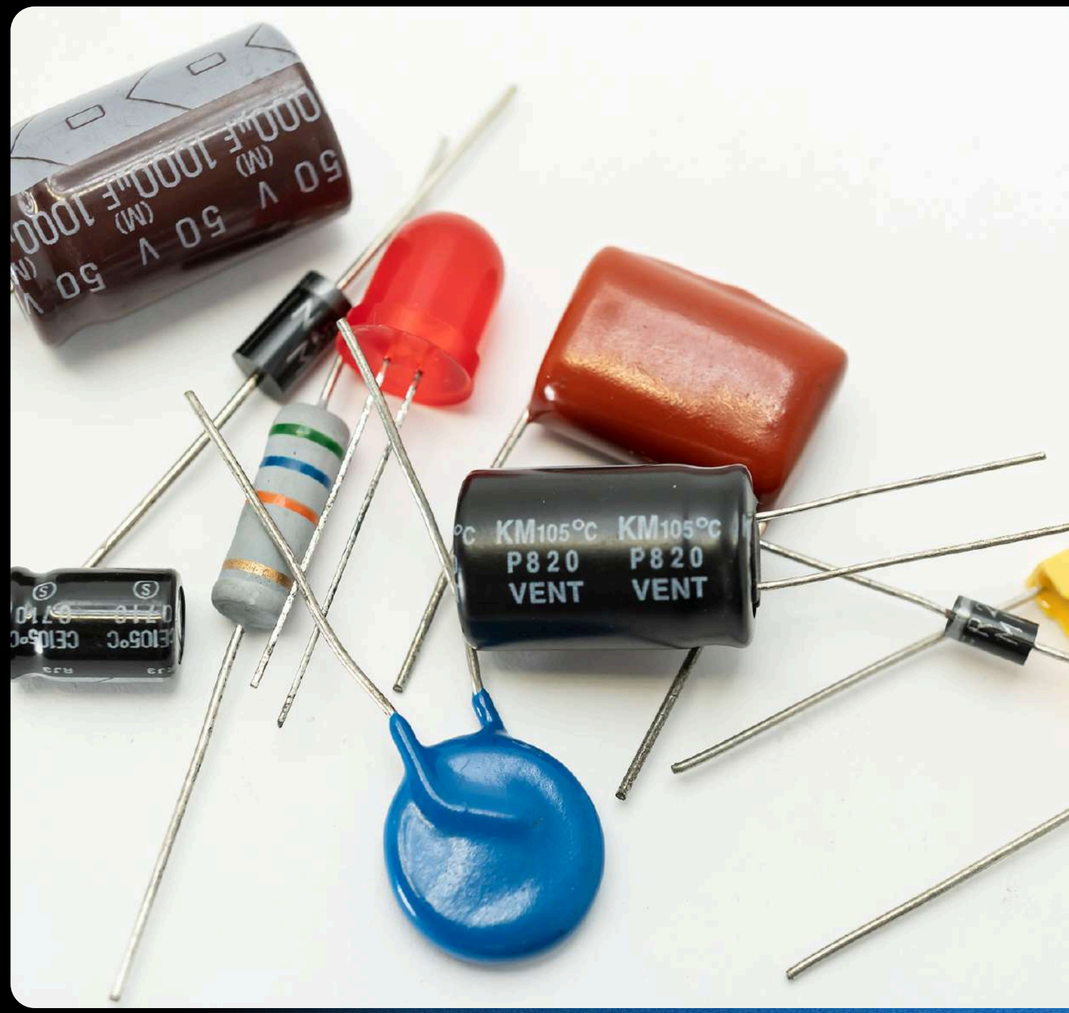
PEDAL CON ARDUINO

Equipo NANA

¿Componentes?

- 1 capacitor de poliéster de 4.7u
- 1 resistencia de 1k
- 1 resistencia de 10k
- 1 resistencia de 1.2k
- 1 resistencia de 150
- 5 resistencias de 5.6k
- 2 sockets para jack 6.3mm
- 1 opamps TL072CP
- 1 ESP32 WROOM-32D
- 1 fuente de voltaje
- 1 amplificador para bajo electrico
- 2 cables tipo Jack 6.3mm macho-macho
- 9 cables tipo caiman
- 1 boton de enganche de 9 pines



Objetivo

Modificar y distorsionar las ondas sonoras así tambien ponemos en práctica el análisis de las mismas usando una interfaz diseñada usando Python como lenguaje de programación, una lectura de datos con Arduino IDE y un ESP32 como receptor de datos.

