UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO FIN DE GRADO

Diseño y construcción de un intercomunicador para bebés

Autor: Alonso Alameda Mora

Tutor: Raquel Cedazo León

Departamento de ingeniería
eléctrica, electrónica,
automática y física aplicada.

Madrid, febrero, 2022

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO FIN DE GRADO

Diseño y construcción de un intercomunicador para bebés

Autor: Alonso Alameda Mora

Tutor: Raquel Cedazo León

Departamento de ingeniería
eléctrica, electrónica,
automática y física aplicada.

Madrid, febrero, 2022

Título del trabajo:

Autor:

Tutor:

Co-tutor:

EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día ..... de ..... de ……. en ....., en la Escuela Técnica Superior de Ingeniería y Diseño Industrial de la Universidad Politécnica de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO

PRESIDENTE

AGRADECIMIENTOS

De bien nacido es ser agradecido

ÍNDICE

AGRADECIMIENTOS I

ÍNDICE I

RESUMEN III

ABSTRACT IV

GLOSARIO V

LISTA DE ABREVIATURAS V
LISTA DE SÍMBOLOS VI
INDICE DE FIGURAS 7

CAPÍTULO 1. INTRODUCCIÓN 8

1.1. INTRODUCCIÓN AL TEMA DEL TFG 8
1.2. MOTIVACIONES 8
1.3. OBJETIVOS 9
1.4. ESTRUCTURA DEL DOCUMENTO 9

CAPÍTULO 2. ESTADO DEL ARTE 11

2.1. ORIGEN 11

CAPÍTULO 3. FUNDAMENTOS TEÓRICOS 13

3.1. ¿QUÉ ES UN INTERCOMUNICADOR? 13
3.2. EL SONIDO 14
3.3. WIFI 14
3.4. I2S 14
3.5. USER DATAGRAM PROTOCOL (UDP) 14

CAPÍTULO 4. DISEÑO HARDWARE 15

4.1. DESCRIPCIÓN DE LOS COMPONENTES UTILIZADOS 15
4.1.1. Microcontrolador 16
4.1.2. Micrófono 17

4.1.3. Altavoz 18
4.1.4. Etapa amplificadora de señal 18
4.1.5. Fuente de alimentación 20
4.1.6. Diodos leds 21
CAPÍTULO 5. DISEÑO SOFTWARE 23
5.1.1. Introducción 23
5.1.2. Conexión inalámbrica 24
5.1.3. Software placa ESP32 25
5.1.4. Software Aplicación móvil 33
CAPÍTULO 6. RESULTADOS 35
6.1. RESULTADOS 35
6.2. DISCUSIÓN 35
CAPÍTULO 7. CONCLUSIONES Y FUTUROS TRABAJOS 37
7.1. CONCLUSIONES 37
7.2. FUTUROS TRABAJOS 37
BIBLIOGRAFÍA 39
ANEXO A. ANEXO DE EJEMPLO 40
A.1. SECCIÓN 1 DEL ANEXO 40

Resumen

Abstract

Abstract

GLOSARIO

LISTA DE ABREVIATURAS

TFG	Trabajo Fin de Grado
ETSIDI	Escuela Técnica Superior de Ingeniería y Diseño Industrial
PCB	Printed Circuit Board (Placa de circuito impreso)
UDP	User Datagram Protocol
IDE	Entorno de Desarrollo Integrado

LISTA DE SÍMBOLOS

Ω	Ohmio
W	Watio
V	Voltios
Hz	Hercios
dB	Decibelios
mA	Miliamperios

INDICE DE FIGURAS

FIGURA 1: ESP32-DEVKITC V4.....16

FIGURA 2: DIAGRAMA DE BLOQUES INMP441.....17

FIGURA 3: PLACA DE INTERFAZ INMP441 17

FIGURA 4: ALTAVOZ 8 Ω.....18

FIGURA 5: RESPUESTA EN FRECUENCIA DEL ALTAVOZ 18

TABLA 1: CARACTERÍSTICAS.....19

FIGURA 6: GRÁFICA POTENCIA A LA SALIDA VS DISIPACIÓN DEL COMPONENTE....19

FIGURA 7: CIRCUITO AMPLIFICADOR CON GANANCIA 200.....19

FIGURA 8: MÓDULO AMPLIFICADOR DE POTENCIA.....20

.....22

FIGURA 9: DIODO LED ROJO ..... ¡ERROR! MARCADOR NO DEFINIDO.

