Trabajo Práctico Integrador

Ingeniería aplicada a la comunicación de datos

Objetivo

El propósito de este trabajo es que los estudiantes desarrollen una aplicación web, de escritorio o móvil, que aborde problemáticas relacionadas con la transmisión de datos, digitalización de señales y medios de. El software deberá representar, simular o resolver aspectos teóricos y/o prácticos de la asignatura.

Etapa 1 - Diseño y planificación (30%)

Entrega: 16/5 (S32) - 16/5 (S31/S33)

- Elegir una de las propuestas que se numeran más abajo para desarrollar.
- Definición de la arquitectura de la aplicación.
- Herramientas y tecnologías a utili zar.
- Mockups o bocetos de la interfaz.

Producto de entrega:

- ✓ Documento con la propuesta elegida, arquitectura, breve descripción de las tecnologías a utilizar y, por último, un boceto de la interfaz.

Criterios de aceptación:

- ✓ Respeto de la fecha de entrega
- ✓ El documento debe ser en formato editable (.docx.)
- ✓ El Mockup puede ser un link a un Figma, Zeplin u otro.
- ✓ El .docx tendrá máximo dos carillas.

Etapa 2 - Desarrollo (40%)

Entrega: 16/6 (S32) - 20/6 (S31/S33)

- Desarrollo del software con funcionalidades básicas operativas
- Prototipo funcional

Productos de entrega:

- ✓ Código fuente de lo desarrollado
- ✓ Video con una demo de las funcionalidades desarrolladas
- ✓ Ejecutable: Carpeta del deploy, link al host, ejecutable o apk

Criterios de aceptación:

- ✓ Respeto de la fecha de entrega
- ✓ El código fuente puede ser un link al repo público de GitHub o el repo comprimido

- ✓ El video no debe durar más de 6 minutos
- ✓ El video debe reflejar claramente cómo se usan las funcionalidades desarrolladas
- ✓ Instrumento de ejecución para que el desarrollo pueda ser testeado por la cátedra.

Etapa 3 - Defensa del Proyecto (30%)

Entrega: 30/6 (S32) - 4/7 (S31/S33)

- Demo de la aplicación en modalidad coloquio.
- Justificación teórica de las decisiones tomadas.
- Presentación estilo PowerPoint

Productos de entregas:

- ✓ Presentación con diapositivas estilo PowerPoint.
- ✓ Ejecutable

Criterios de aceptación:

- ✓ La presentación debe contar con un slide que describa al grupo
- ✓ La presentación debe dar cuenta de cómo funciona la aplicación
- ✓ La presentación debe justificar desde el punto de vista teórico la funcionalidad que forma parte de la demo
- ✓ La presentación no debe superar 8 slides
- ✓ La demo dura como máximo 15 minutos
- ✓ La cátedra debe poder ejecutar la aplicación final

Propuestas de desarrollo

A continuación, se enumeran una lista de propuestas para que cada grupo elija cuál desarrollar. Tener en cuenta que las funcionalidades marcadas en negrita son de desarrollo obligatorio debiendo cada grupo, además, desarrollar al menos alguna otra funcionalidad.

1. Simulador de Conversión de Señales (ADC - Digitalización de Señales)

Un software interactivo que permita ingresar una señal analógica (por ejemplo, una onda sinusoidal) y visualizar su digitalización en tiempo real mediante distintos métodos de muestreo y cuantización.

Funcionalidades:

- **Generación de señales analógicas** (sinusoides, cuadradas, ruido, etc.).
- Simulación del proceso de muestreo con diferentes tasas de muestreo (ej: 8 kHz, 44.1 kHz).
- Cuantización con distintos niveles de bits (ej: 8 bits, 16 bits, 24 bits).
- **Visualización comparativa de la señal original vs. la digitalizada.**
- Aplicación del Teorema de Nyquist y demostración de aliasing cuando la tasa de muestreo es insuficiente.

Tecnologías sugeridas:

- Python con Matplotlib y NumPy (para simulaciones).
- WebApp con React + D3.js (para visualización interactiva).

2. Conversor de Audio Analógico a Digital

Una aplicación que permita cargar o grabar audio en formato analógico (o simulado) y convertirlo a distintas resoluciones digitales, mostrando cómo varían la calidad y el tamaño del archivo según la configuración de muestreo y cuantización.

Funcionalidades:

- **Grabación de audio en tiempo real con un micrófono.**
- Conversión del audio a diferentes tasas de muestreo (ej: 8 kHz, 16 kHz, 44.1 kHz, 96 kHz).
- Cuantización con distintas profundidades de bits (8, 16, 24 bits).
- Comparación de espectros de frecuencia antes y después de la conversión.
- **Exportación del audio digitalizado en formatos como WAV o MP3.**

Tecnologías sugeridas:

- Python con pydub y librosa (para procesamiento de audio).
- Aplicación web con WebRTC + Web Audio API (para grabación y conversión en el navegador).

3. Digitalización de Imágenes

Un conversor que tome imágenes en formato analógico (simuladas) y las convierta en datos digitales mediante muestreo y cuantización de color.

Funcionalidades:

- **Carga de imágenes en alta resolución.**
- Aplicación de muestreo con distintos niveles de resolución (ej: 100x100, 500x500, 1000x1000).
- Reducción de la profundidad de bits por canal (ej: 1 bit, 8 bits, 24 bits).
- Comparación de la imagen original vs. la digitalizada.
- Aplicación de compresión para reducir el tamaño del archivo.

Tecnologías sugeridas:

- Python con OpenCV y PIL (para manipulación de imágenes).
- Aplicación web con React + Canvas API.

4. Transmisión Digital de Señales y Compresión

Una aplicación que permita transmitir una señal digitalizada a través de distintos medios (ej: TCP/IP o serial) y analizar cómo la calidad se ve afectada por la compresión y la pérdida de datos.

Funcionalidades:

- **Conversión de señales analógicas en paquetes digitales.**
- Aplicación de distintos esquemas de compresión (ej: PCM, ADPCM, MP3).
- Simulación de errores en la transmisión y cómo afecta la reconstrucción de la señal.
- **Visualización de la reconstrucción de la señal en el receptor.**

Tecnologías sugeridas:

- Python con sockets y scipy.signal (para simulaciones).
- Aplicación web con WebSockets y Web Audio API.

5. Codificación de datos

Una aplicación interactiva que permita comprimir y descomprimir mensajes o archivos de texto utilizando los algoritmos de Huffman y Shannon-Fano, comparando sus resultados en términos de eficiencia y tamaño comprimido. La aplicación debe mostrar gráficamente el proceso de codificación y los resultados obtenidos.

Funcionalidades:

- **Carga de texto (por entrada directa o archivo .txt)**
- Cálculo de frecuencia de símbolos
- Construcción de árbol y tabla de códigos (Huffman y Shannon-Fano)

- Codificación y decodificación del mensaje
- **Comparación de resultados: tasa de compresión, eficiencia, longitud promedio de código**
- Visualización de tabla de símbolos y sus códigos
- Gráfico de barras con frecuencias y longitudes de códigos
- Compresión de mapas de bits de imágenes en blanco y negro

Tecnologías sugeridas:

- Python: con Tkinter, PySimpleGUI o consola + matplotlib para gráficas
- JavaScript (alternativa web): con HTML/CSS y Chart.js o D3.js para visualizaciones
- Estructuras recomendadas: árboles binarios, colas de prioridad (heapq en Python).