UNIVERSIDAD SIMÓN BOLÍVAR
VICERRECTORADO ACADÉMICO
DECANATO DE ESTUDIOS TECNOLÓGICOS
COORDINACIÓN DE TECNOLOGÍA ELÉCTRICA Y ELECTRONICA
LABORATORIO DE PROYECTOS

INFORME FINAL DEL PROYECTO

Prototipo de un Amplificador de Audio con Control Inalámbrico Utilizando el Módulo ESP8266

(Tecnología Electrónica)

Autor: Adrian Mayora
Carnet: 16-00131
C.I: 26.180.109

Profesor: Alberto Armengol

Camurí Grande, diciembre de 2022

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En el presente informe se mostrará la investigación y desarrollo del proyecto de un amplificador de audio con control inalámbrico, este podrá ser controlado mediante una aplicación móvil y como medio de comunicación usaremos el protocolo MQTT.

El proyecto se desarrollará por partes, enfocándonos primeramente en el diseño de la fuente de alimentación, siguiendo por etapa de regulación y terminando con el preamplificador con ecualizador y el vúmetro, se simulará y realizará los cálculos necesarios para cada una de estas partes.

Al finalizar el diseño y prueba del circuito a controlar, se mostrará el diseño del circuito necesario para la ejecución del control de encendido y apagado inalámbrico, tanto del amplificador como del vúmetro, y se mostrará diseñó del programa para el ESP8266 en la plataforma de Arduino.

Por último se desarrollará el diseño del circuito para la medición de temperatura, se finalizará el diseño del programa para el ESP8266 y demostrará funcionamiento del prototipo final.

PRESENTACION DE LA COMPAÑÍA Y OBJETIVO DEL PROYECTO

Nuestra empresa MAYORA c.a está conformada por 3 técnicos altamente capacitados y recién graduados de la universidad Simón Bolívar.

Integrantes de nuestro equipo y habilidades

Adrian Mayora: técnico con más experiencia en electrónica y en caza de fallas, y capacitado en diferentes paquetes de software necesarios: simuladores, graficadores o herramientas de planificación y control de proyecto.

Henrry Lannister: aunque también conoce de electrónica, tiene especiales habilidades espaciales y en asuntos más mecánicos, como: Mecánica, Circuitos Impresos, Montajes, Chasis, etc.

Bob Stark: también técnico en electrónica, pero tiene, especiales habilidades en administración, planificación, manejo de costos, diseño o redacción de informes técnicos.

Cada uno ellos podrían, de ser necesario, colaborar con otro en alguna de las tareas que le ha sido asignada.

Recursos de la compañía

La empresa dispone de recursos de un solo computador con su impresora, una mesa de trabajo con los instrumentos básicos, necesarias para realizar el proyecto, tales como protoboard, tester, osciloscopio, fuente dc variable y generador de funciones.

Objetivo del proyecto

Nuestro objetivo es el desarrollo de un prototipo de un amplificador de audio con control inalámbrico, este producto podrá ser controlado mediante una aplicación móvil y como medio de comunicación usaremos el protocolo MQTT “Message Queue Telemetry Transport” y la red telefónica móvil.

Búsqueda del problema y solución

Una situación de uso recurrente con la llegada de las nuevas tecnologías, es la búsqueda de automatizar procesos cotidianos, por ejemplo, estamos sentados en nuestra sala, utilizando nuestro teléfono móvil o acostados en nuestra habitación revisando las redes sociales y queremos escuchar algo de música para relajarnos o escuchar podcast para entretenernos, en tal caso debemos levantarnos para encender nuestra corneta y conectar un estorboso cable desde nuestro teléfono móvil.

O por ejemplo se nos hizo tarde, salimos corriendo al trabajo y no recordamos si apagamos nuestro reproductor, en este caso sería interesante saber si esta encendida en cualquier momento desde nuestro teléfono, y poder apagarlo, otro caso por ejemplo es que tengamos nuestro sistema de amplificación conectado al televisor, estemos acostados y tengamos el amplificador apagado.

Esas situaciones antes mencionadas son muy frecuentes y muy molestas, y se podrían resolver muy fácilmente añadiendo un sistema de control inalámbrico a nuestros sistemas de amplificación.

CARACTERISTICAS QUE DEBERA TENER EL PROTOTIPO DE AMPLIFICADOR DE AUDIO CON CONTROL INALAMBRICO

1. El amplificador será controlado por una aplicación, en este caso utilizaremos aplicación MQTT DASHBOARD disponible en el sistema operativo Android y como medio de comunicación usaremos el protocolo MQTT "Message Queue Telemetry Transport" y la red telefónica móvil.
2. Desde la aplicación podremos controlar el encendido y apagado del amplificador.
3. Desde la aplicación podremos ver si el amplificador se encuentra encendido o apagado.
4. La aplicación mostrará constantemente la temperatura del amplificador.
5. Si por algún motivo existe un sobre calentamiento causado por situaciones externas al amplificador, recibiremos una notificación en la aplicación MQTT DASHBOARD y se encenderá automáticamente un pequeño ventilador para disminuir la temperatura en la etapa de potencia.
6. El amplificador tendrá incorporado un vúmetro, el cual podremos activar y desactivar con la aplicación.
7. Por último, al encenderse el amplificador este ejecutará un audio indicando el encendido del mismo.

MARCO TEÓRICO

Tipos de amplificadores de audio según su potencia

El propósito de este informe no es examinar cada uno de los tipos de amplificadores que hay en el mercado, sino buscar uno que se adapte a nuestras necesidades, por tal motivo en la tabla de abajo vemos un resumen de cada uno de estos, las características, ventajas y desventajas que presentan entres si:

Parámetros	Clase A	Clase B	Clase AB	Clase D	Clase H
Ciclo de operación	360°	180° - 360°	180°	---	---
Región de trabajo del transistor	Activa	Corte	Activa	Activa	Saturación
Consumo	Alto	<Clase AB	< Clase A	>Clase AB	>Clase AB
Fuente recomendada	Lineal	Simétrica	Simétrica	Conmutada	Conmutada
Distorsión	Media	Alta	Baja	Media	Media
Rendimiento	20 – 25%	25 - 60%	50 - 60%	> 90%	> 90%
Pérdidas en calor	Altas	Bajas	Medias	Bajas	Bajas
Peso	Alto	Medio	Medio	<Clase AB	<Clase AB
Volumen	Alto	Medio	Medio	<Clase AB	<Clase AB
Aplicación	Circuitos de audio, equipos domésticos de alta gama	Sistemas telefónicos, transmisores de seguridad portátiles, sistemas de alerta.	Circuitos de audio de alta calidad	Circuitos de audio con menor calidad de sonido	Circuitos de audio con calidad media de sonido

Tabla 1. Tipos de Amplificadores – Tabla realizada por Nicole Salinas

El cuadro anterior realizado por Nicole Salinas Robalino, nos da una idea de los amplificadores de potencia más utilizados hasta la fecha de acuerdo a su clase, se puede observar que cada uno tiene ventajas y desventajas con respecto a los demás, más adelante veremos cuál fue el escogido y el porqué de esta elección.

Etapas de un amplificador

En la figura de abajo podemos ver cada una de las etapas que debe tener un amplificador, la primera etapa llamada sumador tiene el objetivo de rechazar el ruido, la siguiente etapa llamada amplificador de tensión, como su nombre lo indica, es la encargada de aumentar la tensión de entrada, sin aumentar demasiado la corriente, y la última etapa también llamada separadora es la encargada de amplificar la corriente, y determinara la clase del amplificador, la cuales mencionamos anteriormente. (Zorzano, 2004)

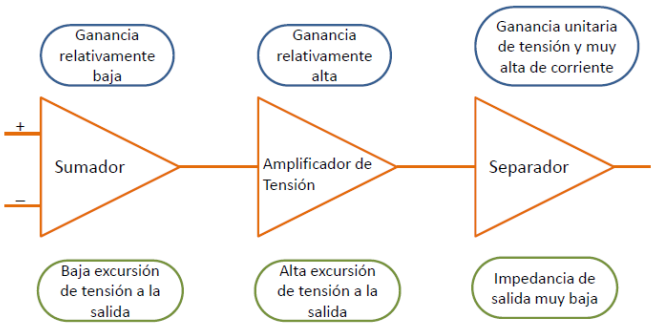

Figura 1. Etapas de un amplificador –disponible docplayer.es, autor desconocido

En la figura de abajo podemos ver cada una de estas etapas visualizadas en el diagrama de un amplificador a modo de ejemplo:

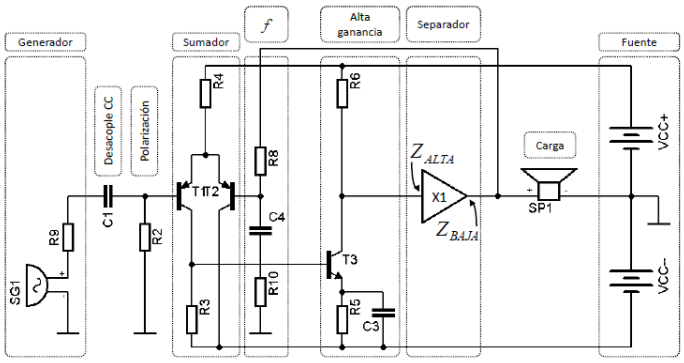

Figura 2. Diagrama ejemplo de etapas de un amplificador – disponible docplayer.es, autor desconocido

Como vemos en la figura de abajo, el circuito anterior se pude resumir en un amplificador operacional, hoy en día existen amplificadores operacionales específicos para audio que garantizan una buena amplificación, como lo es el caso de los amplificadores de la serie TDA de STMicroelectroics. (Zorzano, 2004)

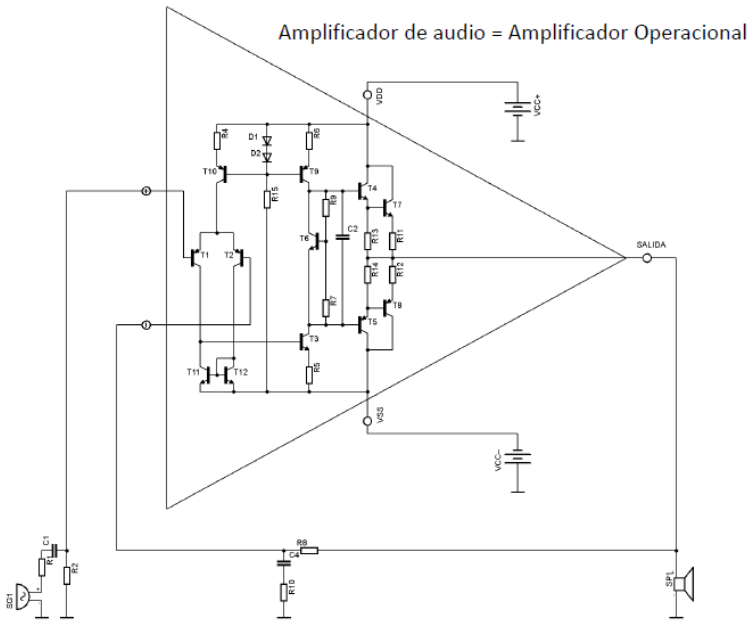

Figura 3. Amplificador Operacional – disponible docplayer.es, autor desconocido

En nuestro caso utilizaremos el amplificador operacional Tl084 en la etapa de pre amplificación como veremos más adelante, este integrado tiene la ventaja de tener una entrada JFET, lo cual nos provee de una buena inmunidad al ruido.

Amplificador a utilizar

Cabe recalcar que al principio se tenía pensado realizar un amplificador de alta potencia, sin embargo, debido a la falta de componentes en el laboratorio, se decidió optar por un amplificador de menos potencia, la idea utilizar circuito integrado TDA2030, el cual se trata de un amplificador operacional de 30W especial para audio, sin embargo, se utilizara el módulo PAM8403, que como veremos se trata de un amplificador muy eficiente en su consumo de energía.

El PAM8403 es un amplificador de audio de clase D de 3 W. que como vimos anteriormente tiene una gran eficiencia, este amplificador ofrece THD+N (distorsión armónica total más ruido) bajo, lo que le permite lograr una reproducción de sonido de alta calidad. La nueva arquitectura sin filtro permite que el dispositivo controle el altavoz directamente, sin necesidad de filtros de salida de paso bajo, lo que ahorra el costo del sistema y el área de PCB. Con la misma cantidad de componentes externos, la eficiencia del PAM8403 es mucho mejor que la de clase AB. Puede extender la duración de la batería, ideal para aplicaciones portátiles. (datasheet.es, s.f.)

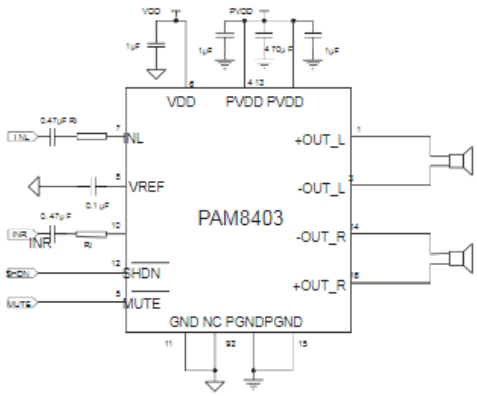

Figura 4. Diagrama del amplificador PAM8403 – obtenido del datasheet del PAM8403 disponible en www.datasheet.es

Abajo se muestran los valores máximos de operación del PAM8403 tomados de su datasheet, los cuales utilizaremos para hacer nuestro proyecto.

Supply Voltage.....6.6V
Input Voltage.....-0.3V to V_{DD}+0.3V
Operation Temperature Range.....-40°C to 85°C
Maximum Junction Temperature.....150°C

Ecualizador

El ecualizador es la parte del circuito encargada de ajustar el volumen en determinados tonos. El tono es la característica del sonido que permite distinguir sonidos tanto graves como agudos, viene determinado directamente por la frecuencia, sin embargo, puede cambiar de acuerdo a la presión. (Salina, 2019)

El ser humano es capaz de escuchar desde los 20 a 20k Hz, para lograr diferenciar los sonidos, el espectro audible se divide en octavas, las cuales son intervalos entre dos sonidos con relación de frecuencias. (Salina, 2019)

OCTAVA	FRECUENCIAS (Hz)
Primera	16 – 32
Segunda	32 – 64
Tercera	64 – 125
Cuarta	125 – 250
Quinta	250 – 500
Sexta	500 – 1000
Séptima	1000 – 2000
Octava	2000 – 4000
Novena	4000 – 8000
Décima	8000 -16000
Decimoprimera	16000 – 32000

Tabla 2. Espectro audible – Tabla realizada por Nicole Salinas

En este caso nos interesa construir un ecualizador que cumpla con los siguientes parámetros del espectro audible:

TONOS	FRECUENCIAS	OCTAVAS
Graves	20 – 256 Hz	4 primeras
Medios	256 Hz – 2 kHz	Quinta, sexta y séptima
Agudos	2 - 20 kHz	Tres últimas

Tabla 3. Rangos de frecuencia del ecualizador a diseñar – Tabla realizada por Nicole Salinas

Modulo ESP8266

El ESP8266 es el nombre de un micro controlador diseñado por Espressif Systems, una compañía china con sede en Shangai. El volumen de producción de estos micro controladores no empezó hasta principios de 2014. El ESP8266 se anuncia a sí mismo como una solución autónoma de redes WiFi que se ofrece como un puente entre los microcontroladores que hasta ahora existían hasta los MCU con WiFi, siendo además capaz de ejecutar aplicaciones independientes. (Granados, 2017)

Si se fuera a usar un ESP8266 salido de fábrica probablemente no se sabría qué hacer. Y es gracias a que los fabricantes los construyen encima de circuitos impresos y placas prefabricadas, que estos quedan listos para nuestro uso. Por eso se da lugar a varias versiones de ESP8266 (Ej: ESP-01, ESP-02…) pero todas con el mismo procesador, lo que las diferencian son el número de pines GPIO expuestos, la cantidad de memoria flash, las dimensiones, la forma de exponer los pines, y otras consideraciones varias relativas a su construcción. Pero desde una perspectiva de programación todas son iguales. (Granados, 2017)

Con el módulo ESP-12E como placa base, el equipo de NodeMCU desarrolló una placa de conexión para su proyecto de firmware de NodeMCU e hizo que el diseño fuera de código abierto. La siguiente imagen muestra el pinout para la placa NodeMCU: (Teis, 2021)

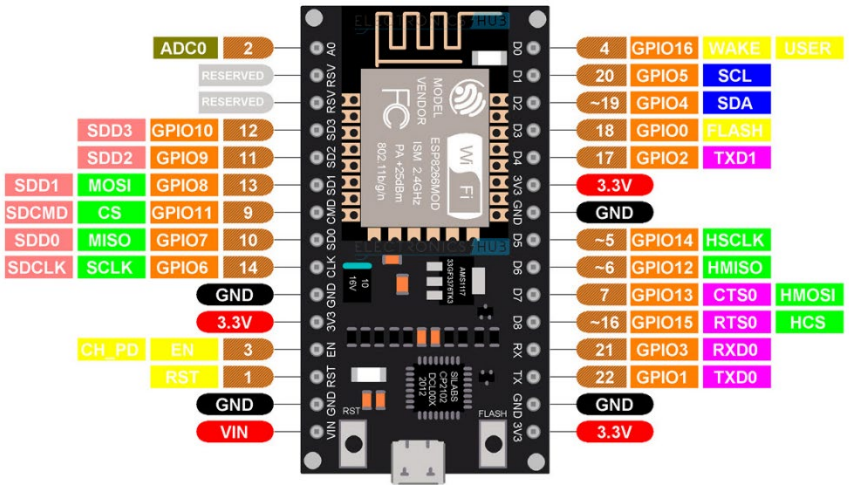

Figura 5. Pines del ESP8266 – Figura tomada de ElectrónicsHub.com, autor Ravi Teis

Especificaciones del ESP8266

Voltaje	3.3 V
Consumo de corriente	10 µA – 170 mA
Memoria Flash	16 MB máx. (512 k normal)
Procesador	Tensilica L106 32 bit
Velocidad del procesador	80 – 160 MHz
Analógico a digital	1 entrada con 10 bit de resolución (1024 valores)
Máximas conexiones simultáneas	5

Tabla 4. Especificaciones del ESP8266– Tabla tomada de "Programando directamente un ESP8266"

Arquitectura del ESP8266:

El ESP8266 tiene una arquitectura de Harvard, con lo cual la CPU puede tanto leer una instrucción como realizar un acceso a la memoria de datos al mismo tiempo, incluso sin una memoria caché. En consecuencia, una arquitectura de computadores Harvard puede ser más rápida para un circuito complejo, debido a que la instrucción obtiene acceso a datos y no compite por una única vía de memoria. (Granados, 2017)

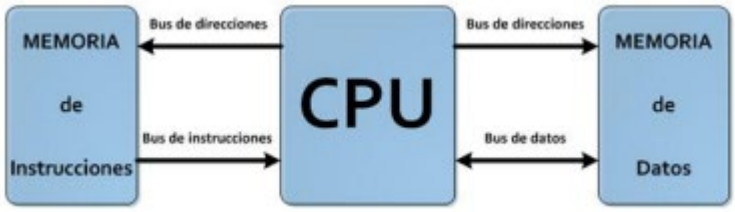

Figura 6. Arquitectura Hardvard – Figura tomada de "Programando directamente un ESP8266" , autor Carles Granados

¿Cómo alimentar NodeMCU?

