Audio Server + Audio Client

Se describe un sistema Cliente-Servidor para enviar y recibir audio entre 2 sistemas ESP32.

Servidor

El servidor está formado por una módulo ESP32 Doit, un modulo codificador de audio a I2S PCM8108 y un módulo decodificador de I2S a audio PCM5102.

El software está desarrollado en el entorno Arduino compatible UECIDE y puede encontrarse en https://github.com/ramonlh/Audio-Client-Server.

El cliente se conecta a una red WiFi que está definida en el código (hay que cambia ésto) en el fichero audioserverUDP.ino:

```
#define WIFI_SSID "myssid"
#define WIFI_PASSWORD "mypassword"
```

El indicativo de llamada se guarda también en este fichero:

```
#define CALLSIGN "EA4GZI"
```

La dirección IP es fija, 192.168.1.148

```
IPAddress EEip={192,168,1,148};   // 4 bytes, dirección IP
```

Ésto es necesario porque se precisa abrir el puerto **3334** y redirigirlo a la IP 192.168.1.148 en el router para que sea posible acceder al servidor desde internet.

Cuando el servidor arranca y se conecta a la red extrae la dirección IP externa accediendo al servidor "i**canhazip.com**".

A continuación envía mediante MQTT al servidor "**broker.mqtt-dashboard.com**" los siguientes *Topics*

ubitx/audioserver/EA4GZI/ip	IP externa
ubitx/audioserver/EA4GZI/port	port (3334)
ubitx/audioserver/EA4GZI/rate	rate value (16000)
ubitx/audioserver/EA4GZI/bits	bits value (24)

Cliente

El cliente consiste en un módulo LyraT, basado en un ESP32 y que incluye un codec ES8388, lector de tarjetas SD y varios pulsadores programables.

El software está desarrollado en el entorno ESP-ADF y puede encontrarse en https://github.com/ramonlh/Audio-Client-Server.

Los parámetros para la conexión están definidos en un fichero de la tarjeta SD, llamado "config.txt" y que puede editarse con un editor de texto en un PC.

Línea 1	Indicativo (EA4GZI)
Línea 2	SSID

Línea 3	Password
Línea 4	Modo (STEREO/MONO)
Línea 5	Rate (16000)
Línea 6	Fuente de sonido (MIC/AUX)

Ejemplo

EA4GZI
myssid
mypassword
STEREO
16000
MIC

Al iniciarse y después de pulsar el pulsador "***Play***", el ESP32 lee el fichero, se subscribe al broker MQTT "**broker.mqtt-dashboard.com**" a los Topics, lee los datos recibidos y se conecta a la dirección y puerto indicados mediante el *rate* y *bits* indicados.

ubitx/audioserver/EA4GZI/ip	IP externa
ubitx/audioserver/EA4GZI/port	port (3334)
ubitx/audioserver/EA4GZI/rate	rate value (16000)
ubitx/audioserver/EA4GZI/bits	bits value (24)

Cuando el servidor detecta la nueva conexión entrante empieza a enviar y recibir datos I2S con el audio. El cliente, a su vez, también empieza a enviar y recibir los datos I2S que el codec ES3888 se encarga de decodificar.

Translated by Google Translate

Audio Server + Audio Client

A Client-Server system is described to send and receive audio between 2 ESP32 systems.

Server

The server consists of an ESP32 Doit module, a PCM8108 I2S audio encoder module and a PCM5102 I2S audio decoder module.

The software is developed in the UECIDE compatible Arduino environment and can be found at https://github.com/ramonlh/Audio-Client-Server.

The client connects to a WiFi network that is defined in the code (this must be changed) in the audioserverUDP.ino file:

```
#define WIFI_SSID "myssid"
#define WIFI_PASSWORD "mypassword"
```

The call sign is also saved in this file:
```
#define CALLSIGN "EA4GZI"
```

The IP address is fixed, 192.168.1.148
```
IPAddress EEip = {192,168,1,148}; // 4 bytes, IP address
```

This is necessary because it is necessary to open port 3334 and redirect it to the IP 192.168.1.148 in the router so that it is possible to access the server from the internet.

When the server starts up and connects to the network, it extracts the external IP address by accessing the server “icanhazip.com”.

Then send the following Topics via MQTT to the server "broker.mqtt-dashboard.com"

ubitx / audioserver / EA4GZI / ip external IP
ubitx / audioserver / EA4GZI / port port (3334)
ubitx / audioserver / EA4GZI / rate rate value (16000)
ubitx / audioserver / EA4GZI / bits bits value (24)

Customer

The client consists of a LyraT module, based on an ESP32 and which includes an ES8388 codec, an SD card reader and several programmable buttons.

The software is developed in the ESP-ADF environment and can be found at https://github.com/ramonlh/Audio-Client-Server.

The parameters for the connection are defined in a file on the SD card, called “config.txt”, which can be edited with a text editor on a PC.

Line 1 Indicative (EA4GZI)

Line 2 SSID
Line 3 Password
Line 4 Mode (STEREO / MONO)
Line 5 Rate (16000)
Line 6 Sound source (MIC / AUX)

Example
EA4GZI
myssid
mypassword
STEREO
16000
MIC

When starting up and after pressing the "Play" button, the ESP32 reads the file, subscribes to the MQTT broker "broker.mqtt-dashboard.com" to the Topics, reads the received data and connects to the address and port indicated by the indicated rate and bits.

ubitx / audioserver / EA4GZI / ip external IP
ubitx / audioserver / EA4GZI / port port (3334)
ubitx / audioserver / EA4GZI / rate rate value (16000)
ubitx / audioserver / EA4GZI / bits bits value (24)

When the server detects the new incoming connection it begins to send and receive I2S data with the audio. The client, in turn, also begins to send and receive the I2S data that the ES3888 codec is responsible for decoding.