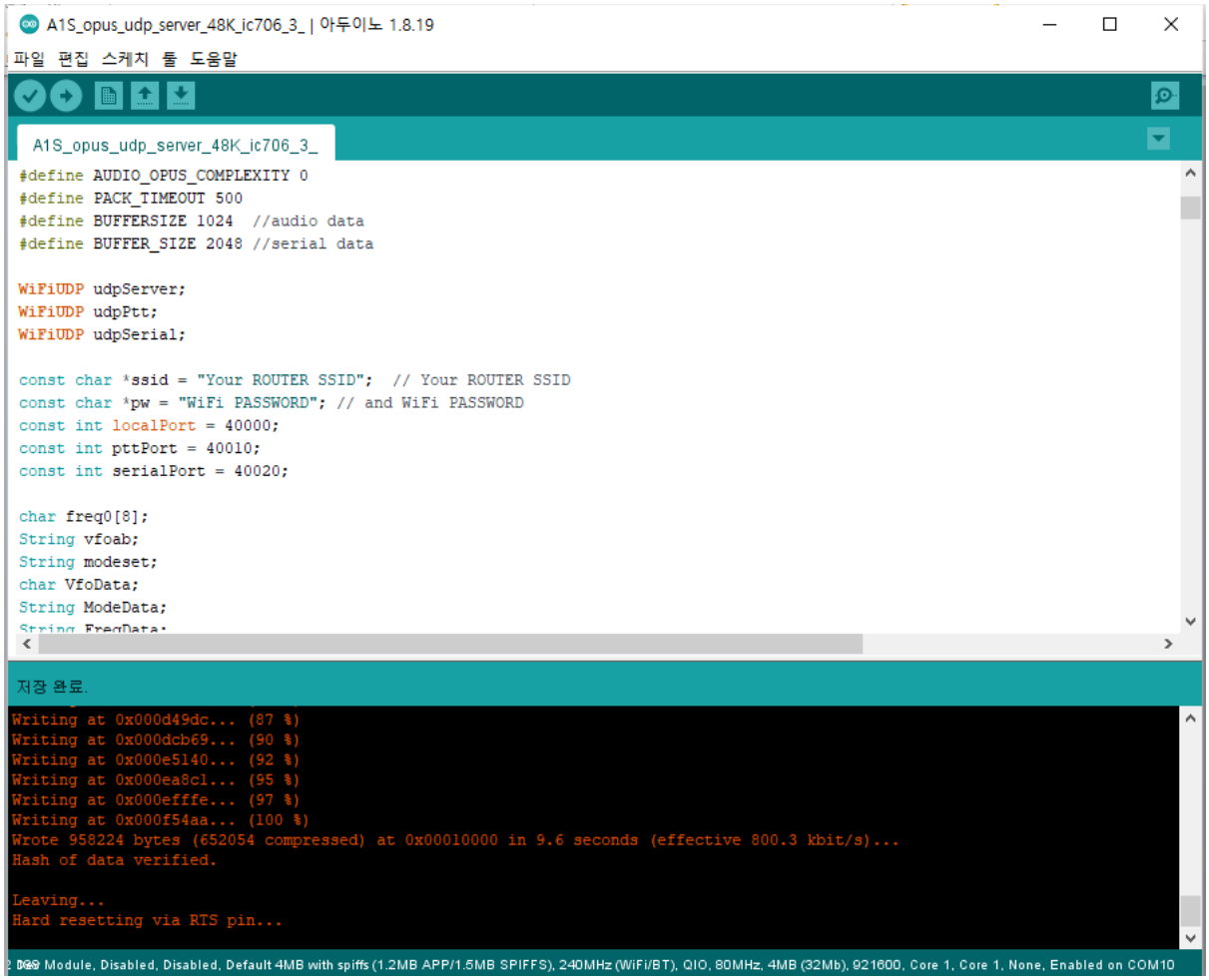


Figure 3 클리이언트 셋업 페이지

서버쪽 아두이노 셋팅

서버쪽은 wifi 매니지먼트 서버가 작동하지 않습니다. 직접 코드에 ssid/password를 넣으셔야 합니다. 서버를 공유기에 잘 접속하게 만드신 후 서버 아이피를 찾아 클라이언트쪽에서 넣어주면 서로 접속합니다. node서버 및 웹 컨트롤 구성하지 않으면 여기까지 하면 됩니다.