Hay dos formas de alimentar la placa NodeMCU. Uno es a través del puerto micro-USB y el otro es a través del pin VIN. Tenga en cuenta que ESP8266EX SoC es compatible con solo 3.3V. Por lo tanto, la placa NodeMCU tiene un IC regulador de 3,3 V (AMS1117 - 3,3). Si tiene una potencia de 5 V regulada, puede aplicarla al pin VIN. Hay tres pines de 3,3 V, que están conectados a la salida de 3,3 V del regulador. (Teis, 2021)

Plataformas para programar el ESP8266

Arduino

Mucho antes de que hubiera un ESP8266, había el Arduino. Una aportación de vital importancia a la comunidad de hardware de código abierto y el punto de entrada para la mayoría de los aficionados en el mundo de los circuitos y procesadores construidos en casa. Una de las principales atracciones del Arduino es su baja complejidad, permitiendo a cada uno la capacidad de construir algo rápidamente y fácilmente. El entorno de desarrollo integrado (IDE) para el Arduino siempre ha sido gratuito para descargar desde Internet. Además de proporcionar un editor de lenguaje C más herramientas para compilar e implementar, Arduino IDE proporciona bibliotecas pre-suministradas de rutinas C que "ocultan" complejos detalles de la implementación que de otra manera podrían ser necesarios cuando se programan las placas de Arduino. (Granados, 2017)

Lua NodeMCU

Es un firmware para el ESP8266 basado en el Espressif Non-OS SDK y usa el lenguaje de programación Lua. Lua es un potente lenguaje de scripting disponible en entornos del ESP8266. La aplicación más popular de Lua para el ESP8266 se conoce como el NodeMCU Lua Firmware y está disponible en su repositorio github. (Granados, 2017)

Javascript

El JavaScript es un lenguaje de alto nivel interpretado. Algunas de sus construcciones centrales son mecanografía suelta, orientación a objetos, soporte de funciones lambda, soporte de cierres y, lo que es más importante, se ha convertido en el lenguaje de la web. Si se está escribiendo una aplicación alojada en el navegador, entonces es una certeza que se escribirá en JavaScript. Por un tiempo ahora JavaScript ha estado ganando terreno en código de servidor a través de proyectos como Node.js. Como un lenguaje para ejecutar código de servidor, tiene un conjunto significativo de características para realizar esta capacidad. (Granados, 2017)

ESP8266 Basic

Basic es un idioma amado por millones de personas. Es como comenzó Microsoft y una de las razones para el crecimiento explosivo de las computadoras en los años 80. Basic es un lenguaje simple pero potente que te permite hacer cosas increíbles sin necesidad de un título en ciencias de la computación. Además de proporcionar un editor de lenguaje Basic en la propia página, ESP8266 Basic proporciona bibliotecas pre-suministradas de rutinas que "ocultan" complejos detalles de la implementación que de otra manera podrían ser necesarios. (Granados, 2017)

Plataforma escogida para realizar el programa del proyecto

Según (Granados, 2017) la plataforma escogida para realzar el programa del proyecto fue Arduino debido a las características mencionadas anteriormente:

- Baja complejidad, permitiendo a cada uno la capacidad de construir algo rápidamente y fácilmente.
- El entorno de desarrollo integrado (IDE) para el Arduino siempre ha sido gratuito para descargar desde Internet.
- Además de proporcionar un editor de lenguaje C más herramientas para compilar e implementar.
- Proporciona bibliotecas pre-suministradas de rutinas C que "ocultan" complejos detalles de la implementación que de otra manera podrían ser necesarios cuando se programan las placas de Arduino.

¿Qué es el Internet de las cosas?

El término IoT “Internet de las Cosas” fue empleado por primera vez en 1999 por el pionero británico Kevin Ashton para describir un sistema en el cual los objetos del mundo físico se podían conectar a Internet por medio de sensores. (Rose, 2015)

Ashton acuñó este término para ilustrar el poder de conectar a Internet las etiquetas de RFID “identificación por radiofrecuencia” que se utilizaban en las cadenas de suministro corporativas para contar y realizar un seguimiento de las mercancías sin necesidad de intervención humana. (Rose, 2015)

Hoy en día, el término Internet de las Cosas se ha popularizado para describir escenarios en los que la conectividad a Internet y la capacidad de cómputo se extienden a una variedad de objetos, dispositivos, sensores y artículos de uso diario. (Rose, 2015)

Modelo Publish/Subscribe

Este modelo de comunicación presenta una alternativa al típico modelo cliente/servidor donde el cliente se comunica directamente con el punto final. Por el contra, el modelo publish/subscribe se basa en la comunicación máquina a máquina, en el que un cliente envía un mensaje particular “publicador” a otro cliente, o varios, que reciben el mensaje “suscriptor”. (Muños)

¿Qué es el protocolo MQTT?

MQTT "Message Queue Telemetry Transport" es un protocolo de comunicación de mensajes basado en el método publish/subscribe, situado por encima del protocolo TCP (protocolo de control de transmisión). Es un protocolo ligero y fácil de implementar que requiere pocos recursos a nivel de procesamiento y ancho de banda. (Muños)

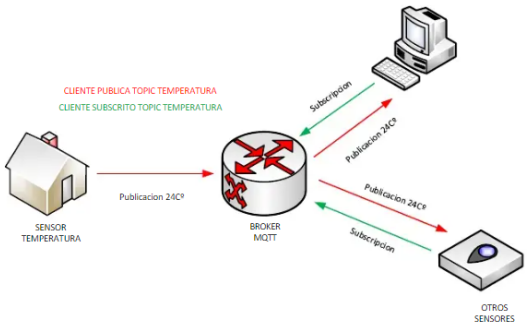

Figura 7. Comunicación utilizando el proco lo MQTT – imagen tomada de "Protocolo MQTT"; autor Juan Muñoz

Al seguir el modelo publish/subscribe los clientes no han de ser conocedores unos de otros por lo que cada cliente solamente ha de conocer la dirección y el puerto del bróker. (Muños)

El bróker MQTT

El broker MQTT es un servicio (software) que implementa el protocolo MQTT y será el encargado de redirigir los mensajes a los diferentes clientes según la subscripción a tópicos. Cuando un cliente publica un mensaje a un tópico el bróker es el encargado de enviar ese mensaje a todos aquellos clientes que estén suscritos. (Muños).

Broker MQTT a utilizar en el proyecto

En este caso se escogió para este proyecto el bróker MyMQttHub, ya que nos ofrece una plataforma fácil de usar para crear proyectos de IoT basados en el protocolo MQTT, Junto con muchas características, además MyQttHub proporciona un plan totalmente gratuito para que puedas desarrollar proyectos, en su página web se pueden observar las opciones que nos ofrece su plan gratuito: https://myqtthub.com/

¿Qué aporta el protocolo MQTT al internet de las cosas?

Según (Muños):

- Es open source, con lo cual es fácilmente integrable al variopinto universo de tecnologías, protocolos y aplicaciones de Internet de las cosas.
- Puesto que se basa en la publicación-subscripción de mensajes, no se necesita saber para quien van los mensajes o de dónde vienen, reduciendo mucho la complejidad de la red.
- Gracias al punto anterior, es un protocolo ligero (cabeceras muy reducidas, comunicación bajo demanda, etc), con lo que se ajusta a redes y dispositivos con pocos recursos y baja velocidad de transmisión, como una típica red de sensores.
- Se basa en comandos muy sencillos y varios modos de gestión de mensajes, y además, no necesita que el contenido del mensaje estén un formato específico.
- Permite unir fácilmente el mundo del Business Intelligence y el BigData con las fuentes de datos, pero también es ideal para la conexión machine-to-machine “M2M”.

Definición de Topic

Un topic se refiere a un tema o asunto concreto. Al final, es un identificador que define el contenido del mensaje o el ámbito de influencia del mensaje (contexto). Con esto se clasifican y discriminan unos mensajes de otros y por extensión, unos nodos de otros. Todo aquel que quiera publicar un mensaje MQTT, es responsable desclasificarlo en un topic concreto y los clientes subscritos a ese topic, recibirán esos mensajes. (Muños)

Los topics en el protocolo MQTT se organizan según una estructura jerárquica en árbol (topic tree), utilizando el carácter "/" para formar un topic de varios niveles. Similar a la ruta a un directorio de carpetas típico de tu ordenador. (Muños)

Ejemplo de topic

Imagina una aplicación de Internet de las Cosas para gestionar un jardín de forma inteligente mediante el protocolo MQTT. El jardín está dividido en varias parcelas, cada parcela con diferentes plantas, árboles y características de riego. El topic raíz podría ser: (Muños)

• Topic más simple = "jardín"

Algunos subniveles podrían ser:

• Jardín/césped

• jardín/frutales/manzanos/temperatura

• jardín/rosales/humedad

Calidades de servicio del protocolo MQTT

Según (Muños) el protocolo MQTT también implementa QoS "calidades de servicio" para poder tener confirmación de la entrega de paquetes. Existen QoS:

- QoS 0, denominada fire and forget, en la que no se realiza ningún tipo de comprobación, el mensaje es mandado solo una vez.

- QoS 1, denominada at laest one, en la que se requiere un reconocimiento, pero que como dice su nombre puede llegar a enviarse más de una vez.

- Qos 2, denominada exacly one, en la que hay un mecanismo de comprobación por la cual solo se entrega exactamente una vez el mensaje sin repeticiones.

En nuestro caso, el bróker escogido solo nos ofrece una calidad de servicio QoS 0 en su versión gratuita, pero no resulta ningún problema para realizar el proyecto.

Librería a utilizar en el proyecto para la comunicación MQTT

Librería PubSubClient

Afortunadamente, integrar MQTT en un procesador como Arduino es muy sencillo gracias a la existencia de varias librerías. La más popular y conocida es la genial librería PubSubClient desarrollada por Nick O'Leary. (Llamas, 2021)

PubSubClient es compatible con una gran variedad de dispositivos e interfaces web. Cuando fue desarrollada, fue principalmente pensada para una combinación de Arduino junto con un shield Ethernet o un shield Wifi. (Llamas, 2021)

No obstante, a estas alturas, resulta más frecuente es emplear PubSubClient en procesadores más avanzados que incorporan WiFi de forma nativa. Como, por ejemplo el ESP8266 y ESP32. (Llamas, 2021)

Características de PubSubClient

Según (Llamas, 2021):

- PubSubClient es un cliente MQTT para microprocesadores y dispositivos IoT.
- Por defecto usa MQTT 3.1.1, aunque puede ser cambiada cambiando la variable MQTT_VERSION en el archivo PubSubClient.h.
- Permite suscribirse a mensajes en QoS 0 o QoS 1, aunque únicamente es posible publicar mensajes en QoS 0.
- El tamaño máximo del mensaje a enviar, incluido la cabecera, es de 256 bytes. Esto es suficiente para la mayoría de proyectos. No obstante, es posible variarlo cambiando la constante MQTT_MAX_PACKET_SIZE en el fichero PubSubClient.h.
- El intervalo de KeepAlive es de 15 segundos por defecto, aunque también es posible cambiarlo mediante la constante MQTT_KEEPALIVE, o llamando a la función estática PubSubClient::setKeepAlive(keepAlive).
- La librería PubSubClient es Open Source, y todo el código está disponible en https://github.com/knolleary/pubsubclient.

DISEÑO DEL DEL PROYECTO

Dividiremos el proyecto en etapas o bloques, de forma que sea más fácil la compresión, análisis y cálculos para realizar nuestro proyecto, en la figura de abajo podemos ver cada una de las etapas que iremos analizando.

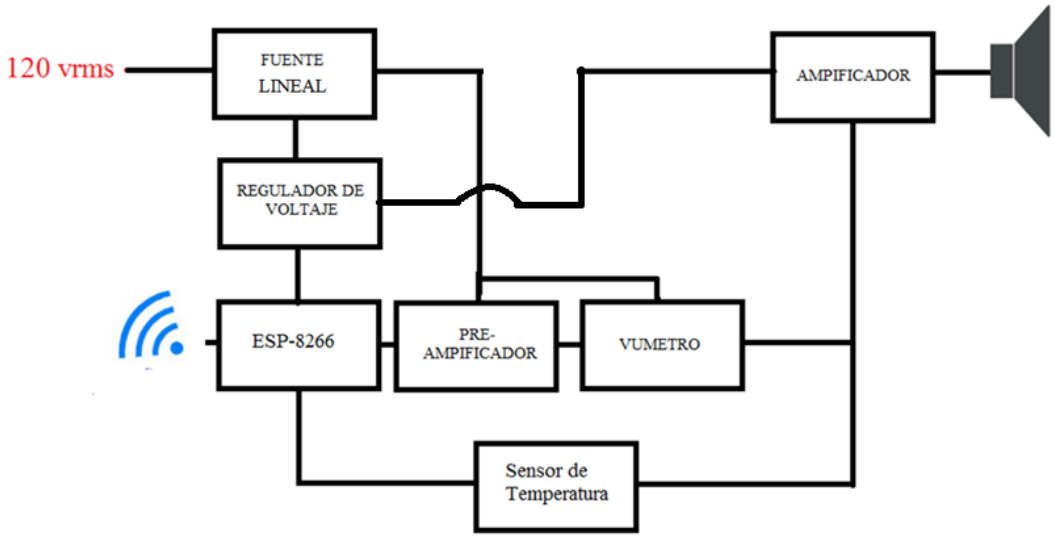

Figura 8. Diagrama simplificado del proyecto – Imagen de autoría propia

En las paginas siguientes comenzaremos el análisis, diseño y simulación de algunas de las etapas correspondientes al hardware del proyecto, de forma que en los siguientes avances podamos enforcarnos meramente a la programación del ESP8266 y su interacción con el hardware.

Fuente de alimentación

En este caso diseñaremos una fuente lineal de 12v dc, la cual consiste en la rectificación de la señal senoidal que viene a nuestros hogares y posterior filtrado para obtener una señal lo más constate posible, tal como vemos en la figura de abajo, se eligió realizar una fuente de 12v debido a que tanto el preamplificador como el vúmetro trabajan a 12v, sin embargo el PAM8403 y el ESP8266 trabajan con máximo 5v, así que utilizaremos un regulador de 12v a 5v para estas 2 últimas etapas, como veremos más adelante.

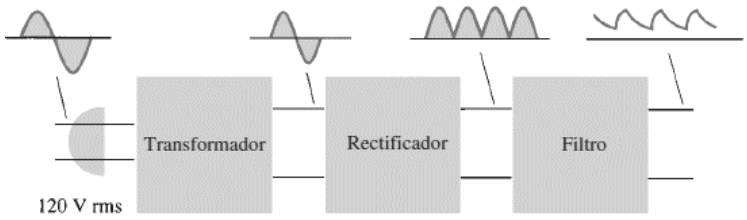

Figura 9. Etapas de una fuente de alimentación lineal – Imagen tomada del libro "Dispositivos Electrónicos" Por T.Floyd

Cálculo del transformador

La señal que viene a nuestros hogares es una onda senoidal, esta como sabemos es una señal de 120 Vrms o 169.71 Vp, sin embargo esta señal antes de ser rectificada pasara por un transformador, en este caso será un trasformador a 9 Vrms de salida, al ser rectificados filtrados estos 9 vrms, tendremos 12v a la salida de la fuente, Generalmente los transformadores deben ser capaces de soportar al menos 2 veces el consumo que tendrá la carga como método de seguridad, en este caso nuestra utilizando el PAM8403 consume apenas 3w por salida, un total de 6w, y el consumo del ESP8266 es apenas de máximo 170mA.3.3v= 0.56W. así que con un transformador de 30w tendremos más que de sobra para soportar cualquier inconveniente.

Para el cálculo del transformador utilizaremos un programa diseñado por el técnico colombiano Jaider Martínez, el cual nos ayuda a obtener todos los valores necesarios para hacer nuestro trasformador.

En el siguiente enlace se puede acceder al artículo donde se encuentra el programa: http://www.videorockola.com/rockolas/software/programa-para-calcular-transformadores/

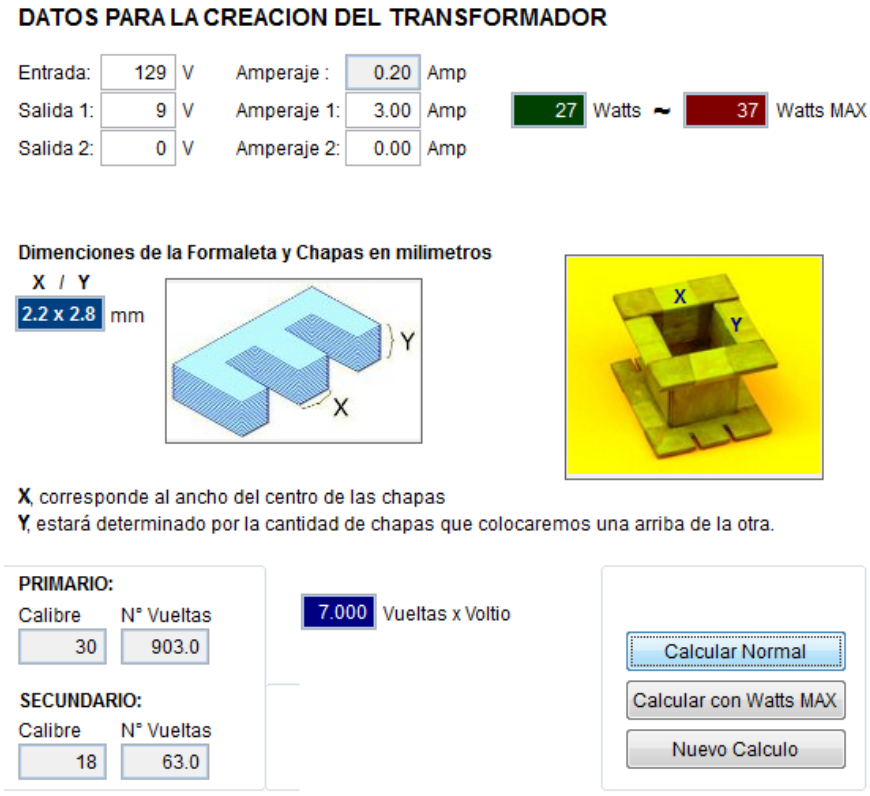

Figura 10. Diseño del transformador – Imagen de autoría propia

Como vemos el programa nos da las dimensiones que deberá tener el transformador, y el calibre de los bobinados primario y secundario, cabe recalcar que no se realizará el transformador de forma real, debido a el tiempo limitado y la falta de recursos, sin embargo, se realizaron los cálculos de todas formas, por si alguien más adelante cuenta con los recursos y el tiempo para para montar el prototipo completo.

Calculo para puente rectificador y capacitor de filtrado

Una vez realizado el trasformador pasaremos a la etapa de rectificación y filtrado.

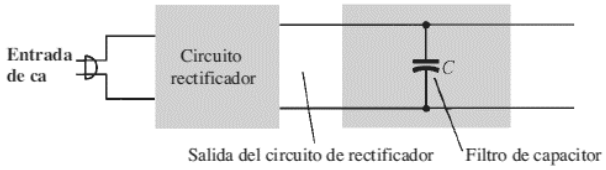

Figura 11. Etapa de filtrado – Imagen tomada del libro "Dispositivos Electrónicos" Por T.Floyd

Como vemos el objetivo de esta etapa es obtener a la salida una señal dc, sin embargo, esto no es del todo posible debido a que siempre tendremos un mínimo rizado a la salida del filtro tal como se muestra en la figura de abajo:

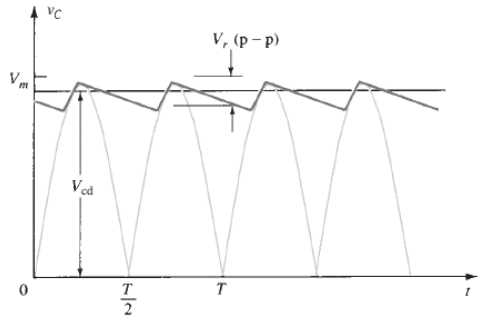

Figura 12. Voltaje de Rizado – Imagen tomada del libro "Dispositivos Electrónicos" Por T.Floyd

En este caso calcularemos el voltaje de rizado para un rizado máximo de 10% a una corriente de 1 amperio:

$$r = \frac{\text{voltaje de rizo (rms)}}{\text{voltaje de cd}} = \frac{V_r(\text{rms})}{V_{cd}} \times 100\%$$

Al despejar la ecuación anterior tendremos:

$$Vr(rms) = \frac{Vcd.r}{100} = \frac{(12).(10)}{100} = 1.2v$$

Ya teniendo el voltaje de rizo podemos obtener la capacitancia de la siguiente ecuación:

$$C = \frac{Icd}{f.Vr(rms)} = \frac{1}{(120).(1.2)} = 6944.44uF$$

El valor anterior no es comercial, por lo cual usaremos 2 capacitores conectados en paralelo de 3300uF a 25v, más que suficiente para nuestro proyecto.

Cabe recalcar que el rizado será de 10% solo para una corriente de 1 amperio, para corrientes inferiores, con las cuelas trabajaremos, el rizado será mucho menor.

