



# Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey

Ingeniería en Tecnologías Electrónicas

## Orejeras Industriales Electrónicas

Clara Janet Rivera Medina A01173512

Andrés Antonio Bravo Orozco A01630783

Héctor Manuel Castellanos Ruiz A01636282

Fernando Eduardo Rascón Espinoza A01566567

Proyecto integral de tecnologías electrónicas

Rogelio de Jesús Peregrina y Alberto Rodríguez Arreola

26 de noviembre de 2022

## Resumen

En varias áreas de trabajo e industria, las condiciones involucran un nivel alto de ruido que puede ser peligroso y dañino para los trabajadores. Las soluciones actuales no cuentan con una alternativa tecnológica que se planea explorar y desarrollar con este proyecto, con el fin de ofrecer una solución aprovechando fundamentos electrónicos, de señales y de software. Al tomarse el concepto existente de las orejeras industriales limitadoras de ruido, es posible implementar un filtro del ruido mediante un sistema de audio digital (haciendo uso de bocinas y micrófono) que elimine los inconvenientes que produce esta solución en específico, entre los cuales está la dificultad al escuchar. El diseño y procedimiento de este proyecto se muestra en este documento.

## ÍNDICE

<b>Introducción</b>	<b>3</b>
<b>Análisis de fundamentos</b>	<b>4</b>
Procesamiento digital de señales	4
DAC	4
Filtro FIR	5
Bluetooth	5
Carga de batería	6
Viabilidad del proyecto	6
<b>Método o procedimiento</b>	<b>8</b>
Planteamiento del sistema	8
Inicialización del protocolo I2S	8
Recepción de la señal de audio	9
Filtrado del audio	10
Envío y recepción de la señal mediante WiFi	11
Aplicación Móvil vinculada por Bluetooth	13
Control de carga	14
Prototipado	19
<b>Análisis de resultados</b>	<b>20</b>
<b>Conclusiones</b>	<b>21</b>
<b>Recomendaciones</b>	<b>22</b>
<b>Bibliografía</b>	<b>23</b>

## Introducción

Los trabajadores en fábricas con maquinaria pesada requieren de tapones, orejeras u algún otro tipo de protección para evitar el daño en los oídos, estas opciones hacen que sea difícil de escuchar a los compañeros y que se produzca más esfuerzo al gritar; incluso, en ciertos casos, los trabajadores optan por retirarse las orejeras para entablar una conversación, lo cual es altamente peligroso; además de daño al sentido del oído al escuchar esta contaminación de ruido con frecuencia, se pueden presentar otros problemas como dolor de cabeza, presión alta, fatiga, irritabilidad, problemas digestivos, y mayor vulnerabilidad a infecciones [1]. Esta pérdida auditiva es permanente y no se puede corregir con ningún dispositivo ni cirugía [2]. Los efectos del ruido tal vez no se sientan inmediatamente, por lo que es simple la decisión de los trabajadores de retirarse las orejeras o tapones auditivos durante un corto periodo de tiempo con la ventaja de escuchar mucho mejor a sus compañeros.

El objetivo principal del proyecto es eliminar la tentación de quitarse los audífonos y la molestia de gritar o no escuchar ocasionalmente a los compañeros. Esto se lograría mediante la propuesta: que las orejeras sean audífonos que permitan escuchar claramente a los demás sin necesidad de gritar. Estos audífonos filtran el ruido de la fábrica que detecta el micrófono. Otro objetivo dentro del proyecto es la inclusión de una conexión a un canal en una app para hablar con los compañeros de trabajo. Las conversaciones ocurren en la nube, mientras que la app solo controla los parámetros del sistema (volumen, filtrado y canal). Otro objetivo es tener una sección de control de carga/descarga de baterías para que el usuario simplemente lo conecte a la línea doméstica para cargarlo, sin tener que reemplazarlas.

Es de suma importancia la comunicación entre los trabajadores, una mala comunicación puede conllevar riesgos o situaciones en las que la integridad física de los trabajadores se vea en peligro. Lo usual es que en lugares en donde el ruido sea muy alto se estén utilizando maquinaria que fácilmente puede dañar a un trabajador si este no toma las medidas necesarias. Nuestro dispositivo tiene como objetivo resolver este problema, ayudará a que los trabajadores puedan comunicarse correctamente entre sí y de esta manera disminuir la probabilidad de un accidente debido a una mala comunicación.

Este sistema se puede implementar en áreas no industriales tales como talleres, aeropuertos o festivales, e incluso actividades como agricultura o jardinería. Las posibilidades no se limitan a la industria, aunque este sería el sector principal.

Para la solución de esta problemática se implementará un par de orejeras que cuenten con micrófono, el cual puede tener calidad limitada dada el presupuesto asignado y la naturaleza del prototipo; sin embargo, se pretende utilizar el micrófono con la funcionalidad mejor adaptada (tal como menos sensibilidad al medio ambiente). Además, el filtro realizado por el ESP32 tampoco es el ideal para un proyecto industrial, pero es muy bueno a nivel prototipo.

Otra limitación es la necesidad de conectarse manualmente a un canal en la nube, mientras que con mayores recursos esto podría automatizarse con proximidad.

Aunque el software y el servidor de la nube es escalable, la funcionalidad del dispositivo solo se mostrará entre 1 solo par. Si se consiguiera el presupuesto necesario para crear más orejeras inteligentes, se podrían conectar a canales de audio junto a los otros.

## Análisis de fundamentos

### Procesamiento digital de señales

El procesamiento digital de señales o DSP por sus siglas en inglés es el proceso mediante el cual una señal analógica continua se convierte a una señal digital para poder ser manipulada matemáticamente por un procesador digital, terminando en su reconstrucción analógica como una señal de salida, este proceso se puede observar en la Fig. 1.

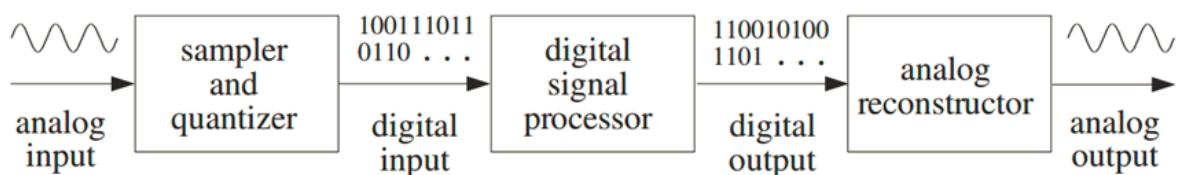


Fig. 1. Proceso de procesamiento digital [1]

La primera etapa del proceso de digitalización es el muestreo, el cual consiste en la conversión de una señal de tiempo continuo en una señal de tiempo discreto tomando sus valores en instantes discretos. La señal de salida no se define a partir de instantes de muestreo. La cuantificación, por otra parte, es la conversión de una señal con valores continuos a un conjunto finito de valores. Después de terminar esta primera etapa ya se cuenta con un equivalente digital de la señal analógica de entrada, la cual tendrá un ligero error dependiendo de los bits de cuantificación usados. La etapa de procesamiento digital de la señal se refiere a las operaciones matemáticas deseadas para modificar la señal digital de entrada. Al terminar la señal se reconstruye analógicamente y se observa como salida. Esto es comúnmente realizado por un DAC [3].

