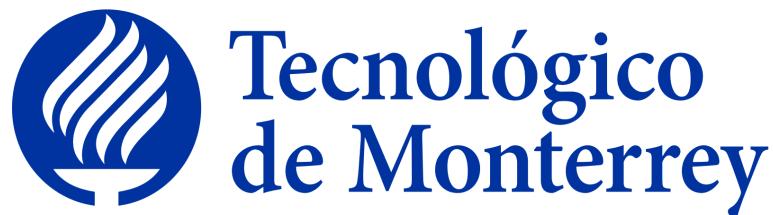


Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus, Querétaro



***Diseño de sistemas embebidos avanzados
(Gpo 501)***

Actividad

Reporte final del reto

Estudiantes:

Esteban Padilla Cerdio	A01703068
Karen Cebreros López	A01704254
Hilda Olivia Beltrán Acosta	A01251916
Aranza Leal Aguirre	A01751706

Profesores:

José Antonio Cantoral
Pedro Perez
Oscar Hernández Uribe

Fecha de entrega:
Viernes 2 de diciembre del 2022

Índice:

Introducción	2
Objetivos	2
Desarrollo	2
Resultados	5
Conclusiones	5
Referencias	5

Introducción:

Para la materia de “Diseño de sistemas embebidos avanzados”, se nos dejó como reto de la materia, realizar un sistema que a través del intercambio de información, entre cliente-servidor, pueda hacer una comparativa de firmas de audio.

Utilizando eso, se busca poder determinar el camino tomado por un vehículo en una bifurcación y el tipo de vehículo que pasó, dependiendo de la comparativa de las frecuencias, del sonido emitido por éste y una firma de sonido que se encontrará ya en el microcontrolador.

Objetivos:

Este proyecto tiene como objetivo, poder poner en práctica los conocimientos adquiridos en los módulos de la materia. Esto incluye lo que es el diseño de un sistema, el procesamiento de señales y la conexión entre cliente-servidor para intercambio de información.

Se espera que para el final del bloque, se pueda demostrar la validación del sistema previamente hecha por el equipo, para mostrar no solo su proceso, sino también los resultados obtenidos.

Desarrollo:

Para el desarrollo de nuestro proyecto, utilizamos los siguientes componentes electrónicos:

- Raspberry pi



Es una computadora compacta, de bajo costo.

Esta contiene varios pines de entrada y de salida, que permiten al consumidor desarrollar diversos proyectos, utilizando el propio sistema operativo que trae consigo esta placa.

- ESP32



Es un módulo WiFi y Bluetooth.

Este componente facilita el diseño de proyectos, ya que puede conectarse a una gran variedad de interfaces externas y cuenta con dos núcleos de procesamiento.

- ky-037



El ky-037 es un sensor de sonido, que cuenta tanto con salida analógica, como digital.

El sensor cuenta con una sensibilidad ajustable, para la detección del sonido.

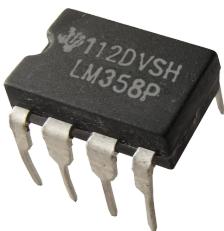
- Arduino nano



Es un microcontrolador compacto, de fácil uso para el desarrollo, que cuenta con pines de entrada analógica y pines de salida digital.

Esta placa, cuenta con un procesador AT328P.

- Op-amp (LM358)



Este circuito integrado, también conocido como amplificador operacional, tiene varias funcionalidades tales como: filtros (paso altos y bajos), sumadores, amplificadores y demás.

Está hecho con dos op-amps de alta ganancia.

Como primer acercamiento para el diseño de nuestro sistema como solución del reto, empezamos utilizando nuestra raspberry pi, 2 ESP32 y tres micrófonos.

Sin embargo, tras tratar de hacer unas cuantas pruebas de sonido, nos percatamos que la calidad y captación de sonido de los micrófonos con los que contábamos, no era muy buena. Por ello decidimos utilizar un amplificador operacional, para que a través de una cierta ganancia, pudiéramos obtener mejores valores de los micrófonos y ahora si poder trabajar con los sonidos y sus frecuencias.