Simulación de la fuente lineal diseñada

Como vemos en la figura de abajo, se simuló el puente rectificador y capacitor de filtrado utilizando los valores antes calculados:

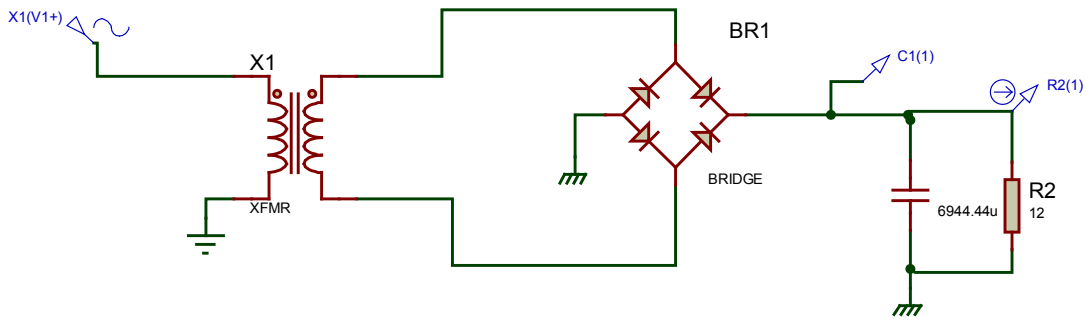

Figura 13. Simulación de la fuente lineal diseñada – Imagen de autoría propia utilizado el software proteus

Y como vemos en la gráfica de abajo, la señal color verde corresponde al voltaje a la salida del filtro y la señal color rojo a la corriente, comprobamos que el voltaje de rizado para el circuito que diseñamos, es de aproximadamente un 10%, para una corriente de carga de que consume 12v/12ohms=1A:

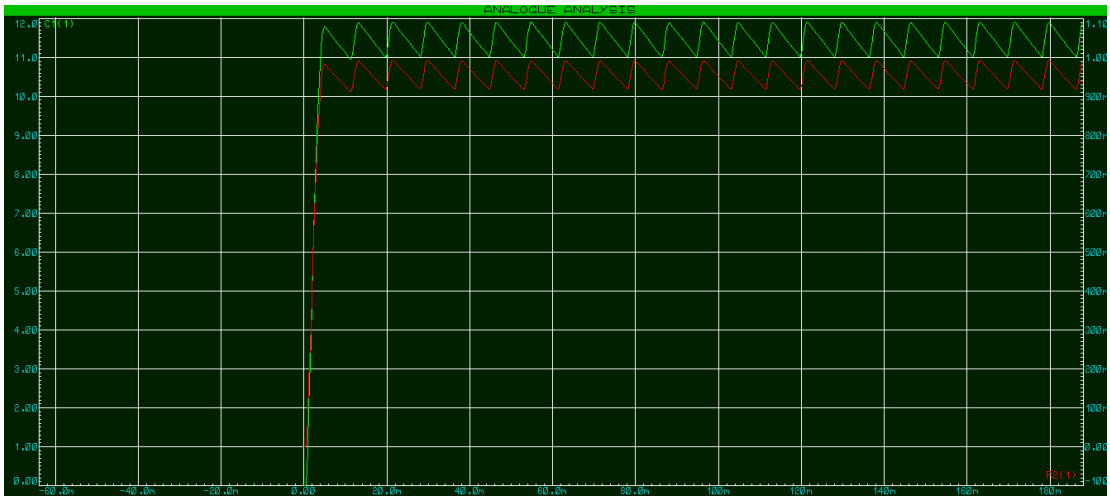

Figura 14. Voltaje de rizo a la salida del filtro – Imagen de autoría propia utilizado el software proteus

Protección de la fuente

Como protección para la fuente, vemos en la figura de abajo que se añadió un fusible de 2 amperios y una resistencia NTC "termistor de coeficiente de temperatura negativo" de 8 ohms, el fusible se emplea para evitar cualquier sobre corriente causada por un cortocircuito y la resistencia NTC para aumentar la durabilidad de los condensadores de filtrado, ya que cuando encendemos un equipo electrónico, los condensadores se encuentran descargados, lo cual genera un aumento de corriente por un pequeño instante mientras se cargan los condensadores, al agregar la resistencia NTC, limitamos esa corriente al inicio mientras se cargan los condensadores, una vez cargados los condensadores, ya ha aumentado la temperatura de la resistencia NTC y por lo tano disminuye su resistencia, dejando pasar toda la corriente a la fuente. (Acadenas, 2020)

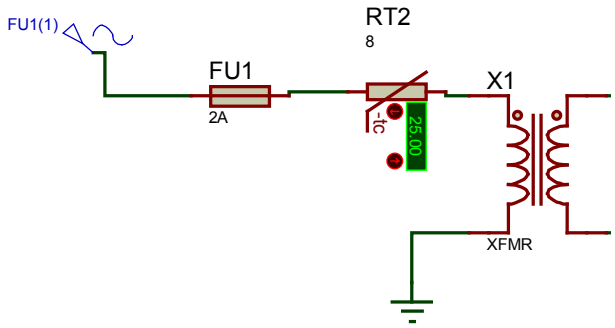

Figura 15. Circuito de protección de la fuente diseñada – Imagen de autoría propia utilizado el software proteus

En el siguiente video realizado por el profesor Aurelio Acadenas, en el minuto 19:20 explica por qué la utilización de una resistencia NTC de bajo valor ayuda a aumentar la vida útil de los condensadores de filtrado:

https://www.youtube.com/playlist?list=PLb_ph_WdlLDny2cGloFSxyRgO8B733jeo

Regulador de voltaje para el ESP8266 y el PAM 8403

Como vimos anteriormente, el ESP8266 tiene un regulador interno de 5v a 3.3v, en su pin Vin por lo cual necesitaremos de un regulador de 12v a 5v para que podamos obtener no solo el voltaje necesario para el ESP8266, sino también para el PAM8403, que como vimos también trabaja con un voltaje de 5v.

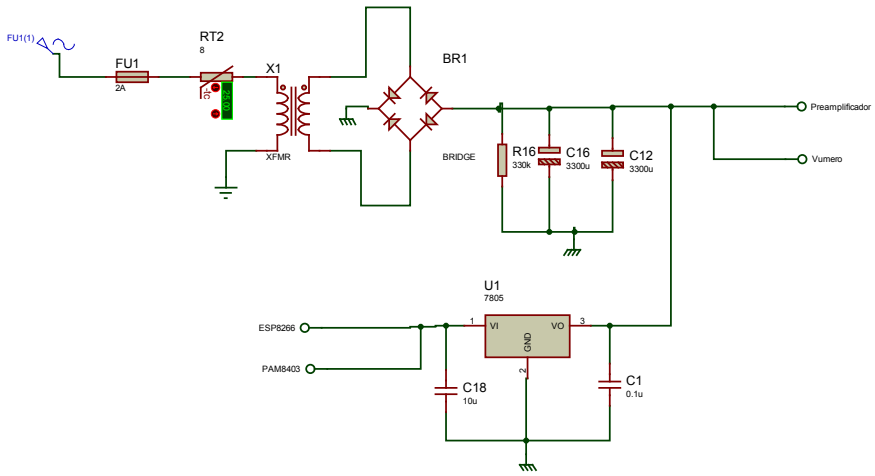

Figura 16. Regulador de voltaje para el ESP8266 – Imagen de autoría propia utilizado el software proteus

El regulador siempre estará trabajando, de forma que el ESP8266 siempre se encuentre encendido mientras el dispositivo se encuentre conectado a la red eléctrica, y podamos ver desde nuestro teléfono si el amplificador esta apagado o encendido, además de controlar el encendido y el apagado, esto último lo desarrollaremos más adelante, cuando diseñemos el programa para el ESP8266 en la plataforma de Arduino.

Como vemos en la fuente se añadió una resistencia de 330k en paralelo a los condensadores de filtrado, esto se hizo como medida de seguridad, de forma que los condensadores se descarguen al desconectar el prototipo de la red eléctrica.

Pre-Amplificador con ecualizador

En este caso el diseñó del pre amplificador se hizo basado el pre amplificador con ecualizador a 3 bandas mostrado en sitio web contruyasuvideorockola.com:

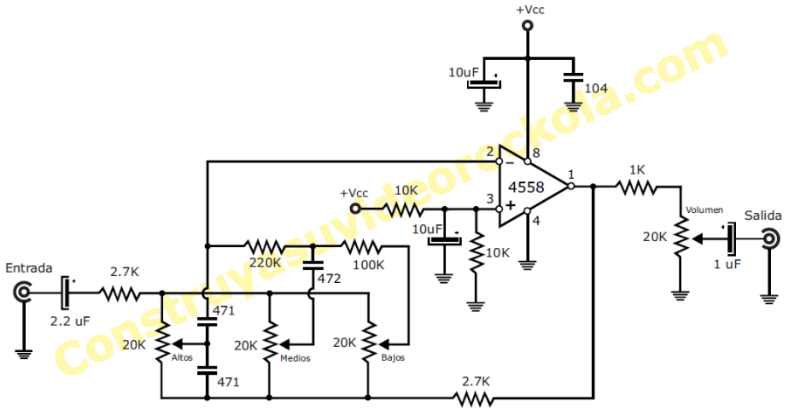

Figura 17. Pre amplificador con ecualizador– Imagen tomada de contruyasuvideorockola.com,

Simulación del Pre amplificador con ecualizador

El circuito que utilizaremos es el mismo que el anterior, con la diferencia que al no contar con el amplificador operacional Tl071, utilizaremos el TL084, un amplificador operacional con características muy similares al TL072, en la figura de abajo se puede ver la simulación del pre amplificador:

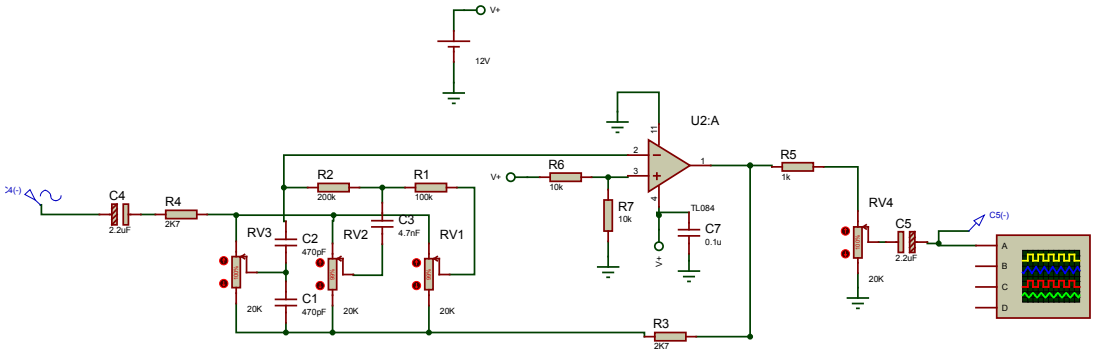

Figura 18. Simulación del Pre amplificador con ecualizador – Imagen de autoría propia utilizado el software proteus

Simulación de la respuesta en frecuencia del pre amplificador

Como mencionamos anteriormente, el ecualizador que realizaremos debe ser capaz de atenuar la señal para frecuencias bajas, medias y altas, en las gráficas de abajo podemos ver el diagrama espectral, del preamplificador para cada una de estas frecuencias, variando los potenciómetros:

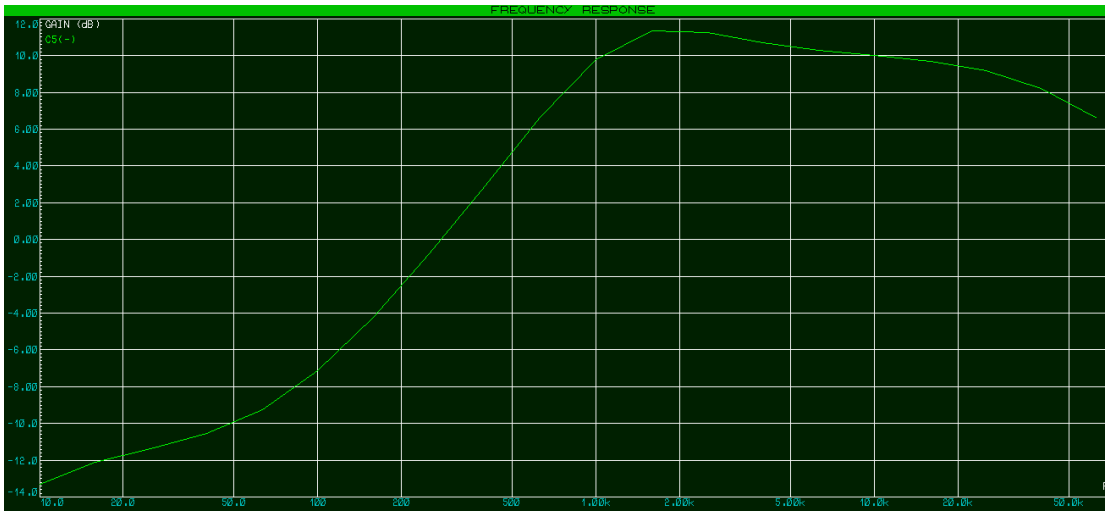

Figura 19. Variación del ecualizador para atenuar frecuencias bajas – Imagen de autoría propia utilizado el software proteus

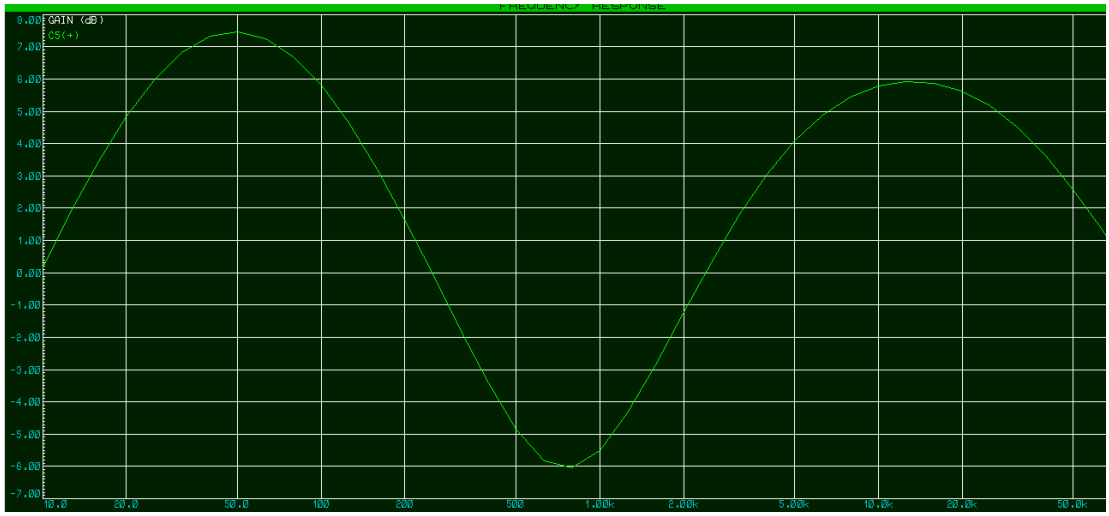

Figura 20. Variación del ecualizador para atenuar frecuencias medias – Imagen de autoría propia utilizado el software proteus

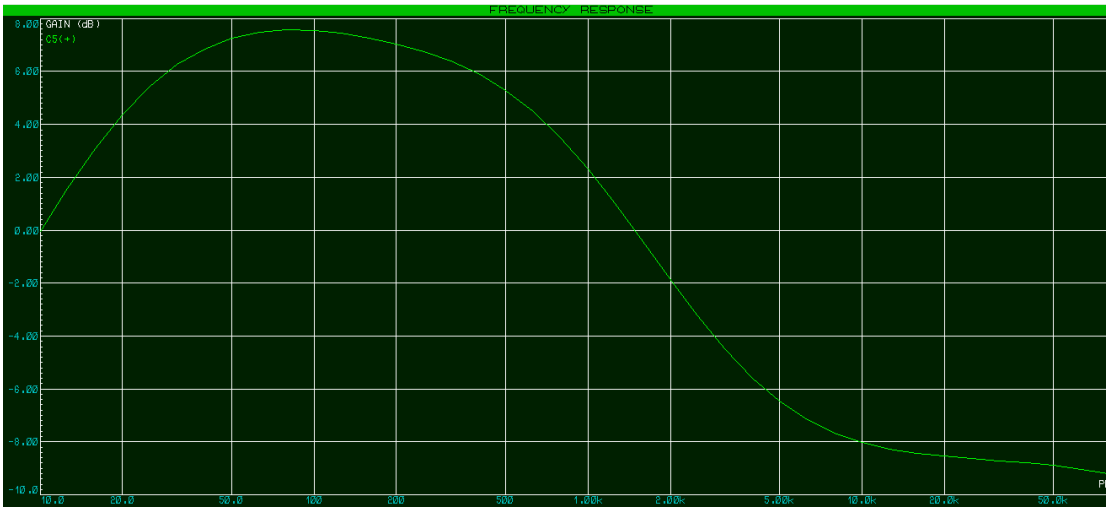

Figura 21. Variación del ecualizador para atenuar frecuencias altas – Imagen de autoría propia utilizado el software proteus

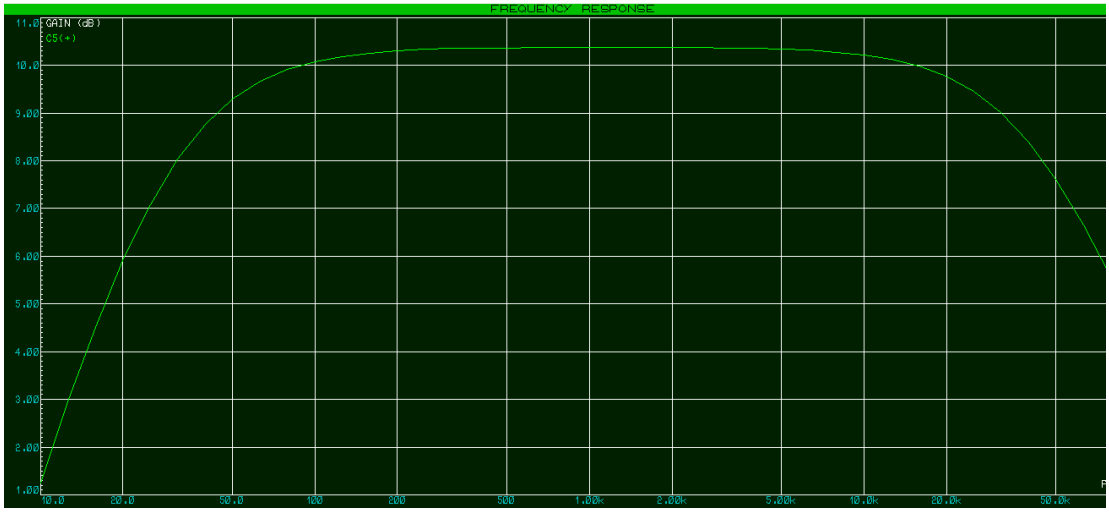

Figura 22. Preamplificador sin variar el ecualizador – Imagen de autoría propia utilizado el software proteus

Como vemos en la figuras anteriores, el preamplificador tiene una ganancia de 10 db y un ancho de banda de aproximadamente 20kHz, sin variar el ecualizador, lo ideal para el oído humano, también observamos que pudimos atenuar frecuencias determinadas, para el caso de bajos 0-256Hz, medios 256-2kHz y agudos de 2kHz a 20khz simplemente variando los potenciómetros, de esta forma comprobamos la respuesta en frecuencia de nuestro pre amplificador, el cual funciona de la manera adecuada en la simulación y cumpliendo con las especificaciones que teníamos plantadas.

Etapa del vúmetro

El vúmetro consistirá en una serie de amplificadores operacionales en modo comparador, con los cuales se activará una escala de leds en pasos que dependerá del voltaje de referencia ajustado en el circuito integrado. En este caso utilizaremos el integrado LM3915 (kitelectronica.com, 2015)

Se utilizará como ejemplo el diagrama mostrado en la figura de bajo, encontrado el sitio web kitelectronica.com:

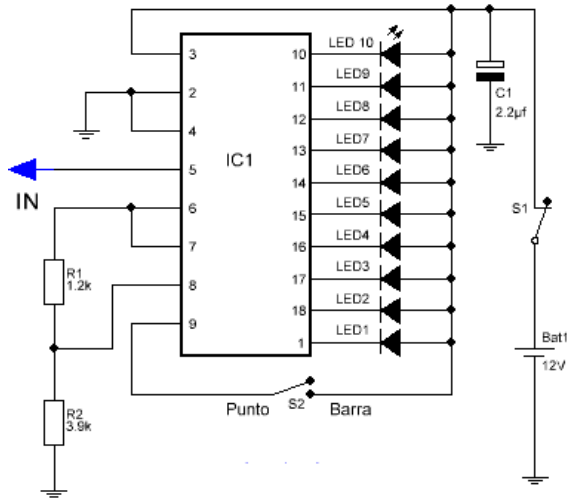

Figura 23. Circuito ejemplo del LM3915– Imagen tomada de kitelectronica.com

A través de los pines 6, 7 y 8 podemos ajustar el voltaje de referencia, para activar la escala de leds. aplicando la ecuación propuesta en la datasheet del LM3915 (Voltaje de referencia = 1.25 * (1+ R2/R1)), obtendremos el rango de voltajes sobre el cual trabajara el LM3915. (kitelectronica.com, 2015).