### DAC

Un DAC también llamado Digital to Analog Converter, es un conversor capaz de transformar la señal digital de cualquier fuente en una señal analógica. Los DACs convierten las valores digitales (de N bits) en un valor de voltaje DC. La calidad de un DAC radica en la resolución, es decir, la cantidad de bits por muestra que se pueden convertir en señales analógicas, su tasa de muestreo y la linealidad [4].

## Filtro FIR

Un filtro FIR es aquel que tiene una respuesta finita al impulso y se caracteriza por ser un sistema no recursivo. Un filtro FIR de orden N se describe mediante la Ecuación 1,

$$y(n) = a_0 x(n) + a_1 x(n - 1) + a_2 x(n - 2) + \dots + a_N x(n - N) \quad \text{Ec.1}$$

donde los coeficientes  $a_0$  son los coeficientes del filtro [5].

En este tipo de filtrado no existe retroalimentación. Además, la respuesta al impulso es de duración finita ya que si la entrada se mantiene en cero durante N períodos consecutivos la salida entonces será cero. La estructura de dicho filtro se puede observar en la Fig. 2.

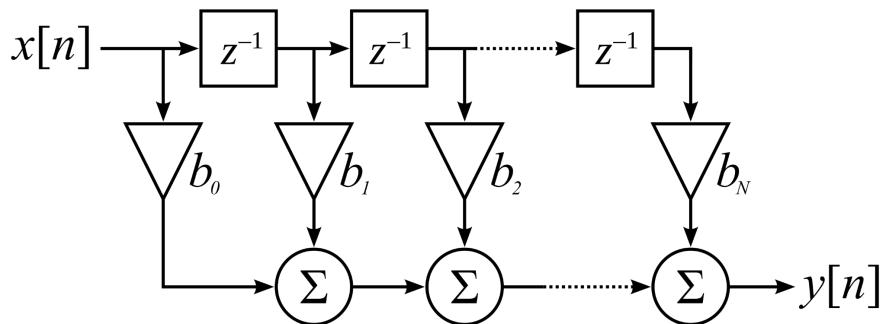


Fig. 2. Estructura de un filtro FIR [3]

Este tipo de filtros tienen un alto interés en aplicaciones de audio ya que son estables y se pueden diseñar para ser lineales y son estables debido a la falta de elementos en el denominador de su función de transferencia.

## Bluetooth

Según SoftwareLab, “Bluetooth es un protocolo de comunicaciones que sirve para la transmisión inalámbrica de datos entre diferentes dispositivos que se encuentran a corta distancia, dentro de un radio de alcance de generalmente diez metros”. Hace uso de la banda ISM, además de las Redes Inalámbricas de Área Personal. Los dispositivos no requieren estar alineados para el uso del protocolo [6].

Los dispositivos deben estar dentro de un radio de alcance que aunque es corto, depende del aparato en cuestión. Los aparatos y sus alcances tienen estas clasificaciones [6]:

- **Clase 1.** Potencia máxima de 100 mW con 100 metros de alcance.
- **Clase 2.** Potencia máxima de 2.5 mW con 5-10 metros de alcance. Son los más comunes.
- **Clase 3.** Potencia máxima de 1 mW y un alcance de un metro.

## Carga de batería

Para la alimentación del dispositivo se utilizarán baterías de litio y polímero (lipo) las cuales son muy utilizadas en dispositivos electrónicos tales como auriculares con Bluetooth, esto debido a que son compactas y recargables. Para el control de su carga se puede utilizar un circuito integrado que incluye al chip TP4056 [7].

## Viabilidad del proyecto

El objetivo del proyecto en pocas palabras es generar una forma de comunicación en un ambiente ruidoso, a continuación se analizan las posibles soluciones a esta problemática, señalando los pros y contras de cada una de las soluciones planteadas.

- Utilización de aplicaciones de mensajería instantánea
- Uso de lenguajes de señas
- Comunicarse utilizando una libreta y pluma
- Designación de pequeños espacios con tratamiento sonoro
- Diseño de audífonos con aislamiento de ruido, micrófono y filtro de voz.

En la Tabla 1 se muestran las ventajas y desventajas de las soluciones con el fin de realizar una comparación y plantear la viabilidad del proyecto.

Tabla 1. Ventajas y desventajas de soluciones del problema

Solución propuesta	Pros	Contras
Utilización de aplicaciones de mensajería instantánea	<ul style="list-style-type: none"> <li>● Fáciles de usar</li> <li>● Diversas aplicaciones ya existentes, no es necesario crear una nueva</li> </ul>	<ul style="list-style-type: none"> <li>● Puede ser un peligro utilizar el teléfono debido a que este puede ser una distracción</li> <li>● Algunas empresas tienen la política de prohibir el uso de celulares en zonas específicas debido a temas de privacidad y de propiedad intelectual</li> </ul>
Uso de lenguajes de señas	<ul style="list-style-type: none"> <li>● Comunicación rápida y efectiva en lugares con ruido</li> </ul>	<ul style="list-style-type: none"> <li>● Difícil de aprender, similar a aprender un nuevo idioma</li> <li>● Gastos de capacitación elevados para enseñar a los empleados lenguaje de señas</li> <li>● La comunicación tiene que ser de frente a frente</li> </ul>

Comunicarse utilizando una libreta y pluma, o una tableta	<ul style="list-style-type: none"> <li>● Solución económica si es con libreta</li> <li>● Fácil de implementar</li> <li>● Fácil de usar</li> </ul>	<ul style="list-style-type: none"> <li>● Los trabajadores tendrían que llevar consigo una pluma y libreta, o una tableta</li> <li>● Método relativamente lento de comunicación</li> <li>● Puede haber casos de una mala comunicación si la letra no es totalmente legible</li> <li>● Adquirir una tableta para cada trabajador puede ser costoso</li> </ul>
Designación de pequeños espacios con tratamiento sonoro	<ul style="list-style-type: none"> <li>● Fáciles de usar</li> <li>● Se podría hablar fácilmente sin importar el ruido del exterior</li> </ul>	<ul style="list-style-type: none"> <li>● Cada vez que los trabajadores quieran hablar se tendrían que ir al espacio asignado</li> <li>● Crear diversas salas con tratamiento sonoro tiene un costo elevado</li> <li>● No viable en fábricas con espacios reducidos</li> </ul>
Diseño de audífonos con aislamiento de ruido, micrófono y filtro de voz.	<ul style="list-style-type: none"> <li>● Fáciles de usar</li> <li>● Permiten una comunicación efectiva y rápida</li> <li>● Permite comunicarse incluso si las personas no se están viendo de frente</li> <li>● Permite realizar otras actividades mientras se hablan con otros trabajadores</li> <li>● Permite comunicarse a largas distancias, incluso entre dos salas separadas.</li> </ul>	<ul style="list-style-type: none"> <li>● Costo considerable del diseño y fabricación del producto</li> <li>● Necesidad de tener una red wifi dentro de las fábricas</li> </ul>

Analizando las posibles soluciones la mejor opción es la utilización de los audífonos con aislamiento de ruido que cuenten con un micrófono con un módulo de filtrado de ruido, ya que es menos costoso que muchas otras soluciones, y cuando no, es una solución más efectiva y moderna que incrementa la productividad y seguridad que se tendría con las alternativas.