라즈베리파이를 이용해서 node 서버 및 웹컨트롤 하려면 A1S 쪽 uart microusb를 라즈베리파이 USB와 연결 해주면 됩니다. (시리얼 데이터를 받아오기 위해서)



라즈베리파이

pi@raspberrypi:~ $ sudo apt-get install nodejs npm 만 입력하면 서버 인스롤 완료

공유된 wsAudioServer 폴더 통쨔로 라즈베리 파이에 복사해서 넣고

pi@raspberrypi:~/wsAudioServer $node server.js 실행하면 서버 작동완료

server.js 파일 수정

audio_client.html server.js

```
const path = require("path");
const express = require("express");
const WebSocket = require("ws");
const { SerialPort } = require('serialport')
const { ReadlineParser } = require('@serialport/parser-readline')
const app = express();
const { spawn } = require('child_process');



const WS_PORT = process.env.WS_PORT || 40040;
const HTTP_PORT = process.env.HTTP_PORT || 8080;

const serialPortPath = '/dev/ttyUSB0';
const baudRate = 115200;
const serialPort = new SerialPort({ path: serialPortPath, baudRate: baudRate }, (err) => {
  if (err) {
    console.error(`Error opening serial port ${serialPortPath} at baud rate ${baudRate}:`, err);
  } else {
    console.log(`Serial port ${serialPortPath} opened at baud rate ${baudRate}`);
  }
});

const parser = serialPort.pipe(new ReadlineParser({ delimiter: '\r\n' }))
const wsServer = new WebSocket.Server({ port: WS_PORT }, () =>
  console.log(`WS server is listening at ws:localhost:${WS_PORT}`)
);

let connectedClients = [];
let arecordProcess = null;
let isPaused = false;

wsServer.on("connection", (ws, req) => {
   const clientId = Math.random().toString(36).substring(7);
  console.log(`Client connected: ${clientId}`);

  connectedClients.push({ id: clientId, socket: ws });

  ws.on("message", (message) => {
     connectedClients.forEach((client, i) => {
```

const WS_PORT = process.env.WS_PORT || 40040; //오디오 소켓 포트

const HTTP_PORT = process.env.HTTP_PORT || 8080; //웹페이지 접속 포트

const serialPortPath = '/dev/ttyUSB0'; //A1S 시리얼 통신 연결

항목은 사용자가 맞게 설정한다.

Audio_client.html 수정

audio_client.html server.js

```
272         player = new PCMPlayer({
273             inputCodec: 'Int16',
274             channels: 1,
275             //sampleRate: 16000,
276             sampleRate: 44100,
277
278         });
279
280         canvas = document.getElementById('waveformCanvas');
281         canvasContext = canvas.getContext('2d');
282
283         const WS_URL = 'ws://61.77.62.48:40040';
284         var ws = new WebSocket(WS_URL)
285         ws.binaryType = 'arraybuffer'
286
287         ws.addEventListener('message', function (event) {
288            const data = event.data;
289            if(playon){
290            drawWaveform(data);
291            player.feed(data);
292            }
293
294         });
295
296         ws.onmessage = function (event) {
297         data = event.data;
298
299
300         if (data.startsWith("RIGDATA")) {
301         var str = data;
302         var freqdata = str.substr(7,10);
303         var vfodata = str.substr(17,1);
304         var modedata = str.substr(18,4);
305         var menu = str.substr(22,2);
306         var f1 = str.substr(24,3);
307         var f2 = str.substr(27,3);
308         var f3 = str.substr(30,3);
309         var mch = str.substr(33,2);
310
311
```

C:₩Users₩NET₩AppData₩Roaming₩MobaXterm₩slash₩RemoteFi UNIX HTML 423 lines Row #1 Col #1

const WS_URL = 'ws://61.77.62.48:40040'; 항목은 자신의 라즈베리파이 아이피를 설정한다.

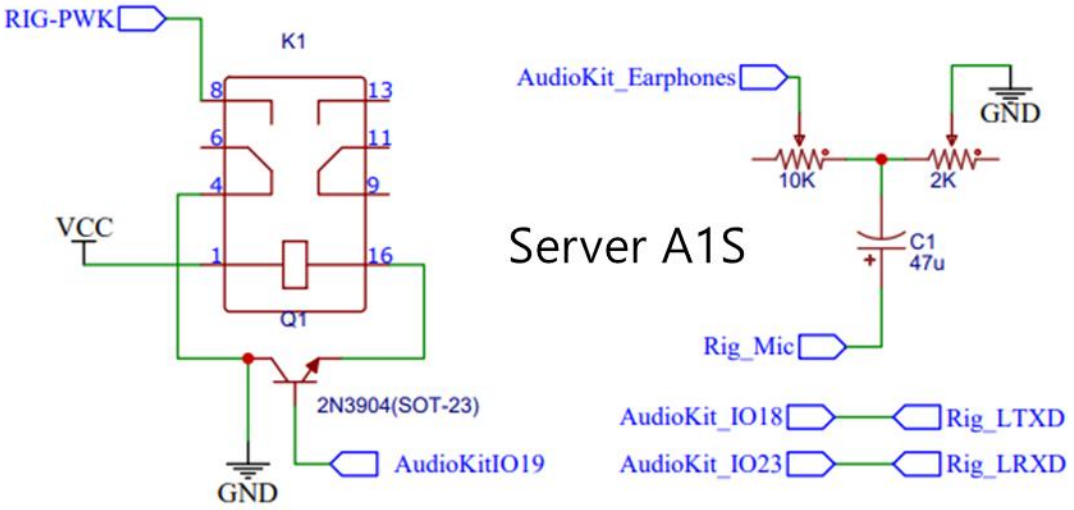
라즈베리파이에서는 IC-706에서 오디오 신호를 가져와 마이크 입력이 넣는다. (웹 페이지에서 오디오 송줄 하기위함)