Resolviendo la formula, obtendríamos el siguiente resultado:

$$1.25 * (220 / 2.7k + 1) = 1.35v.$$

Por lo cual para el vúmetro que deseñaremos cada led encenderá para los siguientes valores de voltaje:

Led 1 = 135mv
Led 2 = 270mv
Led 3 = 405mv
Led 4 = 540mv
Led 5 = 675mv
Led 6 = 810mv
Led 7 = 945mv
Led 8 = 1.08v
Led 9 = 1.2v
Led 10 = 1.35v

En la figura de abajo podemos ver el circuito simulado en proteus:

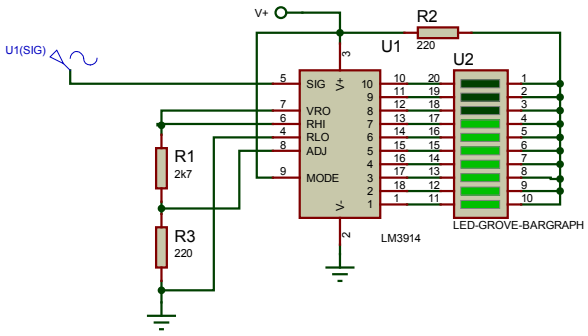

Figura 24. Simulación del vúmetro – Imagen de autoría propia utilizado el software proteus

Como vemos en la figura anterior se añadió una resistencia 220 ohms entre el voltaje de suministro y los leds, eso para limitar la corriente que pasan por los leds, este circuito funcionó de la manera correcta en el simulador en los rangos de voltaje establecidos.

PRIMERA PRUEBA EN EL LABORATORIO DEL PREAMPLIFICADOR CON ECUALIZADOR

El día martes 21 de noviembre de 2022 se realizó la primera prueba física del Pre amplificador con ecualizador en el laboratorio, la prueba consistió en inyectar una onda senoidal y variar la frecuencia para comprobar la respuesta en frecuencia del preamplificador, la señal ciertamente era amplificada, pero a la hora de variar los potenciómetros para señales de frecuencias agudas medias y bajas, no se generaba ningún cambio a la salida:

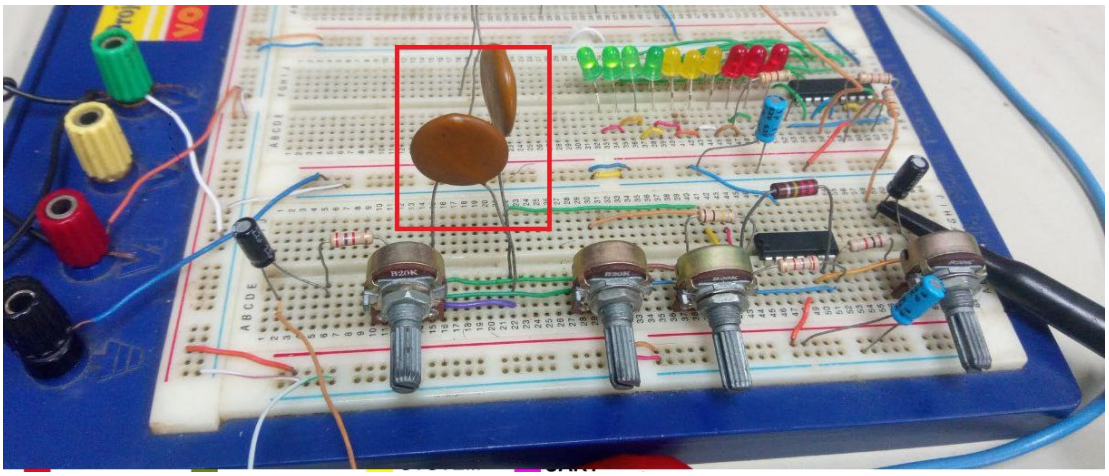

Figura 25. Preamplificador montado en el laboratorio– Imagen de autoría propia

Sin embargo, desde el principio se tenía la sospecha de que le problema estaba en los capacitores de filtrado de 470pF mostrados en la figura anterior, debido a que estos son los encargados de filtrar las señal para valores altos de frecuencia, un cambio en estos capacitores cambiaria el ancho de banda para filtrar las frecuencias altas, lo cual afectaría el filtrado de las frecuencias medias y bajas, una vez cambiados los capacitores el resultado fue el que se muestra las paginas siguientes:

CONSIDERACIONES ANTES DE REALIZAR LA SEGUNDA PRUEBA DEL PREAMPLIFICADOR

Abajo se muestran los valores máximos de operación del PAM8403 tomados del sitio web datasheet.es:

Supply Voltage.....6.6V
Input Voltage.....-0.3V to V_{DD}+0.3V
Operation Temperature Range.....-40°C to 85°C
Maximum Junction Temperature.....150°C

Como vemos el PAM8403, puede trabajar con un voltaje de entrada máximo de apenas 0.3vp, voltajes superiores a este saturaran el amplificador y provocara distorsión a la salida, por lo cual debemos ajustar el voltaje a la salida del preamplificador para que trabaje entre -0.3v y 0.3v.

Esto se hizo aumentando el valor del resistor en serie al potenciómetro de volumen poco a poco, hasta alcanzar un voltaje apto para el PAM8403, en este caso el resistor para alcanzar este voltaje fue de 100k, de forma que se forme un divisor de voltaje que haga que la señal previamente amplificada resulte atenuada al llegar al PAM8403.

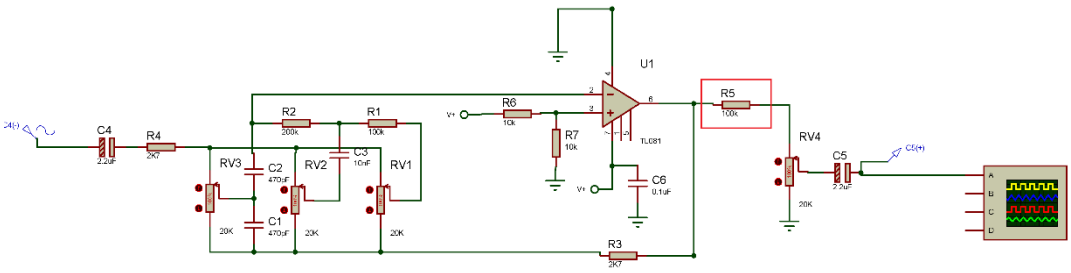

Figura 26. Cambio en el preamplificador con ecualizador – Imagen de autoría propia utilizado el software proteus

Sin embargo, el vúmetro que diseñamos trabaja para un voltaje de entrada de 0 a 1.3v, y este es el voltaje mínimo al que puede trabajar vúmetro, esto ocasiona que no podamos colocar la salida del preamplificador directamente el vúmetro en paralelo con el PAM8403, debido a que el vúmetro apenas detectaría el voltaje de 0 a 0.3v y no de 0 a 1.3v, de forma que encenderían solamente los 2 primeros leds, por lo cual conectaremos el vúmetro justo antes del divisor resistivo, como vemos en la figura de abajo, de forma que la señal que entre al vúmetro, sea la señal amplificada y no la señal atenuada por el divisor resistivo que va hacia el PAM8403, tal como vemos en la figura de abajo:

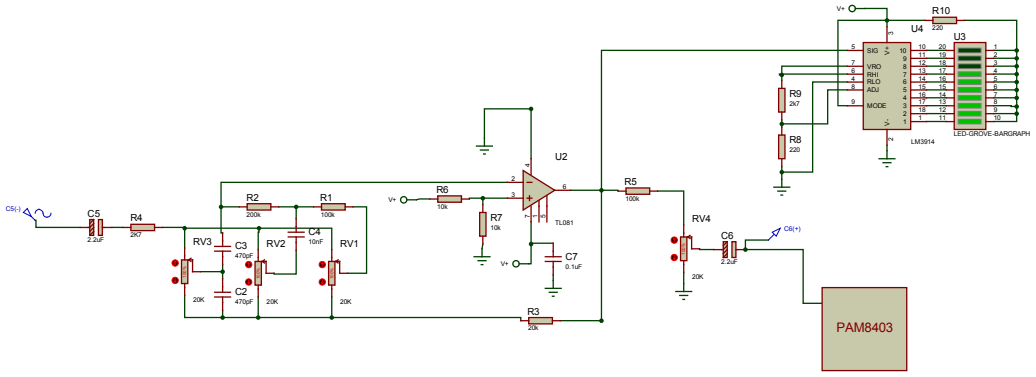

Figura 27. Conexión del preamplificador al PAM8403 y al vúmetro – Imagen de autoría propia utilizado el software proteus

SEGUNDA PRUEBA EN EL LABORATORIO DEL PREAMPLIFICADOR CON ECUALIZADOR

Teniendo en cuenta las consideraciones anteriores, el día martes 29 de diciembre de 2022 se realizó nuevamente la prueba del preamplificador. Para esta prueba se procedió a calcular experimentalmente el ancho de banda del preamplificador sin variar el ecualizador, tal como hicimos en la simulación.

Para esto se introdujo una onda senoidal de 300mVpp de amplitud a la entrada del preamplificador con ayuda del generador de señales, posteriormente se varió la frecuencia el generador hasta conseguir una amplitud máxima a la salida del preamplificador, a partir de este voltaje se obtuvieron los puntos de corte superior e inferior, buscando aproximadamente el 70% del voltaje máximo , o lo que es lo mismo una reducción de 3db del voltaje máximo y anotando la frecuencia en los que se encontraban estos voltajes, el resultado de este procedimiento se muestra en el cuadro de abajo:

Frecuencia (Hz)	Voltaje a la salida del Preamplificador (Vp)
100	0.17
4k	0.25
14k	0.17

Tabla 5. Datos de la prueba en el laboratorio de la respuesta en frecuencia del pre amplificador sin variar el ecualizador – Tabla de autoría propia utilizado Microsoft Word 2019

Del cuadro anterior podemos concluir que el ancho de banda del preamplificador sin variar el ecualizador va desde los 100hz hasta los 14kHz, un poco menor al resultado ideal que obtuvimos al realizar la prueba en el simulador, debido a que no estamos utilizando componentes no ideales sin embargo resultado fue el óptimo, ya que tonos tan altos son casi imperceptibles para el oído humano, en la figura de abajo se puede ver un gráfico del cuadro anterior:

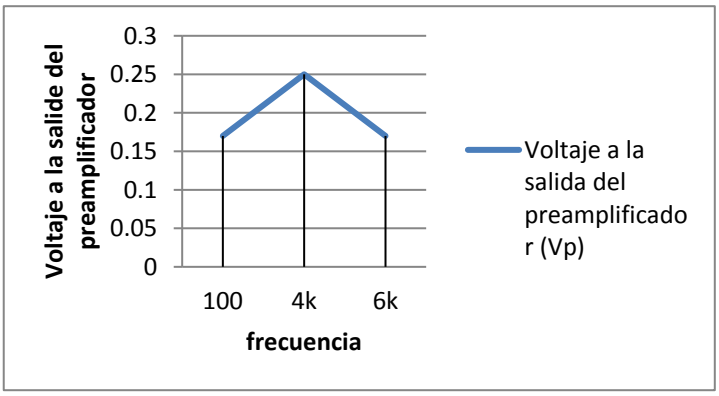

Figura 28. Gráfica de la prueba en el laboratorio de la respuesta en frecuencia del pre amplificador sin variar el ecualizador – Imagen de autoría propia utilizado el Microsoft Word 2019

Luego se hizo la prueba al variar el ecualizador de forma de atenuar frecuencias bajas (0-256 Hz), medias (256Hz-2k) y altas (2k-20khz), se obtuvieron los siguientes resultados al inyectar nuevamente un tono de 300mVp y tomar 4 mediciones de la señal a la salida, para distintas frecuencias de la señal de entrada:

Para una atenuación máxima de las frecuencias bajas:

Frecuencia (Hz)	Voltaje a la salida del Preamplificador (Vp)
100	0.08
4k	0.25
6k	0.25
14k	0.17

Tabla 6. Datos de la prueba en el laboratorio de la respuesta en frecuencia del pre amplificador al variar el ecualizador para atenuar frecuencias bajas – Tabla de autoría propia utilizado Microsoft Word 2019

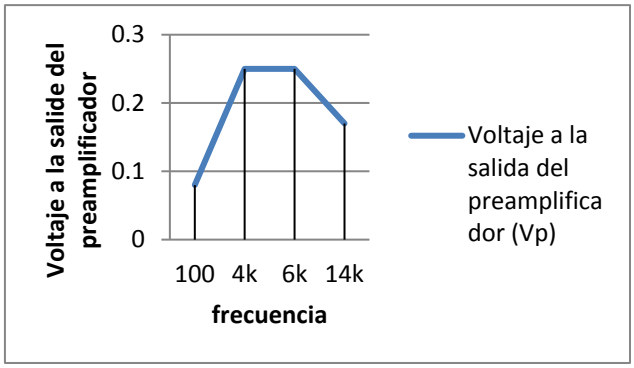

Figura 29. Gráfica de la prueba en el laboratorio de la respuesta en frecuencia del pre amplificador al variar el ecualizador para atenuar frecuencias bajas – Imagen de autoría propia utilizado Microsoft Word 2019

Para una atenuación máxima de las frecuencias medias:

Frecuencia (Hz)	Voltaje a la salida del Preamplificador (Vp)
100	0.15
4k	0.08
6k	0.07
14k	0.13

Tabla 7. Datos de la prueba en el laboratorio de la respuesta en frecuencia del pre amplificador al variar el ecualizador para atenuar frecuencias medias – Tabla de autoría propia utilizado Microsoft Word 2019

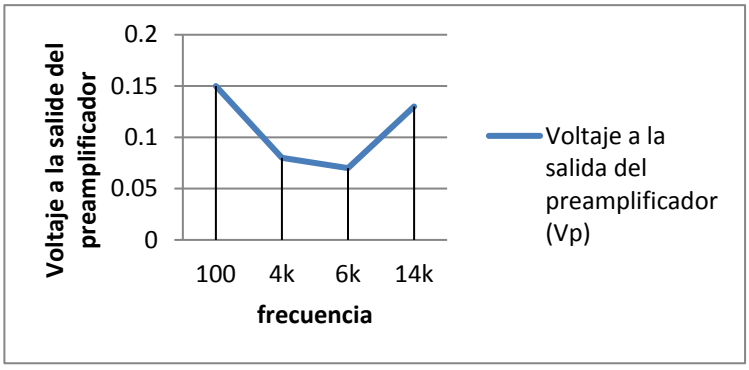

Figura 30. Gráfica de la prueba en el laboratorio de la respuesta en frecuencia del pre amplificador al variar el ecualizador para atenuar frecuencias medias – Imagen de autoría propia utilizado Microsoft Word 2019

Para una atenuación máxima de las frecuencias altas:

Frecuencia (Hz)	Voltaje a la salida del Preamplificador (Vp)
100	0.15
4k	0.23
6k	0.20
14k	0.06

Tabla 8. Datos de la prueba en el laboratorio de la respuesta en frecuencia del pre amplificador al variar el ecualizador para atenuar frecuencias altas – Tabla de autoría propia utilizado Microsoft Word 2019

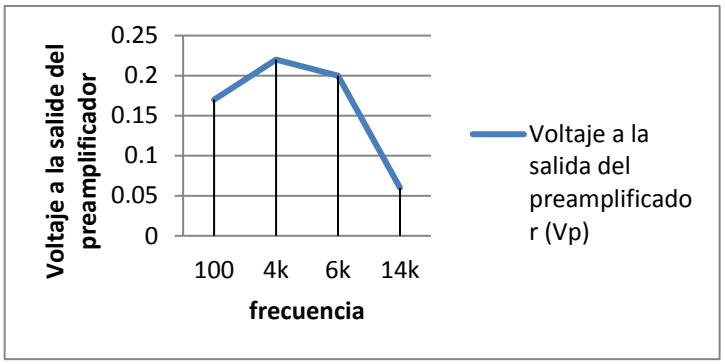

Figura 31. Gráfica de la prueba en el laboratorio de la respuesta en frecuencia del pre amplificador al variar el ecualizador para atenuar frecuencias altas – Imagen de autoría propia utilizado Microsoft Word 2019

Como vemos el ecualizador montado en el laboratorio no es perfecto debido a que no estamos utilizado componentes ideales como en la simulación, sin embargo, cumple su función de atenuar la señal de entrada en los rangos de frecuencia que establecimos, y generando los rangos de voltajes necesarios para el PAM8403 y el para el vúmetro.

CIRCUITO PARA CONTROL DE ENCENDIDO DEL AMPLIFICADOR

El ESP8266 puede proveer por sus pines tan solo 3.3v y una corriente de apenas 12mA, sin embargo podemos utilizar estos 3.3v para controlar el encendido del PAM8403, utilizando a las salidas de sus pines un transistor como conmutador, en este caso necesitaremos un transistor que soporte mínimo 6w/5v=1.2A de corriente de colector y 5v entre colector y emisor, del tipo PNP, en nuestro caso utilizaremos el transistor NTE 219, debido a que es el único que se consiguió con mayores características que las mencionadas disponible en la universidad, este transistor es capaz de soportar una corriente máxima de colector 15 A y un voltaje de colector a emisor de 60v, mucho más de lo que necesitamos.

Como vimos, la corriente máxima en el colector será la corriente máxima a la que puede trabajar el PAM8403, la cual es de 1.2A, el hfe "ganancia de corriente directa del parámetro hibrido emisor común" del transistor a utilizar según su datasheet puede variar entre 20 y 70, así que tomaremos un punto medio de 45, con estos datos podemos calcular la corriente de base en saturación:

$$Ib(sat) = \frac{ic(sat)}{hfe} = \frac{1.2}{45} = 30\text{mA}$$

Sin embargo, la corriente máxima que puede proporcionar el ESP8266 es de 12mA, así que calcularemos la resistencia de base para 10mA de corriente de base.

$$Rb = \frac{Vcc - 0.7 - \text{Vbb}}{Ib(sat)} = \frac{5\text{v} - 0.7 - 0}{10mA} = 430ohms$$

Donde Vcc representa el voltaje de la fuente que alimenta el PAM8403 y Vbb representa el voltaje de 0v en el pin D6 del ESP8266.

Como vemos en la figura de abajo se simuló el consumo del PAM8403 con una lampara de 6w, y como vemos obtuvimos aproximadamente los 10 mA de corriente de base en saturación para la resistencia de base que calculamos anteriormente.

Como se mencionó anteriormente se calculó este valor de corriente de base debido a que el ESP8266 soporta solo 12mA por sus pines de salida, por lo cual, como vemos en la figura de bajo seremos capaces de proporcionar al PAM8403 una corriente máxima de 424mA, sin embargo ya que utilizaremos un solo canal del PAM8403, esto realmente no ocasionara grandes problemas, ya que simplemente no utilizaremos el PAM8403 a su máxima capacidad.

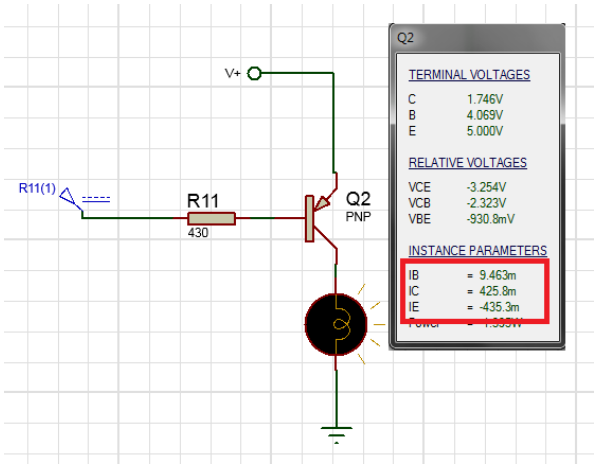

Figura 32. Prueba del control de encendido del ampliador– Imagen de autoría propia utilizado el software proteus

Si se desea utilizar el PAM8403 a su máxima capacidad utilizando 2 canales, se puede añadir un transistor en paralelo al transistor Q2, que se muestra en la figura anterior, de forma que los 10mA sean pre amplificados antes de entrar a la base de Q2 y así alcanzar los 30mA en su base para obtener una corriente de 1.2A en el colector de Q2, como se calculó anteriormente, esto último no se realizara por falta de tiempo y de componentes a la hora de realizar este informe.

.

Como mencionamos anteriormente, el encendido y el apagado del amplificador será de forma inalámbrica, sin embargo, este contará también con un botón de encendido manual, de forma que no se necesite estar conectado a la red wifi para encender el amplificador, esto lo haremos conectando un pulsador en configuración pull up en el pin D5 del ESP8266, de forma que cuando presionemos el pulsador. el ESP8266 interprete un 0 lógico en el pin D5, que hará que el pin D7 se ponga nivel bajo, de forma que el transistor PNP conduzca entre emisor y colector y deje pasar los 5v de alimentación al PAM8403, tal como calculamos u simulamos anteriormente.