Diseñamos dos sistemas diferentes. Uno para la raspberry pi y otro para los nodemcu:

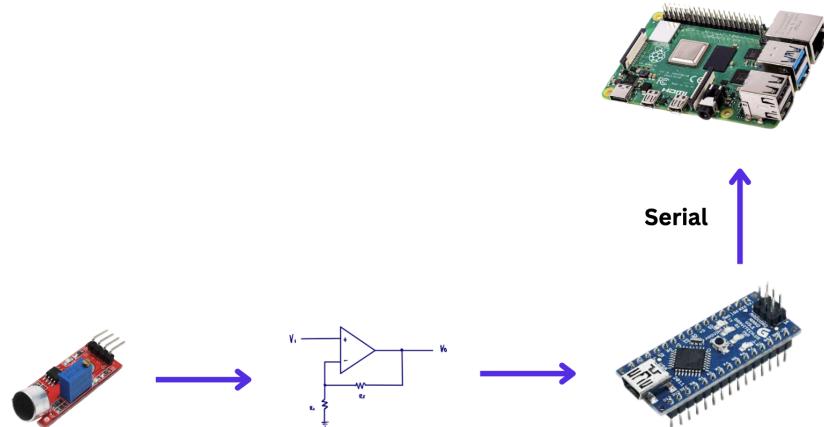


Figura 1. Diagrama de conexión entre la raspberry pi, el arduino nano y el ky-037.

En este primer diseño, podemos observar que también utilizamos el arduino nano como adc, ya que no contábamos con el módulo original visto en clase.

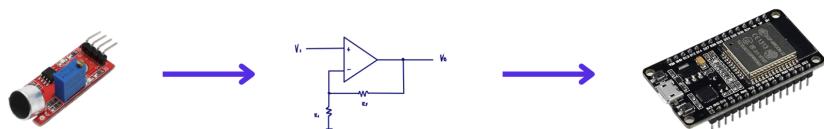


Figura 2. Diagrama de conexión entre el ESP32 y el ky-037.

El segundo diseño, lo utilizamos para ambos circuitos de cada camino de la bifurcación.

Para el amplificador operacional utilizado, primero diseñamos un circuito de ganancia variable, con el objetivo de poder ajustarla fácilmente, dependiendo del micrófono:

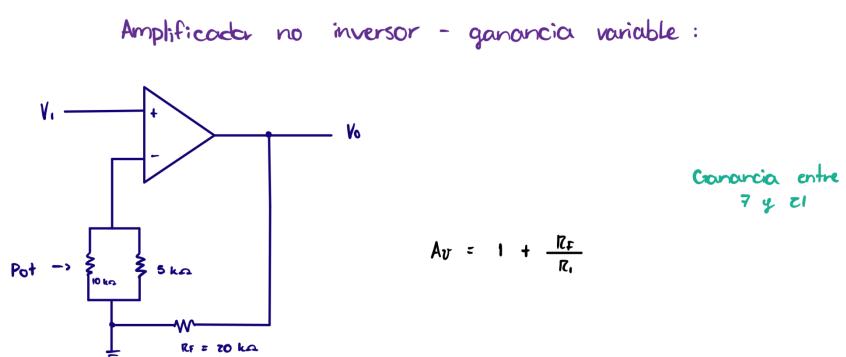


Figura 3. Op-amp de ganancia variable.

No obstante, al realizar unas cuantas pruebas, nos dimos cuenta que no iba a ser necesario y por ello, para simplificar aún más nuestro circuito, decidimos hacer un segundo diseño, pero esta vez para tener una ganancia fija; así como se puede apreciar en la segunda imagen.

Para el circuito de ganancia fija, ocupamos un valor de $10\text{ k}\Omega$ para la R_f y $1\text{ k}\Omega$ para la R_i .

Amplificador no inversor - ganancia fija :

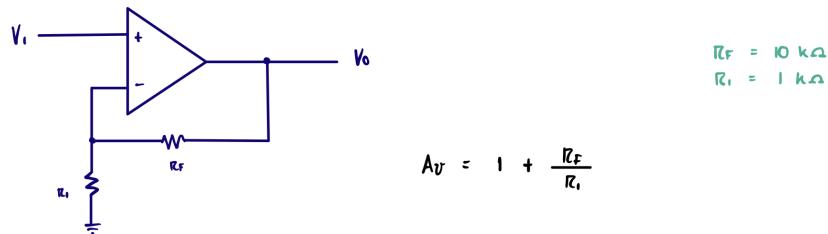


Figura 4. Op-amp de ganancia fija.