## Método o procedimiento

### Planteamiento del sistema

En el diagrama de bloques presentado en la Fig. 4, se muestran los componentes necesitados (un módulo micrófono, un microcontrolador ESP32, audífonos que se adaptarán a orejeras industriales para la atenuación de ruido y un amplificador, el PAM8403), además de un módulo que se encargará de la carga para la batería de litio que alimenta el sistema (TP4056).

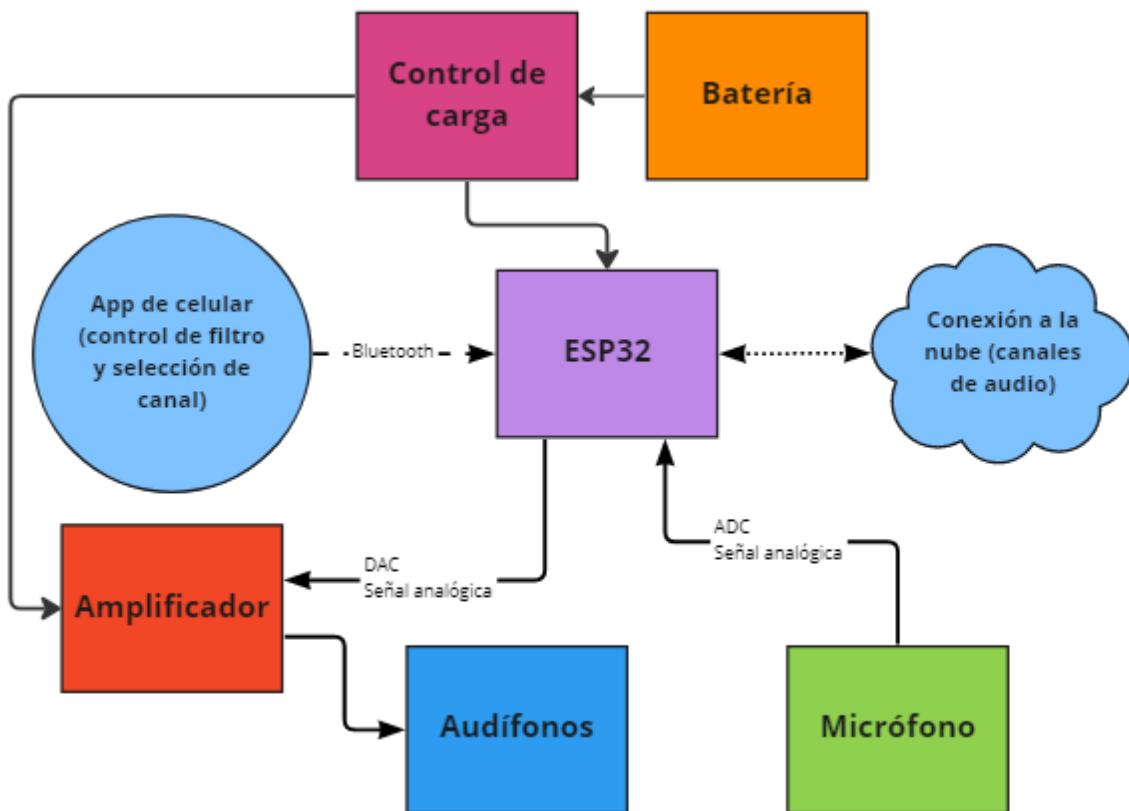


Fig. 4. Diagrama de bloques del sistema

### Inicialización del protocolo I2S

El primer paso para el funcionamiento del proyecto es dar de alta en el sistema el protocolo I2S que es usado por el microcontrolador como un DMA (Direct Memory Access). Este se encarga de reducir la carga del procesador al encargarse él de esta conexión, y hará que el procesamiento del resto del código (como el filtrado y la transmisión-recepción de los datos mediante la nube) se lleve a cabo de forma más rápida. La inicialización se presenta en la Fig. 5 y tiene algunas configuraciones notables. La primera es que el modo involucra la implementación de 2 modos simultáneamente (transmisión y recepción); además, activa el ADC integrado (que se usará para conectarse al micrófono) y el DAC (que se usará para reproducir el audio en los audífonos).

Determinar la frecuencia de muestreo tomó terminar todo el código y medir los tiempos entre la transmisión y recepción de los buses de datos para intentar aproximar el periodo de muestreo a estos tiempos; esto asegura que el audio se escuche en su calidad óptima. La frecuencia de muestreo que se escogió fue de 4000 Hz.

Los bits por muestra son de 16, el formato involucra 2 canales independientes (izquierdo y derecho) y el formato de comunicación es de 2 bytes del canal izquierdo seguidos de 2 bytes del canal derecho, repetitivamente hasta llenar el buffer.

```
void i2s_init(void)
{
    i2s_config_t i2s_config = {
        .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX | I2S_MODE_TX | I2S_MODE_DAC_BUILT_IN | I2S_MODE_ADC_BUILT_IN),
        .sample_rate = 4000,
        .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
        .channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
        .communication_format = I2S_COMM_FORMAT_STAND_MSB,
        .dma_buf_count = 16,
        .dma_buf_len = 128,
        .use_apll = 1
    };
    //install and start i2s driver
    i2s_driver_install(I2S_NUM_0, &i2s_config, 4, NULL);
    //init DAC pad
    i2s_set_dac_mode(I2S_DAC_CHANNEL_BOTH_EN);
    //init ADC pad
    i2s_set_adc_mode(ADC_UNIT_1, ADC1_CHANNEL_7);
    i2s_adc_enable(I2S_NUM_0);
}
```

Fig. 5. Código de inicialización de I2S

## Recepción de la señal de audio

Se consiguió un micrófono adecuado que no es muy sensible al ruido, el MAX4466. Para recibir la señal mediante este sensor, se utiliza la función del protocolo I2S, *i2s\_read()*. El buffer que se usa para capturar la información es *i2s\_read\_buff* y su tamaño fue definido como 200, que es el tamaño de paquete que se usará en el resto del código.

La información que maneja el ADC es de 12 bits, mientras que el tamaño que puede físicamente transmitir el DAC es de 8 bits. Dado esto, fue necesario crear una función para escalar la información antes de manipularla más. El código de la recepción se muestra en la Fig. 6 mientras que el de la función de escalamiento está en la Fig. 7.

```
// READ AND SCALE STAGE
// read data from I2S bus, in this case, from ADC.
i2s_read(I2S_NUM_0, (void *)i2s_read_buff, i2s_read_len, &bytes_read, portMAX_DELAY);
//process data and scale to 8bit for I2S DAC
i2s_adc_data_scale(i2s_temp_buff, i2s_read_buff, i2s_read_len);
```

Fig. 6. Código para la recepción de la señal

```
void i2s_adc_data_scale(uint8_t* d_buff, uint8_t* s_buff, uint32_t len)
{
    uint32_t j = 0;
    uint32_t dac_value = 0;
    for (int i = 0; i < len; i += 2) {
        dac_value = (((uint16_t) (s_buff[i + 1] & 0xf) << 8) | ((s_buff[i + 0]))));
        d_buff[j++] = 0;
        d_buff[j++] = dac_value / 16;
    }
}
```

Fig. 7. Código para el escalamiento del paquete

## Filtrado del audio

Para generar los coeficientes del filtro diseñado, el cual es un pasabanda para las frecuencias de voz (se seleccionó 300 - 1900 Hz), se utilizó el programa *filterDesigner* de MATLAB. Se seleccionó un filtro Least-squares en la categoría de FIR, con un orden de 10. Este diseño se observa en la Fig. 8. Se generaron los coeficientes y se exportaron a un header de C que los declara como un arreglo constante de enteros.

