

Apéndice D. Web App

Como parte de la solución de monitoreo remoto, se desarrolló una aplicación web (Web App) utilizando Google Apps Script. Esta aplicación permite visualizar los datos espectrales almacenados en una hoja de cálculo de Google Sheets, presentándose en gráficos interactivos generados con Plotly.

URL	del	Web	Service:
https://script.google.com/macros/s/AKfycbXPjACWJlfD5BwK8c23FUJslQIFXA9uWYpTh_oWFXU0dex6oM4Z-OhnGj9fMc1s3e/exec			

URL	del	Código	Fuente:
https://script.google.com/d/1qvDR2ZIHIA7EZrntiSMgkFizprUkyYyolzeBQ2KzOszlwmwR7Y2KYSuZ/edit?usp=sharing			

El propósito de esta aplicación web es proporcionar una visualización dinámica y personalizable de los espectros capturados. Para ello, permite al usuario seleccionar diversos parámetros de configuración, tales como: la hoja dentro de Google Sheets donde se almacenan los datos, la unidad de frecuencia (Hz, kHz, MHz, GHz) y la escala de visualización, que puede expresarse en decibelios (dB) o en magnitud al cuadrado (mag^2).

Figura 1

Datos de la hoja de cálculo.

```

2
3 // 1) Servir el HTML
4 function doGet(e) {
5     return HtmlService
6         .createHtmlOutputFromFile('Index')
7         .setTitle('Gráfica del Espectro')
8         .setXFrameOptionsMode(HtmlService.XFrameOptionsMode.ALLOWALL);
9 }
10 // 3) Devuelve lista de hojas para el <select>
11 function obtenerNombresHojas() {
12     var ss = SpreadsheetApp.openById("12wzyVjcI18qGjRYBwKNHRmxd0t0NYiDhGSMLt906STjg");
13     return {
14         nombresHojas: ss.getSheets().map(function(sh){ return sh.getName(); })
15     };
16 }
17 // 4) Lee siempre la fila 2 de la hoja seleccionada
18 function obtenerDatosEspectro(M, desplazamientoTiempo, hojaActual) {
19     var ss = SpreadsheetApp.openById("12wzyVjcI18qGjRYBwKNHRmxd0t0NYiDhGSMLt906STjg");
20     var sheet = ss.getSheetByName(hojaActual);
21
22     // Fila 2: [Fecha, lat, lon, Azimut, Elev, Alt, Desc, finicial, fpaso, N, Datos]
23     var fila = sheet.getRange(2, 1, 1, sheet.getLastColumn()).getValues()[0];
24     var finicial = fila[7];
25     var fpaso = fila[8];
26     var N = fila[9];
27     var datosStr = fila[10];
28     var datosArr = datosStr.split(',').map(Number);
29
30     return {
31         listaDatos: [{
32             N: N,
33             datos: datosArr,
34             frecuenciaInicial: finicial,
35             pasoFrecuencia: fpaso
36         }]
37     };
38 }

```

Nota. En esta imagen se va a leer los últimos datos que registró la hoja de cálculo.

La función `doGet(e)` es entregar a la página el archivo `Index.html`, al cargar la página, el script cliente invoca mediante `google.script.run` la función `obtenerNombresHojas()`. En el servidor, ésta abre la hoja de cálculo usando su ID, extrae los nombres de todas sus pestañas y los devuelve al cliente. Luego, el cliente recibe esa lista y la inserta como opciones en un select, permitiendo al usuario elegir qué hoja de datos desea visualizar.

obtenerDatosEspectro vuelve a abrir el mismo spreadsheet y selecciona la pestaña cuyo nombre recibió en hojaActual. Con `sheet.getRange(2, 1, 1, sheet.getLastColumn()).getValues()[0]` lee la fila 2 completa, desde la primera hasta la última columna, devolviendo un array llamado `fila`. Según la convención de tu hoja, los índices 7, 8 y 9 de ese array contienen la frecuencia inicial (`fInicial`), el paso entre muestras (`fPaso`) y el número de muestras (`N`), respectivamente. El índice 10 contiene una cadena CSV con los valores del espectro; separarla con `split(',')` y convertir cada fragmento a número con `map(Number)` produce `datosArr`, un array de magnitudes. Finalmente, la función retorna un objeto cuyo atributo `listaDatos` incluye un único elemento con las propiedades `N`, `datos`, `frecuenciaInicial` y `pasoFrecuencia`, que el cliente usará para construir los ejes X e Y y trazar la curva con Plotly.