Finalmente, al juntar ambos circuitos mencionados anteriormente (y agregando algunos componentes extras, como leds y unas cuantas resistencias de diversos valores), nos quedó así nuestro bosquejo final del circuito:

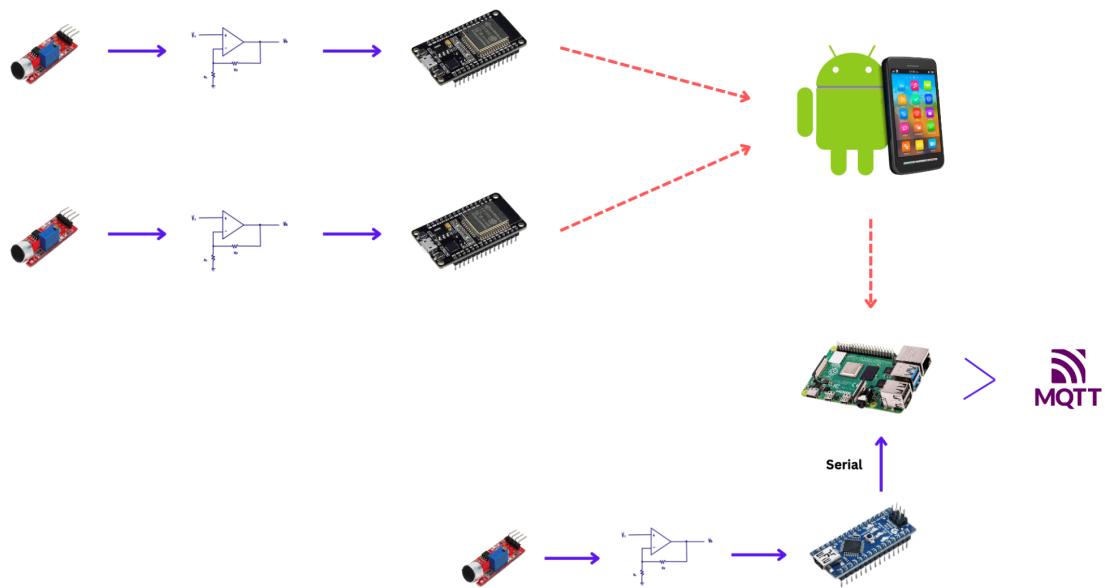


Figura 5. Diseño final del sistema.

Por el lado de las herramientas computacionales con las que hicimos el intercambio de información entre cliente-servidor y los programas para el análisis de señales, fueron las siguientes:

- Arduino IDE



Es un entorno de desarrollo integrado, que trabaja con Arduino.

Este contiene numerosas librerías para el uso de múltiples placas y sensores, con las que se facilita el desarrollo de proyectos.

- Mosquitto



Es un broker con protocolo de Mqtt, para el servicio de mensajería; lo cual se hace a través de un método de suscripción y publicación.

Por su apertura gratuita y fácil uso, es una herramienta altamente utilizada para el transporte ligero de información.

Como básicamente se menciona en la parte de arriba, utilizamos el broker de Mosquitto para poder realizar el intercambio de información de los módulos ESP32 a la raspberry.

Diseñamos un diagrama, para explicar de una manera más visual, como es que funciona el intercambio de información, a través de un broker mqtt:

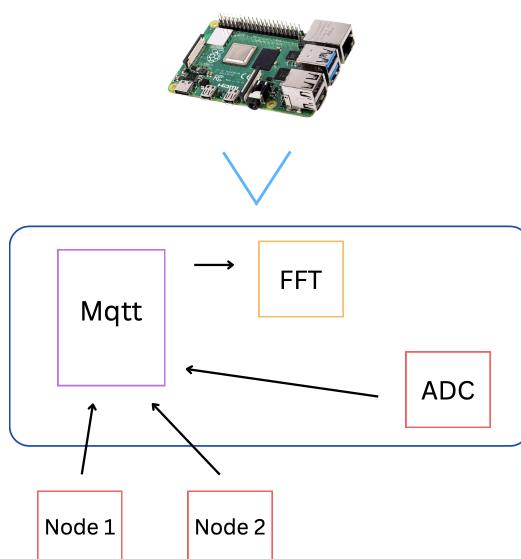


Figura 6. Diagrama del funcionamiento del Mqtt en nuestro sistema.