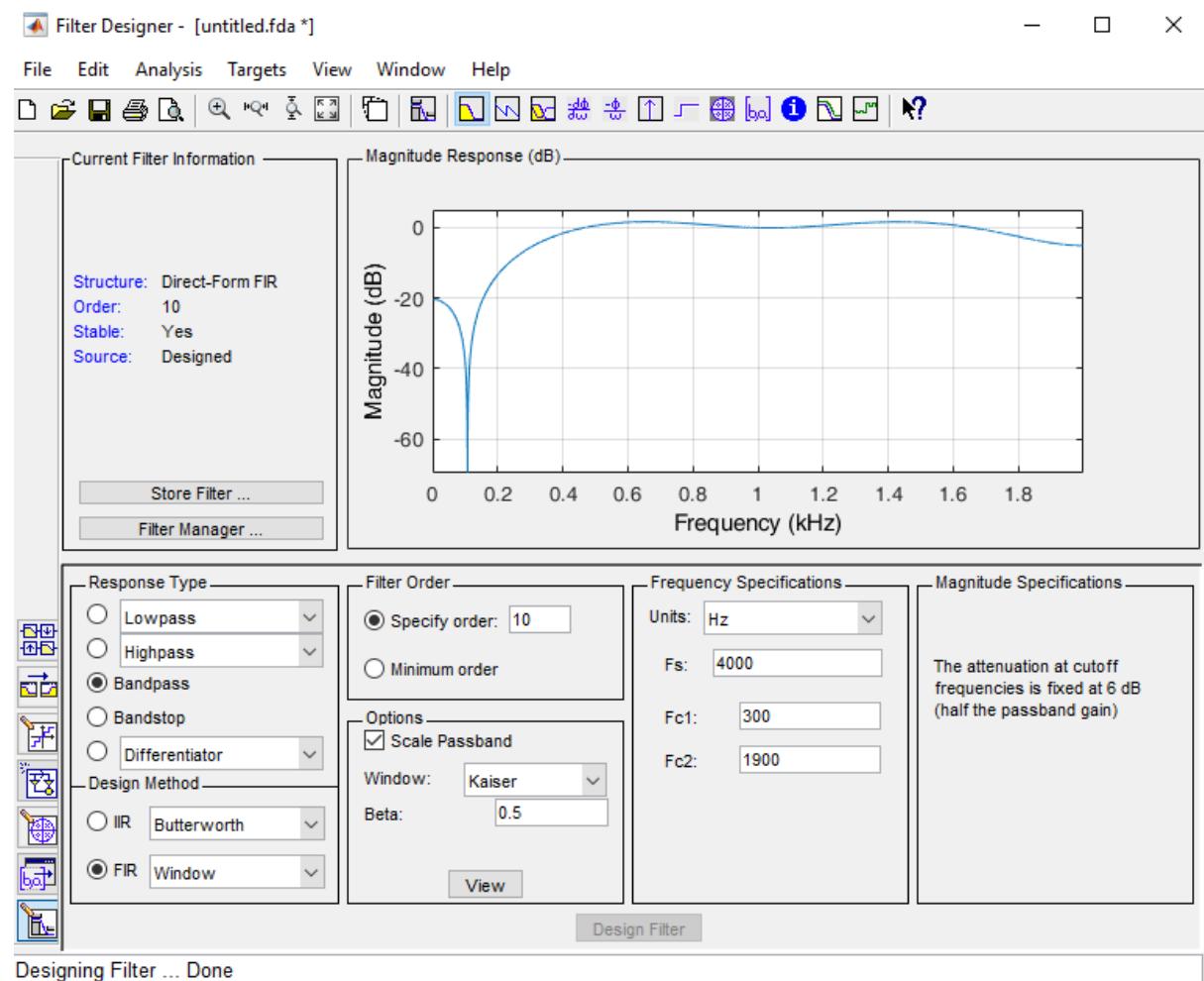


Fig. 8. Diseño del filtro mediante FilterDesigner

Se inicializó el filtro con la línea de código `dsps_fir_init_f32(&myfir, (float *)&coeffs, states, ORDER+1)` para ambos canales. Es necesario mencionar que para que funcione, se deben usar arreglos con varias muestras, por lo que se van recolectando antes de aplicar el filtro. En el ciclo principal, se debe separar el buffer de lectura en canales izquierdo y derecho para asegurar un filtrado correcto; después, se ejecuta el filtro (que entrega valores flotantes) y se convierte de vuelta a un entero de 8 bits desde un rango de [-1,1] con la función `dsps_mulg_f32_ae32()`. Finalmente, se juntan de nuevo ambos canales en un solo buffer. Todo este proceso se muestra en la Fig. 9.

```

// FILTER STAGE
// input converted to separated float channels L and R
for(i=0;i<i2s_read_len;i+=4){
    temp=(int16_t*)(&i2s_temp_buff[i]);
    templ[i>>2]=(float) (*(temp))/32768.0;
    tempr[i>>2]=(float) (*(temp+1))/32768.0;
}
dsps_fir_f32_ae32(&myfirL,templ,templ2,i2s_read_len>>2);
dsps_fir_f32_ae32(&myfirL,templ,templ2,i2s_read_len>>2);

// convert [-1,1] float to integer, i=f*32768 + 32768
dsps_mulc_f32_ae32(templ2,templ,i2s_read_len>>2,32768.0,1,1);
dsps_mulc_f32_ae32(tempr2,tempr,i2s_read_len>>2,32768.0,1,1);
dsps_addc_f32_ae32(templ,templ2,i2s_read_len>>2,32768.0,1,1);
dsps_addc_f32_ae32(tempr,tempr2,i2s_read_len>>2,32768.0,1,1);

// output converted to merged interleaved LR output
for(i=0;i<i2s_read_len;i+=4){
    temp=(int16_t*)(&i2s_temp_buff[i]);
    *temp=(int16_t)(templ2[i>>2]);
    *(temp+1)=(int16_t)(tempr2[i>>2]);
}

```

Fig. 9. Código de la etapa del filtrado

### Envío y recepción de la señal mediante WiFi

Después de recuperar la señal del ADC, los datos recuperados en alrededor de 100 muestras por el microcontrolador emisor se envían a un servidor local, este servidor envía los datos al microcontrolador receptor para su reproducción a través del DAC integrado. El servidor, escrito en Python, se inicializa con los valores de la Fig. 10, lleva a cabo la conexión mostrada en la Fig. 11 y se ejecuta en la Fig. 12. En resumen, se devuelven los valores string a su formato original de byte, se leen los primeros dígitos para observar el canal deseado, y finalmente si los canales son iguales, los sistemas se envían información mutuamente.

```

SHARED_UDP_PORT = 4210

sock_esp1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # Internet # UDP
sock_esp1.connect((UDP_IP_ESP1, SHARED_UDP_PORT))

sock_esp2 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # Internet # UDP
sock_esp2.connect((UDP_IP_ESP2, SHARED_UDP_PORT))