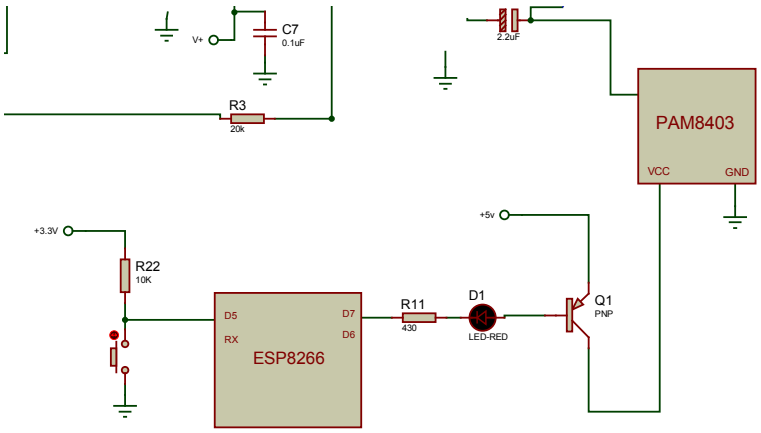

Figura 33. Circuito para control de encendido del amplificador – Imagen de autoría propia utilizado el software proteus

CIRCUITO PARA EL CONTROL DE ENCENDIDO DEL VÚMETRO

Para el caso de vúmetro cada led consume aproximadamente 6mW, al estar encendidos todos los leds tendremos un consumo máximo de aproximadamente 60mw, cada led necesita una corriente de aproximadamente 3mA para encender, por lo cual los 10 leds en paralelo consumirán 30mA, con estos 30mA podemos obtener la potencia que consume la resistencia que limita la corriente por los leds: (220ohms).(30mA)=198mw, en este caso necesitaremos un transistor que soporte mínimo (198m+60)mw/12v=21.5mA de corriente de colector y 12 voltios entre colector y emisor, en nuestro caso utilizaremos el transistor 2n2222, este transistor es capaz de soportar una corriente máxima de colector 800mA y un voltaje de colector a emisor de 40v, mucho más de lo que necesitamos.

Como vimos, la corriente máxima en el colector será de 21.5mA, el hfe del transistor a utilizar es típicamente de 100, así que esto podemos calcular la corriente de base:

$$Ib(sat) = \frac{ic(sat)}{hfe} = \frac{21.5\text{m}}{100} = 215\text{uA}$$

.

La resistencia de base será entonces de:

$$Rb = \frac{Vbb - 0.7}{Ib(sat)} = \frac{3.3\text{v} - 0.7}{215u} = 12.09\text{k}ohms$$

Donde Vbb representa el voltaje de 3.3v que viene del pin D6 del ESP8266.

Como vemos en la figura de abajo, al igual que con el PAM8403, se simuló el consumo del vúmetro con una lampara de 258mw, y como vemos obtuvimos aproximadamente los 215uA de corriente de base en saturación, con los cálculos para la resistencia de base que hicimos anteriormente.

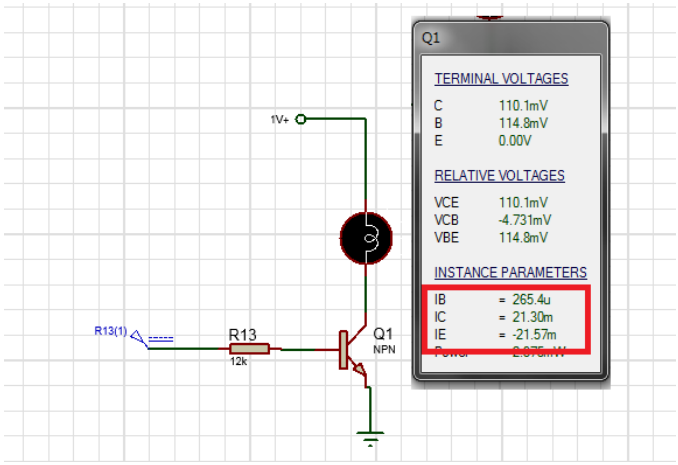

Figura 34. Prueba del control de encendido del vúmetro – Imagen de autoría propia utilizado el software proteus

El vúmetro solo se podrá encender y apagar de forma inalámbrica, y esto se hará a través del pin D6, si este pin transmite un nivel alto de tensión, el transistor NPN conducirá entre emisor y colector, conectando el circuito del vúmetro a tierra y haciendo que el vúmetro empiece a trabajar, como vemos en la figura de abajo:

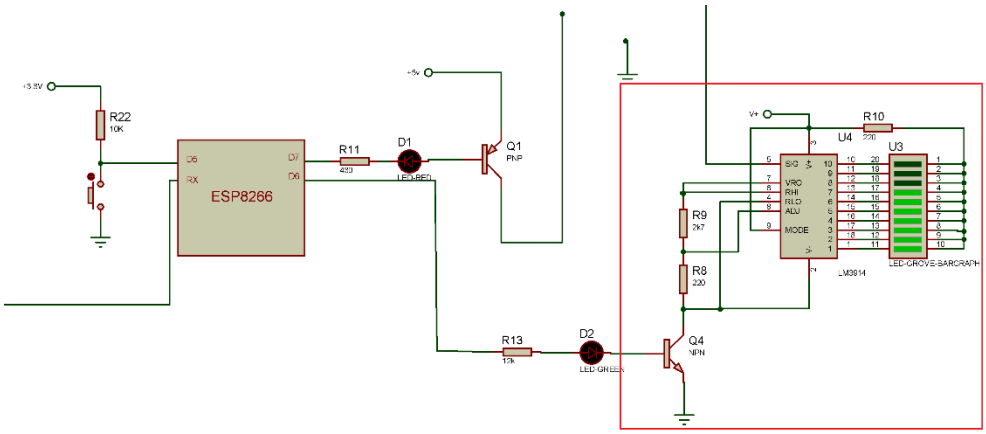

Figura 35. Circuito para control de encendido del vúmetro– Imagen de autoría propia utilizado el software proteus

EJECUCIÓN DE AUDIO AL ENCENDER EL AMPLIFICADOR

Al principio se tenía pensado que el amplificador que diseñamos tuviera una radio, pero lamentablemente esto no se podrá llevar a cabo, debido a las dificultades encontradas al utilizar el protocolo MQTT para la recepción de datos en tiempo real, sin embargo, en la búsqueda de la recepción de audio utilizando el ESP8266, se encontró una forma de utilizar la memoria flash del ESP8266 para almacenar los datos hexadecimales de un archivo de audio en formato .wav y poder ejecutarlo en cualquier parte del código.

Al ejecutarse el audio, este se transmitirá por el pin Rx del ESP8266 debido a que este no cuenta con pin DAC para la conversón de datos binarios a analógicos. Esto último hace necesario la utilización de la librería ESP8266 audio, para la conversión y transmisión de a través del pin Rx.

Todo el proceso anterior está explicado y se puede ver en el siguiente video: https://youtu.be/p6BzdsJR4mE

Explicado lo anterior, utilizaremos esto para que en el momento que encendamos nuestro amplificador se ejecute audio que nos indique que se ha encendido el amplificador.

CIRCUITO PARA LA EJECUCIÓN DEL AUDIO AL ENCENDER EL AMPLIFICADOR

El circuito para la ejecución del audio consta simplemente de una resistencia de 10k ohms conectada entre el pin Rx y la entrada del pre amplificador, esta resistencia cumple la función de atenuar la señal que proviene del ESP8266, el valor de la resistencia fue tomado al variar poco a poco el valor de la resistencia hasta obtener una señal entre 0.3v y -0.3v en la entrada del PAM8403.

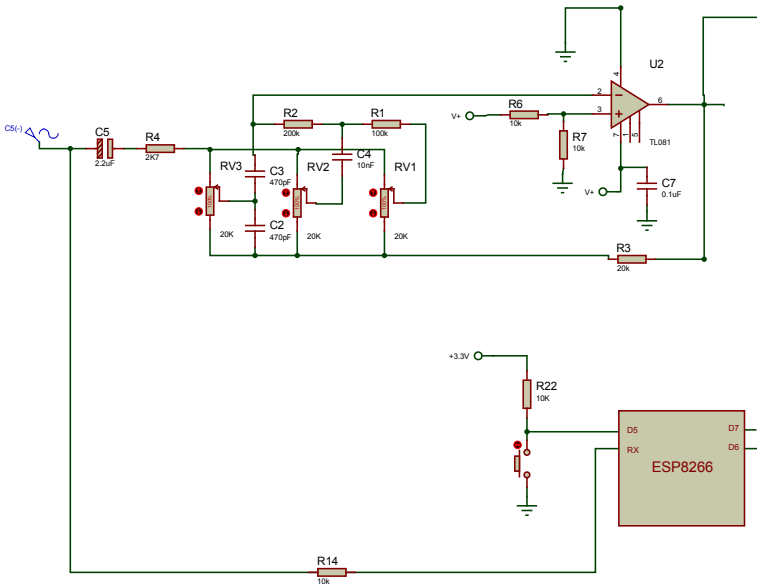

Figura 36. Circuito para la ejecución del audio al encender el amplificador – Imagen de autoría propia utilizado el software proteus

CONFIGURACIÓN DE LA APLICACIÓN MÓVIL

Como mencionamos anteriormente, la aplicación a utilizar será MQTTTDASHBORAD disponible en para el sistema operativo Android, En la figura de abajo se pude ver la configuración para la conexión al bróker desde la aplicación, la cual consiste simplemente en introducción de los datos con los que nos registramos en el bróker.

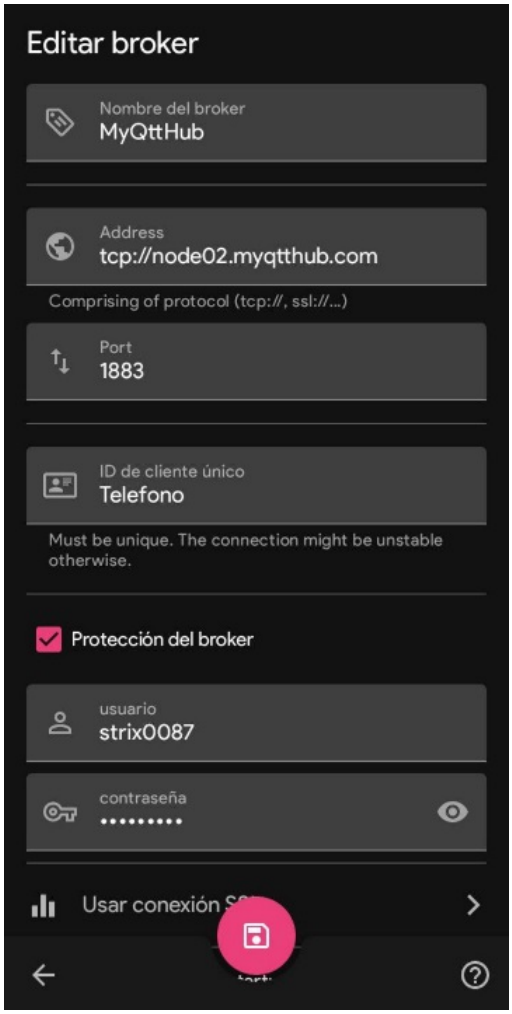

Figura 37. Configuración de la aplicación para la conexión al broker – Imagen de autoría propia

Luego de configurar la aplicación para conectarnos al bróker, podemos añadir diversas herramientas para interactuar, como por ejemplo un pulsador, este pulsador debe configurarse con el topic y el payload correspondiente para la acción que queremos que se realice al presionarlo, el topic lo describimos en el marco teórico, es un identificador que define el contenido del mensaje, en este caso amplificador y el payload es la instrucción que queremos que se realice, en este caso encender:

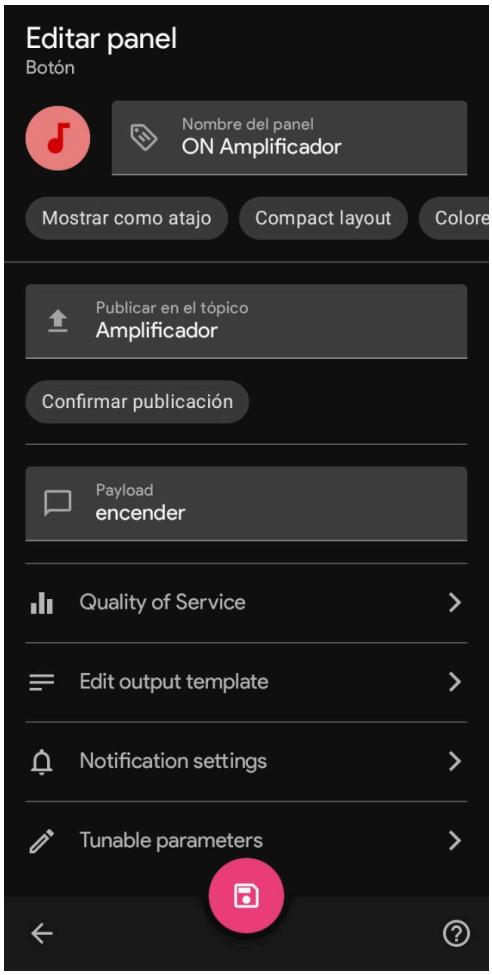

Figura 38. Configuración de herramientas de interacción de la aplicación – Imagen de autoría propia

El proceso de creación del bróker y configuración de la aplicación se puede ver en este video: https://youtu.be/vuuWWPw9ZNs

DISEÑO DEL PROGRAMA PARA EL ESP8266

Librería encendido.h

Como mencionamos anteriormente para la ejecución del audio debemos guardar los datos hexadecimales del audio en la memoria flash del ESP8266, para ello crearemos una librería con el nombre encedido.h, tal como vemos en la figura de abajo:

```
PROYECTO_FINAL_RESPALDO.lnk    encendido.h
const unsigned char encendido[] PROGMEM = {
  0x52, 0x49, 0x46, 0x46, 0x2E, 0x0A, 0x01, 0x00, 0x57, 0x41, 0x56, 0x45, 0x66, 0x6D, 0x74, 0x20,
0x10, 0x00, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00, 0x22, 0x56, 0x00, 0x00, 0x22, 0x56, 0x00, 0x00,
0x01, 0x00, 0x08, 0x00, 0x64, 0x61, 0x74, 0x61, 0x0F, 0x09, 0x01, 0x00, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
```

Figura 39. Librería encendido.h– Imagen de autoría propia utilizado la plataforma de Arduino

Como vemos a pesar de ser un audio corto, de apenas 2 segundos, la cantidad de datos a almacenar es relativamente grande.

Librerías a utilizar

Las librerías que utilizaremos serán ESP8266WiFi, la cual nos ayudará a conectarnos a la red wifi, la librería PubSubClient, la cual fue descrita en el marco teórico y nos ayudara a conectarnos al bróker, y en la transmisión y recepción de datos por medio del protocolo MQTT, la librería encendido la cual creamos anteriormente, donde se encuentran los datos del audio a ejecutar, y por ultimo las librerías AudioFileSourcePROGMEM.h, AudioGeneratorWAV.h y AudioOutputI2SNoDAC, las cuales vienen incluidas en al descargar la librería ESP8266 Audio, y serán las encargadas de ejecutar el audio al encender el amplificador:

```
#include <ESP8266WiFi.h>;                       // Nos ayuda a conectarnos a la red wifi
#include <PubSubClient.h>;                      // Nos ayuda a conectarnos al broker MQTT

#include "encendido.h"                          // Datos del audio
#include "AudioFileSourcePROGMEM.h"             // Nos ayudan a ejecutar el audio de encendido
#include "AudioGeneratorWAV.h"                  //
#include "AudioOutputI2SNoDAC.h"                //
```

Figura 40. Librerías a utilizar– Imagen de autoría propia utilizado la plataforma de Arduino

Datos de la red wifi y del bróker a utilizar

Comenzamos el programa, añadiendo como variables los datos de la red wifi a la cual nos conectaremos y los datos del bróker, en nuestro caso es el bróker MyHubMQTT, como mencionamos en el marco teórico:

```
//--------------------------------------DATOS DEL WIFI-----------------------------------------------------------------------

const char* ssid = "A9";                                //Ingreso el nombre de la red wifi ala que me deseo conectar
const char* password = "12345678";                      //Ingreso la contraseña de la red wifi ala que me deseo conectar


//--------------------------------------DATOS DEL BROKER---------------------------------------------------------------------

const char* mqtt_server = "node02.myqtthub.com";        // ingreso la direccion del broker al que me deseo conectar
const char* Id = "ESP8266";                             // ingreso la Id del broker al que me deseo conectar
const char* User = "strix0087";                         // ingreso el nombre de usuario con el que me registre en el broker
const char* CodePass = "draimy123";                     // ingreso la contraseña con la que que me registre en el broker
```

Figura 41. Datos de la red wifi y del bróker a utilizar – Imagen de autoría propia utilizado la plataforma de Arduino

Otras variables

Seguimos el programa añadiendo otras variables que utilizaremos más adelante:

```
//-----------------------OTRAS VARIABLES VARIABLES----------------------------

AudioGeneratorWAV *wav;
AudioFileSourcePROGMEM *file;
AudioOutputI2SNoDAC *out;

WiFiClient espClient;
PubSubClient client(espClient);
String _topic;
String _payload;


String strPulsador;                  // indica si el amplificador esta apagado o encendido
String strvumetro;                   // indica si el amplificador esta apagado o encendido
int newButtonState= 0;               // indica el estadon actual del botón de encendido y apagado del amplificador
int oldButtonState= 0;               // indica el estado anterior del botón de encendido y apagado delamplificador
int ampliState= 1;                   // Indica el encedido y apagado del amplificador
```

Figura 42. Otras variables – Imagen de autoría propia utilizado la plataforma de Arduino

Función setup_wifi

La función setup_wifi se encarga de conectarnos a la red wifi, como vemos si no estamos conectados a la red wifi se ejecutará un bucle con las funciones audio y PushButton, las cuales, como veremos más adelante, son las encargadas de encender y apagar el amplificador de forma manual, utilizando un pulsador, de esta forma nos aseguramos de que en todo momento podamos encender y apagar el amplificador, sin necesidad de estar conectados a la red wifi.

```
//----------------------------CONEXIÓN AL WIFI----------------------------------------------------

void setup_wifi() {
  delay(10);                                    // Espero 10ms
  Serial.println();                             // Comenzamos a coectarnos al wifi
  Serial.print("\nConectando a ");               // Imprimo conectado a
  Serial.println(ssid);                         // imprimo en nombre de la red wifi
  WiFi.begin(ssid, password);                   // comienzo a conectarme al broker

  while (WiFi.status() != WL_CONNECTED) {       // Mientras no este conectado al broker
   delay(10);
   audio();                                     // Preparo la reproducción del audio
   PushButton();                                // Ver si se precionó el boton de encendido
  }
  Serial.println("");                           // Si ya realice la conexión
  Serial.println("\nConectado al WiFi");        // Imprimir conectado al wifi
  Serial.println("\n Direccion IP : ");         // Imprimo la direccion Ip del dispositivo conectado
  Serial.println(WiFi.localIP());               //
}
```

Figura 43. Función setup_wifi – Imagen de autoría propia utilizado la plataforma de arduino

Función PushButton

Como mencionamos anteriormente la función PushButton, será la encargada del encendido manual del amplificador, al ejecutarse comprobamos si el pulsador fue presionado, en caso afirmativo cambiamos el estado del amplificador, si estaba apagado, al presionar estará encendido y viceversa, luego se ejecutará el audio de encendido guardado en la memoria flash y se mandará un mensaje al cliente del cambio de estado del amplificador con el topic "pulsador", de forma que podamos saber si el amplificador a esta en encendido o apagado en todo momento desde la aplicación.

```
//------------------------ENCENDIDO Y APAGADO DEL AMPLIFICADOR-------------------------------------------

    void PushButton() {
    newButtonState=digitalRead(D5);                                          // Guardo el valor del pin D5
    if(newButtonState != oldButtonState){                                    // Si el valor es distinto al valor anterior
    oldButtonState = newButtonState;                                         // Guardo el nuevo
    if (newButtonState == 0) {                                               // Si el valor nuevo es igual a 0
    ampliState = !ampliState;                                                // Negar el estado (ON/OFF) del amplificador
    if (ampliState == 0) {                                                   // Si el el estado es 0
    strPulsador = "encendido";                                               // Indico que el amplificador esta encendido
    file = new AudioFileSourcePROGMEM( encendido, sizeof(encendido) );       // Ejecuto el audio de encendido
    out = new AudioOutputI2SNoDAC();                                         //
    wav = new AudioGeneratorWAV();                                           //
    wav->begin(file, out);                                                   //
}
    else {
    strPulsador = "apagado";                                                 // Sino vndico que el amplificador esta apagado
}
    digitalWrite(D7, ampliState);                                            // Encendemos o apagamos el amplificaodor
    client.publish("pulsador",String(strPulsador).c_str());                  // Aviso alcliente si el amplificdor esta encendid
    Serial.println("\nEnviando: ");                                         // Imprimo que se envio el mensaje alcliente
    Serial.println(strPulsador);
    delay(1000);
  }
  }
}
```

Figura 44. Función PushButton – Imagen de autoría propia utilizado la plataforma de arduino

Función audio

La función audio debe ejecutarse justo antes de ejecutar un audio y se encarga de parar la ejecución del audio una vez finalizado su reproducción

```
//-------------------------------------------------audio-------------------------------------------------------------

void audio()
{
   if (wav->isRunning()) {
    if (!wav->loop()) wav->stop();
  } else {
    audioLogger = &Serial;
  }
}
```

Figura 45. Función audio – Imagen de autoría propia utilizado la plataforma de Arduino

Función reconect

La función reconect se ejecuta al perder la conexión con el broker y se encarga de reconectarnos al mismo, en la figura de abajo podemos ver que si no estamos conectados al bróker se ejecutará en bucle la función audio y PushButton, para que en caso de perder la conexión al broker en todo momento podamos apagar y encender el amplificador de forma manual utilizando el pulsador. Como vemos en la figura de abajo, al momento de conectarnos al bróker nos subscribimos a los tópicos amplificador y vúmetro, de forma que recibiremos mensajes solo si provienen de estos 2 tópicos a los que nos subscribimos.

```
//-------------------------------------------------------CONECTAR AL BROKER-------------------------------------------------------------------

void reconnect() {
  while (!client.connected()) {                                    // Esta funcion se ejecuta cada vez que no estamos conectados al servidor MQTT
    Serial.println("\nIntentando conectar al servidor MQTT... ");   // Mientras no se ejecute imprimir  Intentando conectar al servidor MQTT...
    if (client.connect(Id,User,CodePass)) {
    } else {
      audio();                                                     // Sino preparar el audio de encendido
      PushButton();                                                // Ver si se ha precionado el botón de encendido
      delay(10);
    }
      Serial.println("\nconnectado");                             // Si logramos conectarnos idicar que nos conectamos al servidor
      client.subscribe("Amplificador");                            // Subcribirnos a al topico Amplificador
      client.subscribe("Vumetro");                                 // Subcribirnos a al topico Vumetro
  }
}
```

Figura 46. Función reconect– Imagen de autoría propia utilizado la plataforma de arduino

Función callback

La función callback se ejecuta cada vez que enviamos o recibimos un mensaje del cliente, por lo cual justo después debemos colocar las instrucciones que queremos que se ejecuten dependiendo del tópico y el payload que se haya recibido o enviado.