Archivo client.py:

Primero, importamos las librerías necesarias para el funcionamiento del código.

```
from scipy.signal import butter, sosfiltfilt
from scipy.fftpack import fft, ifft, fftshift
from numpy import pi, cos, sin, convolve
import paho.mqtt.client as mqtt
import matplotlib.pyplot as plt
import time
import numpy as np
```

Nos suscribimos por medio de la función `on_connect`, la cual nos permite suscribirnos a ambos ESP32 y Raspberry. Al momento de que la conexión se pierda, la suscripción se renovará una vez que nos conectemos de nuevo. Utilizamos la función Hamming de NumPy para realizar la multiplicación de la señal con la ventana en el tiempo, y regresar el valor absoluto de la transformada rápida de fourier después de obtener el producto. Se calcula el valor máximo de frecuencia en la señal filtrada, para proceder a comparar las señales y encontrar coincidencias en los vehículos que pasan por ambos caminos. Para mandar el mensaje, utilizamos dos bytes, de los cuales solo utilizamos 10 bits para guardar los datos y el bit número 16 es utilizado para notificar que pasó un segundo y la Raspberry ya puede leer los datos.

Archivo read_mic.py

Librerías a importar.

```
import serial
import paho.mqtt.client as mqtt
import time
```

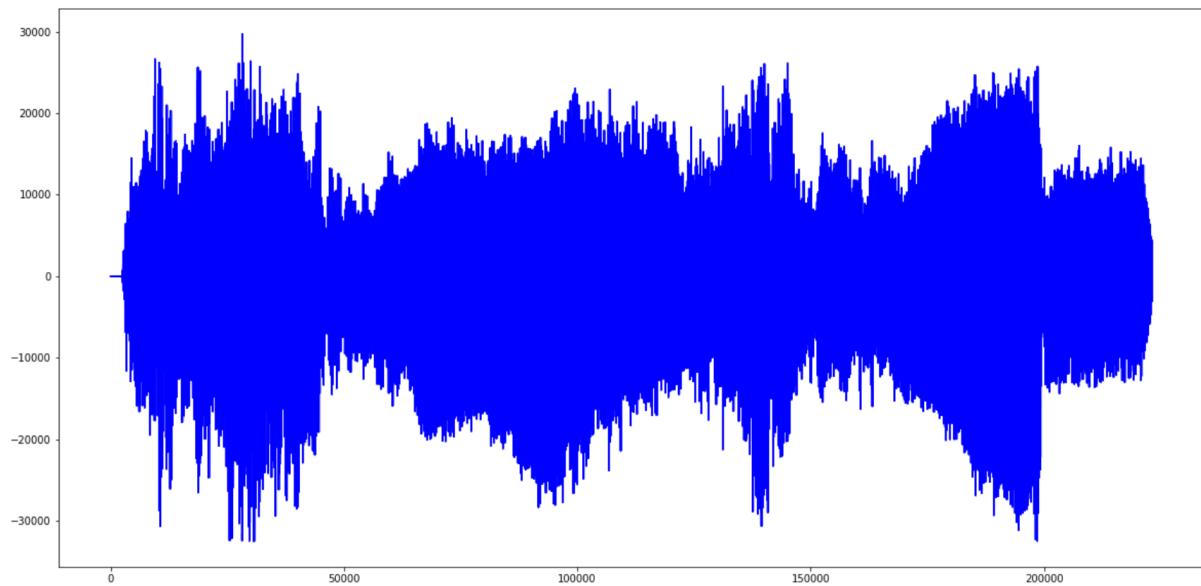
Nos conectamos por mqtt para publicar los datos de las lecturas de los micrófonos. Va a estar leyendo los datos del micrófono, desde el puerto serial, cada segundo, en cuanto se cumpla el segundo se publicarán 16 bits, donde solo el último estará encendido. De lo contrario, se publicarán 16 bits, donde los primeros 10 representan los datos leídos.