```

Fig. 10. Inicialización y conexión del servidor mediante UDP

```

def loop():
    arr0 = bytearray(200)
    while True:
        dat_esp1 = sock_esp1.recv(2048)
        dat_esp2 = sock_esp2.recv(2048)

        packet_bin_esp1 = bytearray(b' ')
        packet_bin_esp2 = bytearray(b' ')

        channel_esp1 = int(dat_esp1[0:3])
        channel_esp2 = int(dat_esp2[0:3])

        for i in range(200):
            dat_int_esp1 = int(dat_esp1[3:][i * 3:i * 3 + 3])
            dat_int_esp2 = int(dat_esp2[3:][i * 3:i * 3 + 3])
            packet_bin_esp1.append(dat_int_esp1)
            packet_bin_esp2.append(dat_int_esp2)

        if channel_esp1 == channel_esp2:
            sock_esp1.send(packet_bin_esp2)
            sock_esp2.send(packet_bin_esp1)
        else:
            sock_esp1.send(arr0)
            sock_esp2.send(arr0)
    
```

Fig. 11. Código para recepción y transmisión de datos desde el servidor

```

if __name__ == "__main__":
    print("init")
    sock_esp1.send('init_connection_esp1'.encode())
    sock_esp2.send('init_connection_esp2'.encode())
    loop()
    
```

Fig. 12. Inicio del programa del servidor

En cuanto a las operaciones que llevan a cabo los microcontroladores, para transmitir se añade un solo byte de información al principio del paquete que contiene el canal de comunicación deseado. Después, se convierte cada dígito a un string para poder enviarlo al servidor mediante UDP. Esto se muestra en la Fig. 13.

```
// TRANSMIT STAGE
char udp_buffer_string[PACKET_SIZE*3+3];
sprintf(sample_data,"%03d",channel_sel);
udp_buffer_string[0] = sample_data[0];
udp_buffer_string[1] = sample_data[1];
udp_buffer_string[2] = sample_data[2];
for(i=1;i<PACKET_SIZE+1;i++){
    sprintf(sample_data,"%03d",i2s_temp_buff[i-1]);
    udp_buffer_string[3*i] = sample_data[0];
    udp_buffer_string[3*i + 1] = sample_data[1];
    udp_buffer_string[3*i + 2] = sample_data[2];
}
Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
Udp.printf(udp_buffer_string, PACKET_SIZE*3+3);
Udp.endPacket();
```

Fig. 13. Código para transmitir información desde el microcontrolador, mediante UDP

En cuanto a la recepción de la información que recibe el microcontrolador del servidor, se trata de intentar leer algún paquete que haya llegado y si no está vacío, asignarlo directamente a un buffer de escritura y entregárselo a la función I2S (que ya estaba previamente configurado para este propósito) para que lo reproduzca en el DAC mediante la función *i2s\_write()*. El proceso se muestra en la Fig. 14.

```
// RECEIVE AND PLAY STAGE
int packetSize = Udp.parsePacket();
if (packetSize) {
    // receive incoming UDP packets
    int len = Udp.read(incomingPacket, PACKET_SIZE);
    for (int i=0; i<len; i++) {
        i2s_write_buff[i] = incomingPacket[i];
    }
    //send data
    i2s_write(I2S_NUM_0, i2s_write_buff, i2s_read_len, &bytes_read, portMAX_DELAY);
}
```

Fig. 14. Código para recibir información desde el servidor, mediante UDP, y reproducirlo

El DAC está conectado al PAM8403, el cual amplifica la señal y permite que además de que el audio se escuche en los audífonos, el volumen pueda ser modulado por el usuario.

### Aplicación Móvil vinculada por Bluetooth

Se implementó una aplicación móvil para controlar los canales a los que puede acceder el dispositivo. La aplicación considera un total de 255 canales de voz a los cuales se tiene acceso, una vez conectado a uno de los microcontroladores a través de Bluetooth. La interfaz observada a través de la aplicación se presenta en la Fig. 15.

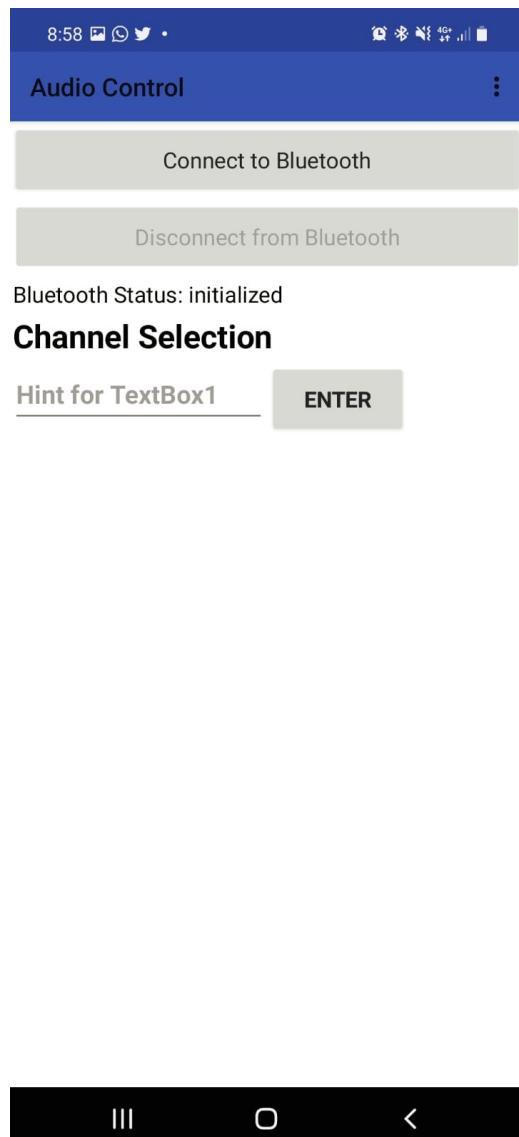


Fig. 15 Aplicación Bluetooth

El código en la Fig. 16 fue agregado para el correcto funcionamiento de la aplicación en el microcontrolador, ya que este mismo debe esperar información de ella con cada iteración del ciclo, para poder actuar con rapidez y cambiar la selección de canal que está al principio de cada paquete.

```
if (ESP_BT.available()){
    channel_sel = ESP_BT.read(); //Read what we receive
}
```

Fig. 16. Código que espera información mediante Bluetooth

### Control de carga

Para cargar una batería con el fin de utilizarla constantemente en un equipo como el desarrollado en el presente documento se necesita un control de carga, ya que las baterías

tienen que ser manejadas con cuidado, con el objetivo de evitar accidentes provocados por sobrecarga. Diseñar un circuito para el control de carga por completo es un proyecto enteramente por separado debido a su complejidad, por lo cual se optó por la selección de componentes ya existentes y su adaptación a las necesidades del proyecto.