Por ejemplo, si recibimos un mensaje con el tópico “Amplificador” y el payload “encender”, se ejecutará el callback y luego las instrucciones que queremos que se ejecuten para este tópico y payload, en este caso será escribir en el pin D7 un nivel bajo de tensión para encender el amplificador, ejecutar el audio de encendido y enviar un mensaje al cliente que amplificador se encuentra encendido, tal como vemos en la figura de abajo:

```
//------------------------RECEPCION Y ENVIO DE DATOS----------------------------

  void callback(char* topic, byte* payload, unsigned int length) {
  String conc_payload_;
  for (int i = 0; i < length; i++) {
    conc_payload_  += (char)payload[i];
  }
  _topic = topic;                                                    // Permite utilizar topico dentro del bucle loop
  _payload = conc_payload_;                                          // Perimite utilizar el payload dentro del bucle loop


    if (_topic == "Amplificador") {                                    // Si el topico es Amplificador
    if (_payload == "encender") {                                    // Y el payload es encender
    digitalWrite(D7, LOW);                                           // Encender amplificador
    ampliState=LOW;                                                  // Indico que se cambio el estado del amplificador
    delay(1000);                                                     // Espero 1 segundo
    file = new AudioFileSourcePROGMEM( encendido, sizeof(encendido) );  // ejecuto el audio de encendido
    out = new AudioOutputI2SNoDAC();                                 //
    wav = new AudioGeneratorWAV();                                   //
    wav->begin(file, out);                                           //
  }

    if (_payload == "apagar") {                                      // Si el payload es apagar
    digitalWrite(D7, HIGH);                                          // Apagar amplificador
    ampliState=HIGH;                                                 // Indico que se cambio el estado del amplificador
  }

    if (ampliState == 0) {                                           // Si el amplificador esta encendido
    strPulsador = "encendido";                                       // variable srtpulsador igual a encendido
  }
    else {
    strPulsador = "apagado";                                         // Sino variable srtpulsador igual apagado
  }
    client.publish("pulsador",String(strPulsador).c_str());          // Aviso al cliente si el amplificdor esta encendido
    Serial.println("\nEnviando: ");                                  // Imprimo que se envio el mensaje alcliente
    Serial.println(strPulsador);

    Serial.print("\nmensaje recibido de [");                         // Indico que se recibio un mensjae
    Serial.print(_topic);
    Serial.println("] ");
    Serial.println(_payload);
  }

    if (_topic == "Vumetro") {                                   // Si el topico es Vumetro
    if (_payload == "encender") {                                // Y el payload es encender
    digitalWrite(D6, HIGH);                                      // Encender Vumetro
    strvumetro = "encendido";                                    // Indico que se cambió el estado del amplificador
    }
    if (_payload == "apagar") {                                  // Si el payload es apagar
    digitalWrite(D6, LOW);                                       // Encender Vumetro
     strvumetro = "apagado";                                     // Sino variable srtpulsador igual no presuionado
    }

    client.publish("vumetro/on/off",String(strvumetro).c_str());  // Aviso al cliente si el vúmetro esta encendido
    Serial.println("\nEnviando: ");                               // Imprimo que se envio el mensaje alcliente
    Serial.println(strPulsador);

    Serial.print("\nmensaje recibido de [");                     // Indicar que se recibio un mensjae
    Serial.print(_topic);
    Serial.println("] ");
    Serial.println(_payload);
    }
}
```

Figura 47. Función callback– Imagen de autoría propia utilizado la plataforma de arduino

Función setup

La función setup se ejecuta al inicio del programa y se encarga de configurar los pines del ESP8266, indicar las variables para ejecutar el audio de encendido, conectarnos a la red wifi, conectarnos al broker y comenzar el envío y recepción de datos:

```
//-----------------------------------------SETUP---------------------------------------------------------

void setup() {
  pinMode(D7, OUTPUT);                                          // Configuramos D7 salida analógica
  digitalWrite(D7, HIGH);                                       // Apagamos el amplificador
  pinMode(D6, OUTPUT);                                          // Configuramos D6 salida digital
  digitalWrite(D6, LOW);                                        // Apagamos el vúmetro
  pinMode(D5, INPUT);                                           // Configuramos el pin D5 como entrada
  digitalWrite(BUILTIN_LED, HIGH);                              // Apagamos el led que viene por defecto en el ESP8266
  Serial.begin(115200);                                         // Configuramos el puerto serie  115200 baudios
  delay(1000);

  audioLogger = &Serial;                                        // Indico las variables para ejecutar el audio
  file = new AudioFileSourcePROGMEM( encendido, sizeof(encendido) );      //
  out = new AudioOutputI2SNoDAC();                              //
  wav = new AudioGeneratorWAV();                                //

  setup_wifi();                                                 // Llamamos a la funsion setup_wifi para conectarnos a la red wifi
  client.setServer(mqtt_server, 1883);                          // Nos conectamos al servidor MQTT
  client.setCallback(callback);                                 // Inicia el callback para recibir y enviar un mensaje
}
```

Figura 48. Función setup– Imagen de autoría propia utilizado la plataforma de arduino

Función loop

La función loop se ejecuta al terminar de ejecutarse la función setup, se encarga de reconectarnos al broker en caso de desconexión, ejecutar la función client.loop, la cual permite gestionar la recepción y envío de datos, y por último ejecutar la funciones audio y PushButton para en todo momento poder encender el amplificador de forma manual.

```
//-----------------------------------------------LOOP---------------------------------------------------------

void loop() {

  if (!client.connected()) {
    reconnect();                                // Si el cliente no esta conectado ejecutar la funcion reconect
  }
  client.loop();                                // Si esta conectado ejecutar bucle
  audio();
  PushButton();                                 // Ver si se a precionado el bóton de encendido
  delay(10);                                    // Esperrar 100ms
}
```

Figura 49. Función loop – Imagen de autoría propia utilizado la plataforma de arduino

CONSIDERACIONES ANTES DE REALIZAR EL CIRCUITO Y DISEÑO DEL PROGRAMA PARA LA MEDICIÓN DE TEMPERATURA

Antes de realizar el análisis para la el diseño del circuito y el programa para la medición de la temperatura, debemos saber que es un termistor y como funciona, ya que este es el dispositivo que utilizaremos para la medición de la temperatura.

Definición de termistor

El termistor es un sensor que varía su resistencia en función a la temperatura a la cual se encuentra expuesto, existen de 2 tipos los NTC "coeficiente negativo de temperatura", los cuales disminuyen su resistencia al aumentar la temperatura y los PTC "coeficiente positivo de temperatura", los cuales aumentan su resistencia al aumentar la temperatura. (Bitwise, 2018)

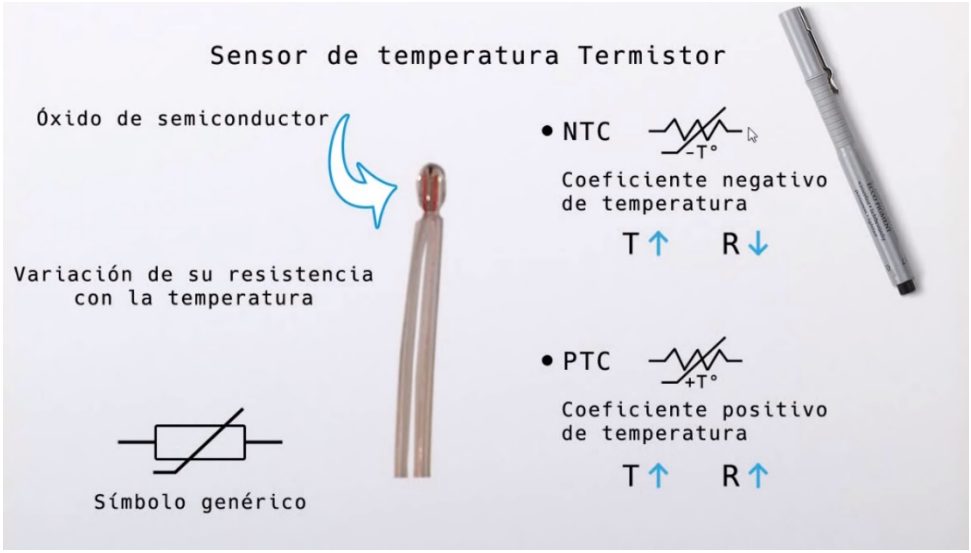

Figura 50. El termistor– Imagen tomada del video Arduino desde cero -Capitulo 4, Autor: Bitwise Ar

Una característica de los termistores NTC y PTC es que su respuesta no es lineal con respecto a la temperatura, como vemos en la gráfica de abajo:

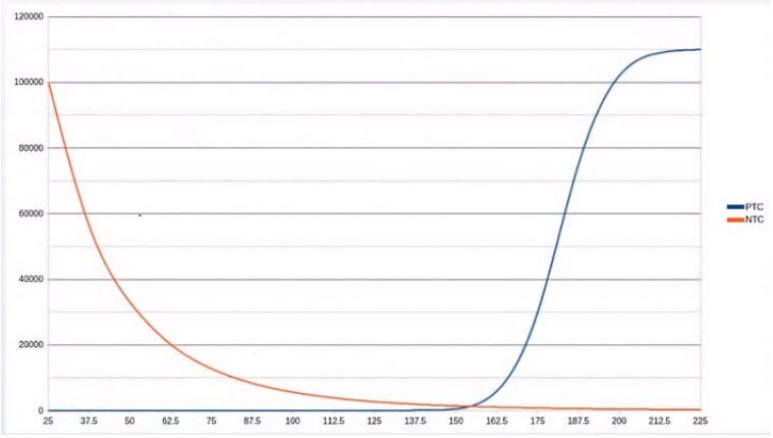

Figura 51. Respuesta del termistor en función a la temperatura – Imagen tomada del video Arduino desde cero -Capitulo 4, Autor: Bitwise Ar

La grafica color rojo representa la respuesta de un termistor NTC a variaciones de temperatura y la gráfica color azul a la respuesta de un termistor PTC.

Para convertir la variación de resistencia del termistor en temperatura se utiliza la ecuación de Stainhart-Hart, la cual podemos observar en la figura de abajo. (Bitwise, 2018)

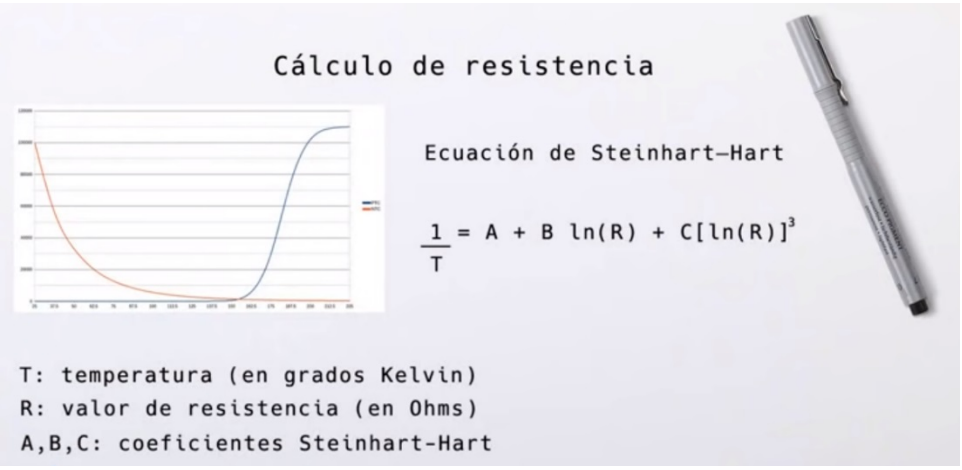

Figura 52. Ecuación de Steinhart-Hart– Imagen tomada del video Arduino desde cero -Capitulo 4, Autor: Bitwise Ar

Donde T es la temperatura en grados kelvin, así que al realizar el programa convertiremos su valor en grados centígrados, como veremos más adelante, R es la resistencia del termistor, A, B y C son los coeficientes de Stainhart-Hart y dependen del modelo de termistor que utilicemos, estos se obtienen de la hoja del fabricante del termistor, pero no siempre esta información es fácil de obtener, otra forma es utilizar una página web que permite obtener dichos coeficientes. (Bitwise, 2018)

Esta página se encuentra en el siguiente enlace: https://www.thinksrs.com/downloads/programs/therm%20calc/ntccalibrator/ntccalculator.html

Por defecto la página trae valores establecidos para un termistor de 10k, donde R1 es el valor de la resistencia que presenta el termistor para 5°, R2 para 25 y R3° para 45°, si estuviéramos trabajando con un termistor de 10k no es necesario hacer ningún cambio, sin embargo se utilizara un termistor de 100 ohms, así que debemos dividir R1, R2 y R3 entre 100, para obtener los valores de los coeficientes tal como vemos en la figura de abajo. (Bitwise, 2018)

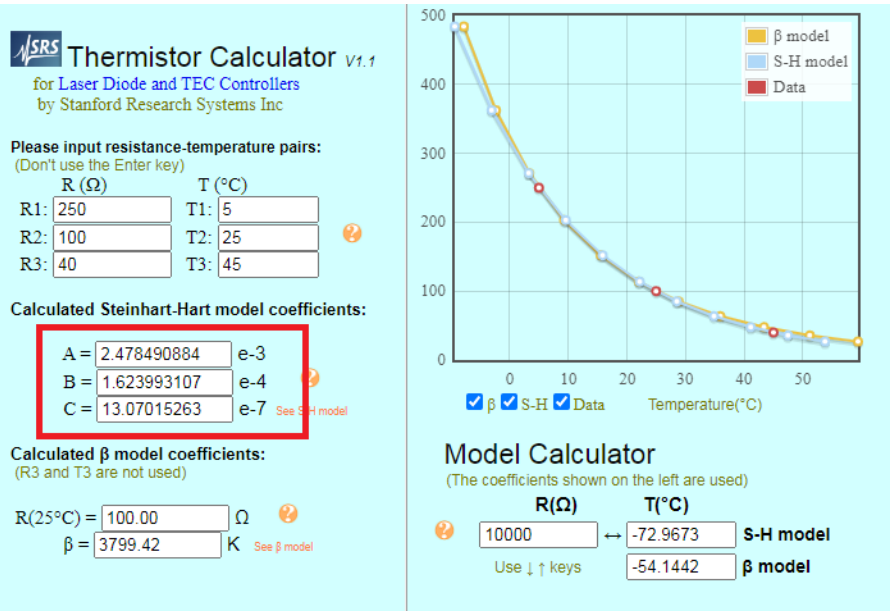

Figura 53. Coeficientes de Steinhart-Hart – Imagen de autoría propia utilizando la herramienta de Thermistor Calculator

Estos coeficientes son los que utilizaremos más adelante para la realización del programa.

Temperatura de trabajo del equipo

Otra cosa importante a tener en cuenta es la temperatura a la que nuestro equipo trabaja sin inconvenientes, en nuestro caso en el datashseet del PAM8403 nos provee de estos datos, como vemos en la figura de abajo:

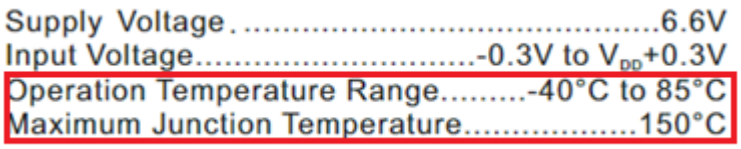

Figura 54. Temperatura del operación de PAM8403 – Imagen tomada de datasheet.es

Como vemos la temperatura máxima a la que puede trabajar el PAM8403 es de 85°, sin embargo no es lo ideal que trabaje a esa temperatura, para saber a qué temperatura encenderemos el ventilador, se realizó una prueba la cual consistió en medir la temperatura del PAM8403 estando trabajando a máxima potencia, el resultado fue una temperatura de 35°, esta medición se hizo utilizando el teléfono Umidigi A9 el cual cuenta con un sensor de temperatura integrado.

Por lo cual encenderemos el ventilador para un incremento mayor al 30% de esta temperatura, en este caso 45°, lo cual indicará un calentamiento producido por un factor externo al amplificador.

DISEÑO DEL CIRCUITO PARA EL CONTROL DE ENCENDIDO DEL VENTILADOR

El control de encendido y apagado se hará con un transistor, al igual que se hizo con el amplificador y con el vúmetro, solo que en este caso se utilizara como carga un pequeño ventilador que trabaja con 12v y 0.06 A, así que necesitaremos un transistor de mínimo 0.06A de corriente de colector y 12 voltios entre colector y emisor, en nuestro caso utilizaremos nuevamente el transistor 2n2222, este transistor es capaz de soportar una corriente máxima de colector 800mA y un voltaje de colector a emisor de 40v, mucho más de lo que necesitamos.

Como vimos, la corriente máxima en el colector será de 0.06A=60mA, el hfe del transistor a utilizar es típicamente de 100, así que con estos datos podemos calcular la corriente de base:

$$Rb = \frac{Ic(sat)}{hfe} = \frac{60\text{m}}{100} = 600\text{uA}$$

La resistencia de base será entonces de:

$$Rb = \frac{Vbb - 0.7}{Ib(sat)} = \frac{3.3\text{v} - 0.7}{0.6mA} = 4.3\text{k}ohms$$

Donde Vbb representa el voltaje de 3.3v que viene del pin D2 del ESP8266.

SIMULACIÓN DEL CIRCUITO PARA EL CONTROL DE ENCENDIDO DEL VENTILADOR

Como vemos en la figura de abajo, se simuló el ventilador con un motor de 12v y 60mA, con ello obtuvimos aproximadamente los 600uA de corriente de base, utilizando el valor de resistencia dada con los cálculos que hicimos anteriormente, igualmente obtuvimos aproximadamente los 60mA de corriente de corriente de colector necesarios para el funcionamiento del ventilador.