Archivo embebidos.ino

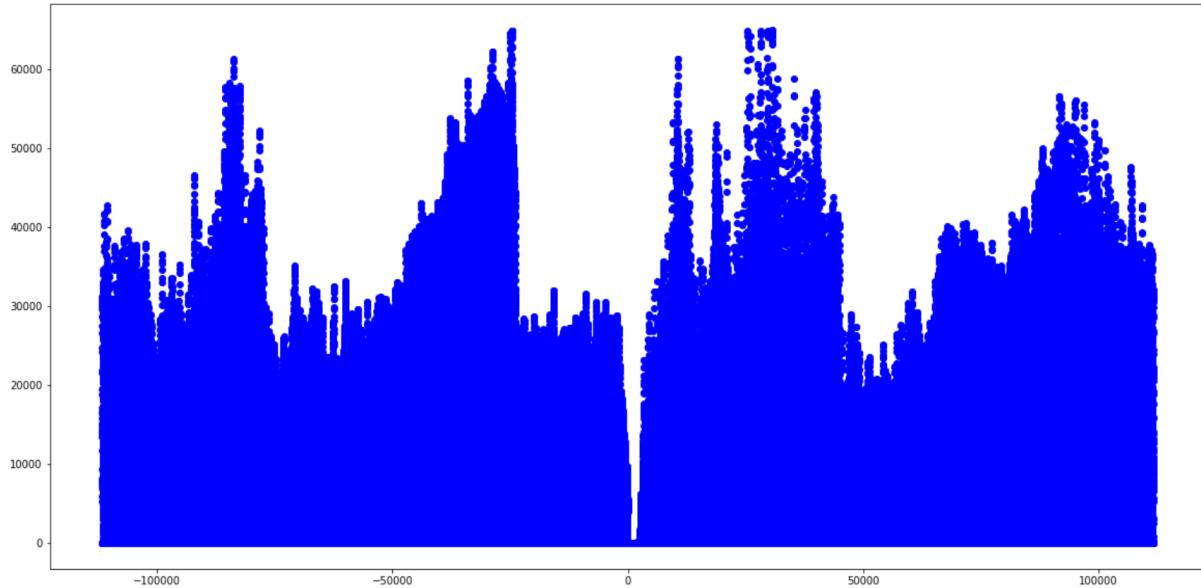
```
#include <ESP8266WiFi.h> //Whe using ESP8266
#include <PubSubClient.h>
```

Definimos el pin en el que estará conectado el micrófono y el valor de la bandera que determina si pasó el segundo o no para mandar los datos. Se realiza la conexión a internet del ESP32 y la conexión al broker Mosquitto, por el protocolo MQTT. Configuramos el puerto serial y el servidor de MQTT, revisamos si la conexión con Mosquitto fue exitosa y comenzamos a leer los datos. Se publican los datos en 10 bits y cuando se llega a 1 segundo se incorpora la bandera donde el bit número 16 está encendido.

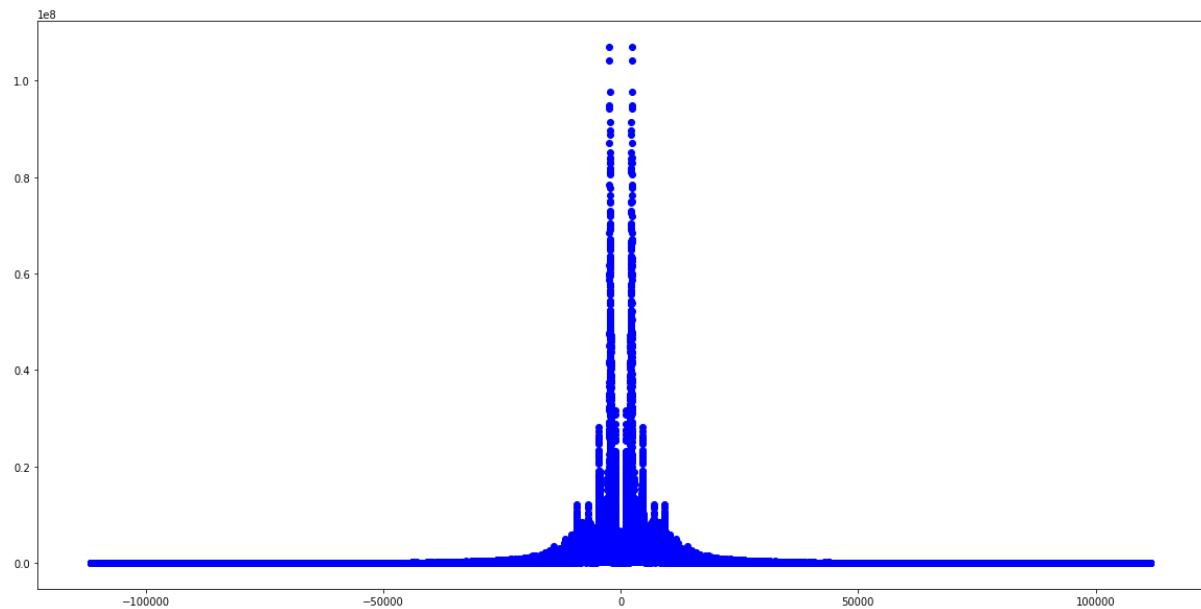
Al recibir la señal en la raspberry, lo primero que hicimos fue graficar los valores de la amplitud en el dominio del tiempo, los cuales se muestran en la siguiente gráfica.