FIGURA 10: CONEXIÓN LEDS CON ESP32.....22

FIGURA 11: DIAGRAMA DE CONEXIÓN INALÁMBRICA.....24

FIGURA 12: FUNCIÓN RECEIVINGTASK().....26

FIGURA 13: DIAGRAMA DE FLUJO FUNCIÓN SENDINGTASK().....27

FIGURA 14: FUNCIONES QUE CREAN LAS TAREAS.....28

FIGURA 15: FUNCIÓN GETUDP_PACKET() 29

FIGURA 16: FUNCIÓN DATA_SCALE().....30

FIGURA 17:FUNCIÓN WIFIAPEVENT() 30

FIGURA 18: FUNCIÓN OPEN_UDP_PORT() 31

FIGURA 19: FUNCIÓN WIFI_INIT() 31

FIGURA 20: I2S_INIT_MIC().....32

Capítulo 1. Introducción

1.1. Introducción al tema del TFG

1.2. motivaciones

1.3. Objetivos

El objetivo de este trabajo es diseñar y construir un intercomunicador especializado para bebes. Como objetivos específicos tenemos:

- Diseño y construcción del hardware del intercomunicador.
- Desarrollo del software que controla todos los componentes del intercomunicador
- Desarrollo de una aplicación móvil que se comunique con el intercomunicador y envía sonido en ambos sentidos
- Realizar pruebas de funcionamiento del intercomunicador.

1.4. Estructura del documento

Capítulo 2. ESTADO DEL ARTE

2.1. ORIGEN

La tecnología de comunicación entre oficinas es anterior a los walkie-talkies por décadas. Después de que Kellogg patentara el primer sistema de intercomunicación telefónica en 1894, la capacidad de comunicarse con los empleados o transmitir informes oficiales a todo el edificio dio un gran paso adelante. Como sugiere el nombre, los primeros walkie-talkies dependían de la tecnología de telefonía, pero los fabricantes continuaron mejorando el sistema mediante la adopción de nuevas tecnologías.

Tubos de lengua

Las empresas del siglo XIX desarrollaron métodos de comunicación entre varias oficinas de patentes antes de Kellogg. A principios de la década de 1800, los ingenieros construyeron tuberías para transmitir sonido entre dos altavoces en el mismo edificio. Algunos sistemas funcionan a distancias de cientos de metros. La tubería se convirtió en una tecnología comercial común en la segunda mitad del siglo. En la década de 1880, el sistema era lo suficientemente sofisticado como para que los parlantes pudieran conectarse a cualquiera de las 25 oficinas diferentes. Algunos ejecutivos prefieren una

alternativa, un sistema de timbre de habitación en habitación para llamar a sus empleados.

Intercomunicadores basados en teléfonos

Este tipo de intercomunicadores se desarrollaron en la década de 1890. Estos sistemas conectaban dos teléfonos de dos oficinas distintas, pero pronto se empezaron a desarrollar conexiones más complejas. Se diseño el primer sistema radial, el cual los usuarios tenían que llamar a una estación central donde se le redirigía a la oficina la cual se querían poner en contacto.

Intercomunicadores durante el siglo XX

Durante el año 1912, los usuarios ya tenían la opción de elegir entre un teléfono o un altavoz el cual les permitía trabajar con las manos desocupadas. En esta década la mayoría de los intercomunicadores todavía se parecía a lo que convencionalmente llamamos como teléfono. A partir de la década de 1950, los intercomunicadores empezaron a ser utilizados como porteros electrónicos para realizar la comunicación y apertura de los edificios. El gran cambio en los intercomunicadores llegó a raíz de la invención del transistor en el año 1951 por William Shockley[1].

A partir de la invención del transistor se inició la etapa de la digitalización. A partir del 1982, los fabricantes de intercomunicadores empezaron a vender dispositivo que además de audio incorporaban video.

Esta información se ha obtenido de la siguiente referencia [2]

Capítulo 3. Fundamentos teóricos

3.1. ¿Qué es un intercomunicador?

Un intercomunicador es un dispositivo electrónico de comunicación bidireccional[1] que permite la comunicación entre dos interlocutores a través de una conexión establecida entre ambos.

[1] **Bidireccional**: Comunicación que se realiza en dos sentidos, desde el emisor al receptor y viceversa.

3.2. El sonido

3.3. WIFI

3.4. I2S

I2S o también conocido como Inter-IC Sound, es un estándar de bus serie que se utiliza para conectar diferentes dispositivos de audio de manera digital. Este estándar tiene la particularidad de poder habilitar el acceso directo a memoria o más comúnmente llamado DMA.

3.5. USER DATAGRAM Protocol (UDP)

Capítulo 4. Diseño Hardware

4.1. Descripción de los componentes utilizados

En este apartado se va a explicar con profundidad las diferentes alternativas que hay en el mercado para cada parte fundamental del proyecto. En primer lugar, para este proyecto se planteó la idea de diseñar dos dispositivos que se comunicaran entre ellos. Esta idea encarecía bastante el sistema y no habría ningún tipo de mejora con respecto a un intercomunicador para bebes convencional. Por tanto, se decidió diseñar solo un dispositivo que se comunicara con la aplicación móvil.

Las partes fundamentales del dispositivo son las siguientes:

- Microcontrolador
- Micrófono
- Altavoz
- Etapa preamplificadora de señal
- Diodos Led
- Fuente de alimentación

4.1.1. MICROCONTROLADOR

Como elección de microcontrolador se ha elegido el ESP-WROOM-32, se trata de un módulo que dispone de WIFI y Bluetooth, el cuál es ideal para el proyecto ya que toda la comunicación entre la aplicación móvil y el microcontrolador va a ser inalámbrica. Además, este módulo tiene un consumo energético cuando se encuentra en reposo muy bajo (5 µA aproximadamente). Este módulo está fabricado por la empresa *Espressif*, la cual tiene una amplia guía de programación, en la cual se detallan cómo programar integrar las diferentes funciones y librerías que tiene el módulo disponible.