En el caso de la batería, se escogió una batería de litio-polímero, es una pila recargable bastante útil al ya contar con un sistema de protección por sí misma, además de estar adecuada las necesidades. Se escogió modelo en concreto debido a sus dimensiones, con el objetivo de evitar un elemento muy voluminoso en el reducido espacio con el que se cuenta para ensamblar en las orejeras, adicionalmente, de acuerdo a datos proporcionados en foros de discusión del ESP32, ESP32 Power consumption considerations - ESP32 Forum, el consumo de energía del microcontrolador permitiría perfectamente la utilización constante del equipo por por lo menos dos horas en el peor de los casos, un tiempo bastante razonable para un prototipo. La batería de litio seleccionada se muestra en la Fig. 17 y el módulo en la Fig. 18. Para el control de carga se tomó la decisión de utilizar un circuito integrado TP4056, el cual consta de un chip del mismo nombre diseñado para cargar baterías de 3.7 voltios a voltaje y corriente constantes. El CI, provee protección térmica, limitación de corriente, entre otros atributos.



Fig. 17. Modelo de batería de litio 602035 B1

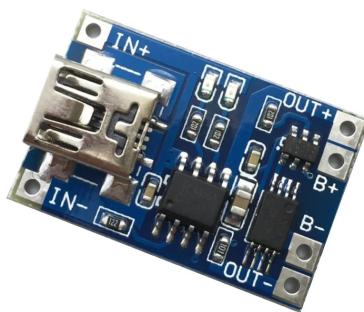


Fig. 18. CI TP4056

El chip cuenta con 8 puertos explicados en la Tabla 2.

Tabla 2. Puertos del CI TP4056

Pin	Función	Descripción
TEMP	Sensor de temperatura	Si el voltaje de entrada de este pin se encuentra por debajo del 45% o por encima del 80% del voltaje de alimentación por más de 0.15 segundos se interpreta como anomalía térmica y se detiene la carga, se puede deshabilitar esta función conectando a tierra.
PROG	Configuración de carga a corriente constante y pin de monitoreo de corriente	La corriente de carga se configura conectando una resistencia entre este PIN y tierra. Así mismo el voltaje en este pin puede ser usado para medir la corriente de carga.
GND	Tierra	Puerto de tierra
VCC	Fuente de voltaje positivo	Alimentación para el circuito
BAT	Conexión para el pin de la batería	Conexión positiva conectada a la batería
STDBY	Estado de la salida de Drain abierto	Cuando se está terminado de cargar se desactiva, de lo contrario está en modo de alta impedancia
CHRG	Estado de la salida de Drain abierto	Pin interno desactivado cuando se está cargando la batería, de lo contrario se encuentra en estado de alta impedancia
CE	Habilitación del chip	Enviando señal a este pin el chip funciona de forma normal, de lo contrario, se desactiva

El circuito integrado comercial, el cual lleva el mismo nombre, ya cuenta con los componentes necesarios para su control de manera integrada, un diagrama de dichos componentes extras se encuentra en la Fig. 19.

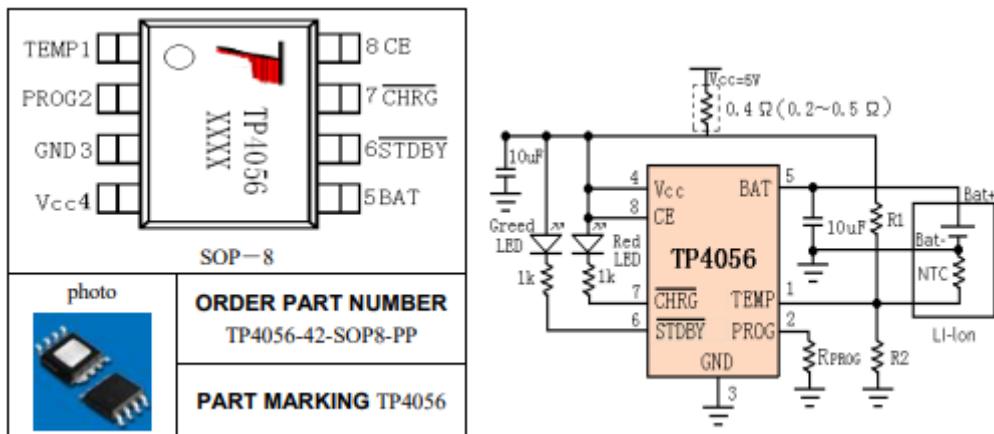


Fig. 19. Diagrama de componentes del CI TP4056

Los pines de Vcc y GND se conectan directamente a la alimentación, la cual es dada por un puerto micro usb embebido, adicionalmente se conecta al pin CE, con la finalidad de que funcione de manera correcta cuando esté alimentado y sea deshabilitado cuando no lo esté. Adicionalmente se conecta un capacitor para amortiguar los picos de voltaje que puedan presentarse. Desafortunadamente no existe actualmente un modelo del TP4056 para simularlo, sin embargo, existe un componente aproximado, del LTC4056, con la diferencia de que este no cuenta con pines para la bandera de STDBY, ni con el pin de protección térmica. Por lo visto en la simulación el circuito funciona tal cual se describe en la hoja de datos. En la Fig. 20 se muestra el diagrama del circuito y en la Fig. 21 se presenta la simulación