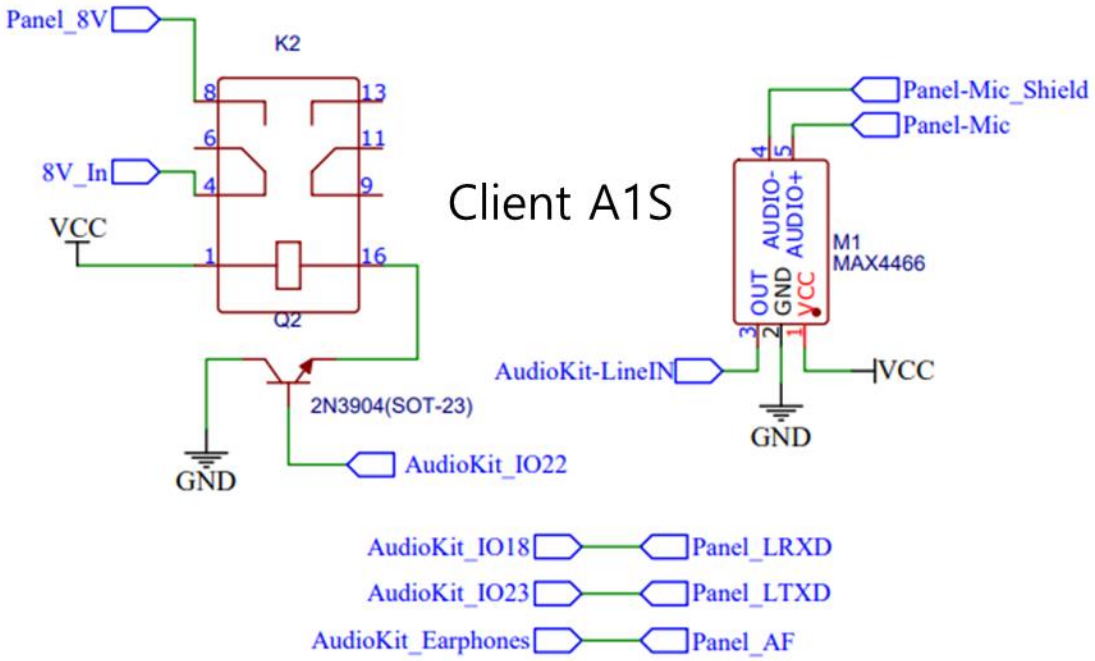
또한 A1S에서 컨트롤 신호를 가져와야함 (라즈베리파이에 A1S를 usb 접속 시키므로 시리얼 데이터를 주고 받는 연결이 된다.)

웹 컨트롤 화면 아직 (시그널 표시 및 주파수 변경 안됨 개선사항)

Node.js 서버는 간헐적 다운이 일어나 주기적으로 다시 실행해 줘야함(개선사항)

하드웨어 연결


RIG-PWK
K1
8
13
6
11
4
9
VCC
1
16
Q1
2N3904(SOT-23)
AudioKitIO19
GND
AudioKit_Earphones
GND
10K
2K
C1
47u
Server A1S
Rig_Mic
AudioKit_IO18
Rig_LTXD
AudioKit_IO23
Rig_LRXD


Panel_8V
K2
8
13
6
11
8V_In
4
9
VCC
1
16
Q2
2N3904(SOT-23)
GND
AudioKit_IO22
Client A1S
Panel-Mic_Shield
Panel-Mic
4
5
AUDIO-
AUDIO+
M1
MAX4466
OUT
GND
VCC
3
2
1
AudioKit-LineIN
VCC
GND
AudioKit_IO18
Panel_LRXD
AudioKit_IO23
Panel_LTXD
AudioKit_Earphones
Panel_AF