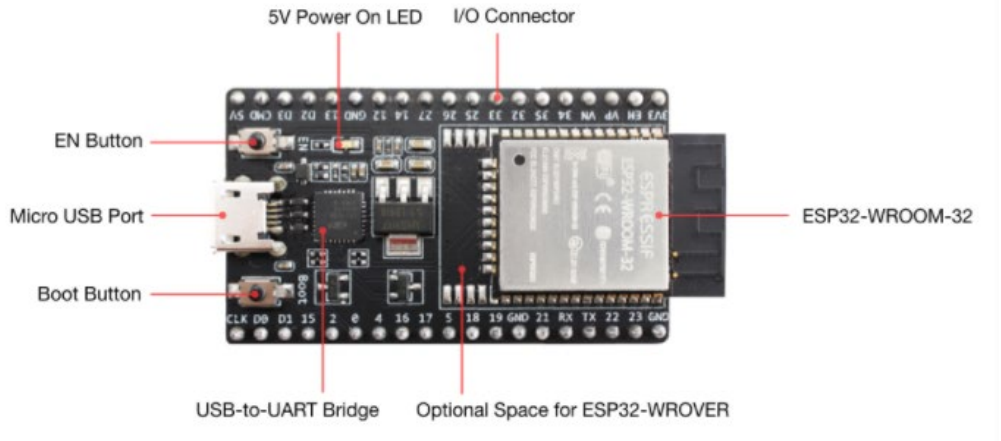


Figura 1: ESP32-DevKitC V4.

Otro valor diferencial en cuanto a otras placas del mercado es su tamaño reducido ya que mide 56x28mm. Esto hace que uno de los requisitos de este proyecto se pueda cumplir ya que es primordial que el intercomunicador tenga un tamaño reducido para que sea lo más portable posible.

Por último, esta placa dispone del estándar I2S, como se explica con más detalle en el apartado 3.4, se va a utilizar para digitalizar la señal de audio recibida del micrófono. Una vez que la señal analógica se encuentre digitalizada se realizará un paquete de formato UDP y se enviará a la aplicación móvil.

EXTENDERSE UN POCO MÁS EN LAS CARACTERISTICAS DE LA PLACA FLASH RAM,ETC

4.1.2. MICRÓFONO

Como micrófono se ha elegido el INMP441. Se trata de un micrófono omnidireccional[2] basado en el protocolo I2S, con una precisión de muestras de hasta 24 bits. Tiene una respuesta en frecuencia desde 60Hz hasta 15kHz, lo que hace que se más que suficiente para integrarlo en el intercomunicador. Además, este micrófono se alimenta a una tensión de 3.3V, que se pueden obtener del regulador DC/DC de 5V a 3.3V que tiene integrado el módulo de ESP32.

Este micrófono viene integrado sobre una placa de interfaz la cual es fácil de poder soldarse posteriormente sobre una PCB.

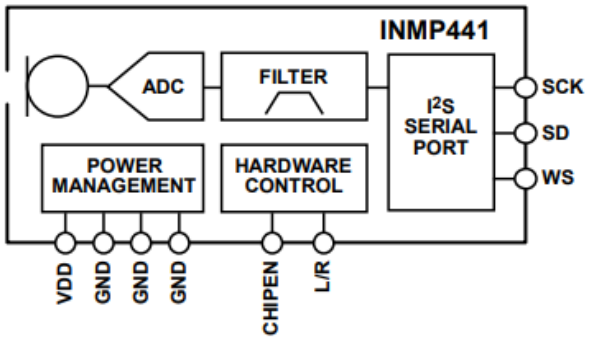


Figura 2: Diagrama de bloques INMP441

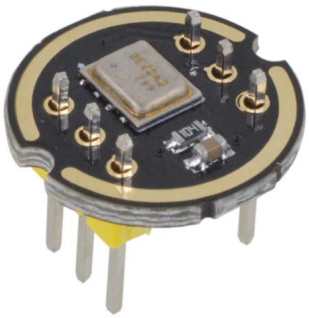


Figura 3: Placa de interfaz INMP441

[2] **Micrófonos omnidireccionales:** son aquellos los cuales tienen una sensibilidad constante, esto significa que capta los sonidos en todas las direcciones. *Fuente: Wikipedia.org*

4.1.3. Altavoz

El altavoz es el encargado de transformar la señal eléctrica que transmite la placa ESP32 en sonido. Para ello se ha elegido un altavoz de 8 Ω, el cual tiene unas dimensiones de 28 mm de diámetro y tiene una potencia nominal de 0.25W. Además, como se puede observar en la Figura 5, la respuesta del altavoz se va a encontrar entre 80-90dB ya que la canción que se va a reproducir no tiene un espectro con altas frecuencias, lo que hace que el altavoz no llegue a saturarse y por tanto reproduzca un sonido con mala calidad.