Obtenemos la Transformada Rápida de Fourier para observar la amplitud en el dominio de la frecuencia, lo cual nos permite identificar las frecuencias que causan el ruido en la señal.

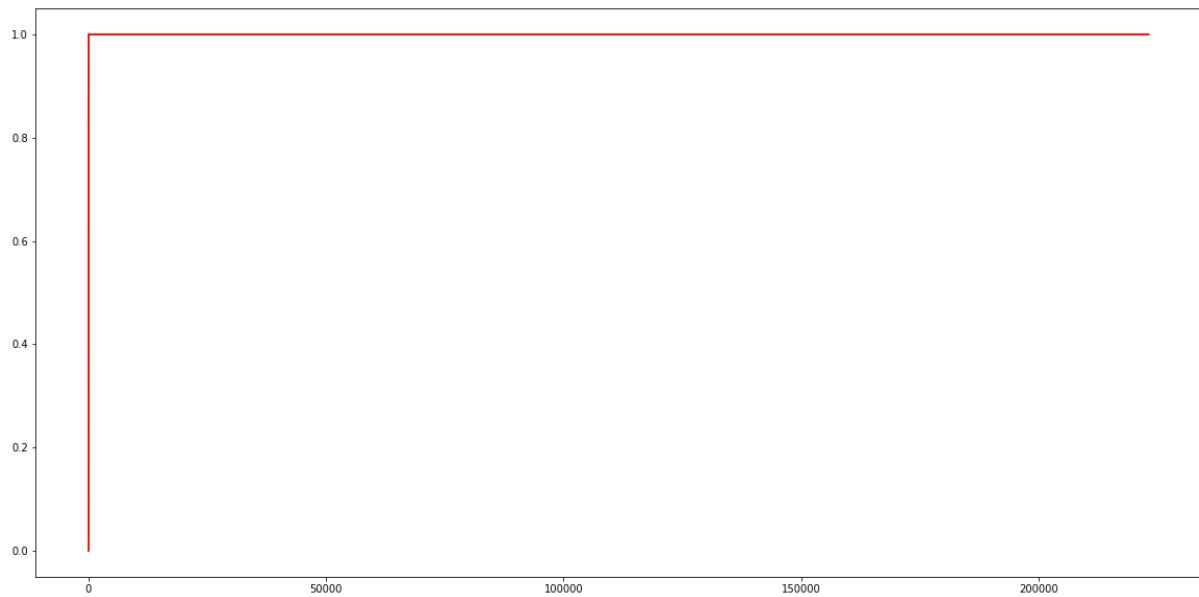


En la gráfica podemos observar la gran cantidad de ruido, ocasionada por el filtrado de algunas señales hacia otras, ocasionando que no podamos apreciar de manera correcta y limpia el audio. Para poder corregir un poco este fenómeno, conocido como Leakage, aplicamos una ventana en el dominio del tiempo. En este caso, optamos por utilizar la ventana de Hamming; para esto, se realizó una multiplicación entre la señal obtenida y la ventana de Hamming en el dominio del tiempo, ya que una multiplicación de dos señales en el dominio del tiempo es igual a una convolución de dos señales en el dominio de la frecuencia. Esta ventana nos permitió obtener una señal más limpia en el dominio de la frecuencia, para así seguir identificando frecuencias que no queremos analizar.