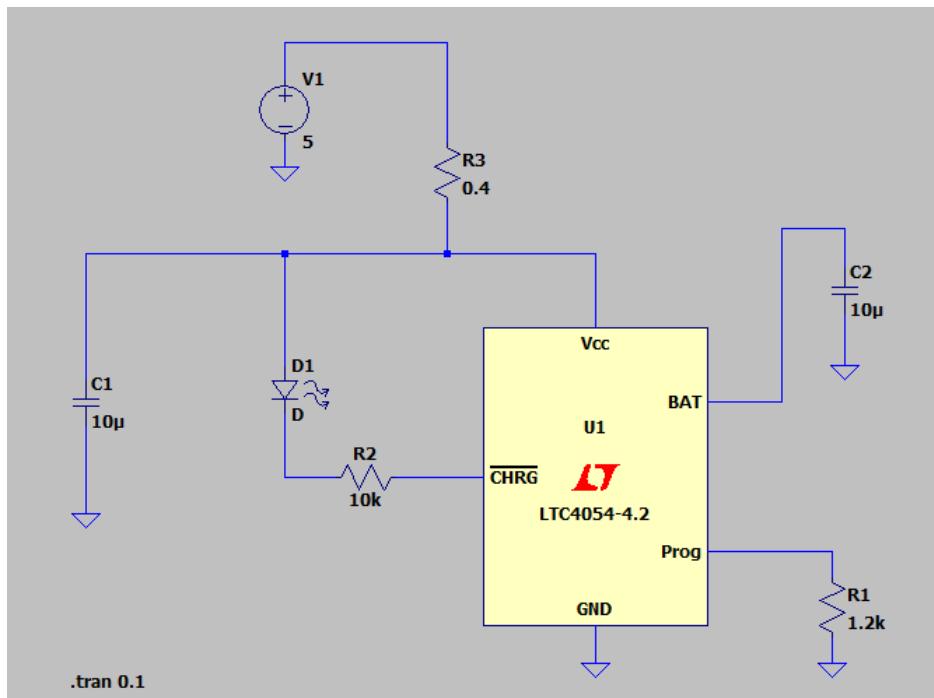


Fig. 20. Diagrama aproximado del CI TP4056

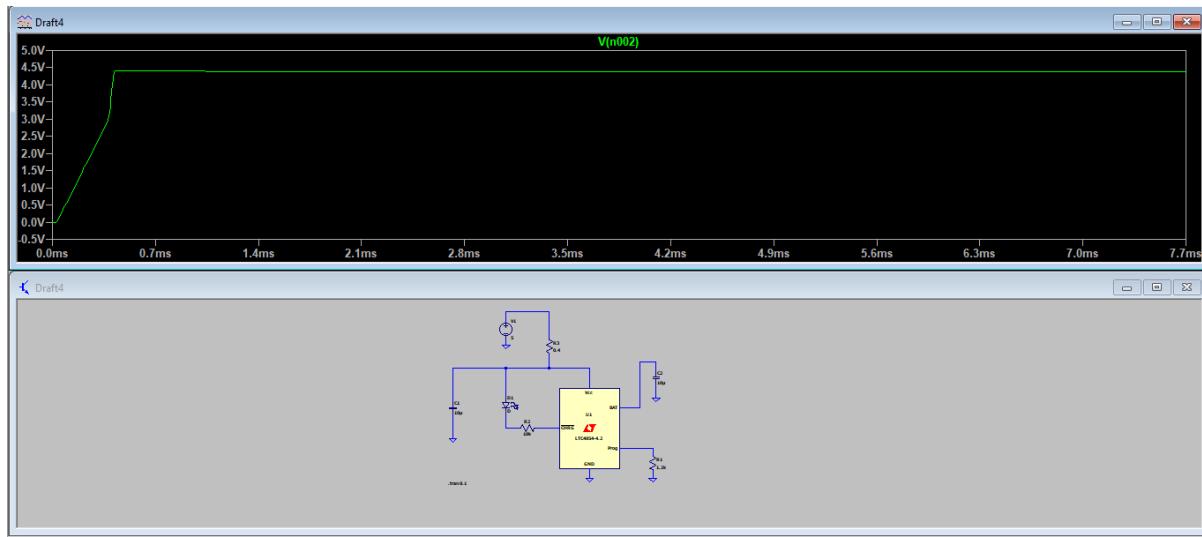


Fig. 21. Simulación del control de carga

El circuito se conecta de la manera en la que se ilustra en la Fig. 22; adicionalmente las mediciones que se hicieron al sistema mientras estaba conectado a alimentación así como solamente la batería. En la Fig. 23 se prueba que el voltaje generado en práctica es suficiente (mayor a 3.3 V).

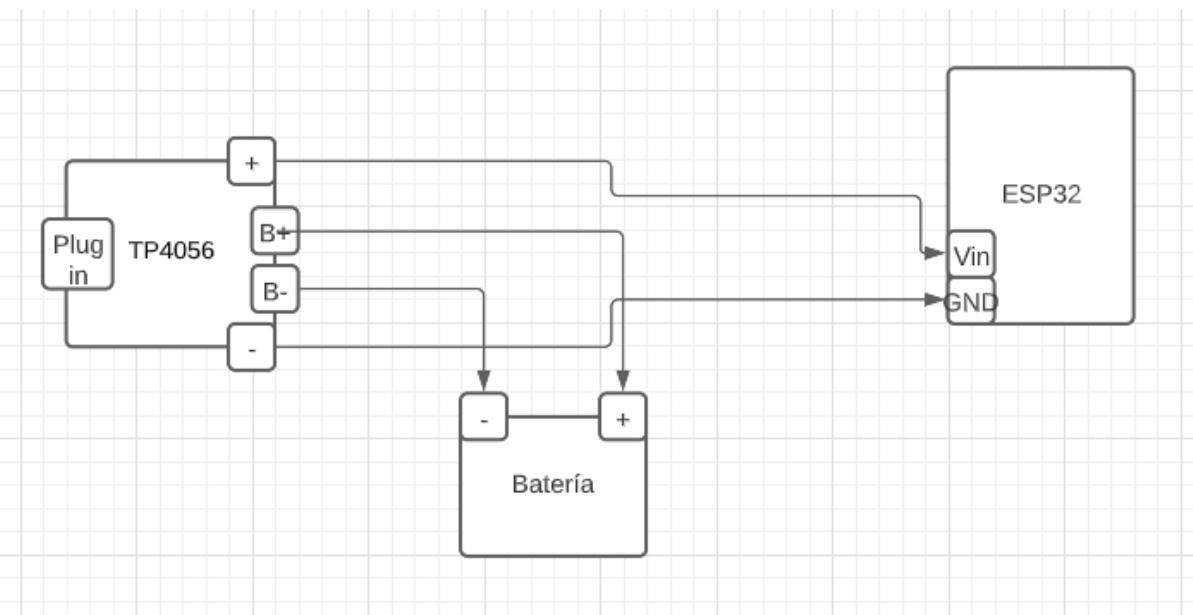


Fig. 22. Diagrama de conexiones para el control de carga

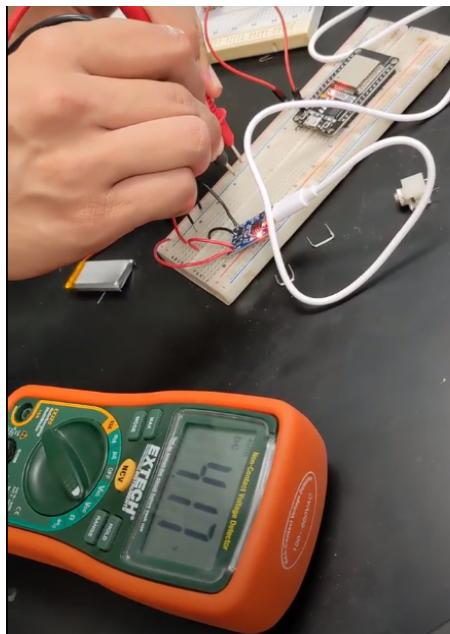


Fig. 23. Voltaje que entrega el control de carga

Una vez conectado se comprobó que el microcontrolador ESP32 era capaz de funcionar alimentándose de la batería, sin embargo, era complicado para una sola lipo de alimentar a todo el sistema una vez añadiendo componentes externos como el amplificador.

### Prototipado

El prototipo final se generó utilizando dos orejeras industriales a las cuales se les adaptó un par de audífonos a cada una (para la salida del audio), un microcontrolador ESP32 (encargado de la recepción, filtro, envío, recepción y salida del audio), un amplificador PAM8403 conectado a la salida del audio del microcontrolador y a los audífonos antes mencionados, y un micrófono MAX4466 conectado al microcontrolador. Además de esto se incluyó un módulo de control de carga y una batería en la parte superior. Las Fig. 24 y 25 muestra el resultado final de este prototipo.