Figura 4: Altavoz 8 Ω

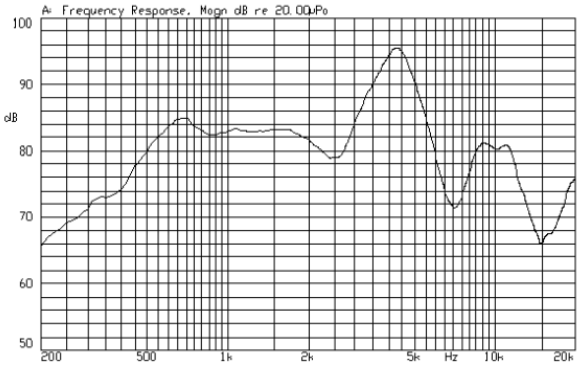


Figura 5: Respuesta en frecuencia del Altavoz

4.1.4. Etapa amplificadora de señal

Esta etapa es necesaria de añadir entre el altavoz y el módulo ESP32 ya que este módulo es capaz de suministrar una señal de salida de 15 mA, lo cual es insuficiente para que en el altavoz se escuche el sonido con claridad. Si se conecta directamente el altavoz al módulo consumiría una potencia máxima que se muestra en la siguiente ecuación:

$$P_{max} = V * I_{max} = 3.3V * 15mA = 49.5mW$$

Esto hace que la potencia máxima consumida sea del 19.8% del valor total del altavoz. Por tanto, hay que añadirle la etapa amplificadora de señal para que ese porcentaje se aproxime al 100% del valor de potencia.

Como amplificador se ha elegido el LM386, el cual es un amplificador operacional diseñado explícitamente para trabajar en labores de sonido. Este amplificador tiene las siguientes características:

Parámetro	Valor
Tensión de alimentación	4-12V
Ganancia en la tensión de salida	26-46dB
Ancho de banda	300kHz
Distorsión armónica total	0.2%

Tabla 1: Características

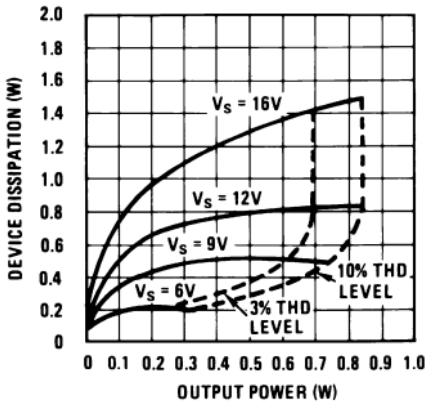


Figura 6: Gráfica potencia a la salida vs disipación del componente

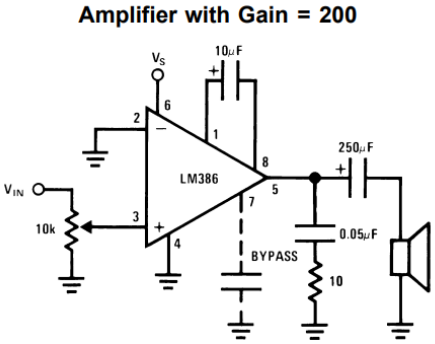


Figura 7: Circuito amplificador con ganancia 200

Como se muestra en la Figura 6, la curva inferior va a ser la que represente la potencia en la salida del amplificador, ya que el sistema se va a encontrar alimentado por 6V. Con esto se observa que la potencia máxima a la salida va a ser de 0.3W lo cual es aproximadamente la misma potencia que soporta el altavoz. Además, si no se superan los 0.3W el THD es del 0.2%, lo cual hace que la señal no se distorsione al aumentar la tensión a la salida del módulo.

Para hacer el diseño lo más simplificado posible, se ha decidido comprar este amplificador operacional pero ya montado y soldado sobre un módulo. Además, ya vienen incluido todos los componentes que aparecen en la Figura 7. Este diseño dispone de un amplificador LM386 con una etapa de filtrado a la salida compuesta por un condensador y una resistencia. También dispone de un potenciómetro a la entrada del amplificador para regular la ganancia total del circuito y diferentes conectores para poder incluirlo dentro del prototipo.

Figura 8: Módulo amplificador de potencia

4.1.5. Fuente de alimentación

Para alimentar todo el prototipo se ha decidido utilizar un conjunto de pilas de 1.5V. Este conjunto está formado por 4 pilas de tipo AAA, las cuales en su conjunto son capaces de suministrar al dispositivo 6V ya que todas se encuentran conectadas en serie entre sí. Se ha decidió elegir el conjunto de pilas ya que la mayoría de las baterías del mercado de dimensiones pequeñas suministraban a la salida una tensión de 3.3V, la cual no era suficiente ya que el módulo amplificador trabaja a una tensión mínima de 6V. Teniendo en cuenta que el conjunto de 4 pilas tiene una capacidad total de 1200 mAh. Se realiza un cálculo para conocer cada cuanto tiempo será necesario cambiar las pilas.

$$Tiempo = \frac{Energía\ suministrada}{Potencia\ consumida\ (Módulo\ ESP32 + Micro)} = \frac{6V * 1200mAh}{80mA * 3.3V + 1.4mA * 3.3V} = 26.86\ horas$$

Por tanto, se concluye que la duración media del conjunto de pilas va a ser de **27 horas** aproximadamente. Este cálculo se ha realizado teniendo como consumo principal el del

módulo ESP32 y el micrófono ya que van a ser los dispositivos que se van a encontrar la mayoría del tiempo funcionando cuando el dispositivo se encuentre encendido.