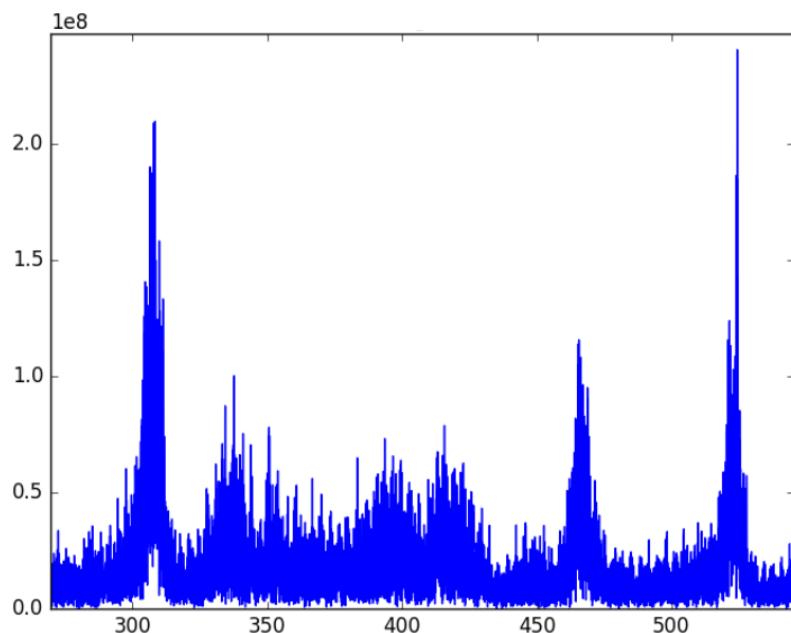


En la gráfica anterior tenemos la amplitud en el dominio de la frecuencia, a la cual se le

aplicó la función ‘fftshift’, que nos permite observar la parte positiva y negativa de la señal centrada en la frecuencia cero. De esta manera, podemos observar que tenemos un pico que no nos interesa, que es ruido que buscamos eliminar, por lo que optamos por utilizar un filtro creado con las funciones de NumPy. Para esto, fue necesario hacer uso de la función ‘ones_like’ con la que pudimos eliminar el valor ubicado en esa frecuencia, obteniendo una señal más limpia.



Con el filtro que se diseñó, que se muestra en la figura anterior, fue posible limpiar la señal y obtener los siguientes valores en el dominio de la frecuencia.



Resultados:



Conclusiones:

A pesar de que el desarrollo de este proyecto fue bastante retador en ciertos aspectos, como equipo creemos que este, nos ayudó a poner en práctica el utilizar varias herramientas (de las cuales ya teníamos un poco de conocimiento previo) en conjunto, para crear un sistema avanzado y cumplir con el objetivo del reto.

Fuimos capaces de superar los obstáculos encontrados a lo largo del desarrollo del proyecto y así mismo, encontrar las áreas de oportunidad en nuestro prototipo, para su posible mejora en un futuro.

Video:

<https://youtu.be/xPGz9ku5zlo>

Referencias:

Aguayo, P. (2019, September 23). Arduino Nano. Arduino.cl - Compra Tu Arduino En Línea. <https://arduino.cl/arduino-nano/>

Beningo, J. (2020, January 21). Cómo seleccionar y usar el módulo ESP32 con Wi-Fi/Bluetooth adecuado para una aplicación de IoT industrial. Digi-Key. <https://www.digikey.com.mx/es/articles/how-to-select-and-use-the-right-esp32-wi-fi-bluetooth-module>

electronics. (2021). LM358N Amplificador Operacional PDIP-8. UNIT Electronics.
<https://uelectronics.com/producto/lm358n-amplificador-operacional-pdip%E2%88%928/>

MCI electronics. (2022, April 26). ¿Que es Raspberry Pi? Raspberry Pi.
<https://raspberrypi.cl/que-es-raspberry/>

UNIT electronics. (2021). Módulo KY-037 Sensor de Sonido. UNIT Electronics.
<https://uelectronics.com/producto/modulo-ky-037-sensor-de-sonido/>

WordPress. (2016, December 11). IDE Arduino.
<https://aprendiendoarduino.wordpress.com/2016/12/11/ide-arduino/>

WordPress. (2021, February 27). Mosquitto.
<https://aprendiendoarduino.wordpress.com/2018/11/19/mosquitto/>