Fig. 24. Primer prototipo final



Fig. 25. Segundo prototipo final

## Análisis de resultados

El resultado final cumplió con la gran mayoría de los requerimientos planteados. El prototipo logró recibir datos de audio a través de un micrófono, ser captados por el ADC y

procesados a través del DMA integrado, además de aplicar un filtrado a la señal para captar únicamente la voz. Esta señal filtrada se manda a través de WiFi utilizando un servidor local como intermediario, el cual envía los datos a un segundo ESP32 quien recibe los datos de audio y utilizando el DMA integrado los manda al DAC. Este proceso es bidireccional, por lo que ambos ESP32 mandan y reciben audio al mismo tiempo. Además de eso se implementó una app controlada por el celular la cual se encarga de cambiar canales de voz al conectarse a alguno de los microcontroladores a través de Bluetooth, se cuenta con un total de 255 canales que podrían ser incrementados dependiendo del uso y las necesidades del prototipo.

Se realizaron las modificaciones necesarias para incluir el control de carga y la batería en el prototipo final, mas no se logró que el código funcionara correctamente al utilizar como fuente de alimentación la batería. No se pudo resolver este fallo en el funcionamiento, se consideró que el voltaje de alimentación era demasiado alto, de 4.2 volts, por lo que se implementó un regulador para llevarlo a 3.3 volts, pero a pesar de este cambio el sistema no funcionó correctamente, por lo que se tuvo que mantener un cable conectado directamente al ESP32 como fuente de alimentación. Es también posible que el voltaje generado no era suficiente para todo el sistema, ya incluyendo el amplificador.

Una limitante que se tuvo en el proyecto fue la frecuencia de muestreo a la cual se obtienen los datos. Se utilizó una frecuencia de 4000 HZ la cual es baja para una aplicación de audio, considerando que una frecuencia de alta calidad serían alrededor de 44100 HZ que es lo común en la industria. La razón por la cual estuvimos limitados a esta frecuencia fue la velocidad de recepción de datos por parte del ESP32, el cual mostraba un ligero retraso al momento de recibir datos por el protocolo UDP, al incrementar la frecuencia el microcontrolador no podía mantenerse recibiendo y transmitiendo datos al mismo tiempo por lo que algunos paquetes se perdían y la señal perdía calidad. Para resolver este problema se podría utilizar una tarjeta más poderosa como lo es una raspberry pi pico la cual funciona a mayores frecuencias y está mejor adaptada para aplicaciones de audio. No se utilizó esta tarjeta ya que incrementan los costos del proyecto en al menos 500 pesos, además de enfrentarnos con este problema en las etapas finales del proyecto lo cual hubiera requerido rehacer la mayoría de los avances en poco tiempo con esta nueva tarjeta. Al juntar la frecuencia de muestreo baja con las conexiones y dispositivos que no eran los más óptimos para aplicaciones de audio de alta calidad, se obtuvo un ligero ruido electrónico a la salida de los audífonos, a pesar de esto el audio que se envía a los audífonos es entendible y cumple con la funcionalidad del dispositivo.

## Conclusiones

Este proyecto permitió entender y estructurar de mejor forma el diseño y desarrollo de dispositivos electrónicos, desde la identificación del problema, el análisis de soluciones hasta el diseño y prototipado del dispositivo. En este caso fue de gran utilidad el uso de microcontroladores para realizar el prototipo, al utilizar el ESP32 fue de gran utilidad contar con las herramientas que este microcontrolador proporciona, por ejemplo el módulo WIFI y Bluetooth, los cuales se utilizaron para configurar y transmitir el audio. A su vez se utilizó el

convertidor DAC para controlar las bocinas y en las primeras iteraciones del proyecto utilizamos el ADC para leer el micrófono. Sin duda tener todas estas características en el mismo microcontrolador agilizó en gran medida el diseño e implementación del sistema. A medida que se avanzaba en el proyecto, se dio cuenta que el poder de procesamiento del ESP32 no sería suficiente para lo que se quería hacer. Aun así el contar con el ESP32 se pudo desarrollar todo el software necesario para establecer la comunicación, esto utilizando un microcontrolador que en general se vende alrededor de los 10 dólares (200 pesos mexicanos), en caso de volver a generar un prototipo más robusto que soporte los 44100 HZ, se podría fácilmente cambiar de microcontrolador a un Raspberry Pi Pico o a un Teensy, manteniendo la mayoría del código e infraestructura previamente desarrollado.

Estas herramientas y módulos que son fácilmente configurables son ideales para el prototipado y hacer las pruebas de concepto. Si bien el producto es funcional, este aún necesita de mayor desarrollo para poder convertirlo en un producto comercial, el siguiente paso si se quisiera seguir desarrollando sería el de diseñar cada parte de forma detallada: por ejemplo, se utilizó un módulo para el micrófono que vale 20 pesos, se podría optar por diseñar el micrófono por completo o buscar proveedores que vendan el mismo módulo pero de mejor calidad. A su vez, se pueden integrar filtros analógicos para disminuir el ruido y la interferencia eléctrica del propio circuito. También, en pasos finales, se podría optar incluso por diseñar un IC que contenga el procesador y únicamente los dispositivos específicos que requiera el producto.

Si se quisiera comercializar el producto aún faltaría invertir mucho dinero y tiempo en el diseño, pero lo importante es que todos los productos inician con algún prototipo funcional al igual que este, a medida que se elabora el prototipo surgen problemas que no se habían contemplado antes al igual que ideas nuevas para solucionar problemas específicos. Al finalizar el prototipo se puede probar el concepto y tomar decisiones en cuanto a si vale o no la pena continuar con el desarrollo de un proyecto.

## Recomendaciones

Para obtener un resultado de mayor calidad se recomendaría utilizar una tarjeta que trabaje a mayores frecuencias como lo es una Raspberry Pi Pico, la cual incrementaría la frecuencia de muestreo a la que opera el sistema. Otra mejora que se podría realizar es la captación y salida del audio, ya que se utiliza un ADC y DAC integrados en la tarjeta, los cuales no son de la mejor calidad. Para resolver esto se podría implementar un micrófono con un ADC integrado que envía la información a través de un protocolo serial (como I2S), y de la misma manera utilizar un DAC externo mandando los datos serialmente desde el microcontrolador. Estas modificaciones incrementarían considerablemente la calidad de la señal de audio escuchada.

## Bibliografía

- [1] Anónimo. (s.f.). Workplace safety - noise, de BetterHealth. Sitio web: pollution <https://www.betterhealth.vic.gov.au/health/healthyliving/workplace-safety-noise-pollution>
- [2] Anónimo. (s.f.). Occupational Noise Exposure, de Occupational Safety and Noise Administration. Sitio web: <https://www.osha.gov/noise/health-effects>
- [3] Peña, F. (s.f.) Procesamiento digital de señales. Tecnológico de Monterrey.
- [4] Marín de la Rosa, J.. (s.f). Fundamentos teóricos: Filtros. Junio, 2021, de Sindicato Español Universitario. Sitio web:  
<http://bibing.us.es/proyectos/abreproj/11375/fichero/MEMORIA%252FFundamentos+teoricos.pdf>
- [5] Anónimo. (2012). ¿QUÉ ES UN DAC?, de Sound&Pixel Planet Sitio web:  
<http://www.sound-pixel.com/blog/%C2%BFqu%C3%A9-es-un-dac>
- [6] Moes, T (2016). Software Lab. Sitio web: <https://softwarelab.org/es/Bluetooth/>
- [7] NanJing Top Power ASIC Corp. (s.f.) TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8