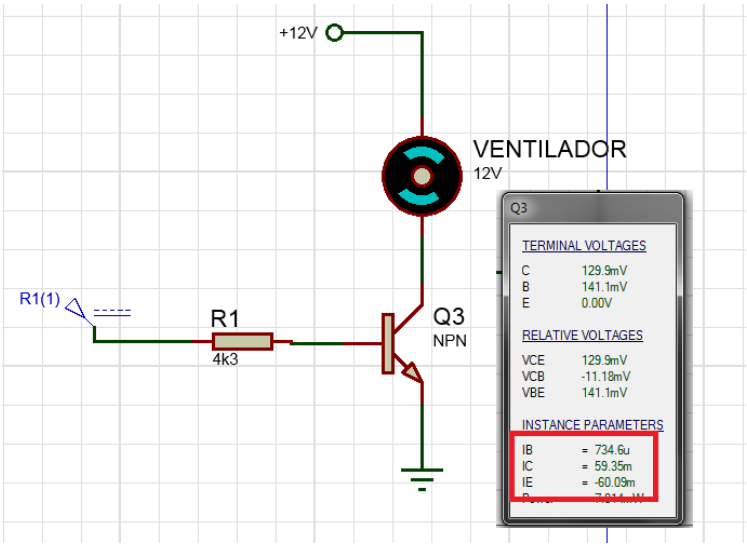

Figura 55. Prueba del circuito para control de encendido del ventilador– Imagen de autoría propia utilizado el software proteus

DISEÑO DEL CIRCUITO PARA LA MEDICIÓN DE TEMPERATURA

Para la medición de la temperatura se formó un divisor resistivo entre termistor ntc de 100 ohms y una resistencia de 100 ohms, de esta forma el voltaje en el pin A0 vendrá dado por la salida del divisor de voltaje y cambiará su valor dependiendo del valor de la resistencia ntc, si la temperatura aumenta el valor de resistencia del termistor disminuirá, aumentando el voltaje en el pin A0 y si la temperatura disminuye ocurrirá lo contrario.

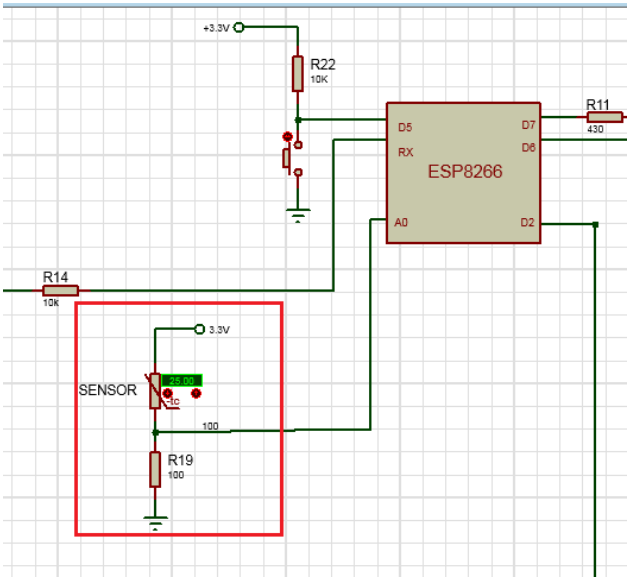

Figura 56. Circuito para la medición de temperatura – Imagen de autoría propia utilizado el software proteus

El circuito para la medición de temperatura mostrado en la figura anterior fue diseñado utilizando como ejemplo el mostrado en el video:
https://youtu.be/8Wry8lwgGtA

DISEÑO DEL PROGRAMA PARA LA MEDICIÓN DE TEMPERATURA Y ENCENDIDO DEL VENTILADOR

Al igual que el circuito anterior, el código diseñado a continuación fue hecho tomando como modelo el mostrado en el video https://youtu.be/8Wry8lwgGtA. Una vez sabiendo esto, podemos comenzar el diseño del programa para medición de temperatura y control de encendido del ventilador.

Variables para la medición de la temperatura

Comenzaremos añadiendo las variables a utilizar, en este caso como vemos se añadieron las constantes de steinhart-hart antes calculadas, se definió la resistencia en serie al termistor como R1 y el voltaje producto del divisor resistivo como V0:

```
//------------------------OTRAS VARIABLES VARIABLES-----------------------------

AudioGeneratorWAV *wav;
AudioFileSourcePROGMEM *file;
AudioOutputI2SNoDAC *out;

WiFiClient espClient;
PubSubClient client(espClient);
String _topic;
String _payload;

String strPulsador;                                          // Indica si el amplificador esta apagado o encendido
String strvumetro;                                           // Indica si el amplificador esta apagado o encendido
int newButtonState= 0;                                       // Indica el estado actual del botón de encendido y apagado del amplificador
int oldButtonState= 0;                                       // Indica el estado anterior del botón de encendido y apagado delamplificador
int ampliState= 1;                                           // Indica el encedido y apagado del amplificador

int Vo;                                                      // Es la tension leida en la entrada A0
float R1 = 100;                                              // Indica el valor la resistencia  en serie a la resistencia NTC
float logR2, R2, TEMPERATURA;                                // R2 es el valor la resistencia NTC y temperatura la variable a medir
float c1 = 2.478490884e-03, c2 = 1.623993107e-04, c3 = 13.07015263e-07;    // Constantes de steinhart-hart
unsigned long previousMillis = 0;                            // Esta variable reincia el tiempo de la lectura de la temperatura a 0
```

Figura 57. Variables para la medición de temperatura – Imagen de autoría propia utilizado la plataforma de arduino

Código para la medición de temperatura

Como vemos en la figura de abajo, el programa para el control de temperatura del amplificador se añadió directamente en la función loop, de forma que se ejecute constantemente una vez estemos conectados a la red wifi y al bróker.

El código para la medición de temperatura comienza condicionando la medición para que se ejecute cada 4ms, una vez pasado este tiempo se lee el valor del pin A0, donde se encuentra conectada la salida del divisor resistivo, luego utilizaremos este valor de voltaje para encontrar su equivalente en resistencia, para posteriormente sacar el logaritmo de esta resistencia calculada y así poder utilizarlo para calcular la temperatura utilizando la ecuación de steinhart-hart, esta ecuación como mencionamos anteriormente nos da la temperatura en grados kelvin, así que debemos convertir el resultado a su equivalente en grados centígrados restando al resultado 273,15°.

Ya teniendo la temperatura añadiremos 2 condicionantes, si la temperatura es mayor a 45°, indicaremos el encendido del ventilador y sino indicaremos el apagado del ventilador, una vez hecho esto se enviará un mensaje al cliente si el ventilador esta encendido o apagado utilizando el topic ventilador, y a su vez se enviará la temperatura utilizando el topic temperatura.

```
//-----------------------------------------------LOOP---------------------------------------------------

void loop() {

  if (!client.connected()) {
    reconnect();                                          // Si el cliente no esta conectado ejecutar la funcion reconect
  }
  client.loop();                                          // Si esta conectado ejecutar bucle
  audio();
  PushButton();                                           // Ver si se a precionado el bóton de encendido

  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= 4000) {           // Envia la temperatura solo si han pasado 4ms
  previousMillis = currentMillis;                         // Indico que ya pasaron 10 segundos
  Vo = analogRead(A0);                                    // Lectura de A0
  R2 = R1 * (1023.0 / (float)Vo - 1.0);                   // Conversion de tension a resistencia
  logR2 = log(R2);                                        // Logaritmo de R2 necesario para ecuacion
  TEMPERATURA = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2));  // Igualo la ecuacion steinhart-hart a la variable temperatura
  TEMPERATURA = TEMPERATURA - 273.15;                     // Convieto de Kelvin a Centigrados
  if (TEMPERATURA >= 45) {                                // Si la temperatura es mayor o igual a 45°
  digitalWrite(D2, HIGH);                                 // Enciendo el ventilador
  strventilador = "encendido";                            // Indico que se cambió el estado del ventilador
  }
  else {
  digitalWrite(D2, LOW);                                  // Apago el ventilador
  strventilador = "apagado";                              // Indico que se cambió el estado del ventilador
  }
  client.publish("ventilador",String(strventilador).c_str());  // Aviso al cliente que hay un sobrecalentamiento
  Serial.println("\nEnviando: ");                          // Imprimo que se envio el mensaje al cliente
  Serial.println(strventilador);

  client.publish("\ntemperatura",String(TEMPERATURA).c_str());  // Envio el valor de temperatura al cliente
  Serial.print(F("\nenviando temperatura a "));             // Indico que el valor de temperatura se envio
  Serial.print(TEMPERATURA);
  Serial.println("...");
  delay(10);                                              // Esperrar 10ms
  }
}
```

Figura 58. Código para la medición de temperatura – Imagen de autoría propia utilizado la plataforma de arduino

PROGRAMA FINAL PARA EL ESP8266 EN LA PLATAFORMA DE ARDUINO

```
/---------------------------------------LIBRERIAS A UTILIZAR-------------------------------------------------
#include <ESP8266WiFi.h>;                          // Nos ayuda a conectarnos a la red wifi
#include <PubSubClient.h>;                         // Nos ayuda a conectarnos al broker MQTT

#include "encendido.h"                             // Datos del audio
#include "AudioFileSourcePROGMEM.h"                // Nos ayudan a ejecutar el audio de encendido
#include "AudioGeneratorWAV.h"                     //
#include "AudioOutputI2SNoDAC.h"                   //

//---------------------------------------DATOS DEL WIFI-------------------------------------------------------

const char* ssid = "A9";                           // Ingreso el nombre de la red wifi ala que me deseo conectar
const char* password = "12345678";                 // Ingreso la contraseña de la red wifi ala que me deseo conectar


//-------------------------------------------------DATOS DEL BROKER-------------------------------------------------------

const char* mqtt_server = "node02.myqtthub.com";   // Ingreso la dirección del broker al que me deseo conectar
const char* Id = "ESP8266";                        // Ingreso la Id del broker al que me deseo conectar
const char* User = "strix0087";                    // Ingreso el nombre de usuario con el que me registre en el broker
const char* CodePass = "draimy123";                // Ingreso la contraseña con la que me registre en el bróker

//-------------------------------------------------OTRAS VARIABLES--------------------------------------------------------

AudioGeneratorWAV *wav;
AudioFileSourcePROGMEM *file;
AudioOutputI2SNoDAC *out;

WiFiClient espClient;
PubSubClient client(espClient);
String _topic;
String _payload;

String strPulsador;                                // Indica si el amplificador esta apagado o encendido
String strvumetro;                                 // Indica si el amplificador esta apagado o encendido
int newButtonState= 0;                             // Indica el estado actual del botón de encendido y apagado del
amplificador
int oldButtonState= 0;                             // Indica el estado anterior del botón de encendido y apagado
delamplificador
int ampliState= 1;                                 // Indica el encendido y apagado del amplificador

int Vo;                                            // Es la tensión leída en la entrada A0
float R1 = 100;                                    // Indica el valor la resistencia  en serie a la resistencia NTC
float logR2, R2, TEMPERATURA;                      // R2 es el valor la resistencia NTC y temperatura la variable a medir
float c1 = 2.478490884e-03, c2 = 1.623993107e-04 ; // Constantes de steinhart-hart
float c3 = 13.07015263e-07;
unsigned long previousMillis = 0;                  // Esta variable reinicia el tiempo de la lectura de la temperatura a 0
String strventilador;                              // Indica si el ventilador esta apagado o encendido
```

```
//-------------------------------------------CONEXIÓN AL WIFI-----------------------------------------------------

// Esta Función se encarga de conectarnos a la red wifi

void setup_wifi() {
  delay(10);                                        // Espero 10ms
  Serial.println();                                 // Comenzamos a conectarnos al wifi
  Serial.print("\nConectando a ");                  // Imprimo conectado a
  Serial.println(ssid);                             // imprimo en nombre de la red wifi
  WiFi.begin(ssid, password);                       // Comienzo a conectarme al broker

  while (WiFi.status() != WL_CONNECTED) {           // Mientras no esté conectado al broker
   delay(10);
   audio();                                         // Preparo la reproducción del audio
   PushButton();                                    // Ver si se presionó el botón de encendido
  }
  Serial.println("");                               // Si ya realicé la conexión
  Serial.println("\nConectado al WiFi");            // Imprimir conectado al wifi
  Serial.println("\n Direccion IP : ");             // Imprimo la dirección Ip del dispositivo conectado
  Serial.println(WiFi.localIP());                   //
}


//----------------------------------------ENCENDIDO Y APAGADO DEL AMPLIFICADOR-------------------------------------
// Esta Función se encarga de detectar si se ha presionado el botón de encendido y apagado del amplificador
// y avisar al cliente si se encuentra apagado o encendido.

    void PushButton() {
    newButtonState=digitalRead(D5);                                     // Guardo el valor del pin D5
    if(newButtonState != oldButtonState){                               // Si el valor es distinto al valor anterior
    oldButtonState = newButtonState;                                    // Guardo el nuevo
    if (newButtonState == 0) {                                          // Si el valor nuevo es igual a 0
    ampliState = !ampliState;                                           // Negar el estado (ON/OFF) del amplificador
    if (ampliState == 0) {                                              // Si el estado es 0
    strPulsador = "encendido";                                          // Indico que el amplificador esta encendido
    file = new AudioFileSourcePROGMEM( encendido, sizeof(encendido) );  // Ejecuto el audio de encendido
    out = new AudioOutputI2SNoDAC();                                    //
    wav = new AudioGeneratorWAV();                                      //
    wav->begin(file, out);                                              //
}
    else {
    strPulsador = "apagado";                                            // Sino indico que el amplificador esta apagado
}
    digitalWrite(D7, ampliState);                                       // Encendemos o apagamos el amplificador
    client.publish("pulsador",String(strPulsador).c_str());             // Aviso al cliente si el amplificador está encendido
    Serial.println("\nEnviando: ");                                     // Imprimo que se envió el mensaje al cliente
    Serial.println(strPulsador);
    delay(1000);
  }
  }
}
```

```
//-----------------------------------------------------CONECTAR AL BROKER-----------------------------------------------------
// Se ejecuta al perder la conexión con el broker y se encarga de reconectarnos al mismo.

void reconnect() {
  while (!client.connected()) {                                  // Esta función se ejecuta cada vez que no estamos conectados
                                                                 // al servidor MQTT
    Serial.println("\nIntentando conectar al servidor MQTT... "); // Mientras no se ejecute, imprimir  Intentando conectar al
                                                                 // servitor MQTT...
    if (client.connect(Id,User,CodePass)) {
    } else {
      audio();                                                   // Sino preparar el audio de encendido
      PushButton();                                              // Ver si se ha presionado el botón de encendido
      delay(10);
    }
      Serial.println("\nconnectado");                            // Si logramos conectarnos indicar que nos conectamos al /
                                                                 // servidor
      client.subscribe("Amplificador");                          // Suscribirnos a al tópico Amplificador
      client.subscribe("Vumetro");                               // Suscribirnos a al tópico Vúmetro
  }
}

//-----------------------------------------------------RECEPCION Y ENVIO DE DATOS-----------------------------------------------
// Se ejecuta al enviar o recibir un mensaje del cliente y le indicará al ESP8266 las instrucciones a seguir

  void callback(char* topic, byte* payload, unsigned int length) {
  String conc_payload_;
  for (int i = 0; i < length; i++) {
    conc_payload_  += (char)payload[i];
  }
  _topic = topic;                                                // Permite utilizar tópico dentro del bucle loop
  _payload = conc_payload_;                                      // Permite utilizar el payload dentro del bucle loop


    if (_topic == "Amplificador") {                               // Si el tópico es Amplificador
    if (_payload == "encender") {                                 // Y el payload es encender
    digitalWrite(D7, LOW);                                        // Encender amplificador
    ampliState=LOW;                                               // Indico que se cambió el estado del amplificador
    delay(1000);                                                  // Espero 1 segundo
    file = new AudioFileSourcePROGMEM( encendido, sizeof(encendido) );  // Ejecuto el audio de encendido
    out = new AudioOutputI2SNoDAC();                              //
    wav = new AudioGeneratorWAV();                                //
    wav->begin(file, out);                                        //
  }

    if (_payload == "apagar") {                                   // Si el payload es apagar
    digitalWrite(D7, HIGH);                                       // Apagar amplificador
    ampliState=HIGH;                                              // Indico que se cambió el estado del amplificador
  }

    if (ampliState == 0) {                                        // Si el amplificador esta encendido
    strPulsador = "encendido";                                    // variable srtpulsador igual a encendido
  }
    else {
    strPulsador = "apagado";                                      // Sino variable srtpulsador igual apagado
  }
    client.publish("pulsador",String(strPulsador).c_str());       // Aviso al cliente si el amplificador está encendido
    Serial.println("\nEnviando: ");                               // Imprimo que se envió el mensaje al cliente
    Serial.println(strPulsador);
```

```
  Serial.print("\nmensaje recibido de [");                    // Indico que se recibió un mensaje
  Serial.print(_topic);
  Serial.println("] ");
  Serial.println(_payload);
 }


  if (_topic == "Vumetro") {                                  // Si el tópico es Vúmetro
  if (_payload == "encender") {                               // Y el payload es encender
  digitalWrite(D6, HIGH);                                     // Encender Vúmetro
  strvumetro = "encendido";                                   // Indico que se cambió el estado del amplificador
  }
  if (_payload == "apagar") {                                 // Si el payload es apagar
  digitalWrite(D6, LOW);                                      // Encender Vúmetro
   strvumetro = "apagado";                                    // Sino variable srtpulsador igual no presionado
  }

  client.publish("vumetro/on/off",String(strvumetro).c_str()); // Aviso al cliente si el vúmetro esta encendido
  Serial.println("\nEnviando: ");                             // Imprimo que se envió el mensaje al cliente
  Serial.println(strPulsador);

  Serial.print("\nmensaje recibido de [");                    // Indicar que se recibió un mensaje
  Serial.print(_topic);
  Serial.println("] ");
  Serial.println(_payload);
  }
}

//-----------------------------------------SETUP---------------------------------------------------------
// Se ejecuta al inicio del programa y se encarga de configurar los pines del ESP8266 conectarnos a la red wifi,
// conectarnos al broker y comenzar el envio y recepción de datos

void setup() {
 pinMode(D2, OUTPUT);                                         // Configuramos D2  como salida digital
 digitalWrite(D7, LOW);                                       // Apagamos el ventilador
 pinMode(D7, OUTPUT);                                         // Configuramos D7  como salida digital
 digitalWrite(D7, HIGH);                                      // Apagamos el amplificador
 pinMode(D6, OUTPUT);                                         // Configuramos D6 como salida digital
 digitalWrite(D6, LOW);                                       // Apagamos el vúmetro
 pinMode(D5, INPUT);                                          // Configuramos el pin D5 como entrada
 digitalWrite(BUILTIN_LED, HIGH);                             // Apagamos el led que viene por defecto en el /
                                                              //ESP8266
 Serial.begin(115200);                                        // Configuramos el puerto serie  115200 baudios
 delay(1000);

 audioLogger = &Serial;                                       // Indico las variables para ejecutar el audio
 file = new AudioFileSourcePROGMEM( encendido, sizeof(encendido) );  //
 out = new AudioOutputI2SNoDAC();                             //
 wav = new AudioGeneratorWAV();                               //

 setup_wifi();                                                // Llamamos a la función setup_wifi para
                                                              //conectarnos a la red wifi
 client.setServer(mqtt_server, 1883);                         // Nos conectamos al servidor MQTT
 client.setCallback(callback);                                // Inicia el callback para recibir y enviar un
                                                              //mensaje
}
```

```
//-----------------------------------------------------------------audio-----------------------------------------------------------
// Esta función debe ejecutarse justo antes de ejecutar un audio y se encarga de parar la ejecución del audio
// una vez finalizado su reproducción

void audio()
{
  if (wav->isRunning()) {
   if (!wav->loop()) wav->stop();
  } else {
   audioLogger = &Serial;
  }
}


//-------------------------------------------------------------------------LOOP-------------------------------------------------------

void loop() {

  if (!client.connected()) {
    reconnect();                                                    // Si el cliente no está conectado ejecutar la función
                                                                    // reconect
  }
  client.loop();                                                    // Si está conectado ejecutar bucle
  audio();
  PushButton();                                                     // Ver si se a presionado el botón de encendido

  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= 4000) {                     // Envía la temperatura solo si han pasado 4ms
  previousMillis = currentMillis;                                   // Indico que ya pasaron 10 segundos
  Vo = analogRead(A0);                                              // Lectura de A0
  R2 = R1 * (1023.0 / (float)Vo - 1.0);                             // Conversión de tensión a resistencia
  logR2 = log(R2);                                                  // Logaritmo de R2 necesario para ecuación
  TEMPERATURA = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2));     // Igualo la ecuación steinhart-hart a la variable
temperatura
  TEMPERATURA = TEMPERATURA - 273.15;                               // Convierto de Kelvin a Centígrados
  if (TEMPERATURA >= 45) {                                          // Si la temperatura es mayor o igual a 45°
  digitalWrite(D2, HIGH);                                           // Enciendo el ventilador
  strventilador = "encendido";                                      // Indico que se cambió el estado del ventilador
  }
  else {
  digitalWrite(D2, LOW);                                            // Apago el ventilador
  strventilador = "apagado";                                        // Indico que se cambió el estado del ventilador
  }
  client.publish("ventilador",String(strventilador).c_str());       // Aviso al cliente que hay un sobrecalentamiento
  Serial.println("\nEnviando: ");                                   // Imprimo que se envió el mensaje al cliente
  Serial.println(strventilador);

  client.publish("\ntemperatura",String(TEMPERATURA).c_str());      // Envió el valor de temperatura al cliente
  Serial.print(F("\nenviando temperatura a "));                     // Indico que el valor de temperatura se envió
  Serial.print(TEMPERATURA);
  Serial.println("...");
  delay(10);                                                        // Esperar 10ms
  }
}
```