4.1.6. Diodos leds

Los diodos leds son los encargados de notificar al usuario ante diferentes funciones que está desempeñando la placa. Para ello, se ha decidido utilizar diodos de color rojo, los cuales tienen una buena intensidad lumínica y buena durabilidad. Para el diseño inicial se han incluido 3 Leds los cuales tienen las siguientes funcionalidades:

- LED 1: Este led es el encargado de notificar al usuario de que se ha conectado correctamente a la red Wifi creada por el intercomunicador. Además, si por cualquier motivo el cliente se desconecta de la red se apaga para que el usuario no tenga dudas si la conexión es correcta.

- LED 2: Este es el diodo que se encuentra en la parte central de la hilera de leds. En este caso, notifica al usuario de que se ha iniciado el modo de escucha y por tanto todo lo que escuche el micrófono se va a enviar a la aplicación móvil, donde se reproducirá el sonido ininterrumpidamente.

- LED 3: En este caso, notifica al usuario que se ha iniciado el modo de reproducción de sonido por el altavoz. Una vez que se inicialice este modo aparecerá por el micrófono una canción que recibe el módulo ESP32 de la aplicación móvil.

Por último, todos los leds tienen conectados en serie una resistencia fija de valor 1kΩ, la cuales son las encargadas de limitar la corriente que circula a través de los diodos. El conjunto de diodo y resistencia se encuentran conectados a la placa como se muestra en la Figura 10, la cual es capaz de suministrar una tensión de 3.3V. Estos diodos tienen una caída de tensión entre sus bornas de 1.9V y una tensión máxima de operación definida por el fabricante de 30 mA. Como se muestra en la siguiente ecuación, la resistencia en serie del diodo debe tener un valor superior al obtenido ya que con ello se obtendrá una corriente menor a la máxima dada por el fabricante y no habrá riesgo de que se fundan.

$$R \geq \frac{Tensión\ de\ alimentación - Caida\ de\ tensión}{Corriente\ admisible} = \frac{3.3V - 1.9V}{30 * 10^{-3}A} = 46.6\ \Omega$$

Figura 9: Diodo LED rojo

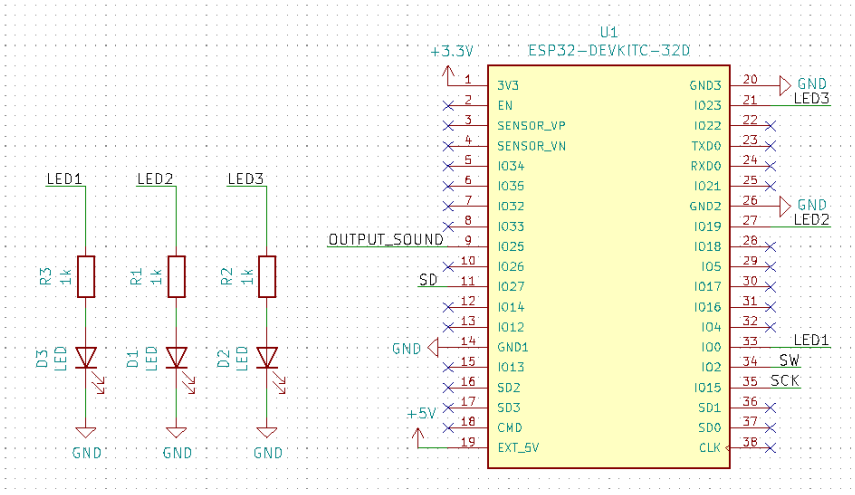


Figura 10: Conexión LEDs con ESP32

Capítulo 5. Diseño Software

5.1.1. Introducción

En el siguiente capitulo se va a explicar con detalle todo el proceso que se ha seguido hasta conseguir que el dispositivo funcione correctamente. En primer lugar, se expone el programa realizado para el módulo ESP32. Este software está basado en el lenguaje de programación C++, el cual se ha estudiado con bastante detalle a lo largo de los estudios de grado.

En segundo lugar, se explicará el funcionamiento y programación de la aplicación móvil diseñada expresamente para este desempeño. Este programa a diferencia del anterior se encuentra desarrollado en Java, el cual me ha costado más ya que no lo había estudiado con anterioridad.

Por último, se va a exponer un programa que se ha realizado para ir comprobando que el sonido se estaba transmitiendo correctamente en ambos sentidos. Este programa se encuentra desarrollado en Python ya que este lenguaje de programación dispone de una librería la cual hacía muy sencillo poder reproducir sonido a través de un socket UDP.