Figura 2

Código HTML de la WebApp parte A.

```

1  <!-- Archivo: Index.html -->
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <title>Gráfica del Espectro</title>
6      <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
7      <style>
8          body, html {
9              height: 100%;
10             width: 100%;
11             margin: 0;
12             display: flex;
13             justify-content: center;
14             align-items: center;
15             flex-direction: column;
16             background: #000;
17             color: #fff;
18         }
19         #graficaEspectro { width: 100%; height: 80vh; }
20         #controles { margin-bottom: 10px; }
21         select { margin-right: 10px; }
22         #error { color: #f66; }
23     </style>
24     <script>
25         let divGrafica;
26         let unidadActual = 'MHz';
27         let escalaActual = 'dB';
28         let hojaActual;
29         const factoresUnidad = { 'Hz':1, 'kHz':1e3, 'MHz':1e6, 'GHz':1e9 };
30
31         // 1) Poblar <select> de hojas
32         function cargarNombresHojas(res) {
33             if (res.error) {
34                 return document.getElementById('error').innerText = res.error;
35             }

```

Nota. En esta imagen se escribe como va a ir organizado la la pagina web(WebApp), el tamaño , las variables y el color que va tomar dicha página web.

Las líneas <!DOCTYPE html> y la etiqueta <html>, indicando que es una página HTML5. En el <head> se fija el título de la pestaña cómo “Gráfica del Espectro” mediante <title>. Justo después se carga la biblioteca de Plotly desde su CDN, lo que permite dibujar gráficos interactivos en el cliente. A continuación, el bloque <style> define reglas CSS para toda la página: tanto <html> como <body> ocupan el 100 % de ancho y alto, se eliminan márgenes y se usa un diseño flex en columna, centrado horizontal y verticalmente, con fondo negro y texto blanco. El contenedor #graficaEspectro recibe 80 vh(vh es viewport height, un hv significa que va a tomar el 1% de la

altura de la ventana del navegador que se está usando) de alto y ancho completo; #controles añade un pequeño margen inferior; los <select> tienen espacio entre sí, y #error se pinta en rojo para mostrar mensajes de fallo.

Para el script, divGrafica, unidadActual, escalaActual, hojaActual son datos ajustables que se visualizan en la gráfica de la webApp.

Figura 3

Código HTML de la WebApp parte B.

```

36   const sel = document.getElementById('seleccionHoja');
37   res.nombresHojas.forEach(h => {
38     const o = document.createElement('option');
39     o.value = o.text = h;
40     sel.appendChild(o);
41   });
42   hojaActual = res.nombresHojas[0];
43   obtenerDatosEspectro();
44 }
45 function obtenerNombresHojas() {
46   google.script.run.withSuccessHandler(cargarNombresHojas)
47   | | | | | | | | .obtenerNombresHojas();
48 }
49
50 // 2) Solicitar datos y graficar
51 function actualizarHoja() {
52   hojaActual = this.value;
53   obtenerDatosEspectro();
54 }
55 function actualizarUnidad() {
56   unidadActual = this.value;
57   obtenerDatosEspectro();
58 }
59 function actualizarEscala() {
60   escalaActual = this.value;
61   obtenerDatosEspectro();
62 }
63 function procesarDatos(res) {
64   if (res.error) {
65     return document.getElementById('error').innerText = res.error;
66   }
67   const { N, datos, frecuenciaInicial, pasoFrecuencia } = res.listaDatos[0];
68   const factor = factoresUnidad[unidadActual];
69   const x = Array.from({length:N}, (_,i)=> (frecuenciaInicial + pasoFrecuencia*i)/factor);
70   const y = escalaActual==='dB'
71   ? datos.map(v=>10*Math.log10(v))

```

Nota. En esta imagen pertenece a la continuación del código HTML de la WebApp extrayendo los valores de la hoja de cálculo.

Cuando la petición de datos al servidor regresa, `procesarDatos(res)` recibe un objeto con `res.listaDatos[0]`, del que extrae cuatro valores: `N`, `datos`, `frecuenciaInicial` y `pasoFrecuencia`. Calcula el factor de unidad correspondiente (`factoresUnidad[unidadActual]`) y arma dos arrays:

El eje X, con `Array.from({length:N}, (_,i) => (frecuenciaInicial + pasoFrecuencia*i)/factor)`.

El eje Y, que si `escalaActual==='dB'` convierte cada valor con $10 \cdot \log_{10}(v)$, o deja los datos crudos en mag^2 .