Códigos

Obtuvimos conocimientos de Python y Arduino IDE, curva logística y amplificadores

```
#V.1.0.0

#Importamos bibliotecas
import pandas as pd #Librería para manipulacion de datos tabulables
import matplotlib.pyplot as plt #Libreria para graficacion de datos

datos=pd.read_csv('draft7.csv') #Abrir el archivo de datos
#Lee el archivo csv y lo nombra datos en DateFrame de Pandas

#Muestra las primeras filas de las columnas
print(datos["V(vout)"].head())
print(datos["V(n003)"].head())

#Se grafica con el eje x lo que esta "limpio" y el eje y lo que esta "sucio"
plt.subplot(1, 1, 1)
plt.plot(datos['V(n003)'], datos['V(vout)']) #Datos que se toman
plt.title('datos') #Titulo del grafico
plt.xlabel('limpio') #Nombre eje x
plt.ylabel('sucio') #Nombre eje y
plt.grid(True)

Este código es utilizado para graficación de comparativa de entrada y salida.

Como paso siguiente se desarrollo el siguiente código en el entorno Arduino IDE para codificar las ecuaciones a utilizar.
```

Grafica la comparativa de datos de entrada y salida

```
#define ENTRADA 34 // Define el pin de entrada analógica que se utilizará para leer datos

// Función para mapear un valor de un rango a otro rango
float mapFloat(float x, float in_min, float in_max, float out_min, float out_max) {
  // Aplica la fórmula de mapeo lineal para transformar el valor x
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

void setup() {
  Serial.begin(9600); // Inicializa la comunicación serie a 9600 baudios
  // Imprime los encabezados para el plotter de datos, separados por tabulaciones
  Serial.println("x\tR1\tR2\tR3\tR4\tR5\tR6\tR7");
}

void loop() {
  float e = 2.71828; // Valor de la constante matemática e (base de los logaritmos naturales)
  float lectura = analogRead(ENTRADA); // Lee el valor analógico del pin definido (ENTRADA)
  // Mapea la lectura del rango [0, 4096] al rango [-0.10, 0.10]
  float x = mapFloat(lectura, 0, 4096, -0.10, 0.10);

  // Se calcula los 6 resultados utilizando una función exponencial para cada uno
  float Resultado1 = (0.4 / (1 + pow(e, 100 * (x + 0.03)))) - 0.2;
  float Resultado2 = (0.6 / (1 + pow(e, 90 * (x + 0.02)))) - 0.3;
  float Resultado3 = (0.8 / (1 + pow(e, 90 * (x + 0.01)))) - 0.4;
  float Resultado4 = (1.0 / (1 + pow(e, 90 * (x + 0.01)))) - 0.5;
  float Resultado5 = (1.2 / (1 + pow(e, 70 * (x + 0.02)))) - 0.6;
  float Resultado6 = (1.4 / (1 + pow(e, 60 * (x - 0.03)))) - 0.7;
  float Resultado7 = (1.6 / (1 + pow(e, 65 * (x - 0.04)))) - 0.8;
  // Calcula el promedio de todos los resultados
  float ResultadoPROMEDIO = (Resultado1 + Resultado2 + Resultado3 + Resultado4 + Resultado5 + Resultado6 + Resultado7) / 7;

  // Se imprime x primero y luego los resultados, todo en una sola línea, separados por tabulaciones
  Serial.print(x, 4); Serial.print('\t'); // Imprime x con 4 decimales
  Serial.print(Resultado1, 6); Serial.print('\t'); // Se imprimen los Resultados del 1 al 7 con 6 decimales
  Serial.print(Resultado2, 6); Serial.print('\t');
  Serial.print(Resultado3, 6); Serial.print('\t');
  Serial.print(Resultado4, 6); Serial.print('\t');
  Serial.print(Resultado5, 6); Serial.print('\t');
  Serial.print(Resultado6, 6); Serial.print('\t');
  Serial.print(Resultado7, 6); Serial.print('\t');
  Serial.println(ResultadoPROMEDIO, 6); // Imprime el promedio y finaliza la línea

  delay(200); // Espera 200 milisegundos antes de la siguiente iteración del bucle
}
```