DIAGRAMA DE FLUJO DEL PROGRAMA FINAL

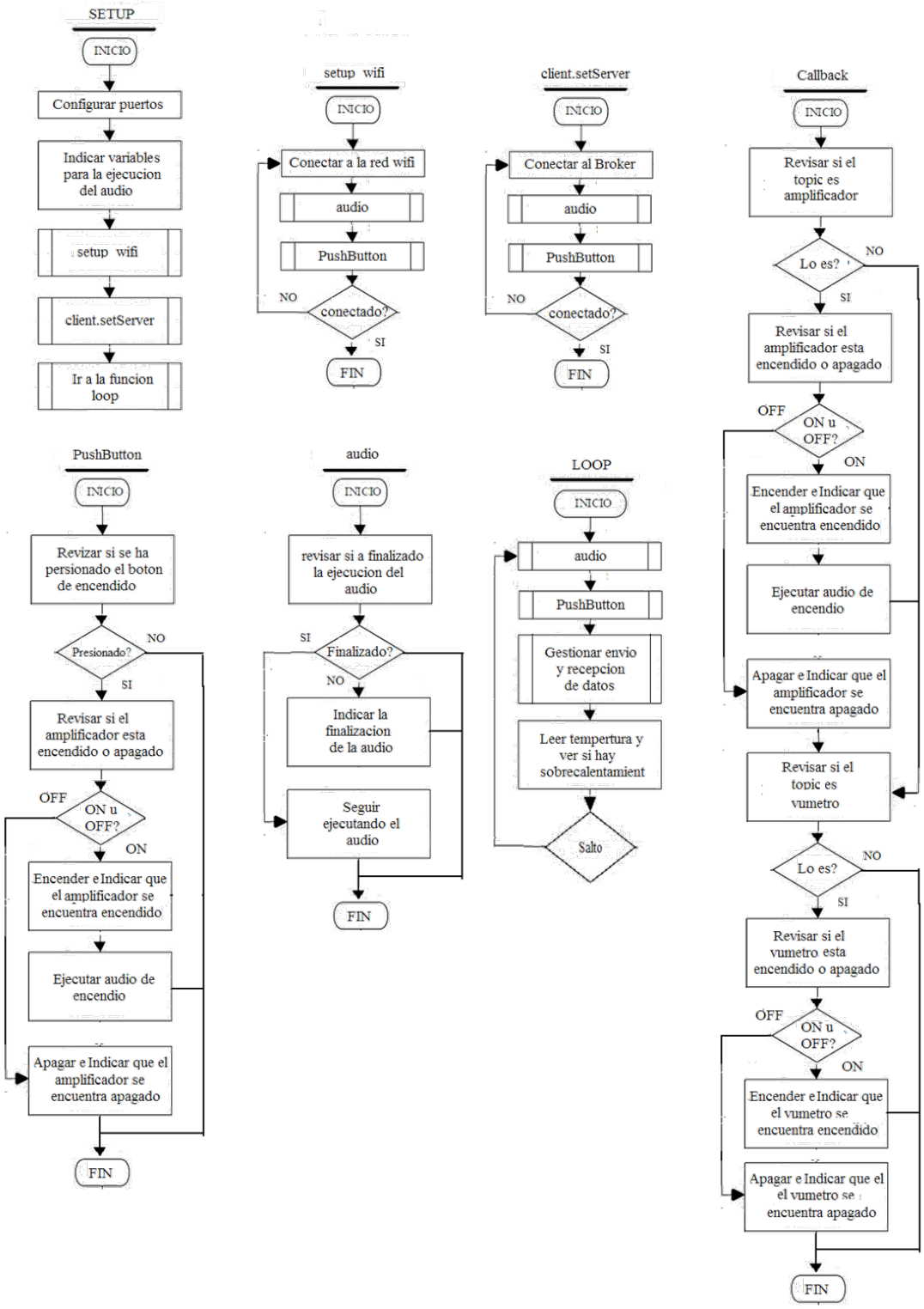

Figura 59. Diagrama de flujo del programa final – Imagen de autoría propia

DIAGRAMA DEL CIRCUITO FINAL DEL PROYECTO

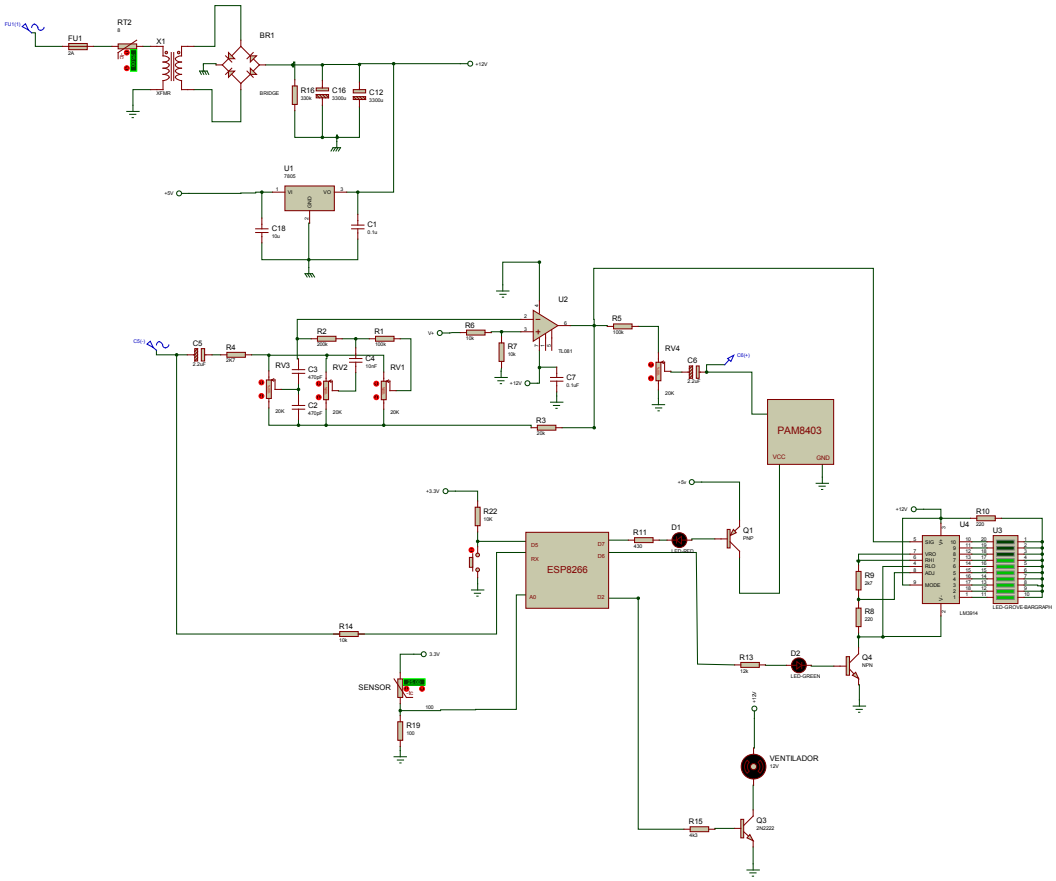

Figura 60. Diagrama del circuito final del proyecto – Imagen de autoría propia utilizado el software de proteus

PROTOTIPO FINAL

En la figura de abajo se muestra el prototipo final, montado en el protoboard:

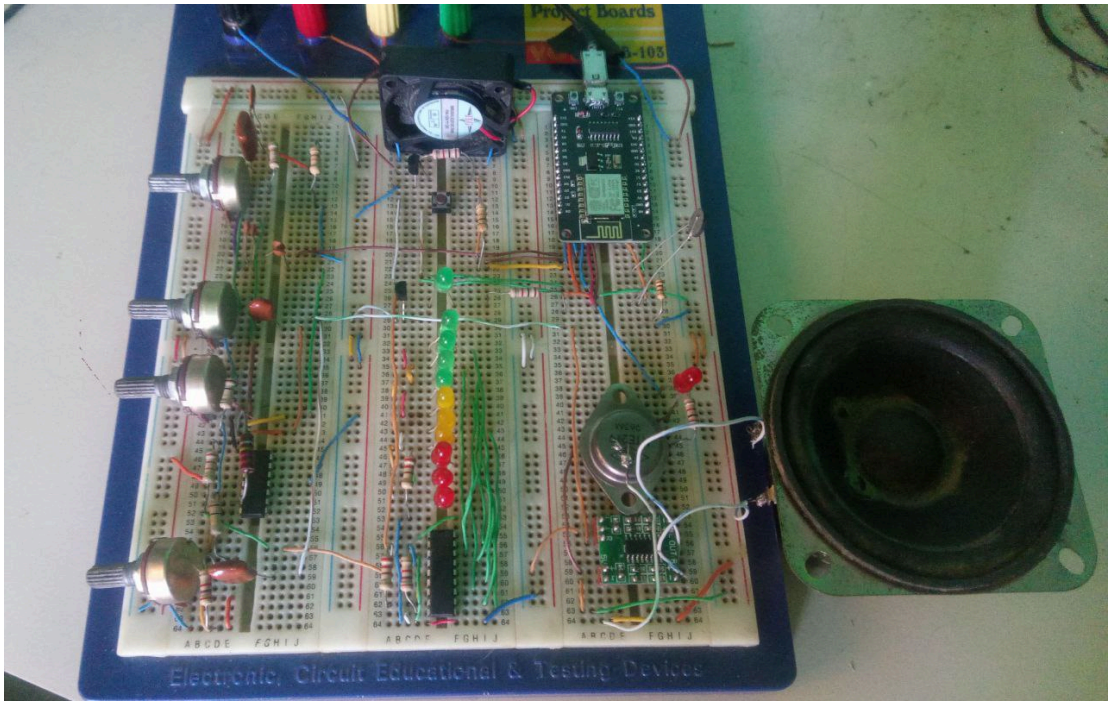

Figura 61. Prototipo final– Imagen de autoría propia

En el enlace de abajo se muestra el funcionamiento del prototipo final:

https://drive.google.com/file/d/16fFqcBq9rB1Fyi92lzwNm0zpvFEQhDhh/view?usp=drivesdk

PLANIFICACION DEL PROYECTO

En la figura de abajo se muestra los pasos seguir en para la ejecución del proyecto y las fechas establecidas, hasta su finalización.

Nombre	Duracion	Inicio	Terminado
Definicion y planificacion del proyecto	4 days	**11/8/22 8:00 a.m.**	**11/11/22 12:00 p.m.**
Discusión de Proyecto	1 day	11/8/22 8:00 a.m.	11/8/22 12:00 p.m.
Planificacion del proyecto utilizando Project libre	4 days	11/8/22 8:00 a.m.	11/11/22 12:00 p.m.
Investigación Proyectos de Referencia	4 days	11/8/22 8:00 a.m.	11/11/22 12:00 p.m.
Investigacin de los lenguajes de programación a utilizar	4 days	11/8/22 8:00 a.m.	11/11/22 12:00 p.m.
Practica con el lenguaje de programacion a utilizar	4 days	11/8/22 8:00 a.m.	11/11/22 12:00 p.m.
Diseño del prototipo	28 days	**11/14/22 8:00 a.m.**	**12/21/22 12:00 p.m.**
Hardware	16 days	**11/14/22 8:00 a.m.**	**12/5/22 12:00 p.m.**
Diseño y simulación la fuente de poder lineal	5 days	11/14/22 8:00 a.m.	11/18/22 12:00 p.m.
Diseño y simulacón del regulador para el ESP8266	5 days	11/14/22 8:00 a.m.	11/18/22 12:00 p.m.
Diseño y simulaón del pre amplificador con ecualizador	5 days	11/14/22 8:00 a.m.	11/18/22 12:00 p.m.
Diseño y simulación del vumetro	5 days	11/14/22 8:00 a.m.	11/18/22 12:00 p.m.
Montaje y prueba del pre amplificador	1 day	11/22/22 8:00 a.m.	11/22/22 12:00 p.m.
Realizacion del primer informe del avance del proyecto	4 days	11/21/22 8:00 a.m.	11/24/22 12:00 p.m.
Investigacion sobre como programar el ESP8266	4 days	11/21/22 8:00 a.m.	11/24/22 12:00 p.m.
Entrega del primer avance	1 day	11/25/22 8:00 a.m.	11/25/22 12:00 p.m.
Prueba del peamplificador con equalizador en conjunto	1 day	11/29/22 8:00 a.m.	11/29/22 12:00 p.m.
Realizacion del segundo informe del avance delproyect	5 days	11/28/22 8:00 a.m.	12/2/22 12:00 p.m.
Entrega del segundo avance	1 day	12/5/22 8:00 a.m.	12/5/22 12:00 p.m.
software	9 days	**12/6/22 8:00 a.m.**	**12/16/22 12:00 p.m.**
Prueba del ESP8266 en el laboratorio	1 day	12/6/22 8:00 a.m.	12/6/22 12:00 p.m.
Realizacion del tercer informe del avance del proyecto	3 days	12/6/22 8:00 a.m.	12/8/22 12:00 p.m.
Diseño del programa para el control del amplificador	4 days	12/7/22 8:00 a.m.	12/12/22 12:00 p.m.
Entrega del tercer avance	1 day	12/9/22 8:00 a.m.	12/9/22 12:00 p.m.
Preba deL control inalambrico del amplificador	1 day	12/13/22 8:00 a.m.	12/13/22 12:00 p.m.
Diseño y simulacion del circuito sensor de temperatura	5 days	12/12/22 8:00 a.m.	12/16/22 12:00 p.m.
Diseño del programa para el control de temperatura	2.5 days	12/12/22 8:00 a.m.	12/14/22 10:00 a.m.
Informe del proyecto y presentación	28 days	**11/14/22 8:00 a.m.**	**12/21/22 12:00 p.m.**
Realizacion del informe final	27 days	11/14/22 8:00 a.m.	12/20/22 12:00 p.m.
Preuba final del prototipo y Presentacion al profesor	0 days	12/20/22 12:00 p.m.	12/20/22 12:00 p.m.
Entrega del informe final	1 day	12/21/22 8:00 a.m.	12/21/22 12:00 p.m.

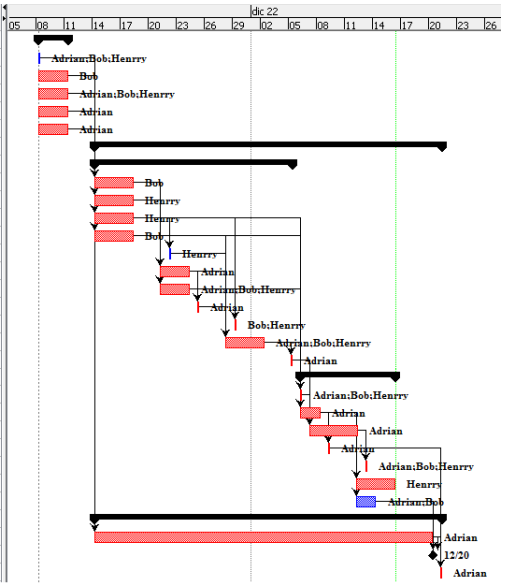

Figura 62. Diagrama de GANTT – Imagen de autoría propia utilizado Project Libre

CONCLUSION

En el presente informe desarrollamos por completo el prototipo del amplificador de audio con control inalámbrico, en las fechas establecidas en el plan de acción, durante las ejecución del mismo se fueron cometiendo errores, los cuales se fueron solucionando conforme se avanzaba en la realización del proyecto, entre los que destacan el fallo en los condensadores de filtrado del ecualizador, lo cual impedía la variación en la atenuación del ecualizador para todas las frecuencias, como se mencionó anteriormente este fallo fue solventado cambiando los condensadores de filtrado, y posteriormente se demostró su funcionamiento mediante la segunda prueba de respuesta en frecuencia del ecualizador.

Otro fallo fue que durante una de las pruebas de la ejecución del audio para el encendido del ESP8266, por accidente se quemó el pin rx del ESP8266, esto ocurrió al dejar el pin rx al aire, el cual estaba conectado a un cable, el extremo de este cable por un pequeño instante tocó accidentalmente una resistencia conectada a 12v lo cual quemó directamente el pin rx, esto se solvento comprando otro ESP8266, una vez resulto este último inconveniente, no hubo más fallos a la hora de montar el prototipo, se pudo completar el programa para el ESP8266 añadiendo el control para la temperatura, se mostró el programa finalizado, el circuito finalizado, el prototipo finalizado y se mostro mediante un video el funcionamiento del prototipo.

REFERENCIAS CONSULTADAS

Acadenas, A. (2020). *SMPS. Como funciona circuito de proteccion de entrada y filtro EMI.* Obtenido de www.youtube.com: https://www.youtube.com/playlist?list=PLb_ph_WdlLDny2cGloFSxyRgO8B733jeo

Amplificadores-de-potencia-de-audio. (s.f.). Obtenido de https://docplayer.es/68574182-Amplificadores-de-potencia-de-audio.html

Bitwise, A. (2018). *Arduino desde cero n español- Capitulo 34.* Obtenido de youtube.com: https://youtu.be/8Wry8lwgGtA

datasheet.es. (s.f.). *PAM8403*. Obtenido de datasheet.es: datasheet.es/PDF/642169/PAM8403-pdf.html

Floyd, T. (2008). *Dispositivos Electronicos.* Obtenido de https://latecnicalf.com.ar/descargas/material/electronicaanalogica/Dispositivos%20Electronicos%208va.edicion-%20Floyd.pdf

Granados, C. (2017). *Programando directamente un ESP8266.* Obtenido de https://upcommons.upc.edu/bitstream/handle/2117/105042/Memoria_TFG_Carles_Ubach.pdf?sequence=1&isAllowed=y

kitelectronica.com. (2015). *Tutorial circuito integrado LM3914 LM3915.* Obtenido de https://www.kitelectronica.com/2016/02/tutorial-ic-lm3914-lm3915.html?m=1

Llamas, L. (2021). *ENVIAR Y RECIBIR MENSAJES POR MQTT CON ARDUINO Y LA LIBRERÍA PUBSUBCLIENT.* Obtenido de https://www.luisllamas.es/enviar-y-recibir-mensajes-por-mqtt-con-arduino-y-la-libreria-pubsubclient/

Muños, J. (s.f.). *Protocolo MQTT.* Obtenido de www.academia.edu: https://www.academia.edu/42163136/Protocolo_MQTT_JUMAMUCA_at_GMAIL_COM

Preamplificador-con-eq-de-3-bandas-y-fuente-simple. (s.f.). Obtenido de videorockola.com: https://www.videorockola.com/proyectos-electronicos/car-audio/preamplificador-con-eq-de-3-bandas-y-fuente-simple/

Rose, k. (2015). *Internet de las cosas- breve reseña.* Obtenido de https://www.internetsociety.org/wp-content/uploads/2017/09/report-InternetOfThings-20160817-es-1.pdf

Salina, D. (2019). *Sistema de amplificación HI-RES con transmisión inalámbrica de audio empleando el protocolo PurePath Wireless.* Obtenido de https://repositorio.uta.edu.ec/jspui/bitstream/123456789/30705/3/Tesis_t1666ec.pdf

Teis, R. (2021). *Asignación de pines de NodeMCU y asignación de pines de ESP-12E | Pin Descripción, Periféricos.* Obtenido de https://www.electronicshub.org/nodemcu-pinout-esp-12e-pinout/#ESP-12E_Modulehttps://www.electronicshub.org/nodemcu-pinout-esp-12e-pinout/#ESP-12E_Module

Zorzano, A. (2004). *Amplificador de audio de alta fidelidad para sistemas activos de altavoces con bajo consumo de energia.* Obtenido de https://dialnet.unirioja.es/descarga/tesis/20788.pdf