Finalmente llama a `Plotly.newPlot(divGrafica, ...)` para dibujar un gráfico de líneas, con el contenedor de fondo negro, cuadrículas grises y un `hovertemplate` que muestra la frecuencia y magnitud al pasar el ratón.

La función `obtenerDatosEspectro()` en el cliente lanza:
`google.script.run,withSuccessHandler(procesarDatos),obtenerDatosEspectro(1, 0, hojaActual)`.

Esto llama a la función en Code.gs, que abre la hoja de cálculo, lee la segunda fila y devuelve los parámetros empaquetados en JSON.

Inicialización en `window.onload`

Cuando se carga la página, `window.onload` hace varias cosas:

Guarda la referencia al `<div id="graficaEspectro">` en `divGrafica`.

Añade `addEventListener('change', ...)` a cada `<select>` (unidad, escala, hoja) para que reaccionen a cambios.

Llama a `obtenerNombresHojas()` para poblar el selector de hojas.

Inicia un `setInterval(obtenerDatosEspectro, 1000)`, que cada segundo vuelve a pedir datos al servidor y redibuja la gráfica de forma automática.

Figura 4

Código HTML de la WebApp parte D.

```

72     : datos;
73     Plotly.newPlot(divGrafica, [{
74         x, y, type:'scatter', mode:'lines',
75         hovertemplate:'Freq: %{x:.2f} ${unidadActual}<br>Mag: %{y:.2f}${escalaActual}<extra></extra>'
76     }], {
77         xaxis:{ title:'Frecuencia (${unidadActual})', gridcolor:'#444' },
78         yaxis:{ title:'Escala (${escalaActual})', gridcolor:'#444' },
79         plot_bgcolor:'#000', paper_bgcolor:'#000'
80     });
81 }
82 function obtenerDatosEspectro() {
83     google.script.run
84         .withSuccessHandler(procesarDatos)
85         .obtenerDatosEspectro(1, 0, hojaActual);
86 }
87
88 window.onload = () => {
89     divGrafica = document.getElementById('graficaEspectro');
90     document.getElementById('seleccionUnidad')
91         .addEventListener('change', actualizarUnidad);
92     document.getElementById('seleccionEscala')
93         .addEventListener('change', actualizarEscala);
94     document.getElementById('seleccionHoja')
95         .addEventListener('change', actualizarHoja);
96     obtenerNombresHojas();
97     // Refrescar cada segundo
98     setInterval(obtenerDatosEspectro, 1000);
99 };
100 </script>
101 </head>
102 <body>
103     <div id="controles">
104         <label>Unidades:
105         <select id="seleccionUnidad">
106             <option>Hz</option><option>kHz</option>
107             <option selected>MHz</option><option>GHz</option>

```

Nota. En esta imagen se va a leer los datos actualizándolos 1000ms actualizando constantemente los valores recientes.

Figura 5

Código HTML de la WebApp parte E.

```

108     </select>
109     </label>
110     <label>Escala:
111     <select id="seleccionEscala">
112         <option>mag2</option><option selected>dB</option>
113     </select>
114     </label>
115     <label>Hoja:
116     <select id="seleccionHoja"></select>
117     </label>
118     </div>
119     <div id="graficaEspectro"></div>
120     <p id="error"></p>
121 </body>
122 </html>

```

Nota. En esta imagen corresponde a las opciones (unidades, Escalam Hoja) en la interfaz gráfica.

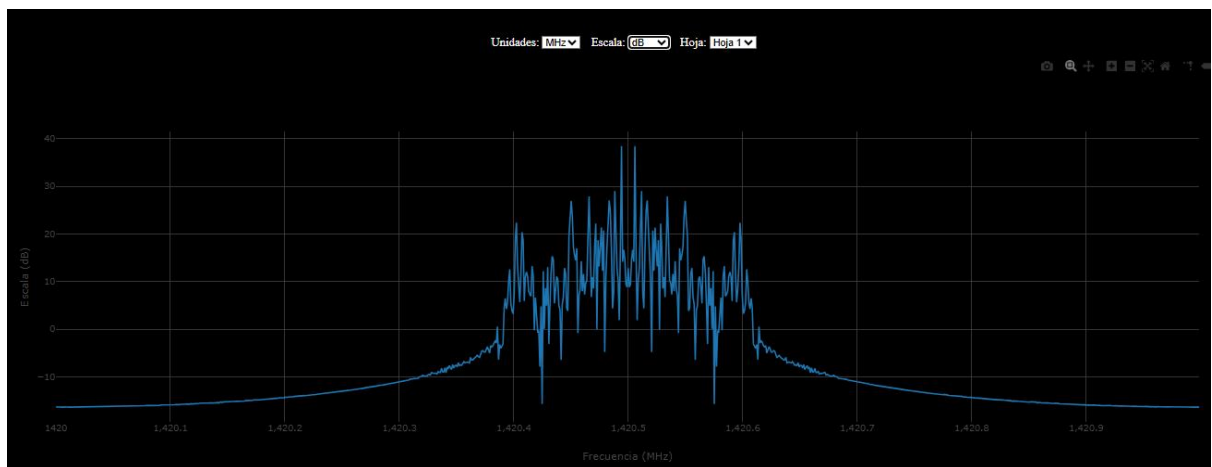
Dentro del <body> hay dos partes principales:

Un <div id="controles"> con tres etiquetas y sus <select> asociados: uno para unidades (Hz, kHz, MHz, GHz), otro para escala (mag² o dB) y un tercero para la hoja de cálculo.

Un <div id="graficaEspectro"> vacío, destino del gráfico, y un <p id="error"> donde se muestran posibles mensajes de error.

Figura 6

Gráfica en la WebApp vista a partir del último dato registrado.



Nota. En la imagen vemos la gráfica creada después de leer el último dato de la hoja de cálculo, en la WebApp podemos cambiar las unidades, la escala y la hoja (si hay más de una) según se requiera y se acomode a lo que el usuario necesite.

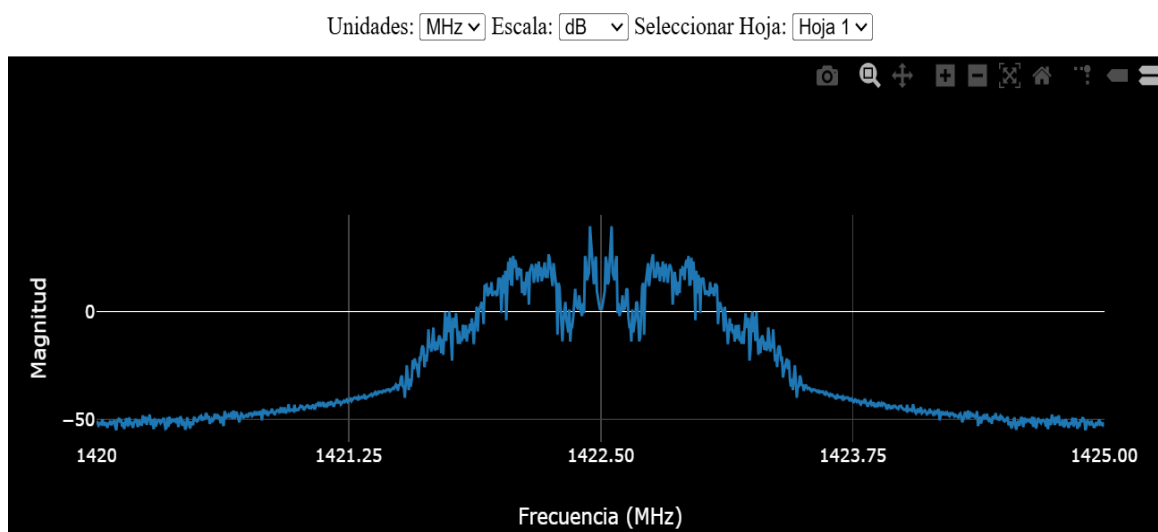
La **Figura 6** es el resultado de la página web, acá podemos ver constantemente como varía la gráfica a partir de los cambios que tiene la hoja de cálculo, esta imagen se puede comparar a la gráfica que genera GNU Radio.

La WebApp no se limita a mostrar datos; también permite al usuario interactuar de forma sencilla y dinámica con la información espectral. Gracias a sus controles intuitivos, es posible elegir la unidad de frecuencia y la escala más adecuada para cada análisis, lo que facilita la

interpretación del espectro en distintos contextos; navegar entre diferentes hojas de cálculo para acceder a varias sesiones o conjuntos de datos almacenados, de modo que se puedan comparar resultados o revisar históricos recientes; e incluso observar actualizaciones automáticas en tiempo real, ofreciendo una experiencia muy cercana a la monitorización continua del espectro.

Figura 7

Visualización en la WebApp.



Nota. La visualización de la gráfica se debe al último dato registrado en la hoja de cálculo. Fuente: Elaboración propia.