5.1.2. CONEXIÓN INALÁMBRICA

Como se ha explicado con anterioridad, se va a realizar una conexión inalámbrica entre el módulo ESP32 y el dispositivo móvil. Para ello, el ESP32 se va a programar de manera que se comporte como Punto de acceso Wifi. Una vez que la red local se haya creado correctamente, el dispositivo móvil se deberá conectar a ella y entra a la aplicación móvil.

Desde la aplicación móvil se podrá elegir entre los dos modos de comunicación implementados. En el primer modo, el dispositivo móvil abrirá el archivo de la canción que se desea enviar para relajar al bebe, guardado dentro de la aplicación. Estos datos posteriormente se irán enviando por orden para que una vez que el ESP32 reciba dicha información la reproduzca por el altavoz del intercomunicador.

El segundo modo, el ESP32 obtendrá el sonido ambiente desde el micrófono que se encuentra integrado en el intercomunicador, realizará una amplificación de señal digitalmente para que se escuche con un volumen más alto y nítido y se enviará al dispositivo móvil donde se reproducirá por el altavoz del dispositivo. El objetivo es que esta comunicación se realice con el menor retardo posible y con la mejor calidad de sonido.

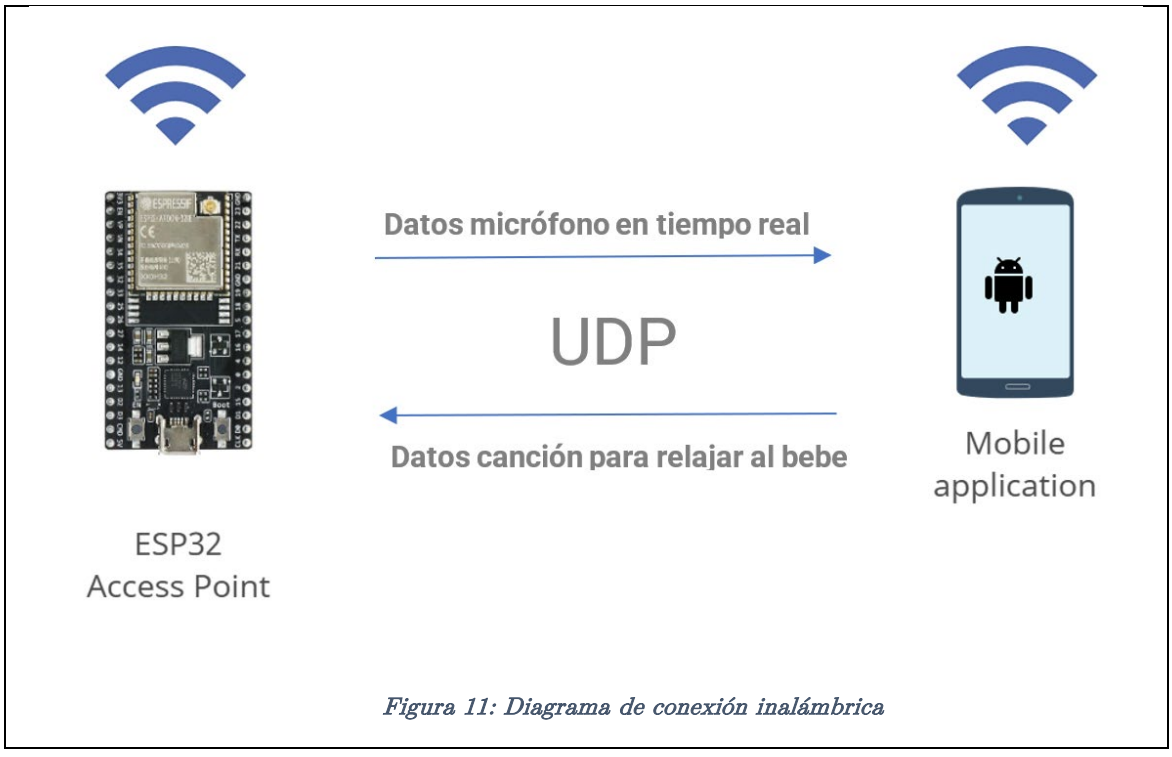


Figura 11: Diagrama de conexión inalámbrica

5.1.3. SOFTWARE PLACA ESP32

Este módulo se ha decidido programarse con el IDE de Arduino ya que tiene ya integradas muchas librerías especializadas para poder desempeñar las tareas que deseo. Además, Arduino tiene una comunidad muy activa en sus foros y si se tiene algún problema con alguna parte del código suele haber alguna solución ya planteada para dicho programa. Otra alternativa para programar el módulo era utilizar Visual Studio Code con la extensión de PlataformIO instalada. El problema con esta extensión era que en muchas ocasiones me aparecían fallos al programar código que hacía que el módulo estuviera constantemente reiniciándose. Este fallo creo que era porque al añadir varias clases al proyecto global hacía que superara la memoria Flash del módulo, haciendo que me imprimiera un fallo del Kernel de la placa y reiniciara constantemente el módulo.

Para realizar el programa completo del módulo se ha utilizado la guía de programación que tiene la empresa Espressif. Se trata de una guía muy completa donde se explican todos los parámetros primordiales para poder programar su módulo. Entre ellos he utilizado con mayor frecuencia el de Bluetooth, Wifi o I2S.