En Arduino IDE se desarrolló el siguiente código para codificar las ecuaciones

Código que puede exportar datos obtenidos de nuestro sonido en formato WAV

```python
V.1.0.3
import serial #Se importan las librerias
import wave
import time

# Configura el puerto serial
puerto = serial.Serial('COM4', 115200)  # Ajusta a tu puerto
time.sleep(2) # Tiempo de delay

# Parámetros de grabación
duracion = 5  # segundos
frecuencia_muestreo = 22050 # Hertz
num_muestras = duracion * frecuencia_muestreo # Se obtiene el numero de muestras multiplicando la duración y la frecuencia de muestreo

datos = bytearray() # Se define "datos" como la función bytearray

print("Grabando...") # Muestra la palabra mientras se hace lo siguiente

while len(datos) < num_muestras: # Cuando datos es menor a el numero de muestras
    if puerto.in_waiting:
        datos.append(puerto.read()[0])  # Lee 1 byte

print("Grabación terminada.")
puerto.close() # Muestra el mensaje y termina el proceso

# Guardar como .WAV
with wave.open("guitarra9.wav", "wb") as wav_file: # Abre el archivo de audio
    wav_file.setnchannels(1)              # Mono
    wav_file.setsampwidth(1)              # 8 bits
    wav_file.setframerate(frecuencia_muestreo) # Se define la frecuencia
    wav_file.writeframes(datos) # Se escriben los datos brindados

print("Archivo guardado como guitarra.wav") # Muestra como se llama el archivo y que ya quedo modificado
```

```cpp
//V.1.0.0
#define ENTRADA 34  // Pin analógico conectado al circuito offset

void setup() {
  Serial.begin(115200);  // Aumenta la velocidad para evitar cuellos de botella
}

void loop() {
  int lectura = analogRead(ENTRADA);  // 0 a 4095 (12 bits)

  // Convertimos a 8 bits (0-255) centrado
  uint8_t muestra8bit = map(lectura, 0, 4095, 0, 255);
  // Enviar la muestra como 1 byte
  Serial.write(muestra8bit);
  delayMicroseconds(45); // Aproximadamente 22.2 kHz
}
```

Limpieza de datos de entrada y disminución de ruido

```ino
//V.1.0.4

#define ENTRADA 34 //Pin analógico
#define pinDAC1 25 //Pin DAC1-GPIO25 (salida analógica real)

void setup() {
  Serial.begin(115200);
  analogReadResolution(12); //Resolución de 0 a 4095-12bits
}

void loop() {
  int LeerDatos = analogRead(ENTRADA); //Leer valor analógico de 0 a 4095
  int ValorDAC = map(LeerDatos, 0, 4095, 0, 255); //12 y 8 bits para el DAC
  dacWrite(pinDAC1, ValorDAC);//Escribir al DAC (0-255)
  //Imprimir datos
  Serial.print("Valor Leído: ");
  Serial.print(LeerDatos);
  Serial.print("DAC: ");
  Serial.print(ValorDAC);

  delay(10);
}
```

Conversión de datos analógicos a voltajes en la salida DAC con el ESP32

Pruebas

El manejo de pruebas fue mediante la conexión de una guitarra eléctrica a través del jack de entrada del circuito, se alimentó por medio de una fuente de voltaje y el ESP32

La prueba con sonido limpio después de pasar por uno de los códigos y el que modifica la onda de sonido teniendo salida al amplificador de bajo para comprobar el sonido.

Agregar una parte de amplificación para que los datos los leyera de manera optima el ESP32 con variación de ecuaciones

Nos ayudó a tener una comprensión a fondo sobre como trabajan las señales de audio y como se pueden modificar

Problema al incluir pantalla y envio de datos al DAC

Demostramos el funcionamiento de un pedal de fuzz y la señal que genera el sonido con el fuzz ya implementado .

Conclusiones y Problemas

THANK YOU

Equipo NANA