A continuación, se va a explicar con detalle todas las funciones implementadas en el código:

- **ReceivingTask():** Esta tarea se encarga en recibir continuamente datos desde la aplicación móvil y reproducirlos por el altavoz del dispositivo. Para ello se crea un buffer de 1024 bytes, el cual va a ser el encargado en almacenar la información. Se utiliza el comando "read" de la librería UDP.h. Este comando espera hasta que llegue un dato de 1024 bytes y lo almacena en el buffer llamado "buff_reciv". Una vez que se ha detectado que ha llegado un dato, se realiza un bucle en el cual se escribe en el conversor digital analógico que tiene dicha placa. Para ello, se utiliza el comando "dacWrite", el cual hay que especificarle el pin de la placa por donde quieres que salga la información (Pin 25) y la muestra que quieres hacer la conversión ("buff_reciv[i]"). Por último, se realiza una espera por cada muestra de 56 µs. Este valor se obtiene porque la canción enviada tiene una frecuencia de muestras de 16kHz. Por tanto, aplicando la siguiente formula se obtiene el valor de retardo:

$$T = \frac{1}{f} = \frac{1}{16000} = 62.5\mu s$$

A este retardo obtenido hay que restarle 7µs aproximadamente que es lo que tarda la placa en realizar la conversión del dato digital en analógico, con lo que finalmente se obtiene el valor de 56 µs de retardo entre cada muestra.

```
void ReceivingTask(void* arg){
  char buff_reciv[1024];
  size_t bytes_write=0;
  while(true){
      bytes_write=Udp.read(buff_reciv,sizeof(buff_reciv));
      if(bytes_write){
        for(int i=0;i<bytes_write;i++){
          dacWrite(25,buff_reciv[i]);
          delayMicroseconds(56);
        }
      }
  }
}
```

Figura 12: Función ReceivingTask()

- **SendingTask():** Esta tarea es la encargada de enviar los datos obtenidos del micrófono a la aplicación móvil a través de un socket UDP.

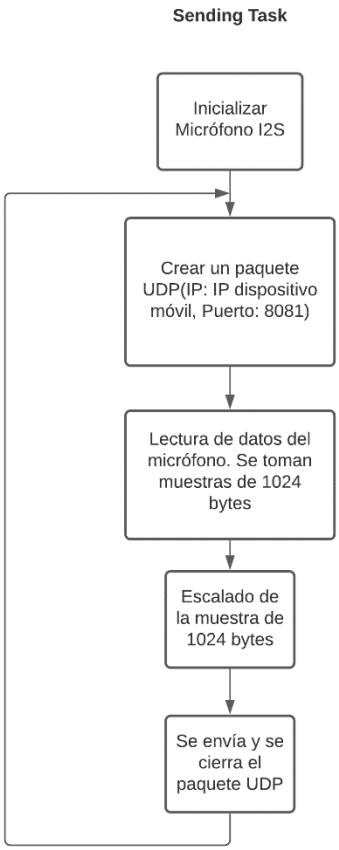


Figura 13: Diagrama de flujo función SendingTask()

```
void startReceiving(){
  Serial.println("INICIO DE RECEPCION...");
  xTaskCreate(ReceivingTask,"ReceivingTask",1024*2,NULL,1,&taskHandler);
}

void startSending(){
  Serial.println("INICIO DE ENVIO...");
  xTaskCreate(SendingTask,"SendingTask",1024*2,NULL,1,&taskHandler);
}

void finish(){
  Serial.println("GRABACION FINALIZADA");
  xTaskCreate(finishTask,"finishTask",configMINIMAL_STACK_SIZE,NULL,1,NULL);
}
```

Figura 14: Funciones que crean las tareas

```
void GetUDP_Packet()
{
   int packetSize = Udp.parsePacket();
   char packetBuffer[1024];
   if ((packetSize>0)&&(packetSize<512))
   {
      Udp.read(packetBuffer, 1024);
      Serial.println();
      Serial.print("Received packet of size ");
      Serial.print(packetSize);
      Serial.print(" from ");
      Serial.print(Udp.remoteIP());
      Serial.print(":");
      Serial.println(Udp.remotePort());
      int dataLen=(size_t)packetSize;
      char message[dataLen+1];
      strncpy(message,(const char *)packetBuffer,dataLen);
      message[dataLen]=0;
      String textString=message;
      Serial.printf("Message received: %s\n",textString);
      if(textString.equals("ENVIA")){
        digitalWrite(ledPin,HIGH);
        Serial.println("Sending...");
        startSending(); //Función que inicia la grabación

      }
      else if(textString.equals("RECIBE")){
        digitalWrite(ledPin2,HIGH);
        Serial.println("Receiving...");
        startReceiving(); //Función que inicia la grabación

      }
      else if(textString.equals("FINALIZA")){
        Serial.println("Finishing tasks...");
        digitalWrite(ledPin2,LOW);
        digitalWrite(ledPin,LOW);
        finish(); //Función que inicia la grabación

      }
    }
   delay(10);
}
```

Figura 15: Función GetUDP_Packet()

```
void 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 * 256 / 2048;
    }
}
```

Figura 16: Función data_scale()

```
void WiFiAPEvent(WiFiEvent_t event, WiFiEventInfo_t info)
{
  if (event == SYSTEM_EVENT_AP_START) {
    Serial.println("AP Started");
  }
  else if (event == SYSTEM_EVENT_AP_STACONNECTED) {
    Serial.println("Client connected");
  }
  else if (event == SYSTEM_EVENT_AP_STADISCONNECTED) {
    Serial.println("Client disconnected");
  }
}
```

Figura 17:Función WiFiAPEvent()

```
void open_udp_port(){
  Serial.println();
  Serial.println("Starting UDP");
   if (Udp.begin(localPort) != 1)
   {
      Serial.println("Connection failed");
      while (true) { delay(1000); }
   }
   Serial.println("UDP successful");
}
```

Figura 18: Función open_udp_port()

```
void wifi_init(){
  Serial.println();
  WiFi.onEvent(WiFiAPEvent, SYSTEM_EVENT_AP_START);
  WiFi.onEvent(WiFiAPEvent, SYSTEM_EVENT_AP_STACONNECTED);
  WiFi.onEvent(WiFiAPEvent, SYSTEM_EVENT_AP_STADISCONNECTED);
  WiFi.mode(WIFI_AP);
  while(!WiFi.softAP(ssid,NULL)){
    Serial.print(".");
    delay(100);
  }
  Serial.print("Nombre: ");
  Serial.print(ssid);
  Serial.println();
  Serial.print("IP address: ");
  Serial.print(WiFi.softAPIP());

}
```

Figura 19: Función wifi_init()

```
void i2s_init_MIC(){
   i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
    .sample_rate =  I2S_SAMPLE_RATE,
    .bits_per_sample = i2s_bits_per_sample_t(I2S_SAMPLE_BITS),
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
    .communication_format = i2s_comm_format_t(I2S_COMM_FORMAT_I2S|I2S_COMM_FORMAT_I2S_MSB),
    .intr_alloc_flags = 0,
    .dma_buf_count = 2,
    .dma_buf_len = 1024,
    .use_apll = 1,
    .tx_desc_auto_clear = false,
    .fixed_mclk = 0
   };

  i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);

   const i2s_pin_config_t pin_config = {
      .bck_io_num = I2S_SCK,
      .ws_io_num = I2S_WS,
      .data_out_num = -1,
      .data_in_num = I2S_SD
  };

  i2s_set_pin(I2S_PORT, &pin_config);
  Serial.println("I2S MICROFONO CONFIGURADO");
}
```

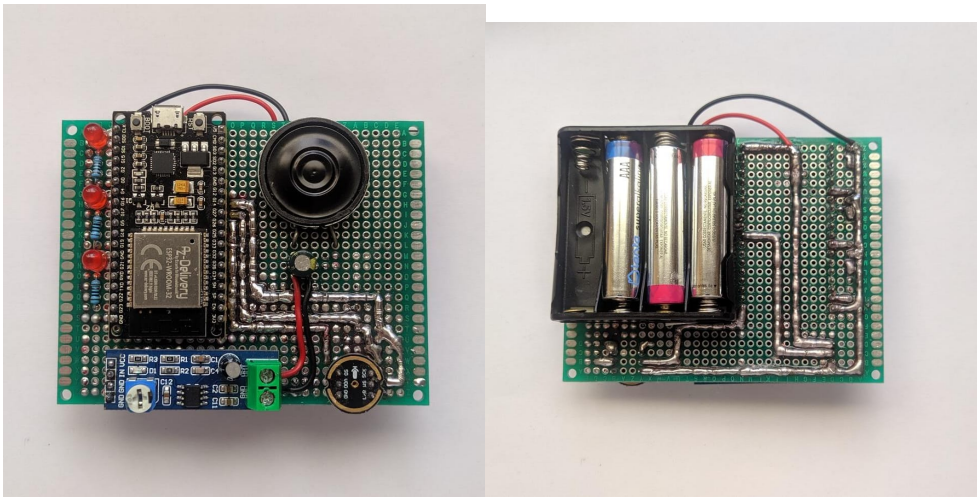
Figura 20: i2s_init_MIC()

5.1.4. Software Aplicación móvil

En este caso se ha utilizado Android Studio para implementar la aplicación desde cero. Se trata de un entorno de programación muy cómodo ya que tienen implementadas muchas librerías de java las cuales son fáciles de importar e implementar en el código.

Capítulo 6. Resultados

6.1. Resultados



6.2. Discusión

Capítulo 7. Conclusiones Y FUTUROS TRABAJOS

7.1. Conclusiones

7.2. FUTUROS TRABAJOS

Bibliografía

[1] “Historia del transistor - Wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Historia_del_transistor (accessed Jan. 19, 2022).

[2] “Historia de los sistemas de intercomunicación / Pretexsa.com.” http://www.pretexsa.com/eM9A20DM.html (accessed Jan. 19, 2022).

Anexo A. Anexo de ejemplo

A.1. Sección 1 del anexo